ToED: Tree of thoughts with Two-tier Prompts for Evading AI-Generated Text Detection

Anonymous ACL submission

Abstract

AI-generated text (AIGT) detection evasion aims to reduce the detection probability of AIGT, helping to identify weaknesses in detectors and enhance their effectiveness and reliability in practical applications. With the increasing demand for AIGT detection in recent years, evasion techniques have gradually become a prominent research focus. Previous evasion methods have relied on manually crafted modification strategies, such as the selection of replacement words and hand-designed examples, which requires expert domain knowledge. To address above limitations, we propose the Tree of Evading Detection (ToED). ToED employs a two-tier mixed prompt to construct a tree structure that guides Large Language Models (LLMs) in autonomously exploring optimal modification strategies, thereby enhancing the ability of AIGT to evade detection. Experimental results demonstrate that our method effectively reduces the average detection accuracy of various AIGT detectors across texts generated by different LLMs, surpassing that of three other baselines and achieving the best performance in evading detection.

1 Introduction

The rapid development of large language models (LLMs), such as GPT-4 (OpenAI, 2023), Qwen (Bai et al., 2023), Mistral (Jiang et al., 2023) and LLaMa2 (Touvron et al., 2023), has greatly enhanced the ability to generate high-quality human-like text. As powerful tools for optimizing content creation, LLMs are widely applied across various fields, including journalism, academia, social media, and more. However, these advancements have also sparked ethical concerns regarding their inherent risks, such as academic dishonesty (Wu et al., 2023; Zeng et al., 2024), fake news (Su et al., 2024; Hu et al., 2024) and false comments (Mireshghallah et al., 2024). To build the first barrier against such threats, developing effective techniques for detecting AI-generated text (AIGT) has become an imperative need.

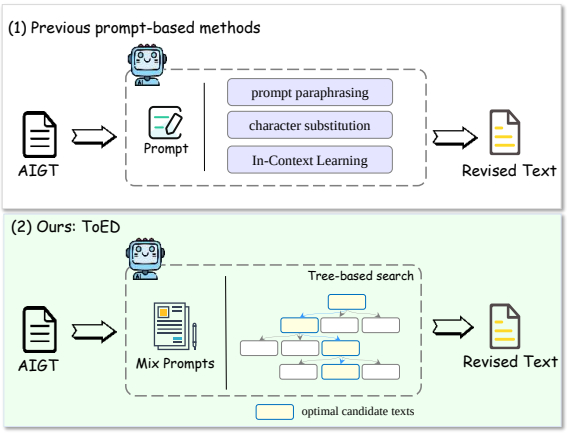

Figure 1: The comparison of our proposed ToED with exiting prompt-based detection evasion methods. (1) Previous prompt-based methods for evading detection primarily relied on designing prompts to guide LLMs to generate results directly. These simple prompts are often insufficient for helping LLMs understand how to modify the given text, leading to poor performance. (2) The proposed ToED constructs a tree structure with mixed prompts, which helps LLMs understand task requirements and guides them to progressively search for a satisfactory modification strategy.

Existing AIGT detection methods can be broadly categorized into two types. The first category, statistical-based methods, detects AIGT by leveraging various statistical differences between AIGT and human-written text (HWT), including entropy, rank, and conditional probability (Gehrmann et al., 2019; Mitchell et al., 2023; Ma and Wang, 2024). The second category, classifier-based methods, involves training classifiers on large datasets that contain both AIGT and human-written text to perform the detection (Yu et al., 2024; Yu et al., 2023; Huang et al., 2024; Zhou et al., 2024b).

While AIGT detection has demonstrated promising performance in many scenarios, recent findings suggest that many of them remain fragile under ad-

versarial conditions. To determine vulnerabilities in AI detectors before deploying them in real-world applications, several studies (Krishna et al., 2023; Zhou et al., 2024a) have explored various evasion techniques. These methods aim to decrease the detection probability of given texts by modifying the text. For instance, Krishna et al. (2023) conducted paraphrasing attacks by fine-tuning T5 (Raffel et al., 2020), and Zhou et al. (2024a) proposed a word importance ranking strategy to determine substitute words and utilized a masked language model for synonym substitution. Although these methods can effectively evade detection by AIGT detectors, they heavily rely on manual strategies manually crafted modification strategies, such as model training approaches and the identification of critical words for replacement, which require expert domain knowledge.

Given the excellent performance of LLMs in recent downstream natural language processing tasks, some researchers have begun exploring a new paradigm for evading detection using LLMs, which involves carefully designing prompts to guide the model in directly generating detection-resistant text. A resent study by Wang et al. (2024a) attempted to use three types of prompts to guide LLMs in generating text capable of evading detection. These prompt included prompt paraphrasing, in-context learning (ICL), and character substitution. While their approach introduced a novel angle via prompt engineering, their empirical results showed that such prompts underperformed traditional evasion strategies, as demonstrated by experimental results in Table 2 in Wang et al. (2024a). This highlights a key challenge in prompt-based evasion: without goal-oriented guidance, LLMs often lack a clear objective or direction for modification when relying solely on simple and straightforward prompts. To further improve the performance of prompt-based evasion methods, Lu et al. (2023) proposed substitution-based in-context example optimization method (SICO), which effectively leverages the ICL ability of LLMs to improve performance in evading detection. However, this approach relies on sentences modified by traditional methods as in-context examples, which means it still inherits the limitations of conventional evasion techniques, such as the need for expert knowledge.

