Examining the Robustness and Resilience of AI-Generated Text Detectors

Supriya P Upadhyaya*, Aarathi Vijayachandran*, and Shashankh M G*

*Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, Germany
{supriya.upadhyaya, aarathi.vijayachandran, shashankh.girish}@st.ovgu.de

Abstract. With the increasing prevalence of Large Language Models (LLMs) generating human-like text, ensuring the reliability of AI-generated text detection has become critical. Various AI-generated text detectors claim high performance across different generative models and attack scenarios. This paper critically examines these claims by conducting comprehensive experiments to evaluate the robustness of popular detectors—BLOOMZ, RoBERTa, and XGBoost—against text generated by diverse LLMs with varying linguistic styles, including cross-model and cross-domain evaluations, as well as their resilience to adversarial and prompt attacks. Our study reveals that detectors fine-tuned on specific models, such as BLOOMZ or ChatGPT, struggle when tested on text generated by other models, highlighting their sensitivity to the underlying style and features of the training data. Furthermore, cross-domain experiments show that detectors trained on text from one domain (e.g., scientific abstracts) perform poorly when applied to another domain (e.g., Wikipedia articles), underscoring the challenge of generalizing across different content types. We demonstrate that achieving perfect robustness and resilience is not feasible. However, we provide an estimate of the amount of additional data, representing new patterns or styles, required to significantly improve the performance of fine-tuned detectors, particularly in cross-model and cross-domain scenarios. These findings emphasize the importance of model adaptability and highlight the limitations of existing detectors when faced with the diverse and evolving nature of AI-generated text. The code is available at https://github.com/SupriyaUpadhyaya/ai-generated-text-detector.

Keywords: AI-generated text detectors · Cross-Model Evaluation · Cross-Domain Evaluation · Prompt Attacks · Adversarial attacks

1 Introduction

AI-generated text refers to content produced by Large Language Models (LLMs), which are trained on vast datasets to predict and generate text based on prompts. These models, such as GPT-3, GPT-4o, Claude 3.5, Gemma and Llama 3.1 can generate coherent and contextually relevant text across various domains, from academic writing to casual conversation. AI-generated text is designed to mimic human writing, often making it difficult to distinguish from human-authored content. AI-generated text detectors are tools designed to differentiate between(classify) human-written content and text generated by LLMs. These detectors could be based on several approaches. Watermarking technology embeds subtle, trackable patterns in model outputs, ensuring traceability and accountability. Zero-shot detectors leverage pre-trained LLMs to identify generated content without further training, though they can lack precision. Fine-tuned LLM detectors are customized on labeled datasets for improved accuracy in identifying AI-generated text. Adversarial learning methods train detectors to recognize outputs by exploiting the weaknesses of generative models. LLMs as detectors involve using large models themselves to assess whether content is AI-generated based on patterns in the text. Finally, human-assisted methods combine expert review with machine predictions for nuanced evaluations, especially where automated tools fall short. Each method addresses different detection needs and contexts.

The rise of LLMs in fields like academia, journalism, and content creation increases the need for detection tools. Unregulated use of AI-generated text can result in plagiarism, fake news, and academic misconduct. Additionally, AI-generated content can include hallucinations or fabricated information, making detection crucial to maintain the integrity of various fields. The detectors play a critical role in ensuring the responsible and ethical use of AI-generated content, help mitigate misuse and harm, combat misinformation, preventing abuse, protect users from manipulative or unsafe interactions, contribute to maintaining trust in AI technologies and enable the practice of responsible AI.

State-of-the-art detectors claim high accuracy rates under controlled conditions. For instance, Desaire et al. (2023a) report that their XGBoost-based classifier detects ChatGPT-generated text with 98-100% accuracy, depending on the prompt. Detector GPT-who (Venkatraman, Uchendu, and Lee (2024))

achieves over 20% improvement in detection accuracy compared to GLTR, GPTZero, DetectGPT, and OpenAI's detector across multiple domains.

However, AI text detectors face several challenges. Evasion and paraphrasing attacks can drastically reduce detector effectiveness while only slightly degrading text quality. Adversarial attacks also pose a significant challenge, as attackers can intentionally craft text to mislead detectors by altering the output of LLMs without affecting its readability. For instance, recursive paraphrasing attacks can reduce detection accuracy effectively bypassing both watermarking-based and non-watermarking-based detectors. These attacks present a major challenge in ensuring the long-term resilience of detectors. Furthermore, many detectors show bias toward non-native English speakers, tending to misclassify human-written content as AI-generated, which presents risks to academic integrity. Also, Detectors trained on older LLMs like GPT-3.5 may experience a slight decline in performance when tested against newer models like GPT-4 suggesting that while detectors can be robust against newer generations, continuous updates are necessary to sustain effectiveness. The design of prompts plays a critical role in AI detection. Detectors that are effective against straightforward prompts may fail when faced with prompts that obscure the use of AI. Studies show that tools like ZeroGPT struggle to detect AI-generated text when the prompt includes human-written abstracts or complex instructions, often misclassifying such content. The performance of detectors can be improved by fine-tuning them with new examples of text generated by more advanced LLMs. However, the exact number of examples needed varies depending on the model.

Despite their high-performance claims, AI-generated text detectors face critical challenges regarding adaptability, robustness against adversarial attacks, and cross-model generalizability. Given the challenges and limitations faced by current AI-generated text detectors, our research aims to address two key questions: (1) How robust and resilient are popular AI-generated text detectors (such as BLOOMZ, RoBERTa, and XGBoost) when faced with text generated by diverse LLMs across different linguistic styles, domains, and attack scenarios? (2) What is minimal fine-tuning data necessary to significantly improve the adaptability and generalization of AI-generated text detectors (e.g., BLOOMZ, RoBERTa, and XGBoost) in cross-model contexts?

By addressing these questions, we seek to contribute a deeper understanding of the robustness, resilience and adaptability of AI-generated text detection tools.

2 Background and Related Work

With the proliferation of Large Language Models (LLMs), the detection of AI-generated text has become an important area of study. Various methods have been proposed to address this issue, particularly in the context of academic integrity and the prevention of text-based plagiarism.

Desaire et al. (2023a) developed a model to detect AI-generated text, specifically within scientific chemistry journals, where AI use is becoming more prevalent. Their approach used an XGBoost classifier based on 20 textual features to differentiate between human-written text and text produced by GPT-3.5 and GPT-4. They achieved 99% accuracy, even when AI-generated content was deliberately designed to mimic human writing. This study demonstrated the utility of AI detection in maintaining the integrity of scientific literature, where journal guidelines now require full disclosure of AI use.

While Desaire et al. (2023a) focused on domain-specific applications, Sadasivan et al. (2023) explored the robustness of AI text detectors across multiple domains. Their research revealed critical vulnerabilities in current detection systems, particularly when subjected to adversarial attacks like recursive paraphrasing. They showed that even sophisticated detectors, including those employing watermarking techniques, could be deceived by these methods. The study highlights a major challenge in ensuring reliable AI detection: the ability of adversaries to evade detection using simple paraphrasing methods without significantly degrading text quality.

Expanding on these efforts, Wu et al. (2023) conducted a comprehensive survey of the methods used to detect LLM-generated text. They explored a variety of approaches, including watermarking, zero-shot learning, adversarial training, and human-assisted detection, and identified key challenges, such as out-of-distribution issues and model ambiguity. Their work emphasized the need for robust detection systems that can adapt to the increasing capabilities of LLMs while ensuring responsible AI usage.

Weber-Wulff et al. (2023) investigated the accuracy and reliability of publicly available AI detection tools in academic settings. Testing 12 freely available and 2 commercial detectors, the researchers found that none of the tools could reliably distinguish between human-written and AI-generated text, with a significant bias toward classifying AI-generated text as human. Furthermore, they demonstrated that common content obfuscation techniques, such as paraphrasing and machine translation, drastically reduced detection accuracy, posing a significant challenge to academic integrity.

Zhou, Wang, and Liu (2023) proposed a framework that introduced adversarial learning to attack AI text detectors. They demonstrated that with minor perturbations like paraphrasing, AI text detectors can misclassify machine-generated text as human-written. Their findings emphasized the need for models to be more robust against adversarial attacks, especially as these attacks could be executed within seconds, evading detection with ease.

Huang, Liu, and Zhang (2023) introduced the Siamese Calibrated Reconstruction Network (SCRN), which outperformed previous detectors by achieving 6.5-18.25% improvement in detection accuracy under adversarial perturbations. They emphasized the importance of robust detection systems that could withstand word- and character-level attacks.

Together, these studies highlight the complexities involved in detecting AI-generated text. While some methods, like those developed by Desaire et al. (2023a), show promise in specific domains, the broader applicability of these detection tools remains limited due to vulnerabilities, as demonstrated by Sadasivan et al. and Weber-Wulff et al. These gaps in the literature suggest a need for more robust, adaptable detection systems to mitigate the misuse of AI-generated content in academic and other high-stakes contexts.

In our research, we examine three AI-generated text detector models, including the **XGBoost Classifier**, **roberta-academic-detector** and **bloomz-560m-academic-detector**. The **XGBoost Classifier**, introduced by Desaire et al. (2023b) was trained using human text data extracted from the abstracts of the first 10 research articles published in the November 2022 issue of ten notable chemistry journals, such as Inorganic Chemistry and ACS Nano. Corresponding AI-generated abstracts were created using prompts derived from these article titles, leveraging ChatGPT-3.5 for text generation. The model identifies differences between human and AI-generated content through 20 text style features (1) grouped into four categories: paragraph complexity, sentence-level diversity in length, punctuation usage, and popular word frequency. Desaire et al. (2023a) reported a 99% classification accuracy, a result we aim to verify and assess within our own experimental framework.The **roberta-academic-detector**, a fine-tuned version of RoBERTa, released to Hugging Face by Bentzen Winje and Sivesind (2023). RoBERTa (Robustly Optimized BERT Pretraining Approach), introduced by Liu et al. (2019),builds on the foundational BERT architecture with significant optimizations, including larger datasets, increased batch sizes, and a focus on masked language modeling without the Next Sentence Prediction objective. The **bloomz-560m-academic-detector**, a model sourced from Hugging Face (Bentzen Winje and Sivesind (2023)). BLOOMZ, introduced by Muennighoff et al. (2023), is a multilingual and multi-task language model that extends the capabilities of BLOOM (BigScience Large Open-science Open-access Multilingual Language Model). BLOOMZ is fine-tuned to perform zero-shot and few-shot tasks across a wide range of languages and domains, making it highly adaptable for diverse natural language processing applications. The multilingual and cross-task training approach of BLOOMZ allows it to generalize well across different linguistic contexts, which is particularly beneficial for distinguishing between AI-generated and human-written text in various scenarios. This adaptability of BLOOMZ and the enhancements in RoBERTa-based detectors enable them to effectively capture and differentiate subtle linguistic patterns, making both models strong candidates for detecting AI-generated content in various contexts.

In this research project, to test the robustness of AI-generated text detector models against adversarial attacks, we use the TextAttack Framework for Adversarial Attacks as defined by Morris et al. (2020). TextAttack is a Python framework for adversarial attacks, data augmentation, and adversarial training in NLP. It provides implementations of 16 adversarial attacks and supports various models and datasets, including transformers like BERT and tasks such as GLUE. It is directly integrated with Hugging Face **transformers** and **nlp** libraries. TextAttack is also available as a Python package that can be installed from **PyPI** or downloaded directly from **GitHub**. TextAttack aims to implement attacks that, given an NLP model, find a perturbation of an input sequence that satisfies the attack's goal while adhering to certain linguistic constraints. Attacking an NLP model can thus be framed as a combinatorial search problem. The attacker must search within all potential transformations to find a sequence of transformations that generate a successful adversarial example. Each attack in TextAttack can be constructed from four components:

- **Goal Function:** A task-specific function that determines whether the attack is successful based on the model's outputs. Examples: untargeted classification, targeted classification, non-overlapping output, minimum BLEU score.
- **Constraints:** A set of conditions that determine if a perturbation is valid with respect to the original input. Examples: maximum word embedding distance, part-of-speech consistency, grammar checker, minimum sentence encoding cosine similarity.

Feature number	Feature type (1-4)	Short description	Greater in
1	1	sentences per paragraph	human
2	1	words per paragraph	human
3	2	')' present	human
4	2	'-' present	human
5	2	';' or ':' present	human
6	2	'?' present	human
7	2	'"' present	ChatGPT
8	3	standard deviation in sentence length	human
9	3	length difference for consecutive sentences	human
10	3	sentence with less than 11 words	human
11	3	sentence with more than 34 words	human
12	4	contains 'although'	human
13	4	contains 'However'	human
14	4	contains 'but'	human
15	4	contains 'because'	human
16	4	contains 'this'	human
17	4	contains 'others' or 'researchers'	ChatGPT
18	4	contains numbers	human
19	4	contains 2 times more capitals than '.'	human
20	4	contains 'et'	human

Table 1: Feature types and their presence in human vs. machine-generated text (ChatGPT) (Desaire et al. (2023b)). Feature types: 1 - paragraph complexity; 2 - punctuation marks; 3 - diversity in sentence length; and 4 - popular words or numbers.

