INSTITUTE OF INFORMATION
AND COMMUNICATION TECHNOLOGIES
BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **26**, No 2
Sofia • 2026 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2026-0019

Robust and Efficient Detection of Large Language Model-Generated Text via the Multi-Feature Accurate Detection (MFAD) Approach

Doaa Mostafa, Sally Ismail, Mostafa Aref

Faculty of Computer and Information Science, Ainshams University, Cairo, Egypt
E-mails: doaa.ahmed74@yahoo.com (*corresponding author*) *sallysaad@cis.asu.edu.eg Mostafa.aref@cis.asu.edu.eg*

Abstract**:** *Recently, Large Language Models* (*LLMs*) *have been able to generate text that closely resembles human writing, raising concerns about academic misuse and misinformation. Existing detection approaches often depend on a single type of feature, require direct access to the underlying models, and are sensitive to variations in text length and paraphrasing. To address these issues, this paper proposes a Multi-Feature Accurate Detection* (*MFAD*) *approach that integrates handcrafted statistical and syntactic features with deep semantic features based on Global Vectors for Word Representation* (*GloVe*) *embeddings, Convolutional Neural Networks* (*CNNs*)*, and Bidirectional Long Short-Term Memory* (*BiLSTM*)*. The results of the experiments on Human ChatGPT Comparison Corpus* (*HC3*) *demonstrate that MFAD achieves 98% accuracy, 96.5% precision, 97.5% recall, 97% F1-score, with a minimum False Positive Rate* (*FPR*) *of 0.01 across multiple domains. Additionally, MFAD demonstrates strong cross-model generalizability across LLMs such as GPT-4, Gemini, and Claude-4, and exhibits resilience to text length variations and paraphrasing.*

Keywords: *Large Language Models* (*LLMs*)*, AI-generated text, Word embedding, Feature extraction, Feature-based detection.*

1. Introduction

Advances in Natural Language Processing (NLP) have notably increased the quality and diversity of text produced by Large Language Models (LLMs). Advanced models like GPT-4 [1], Claude-4 [2], and Gemini [3] produce text that closely resembles human writing in a variety of applications, such as coding, writing emails, writing scientific articles, writing essays, asking and answering questions, etc. However, the ability of LLMs to produce human-like text fairly quickly and in potentially harmful ways has generated concern regarding their use in relation to disinformation contexts [5], academic dishonesty [5, 6, 9], fake news [4, 10], and fake product reviews [4-8].

The four primary approaches for identifying LLM-generated text are feature-based, watermarking, neural-based, and human-assisted [11]. Watermarking-based approaches embed hidden signals within the generated text, but require access to the entire model, which is impractical for advanced models. Feature-based use aspects, such as syntax, grammar, and various linguistic features for differentiating between human-written and LLM-generated text, yet these approaches struggle with advanced LLMs [12]. The watermarking approach also struggles with small modifications like inserting spaces or using a word's synonym, which can affect the accuracy of watermark-detection [11, 12]. Neural-based approaches employ pretrained transformers, such as BERT [13] and RoBERTa [14], to identify LLM-generated text. While neural-based approaches produce reliable results, they require high computing resources and large training datasets [8, 9]. Finally, there are human-assisted approaches that rely on human cognition and analytical capabilities for the detection of LLM-generated text. The human ability to distinguish generated text across domains becomes even more difficult as modern LLMs generate text that is increasingly indistinguishable from human writing [11].

This paper addresses the challenges of existing detection approaches by proposing an MFAD approach. This approach consists of six stages: preprocessing, handcrafted features extraction, text representation, semantic feature extraction, feature fusion, and classification. Preprocessing is the initial stage, and this is done by cleaning and normalizing input text. Syntactic and statistical handcrafted features are extracted to capture structural properties for the next stage (text representation). Following the handcrafted syntactic and statistical feature extraction, the GloVe [15] model is used for text representation as it creates fixed-length dense vectors representing tokens generated from word co-occurrence statistics, while encoding syntactic and semantic relationships. Learning semantic features is a subsequent stage of using a CNN [16] to capture local structure at the phrase-level, followed by a BiLSTM to learn dependencies in context and in both forward and backward directions unconditionally. The output, handcrafted, and semantic features are combined to complete the workflow to obtain a classification. Unlike traditional feature-based approaches that are often obsolete with evolving LLMs, the MFAD approach's use of a mixture of handcrafted and deep semantic features allows MFAD to remain robust and reliable for detecting LLM-generated text, even when applied to more advanced LLMs, given that it identifies many explicit stylistic signposts and relies on more implicit contextual encoding mechanisms to capture inherent meaning.

This paper is structured as follows. Related work in Section 2 discusses recent studies on LLM-generated text detection. Section 3 details the design of the proposed approach. Section 4 presents the dataset description, performance measures, and experiment results in different evaluation settings. The discussion of the results is provided in Section 5. Finally, Section 6 concludes the research and highlights future directions.

2. Related work

In this section, a more thorough and perceptive analysis of the recent research focused on detecting LLM-generated text is presented. DetectGPT is presented in [17], which uses log probabilities from the target model and random context perturbations from a pre-trained language model (T5 [18], paraphraser). The log probability of the original text from the target model is compared to the log probability of the text with random perturbations: if the original text has a higher probability, it is more likely to be LLM-generated; if the log probability is lower, it is more likely human-written. The main downside of this approach is the high computational cost from repeated randomness and re-evaluation [11]. Fast-DetectGPT [19] is an efficient version of DetectGPT that reduces the computational cost of repeated perturbations. Instead of generating many perturbed versions, Fast-DetectGPT uses a single forward pass through a masked language model (MLM) to approximate the effect of perturbations, speeding up detection while maintaining high accuracy for detecting LLM-generated text. However, the issue remains that Fast-DetectGPT's response may not be as accurate as the original DetectGPT due to the continued reliance on approximation methods.

ConDa [20] is a contrastive domain adaptation framework designed to identify LLM-generated text. It is built upon a pre-trained RoBERTa model and trained on labeled datasets, with a primary focus on detection within the news domain. ConDa performs well in black-box settings and remains robust across different LLMs by exploiting subtle generation artifacts, such as consistent stylistic patterns and statistical irregularities introduced during text generation. However, ConDa relies on multiple models, which increases computational complexity and deployment cost [21].

A Regularized Deep Neural Network (RDNN) is introduced in [22]. RDNN combines a CNN, a BiLSTM encoder, and a distributed highway network. The CNN first extracts lexical stylometric features, which are then processed by the BiLSTM and converted into a syntactic feature-vector representation. The feature vectors are then put through the distributed highway network, which applies regularization to reduce generalization errors. The regularized features are then fed into a bidirectional decoder to extract the author's writing style. Finally, a fully connected layer with a SoftMax function performs the classification (LLM-generated text or human-written) [21]. AuthentiGPT [23] is a detection approach that combines linguistic features, semantic embeddings, and watermark analysis to recognize text produced by large language models. AuthentiGPT uses a fine-tuned discriminator that has been trained on large datasets of both human-written and AI-generated content. However, this approach has higher computational costs and requires large labeled datasets.

The Watermarking framework for LLMs (WLLM) [24] is a watermark-based approach for detecting LLM-generated text by embedding hidden signals during the text generation process. WLLM modifies token sampling by using a hash function that divides the vocabulary into two lists, green and red, to bias the model toward selecting tokens from the green list. WLLM creates an imperceptible watermark that can later be identified by checking whether the token distribution matches the

expected bias. Its main advantages are that it is lightweight and does not require modification of the model architecture [11]. The primary disadvantage of WLLM is that it is susceptible to paraphrasing, which can readily eliminate or diminish the effectiveness of the watermark [21].