To address the above challenges, we aim to develop an effective text modification strategy that requires no human intervention, improving the efficiency and performance of AIGT detection evasion task. Therefore, we propose the Tree of Evading Detection (ToED). Inspired by the Tree of Thought (ToT) reasoning (Yao et al., 2023), ToED aims to construct a tree structure that guides LLMs in autonomously exploring optimal modification strategies. To this end, we design two-tire mixed prompts, consisting of an ICL-based low-level prompt and a feedback-based high-level prompt, to guide the LLMs in effectively building the tree and searching for the optimal modification strategy. Specifically, the original generated text serves as the root node, with different child nodes representing candidate texts generated by making different modifications to the parent node. Each branch, consisting of several modifications, represents a specific modification strategy. The process begins with the construction of the ICL-based low-level prompt to generate candidate texts with LLM. Meanwhile, the detection probability of each candidate text is yielded by a proxy detector. Based on the probabilities, we then design a feedback-based high-level prompt to guide LLM in modifying a specific candidate text, reducing its detection probability. As shown in Figure 1, unlike the flat, one-step prompting strategies used in prior work, ToED adopts a hierarchical prompt tree that enables LLMs to iteratively refine salient features during the evasion process, leading to more effective detection evasion.

We evaluate the performance of the proposed ToED against four AIGT detectors on various datasets. Experimental results demonstrate that the performance of ToED surpasses that of three other baselines, achieving the best evasion detection performance. Our main contributions can be summarized as follows:

- We propose Tree of Evading Detection (ToED), a tree-structured framework that enables LLMs to simultaneously explore diverse candidates and iteratively refine evasion strategies, leading to broader coverage of the modification space in prompt-based evasion.
- We design a two-tier mixed prompt to guide LLMs in effectively building the tree and searching for the optimal modification strategy, containing an ICL-based low-level prompt for initial rewriting and a feedback based high-level prompt that steers the search toward low-detectability candidates.
- Experimental results demonstrate that our pro-

posed method effectively evades detection by four detectors, including chatgpt-detector (Guo et al., 2023), RADAR (Hu et al., 2023), GLTR (Gehrmann et al., 2019) and Fast-detectGPT (Bao et al., 2024). It reduces the average detection accuracy of these detectors for text generated by different LLMs to 47.13%.

2 Related Work

2.1 AIGT detection methods

Mitchell et al. (2023) proposed a text perturbation method to measure the log probabilities difference between original and perturbed texts. Su et al. (2023) proposed a zero-shot method that measures the log-probability difference between original text and perturbed text using text perturbation techniques, significantly improving AIGT detection performance. Venkatraman et al. (2024) argued that humans tend to evenly distribute information during language production, whereas AI-generated text may lack this uniformity. Therefore, they introduced uniform information density features to quantify the smoothness of token distribution, aiding in the identification of AI-generated text. These methods detected AIGT by leveraging various statistical differences between AIGT and human-written text.

Additionally, Tian et al. (2024) introduced a length-sensitive multiscale positive-unlabeled loss, which enhanced the detection performance for short texts while maintaining the detection efficacy for long texts. These methods achieve strong detection performance in detecting datasets belonging to the same domain as the training set, but usually fail when faced with datasets that are not in the domain of the training set. To enhance the generalization ability of model in known target domains, Verma et al. (2024) proposed Ghostbuster, a method that processes documents through a series of weaker language models, conducted a structured search over possible combinations of their features, and then trained a classifier on the selected features to predict whether the documents were AI-generated. These methods involved training models on large labeled datasets to detect AIGT. Existing detectors are capable of achieving high detection performance, which places higher demands on evasion detection methods.

2.2 Detection evasion methods

To reveal vulnerabilities in AI detectors before they are deployed in real-world applications, Krishna *et al.* (2023) conducted a paraphrasing attack method named discourse paraphraser (DIPPER). This method fine-tuned T5 (Raffel et al., 2020) by aligning, reordering, and calculating control codes, enabling the model to perform diverse modifications and paraphrasing of text without altering its original meaning. Zhou et al. (2024a) proposed humanizing machine-generated content (HMGC). This method first trained a surrogate model based on the output of the victim detector to simulate the victim detector. Using adversarial detection methods, it modified the text to induce errors in the detection of proxy model. HMGC calculated the importance score for each token, ranked them to determine the candidate tokens for modification, and employed a masked language model to perform synonym replacement on the selected tokens. To leverage the capabilities of LLMs, Lu et al. (2023) proposed the substitution-based in-context example optimization (SICO) method. This method guide the model to generate more human-like text by using samples that have undergone synonym replacement and paraphrasing as in-context examples. Although these methods achieve good performance in evading detection, using simple prompt is difficult for LLMs to fully understand the task objective of evading detection, and these methods still rely on domain expertise. Therefore, to sufficiently leverage the reasoning capabilities of LLMs, We propose a novel automated evasion method ToED. Its details will be introduced in the following sections.

