Large Language Models can be Guided to Evade AI-Generated Text Detection

Anonymous authors
Paper under double-blind review

Abstract

Large language models (LLMs) have shown remarkable performance in various tasks and have been extensively utilized by the public. However, the increasing concerns regarding the misuse of LLMs, such as plagiarism and spamming, have led to the development of multiple detectors, including fine-tuned classifiers and statistical methods. In this study, we equip LLMs with prompts, rather than relying on an external paraphraser, to evaluate the vulnerability of these detectors. We propose a novel **S**ubstitution-based **I**n-**C**ontext example **O**ptimization method (SICO) to automatically construct prompts for evading the detectors. SICO is cost-efficient as it requires only 40 human-written examples and a limited number of LLM inferences to generate a prompt. Moreover, once a task-specific prompt has been constructed, it can be universally used against a wide range of detectors. Extensive experiments across three real-world tasks demonstrate that SICO significantly outperforms the paraphraser baselines and enables GPT-3.5 to successfully evade six detectors, decreasing their AUC by 0.5 on average. Furthermore, a comprehensive human evaluation as well as a validation experiment in the wild show that the SICO-generated text achieves human-level readability and task completion rates. Finally, the strong performance of SICO exhibits its potential as a reliable evaluation tool for future detectors.

1 Introduction

The rapid advancement of large language models (LLMs), such as GPT (Brown et al., 2020), PaLM (Chowdhery et al., 2022), and LLaMa (Touvron et al., 2023), has led to a largely-increased capacity for generating high-quality human-like text. However, there are also growing concerns surrounding the misuse of these models, including generating fake product reviews (Adelani et al., 2020; Lin et al., 2022) and misinformation (Lin et al., 2022), enabling academic dishonesty (Stokel-Walker, 2022), and producing misleading answers on websites (StackOverflow, 2023).

In response to these challenges, several methods for detecting AI-generated text have been proposed recently, ranging from fine-tuned classifiers (Guo et al., 2023; Solaiman et al., 2019), statistical methods (Mitchell et al., 2023), to watermarking (Kirchenbauer et al., 2023). There are also online detection services provided by companies such as GPTzero (Tian, 2023). However, the robustness of these detection methods has not been thoroughly evaluated. Recent studies (Krishna et al., 2023; Sadasivan et al., 2023) have shown the vulnerability of these detectors to the so-called *paraphrase attacks*, which adopt an external paraphraser to rewrite the text generated by LLMs to evade detectors.

In this work, rather than relying on an external paraphraser, we explore equipping LLMs with carefully constructed prompts to evade detectors. The intuition is that, given the remarkable capabilities of LLMs, appropriate prompts can guide these models to potentially achieve and even exceed the evasion performance level of smaller external paraphrasers. We propose **SICO**, a **S**ubstitution-based **I**n-**C**ontext example **O**ptimization method, to automatically construct such prompts based on human-generated examples. Specifically, SICO iteratively substitutes words and sentences within the in-context examples to provide more representative demonstrations for LLMs to generate text that cannot be detected, where the substitution procedure is directed by a proxy detector (see Figure 1 for an overview of SICO).

We assess the evasion performance of SICO across three real-world tasks that are susceptible to the misuse of LLMs, i.e., academic essay writing, open-ended question answering, and fake review generation. The results demonstrate that SICO consistently outperforms the paraphraser baselines, leading to a decrease in AUC by approximately 0.5 on average for six existing detectors. Additionally, a comprehensive human evaluation involving 600 examples shows that the SICO-generated text is comparable to, and in some cases even better than, human-written text in terms of readability and task completion rates. To further evaluate the practical utility of SICO, we deploy it on Reddit, an online social platform, to generate responses for users' questions. The high percentage of generated responses that are liked by Reddit users shows that SICO is capable of generating human-approved content while being barely identified as AI. In addition to its strong evasion performance, SICO is also cost-efficient and easy to use. Unlike paraphraser-based methods that often require extensive computational resources – as evidenced by the fine-tuning of a 13B model on a large dataset (Krishna et al., 2023) – SICO only requires 40 human-generated examples and a limited number of LLM inferences (e.g., costing approximately 1 USD using the GPT-3.5 API). Besides, once a task-specific prompt has been constructed by SICO, it can be universally used against a wide range of detectors.