The Selective Watermarking via Entropy Threshold (SWEET) approach proposes an enhanced approach for watermarking text produced with a Large Language Model (LLM) [25]. Rather than watermarking every token, as is typical for watermarking text, the SWEET method targets only tokens that have high entropy in their predicted probability distribution and are therefore uncertain. By utilizing a selective watermarking approach, the method increases text naturalness and fluency, decreases the level of detection compared to prior watermarking approaches, and decreases linguistic disturbance. The SWEET method is more robust than text watermarking methods in a traditional setting; however, the SWEET method is still vulnerable to excessive paraphrasing of content [11]. The Giant Language Model Test Room (GLTR) is a detection approach that identifies LLM-generated text by analyzing the statistical patterns indicative of a word's probabilities [26]. The main premise of GLTR is that LLM-generated text will rely on using more predictable, high-probability words than human-written text. GLTR depicts these patterns by coloring tokens based on output rank from a reference language model, by coloring the most likely words differently from the less likely words [21].

3. Proposed methodology

The proposed approach consists of six stages: preprocessing, handcrafted feature extraction, text representation, semantic feature extraction, feature fusion, and text classification. Preprocessing is performed first to clean the text by removing irrelevant or extraneous content.

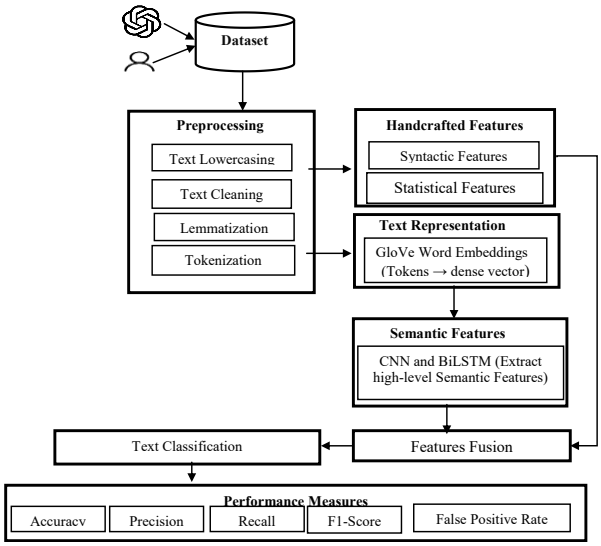

Fig. 1. The design of the proposed approach (MFAD) for detecting LLM-generated text

After preprocessing, handcrafted statistical and syntactic features are extracted. Then GloVe is used for text representation, converting each token into a fixed-length dense vector. Semantic features are extracted using a CNN to capture local semantic patterns. These semantic features are subsequently combined with the handcrafted features, as well as contextual representations obtained from BiLSTM. The combined feature set is passed through a fully connected layer to perform text classification. The design of the proposed approach is illustrated in Fig. 1.

3.1. Text preprocessing

Text preprocessing is essential for removing unnecessary elements such as URLs, emails, symbols, and other non-informative elements [27]. The goal of the preprocessing is to obtain salient information from text. The preprocessing steps comprise text lowercasing and removing hashtags, URLs, emails, numbers, and extra spaces, applying word lemmatization to reduce words to their base form, encoding emojis into textual descriptions, and tokenizing the text into individual tokens.

3.2. Handcrafted feature extraction

Handcrafted statistical and syntactic features are extracted to provide quantifiable analysis of linguistic patterns. The structural characteristics of writing are formalized by using explicitly defined measures of sentence structure, lexical composition, and syntactic patterns, as listed in Table 1.

Table 1. The handcrafted features

Type	Handcrafted features
Statistical	The total word count of the text
	The total sentence count of the text
	The number of distinct words in the text
	The Type-Token Ratio (TTR) is a metric used to measure vocabulary diversity
	The total stop-word count of the text
	The total number of punctuation marks is present in the text
	The total discourse marker counts in the text
	The total number of spelling errors in the text
	The total number of grammar errors in the text
	Readability Scores (Flesch Reading Ease) [30]: reflects how easy the text is to read; higher scores indicate easier readability
	The total syllable count in the text
	Average-Sentence Length is the mean number of words per sentence, calculated as the total word count divided by the total sentence count [30]
	The average word length in characters, computed as the number of characters divided by the total number of words
Syntactic	Part-Of-Speech (POS) tag distribution, representing the frequencies of NN (noun, singular), NNS (noun, plural), VB (verb, base form), VBD (verb, past tense), JJ (adjective), RB (adverb), PRP (personal pronoun), IN (preposition), VBG (verb, present participle), CC (coordinating conjunction), and DT (determiner) [30]
	Sentence Complexity is the Depth of the syntactic tree, such as the average of the dependency tree, the depth, the maximum of dependency tree depth, and the number of subordinate clauses
	The top bigram/trigram frequency is defined as the highest occurrence of a two- or three-word sequence within the text [30]

These include readability metrics calculated by the Flesch Reading Ease (FRE) score, Part-Of-Speech (POS) tag frequency distributions, and n-gram statistics. Lexical diversity (vocabulary richness) is measured by the

Type-Token Ratio (TTR), which is calculated as the number of unique words (types) divided by the total number of words (tokens) in the text. This metric quantifies the variability of word usage and is commonly used to assess how diverse or repetitive a text is stylistically. Additional statistical features that capture textual complexity and stylistic consistency include word and sentence counts, stop-word frequency, punctuation density, syllable counts, average sentence length, and self-detected spelling and grammar errors. Moreover, POS tag frequencies and the most frequent bigrams/trigrams are used to represent recurring syntactic and local word sequence patterns. These patterns differ between human-written and LLM-authored text, as human-written text is generally expected to have more varied syntactic patterns, while LLM-authored text is expected to have more regularized POS tag distributions and more repetitive n-gram patterns due to probabilistic text generation [28, 29]. All handcrafted features and their descriptions are summarized in Table 1 [30].

3.3. Text representation

Splitting the text into individual tokens is the last step in text preprocessing. These tokens are then passed to GloVe to derive global statistical relationships and local contextual meaning of words within a dataset. GloVe represents each word as a dense vector in a finite-dimensional continuous space (50, 100, 200, 300), where semantically similar words are located closer together [15]. These representations are obtained from statistics of word co-occurrence across a large corpus, which allows GloVe to learn about semantic relationships among the words. GloVe embeddings are context independent because they assign a fixed vector to each word, although they explicitly relate the vectors to a global co-occurrence pattern, which still allows vector-space proximity, resulting in an indication of semantic proximity to the actual word meaning; this is one of the properties that differentiates GloVe word embeddings from context-dependent embeddings that are produced by models like BERT. These numerical vectors are then used as input to the CNN to produce high-level semantic features.