3 Methodology

3.1 Problem Formula

Given a AI-generated text x, where the detection probability of x is $p(x)$. The goal of the detection evasion task is to modify x into $\tilde{x}$, ensuring that $p(\tilde{x}) < \sigma$, where σ is a hyperparameter that defines the detection threshold. We define $M(\cdot \mid \mathcal{P} \oplus x)$ as the function that modifies the text x using the language model, guided by the prompt $\mathcal{P}$, where $\oplus$ denotes the concatenation operation. The prompt-based detection evasion task involves designing $\mathcal{P}$ to guide the LLM in modifying the text into $\tilde{x} = M(\mathcal{P} \oplus x)$, while aiming to ensure that $p(\tilde{x}) < \sigma$.

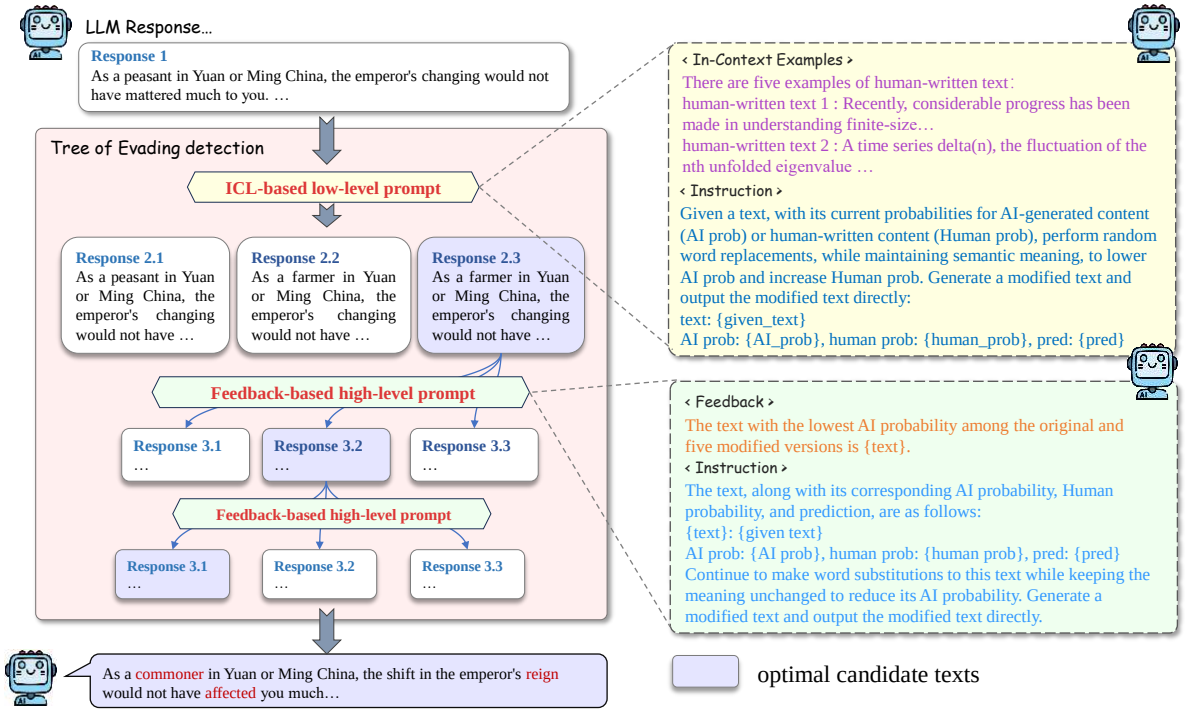

Figure 2: The overall framework of ToED. ToED adopts a two-tier mixed prompt strategy to construct a tree structure that enables progressive exploration of evasion strategies. The low-level prompt, built upon in-context learning, provides several representative examples and a task instruction to guide the LLM in generating diverse initial candidates. The high-level prompt incorporates both an instruction and dynamic feedback that identifies the current candidate with the lowest detection probability, instructing the LLM to further refine it. This feedback-driven process will be repeated multiple times until the detection probability of a candidate text falls below the threshold σ.

3.2 Tree of Evading Detection

In this work, we propose a novel method, Tree of Evading Detection (ToED), to address the challenges of evading AIGT detection effectively. Our method leverages the strengths of the "Tree of Thoughts" framework, treating candidate texts generated by LLMs as intermediate reasoning steps and enabling hierarchical search over varying depths and widths.

The overall framework of ToED is shown in Figure 2. A two-tier mixed prompt system containing an ICL-based low-level prompt and a feedback-based high-level prompt is designed. First, LLMs generate various candidate texts under the guidance of the low-level prompt. Then, leveraging the feedback from the proxy detector, the LLM searches along the current optimal candidate text under the guidance of the high-level prompt. Notably, if all current candidate texts perform poorly, ToED allows backtracking to the previous node to reconstruct new candidate texts. The latter process will be repeated multiple times until the detection probability of a candidate text falls below the threshold σ, at which point it will be output as the final result.