- **Transformation:** A method that, given an input, generates a set of potential perturbations. Examples: word embedding word swap, thesaurus word swap, homoglyph character substitution.
- **Search Method:** A strategy that successively queries the model and selects promising perturbations from a set of transformations. Examples: greedy search with word importance ranking, beam search, genetic algorithm.

Out of the 16 adversarial attacks implemented by TextAttack, we selected 4 for our experiments, as shown in Table 2.

Attack Type	Goal Function	Constraints	Transformation	Search Method
deepwordbug (Gao et al. (2018))	{Untargeted, Targeted} Classification	Levenshtein edit distance	{Character Insertion, Character Deletion, Neighboring Character Swap, Character Substitution}	Greedy-WIR
pruthi (Pruthi, Dhingra, and Lipton (2019))	Untargeted Classification	Minimum word length, Maximum number of words perturbed	{Neighboring Character Swap, Character Deletion, Character Insertion, Keyboard-Based Character Swap}	Greedy search
pwws (Ren et al. (2019))	Untargeted Classification	None	WordNet-based synonym swap	Greedy-WIR (saliency)
textfooler (Jin et al. (2020))	Untargeted {Classification, Entailment}	Word Embedding Distance, Part-of-speech match, Use sentence encoding cosine similarity	Counter-fitted word embedding swap	Greedy-WIR

Table 2: Selected 4 TextAttack adversarial attack types categorized within framework: search method, transformation, goal function, constraints. Greedy Search with Word Importance Ranking is abbreviated as Greedy-WIR (Morris et al. (2020)).

3 Datasets

The main dataset used for this research project is the **M4 dataset** (Wang et al. (2023)). M4 is a **M**ulti-generator, **M**ulti-domain, and **M**ulti-lingual dataset specifically designed for the detection of **M**achine-

generated texts. It is aimed at supporting black-box detection methods, where the internal parameters of large language models (LLMs) are not required for detecting machine-generated content.

The M4 dataset includes both human-written texts and their corresponding machine-generated versions. The human-written texts are gathered from a diverse range of sources, covering several languages and domains. For English, sources include Wikipedia (March 2022 version), WikiHow, Reddit (ELI5 - Explain Like I'm Five), arXiv abstracts, and PeerRead. For other languages, the sources are Baike/Web QA (Chinese), RuATD (Russian), Arabic Wikipedia (Arabic), True & Fake News (Bulgarian), Indonesian newspapers (id_newspapers_2018), and Urdu news (Urdu). This wide range of sources ensures that the dataset is representative of multiple domains and languages, making it highly versatile for machine-generated text detection tasks.

In total, the M4 dataset contains approximately 147,000 parallel human–machine text pairs, with 102,000 pairs in English and 45,000 in other languages. The non-English data includes 9,000 pairs each for Chinese, Russian, and Bulgarian, and 6,000 pairs each for Urdu, Indonesian, and Arabic. Additionally, the dataset includes over 10 million non-parallel human-written texts, providing a large corpus for training and evaluation purposes.

The machine-generated texts in M4 were produced using state-of-the-art LLMs, including GPT-4, ChatGPT, GPT-3.5 (text-davinci-003), Cohere, Dolly-v2, BLOOMZ 176B, and Flan-T5. These texts were generated using various prompts designed to simulate real-world usage, with different styles such as formal, informal, and simplified language. A summary of the contents of M4 dataset from original paper Wang et al. (2023) is shown in Figure 1. Since the publishing of the original paper, the authors have continued to update this dataset.

Source/ Domain	Data License	Language	Total Human	Parallel Data							
				Human	Davinci003	ChatGPT	GPT4	Cohere	Dolly-v2	BLOOMz	Total
Wikipedia	CC BY-SA-3.0	English	6,458,670	3,000	3,000	2,995	3,000	2,336	2,702	3,000	20,033
Reddit ELI5	Huggingface	English	558,669	3,000	3,000	3,000	3,000	3,000	3,000	3,000	21,000
WikiHow	CC-BY-NC-SA	English	31,102	3,000	3,000	3,000	3,000	3,000	3,000	3,000	21,000
PeerRead	Apache license	English	5,798	5,798	2,344	2,344	2,344	2,344	2,344	2,344	19,862
arXiv abstract	CC0-public domain	English	2,219,423	3,000	3,000	3,000	3,000	3,000	3,000	3,000	21,000
Arabic-Wikipedia	CC BY-SA-3.0	Arabic	1,209,042	3,000	–	3,000	–	–	–	–	6,000
True & Fake News	MIT License	Bulgarian	94,000	3,000	3,000	3,000	–	–	–	–	9,000
Baike/Web QA	MIT license	Chinese	113,313	3,000	3,000	3,000	–	–	–	–	9,000
id_newspapers_2018	CC BY-NC-SA-4.0	Indonesian	499,164	3,000	–	3,000	–	–	–	–	6,000
RuATD	Apache 2.0 license	Russian	75,291	3,000	3,000	3,000	–	–	–	–	9,000
Urdu-news	CC BY 4.0	Urdu	107,881	3,000	–	3,000	–	–	–	–	6,000
Total				35,798	23,344	32,339	14,344	13,680	14,046	14,344	147,895

Fig. 1: Summary of M4 dataset, which includes non-parallel human data and parallel human and machine-generated texts (Wang et al. (2023)).

The M4 dataset underwent several quality tests to ensure its reliability. Data cleaning was performed to remove artifacts like bullet points, and redundant formatting, including eliminating references and short paragraphs from human-written texts. Multiple prompts (2-8) were used for each model to generate diverse outputs, ensuring the text reflected real-world scenarios. Text length was controlled, with machine-generated texts having a minimum of 1,000 characters for English to avoid overly short content. Human evaluation was conducted by annotators proficient in NLP, who compared human and machine-generated texts, revealing challenges in detecting machine-generated content, particularly for non-native speakers. N-gram analysis showed that human-written texts had a richer vocabulary, which helped develop better black-box detection methods. Finally, several detectors, including RoBERTa, ELECTRA, and XLM-R, were evaluated for their ability to detect machine-generated texts, with their performance compared using precision, recall, and F1 scores.

3.1 Reasons for Choosing M4 Dataset

The reasons for choosing M4 as the main dataset for this research project are summarized below:

– **Black-box Applicability:** M4 is designed for black-box detection, meaning it focuses on scenarios where the internal workings of the text generator are unknown. This is particularly useful for real-world applications where users may not have access to the LLM's architecture or parameters, only the

output. By supporting black-box detection, M4 provides a practical solution for detecting machine-generated texts across a wide range of applications without needing access to the underlying model.

- **Multi-generator Advantage:** M4 includes texts generated by multiple large language models (LLMs) such as GPT-3.5, GPT-4, Cohere, Dolly-v2, Flan-T5, and BLOOMZ. This diversity allows for comprehensive testing and model development to detect text from various types of LLMs, enhancing the generalization ability of the detection models. By incorporating multiple generators, M4 ensures that the trained detection systems can handle a broader range of machine-generated content.
- **Multi-domain Coverage:** The dataset covers a variety of domains such as Wikipedia, Reddit, arXiv, and news articles. This wide range of content allows researchers to test detection models across different types of writing, from formal academic language to informal social media posts. The multi-domain nature of M4 ensures that models can adapt and perform well in different contexts, making them more robust and versatile.
- **Multi-lingual Capability:** M4 is multi-lingual, with texts available in English, Chinese, Russian, Arabic, Bulgarian, Indonesian, and Urdu. This allows for the development of machine-generated text detection models that can function across different languages, making M4 particularly useful for global applications. It helps ensure that models trained on this dataset can handle the linguistic nuances present in various languages, enhancing their effectiveness in multi-lingual environments.
- **Human-Written Texts and Corresponding Machine-Generated Versions:** M4 includes parallel datasets of human-written texts and their machine-generated counterparts across multiple domains and languages. This parallel structure provides a powerful foundation for comparing and contrasting human-authored content with machine-generated text. By having both versions available, the dataset enables precise training of detection models, improving their ability to distinguish subtle differences in structure, style, and content, which is crucial for enhancing accuracy in real-world machine-generated text detection.
- **Prompts Used:** M4 leverages a diverse set of carefully crafted prompts (ranging from 2 to 8 per model) to generate machine-written texts, ensuring that the outputs reflect a variety of real-world scenarios. These prompts cover different writing styles and tones, such as casual, expert, or formal, making the machine-generated texts more representative of the actual use cases of LLMs. This diversity in prompts allows for better generalization of models trained on M4, as they learn to detect machine-generated texts produced under a wide range of conditions.

Overall, M4 serves as a critical benchmark for evaluating the performance of machine-generated text detection systems. Its multi-generator, multi-domain, and multi-lingual scope, along with the inclusion of both human and machine-generated texts, makes it a versatile and valuable resource for research in this area.

3.2 Subset of the M4 Dataset Selected for Experiments

For this research project, we primarily focused on the arXiv abstracts from the M4 dataset, with Wikipedia articles being used for minor experiments. The subset of the latest version of M4 used for all experiments is summarized in Table 3. The arXiv abstracts subset was chosen because research papers follow a standard format, allowing for easy extraction of abstracts using the Python arXiv module, and to maintain a focus within the academic domain. For both the arXiv abstracts and Wikipedia articles, there are 3,000 human-written original texts and their corresponding machine-generated texts, produced using ChatGPT, text-davinci-003, Cohere, and BLOOMZ 176B. Additionally, for the arXiv abstracts, there are machine-generated texts produced using Flan-T5.

Source/Domain	Language	Human	ChatGPT	BLOOMZ	Davinci003	Cohere	Flan-T5	Total
Wikipedia	English	3,000	2,995	3,000	3,000	2,336	-	14,331
arXiv abstract	English	3,000	3,000	3,000	3,000	3,000	3,000	21,000

Table 3: Selected Sub-set of M4 Dataset.

3.3 Training, Validation, and Test Splits

The selected subset of the M4 dataset (Section 3.2) was divided into Training, Validation, and Test sets. The human-written abstracts were combined with their corresponding machine-generated abstracts from various models. After merging, the combined dataset was split into training, validation, and test sets, using random selection without repetition. The dataset split follows this distribution:

- **Training set**: 64% of the total data,
- **Validation set**: 16% of the total data,
- **Test set**: 20% of the total data.

Separate training, validation, and test sets were created for each human-machine pair outlined in Table 3. These sets ensure that both human and machine-generated abstracts are evenly represented across all data splits. An example of this dataset split is shown in Table 4.

Total 6000 abstracts = 3000 human-written abstracts + 3000 corresponding machine-generated abstracts									
	Training - 64%			**Validation - 16%**			**Test - 20%**		
	Total	**Human**	**Machine**	**Total**	**Human**	**Machine**	**Total**	**Human**	**Machine**
Abstracts	3840	1920	1920	960	480	480	1200	600	600

Table 4: Distribution of 3000 human-written arXiv abstracts and 3000 corresponding machine-generated abstracts from BLOOMZ into Training, Validation, and Test sets.

3.4 Extension of the M4 Dataset

We created several modified versions of the subset of the M4 dataset (Section 3.2) to be used in specific experiments, as described below:

Extension with Paraphrased Machine-Generated Text: Krishna et al. (2024) introduces an 11-billion-parameter paraphrase generation model, DIPPER (**Di**scourse **P**ara**p**hras**er**), which is capable of paraphrasing paragraphs while conditioning on surrounding context, as well as controlling lexical diversity and content reordering. The DIPPER model was utilized to paraphrase the M4 dataset's machine-generated arXiv abstracts from ChatGPT and BLOOMZ. These paraphrased machine-generated texts were then combined with their corresponding human-written texts and subsequently split into training, validation, and test sets, following the procedure described in Section 3.3.

Extension with Machine-Generated Text from New LLMs:

- **Extension with Machine-Generated Text from LLaMA-3:** For cross-model robustness experiments (discussed in Section 4.2), we created a version of the subset of the M4 Dataset (Section 3.2) containing machine-generated arXiv abstracts from LLaMA-3. To achieve this, we used the BLOOMZ-generated arXiv abstracts from the M4 dataset and generated prompts based on the corresponding arXiv paper titles. The prompts were designed with instructions to generate a 150-250 word single-paragraph abstract. These prompts were then passed to the **meta-llama/Meta-LLaMA-3-8B-Instruct** model from Hugging Face to obtain machine-generated arXiv abstracts. The machine-generated abstracts were combined with their corresponding human-written abstracts, and the dataset was split into training, validation, and test sets, following the procedure outlined in Section 3.3.
- **Extension with Machine-Generated Text from LLaMA-3.1:** As a base for prompt-attack experiments (discussed in Section 4.7), we created a version of the subset of the M4 dataset (Section 3.2) containing machine-generated arXiv abstracts produced by the LLaMA-3.1 model. For this, we utilized the ChatGPT-generated arXiv abstracts test set (Section 3.3), created prompts using the corresponding arXiv paper titles, and provided instructions to generate a 150-250 word single-paragraph abstract. These prompts were passed to the **LLaMA-3.1** model, accessed via Ollama, to generate the machine-generated arXiv abstracts. The resulting machine-generated abstracts were then combined with the corresponding human-written abstracts. Finally, 100 random pairs of human-written and machine-generated abstracts were selected as the base for the prompt-attack datasets, which will be further developed in Section 3.4.