Considering the importance of detecting AI-generated text to avoid their misuse, the results presented in this work certainly reveal the vulnerability of the existing detectors. Besides, this work presents the first empirical evidence that LLMs can evade detectors through a prompt-guided approach. Finally, the strong evasion performance of SICO suggests that it can be used as a standard evaluation tool for any future AI-generated text detectors. We hope that these findings can better facilitate the research concerning the responsible use of LLMs. To summarize, our main contributions are:

- We introduce SICO, a novel in-context example learning method, to automatically construct prompts that can guide LLMs to evade detectors.
- With low cost, SICO achieves strong performance in evading six existing detectors across three tasks, significantly outperforming the paraphraser baselines.
- A comprehensive human evaluation, as well as a validation experiment in the wild, verifies that the SICO-generated text achieves human-level readability and task completion rates.

2 Related works

2.1 AI-generated text detection

In recent years, the research community has developed a wide range of detectors for AI-generated contents. In general, these detectors can be classified into three categories: training-based, statistical, and watermarking methods. Training-based methods treat the detection problem as a binary classification task, where neural networks are trained using AI-generated text and human-written text. Early studies utilized classifiers to identify fake reviews (Hovy, 2016) and fake news (Zellers et al., 2019). More recently, researchers have trained classifiers using text generated by LLMs, such as the GPT-3.5 detector (Guo et al., 2023) and GPT-2 detector (Solaiman et al., 2019).

Statistical methods, on the other hand, focus on zero-shot detection without any additional training overhead. These methods seek to distinguish between human-written text and AI-generated text based on the statistical characteristics of text, such as the statistical irregularities in measures like entropy (Lavergne et al., 2008), perplexity (Beresneva, 2016) and token rank (Gehrmann et al., 2019). A recent method, DetectGPT (Mitchell et al., 2023), exploits the phenomenon that AI-generated text tends to lie in the negative curvature regions of log probability of text.

The watermarking methods involve modifying the LLM's text generation process to imprint specific patterns on the generated text, such that it can be detected (Abdelnabi & Fritz, 2021; Grinbaum & Adomaitis, 2022; Kirchenbauer et al., 2023). Although the proposed method SICO primarily focuses on the first two types of detection methods, it can also help evade watermarking when acted as an external paraphraser, as shown in Appendix G.