3.4. Semantic feature extraction

Local and contextual semantic information is captured using a hybrid CNN–BiLSTM architecture. Given an input sequence of word embeddings $X = [x_1, x_2, x_3, \dots, x_T]$. A one-dimensional Convolutional Neural Network (CNN) is first applied to capture local semantic patterns. To accommodate texts of varying lengths, sequences are padded or truncated to a fixed length of 300 tokens, ensuring uniform input size for the CNN. We use 128 convolutional filters of kernel size 5, a stride of 1, and "valid" padding (no zero-padding). Each convolutional filter produces a feature map representing local *n*-gram semantic patterns. Each convolutional filter produces a feature map representing local *n*-gram semantic patterns:

(1) $$c_i = f(W \cdot X_{i:i+k-1} + b),$$

where W and b are the filter weights and bias, k is the kernel size, and f is the ReLU activation function. A max-pooling (pool size of 2) operation over time is applied to each feature map to retain the most salient activation:

(2) $$\hat{c}^{(j)} = \max\left(\mathbf{c}^{(j)}\right).$$

This produces robust local semantic features while reducing dimensionality. The pooled convolutional outputs are then reorganized into a sequential representation:

(3) $$C = [c_1, c_2, \ldots, c_{T-k+1}]^{\top}.$$

The sequence C is subsequently fed into a bidirectional LSTM (BiLSTM) with a hidden state size of 128 for both forward and backward directions. At each time step t, the forward and backward hidden states are computed as:

(4) Forward LSTM $$\overrightarrow{\mathrm{h}}_t = \mathrm{LSTM}_{\mathrm{fwd}}(\mathbf{c}_t, \overrightarrow{\mathrm{h}}_{t-1}),$$

(5) BackwardLSTM $$\overleftarrow{\mathrm{h}}_t = \mathrm{LSTM}_{\mathrm{bwd}}(\mathbf{c}_t, \overleftarrow{\mathrm{h}}_{t+1}),$$

and concatenated to form the context-sensitive representation:

(6) $$\mathrm{h}_t = [\,\overrightarrow{\mathrm{h}}_t\,;\,\overleftarrow{\mathrm{h}}_t\,],$$

the final BiLSTM output sequence is

(7) $$H = [h_1, h_2, \ldots, h_{T-k+1}]^{\top}.$$

By combining convolutional feature extraction with bidirectional sequential modeling, the CNN-BiLSTM captures hierarchical semantic information, where local phrase-level semantics from the CNN are integrated with long-range contextual dependencies captured by the BiLSTM.

The proposed CNN-BiLSTM representation generator is trained jointly with the classification objective in a unified end-to-end framework, in contrast to methods that pre-train a feature extractor independently. The embedding layer is initialized with pre-trained GloVe vectors and kept fixed to provide a stable semantic prior, while the convolutional and BiLSTM layers are learned directly from the labeled dataset through backpropagation. This enables the network to derive task-specific representations that are suited to the goal of human-versus-machine discrimination. Prior to classification, the learned deep representation is fused with handcrafted stylometric and statistical features that are deterministically extracted from the text. Binary cross-entropy loss is used to optimize the entire architecture at once, allowing coordinated learning from stylistic and semantic cues.

3.5. Features fusion

The syntactic and statistical features are concatenated with the semantic feature vectors produced by the CNN-BiLSTM. The result is a single unified feature representation for a single text sample. The syntactic and statistical features represent the formal structure of the text and its distributional properties. While the semantic features describe the potential meaning of the text and its contextual dependencies. Taken together, the two complementary features enhance the robustness of the MFAD approach in classifying text as either human-written or LLM-produced.

3.6. Text classification

The concatenated feature vector is passed to a fully connected (dense) layer to perform binary classification, determining whether the text is human-written or generated by a language model. A sigmoid activation function is utilized to map the outputs to values between 0 and 1 for binary classification tasks (text is produced by human authors or LLM-generated).

4. Experiments and results

4.1. Dataset

Experiments are conducted on the HC3 dataset [28]. The HC3-English dataset comprises approximately 24,322 questions along with corresponding answers from ChatGPT and humans. The HC3 dataset comprises multiple domains, such as open-domain discussions, health, law, finance, and psychology. The HC3-English comprises five datasets: medicine, reddit_eli5, finance, wiki_csai, and open_qa. Each dataset contains questions, responses from humans, and answers from ChatGPT. Human responses are taken from freely available question-answering datasets and wiki text (Stack Overflow, Quora), while ChatGPT (GPT-3.5) answers are obtained by manually inputting the questions. Table 2 presents the dataset statistics and sources.

Table 2. HC3-English dataset statistics and sources

Datasets	Number of question	Human answer count	ChatGPT answer count	Source
All	24,322	58,546	26,903	
reddit_eli5	17,112	51,336	16,660	ELI5[31]
Finance	3933	3933	4503	FiQA dataset [32]
Medicine	1248	1248	1337	Medical dialog dataset [33]
open_qa	1187	1187	3561	WikiQA dataset [34]
wiki_csai	842	842	842	Descriptions of concepts in computer science

4.2. Experimental setup

The experiments were run on a Lenovo laptop with 12.0 GB of RAM, a 64-bit operating system, and a Core i5 processor. The experiments pipeline is implemented in Python using established machine-learning and NLP libraries. Classification performance metrics, including accuracy, precision, recall, and F1-score, are computed by scikit-learn [35]. Basic text preprocessing operations, such as tokenization, stop-word removal, and lemmatization, are performed by the Natural Language Toolkit (NLTK) [36]. Deep neural network models are trained and evaluated using TensorFlow [37]. Syntactic and statistical features reported in Table 1 are extracted by using multiple NLP libraries. Specifically, spaCy is employed for advanced syntactic analysis (syntactic complexity and dependency-based features) [38]. TextStat is used to compute readability and text complexity metrics [39]. LanguageTool is utilized for grammar error detection [40]. PySpellChecker is used to compute the number of spelling errors in the text [41].

The dataset is split at random into 80% training data and 20% test data. This split is applied concurrently to the tokenized text sequences, the handcrafted syntactic/statistical features, and the class labels. In order to track training performance and avoid overfitting, an extra internal validation split of 10% is then taken exclusively from the training set during model training. Because of this, the effective data usage is approximately 72% for training, 8% for validation, and 20% for testing. Until the final evaluation, the test set is completely hidden. The hyperparameters in all the experiments are defined in Table 3.

Table 3. Hyperparameters configuration

Parameters	Value
Activation function (for feature learning)	ReLU
Activation function (for classification)	Sigmoid
Optimizer	Adam
Loss function	binary_crossentropy
size of batch	32
Number of epochs	5
Dropout-rate	0.5
Glove embedding dimensions	100
Pooling strategy	max
Hidden units	128
Kernel-size	5
CNN filters	128
Dataset splits	80 % → training, 20% → testing

4.3. Performance measures

The performance of the MFAD approach is evaluated by accuracy, precision, recall, F1 score, and False Positive Rate (FPR) in this process [11]. FPR is particularly reported to quantify the proportion of human-written texts misclassified as LLM-generated,

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}, \tag{8}$$

$$\text{Precision} = \frac{TP}{TP+FP}, \tag{9}$$

$$\text{Recall} = \frac{TP}{TP+FN}, \tag{10}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{11}$$

$$\text{FPR} = \frac{FP}{FP+TN}. \tag{12}$$

Here, True Positive (TP) is the number of LLM-generated texts correctly labeled as LLM-generated. True Negative (TN) is the number of human-written texts correctly labeled as human-written. False Positive (FP) is the number of human-written texts wrongly assigned to LLM-generated texts. False Negative (FN) is the number of LLM-generated texts that were misclassified as human-written.

The source code and dataset are available at the following repository: **https://github.com/doaaahmed7490-netizen/MFAD**

4.4. Experimental results

This section presents the evaluation of the MFAD approach on the HC3 dataset, including domain-specific evaluation, cross-model generalization, robustness to text-length variations, and resilience to text paraphrasing.

4.4.1. Domain-specific evaluation

The MFAD approach is evaluated on each domain of the HC3 dataset. Table 4 and Figs 2 and 3 present the MFAD evaluation metrics for each domain in the HC3 dataset.