Extension with Prompt-Attack-Oriented Machine-Generated Text: For the prompt-attack experiments on AI-generated text detectors (to be discussed in Section 4.7), we used the 100 pairs of human-written and LLaMA-3.1-generated abstracts to create prompt-attack-oriented datasets. To achieve this, the machine-generated abstracts were integrated into attack-instruction prompts and passed through the **Gemini-1.5-Flash** model from Google AI, producing prompt-attacked machine-generated abstracts.

The prompt-attacked machine-generated abstracts were then combined with the corresponding human-written abstracts to form prompt-attack-oriented dataset. For each prompt-attack type (to be discussed in Section 4.7), the corresponding prompt-attack-oriented dataset was created using the Gemini-1.5-Flash model.

3.5 ML-LLaMA-3 Dataset of ArXiv Machine Learning Research Paper Abstracts

For the cross-domain robustness experiments (which will be discussed in Section 4.3), we developed a new dataset, referred to as the **ML-LLaMA-3 dataset**, following the same format as the M4 dataset. This dataset consists of pairs of human-written and machine-generated arXiv abstracts. The ArxivPapers dataset, defined by Kardas et al. (2020), comprises a corpus of over 100,000 scientific papers related to machine learning, sourced from ArXiv.

Using the code provided by Kardas et al. (2020), we created a sub-dataset that contains only the arXiv paper IDs and titles. With this sub-dataset, and using the Python arXiv module, we extracted the original human-written abstracts from the first 3,000 papers in the ArxivPapers dataset.

For these 3,000 papers, we generated prompts based on their titles and used the **meta-llama/Meta-LLaMA-3-8B-Instruct model** from Hugging Face to produce machine-generated abstracts. After filtering out rows where the machine-generated abstracts were too short (less than 200 characters), we finalized the ML-LLaMA-3 dataset, which contains 2,973 pairs of human-written and machine-generated arXiv abstracts.

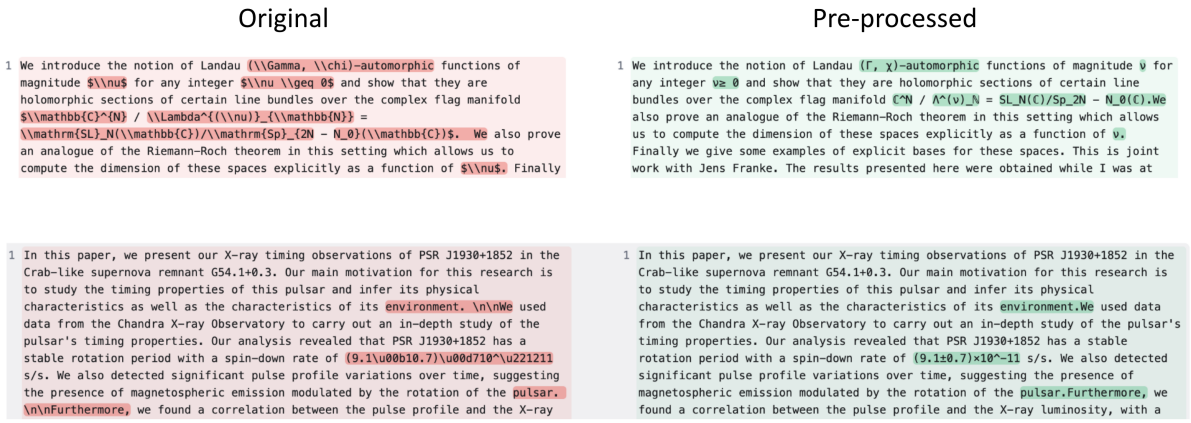


Fig. 2: Comparison of original text Vs preprocessed text highlighting new line characters, LaTeX code, tab characters, non-printable characters, and Unicode symbols

3.6 Data Pre-Processing

During the experiments to test Cross-Model Robustness (Section 4.2), we observed significant bias (Section 5) in the behavior of the AI-generated text detector models under study. Consequently, all datasets created in Sections 3.3, 3.4 and 3.5 were subjected to a comprehensive pre-processing step to eliminate any potential biases and improve the accuracy and consistency of the model evaluation results.

During this step, new line characters, LaTeX code, tab characters, non-printable characters, and Unicode symbols were removed from both the human-written and machine-generated text. This pre-processing step was critical to ensure that the AI-generated text detectors could focus on the meaningful content within the input text, thereby improving their ability to accurately detect patterns and classify the text as either human-written or machine-generated.

If new lines, LaTeX code, tabs, non-printable characters, and Unicode symbols remain in the text during training or fine-tuning, the detector models may learn to recognize these characters as features

of the human-written or machine-generated text. This can introduce bias in the model (as we observe in Section 5), where it may correctly classify input text containing these special characters but struggle to accurately classify text that lacks them. Consequently, the model may fail to generalize to unseen input that does not contain such characters.

Furthermore, the presence of these unnecessary characters can introduce noise, disrupt the natural flow of the text, confuse tokenization processes, and degrade the overall performance of the detector models. By pre-processing the data to remove or normalize these elements, we ensure cleaner and more consistent text inputs, which significantly enhances the accuracy, robustness, and reliability of the models.

4 Experiments

4.1 Experiment Setup

Evaluation Metrics: To evaluate the performance of our AI-generated text detector models, we used the following key metrics: Confusion Matrix, Accuracy, Precision, Recall, and F1-Score. These metrics provide a comprehensive evaluation of the models' ability to classify text as either human-written or machine-generated. Depending on the requirements of each experiment, one or more of these metrics were applied.

- **Confusion Matrix**
  The confusion matrix is a tabular representation of the model's predictions versus the actual outcomes. For AI-generated text detectors, the matrix shows how well the model distinguishes between machine-generated text and human-produced text. It breaks down the classification into four categories:
  - **True Positives (TP)**: Machine-generated text correctly classified as machine-generated.
  - **True Negatives (TN)**: Human-produced text correctly classified as human-produced.
  - **False Positives (FP)**: Human-produced text incorrectly classified as machine-generated (Type I error).
  - **False Negatives (FN)**: Machine-generated text incorrectly classified as human-produced (Type II error).

Actual \Predicted	**Class 0 - Human-Produced**	**Class 1 - Machine-Generated**
Class 0 - Human-Produced	TN	FP
Class 1 - Machine-Generated	FN	TP

Table 5: Confusion Matrix of AI-Generated Text Detector.

  For a high-performing AI-generated text detector, the expected values in the confusion matrix should be:
  - **True Positives (TP)** and **True Negatives (TN)** should be as high as possible, ideally above 95%, meaning the detector correctly classifies the vast majority of both machine-generated and human-produced texts.
  - **False Positives (FP)** and **False Negatives (FN)** should be minimized, ideally below 5%, to reduce errors in classification.
- **Accuracy**
  Accuracy represents the proportion of correct predictions (both true positives and true negatives) out of the total number of predictions made by the model. It is a general measure of how often the model is correct. A well-performing AI-generated text detector should have an accuracy greater than 90%. However, accuracy might not always reflect the model's effectiveness in cases where the dataset is imbalanced.

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

- **Precision**
  Precision measures the proportion of true positive predictions (correctly identified machine-generated texts) out of all positive predictions made by the model. It indicates how accurate the model is when

it predicts a text as machine-generated. A high precision value (preferably above 90%) is expected for a good model, as this indicates that when the model predicts a text as machine-generated, it is likely correct.

$$\text{Precision} = \frac{TP}{TP + FP}$$

– **Recall (Sensitivity)**
Recall, or sensitivity, measures the proportion of true positives out of all actual machine-generated texts. It evaluates the model's ability to detect machine-generated text when it is present. A high recall (above 90%) is desirable for a good model, as it indicates the model is capable of detecting most of the machine-generated text. However, a high recall can sometimes come at the cost of lower precision, meaning the model might classify more human-written text as machine-generated.

$$\text{Recall} = \frac{TP}{TP + FN}$$

– **F1-Score**
The F1-Score is the harmonic mean of precision and recall, balancing both metrics to give an overall performance measure. It provides a single metric that takes both false positives and false negatives into account. The F1-Score is particularly useful when there is an uneven class distribution or when both precision and recall are important. A high F1-Score (above 90%) indicates a good balance between precision and recall, showing that the model performs well in detecting both human-written and machine-generated texts.

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

Detectors under study: From Bentzen Winje and Sivesind (2023) and Desaire et al. (2023b), we know that the three AI-generated text detector models under study, namely, **roberta-academic-detector**, **bloomz-560m-academic-detector**, and the **XGBoost Classifier**, were trained on ChatGPT-based datasets.

Experiment Settings: The entire project was implemented using Python 3.10 in the Google Colab environment. Based on availability, three different GPUs were utilized for training or fine-tuning the detector models under study: 40 GB NVIDIA A100 GPU, 24 GB NVIDIA L4 GPU, and 16 GB NVIDIA T4 GPU. All datasets from Sections 3.3, 3.4 and 3.5 were stored in the **.jsonl** format for efficient handling and processing.

4.2 Cross-model Robustness:

Cross-model robustness of an AI-generated text detector model refers to the model's ability to maintain high performance when detecting text generated by different AI models, not just the one it was trained on. It ensures the model generalizes well across various machine-generated texts, even from unseen models.

Fine-tuning a model on data created by different models can improve its cross-model robustness. It exposes the model to diverse text generation styles, enabling it to detect subtle differences in machine-generated texts.

Without fine-tuning, the model may perform well only on the type of machine-generated text it was trained on and may struggle to detect texts from unseen models. With fine-tuning, the model should generalize better across different types of AI-generated texts, improving its ability to detect machine-generated content from various sources.

To test cross-model robustness, the test sets of the M4 arXiv abstract dataset (from Sections 3.3 and 3.4), generated by the same model used in training (i.e., ChatGPT) and different models (i.e., BLOOMZ, Davinci, Cohere, Flan-T5 and LLaMA-3), were passed through the three pre-trained detector models. This process was repeated after separately fine-tuning the three detectors using the training and validation sets of the M4 arXiv abstract dataset from ChatGPT and BLOOMZ. The performance of each model was evaluated separately using the metrics from Section 4.1, both before and after fine-tuning.

4.3 Cross-domain Robustness:

Cross-domain robustness is the ability of an AI-generated text detector model to generalize and maintain performance when classifying texts across different domains (e.g., academic papers, social media posts, or news articles), beyond the domain it was trained on.

Fine-tuning a model on data from different domains enhances cross-domain robustness by exposing the model to diverse text patterns and structures. This helps the model generalize better and improve performance across various domains.

Without fine-tuning, the model may struggle with data from domains it was not trained on, leading to lower accuracy. With the help of fine-tuning, the model should show better performance across multiple domains, improving its ability to handle new types of data.

For Cross-Domain Robustness Testing:

- The three pre-trained AI-generated text detector models were fine-tuned separately using the training and validation sets of the ML-LLaMA-3 Dataset of ArXiv Machine Learning Research Paper Abstracts (Section 3.5). After fine-tuning, the test set of the same dataset was passed through the detector models, and the evaluation metrics were recorded.
- The three pre-trained detector models were fine-tuned separately using the training and validation sets of the M4 arXiv abstract dataset from ChatGPT and BLOOMZ. The test sets from the ML-LLaMA-3 Dataset and M4 Wikipedia articles dataset from ChatGPT, BLOOMZ, Davinci, and Cohere (Section 3.3) were passed through the detector models, and the evaluation metrics were recorded.
- The three detector models, already fine-tuned on the M4 arXiv abstract dataset from BLOOMZ, were further fine-tuned using the training and validation sets of the M4 Wikipedia articles dataset from BLOOMZ. The following test sets were then passed through the double fine-tuned models, and the evaluation metrics were recorded:
  - M4 arXiv abstract dataset from BLOOMZ, ChatGPT, Davinci, Cohere, and Flan-T5.
  - M4 Wikipedia articles dataset from BLOOMZ, ChatGPT, Davinci, and Cohere.
- The three detector models, already fine-tuned on both M4 arXiv abstract and Wikipedia articles datasets from BLOOMZ, were tested on the test set of the ML-LLaMA-3 Dataset, and the evaluation metrics were recorded for each model separately.

4.4 Robustness against Paraphrased Machine-Generated Text:

Robustness against Paraphrased Machine-Generated Text refers to an AI-generated text detector's ability to correctly identify machine-generated text that has been paraphrased to evade detection. Paraphrasing can alter the structure of sentences and words while retaining the same underlying meaning, challenging detectors.

Fine-tuning a model on paraphrased machine-generated data from different models improves its robustness by exposing it to various paraphrased forms of machine-generated text. This enables the detector to learn patterns beyond surface-level wording.

A model without fine-tuning may fail to detect paraphrased machine-generated text since it is optimized for the exact structure of the text it was trained on. This can result in a higher false-negative rate (i.e., more machine-generated texts incorrectly classified as human-written). A fine-tuned model is expected to perform better on paraphrased texts, as it learns to detect underlying semantic and stylistic cues common in machine-generated text, leading to improved detection accuracy across paraphrased content. It should be more resilient to text that has been syntactically altered but still machine-generated.

The following fine-tuned variants of the three AI-generated text detector models were evaluated by passing the test set of the paraphrased version of the M4 dataset's machine-generated arXiv abstracts from ChatGPT and BLOOMZ (Section 3.4) separately, and the evaluation metrics were recorded:

- Models fine-tuned separately using the training and validation sets of the M4 arXiv abstract dataset from ChatGPT and BLOOMZ.
- Models double fine-tuned using the training and validation sets of the M4 arXiv abstract and Wikipedia articles datasets from BLOOMZ.