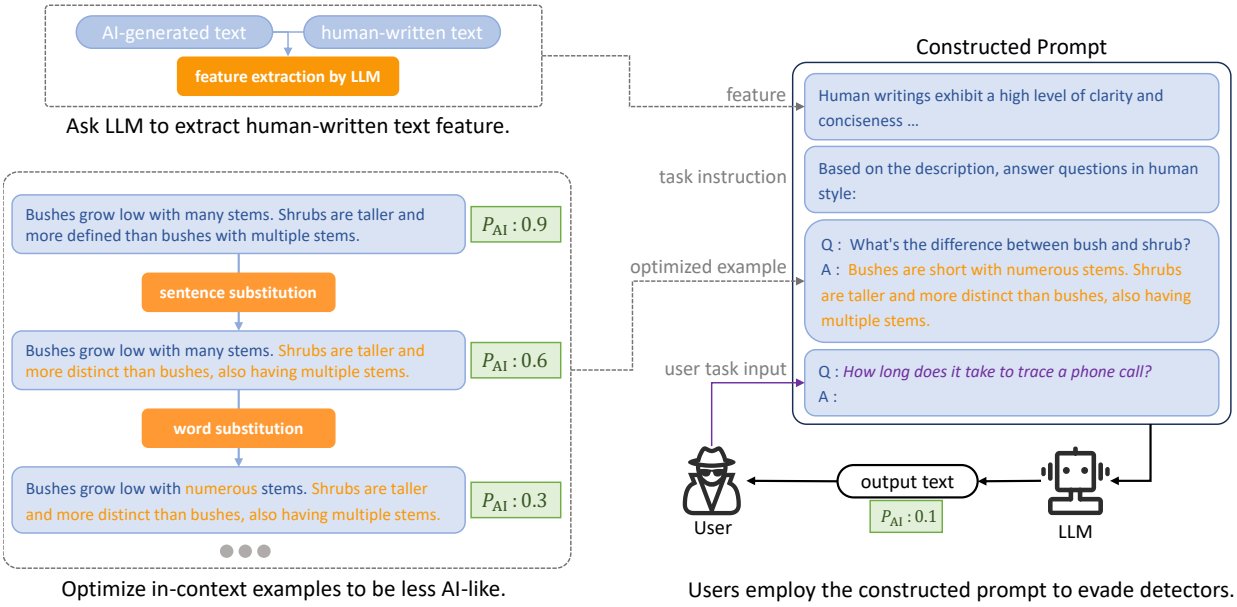


Figure 1: Illustration of how SICO generates prompts for the question answering task. The probability P_{AI}, as predicted by the proxy detector, indicates the likelihood that the given text is AI-generated. Once SICO prompt is constructed, it serves as a template, allowing users to insert various task inputs (highlighted in purple text).

2.2 In-context learning

With the increasing scales of models and corpora (Devlin et al., 2019; Radford et al., 2019; Chowdhery et al., 2022; Gou et al., 2022), LLMs have demonstrated the in-context learning (ICL) ability, allowing them to perform tasks with only a few examples provided as demonstrations (Brown et al., 2020). Recent studies have focused on designing demonstrations during inference, which can be divided into demonstration selection, ordering, and formatting (Dong et al., 2022). Specifically, demonstrations can be selected based on unsupervised metrics or supervised strategies (Kim et al., 2022; Gonen et al., 2022; Rubin et al., 2022). For ordering, Liu et al. (2021) sorted examples by their distances to the input. Regarding demonstration formatting, Wei et al. (2022) proposed the so-called chain-of-thoughts (COT) format, and subsequent works have developed automatic COT (Zhang et al., 2022). In contrast to these works, we focus on iteratively optimizing demonstrations through substitutions. In principle, the proposed method SICO can be used in combination with the above-mentioned methods, potentially leading to improved performance.

3 Substitution-based in-context example optimization (SICO)

The illustration of SICO is presented in Figure 1. First, LLM is asked to extract language features of human-written text. Then, the in-context examples are initialized and optimized. The final prompt is composed of the feature, task instruction, and optimized in-context examples. Below, we first describe how to evaluate a prompt during its optimization and then elaborate all the steps of SICO.

3.1 Prompt evaluation

Given a natural language processing task, denote the task input as x. To assess the utility of a prompt p, we first collect a set of task inputs, X_{eval}. For each input $x \in X_{eval}$, p and x are first concatenated (denoted by $p \oplus x$) and fed into the LLM, whose output text (denoted by $\text{LLM}(p \oplus x)$) is then classified by a proxy detector. Let $\mathcal{P}_{\text{AI}}$ be the predicted probability of $\text{LLM}(p \oplus x)$ to be AI-generated, then the utility score of prompt p, denoted by $\mathcal{U}(p)$, is defined as one minus the averaged predicted probability across X_{eval} (the

Algorithm 1 Substitution-based in-context example optimization (SICO)

Require: large language model LLM, prompt utility function $\mathcal{U}(\cdot)$, $D = \{(x_{\text{ic}}^k, y_{\text{AI}}^k, y_{\text{human}}^k)\}_{k=1}^K$, $\mathbf{X}_{eval}$, total iteration number N