Table 4. The evaluation metrics of the MFAD approach for each domain in the HC3 dataset

Datasets	Accuracy, %	Precision, %	Recall, %	F1-score, %	FPR
Medicine	98	97	99.3	98.14	0.02
Finance	94.6	93.6	96.75	95.23	0.07
open_qa	95.5	96.8	97.2	97	0.08
reddit_eli5	98.68	98.6	96	97.3	0.004
wiki_csai	93.21	94.3	93.8	94	0.1
All	98	96.5	97.5	97	0.01

Table 4 and Figs 2 and 3 present the results of the MFAD approach, which show strong performance for the different domains of the HC3 dataset.

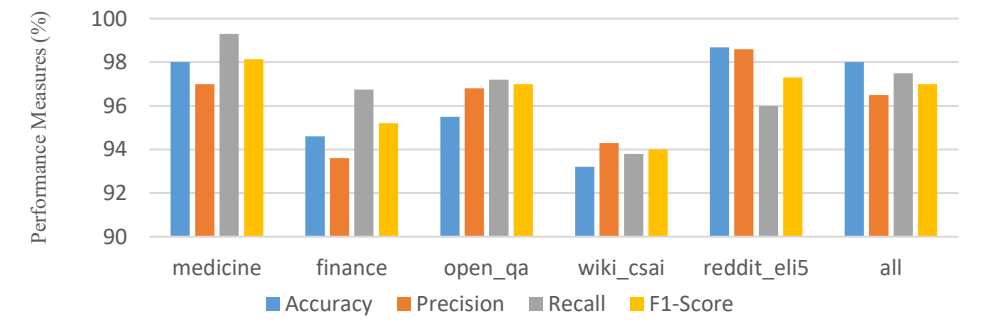

Fig. 2. Evaluation metrics of the MFAD approach for each domain in the HCE dataset

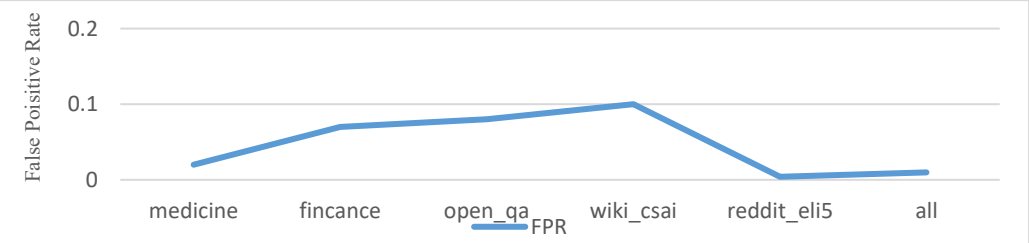

Fig. 3. FPR Evaluation of the MFAD approach for each domain in the HCE dataset

In the medicine domain, MFAD achieved an accuracy of 98%, precision of 97%, recall of 99.3%, F1-score of 98.14%, and FPR of 0.02, indicating highly dependable LLM-generated text identification and minimal misclassification of human-written text. In the finance domain, MFAD obtained an accuracy of 94.6%,

precision of 93.6%, recall of 96.75%, F1-score of 95.23%, and FPR of 0.07, suggesting slightly lower performance, but it is still robust when handling technical finance terms. In the open_QA domain, MFAD achieved an accuracy of 95.5%, precision of 96.8%, recall of 97.2%, F1-score of 97%, and FPR of 0.08, indicating the model showed balanced performance, although the FPR was slightly higher at 0.08, indicating very few human-written responses were misclassified as LLM-generated. The MFAD produced an accuracy of 93.21%, a precision of 94.3%, a recall of 93.8%, an F1-score of 94%, and a False Positive Rate (FPR) of 0.1 in the wiki_csai domain, which was the least favorable overall result across domains. The MFAD produced the highest overall accuracy (98.68%) and the lowest FPR (0.004) in the reddit_eli5 domain, suggesting a suitable efficacy of the MFAD in conversational text types. Lastly, across all domains (all), the MFAD output an accuracy of 98%, an F1-score of 97%, and an FPR of 0.01, which indicates the robustness and reliable detection capabilities of the MFAD in processing different textual styles.

4.4.2. Cross-model generalization

To evaluate the performance of the proposed MFAD approach on advanced text generation models, including GPT-4, Gemini, and Claude-4, we reconstructed two datasets using these models. More precisely, on both the medicine and finance datasets, we prompted each target model with the same set of questions to produce 1,248 responses for the medicine dataset and 805 responses for the finance dataset. The prompt used was "Answer the following question in one paragraph." A uniform generation configuration was used to query all models, and default parameters were used to minimize variability among models. The generated answers were then gathered and saved as a JSON file. The human-written answers were already available in the original dataset and were used unchanged as the negative class. The prompting procedure was applied only to regenerate the ChatGPT responses. These datasets were then utilized to evaluate the cross-model generalization performance of the MFAD approach on outputs from advanced large language models. The proposed MFAD approach was first evaluated on the newly generated medicine and finance datasets, and subsequently trained on the medicine dataset produced by one model and tested on the corresponding dataset generated by another model to further assess cross-model generalization.

1) Model-specific evaluation

The proposed approach is evaluated on medicine and finance datasets generated by GPT-3.5, GPT-4, Gemini, and Claude-4. Table 5 presents the evaluation metrics of the MFAD approach on medicine and finance datasets generated from these LLMs.

The results in Table 5 demonstrate that the proposed MFAD approach can consistently achieve superior performance across both the medicine dataset and finance dataset generated from various language models, highlighting its robustness and cross-model generalization capability. On the medicine dataset, MFAD achieves the highest accuracy values (ranging from 96.4% to 99%) for all of the compared models. MFAD performs best on GPT-4 and Claude-4 outputs, achieving

99% accuracy and F1-scores of approximately 99%, reflecting a strong balance between precision and recall. The very high recall values observed (up to 99.6%) indicate that MFAD is highly effective in the task of distinguishing LLM-generated medical text, and their low false positive rates (0.01-0.02) indicate that human-written samples are slightly prone to misclassification. Slightly lower performance on Gemini-generated medical data (96.4% accuracy) may be attributed to stylistic or structural differences in Gemini outputs, yet MFAD still maintains strong discriminative capability. In the finance dataset, MFAD obtains performances ranging from 93% to 96.3% across all models. The highest performance is observed on GPT-4 and Claude-4 datasets, with F1-scores over 96% indicate reliable detections despite domain complexity. While the false positive rates are higher in the finance domain (0.03-0.07), they are still low, indicating that MFAD maintains a good balance between sensitivity and specificity. The robustness of the performance across many LLMs in both domains indicates that MFAD generalizes effectively beyond any specific generative model.

Table 5. Evaluation metrics of the proposed MFAD approach on medicine and finance datasets generated by various LLMs

Model	Dataset	Accuracy, %	Precision, %	Recall, %	F1-score, %	FPR
GPT-3.5	Medicine	98	97	99.3	98.14	0.02
GPT-4		99	98.8	99.2	99	0.01
Gemini		96.4	97.6	95.3	96.4	0.02
Claude-4		99	98.4	99.6	99	0.01
GPT-3.5	Finance	93.2	93.9	95.9	94	0.07
GPT-4		96.35	95.2	97.6	96.4	0.05
Gemini		93	94.4	93.3	93.9	0.05
Claude-4		96.3	98	94.5	96.3	0.03

2) Cross-model evaluation

To evaluate MFAD cross-model generalization, the approach was trained on the medicine dataset generated by one language model and tested on the medicine dataset generated by another. The reconstructed medicine dataset contains a total of 1,248 samples, of which 80% (998 samples) were used for training, and the remaining 20% (250 samples) were reserved for testing. Table 6 illustrates the evaluation metrics of the MFAD approach when trained on a medicine dataset generated from a specific model and tested on another LLM.