4.5 Determine Optimal Training Subset:

Subset Training of an AI-Generated Text Detector Model refers to training or fine-tuning the model on a smaller subset of the full dataset to determine the minimum amount of data required for optimal performance. This method helps identify the percentage of training data necessary for the model to achieve its best results.

Subset training is crucial for reducing the amount of data used for training, thereby minimizing training time and computational costs. By identifying the optimal subset, it ensures efficient model training without overusing resources.

Fine-tuning using subset training allows the model to achieve optimal performance with a smaller subset of data, reducing the need for large datasets and accelerating the training process. The model can generalize effectively even when trained on fewer examples, improving efficiency. In contrast, without fine-tuning, the model may require the full dataset to reach similar levels of performance, making it less efficient in terms of time and computational resources. Fine-tuning helps to optimize both performance and resource utilization in the training process.

For each percentage in the set (1, 2, 4, 8, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100), a corresponding percentage of data was taken from the training set (in order from the beginning) of the M4 arXiv abstract dataset from Cohere, and the **roberta-academic-detector** and **bloomz-560m-academic-detector** models were fine-tuned. The respective test set of the M4 arXiv abstract dataset from Cohere was then used to evaluate these models separately, and the evaluation metrics were recorded. The training and evaluation for each subset were conducted independently of the others.

The progression (1, 2, 4, 8, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100) is a geometric-like progression with incremental increases. It is used for subset training to systematically evaluate model performance as the amount of training data increases. By starting with small subsets and gradually expanding, this approach helps determine the minimum amount of data required to achieve optimal performance, minimizing training time and computational costs while ensuring the model generalizes well across larger datasets.

4.6 Robustness against Adversarial Attacks:

Adversarial attacks on an AI-generated text detector model involve deliberate manipulations of machine-generated text to evade detection. These manipulations include subtle changes, such as paraphrasing, adding irrelevant characters, or altering grammar and syntax, designed to trick the detector into misclassifying machine-generated text as human-written. Robustness against adversarial attacks is the model's capability to correctly classify these manipulated or adversarial texts. A robust detector maintains its accuracy and successfully identifies machine-generated content even after adversarial modifications have been made.

As mentioned in Section 2, to test the robustness of AI-generated text detector models against adversarial attacks, we use the TextAttack Framework for Adversarial Attacks.

We downloaded the TextAttack code from GitHub and plugged in the **roberta-academic-detector** model into the framework. The selected adversarial attacks—**deepwordbug, pruthi, pwws, and textfooler**—were executed in the Colab environment. The model accuracy was recorded both before and after each attack to evaluate their impact on detection performance.

4.7 Robustness against Prompt Attacks:

Prompt attacks on an AI-generated text detector model occur when input texts are altered to evade detection by using carefully designed instruction prompts created through prompt engineering. These attacks often involve changes in wording structure, or addition of irrelevant information into the input text, to trick the model into classifying machine-generated text as human-written, thus bypassing the detector's ability to recognize AI-generated content. Robustness against prompt attacks refers to the model's ability to maintain accurate detection of machine-generated text, even when adversarial prompts are used to disguise it. A robust model effectively recognizes AI-generated content despite the modifications in the inputs using prompt engineering.

For this research project, we tested the robustness of detector model against the following 8 types of prompt attacks:

- **Spelling and punctuation errors**: Deliberate introduction of typos or incorrect punctuation to confuse detectors.

- **Change tense**: Modifying the verb tense (e.g., past to present or future) to alter the structure without changing the meaning.
- **Change of voice**: Switching between active and passive voice to evade detection by modifying sentence structure.
- **Synonyms**: Replacing words with synonyms to maintain meaning while altering detectable patterns.
- **Informal language or slang**: Introducing casual or colloquial expressions to make the text appear less formal and more human-like.
- **Sentence restructuring**: Reordering sentences or breaking them up to change syntax.
- **Idiomatic expressions**: Adding idioms or culturally specific phrases to add human-like tone.
- **Grammar mistakes**: Intentionally adding grammatical errors to mimic human mistakes and mislead the detector.

For each of the 8 prompt-attack types above, the corresponding prompt-attack-oriented dataset was created using the Gemini-1.5-Flash model, as described in Section 3.4.

Figures 3 and 4 present examples from prompt-attack-oriented datasets. Figure 3 illustrates a synonym-based prompt attack, while Figure 4 shows a slang-based prompt attack.

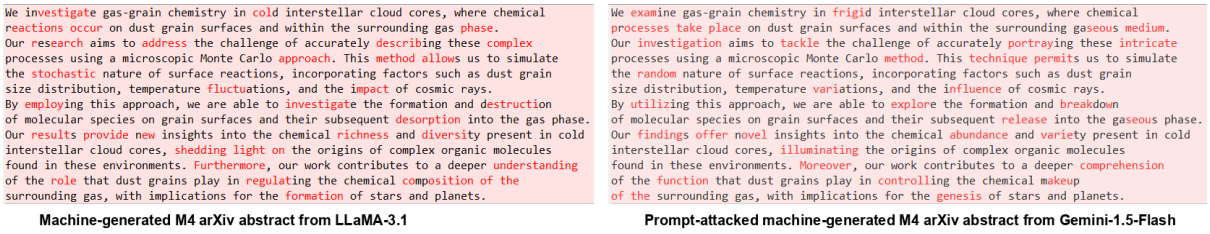


Fig. 3: An example of prompt attack on a machine-generated M4 arXiv abstract, where words are replaced with their synonyms without changing the overall meaning of the text.

Fig. 4: An example of a prompt attack on a machine-generated M4 arXiv abstract, where informal language or slang is added to make the text more human-like.

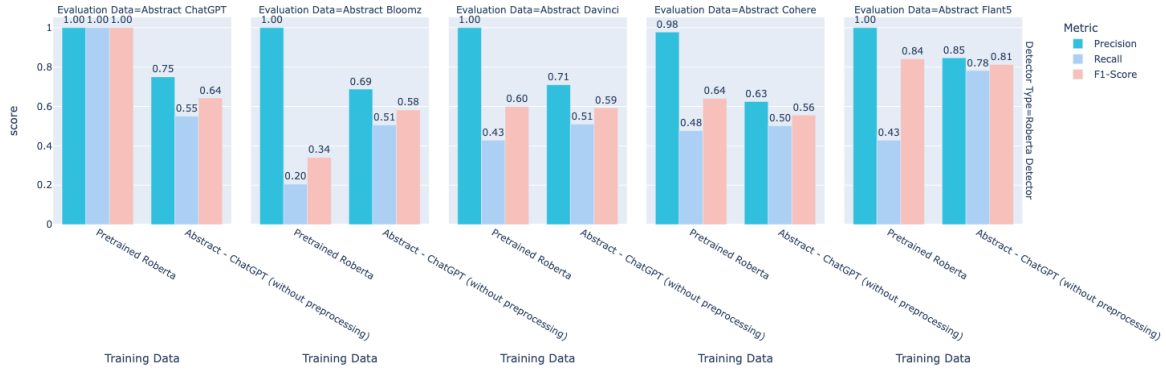
The 8 prompt-attack-oriented datasets were then independently passed, one by one, to the **roberta-academic-detector** model, and the corresponding evaluation metrics were recorded.

5 Results and Discussion

In this section, we present the evaluation metrics reflecting the performance of the three detectors: RoBERTa Detector, BLOOMZ Detector, and XGBoost Detector. Each of these detectors was subjected to a series of experiments using the M4 dataset and it's extension. The performance of these models is discussed in detail in the following subsections. Notably, each detector faced significant challenges under the conditions presented by the M4 and it's extension dataset, revealing key insights into their robustness and limitations. Comprehensive experiment results are available in the report hosted on our GitHub repository.

(a) XGBoost Detector

(b) Bloomz Detector



(c) RoBERTa Detector

Fig. 5: Evaluation metrics for pre-trained/fine-tuned XGBoost, Bloomz and Robereta detector on the M4 Abstract dataset. The dataset is not preprocessed and includes newline characters, LaTeX code, tabs, non-printable characters, and Unicode symbols.

In this subsection, we evaluate the performance of both pretrained and fine-tuned versions of the RoBERTa and BLOOMZ detectors on abstract generations produced by different language models, as sourced from the M4 dataset. Initially, the dataset is used without any preprocessing to assess the sensitivity of the detectors to raw text inputs. However, due to an observed bias in the model responses, a preprocessed version of the dataset is subsequently employed to ensure a more balanced evaluation.

Additionally, two XGBoost models are trained for comparison: one on abstract generations from ChatGPT and another on BLOOMZ abstract generations. These models allow us to explore the impact of different source models on the performance of the XGBoost Classifier with 20 style features Desaire et al. (2023a).

Evaluating pre-trained/fine-tuned model on M4 dataset:

- **Pretrained BLOOMZ Detector:** The pretrained BLOOMZ detector achieves a very high F1 score of 1.0 when detecting generations from the ChatGPT model (Figure 5b) since it was data for pretraining included the ChatGPT research abstracts. However, it performs poorly on generations from all other models including BLOOMZ. This suggests that while the model is excellent at detecting its own outputs, it struggles to generalize to other models' text.
- **Fine-Tuned BLOOMZ Detector:** After fine-tuning, the BLOOMZ detector achieves near-perfect performance across all models (Figure 5b).
- **Pretrained RoBERTa Detector:** The pretrained RoBERTa model performs well on Abstract ChatGPT generations, achieving a perfect F1 score of 1.0 (Figure 5c). However, its performance decreases significantly on other models. For example, on Abstract Cohere, it manages only an F1 score of 0.64, and on Abstract BLOOMZ, the F1 score drops further to 0.34. These lower scores highlight the model's inability to generalize across different generation styles.
- **Fine-Tuned RoBERTa Detector:** When fine-tuned on the ChatGPT dataset without preprocessing, while the RoBERTa detector shows improved performance from 0.34 to 0.5824 F1 score for BLOOMZ abstract, the performance on ChatGPT abstract itself decreases from a perfect F1 score of 1.0 to 0.75 (Figure 5c).
- **XGBoost Detector:** The XGBoost model trained on ChatGPT abstract generations achieves an F1 score of 0.93 on the same ChatGPT-generated data (Figure 5a). However, it performs poorly on other models, with an F1 score of only 0.25 on Abstract BLOOMZ and 0.33 on Abstract Davinci generations. Similarly, the XGBoost model trained on BLOOMZ abstracts performs best on Abstract BLOOMZ, with an F1 score of 0.79, but struggles with other models, such as Abstract ChatGPT (0.39) and Abstract Davinci (0.42).

Across the detectors, both pretrained and fine-tuned, the observed high performance—especially the perfect F1 scores on certain models—largely stems from the models' sensitivity to formatting artifacts such as new lines, latex code, tabs, non-printable characters, and Unicode symbols. These artifacts artificially inflate performance, masking the true capability of the detectors in recognizing the stylistic or content-based features of generative texts. Therefore, preprocessing is necessary to remove these artifacts, ensuring a more accurate and unbiased evaluation of the detectors' generalization ability across different language models.

Evaluating model fine-tuned on preprocessed M4 dataset:

- **BLOOMZ Detector Fine-Tuned on BLOOMZ Generations:** When the BLOOMZ detector is fine-tuned on BLOOMZ generations, it performs well on its own model, with an F1 score of 0.99 on Abstract BLOOMZ and comparatively good F1 score of 0.93 on Abstract ChatGPT (Figure 6). However, its performance on other models is significantly lower. For instance, on Abstract Cohere, the F1 score drops further to 0.55.
- **BLOOMZ Detector Fine-Tuned on ChatGPT Generations:** The BLOOMZ detector fine-tuned on ChatGPT generations performs well on Abstract ChatGPT with an F1 score of 0.99 but shows poor performance on all other models. Notably, on Abstract BLOOMZ, the F1 score plummets to 0.10, and on Abstract Cohere, the F1 score is a mere 0.24 (Figure 6).
- **RoBERTa Detector Fine-Tuned on ChatGPT Generations:** The RoBERTa detector fine-tuned on ChatGPT generations performs exceptionally well on Abstract ChatGPT, with an F1 score of 1.0 (Figure 6). However, it struggles with other model generations. For example, on Abstract BLOOMZ, the F1 score is 0.42, and on Abstract Cohere, the F1 score is 0.73, reflecting a drop in performance when the detector encounters text from different models.

– **RoBERTa Detector Fine-Tuned on BLOOMZ Generations:** Unlike the other detectors, the RoBERTa detector fine-tuned on BLOOMZ generations shows strong performance across all models. For instance, it achieves an F1 score of 0.995 on Abstract ChatGPT, 0.992 on Abstract BLOOMZ, and 0.955 on Abstract Cohere (Figure 6). This suggests that fine-tuning on BLOOMZ enables the RoBERTa detector to generalize better across various models' generations, making it the most versatile model in this evaluation.
– **XGBoost Detector:** The XGBoost classifier trained on ChatGPT generations shows moderate performance, with an F1 score of 0.84 on Abstract ChatGPT but performs poorly on other models, achieving an F1 score of nearly 0 on Abstract BLOOMZ and 0.24 on Abstract Cohere (Figure 6). The XGBoost model trained on BLOOMZ generations fares worse, with an F1 score of only 0.77 on Abstract BLOOMZ and 0.12 on Abstract ChatGPT. This highlights the limited generalizability of XGBoost when trained on specific model generations, especially after preprocessing.