3.3 Two-tier Mixed Prompts

ICL-based Low-level Prompt. An intuitive approach to evading detection with LLMs is by using direct prompts, which allow the model to autonomously choose from a variety of modification operations, such as synonym replacement, paraphrasing, and style substitution. However, the modified text generated by a one-time direct prompt is unlikely to reduce the detection probability. As shown in Figure 3, using a one-time direct prompt can even increase the detection probability of the modified text to 93.99%. If we reduce the choice space of language model by employing a one-time, restricted direct prompt strategy that only allows synonym replacement, the resulting sentence is better than a one-time direct prompt, but still unlikely to meet the requirement of lowering the detection probability. As shown in Figure 3, we obtained a sentence with a detection probability of 88.49%.

Therefore, we attempt to expand the model's search space by generating multiple results in one go, selecting the optimal one, and further refining

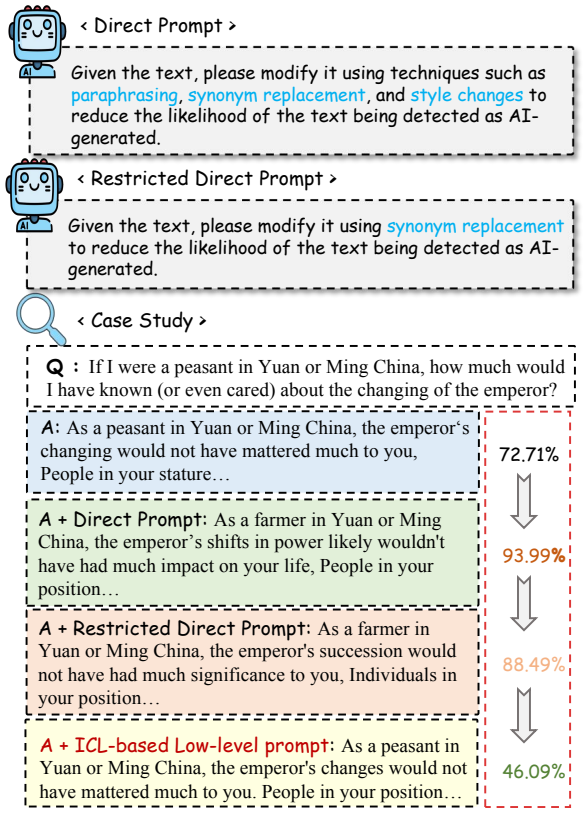

Figure 3: Prompt Example and Case Study using GPT-4o. We use a RoBERTa-based detector (Guo et al., 2023) to provide the detection probabilities.

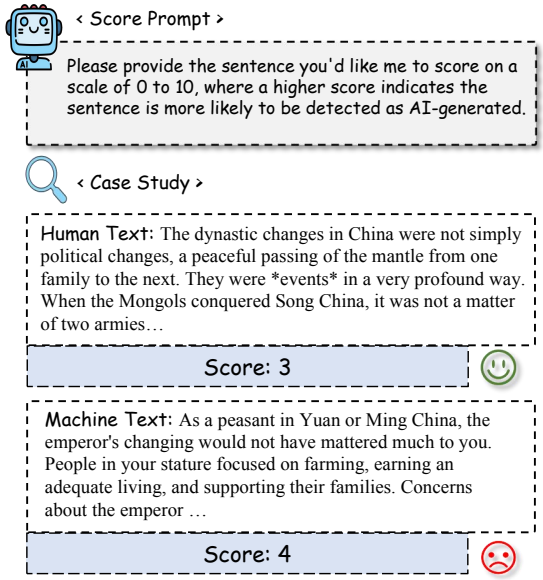

Figure 4: Score Prompt Example and Case Study using GPT-4o.

it. The model will continue selecting the best option based on the detection probability after each prompt, ensuring that every prompt yields a better candidate than the previous one.

Given the strong ICL capabilities of LLMs (Radford et al., 2019; Brown et al., 2020; Chowdhery et al., 2023), in the first step of generating candidates, to avoid indiscriminate modifications by LLM, we employ ICL to provide the model with few examples to guide it in modifying the text towards reducing the detection probability.

In selecting the optimal candidate text, to avoid random and indiscriminate choices by the model and help the LLM better understand the task requirements, we provide the detection probability of each sentence as a reference. However, directly using the LLM to evaluate the quality of specific candidate texts is unreliable, as shown in Figure 4. Therefore, we introduce a proxy detector in the process of evaluating the detection probabilities, which can accurately assess the effectiveness of the candidate sentences.

As shown in the case study in Figure 3, our proposed ICL-based low-level prompt, which introduces in-context examples and the detection probability of given text, further reduces the detection probability of the modified text to 46.09%. The case study indicates that our proposed ICL-based low-level prompt can effectively guide the LLM in generating higher-quality candidate texts compared to other two prompts.

Feedback-based High-level Prompt. It consists of two components: feedback and instruction.

(1) Feedback. To guide the LLMs in generating candidate texts in each iteration, we introduce a feedback mechanism. Specifically, we first use a proxy detector to score each generated candidate text and use the candidate text with the lowest detection probability as the optimal choice for the next iteration. We then provide a supervised feedback signal to LLMs by introducing the phrase *"The text with the lowest AI probability among the original and the modified versions is..."* to stimulate the reasoning ability of LLMs.