The evaluation of the proposed MFAD approach across various LLMs, indicating its strong cross-model generalization, is presented in Table 6. When the MFAD approach was trained on GPT-3.5 and tested on other LLMs, the MFAD approach achieved high accuracy (above 98%), with particularly reliable results on Gemini (98.8%) and Claude-4 (98.8%). The precision and recall balance show that the model not only detects LLM-generated text reliably but also minimizes false positives, as reflected by the very low FPR (0.001-0.02). When MFAD is trained on the GPT-4 model, its performance decreases slightly compared to training on the GPT-3.5 model. This is also more noticeable when assessed on both the GPT-3.5 (95.13% accuracy) and Gemini (96.67% accuracy) evaluation sets. Still, the precision and F1-score are greater than 96%, while the FPR remains below 0.02. This shows that MFAD demonstrates a good amount of stability while also showing

an indication of slightly less consistency with cross-generalization, with the precision of detection while trained on GPT-4 data, when evaluated on datasets from older LLMs, is slightly lower. When trained on the Gemini model, MFAD's performance varied more noticeably. When tested on GPT-4, it achieved an accuracy of 93.55%, demonstrating that detecting GPT-4 generations is more challenging. However, Gemini training generalized exceptionally well to Claude-4, yielding the highest overall performance with an accuracy of 99.44%, 100% recall, and only 0.01 FPR. Finally, when trained on Claude-4, MFAD displays limited cross-model transferability, achieving an accuracy of 93.7%-95.2% with substantial FPRs (up to 0.04), suggesting that it is a less generalized source of training than GPT-3.5 or GPT-4.

Table 6. Evaluation metrics of the MFAD approach when trained on a medicine dataset generated by a specific model and tested on another LLM

Model		Accuracy, %	Precision, %	Recall, %	F1-score, %	FPR
Train	Test					
GPT-3.5	GPT-4	98.2	99.34	96.96	98.13	0.006
	Gemini	98.8	99.84	97.76	98.79	0.001
	Claude-4	98.8	97.65	100	98.81	0.02
GPT-4	GPT-3.5	95.13	99	93.47	96.16	0.009
	Gemini	96.67	97.47	95.83	96.65	0.02
	Claude-4	98.36	99.6	97.12	98.34	0.004
Gemini	GPT-3.5	95.59	97.89	93.5	95.64	0.02
	GPT-4	93.55	99.73	92.34	95.89	0.002
	Claude-4	99.44	98.89	100	99.44	0.01
Claude-4	GPT-3.5	94.2	96.8	93.2	94.97	0.03
	GPT-4	93.7	95	96.2	95.6	0.04
	Gemini	95.2	94.8	95.2	95	0.01

4.4.3. Robustness evaluation across text length

The MFAD approach is evaluated across variations in text length by dividing each dataset into different token count categories: <50, 50-100, and >100 tokens. The evaluation results for these categories are presented in Table 7 and Fig. 4.

Table 7. Evaluation metrics of the MFAD approach across variations in text lengths

Dataset	Text length (Tokens)	Accuracy, %	Precision, %	Recall, %	F1-score, %
Medicine	≤50	97.52	96.71	100	98.33
	51-100	98.11	97.3	98.51	97.90
	>100	98.37	97.47	99.14	98.30
Finance	≤50	95.55	94.2	97.01	95.58
	51-100	94.79	93.70	96.72	95.20
	>100	94.15	93.25	96.46	94.83
open_qa	≤50	95.55	96.40	96.72	96.56
	51-100	95.97	96.03	97.11	96.57
	>100	96.39	95.82	97.47	96.64
reddit_eli5	≤50	98.88	98.43	96.64	97.53
	51-100	98.62	97.66	95.57	96.60
	>100	98.86	97.04	97.12	97.08
wiki_csai	≤50	92.33	93.25	93.2	93.22
	51-100	93.68	94	94.15	94.07
	>100	92.8	92.7	93.81	93.25
All	≤50	98.32	97.39	97.39	97.39
	51-100	98.38	97.51	97.11	97.31
	>100	98.09	97.10	96.86	96.98

The evaluation of the proposed approach (MFAD) across different text lengths, utilizing all datasets available in HC3, is illustrated in Table 7 and Fig. 4.

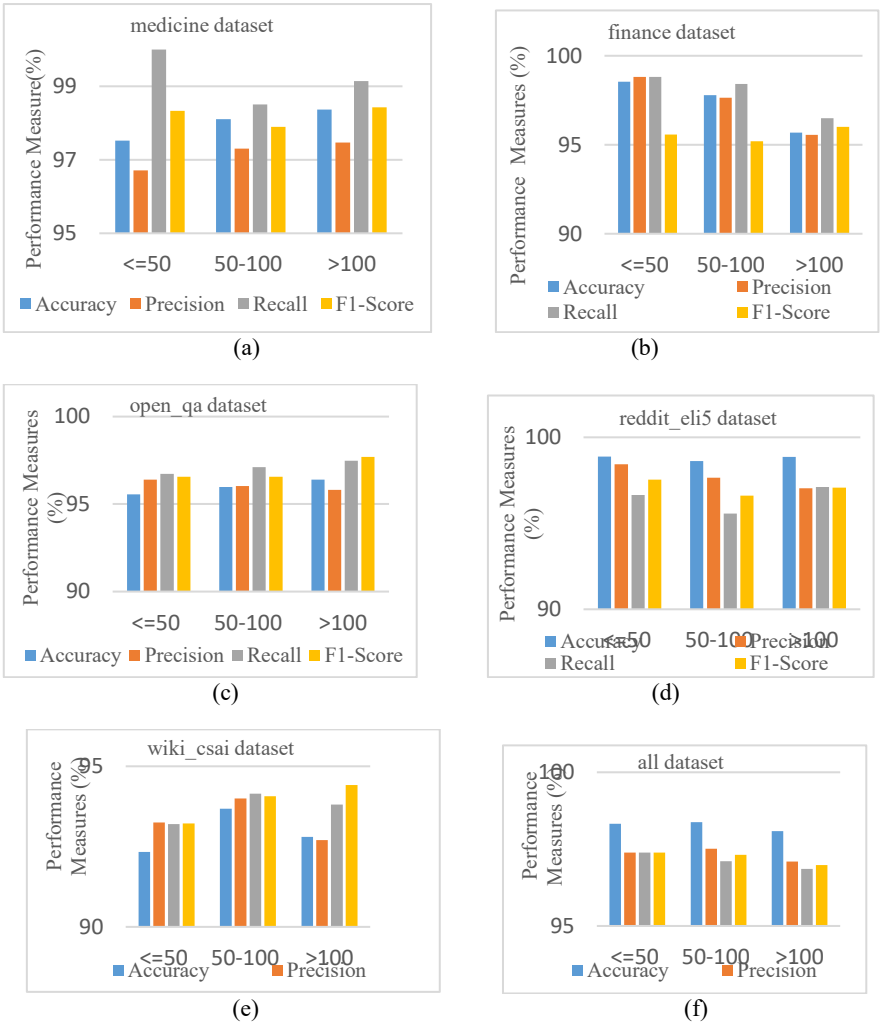

Fig. 4. Evaluation metrics of the MFAD approach across variations in text lengths

The results show that MFAD performs well regardless of the text length used across domains since the accuracy remains consistently high and even exceeds 99%. In the medicine domain, the MFAD achieves accuracy for all text lengths above 97.6% with precision above 96.5%, recall near 100%, and an F1-score exceeding 98.33 as well. In the reddit_eli5 domain, the performance remained strong regardless of length at greater than 97% precision, greater than 98% accuracy, greater than 96% recall, and an F1-score above 96, demonstrating the MFAD's capacity to capture informal writing styles. However, the finance domain recorded a minor drop in its accuracy in longer texts (>100 tokens, 94.15%). In the open_qa domain, MFAD achieves stable levels of performance, remaining close to 95-96% accuracy, but increased recall and F1-scores for longer texts, suggesting

that additional context supports detection across varied question–answering scenarios. The wiki_csai domain produces the lowest results, indicating an accuracy of 92-93%, precision of 92-94%, recall of 93-94%, and an F1-score in the range of 93-94%. In all (all domains), MFAD achieves significant robustness with accuracy above 98% across all lengths and little change in performance.