This experiment answers our research question on cross-model robustness of the detectors(Section 1). The detectors are not robust to cross-domain examination with an exception of 'RoBERTa Detector Fine-Tuned on BLOOMZ Generations'. However, fine-tuning improves the models' detection abilities when evaluated on the specific dataset they were trained on.

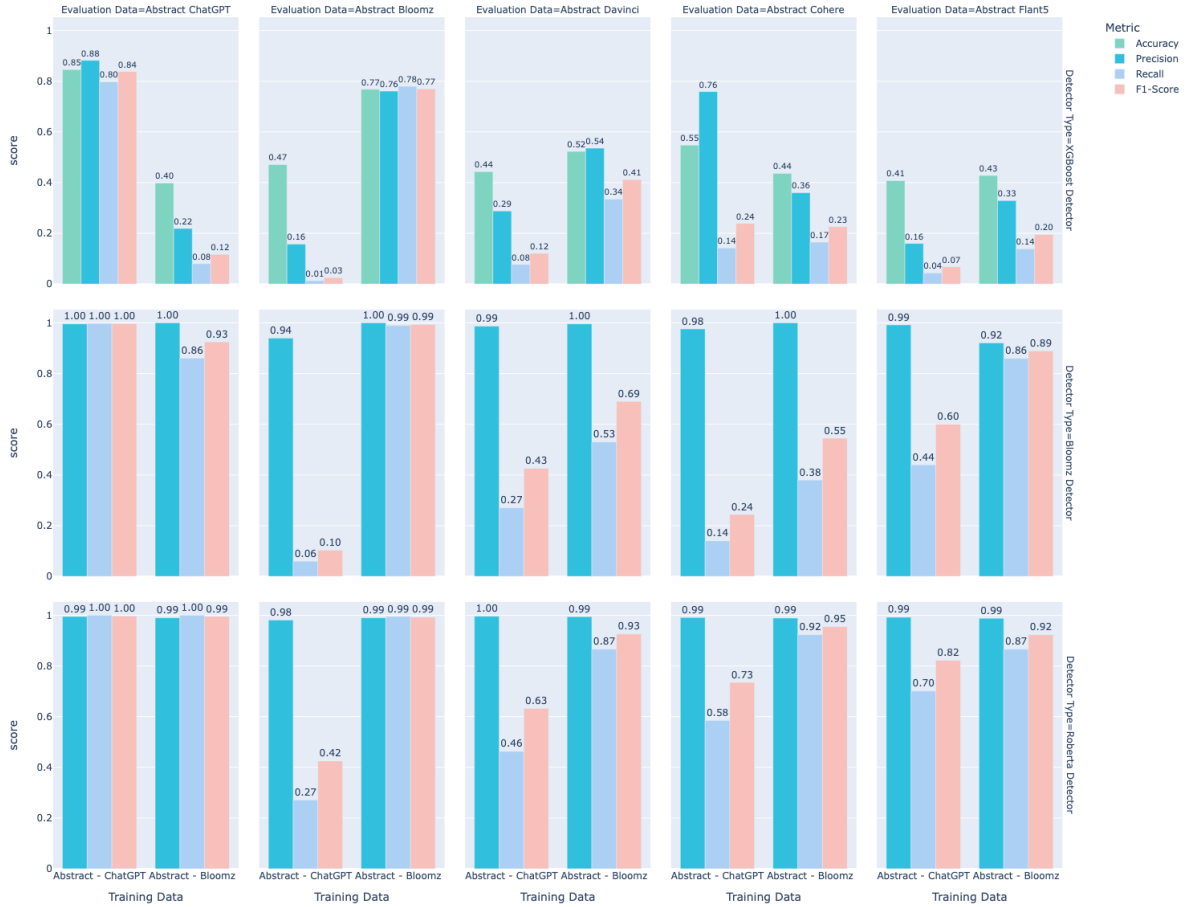


Fig. 6: Evaluation metrics for XGBoost, Bloomz and Robereta detector fine-tuned on the preprocessed M4 Abstract dataset.

5.1 Quality Analysis of Cross-Model evaluation:

In this section, we delve deeper into the qualitative aspects of text features that influence the performance of the XGBoost detector in cross-model evaluation scenarios. Our focus is on understanding how the lin-

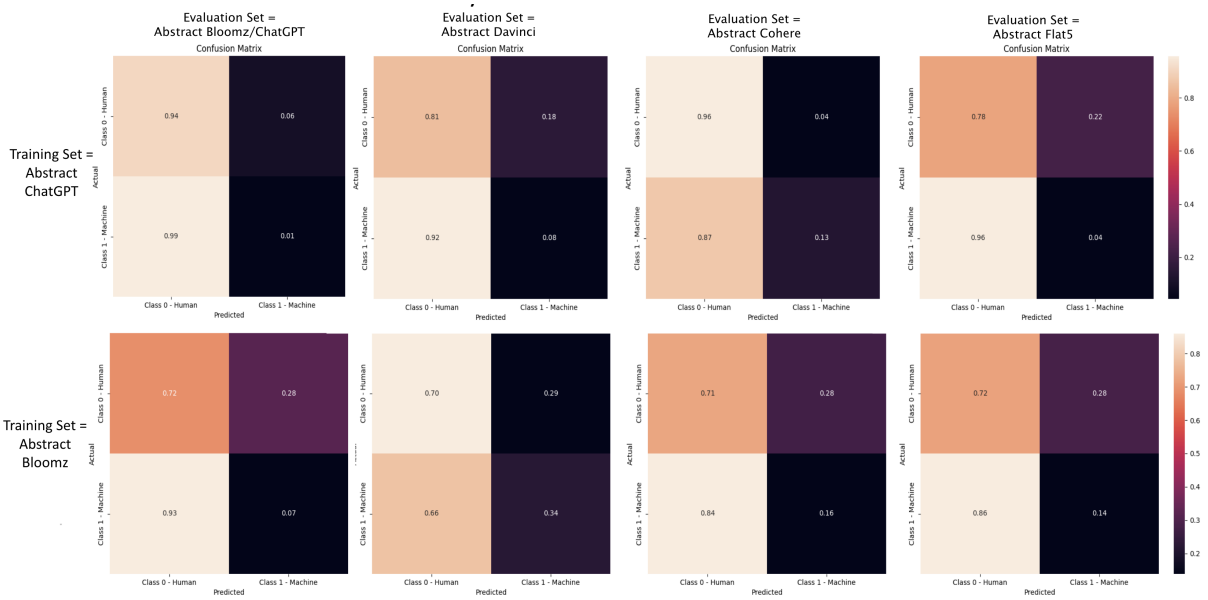


Fig. 7: Confusion matrices for cross-model evaluation of XGBoost detector.

guistic and structural characteristics of AI-generated text impact the detection capabilities of XGBoost, especially when trained on specific data sources and tested across diverse generative models.

To achieve this, we analyze two variants of the XGBoost detector:

- XGBoost trained on ChatGPT Abstracts
- XGBoost trained on BLOOMZ Abstracts

  This analysis provides critical insights into how text features and the choice of training data influence detector performance in diverse scenarios.

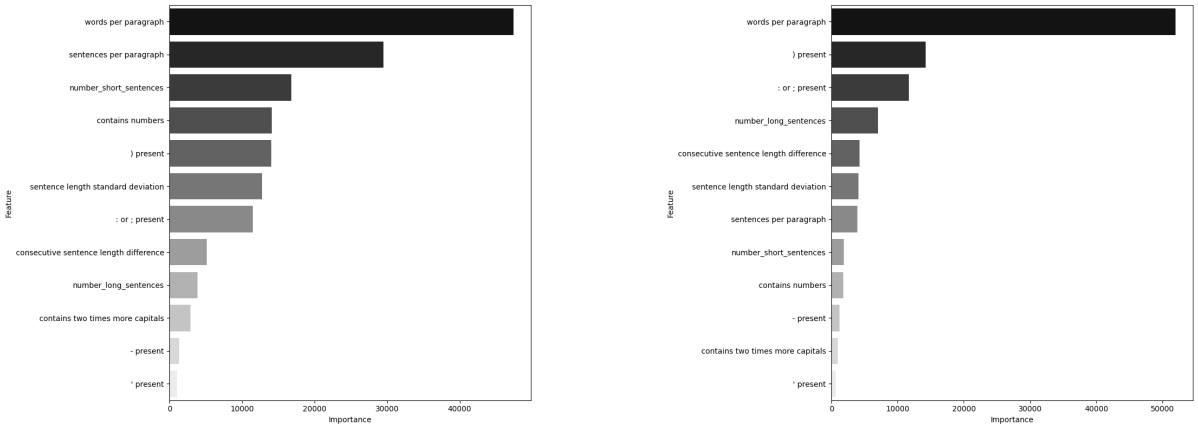
Misclassification of Machine-Generated Text as Human Text The confusion matrices presented in (Figure 7) reveal a recurring trend wherein the XGBoost detector always misclassifies AI-generated text (Class 1) as human text (Class 0). This behavior is consistent across all evaluation sets and both training configurations, indicating a systemic limitation in the detector's ability to accurately identify machine-generated content.

When the XGBoost detector is trained on ChatGPT-generated abstracts, a high false-negative rate is observed, particularly for the evaluation sets derived from BLOOMZ (0.99) and Flat5 (0.96). Although the false-negative rate slightly decreases for evaluation sets from Cohere (0.87) and Davinci (0.92), the detector still struggles to distinguish machine-generated text, misclassifying a significant portion as human text. Similar trends are observed when the detector is trained on BLOOMZ-generated abstracts, with false-negative rates ranging from 0.93 (BLOOMZ evaluation set) to 0.66 (Davinci evaluation set).

Analysis of Feature Importance for XGBoost Detectors: The feature importance visualizations in (Figure 8) highlight notable differences in the factors influencing the classification decision of XGBoost detectors trained on ChatGPT and BLOOMZ-generated abstracts.

- **Dominance of "Words per Paragraph":** For both detectors, "words per paragraph" emerges as the most influential feature, underscoring its critical role in distinguishing machine-generated text from human text. However, the magnitude of its importance varies significantly. This suggests that BLOOMZ-generated abstracts exhibit a stronger reliance on paragraph-level word counts as a stylistic marker compared to ChatGPT-generated abstracts. This also indicates the reduced influence of other features for XGBoost detectors trained on BLOOMZ-generated abstracts.
- **Variation in Secondary Features:** In the ChatGPT-trained detector, "sentences per paragraph" and "number of short sentences" are the second and third most influential features, emphasizing sentence structure as a key discriminator for ChatGPT text. Conversely, the BLOOMZ-trained detector prioritizes punctuation-based features, such as the presence of ")" and ":" or ";", indicating that the BLOOMZ-generated text relies more on specific syntactic patterns.

– **Divergence in Linguistic Patterns:** Features like "consecutive sentence length difference" and "sentence length standard deviation" play a more significant role in the ChatGPT-trained detector.The Bloomz-trained model, in contrast, places relatively lower importance on such sentence-level features, focusing more on simpler metrics like sentence and word counts.



(a) XGBoost trained on ChatGPT abstracts (b) XGBoost trained on Bloomz abstracts

Fig. 8: Comparison of most influencing features for XGBoost detector trained on generations from two different language models