Extract language feature t_{feature} using $\{(y_{\text{AI}}^k, y_{\text{human}}^k)\}_{k=1}^K$ and LLM
Construct in-context outputs $y_{\text{ic}}^k = \text{LLM}(t_{\text{feature}} \oplus p_{\text{para}} \oplus y_{\text{AI}}^k),\ \forall k \in \{1, ..., K\}$
Initialize $p^* \leftarrow t_{\text{feature}} \oplus p_{\text{task}} \oplus \{(x_{\text{ic}}^k, y_{\text{ic}}^k)\}_{k=1}^K$
for $n = 1$ to N **do**
  for $k = 1$ to K **do**
    Generate sentence-level / word-level substitutions C^k of y_{ic}^k, switching based on n
    Optimize y_{ic}^k using Algorithm 2: $\hat{y}_{\text{ic}}^k \leftarrow \text{GreedyOPT}(y_{\text{ic}}^k, C^k)$
  end for
  Construct new prompt $\hat{p}$: $\hat{p} \leftarrow t_{\text{feature}} \oplus p_{\text{task}} \oplus \{(x_{\text{ic}}^k, \hat{y}_{\text{ic}}^k)\}_{k=1}^K$
  if $\mathcal{U}(\hat{p}) > \mathcal{U}(p^*)$ **then**
    Update in-context examples $y_{\text{ic}}^k \leftarrow \hat{y}_{\text{ic}}^k$ and update the best prompt $p^* \leftarrow \hat{p}$
  end if
end for
return p^*

higher $\mathcal{U}$, the better):

$$\mathcal{U}(p) = 1 - \frac{1}{|\mathbf{X}_{eval}|} \sum_{x \in X_{eval}} \mathcal{P}_{\text{AI}}(\text{LLM}(p \oplus x)). \tag{1}$$

3.2 Prompt Construction

Data collection We first collect a set of K triplets, i.e., $D = \{(x_{\text{ic}}^k, y_{\text{AI}}^k, y_{\text{human}}^k)\}_{k=1}^K$, where x_{ic}^k is a task input and $y_{\text{AI}}^k, y_{\text{human}}^k$ are the corresponding outputs generated by the LLM and humans, respectively. Note D is used for prompt construction and it is independent of X_{eval} which is used for prompt evaluation.

Feature extraction This step involves the K pairs of AI-generated and human-written outputs from D, denoted by $\{(y_{\text{AI}}^k, y_{\text{human}}^k)\}_{k=1}^K$. We provide LLM with these pairs and ask LLM to extract the distinct linguistic features of human-written text, denoted as t_{feature}.

In-context example optimization We initialize the in-context examples as $(x_{\text{ic}}^k, y_{\text{ic}}^k)$, where y_{ic}^k is generated by paraphrasing y_{AI}^k. More specifically, the text feature t_{feature} is concatenated with a paraphrasing instruction to instruct LLM to paraphrase y_{AI}^k to generate y_{ic}^k .

Then the in-context output y_{ic} is iteratively optimized to be less AI-like, which is directed by the proxy detector. By presenting more and more representative in-context demonstrations to LLM, it is expected to understand how to generate human-like outputs. Formally, the optimization goal can be expressed as:

$$y_{\text{ic}}^* = \underset{y'_{\text{ic}} \in \text{SIM}(y_{\text{ic}})}{\arg\min}\ \mathcal{P}_{\text{AI}}(y'_{\text{ic}}), \tag{2}$$

Algorithm 2 Greedy text optimization (GreedyOPT)

Require: Text y, substitutions C of y, proxy detector $\mathcal{P}_{\text{AI}}$

$C_{i,*} = \underset{C_{i,j}}{\arg\min}\ \mathcal{P}_{\text{AI}}(y_{(i,j)}),\ \forall y_i \in y$
where $y_{(i,j)} = \text{SUB}(y_i, C_{i,j})$
for each y_i in y **do**
  $y \leftarrow \text{SUB}(y_i, C_{i,*})$
end for
return y

where $\text{SIM}(y_{\text{ic}})$ denotes the set of text that is semantically similar to y_{ic}. The goal of setting such semantic restriction is to maintain the usability of the text during optimization. In SICO, we generate semantically similar text by replacing words with their synonyms and rephrasing sentences. This is explained in detail below.