(2) Instruction. Similar to low-level prompt, the instruction in high-level prompt also guide LLMs to modify the text with lowest detection probability based on the feedback received in new iterations via restricted direct prompt. Meanwhile, the text detection probability from proxy detector is provided to help LLMs generate better candidate texts.

By combining the reasoning capabilities of LLMs with feedback-driven refinement, ToED can generate high-quality, detection-evading texts, which offers a systematic approach to optimizing

text modification strategies and enhancing the evasion of AIGT detection systems.

4 Experiment

4.1 Experiment Setting

Datasets. To evaluate the performance of ToED in modifying texts generated by various LLMs, we use the M4 (Wang et al., 2024b) and RAID (Dugan et al., 2024) as the benchmark datasets. M4 is multigenerator, multi-domain, and multi-lingual corpus for the AIGT detection task. We sample 500 examples from Reddit Davinci (R-Davinci) and Reddit ChatGPT (R-ChatGPT) (Ouyang et al., 2022) respectively, with each sample consisting of 500 human-written texts and 500 AIGT samples. Additionally, RAID is also the latest AIGT detection dataset, including over 6 million generations spanning 11 models, 8 domains, 11 adversarial attacks, and 4 decoding strategies. In this paper, We sample 500 human-written texts and 500 texts generated by LLaMA2-70B-Chat (Touvron et al., 2023) and GPT-4 (OpenAI, 2023) respectively.

AIGT Detectors. We employ various AIGT detectors to evaluate the performance of ToED: *(1) chatgpt-detector* (Guo et al., 2023) is a widely used and powerful pretraining-based detector. *(2) Radar* (Hu et al., 2023) is a robust detector based on adversarial learning. *(3) GLTR* (Gehrmann et al., 2019) proposes three simple tests to assess whether the text is generated in a specific assumed manner. In this work, we use the most powerful Test-2 feature, which is the absolute rank of a word, consistent with Guo et al. (2023). *(4) Fast-detectGPT* (Bao et al., 2024) utilizes conditional probability curvature to elucidate discrepancies in word choices between LLMs and humans within a given context. In this study, both the sampling model and the scoring model used in the method are GPT-2 (Radford et al., 2019).

Baselines. We use three evasion methods as baselines to evaluate the performance of ToED: *(1) DIPPER* (Krishna et al., 2023) rewrites text by fine-tuning T5. Following Zhou et al. (2024a), we employ dipper with lex=40 and order=40, the most effective setting in their paper. *(2) HMGC* (Zhou et al., 2024a) employs adversarial strategies to replace key terms critical for detection with synonyms. Consistent with the original, we compute the text perplexity using pythia-2.8b-deduped (Biderman et al., 2023). Additionally, the victim detector used in this method also employs chatgpt-detector. *(3) SICO* (Lu et al., 2023) guides the model to generate more human-like text by using samples that have undergone synonym replacement and paraphrasing attacks as in-context examples. In this study, we use GPT-3.5 to paraphrase text.

Evaluation metrics. We use the accuracy (ACC) and F1 score to evaluate the performance of detectors. In our experiments, lower values indicate better evasion performance.

Implementation Setting. In our experiments, we conduct experiments using Grok-beta (xAI team, 2024) and GPT-3.5, respectively. And chatgpt-detector is employed as proxy detector across all scenarios. We set the detection threshold $\sigma = 0.5$. The maximum number of iterations for the tree is set to 10, and the number of candidate texts generated per iteration is set to 4. By default, we provide 5 human-written texts as in-context examples in our experiments. The modification process terminates when the detection probability of a candidate text is lower than σ, or when the maximum number of iterations is reached. We evaluate the performance of each detector on 8GB RTX 4090, with a batch size of 1 for the test set.

4.2 Main Results

We compare our method with three baselines across four datasets, as shown in Table 1. ToED consistently outperforms the other baselines by causing a greater reduction in the average detection accuracy of the four detectors across all datasets. Notably, when using Grok-beta, the average detection performance across all four datasets drops below 50%, with a decrease of approximately 30% in detection accuracy compared to the original texts. We believe that DIPPER lacks targeted design for the characteristics of the detectors, which results in its mediocre performance in evasion detection tasks. HMGC is adversarially optimized against a target detector, ensuring that the modified text can evade detection by that specific detector. However, it exhibits poor generalization to unseen detectors. As shown in Table 1, when optimizing GPT-4-generated text, the accuracy of the target detector (chatgpt-detector) drops to 50%, indicating complete evasion. Nevertheless, the optimized text remains detectable by other detectors. For example, the detection accuracy of fast-detect decreases by only 0.1%. As for the SICO, We argue that such directly generated evasion methods do not alter the underlying probability distribution of the generated text and thus retain intrinsic characteristics