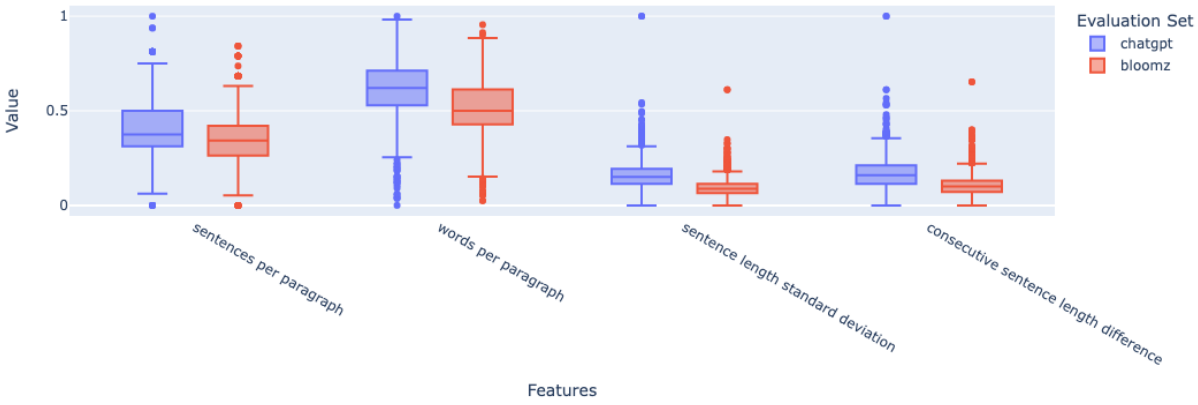
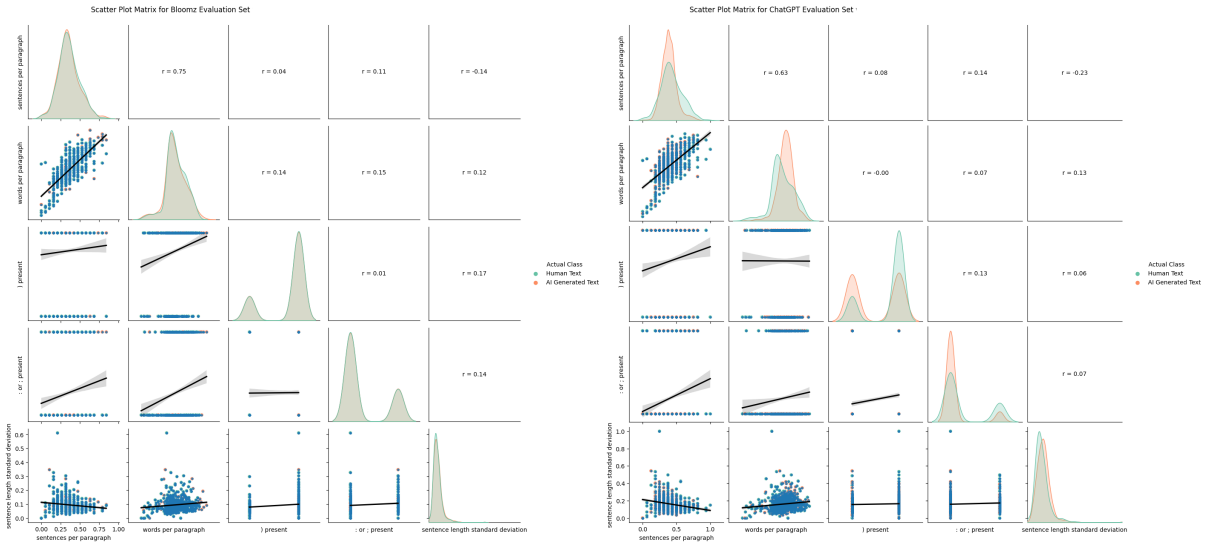


Fig. 9: Boxplot comparing the distribution of influential features in ChatGPT generations Vs Bloomz Generations

Cross-model text style feature disparities: The boxplot visualization in Figure 9 illustrates the distribution of text style features across ChatGPT- and BLOOMZ-generated text.

Distinct differences in feature distributions are evident between the two datasets. For sentences per paragraph, ChatGPT's distribution shows a wider range and higher variance compared to BLOOMZ, which has a tighter spread and lower median. Similarly, the distribution of words per paragraph reveals that ChatGPT exhibits greater variability, while BLOOMZ's output tends to be more uniform and centered around a lower median.

For sentence length standard deviation, ChatGPT-generated text demonstrates higher variability, with a more dispersed range of values compared to BLOOMZ. This suggests that ChatGPT tends to produce text with fluctuating sentence lengths, whereas BLOOMZ-generated sentences are more consistent in length. Additionally, the consecutive sentence length difference metric further emphasizes this



(a) Bloomz Generations (b) ChatGPT Generations

Fig. 10: Scatterplot matrix comparing the distribution of influencing "Machine Text" Vs "AI Generated Text" in ChatGPT and Bloomz generations

divergence, as ChatGPT has a broader range of values and more extreme outliers, while BLOOMZ displays a narrower range with fewer outliers.

These significant stylistic differences indicate that a machine learning model, such as an XGBoost decision detector, trained on features derived from one dataset (e.g., ChatGPT-generated text) may struggle to generalize to the other (e.g., BLOOMZ-generated text). The misalignment in feature distributions can lead to poor detection performance as the decision boundaries learned from one dataset do not adequately capture the characteristics of the other.

- **Limitation of Normalization in Feature Scaling:** One limitation in the feature normalization process involves independently applying Min-Max scaling on training and test sets. When normalization is done separately, the feature ranges in training and test sets can differ. For example, if the training set has a range of 2 to 54 and the test set has a range of 6 to 220 (perhaps due to outliers), the normalized values will not correspond to the same scale. As a result, a value of 1 in the normalized training set could represent a very different original value compared to a value of 1 in the normalized test set. This inconsistency makes comparisons between training and test distributions challenging and can introduce inaccuracies in model evaluation.
- **Justification for Applicability in this Case:** In our analysis, this limitation does not significantly impact the results. All the abstracts analyzed, whether from ChatGPT or Bloomz, are generally constrained within a similar length range due to the nature of academic abstracts. This uniformity minimizes the variability between training and test datasets, reducing the risk of range inconsistencies. Moreover, since the primary goal of our analysis is to compare stylistic patterns rather than absolute feature magnitudes, the effect of any potential discrepancies introduced by separate normalization is negligible in this context.

Analysis of Diagonal in Scatterplot Matrice: The scatterplot matrices in Figure 10 compare key features for ChatGPT-generated text and Bloomz-generated text against human-written text. The diagonals in these matrices represent the distributions of individual features, such as sentence length, words per paragraph, and specific stylistic elements.

In the Bloomz matrix (Figure 10a), the diagonal reveals significant overlap between the distributions of human-written and AI-generated text. This overlap indicates that the stylistic and structural features of Bloomz-generated text are highly coherent with human-written text. Consequently, detectors may struggle to classify Bloomz-generated text based on these features, as they lack distinctive separation.

Conversely, the ChatGPT matrix (Figure 10b) shows less overlap in the distributions along the diagonal. This suggests that the stylistic features of ChatGPT-generated text deviate more from human-

written text. These distinctive patterns make it feasible to utilize these features for classification by detection models.

In Section 1, one of the research aim was to understand limitations of AI-generated text detectors. Qualitative analysis highlights that the stylistic and structural features of text differ significantly across language models like ChatGPT and Bloomz. While some features may effectively distinguish ChatGPT-generated text from human text, these same features may fail for Bloomz-generated text due to closer coherence with human writing. This underscores the limitation of carefully handpicking and tailoring features based on the specific characteristics of each language model's generation style to improve the accuracy and robustness of detection systems.

5.2 Evaluation on Cross-Domain generations

Here are the observations based on the results in (Figure 11) considering the evaluation of detectors trained or fine-tuned on abstract generations but evaluated on Wiki article generations from various LLM models:

- **BLOOMZ Detector fine-tuned on BLOOMZ abstracts:** The BLOOMZ detector fine-tuned on BLOOMZ arXiv generations works well on BLOOMZ Wiki generations (Figure 11a), especially after additional fine-tuning with BLOOMZ Wiki data. However, it is unable to detect Wiki generations from other models, such as ChatGPT or Cohere.
- **BLOOMZ Detector fine-tuned on ChatGPT abstracts:** The BLOOMZ detector fine-tuned on ChatGPT arXiv abstract generations (Figure 11a) does not perform well across any Wiki datasets, including those from BLOOMZ or ChatGPT itself, demonstrating poor transferability between domains and models.
- **RoBERTa Detector:** The RoBERTa detector fine-tuned on ChatGPT arXiv generations (Figure 11b) performs well on ChatGPT Wiki generations but struggles with other models' Wiki data. The RoBERTa detector fine-tuned on BLOOMZ arXiv generations does not perform well for any Wiki generation dataset, highlighting limited cross-domain detection capability.
- **XGBoost Detector:** The XGBoost classifier (Figure 11a) performs poorly across all Wiki generations, suggesting that the model behaves like a fair coin flip or even worse when detecting text generations from different LLMs in a new domain (Wiki), regardless of the training domain (Abstract).

These findings help us answer the cross-model resilience in our research aim (Section 1). Observations indicate that fine-tuning on a specific model's data improves performance on that same model's generation, but there is little generalization across domains.

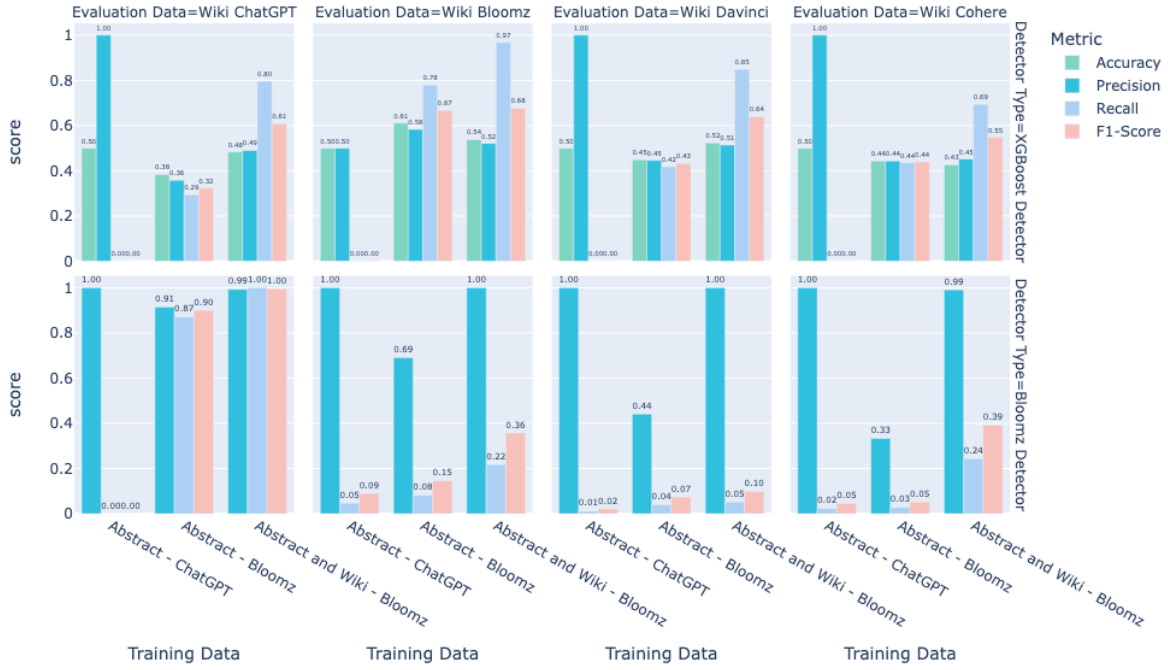
5.3 Evaluating model fine-tuned on generations from multiple-domain

The results in Figure 12 highlight the impact of different training datasets on the performance of both models. When trained on ML Abstracts, both the BLOOMZ detector and XGBoost classifier show their best performance. For the BLOOMZ detector, the training on ML Abstracts achieves near-perfect results with a precision of 1.0, recall of 0.998, and an F1 score of 0.999, indicating its high capability in handling LLaMA-3-generated abstracts within this domain. Similarly, the XGBoost classifier, while not as strong, also performs best on ML Abstracts, with an F1 score of 0.677, showing that domain-specific fine-tuning leads to the most accurate results for both models.

Training on mixed-domain data, such as Abstract, Wiki, and ML Abstract, shows more balanced but slightly lower performance. The BLOOMZ detector achieves a recall of 0.74 and an F1 score of 0.563, reflecting its inability to generalize across multiple domains. The XGBoost classifier, while achieving a perfect recall of 1.0 on this dataset, has lower precision (0.5), indicating that the model becomes more sensitive but less specific when trained on diverse data sources.

Models trained on ChatGPT-generated Abstracts and BLOOMZ-generated Abstracts show the weakest performance. For the BLOOMZ detector, training on ChatGPT yields a lower recall (0.66) and F1 score (0.765), while training on BLOOMZ leads to the lowest F1 score of 0.401. Similarly, for the XGBoost classifier, training on ChatGPT results in the lowest F1 score of 0.221, with poor recall (0.148), and training on BLOOMZ also delivers suboptimal results with an F1 score of 0.375.

This further helps understand our research interest in cross-domian examination (Section 1). Observations suggests that fine-tuning on multi-domain patters degrades both in-domain as well as cross-domain performance of the detector.



(a) XGBoost Detector and Bloomz Detector

(b) RoBERTa Detector

Fig. 11: Cross-Domain Evaluation metrics for XGBoost, Bloomz and Robereta detector fine-tuned on the preprocessed M4 Abstract dataset.