Substitution type To generate y'_{ic} that is semantically similar to y_{ic}, we employ substitution at word level and sentence level in turn. For word-level substitution, we use WordNet (Miller, 1998), a lexical database

of English words, to construct a synonym substitution set. We restrict substitutions to content words that carry meanings and ensure that the substitution would not change the part-of-speech tags. We use a mask language model to filter out the candidate words that not fits the context. For sentence-level substitution, we utilize a paraphrasing instruction combined with extracted feature, denoted as $t_{\text{feature}} \oplus p_{\text{para}}$. This combined instruction is used to prompt LLM to generate paraphrases for each sentence in y_{ic}.

Algorithm As illustrated in Algorithm 1, SICO would optimize $\{y_{\text{ic}}^k\}_{k=1}^K$ for N iterations (lines 4-17). At each iteration, each y_{ic}^k would be optimized by greedy substitution (line 11), as presented in Algorithm 2. Specifically, for the i-th original word/sentence y_i in the text y, let $C_{i,j}$ denote its j-th synonym/paraphrase, and let $\text{SUB}(y_i, C_{i,j})$ denote the new text resulting from substituting y_i with $C_{i,j}$. For each y_i, SICO finds the best synonym/paraphrase $C_{i,*}$ by checking which $C_{i,j}$ gives the lowest AI-probability when substituting y_i (Line 1 in Algorithm 2).

After obtaining the optimized in-context output $\hat{y}_{\text{ic}}$, the new prompt is constructed as $\hat{p} = t_{\text{feature}} \oplus p_{\text{task}} \oplus \{(x_{\text{ic}}^k, \hat{y}_{\text{ic}}^k)\}_{k=1}^K$, where p_{task} is the task instruction, as illustrated in Figure 1. Then $\hat{p}$ would be compared with the current best prompt p^* based on their utility scores as defined in Eq. (1). If $\hat{p}$ scores higher, SICO replaces p^* with it. After N iterations, p^* is returned as the final prompt. More implementation details of SICO are shown in Appendix A.

3.3 SICO for Paraphrasing

The approach described above directly generates the task output to evade detectors. We refer to this direct approach as SICO-Gen. Alternatively, SICO can be easily adapted for paraphrasing, which we term as SICO-Para. Instead of direct generation, SICO-Para evades detectors in two steps. Initially, LLM produces an intermediate task output, typically incapable of evading detectors. Then. this output is paraphrased using SICO-Para to successfully evade detectors. Switching from SICO-Gen to SICO-Para requires only two adjustments: (1) the task input x is set to the AI-generated output text in D and $\mathbf{X}_{eval}$; (2) task instruction p_{task} is modified to paraphrasing instruction.

4 Experiments

4.1 Experimental setup

Tasks & datasets We consider three real-world tasks that are susceptible to the misuse of LLMs, i.e., academic essay writing (Writing), open-ended question answering (QA), and fake review generation (Review). We use GPT-3.5, one of the most powerful LLMs, to complete the tasks and generate text in our experiments.

For academic writing, we employ Wikipedia paragraphs from SQuAD dataset (Rajpurkar et al., 2016) as human-written text. Following the approach in Mitchell et al. (2023), we use the first 30 words of these paragraphs as task inputs and ask GPT-3.5 to complete the rest. For open-ended question answering, we sample questions from Eli5 (Fan et al., 2019) dataset and ask GPT-3.5 to generate answers, following Krishna et al. (2023). For fake review generation, we first instruct GPT-3.5 to extract the business name and five keywords from human-written reviews from Yelp dataset (Zhang et al., 2015), and then generate fake reviews based on the extracted information with specified sentiment. For each task, we collect 200 examples from GPT-3.5 (called original AI-generated text) and 200 human-written examples from corresponding dataset. More details about dataset can be found in Appendix F.