Dataset	Method	Orignal		DIPPER		HMGC		SICO		ToED (ours)	
		ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
R-Davinci	chatgpt-detector	69.40	56.41	94.70	94.45	49.60	0.00	54.90	19.03	49.90	1.18
	Radar	80.10	76.11	70.30	59.59	82.00	78.87	60.50	38.00	64.60	47.79
	GLTR	68.17	45.58	71.82	46.10	67.04	45.64	27.80	26.14	10.30	10.24
	fast-detectGPT	72.00	77.95	72.50	78.43	71.30	77.28	46.40	47.14	86.50	85.56
R-ChatGPT	chatgpt-detector	93.00	92.54	96.60	6.51	49.60	0.00	55.10	19.68	50.09	5.03
	Radar	95.90	95.86	81.80	78.59	96.30	96.28	67.40	53.82	71.20	61.29
	GLTR	68.00	45.38	69.72	46.10	62.75	45.25	30.80	28.48	11.40	11.38
	fast-detectGPT	72.50	78.43	72.50	78.43	69.90	75.90	47.30	48.48	82.70	81.46
LLama2	chatgpt-detector	76.70	69.62	75.80	68.07	50.00	0.00	50.00	0.00	50.00	0.00
	Radar	72.10	61.83	70.10	57.95	79.10	73.91	50.08	5.02	57.90	28.52
	GLTR	64.83	60.31	90.23	80.23	63.83	59.40	40.10	32.22	37.40	27.59
	fast-detectGPT	85.00	86.70	85.90	87.60	84.90	86.60	62.60	58.63	6.30	0.64
GPT-4	chatgpt-detector	54.90	17.85	66.30	49.17	50.00	0.00	50.00	0.00	50.00	0.00
	Radar	84.60	82.01	97.70	97.67	84.60	82.00	74.80	66.75	69.40	56.53
	GLTR	86.53	78.36	96.73	86.69	86.51	78.26	63.70	63.30	38.30	29.17
	fast-detectGPT	73.20	73.41	82.80	84.42	73.10	73.29	42.60	18.23	17.02	13.39
Average		76.06	68.65	80.97	68.75	70.03	54.54	51.51	32.81	**47.13**	**28.74**

Table 1: The effectiveness of detection evasion methods across different detectors for various LLMs.

of machine-generated content. For example, on GPT-4-generated text, applying SICO results in detection accuracies of 63.70% for GLTR and 42.60% for Fast-detectGPT, both of which are higher than the accuracies achieved by our method, which are 38.30% and 17.02%, respectively. As a result, their performance tends to be limited when attempting to evade statistical-based detectors.

4.3 Ablation Study

Effectiveness Validation of the Mixed Prompts. We validate the effectiveness of the mixed prompt through ablation experiments. As shown in Table 2, "w/o ICL" represents removing in-context examples in low-level prompt and directly using instructions to guide the LLM for text modification. "w/o Feedback" represents removing the feedback component from the high-level prompt. The results show that removing the in-context examples generally leads to an increase in average evade detection performance form 47.13% to 50.79%, demonstrating that the LLM is able to understand the task objectives under the guidance of ToED. Additionally, removing feedback results in an increase of both accuracy and F1 score, which demonstrates that the LLM can benefit from the feedback provided by the proxy detector.

Performance Validation of Different Proxy Detectors. We conduct experiments using the robust detector RADAR as the proxy detector, with the results shown in Figure 5. We find that using different proxy detectors can effectively reduce the average detection accuracy, but the choice of proxy detector significantly impacts the evasion effectiveness. Experiments show that the average detection accuracy with chatgpt-detector is consistently better than with the robust detection model RADAR. We believe that chatgpt-detector offers sufficiently informative feedback for the LLM to grasp the task objective, while more advanced proxy detectors may hinder the LLM's ability to discover an effective modification strategy. This result aligns with our previous observation, confirming that the LLM adapts its modification strategy according to the feedback from different proxy detectors, leading to

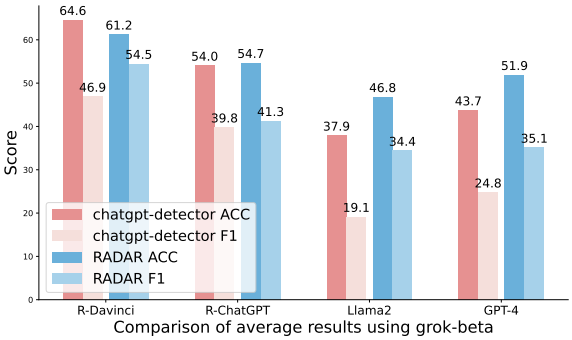

Figure 5: Comparison of results of ToED with different proxy detectors.

Dataset	Method	Grok-beta					
		ToED		ToED w/o Feedback		ToED w/o ICL	
		AUROC	F1	AUROC	F1	AUROC	F1
R-Davinci	chatgpt-detector	49.90	1.18	49.60	0.00	49.60	0.00
	Radar	64.60	47.79	65.00	48.68	60.90	39.00
	GLTR	10.30	10.24	10.00	9.93	9.20	9.08
	fast-detectGPT	86.50	85.56	86.10	85.10	89.20	88.68
R-ChatGPT	chatgpt-detector	50.09	5.03	49.50	9.66	48.40	5.84
	Radar	71.20	61.29	70.40	59.78	67.60	54.24
	GLTR	11.40	11.38	11.20	11.18	8.70	8.55
	fast-detectGPT	82.70	81.46	89.29	82.09	93.22	84.71
LLama2	chatgpt-detector	50.00	0.00	50.00	0.00	50.00	0.00
	Radar	57.90	28.52	58.30	29.68	58.60	30.54
	GLTR	37.40	27.59	37.50	27.78	46.00	41.33
	fast-detectGPT	6.30	0.64	7.70	6.50	7.40	1.91
GPT-4	chatgpt-detector	50.00	0.00	50.00	0.00	50.00	0.00
	Radar	69.40	56.53	68.80	55.30	80.90	76.68
	GLTR	38.30	29.17	38.30	29.17	73.90	73.90
	fast-detectGPT	17.02	13.39	16.60	12.76	19.07	23.11
Average		**47.13**	**28.74**	47.40	29.23	50.79	33.60

Table 2: Ablation study of mixed prompts.