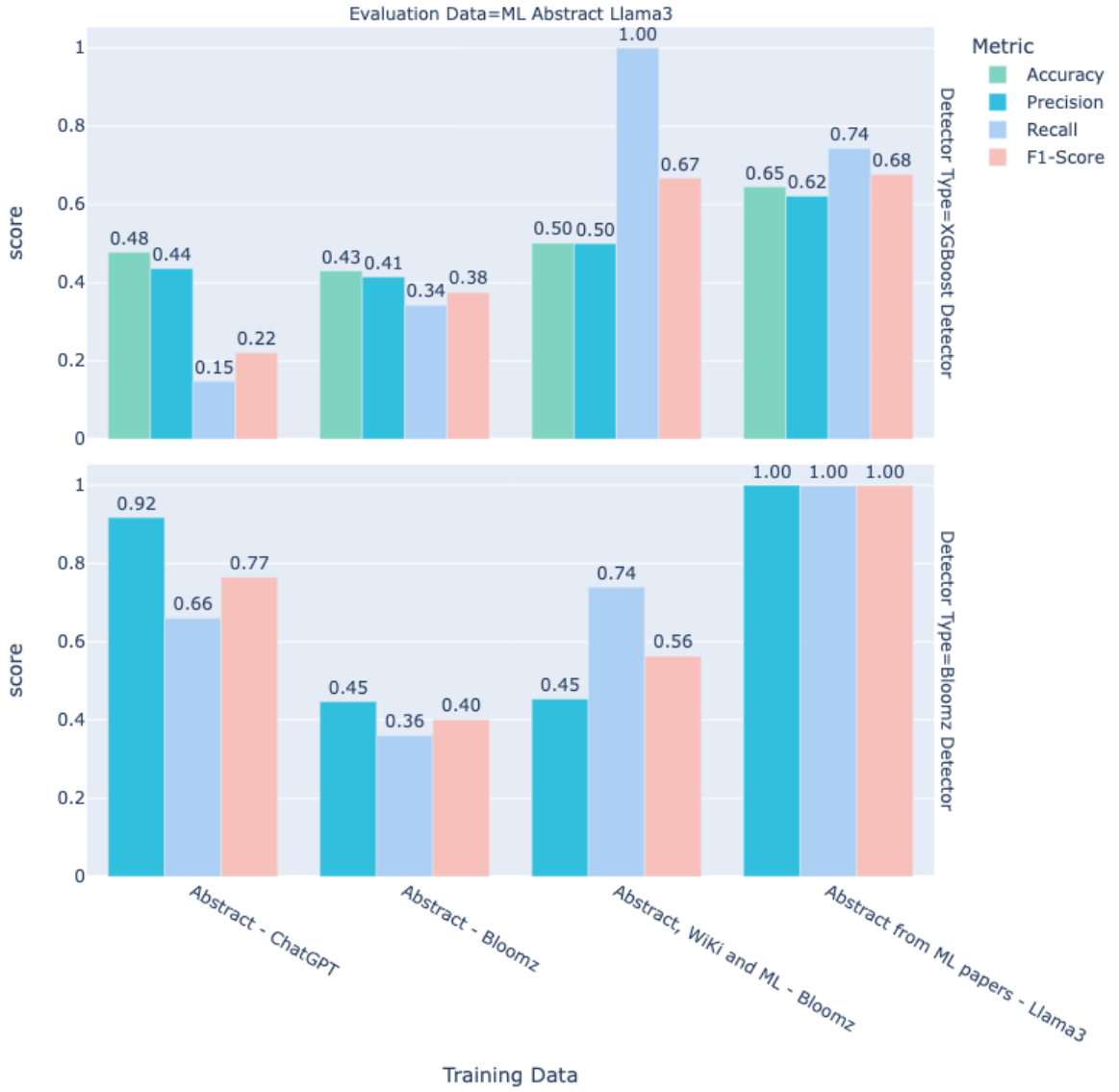
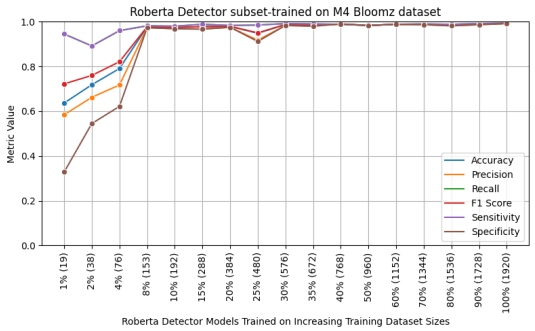
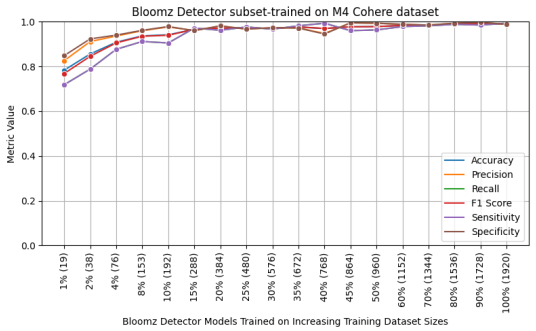


Fig. 12: Cross-Domain Evaluation metrics for XGBoost, Bloomz and Robereta detector fine-tuned on multiple domains.



(a) Performance of RoBERTa detector subset-trained on M4 BLOOMZ Abstract dataset. The ticks on x-axis is the number of subset-records for each model



(b) Performance of BLOOMZ detector subset-trained on M4 Cohere Abstract dataset. The ticks on x-axis is the number of subset-records for each model

Fig. 13: Subset Training

5.4 Evaluating model fine-tuned on subset of training data

Based on the plots in Figure 13, which evaluate the RoBERTa and BLOOMZ detectors trained on subsets of generative data, we can observe the following trends:

RoBERTa Detector (Figure 13a):

- Initially, performance increases sharply as more data is added, with significant improvements seen between 19 records to 153 records of the data. For example, accuracy, precision, recall, and F1 score all rise steeply within this range.
- Beyond 153 records, the performance metrics (accuracy, precision, recall, F1 score, sensitivity, and specificity) stabilize, indicating diminishing returns from additional data.
- This suggests that RoBERTa can effectively learn the patterns and style of the new data with as little as 153 of the available training data, beyond which additional data provides minimal benefit. There is a noticeable slight dip in performance at 480 records of the dataset size, particularly in metrics such as recall and sensitivity. This dip could be attributed to a temporary overfitting or instability as the model adjusts to the added data. It is common for models to exhibit such fluctuations when new subsets of data introduce more variability or complexity, leading to a short-term decline in performance before stabilizing again.

BLOOMZ Detector (Figure 13b):

- Similar to the RoBERTa detector, the BLOOMZ detector shows rapid improvements in performance up to 153 records size dataset. Accuracy, precision, recall, and other metrics improve significantly within this range.
- From 192 records size to 480 records size, the performance metrics continue to increase but at a slower rate. After 480 records size, the curves largely plateau, indicating that the model has learned the necessary patterns. By 672 records size and beyond, the model achieves near-maximum performance across all metrics, such as F1 score, sensitivity, and specificity.
- This indicates that the BLOOMZ detector requires only a moderate amount of data 192-288 records to capture the style of the new generative data, and adding more data beyond this point yields limited additional improvement.

In summary, both detectors show significant performance gains with $\tilde{1}53$ to 288 training data from a new generative pattern, after which the benefit of adding more data diminishes.

5.5 Adversarial Attacks

In Figure 14, we evaluate the performance of the RoBERTa detector under various adversarial attack types, including *deepwordbug*, *pruthi*, *pwws*, and *textfooler*. A significant drop in accuracy can be observed under some attack scenarios:

- The **accuracy** of the RoBERTa detector without attack is consistently at 1.0 across all attack types. However, under attack, the accuracy significantly decreases, particularly for *textfooler* (0.4) and *deepwordbug* (0.65). This highlights the vulnerability of the detector to these adversarial methods.
- For **pwws**, accuracy under attack remains relatively higher at 0.75, though still lower than without attack, whereas *pruthi* exhibits no drop in accuracy (1.0 under attack).
- The **average perturbed word** metric shows varying levels of perturbation for different attack types. *textfooler*, which perturbs 12.22% of the words, causes the largest decrease in accuracy, while *pruthi* perturbs only 1.0% of words and does not affect accuracy.

One additional observation is that while these attacks significantly impact model performance, the **perturbed text** is often not close to realistic language usage, limiting the applicability of such tests. These adversarial methods introduce nonsensical changes to the text, making it less representative of real-world adversarial inputs, where the modified text is typically expected to remain semantically coherent and grammatically correct.

For example, the adversarial attacks shown in the Figure 15 demonstrate how subtle perturbations in text can significantly alter the meaning and context. For example, in the pwws attack, the word "present" in "The present paper focuses on..." is changed to "acquaint," which confuses the subject and

context of the paper. Similarly, in the deepwordbug attack, a critical term like "3PN-accurate" is altered to "3Pd-accurate," introducing an incorrect and meaningless scientific reference.

These changes disrupt the original intent and create sentences that are far from the intended meaning, raising questions about the practical value of such adversarial attacks when they generate unrealistic and nonsensical text. Thus, while the results demonstrate the vulnerability of the detector, the nature of the perturbations suggests that these tests may not fully reflect practical adversarial scenarios.

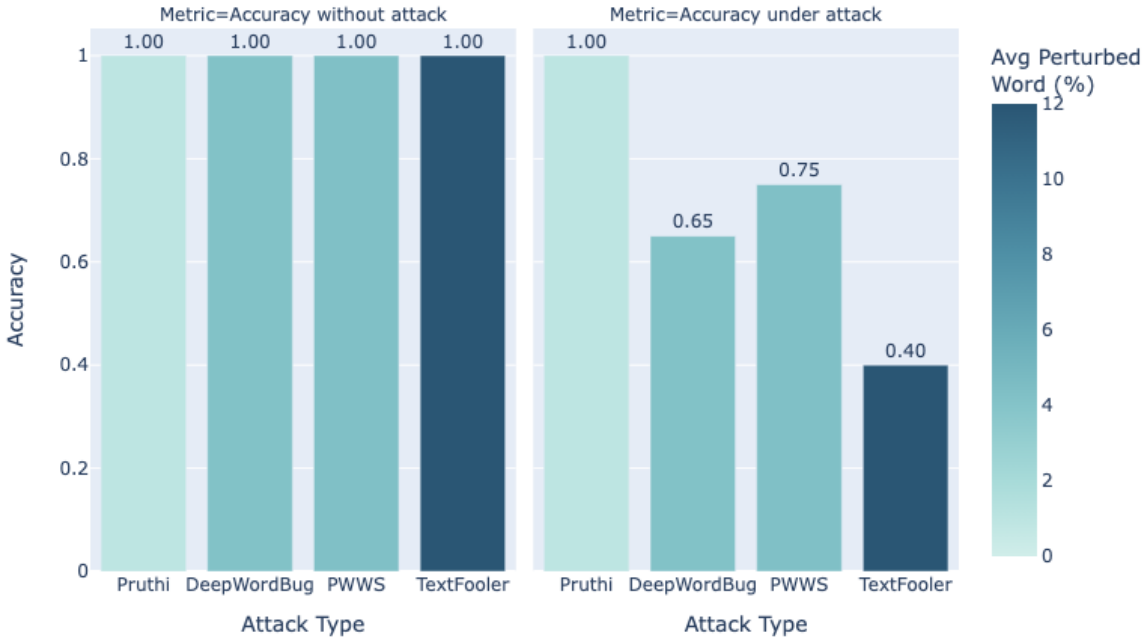


Fig. 14: Evaluation metrics for XGBoost, Bloomz and Robereta detector fine-tuned on the preprocessed M4 Abstract dataset.

5.6 Prompt Attacks

Table 6 shows the performance of the RoBERTa detector under various types of prompt attacks using Gemini-1.5-Flash. These attacks include synonyms, informal language or slang, sentence restructuring, idiomatic expressions, and grammar mistakes. The RoBERTa detector performs exceptionally well across all metrics—Precision, Recall, Accuracy, and F1 Score—scoring 1.0 in most categories. However, it shows a slight decrease in recall (0.86), accuracy (0.93), and F1 Score (0.92) for "Informal language or slang," indicating a minor vulnerability in detecting such prompt modifications compared to the other attack types, where it maintains perfect scores.

This suggests that while the RoBERTa model is highly robust to simple linguistic changes, its detection of informal or slang language may warrant further attention for full robustness.

6 Conclusion

In conclusion, this study provides a comprehensive evaluation of three detectors—RoBERTa, BLOOMZ, and XGBoost—on the M4 dataset, answering our first research question in Section 1 on robustness and resiliance of the detectors. The sensitivity of the detectors to formatting artifacts such as LaTeX code and non-printable characters highlights the importance of preprocessing. Once these artifacts were removed, the true limitations of the models became apparent, underscoring the need for detectors that rely on deeper linguistic and stylistic features rather than superficial formatting cues.

original_text	perturbed_text
We address the problem of constructing high-accuracy, faithful analytic waveforms describing the gravitational wave signal emitted by inspiralling and coalescing binary black holes. We work within the Effective-One-Body (EOB) framework and propose a methodology for improving the current (waveform)implementations of this framework based on understanding, element by element, the physics behind each feature of the waveform, and on systematically comparing various EOB-based waveforms with ``exact'' waveforms obtained by numerical relativity approaches. The present paper focuses on small-mass-ratio non-spinning binary systems, which can be conveniently studied by Regge-Wheeler-Zerilli-type methods. Our results include: (i) a resummed, 3PN-accurate description of the inspiral waveform, (ii) a better description of radiation reaction during the plunge, (iii) a refined analytic expression for the plunge waveform, (iv) an improved treatment of the matching between the plunge and ring-down waveforms. This improved implementation of the EOB approach allows us to construct complete analytic waveforms which exhibit a remarkable agreement with the ``exact'' ones in modulus, frequency and phase. In particular, the analytic and numerical waveforms stay in phase, during the whole process, within $\pm 1.1 %$ of a cycle. We expect that the extension of our methodology to the comparable-mass case will be able to generate comparably accurate analytic waveforms of direct use for the ground-based network of interferometric detectors of gravitational waves.	We address the problem of constructing high-accuracy, faithful analytic waveforms describing the gravitational wave signal emitted by inspiralling and coalescing binary Negro holes. We work within the Effective-One-Body (EOB) framework and propose a methodology for improving the current (waveform)implementations of this framework based on understanding, element by element, the physics behind each feature of the waveform, and on systematically comparing various EOB-based waveforms with ``exact'' waveforms obtained by numerical relativity approaches. The acquaint paper focuses on small-mass-ratio non-spinning binary systems, which can be conveniently studied by Regge-Wheeler-Zerilli-type methods. Our results include: (i) a resummed, 3PN-accurate description of the inspiral waveform, (ii) a better description of radiation reaction during the plunge, (iii) a refined analytic expression for the plunge waveform, (iv) an improved discussion of the oppose between the plunge and ring-down waveforms. This improved implementation of the EOB approach allows us to build stark analytic waveforms which exhibit a remarkable understanding with the ``exact'' ones in modulus, frequency and phase. In particular, the analytic and numerical waveforms stay in phase, during the hale process, within $\Pm 1.1 %$ of a cycle. We expect that the extension of our methodology to the comparable-mass event will be able to give comparably accurate analytic waveforms of direct use for the ground-based network of interferometric detectors of gravitational waves.