4.4.4. Robustness evaluation against text paraphrasing

Table 8 and Fig. 5 present the evaluation metrics of MFAD against text paraphrasing. The MFAD approach is evaluated on detecting paraphrased LLM-generated texts that are semantically similar but syntactically different. The T5-base model, a pre-trained text-to-text Transformer presented in [42], is used to generate paraphrases. All NLP tasks, including paraphrasing, are designed to map an input text sequence to an output text sequence in the unified text-to-text transfer learning framework used to train T5. The model uses an encoder–decoder architecture, in which the decoder creates a rephrased sentence based on the contextualized semantic representation that the encoder created from the input sentence. While displaying lexical and syntactic variation, such as word substitutions, reordering, and structural reformulation, without changing the underlying meaning, T5's paraphrased outputs are intended to be semantically equivalent to the original text. This allows the evaluation of the robustness of the MFAD approach under realistic paraphrasing-based text transformations. The MFAD approach is then applied to the paraphrased texts, and the performance metrics are compared with those from the original test set.

Table 8. Evaluation metrics of the MFAD approach against text paraphrasing

Dataset	Type	Accuracy, %	Precision, %	Recall, %	F1-score, %
Medicine	Original	98.45	97.60	99.65	98.62
	Paraphrased	97	95.89	98.95	97.40
Finance	Original	95.10	95.45	96.64	96.04
	Paraphrased	94.3	94.11	96.33	95.21
open_qa	Original	96.40	97.15	97.61	97.28
	Paraphrased	95.02	96.36	95.77	96.06
reddit_eli5	Original	98.5	98.3	96.43	97.36
	Paraphrased	96.6	97.6	95.87	96.73
Wiki_csai	Original	93.36	93.28	94.37	93.82
	Paraphrased	92.1	92.86	93.15	93
All	Original	98.20	96.41	97.93	97.16
	Paraphrased	97.5	95.71	97.10	96.40

Table 8 and Fig. 5 show the evaluation metrics of the MFAD approach with both the original and paraphrased datasets. In the medicine domain, accuracy drops from 98.45% to 97%, and F1-score drops from 98.62% to 97.40%, whereas recall remains very high at 98.95% this indicates that paraphrasing affected precision but not sensitivity. A similar decline is shown in the finance domain, with an accuracy drop from 95.10% to 94.30%, the wiki_csai domain from 93.36% to 92.06%, and the reddit_eli5 domain from 98.50% to 96.60%. However, the open_qa and reddit_eli5 domains exhibit more stable precisions and recall after paraphrasing. In the all-dataset, MFAD shows a similar accuracy drop from 98.20% to 97.50%, and F1-score drop from 97.16% to 96.40%. Overall, these results show that

paraphrasing introduces minor performance degradation across domains, although the MFAD approach presents strong detection performance.

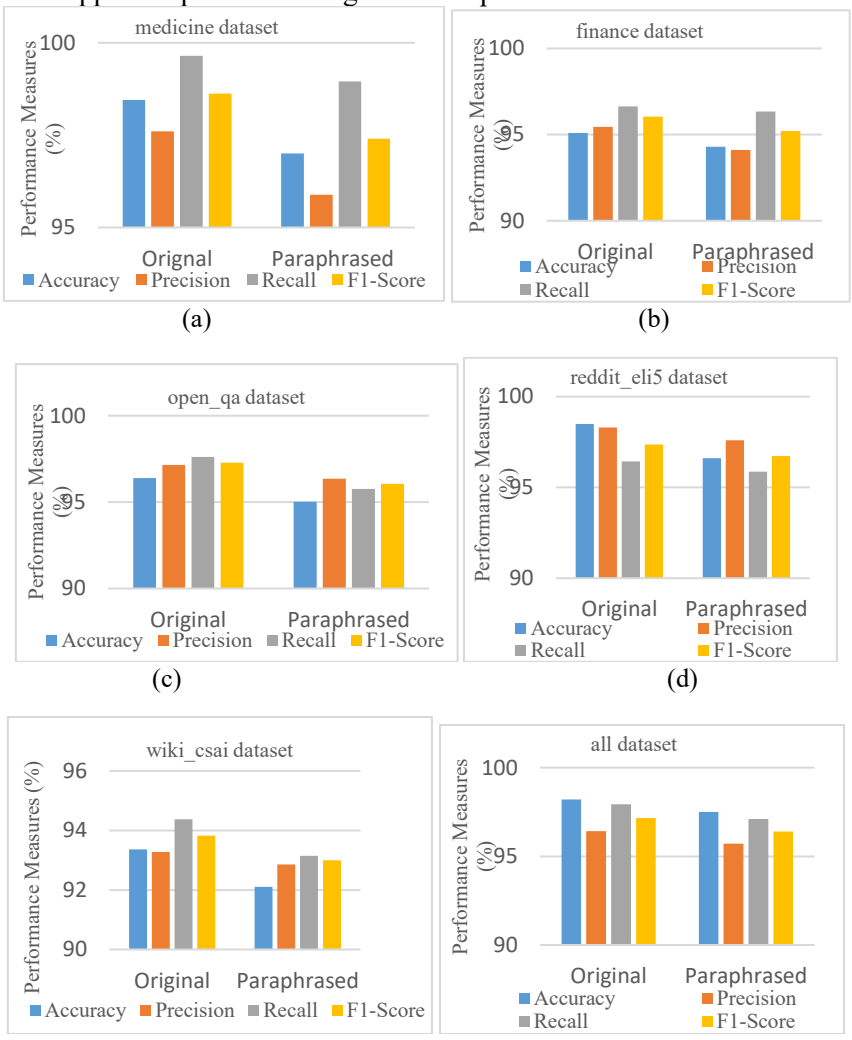

Fig. 5. Evaluation metrics of the MFAD approach against text paraphrasing

4.4.5. Performance comparison between MFAD and a baseline approach

This section compares the proposed MFAD approach with two baseline feature-based methods: the Regularized Deep Neural Network (RDNN) and AuthentiGPT. In accordance with the specifications in their original publications, both baselines were independently re-implemented. RDNN, AuthentiGPT, and MFAD were trained and assessed using the same dataset splits and experimental conditions to guarantee a fair and controlled comparison. The results, which are shown in Table 9, show that on the entire HC3 dataset, MFAD consistently performs better than both baseline approaches.

On the full HC3 dataset, the proposed MFAD approach performs better than the two baseline approaches, with an accuracy of 98% compared to 93.64% for RDNN and 92.4% for AuthentiGPT. MFAD also has superior precision (96.5%), recall (97.5%), and F1-score (97%), which clearly indicates that it is a more accurate approach for detecting LLM-generated text with less chance of misclassifying human-generated text. On the other hand, RDNN and Authenti GPT perform similarly but not as well, with slightly better precision than recall, which indicates a trade-off between the two. Overall, these results clearly indicate that the combination of semantic and handcrafted syntactic/statistical features in the MFAD approach is more robust and balanced than the two baseline approaches.