Dataset	Grok-beta		GPT-3.5	
	ACC	F1	ACC	F1
R-Davinci	52.83	36.19	53.97	48.09
R-ChatGPT	54.05	39.79	49.60	45.70
Llama2	37.90	14.19	69.00	54.12
GPT-4	43.73	24.77	60.05	39.53
Average	**47.73**	**28.74**	58.15	48.86

Table 3: Comparison of results using different LLMs.

varying attack performance.

Performance Validation of Different LLMs. As shown in Table 3, we compare the performance of different LLMs used in ToED and find that Grok-beta outperforms GPT-3.5 in most cases. This indicates that as the capabilities of the LLMs improve, the performance of ToED will also enhance. Notably, when detecting texts generated by ChatGPT, using ChatGPT for evading detection yields better performance than using Grok. This indicates that modifying texts with the same LLM as the one used for generation is more effective than using a more advanced LLM for modification.

Additionally, we analyze the impact of the number of candidate texts generated per iteration. Experimental results show that performance of ToED improves as the number of candidates increases. Detailed results can be found in Appendix A.

5 conclusion

In conclusion, we propose a novel prompt-based detection evasion method ToED. The crux of ToED is to build a tree structure with two-tier mixed prompt to search for a satisfactory modification strategy, improving the detection evasion ability of AIGT. The two-tier mixed prompt consist of an ICL-based low level prompt and a feedback-based high level prompt. The former consists of several in-context examples and an instruction, used to guide the LLM in generating candidate texts. The latter includes a feedback and an instruction. The feedback reveals the text with the lowest detection probability among all previous candidate texts, and the instruction guides the LLM to modify this specific text. Our extensive experiments on evasion demonstrate the superior performance of ToED, which significantly reduces the detection capabilities of virous existing AIGT detectors among texts generated by different LLMs.

Limitations

ToED requires multiple API calls, which may lead to increased costs. In addition, experimental results show that using proxy detectors to provide feedback can effectively reduce the average detection accuracy across multiple detectors. However, since each modification requires the proxy detector to

evaluate the text, this incurs additional resource consumption and reduces the overall practicality of the method. Future work will explore more lightweight and user-friendly alternatives.

Ethics Statement

The primary objective of this paper is not to provide techniques for evading AIGT detection systems, but rather to expose vulnerabilities in current detection mechanisms. With the widespread adoption of LLMs, adversaries can more easily leverage these models to generate detection-resistant text through carefully crafted prompts. This study aims to call upon the research community to prioritize the development of more robust text detection methods to address these emerging challenges. We firmly believe that with increased attention and efforts toward this issue, the research community can devise more sophisticated and effective techniques to enhance the robustness and reliability of machine-generated text detection systems in the face of evolving adversarial threats. We affirm that all research activities strictly adhere to academic ethics and privacy protection principles. All data used complies with relevant laws and regulations and does not infringe upon the rights of any individuals or organizations.

References

Jinze Bai, Shuai Bai, Yunfe Chu, and et al. 2023. Qwen Technical Report. *arXiv e-prints*, arXiv:2309.16609.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2024. Fast-detectGPT: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *The Twelfth International Conference on Learning Representations*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Tom B. Brown, Benjamin Mann, Nick Ryder, and et al. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Aakanksha Chowdhery, Sharan Narang, and Jacob Devlin et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Liam Dugan, Alyssa Hwang, Filip Trhlík, Andrew Zhu, Josh Magnus Ludan, Hainiu Xu, Daphne Ippolito, and Chris Callison-Burch. 2024. RAID: A shared benchmark for robust evaluation of machine-generated text detectors. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12463–12492, Bangkok, Thailand. Association for Computational Linguistics.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. 2019. GLTR: Statistical Detection and Visualization of Generated Text. *arXiv e-prints*, arXiv:1906.04043.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *Preprint*, arXiv:2301.07597.

Beizhe Hu, Qiang Sheng, Juan Cao, and et al. 2024. Bad actor, good advisor: Exploring the role of large language models in fake news detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(20):22105–22113.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2023. Radar: Robust ai-text detection via adversarial learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 15077–15095. Curran Associates, Inc.

Guanhua Huang, Yuchen Zhang, Zhe Li, and et al. 2024. Are AI-Generated Text Detectors Robust to Adversarial Perturbations? *arXiv e-prints*, arXiv:2406.01179.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, and et al. 2023. Mistral 7B. *arXiv e-prints*, arXiv:2310.06825.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, and et al. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. In *Advances in Neural Information Processing Systems*, volume 36, pages 27469–27500. Curran Associates, Inc.