Detectors Six representative detectors belonging to three different types are considered. Details of these detectors can be found in Appendix C.

Training-based methods. (i) GPT-3.5 Detector (GPT3-D) (Guo et al., 2023): a RoBERTa model (Liu et al., 2019) fine-tuned on text generated by GPT-3.5. (ii) GPT2 Detector (GPT2-D) (Solaiman et al., 2019): a RoBERTa detector officially released by OpenAI, fine-tuned on GPT2-generated text.

Statistical methods. (i) DetectGPT (Mitchell et al., 2023) evaluates the variation in a language model's log probability by introducing minor perturbations to the detected text. (ii) Log-Rank (Mitchell et al., 2023) is a statistical method that employs a language model to compute the mean prediction rank of each token in a

Table 1: AUC scores of detectors on text generated by different methods. "–" refers to the detector's AUC score on the original AI-generated text, without applying any evasion methods. Symbol '*' represents that SICO uses GPT3-D as the proxy detector for prompt construction. For each detector, the lowest AUC score is indicated in **bold**, and the second-lowest is underlined.

Dataset	Method	GPT3-D*	GPT2-D	GPTzero	OpenAI-D	DetectGPT	Log-Rank
Writing	–	0.908	0.848	0.779	0.789	0.834	0.914
	Parrot	0.666	0.645	0.632	0.744	0.502	0.577
	DIPPER	0.736	0.907	0.689	0.750	0.550	0.684
	GPT-Para	0.879	0.623	0.631	0.690	0.569	0.713
	Human Prompt	0.852	0.560	0.491	0.655	0.676	0.759
	SICO-Para	**0.239**	0.332	0.290	0.488	**0.149**	**0.147**
	SICO-Gen	0.242	**0.099**	**0.184**	**0.311**	0.441	0.318
QA	–	0.981	0.906	0.923	0.781	0.876	0.956
	Parrot	0.922	0.837	0.849	0.698	0.689	0.806
	DIPPER	0.888	0.962	0.869	0.722	0.604	0.782
	GPT-Para	0.956	0.797	0.811	0.699	0.640	0.782
	Human Prompt	0.912	0.625	0.791	0.656	0.662	0.757
	SICO-Para	**0.407**	0.576	0.572	0.541	**0.178**	**0.183**
	SICO-Gen	0.668	**0.489**	**0.494**	**0.524**	0.497	0.535
Review	–	0.925	0.952	0.939	0.960	0.808	0.982
	Parrot	0.871	0.934	0.913	0.882	0.654	0.893
	DIPPER	0.875	0.984	0.888	0.824	0.515	0.814
	GPT-Para	0.899	0.851	0.833	0.925	0.542	0.864
	Human Prompt	0.839	0.610	0.856	0.858	0.619	0.851
	SICO-Para	0.465	**0.264**	0.599	**0.540**	**0.270**	**0.300**
	SICO-Gen	**0.455**	0.619	**0.399**	0.607	0.485	0.583

text, given its preceding context. We utilize a relatively smaller language model, GPT2-medium (Radford et al., 2019), for both methods. Because Mireshghallah et al. (2023) find that smaller language models have better detection performance than larger ones.

APIs.[1] (i) GPTzero (Tian, 2023) is a widely-used commercial detector, cooperated with many academic organizations. (ii) OpenAI Detector (OpenAI-D) (OpenAI, 2023) is officially offered by OpenAI, fine-tuned from a language model.