Table 9. Performance analysis of the proposed MFAD approach compared with baseline approaches

Dataset	Approach	Accuracy, %	Precision, %	Recall, %	F1-score, %
HC3 (All)	RDNN	93.64	93.31	92.6	92.95
	AuthentiGPT	92.4	93	91	92
	MFAD	98	96.5	97.5	97

4.4.6. Ablation study

An ablation study was conducted to evaluate the contribution of each component in the proposed approach, MFAD, focusing on different feature types, word embeddings, and sequence modeling architectures. First, handcrafted syntactic and statistical stylometric indicators are evaluated to measure their individual contribution. Next, semantic features extracted using CNN combined with sequence models such as RNN, LSTM, GRU, and BiLSTM, with GloVe and Word2Vec embeddings, are evaluated. Finally, the reference model (proposed approach) combines handcrafted features with semantic features, which are learned with GloVe, CNN, and BiLSTM techniques.

Table 10. Ablation study on the complete HC3 dataset (all)

Models	Accuracy, %	Precision, %	Recall, %	F1-score, %
Handcrafted Features only	63.6	55.3	57.8	56.5
GloVe-CNN-BiLSTM	95.9	93.5	95.5	94.5
Glove+CNN+LSTM	94.8	93	94.25	93.62
GloVe-CNN-GRU	92.66	90.8	92.36	92.56
Glove-CNN-RNN	88.65	82	81.8	81.91
Word2Vec-CNN-BiLSTM	92.6	93.17	94.23	93.69
Word2Vec-CNN-LSTM	93	91.15	92.34	91.74
Word2Vec-CNN-GRU	90.2	89.23	90.87	90.04
Word2Vec-CNN-RNN	86.4	79.9	80.24	80.07
Handcrafted+ Semantic: (GloVe-CNN-BiLSTM) (Proposed Approach MFAD)	98	96.5	97.5	97

As shown in Table 10, the ablation study demonstrates the impact of both feature representation and model architecture on detection performance. The accuracy (63.6%) and F1-score (56.5%) with handcrafted features are low, proving that statistical and stylistic features are insufficient to distinguish between human and AI-generated text. Adding semantic embeddings improves the accuracy of all models. GloVe embeddings perform better than Word2Vec embeddings, proving

the superiority of GloVe in global semantic representation [43]. From the architectural point of view, BiLSTM performs better than LSTM, GRU, and RNN, proving the effectiveness of bidirectional representation for long-distance dependencies, while GRU and RNN decrease the precision and F1-score. The best accuracy (98%) and F1-score (97%) are obtained by the proposed MFAD model (Handcrafted + GloVe-CNN-BiLSTM), proving that a combination of Handcrafted syntactic and statistical features with deep semantic features provides a more comprehensive and discriminative feature representation.

5. Discussion

The results present the reliability and robustness of the MFAD approach in detecting LLM-generated text. In domain-specific evaluations, the MFAD approach performs particularly well on informal (reddit_eli5) and structured (medicine) datasets, although performance decreases on more factual datasets such as wiki_csai. This implies that sentences with factual information are challenging to detect because their inherent simplicity and lack of complex stylistic variations make them closely resemble human writing. In cross-model generalization, the MFAD approach achieves positive results for both newer and older LLMs. Notably, Gemini was slightly more difficult to detect; nevertheless, MFAD maintained ≥96% accuracy. Strong detection was achieved for GPT4 and Claude-4 outputs. In addition, the MFAD approach is robust for various text lengths with only small fluctuations in its resulting performance, which allows for more consistent outputs across varying text source conditions. Despite a very slight drop in performance metrics when evaluated against paraphrased texts compared to original texts across the datasets, MFAD continues to demonstrate strong detection capability. Although MFAD achieves high accuracy on the HC3 dataset, its generalizability to other datasets with different linguistic properties or domains is also an area for future research.

6. Conclusion

The proposed MFAD approach integrates handcrafted syntactic and statistical features with deep semantic representations. The MFAD approach has the interpretability of linguistic and statistical features and the representational power of deep learning models, making it more innovative and robust than approaches that depend on a single type of feature. MFAD consists of six stages: text preprocessing, handcrafted feature extraction, text representation with GloVe, semantic feature extraction by CNN and BiLSTM, feature fusion, and text classification. The preprocessing stage includes several steps, such as cleaning, normalization, lemmatization, and tokenization. Handcrafted feature extraction includes obtaining both syntactic and statistical features from the given text. For text representation, GloVe represents tokens as dense, fixed-length vectors. A CNN is used to capture local phrase patterns, while contextual dependencies are captured by BiLSTM. Finally, handcrafted and semantic features are concatenated and passed through a

fully connected layer for classification. All experiments are conducted on the HC3 dataset, which showed that MFAD achieved an accuracy of 98%, precision of 96.5%, recall of 97.5%, and F1-score of 97%, with the minimum False Positive Rate (FPR) being 0.01 across multiple domains. Further, MFAD was robust against adversarial contexts, which would include paraphrasing. It also remained stable at different lengths of texts and was adaptable to advanced LLMs, such as GPT-4, Gemini, and Claude-4. Future research will focus on extending MFAD to larger and more diverse multilingual datasets. Also, future research directions will include the evaluation of the performance of the MFAD approach in specialized domains, such as scientific literature.

References

1. Achiam, J., S. Adler, S. Agarwal, L. Ahmad, I. Akkaya. GPT-4 Technical Report. – arXiv Preprint arXiv:2303.08774, 2023. DOI:10.48550/arXiv.2303.08774.
2. Priyanshu, A., Y. Maurya, Z. Hong. AI Governance and Accountability: An Analysis of Anthropic's Claude. – arXiv Preprint arXiv:2407.01557, 2024. DOI:10.48550/arXiv.2407.01557.
3. Team, G., R. Anil, S. Borgeaud, J. Alayrac, J. Yu, R. Soricut, D. Silver. Gemini: A Family of Highly Capable Multimodal Models. – arXiv Preprint arXiv:2312.11805. 2023. DOI:10.48550/arXiv.2312.11805.
4. Pagnoni, A., M. Graciarena, Y. Tsvetkov. Threat Scenarios and Best Practices to Detect Neural Fake News. – In: Proc. of 29th International Conference on Computational Linguistics, 2022, pp. 1233-1249.
5. Weidinger, L., J. Mellor, M. Rauh, C. Griffin, J. Uesato, P. Huang, M. Chen. Ethical and Social Risks of Harm from Language Models. – arXiv Preprint arXiv:2112.04359, 2021. DOI: 10.48550/arXiv.2112.04359.
6. Mirsky, Y., A. Demontis, J. Kotak, R. Shankar, D. Gelei, L. Yang, X. Zhang, M. Pintor. The Threat of Offensive AI to Organizations. – Computers & Security. 2023. DOI: 10.1016/j.cose.2022.103006.
7. Ahmed, A., A. Aljabou. Detecting Fake News Using Machine Learning: A Systematic Literature Review. – arXiv Preprint arXiv:2102.04458, 2021. DOI: 10.48550/arXiv.2102.04458.
8. Adelani, D. I., H. Mai. Generating Sentiment-Preserving Fake Online Reviews Using Neural Language Models and Their Human and Machine-Based Detection. – In: Proc. of International Conference on Advanced Information Networking and Applications 2020, pp. 1341-1354.
9. Tang, R., X. Hu. The Science of Detecting LLM-Generated Text. – Communications of the ACM. 2024, pp. 50-59. DOI:10.1145/3624725.
10. Floridi, L., M. Chiriatti. GPT-3: Its Nature, Scope, Limits, and Consequences. – Minds and Machines, 2020, pp. 681-694. DOI:10.1007/s11023-020-09548-1.
11. Wu, J., S. Yang, R. Zhan, Y. Yuan, L. S. Chao, D. F. Wong. A Survey on LLM-Generated Text Detection: Necessity, Methods, and Future Directions. – Computational Linguistics, 2025, pp. 275-338. DOI: 10.1162/coli_a_00549.
12. Fraser, K. C., H. Dawkins, S. Kiritchenko. Detecting AI-Generated Text: Factors Influencing Detectability with Current Methods. – Journal of Artificial Intelligence Research, 2025, pp. 2232-2278, DOI: 10.1613/jair. 1.16665.
13. Devlin, J., M. W. Chang. Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding. – In: Proc. of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Volume 1 (Long and Short Papers), 2019, pp. 4171-4186. DOI: 10.18653/v1/N19-1423.

14. Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy. Roberta: A Robustly Optimized BERT Pretraining Approach. – arXiv Preprint arXiv:1907.11692, 2019. DOI: 10.48550/arXiv.1907.11692.
15. Pennington, J., R. Socher, C. D. Manning. Glove: Global Vectors for Word Representation. – In: Proc. of Conference on Empirical Methods in Natural Language Processing (EMNLP'14), 2014, pp. 1532-1543.
16. Zhou, H. Research on Text Classification Based on TF-IDF and CNN-LSTM. – In: Journal of Physics: Conference Series, 2022. DOI: 10.1088/1742-6596/2171/1/012021.
17. Mitchell, E., Y. Lee, A. Khazatsky, C. D. Manning, C. Finn. DetectGPT: Zero-Shot Machine-Generated Text Detection Using Probability Curvature. – In: Proc. of International Conference on Machine Learning, 2023, pp. 24950-24962.
18. T5

    https://huggingface.co/docs/transformers/en/model_doc/t5
19. Bao, G., Y. Zhao, Z. Teng, L. Yang, Y. Zhang. Fast-Detectgpt: Efficient Zero-Shot Detection of Machine-Generated Text via Conditional Probability Curvature. – arXiv Preprint arXiv:2310.05130, 2023. DOI: 10.48550/arXiv.2310.05130.
20. Bhattacharjee, A., T. Kumarage, R. Moraffah, H. Liu. Conda: Contrastive Domain Adaptation for AI-Generated Text Detection. – arXiv Preprint arXiv:2309.03992, 2023. DOI: 10.48550/arXiv.2309.03992.
21. Fraser, K. C., H. Dawkins, S. Kiritchenko. Detecting AI-Generated Text: Factors Influencing Detectability with Current Methods. – Journal of Artificial Intelligence Research. 2025, pp. 82:2233-2278, DOI: 10.1613/jair.1.16665.
22. Modupe, A., T. Celik. Post-Authorship Attribution Using a Regularized Deep Neural Network. – Applied Sciences. 2022. DOI:10.3390/app12157518.
23. Guo, Z., S. Yu. Authentigpt: Detecting Machine-Generated Text via Black-Box Language Models Denoising. – arXiv Preprint arXiv:2311.07700, 2023. DOI: 10.48550/arXiv.2311.07700.
24. Kirchenbauer, J., J. Geiping, Y. Wen. A Watermark for Large Language Models. – In: Proc. of International Conference on Machine Learning, 2023, pp. 17061-17084.
25. Lee, T., S. Hong, J. Ahn, I. Hong, H. Lee, S. Yun, J. Shin, G. Kim. Who Wrote This Code? Watermarking for Code Generation. – arXiv Preprint arXiv:2305.15060, 2023. DOI: 10.48550/arXiv.2305.15060.
26. Gehrmann, S., H. Strobelt, A. M. Rush. Gltr: Statistical Detection and Visualization of Generated Text. – arXiv Preprint arXiv:1906.04043, 2019. DOI:10.48550/arXiv.1906.04043.
27. Arusada, M. D., N. A. Putri, A. Alamsyah. Training Data Optimization Strategy for Multiclass Text Classification. – In: Proc. of 5th International Conference on Information and Communication Technology (ICoIC7'17) 2017, pp. 1-5. DOI: 10.1109/ICoICT.2017.8074652.
28. Guo, B., X. Zhang, Z. Wang, Y. Wu. How Close is ChatGPT to Human Experts? Comparison Corpus, Evaluation, and Detection. – arXiv Preprint arXiv:2301.07597, 2023.
29. Muñoz, A., C. G. Rodríguez, D. Vilares. Contrasting Linguistic Patterns in Human and LLM-Generated News Text. – Artificial Intelligence Review, 2024.
30. Opara, C. StyloAI: Distinguishing AI-Generated Content with Stylometric Analysis. – In: Proc. of International Conference on Artificial Intelligence in Education 2024, pp. 105-114. DOI: 10.48550/arXiv.2405.10129.
31. Fan, A., Y. Jernite, E. Perez, D. Grangier, J. Weston, M. Auli. ELI5: Long-Form Question Answering. – arXiv preprint arXiv:1907.09190, 2019. DOI: 10.48550/arXiv.1907.09190.
32. Maia, M., S. Handschuh, A. Freitas, B. Davis, R. McDermott, M. Zarrouk, A. Balahur. Www'18 Open Challenge: Financial Opinion Mining and Question Answering. – In: Proc. of the Web Conference 2018, Companion Proceedings of 2018, pp. 1941-1942. DOI: 10.1145/3184558.319230.
33. Chen, S., Z. Ju, X. Dong, H. Fang, S. Wang, Y. Yang, J. Zeng, R. Zhang, M. Zhou, P. Zhu. Meddialog: A Large-Scale Medical Dialogue Dataset. – arXiv Preprint arXiv:2004.03329, 2020.

34. Y a n g, Y., W. T. Y i h, C. M e e k. Wikiqa: A Challenge Dataset for Open-Domain Question Answering. – In: Proc. of Conference on Empirical Methods in Natural Language Processing, 2015, pp. 2013-2018.
35. P e d r e g o s a, F., G. V a r o q u a u x, A. G r a m f o r t, V. M i c h e l, B. T h i r i o n, J. V a n d e r p l a s. Scikit-Learn: Machine Learning in Python. – Journal of Machine Learning Research, 2011 Nov 1; 12:2825-30.
36. B i r d, S. NLTK: the Natural Language Toolkit. – In: Proc. of COLING/ACL 2006 Interactive Presentation Sessions 2006, pp. 69-72.
37. A b a d i, M., A. A g a r w a l, P. B a r h a m, E. B r e v d o, Z. C h e n, C. C i t r o, C o r r a d o, S. G h e m a w a t. Tensorflow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. – arXiv Preprint arXiv:1603.04467, 2016.
38. H o n n i b a l, M. SpaCy 2: Natural Language Understanding with Bloom Embeddings. – Convolutional Neural Networks, and Incremental Parsing, 2017.
39. Textstat.
https://textstat.org/.
40. LanguageTool.
https://pypi.org/project/language_tool_python/
41. Pyspellchecker.
https://pypi.org/project/pyspellchecker/
42. R a f f e l, C., N. S h a z e e r, A. R o b e r t s, K. L e e, S. N a r a n g, M. M a t e n a, W. L i. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. – Journal of Machine Learning Research, 2020, pp. 1-67.
43. M i k o l o v, T., K. C h e n. Efficient Estimation of Word Representations in Vector Space. – arXiv Preprint arXiv:1301.3781, 2013.

Received: 15.10.2025, First revision: 12.01.2026, Second revision: 13.03.2026, Accepted: 25.03.2026