Ning Lu, Shengcai Liu, Rui He, Qi Wang, Yew-Soon Ong, and Ke Tang. 2023. Large language models can be guided to evade ai-generated text detection. *arXiv preprint arXiv:2305.10847*.

Shixuan Ma and Quan Wang. 2024. Zero-shot detection of LLM-generated text using token cohesiveness. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17538–17553, Miami, Florida, USA. Association for Computational Linguistics.

Niloofar Mireshghallah, Justus Mattern, Sicun Gao, and et al. 2024. Smaller language models are better zero-shot machine-generated text detectors. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 278–293, St. Julian's, Malta. Association for Computational Linguistics.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, and et al. 2023. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. *arXiv e-prints*, arXiv:2301.11305.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

OpenAI. 2023. GPT-4 Technical Report. *arXiv e-prints*, arXiv:2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, and et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Jinyan Su, Claire Cardie, and Preslav Nakov. 2024. Adapting fake news detection to the era of large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1473–1490, Mexico City, Mexico. Association for Computational Linguistics.

Jinyan Su, Terry Zhuo, Di Wang, and Preslav Nakov. 2023. DetectLLM: Leveraging log rank information for zero-shot detection of machine-generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12395–12412, Singapore. Association for Computational Linguistics.

Yuchuan Tian, Hanting Chen, Xutao Wang, Zheyuan Bai, QINGHUA ZHANG, Ruifeng Li, Chao Xu, and Yunhe Wang. 2024. Multiscale positive-unlabeled detection of AI-generated texts. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Louis Martin, Kevin Stone, and et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv e-prints*, arXiv:2307.09288.

Saranya Venkatraman, Adaku Uchendu, and Dongwon Lee. 2024. GPT-who: An information density-based machine-generated text detector. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 103–115, Mexico City, Mexico. Association for Computational Linguistics.

Vivek Verma, Eve Fleisig, Nicholas Tomlin, and Dan Klein. 2024. Ghostbuster: Detecting text ghostwritten by large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1702–1717, Mexico City, Mexico. Association for Computational Linguistics.

Yichen Wang, Shangbin Feng, Abe Hou, and et al. 2024a. Stumbling blocks: Stress testing the robustness of machine-generated text detectors under attacks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2894–2925, Bangkok, Thailand. Association for Computational Linguistics.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Toru Sasaki, Thomas Arnold, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024b. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1369–1407, St. Julian's, Malta. Association for Computational Linguistics.

Kangxi Wu, Liang Pang, Huawei Shen, and et al. 2023. LLMDet: A third party large language models generated text detection tool. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2113–2133, Singapore. Association for Computational Linguistics.

xAI team. 2024. [link].

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.

Xiao Yu, Kejiang Chen, Qi Yang, and et al. 2024. Text fluoroscopy: Detecting LLM-generated text through intrinsic features. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15838–15846, Miami, Florida, USA. Association for Computational Linguistics.

Xiao Yu, Yuang Qi, Kejiang Chen, and et al. 2023. DPIC: Decoupling Prompt and Intrinsic Characteristics for LLM Generated Text Detection. *arXiv e-prints*, arXiv:2305.12519.

Zijie Zeng, Lele Sha, and Yuheng et al. Li. 2024. Towards automatic boundary detection for human-ai collaborative hybrid essay in education. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(20):22502–22510.

Ying Zhou, Ben He, and Le Sun. 2024a. Humanizing machine-generated content: Evading AI-text detection through adversarial attack. In *Proceedings of*

the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), pages 8427–8437, Torino, Italia. ELRA and ICCL.

Yinghan Zhou, Juan Wen, Jianghao Jia, and et al. 2024b. C-net: A compression-based lightweight network for machine-generated text detection. *IEEE Signal Processing Letters*, 31:1269–1273.

A The Impact of the Number of Candidate Texts Generated Per Iteration.

As shown in Figure 6, increasing the number of candidates text enables the model to search more modification strategies, which can enhance evade detection performance by identifying an effective modification. However, when the number of generated candidate texts is too large, it also make it difficult for the LLM to find an effective modification strategy, leading to a decrease in performance. This is consistent with the analysis presented above.

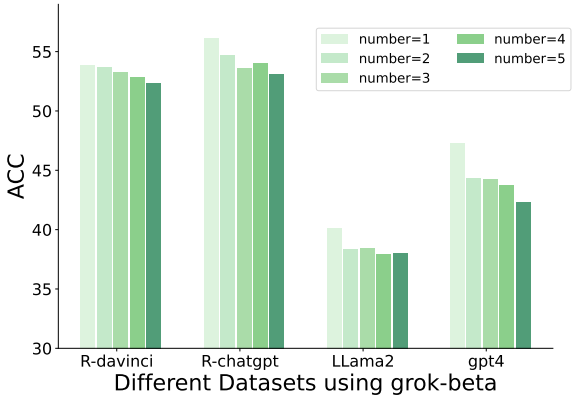

Figure 6: The average ACC of ToED using Grok-beta across four LLMs as the number of candidate texts generated per iteration increases.