(a) Example of pwws attack

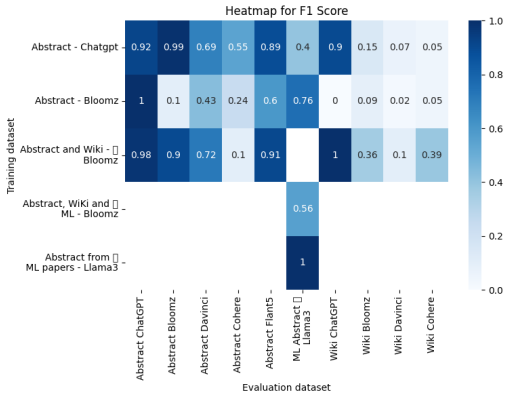
original_text	perturbed_text
We address the problem of constructing high-accuracy, faithful analytic waveforms describing the gravitational wave signal emitted by inspiralling and coalescing binary black holes. We work within the Effective-One-Body (EOB) framework and propose a methodology for improving the current (waveform)implementations of this framework based on understanding, element by element, the physics behind each feature of the waveform, and on systematically comparing various EOB-based waveforms with ``exact'' waveforms obtained by numerical relativity approaches. The present paper focuses on small-mass-ratio non-spinning binary systems, which can be conveniently studied by Regge-Wheeler-Zerilli-type methods. Our results include: (i) a resummed, 3PN-accurate description of the inspiral waveform, (ii) a better description of radiation reaction during the plunge, (iii) a refined analytic expression for the plunge waveform, (iv) an improved treatment of the matching between the plunge and ring-down waveforms. This improved implementation of the EOB approach allows us to construct complete analytic waveforms which exhibit a remarkable agreement with the ``exact'' ones in modulus, frequency and phase. In particular, the analytic and numerical waveforms stay in phase, during the whole process, within $\pm 1.1 %$ of a cycle. We expect that the extension of our methodology to the comparable-mass case will be able to generate comparably accurate analytic waveforms of direct use for the ground-based network of interferometric detectors of gravitational waves.	We address the problem of constructing high-accuracy, faithful analytic waveforms describing the gravitational wave signal emitted by inspiralling and coalescing binary black holes. We work within the Effective-One-Body (OB) framework and propose a methodology for improving the current (wavform)implementations of this framework based on understanding, element by element, the physics behind each feature of the waveform, and on systematically comparing various EOB-baesd waveforms with ``exact'' waveforms obtained by numerical relativity approaches. The present paper focuses on small-mass-ratHo non-pinning binary systems, which can be conveniently studied by Regge-Wheeler-Berilli-type methods. Our results include: (i) a resummed, 3Pd-accurate description of the inspirTal waveform, (I) a better description of radiation reaction during the lunge, (iii) a refined anafytic expression for the plunge waveform, (iv) an improved treatment of the matching between the plunge and ring-down waveforms. TThis improved implementation of the EuB approach allows us to construct compelte analytic waveforms which exhibit a remarkable agreement with the ``exact'' ones in modulus, frequency and phase. In particular, the analytic and numerical waveforms stay in phase, during the hwole process, within $\pl 1.1 %$ of a cycle. We expect that the extension of our methodology to the comarable-mass case will be ble to generate comparably accurate analytic waveforms of direct use for the ground-based network of interfermoetric detectors of gravitational waves.

(b) Example for deepwordbug attack

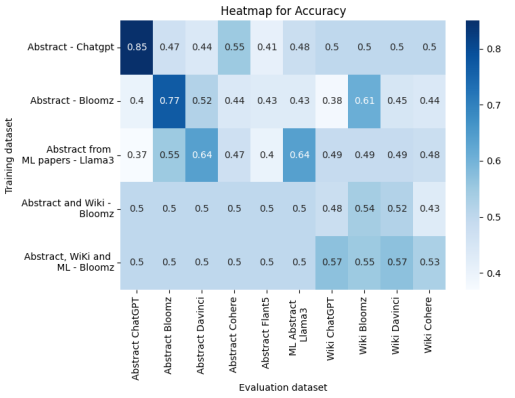
Fig. 15: Example of original text and perturbed text for adversarial attacks

RoBERTa Detector				
Prompt Attack Using Gemini-1.5-Flash \ Metric	Precision	Recall	Accuracy	F1 Score
Synonyms	1.0	1.0	1.0	1.0
Informal language or slang	1.0	0.86	0.93	0.92
Sentence restructuring	1.0	1.0	1.0	1.0
Idiomatic expressions	1.0	1.0	1.0	1.0
Grammar mistakes	1.0	1.0	1.0	1.0

Table 6: Performance of RoBERTa detector under various Prompt attacks using Gemini-1.5-Flash



(a) F1 Score heatmap for BLOOMZ based AI generated text detectors



(b) Accuracy heatmap for XGBoost Classifier based AI generated text detectors

Fig. 16: Heatmap depicting the evaluation metrics for BLOOMZ(a) and XGBoost(b) based AI generated text detectors

The results demonstrate that while fine-tuning on specific model generations improves detection accuracy within the same model, the detectors struggle to generalize cross-model and cross-domain scenarios. Notably, both the BLOOMZ and RoBERTa detectors, after fine-tuning, achieved high F1 scores when evaluated on the datasets they were trained on, but their performance decreased significantly for cross-model or cross-domains. Similarly, the XGBoost detector, though performing well within the trained domain, exhibited poor cross-model and cross-domain generalizability. Despite the improvements seen with fine-tuning, the detectors' performance remained highly dependent on the specific training data, questioning the questioning robustness of the detectors.

When applying adversarial attacks, the detectors demonstrated a decline in accuracy. These attacks effectively exposed weaknesses in the detector models' robustness, particularly in detecting adversarially crafted text. Similarly, prompt-based attacks, where a new generation model was guided by variations in prompt construction, resulted in degraded performance. The model misclassified many generations that maintained the original intent but varied stylistically, showing that prompt variability introduces new challenges for detectors.

Additionally, the second research aim in Section 1 is studied to experimentally estimated that fine-tuning existing detectors with an additional data of approximately 153-288 records(which was 8-15% of our training set of 1920 records) representing newer patterns or styles can significantly improve performance, particularly in cross-model.

Overall, the findings suggest that while current detectors can effectively identify AI-generated text under controlled conditions, their limited generalizability and adaptability points to significant challenges in robustness and resilience of AI-generated text detectors, highlighting the need for future work on more versatile and robust detection methods.

Limitations

- **Limited Evaluation of Prompt Attacks:** While our study evaluates the resilience of detectors against adversarial attacks, the analysis of prompt-based attacks remains limited.
- **No Qualitative Analysis for LLM-Based Detectors:** Our qualitative analysis was restricted to the XGBoost detector. We did not conduct in-depth explainable AI (XAI) evaluations for LLM-based detectors like RoBERTa and BLOOMZ.
- **Dataset Limitations:** The dataset used for this study is based on M4 and its extensions, which do not include text generated by the latest versions of large language models.
- **Use of Limited Detector Architectures:** Our evaluation is restricted to two types of detectors: LLM-based detectors (e.g., RoBERTa, BLOOMZ) and XGBoost. Incorporating a broader variety of detectors could have provided a more comprehensive understanding of detection capabilities.
- **Compute Constraints on Detector Parameters:** Due to limited computational resources, we tested lower-parameter versions of detectors. This restriction might have impacted their overall performance.

Future Work

- **Incorporation of Handcrafted Features:** Future studies should explore integrating syntactic, semantic, and statistical features into detector architectures to enhance the cross-model generalizability of detectors.
- **Strategies for Improved Cross-Domain Generalization:** Developing strategies, such as domain adaptation techniques or domain-invariant feature extraction, could improve detector robustness in real-world applications.
- **Exploration of Ensemble Learning:** Combining multiple detectors into an ensemble framework could be a promising direction for enhancing robustness and accuracy.
- **Evaluation Against Latest LLM Generations:** Future work should include detectors trained and tested on text generated by the latest LLMs to reflect the current state of the art.

By addressing these limitations and pursuing the outlined future directions, the detection of AI-generated text can be significantly improved, particularly in handling the diverse and evolving landscape of LLM outputs.

References

[1] Heather Desaire, A. E. Chua, M.-G. Kim, and D. Hua. "Accurately detecting AI text when ChatGPT is told to write like a chemist". In: *Cell Reports Physical Science* 4.101672 (2023). DOI: 10.1016/j.xcrp.2023.101672.

[2] Saranya Venkatraman, Adaku Uchendu, and Dongwon Lee. "GPT-who: An Information Density-based Machine-Generated Text Detector". In: *Findings of the Association for Computational Linguistics: NAACL 2024*. Ed. by Kevin Duh, Helena Gomez, and Steven Bethard. Mexico City, Mexico: Association for Computational Linguistics, June 2024, pp. 103–115. DOI: 10.18653/v1/2024.findings-naacl.8. URL: https://aclanthology.org/2024.findings-naacl.8.

[3] Vinu Sankar Sadasivan, A. Kumar, S. Balasubramanian, W. Wang, and S. Feizi. "Can AI-generated text be reliably detected?" In: *arXiv preprint arXiv:2303.11156v3* (2023). URL: https://arxiv.org/abs/2303.11156v3.

[4] Junchao Wu, Shu Yang, Runzhe Zhan, Yulin Yuan, Derek F. Wong, and Lidia S. Chao. "A survey on LLM-generated text detection: Necessity, methods, and future directions". In: *arXiv preprint arXiv:2310.14724v2* (2023). URL: https://arxiv.org/abs/2310.14724v2.

[5] Debora Weber-Wulff, Alla Anohina-Naumeca, Sonja Bjelobaba, Tomáš Foltýnek, Jean Guerrero-Dib, Olumide Popoola, Petr Šigut, and Lorna Waddington. "Testing of detection tools for AI-generated text". In: *International Journal for Educational Integrity* 19.26 (2023). DOI: 10.1007/s40979-023-00146-z.

[6] Chenghao Zhou, Yao Wang, and Jiahao Liu. "Adversarial Attacks on AI Text Detectors: A Study on Robustness". In: *arXiv preprint arXiv:2404.01907v1* (2023). URL: https://arxiv.org/abs/2404.01907.

[7] Wei Huang, Bing Liu, and Shuming Zhang. "Siamese Calibrated Reconstruction Network for Robust AI Text Detection". In: *arXiv preprint arXiv:2406.01179v2* (2023). URL: https://arxiv.org/abs/2406.01179.

[8] Heather Desaire, Aleesa E Chua, Madeline Isom, Romana Jarosova, and David Hua. "Distinguishing academic science writing from humans or ChatGPT with over 99% accuracy using off-the-shelf machine learning tools". In: *Cell Reports Physical Science* 4.6 (2023).

[9] Andreas Bentzen Winje and Nicolai Sivesind. "Turning Poachers into Gamekeepers: Detecting Machine-Generated Text in Academia Using Large Language Models". PhD thesis. June 2023.

[10] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL]. URL: https://arxiv.org/abs/1907.11692.

[11] Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. *Crosslingual Generalization through Multitask Finetuning*. 2023. arXiv: 2211.01786 [cs.CL]. URL: https://arxiv.org/abs/2211.01786.

[12] John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. "Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp". In: *arXiv preprint arXiv:2005.05909* (2020).

[13] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. "Black-box generation of adversarial text sequences to evade deep learning classifiers". In: *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE. 2018, pp. 50–56.

[14] Danish Pruthi, Bhuwan Dhingra, and Zachary C Lipton. "Combating adversarial misspellings with robust word recognition". In: *arXiv preprint arXiv:1905.11268* (2019).

[15] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. "Generating natural language adversarial examples through probability weighted word saliency". In: *Proceedings of the 57th annual meeting of the association for computational linguistics*. 2019, pp. 1085–1097.

[16] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. "Is bert really robust? a strong baseline for natural language attack on text classification and entailment". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 05. 2020, pp. 8018–8025.

[17] Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, and et al. "M4: Multi-Generator, Multi-Domain, and Multi-Lingual Black-Box Machine-Generated Text Detection". In: *arXiv preprint arXiv:2305.14902* (2023). URL: https://arxiv.org/abs/2305.14902.

[18] Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. "Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense". In: *Advances in Neural Information Processing Systems* 36 (2024).

[19] Marcin Kardas, Piotr Czapla, Pontus Stenetorp, Sebastian Ruder, Sebastian Riedel, Ross Taylor, and Robert Stojnic. "Axcell: Automatic extraction of results from machine learning papers". In: *arXiv preprint arXiv:2004.14356* (2020).