Baselines We consider four paraphrasing baselines that evade detectors by paraphrasing the original AI-generated text. Specifically, two recently proposed methods are considered: (1) *Parrot* (Sadasivan et al., 2023) and (2) *DIPPER* (Krishna et al., 2023). Both methods employ an external neural network specifically trained for paraphrasing. In addition, we include two prompting baselines to instruct GPT-3.5 to paraphrase the original AI-generated text: (3) *GPT-Para* that uses the straightforward instruction *"Paraphrase this"* to assess the capabilities of GPT-3.5 without intricate prompt engineering, and (4) *Human Prompt* that utilizes a human-designed prompt. More details can be found in Appendix A.2.

Evaluation metrics We use the area under the ROC curve (AUC) to measure the performance of detectors. The ROC curves are also illustrated to show the detection performance under different classification thresholds. For each task, we evaluate AUC score using 200 human-written text and 200 original or paraphrased AI-generated text. For each task input, we run each evasion method *only once*, instead of repeating multiple times until successful evasion, to simulate real-world scenarios where the target detector is inaccessible.

[1]We consider the API versions of May 15, 2023. For OpenAI-D, we follow the implementation of Krishna et al. (2023).

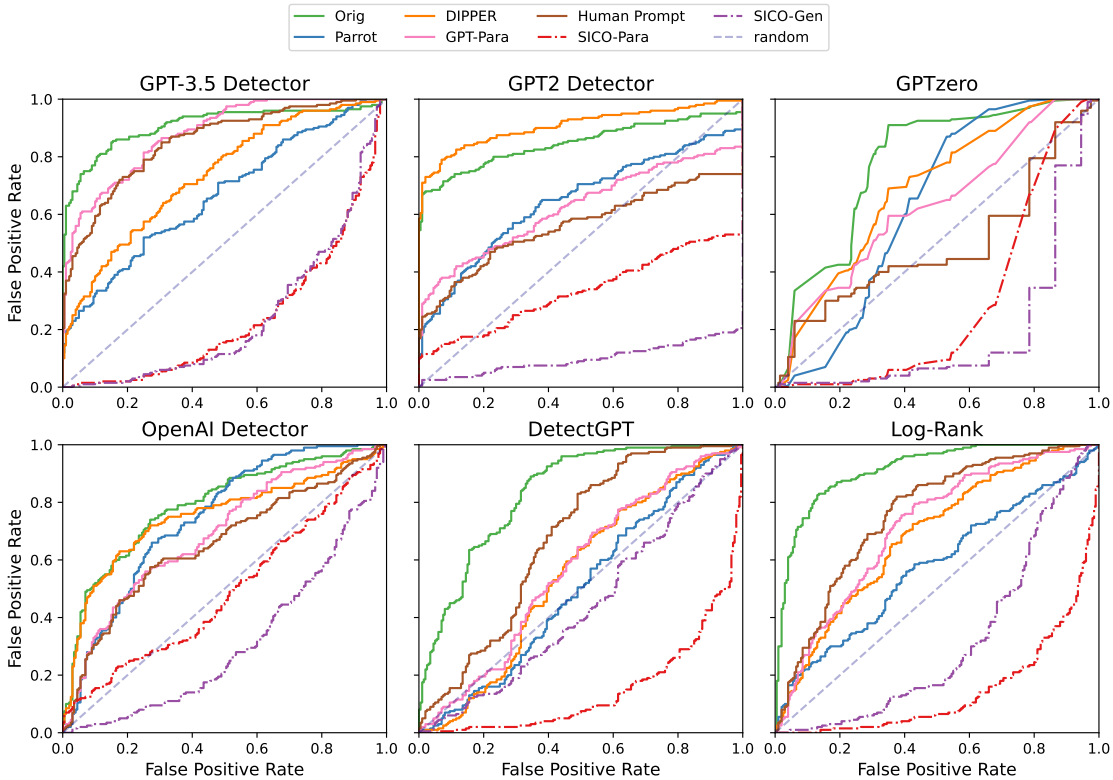


Figure 2: ROC curves of six detectors on the text generated by different evasion methods on academic writing task.

Experimental settings We set $|\mathbf{X}_{eval}| = 32$, $K = 8$, $N = 6$, and use GPT-3.5, specifically *gpt-3.5-turbo-0301*, as the LLM, where the inference parameters are kept in default. And we use GPT3-D as the proxy detector. Experiments using other LLMs and proxy detectors are presented in Section 5.2.

4.2 Evasion Performance and Analysis

Table 1 presents the performance of SICO and other baselines against six detectors in AUC score. SICO consistently outperforms other baselines by a substantial margin in all cases. Notably, in most cases, SICO reduces the AUC score to less than 0.5, equivalent to the expected performance of a random classifier. Figure 2 shows the ROC curves of evasion methods on academic writing task. One can clearly observe that SICO curves lie below others along different thresholds, often lower than the random classifier curve. More evasion results including ROC cures and detection rates are shown in Appendix H.

One interesting trend is that SICO-Para consistently outperforms SICO-Gen against statistical detectors, i.e., DetectGPT and Log-Rank. We speculate this performance difference comes from the varying influence of the prompt on the generated text between the two methods. In SICO-Para, the distribution of generated text is largely influenced by the original AI-generated text, which is in the prompt. However, in SICO-Gen, the distribution of generated text depends more on the previously generated text. Given that statistical detectors have access to the newly generated text but not the prompt, their estimation of token probability becomes less accurate for SICO-Para text, thus misleading the detection. It might also explain why GPT-Para can reduce the performance of statistical detectors.

4.3 Human Evaluation

From the users' perspective, using AI-generated text goes beyond evading detection systems; the usability of text is equally critical. For example, for academic writing task, users expect the text to be readable, properly formatted, and relevant to the given topic. Therefore, we evaluate the usability of text based on two criteria: readability and task completion rate. For each task, we randomly sample 200 examples generated by four methods (50 per method), including human-written text. Then we ask three human annotators to rate the readability of text on a scale from 1 to 5, and judge if the text accomplishes the task's goal. More details of human evaluation are shown in Appendix D.

As shown in Table 2, both SICO-Gen and SICO-Para demonstrate superior performance over DIPPER in terms of task completion and readability over three tasks. Furthermore, SICO-generated text performs competitively compared with human-written text in both metrics, with a negligible difference less than 0.1. In contrast, DIPPER exhibits inferior performance relative to human-written text, particularly with a notable 0.27 decline in readability.

Table 2: Human evaluation results. "Avg.D." represents the average difference between the results achieved by the evasion method and the results achieved by human-written text on the three tasks. The best value of each task is set **bold**.

Method	Readability ↑				Task Completion Rate % ↑			
	Writing	QA	Review	Avg.D. ↑	Writing	QA	Review	Avg.D. ↑
DIPPER	3.52	4.12	3.42	-0.27	70.6	100	61.6	-13.3
SICO-Para	3.68	**4.36**	3.58	-0.09	82.0	100	72.4	-5.9
SICO-Gen	3.84	4.28	**3.70**	**-0.02**	93.6	100	69.6	**-2.9**
Human-Written	**3.92**	**4.36**	3.60	-	**98.2**	100	**73.8**	-

4.4 Real-life Experiments

To further assess the applicability of SICO in real-world, we simulate one potential misuse case of LLM, where SICO is deployed as an automatic reply bot on Reddit[2], a popular online social platform. We wrote a script to monitor the new posts submitted in the community for asking questions and used GPT-3.5 equipped with SICO-Para to automatically reply them. The prompt we used is trained for question answering task. On Reddit, except for giving comments, users can express their approval of other's responses by clicking the "like" or "dislike" button. To minimize the social impact, we limit the number of responses to 40 and deleted them after collecting results.

Table 3: Quantitative results of Reddit test.

Like %	Dislike %	Comment %
40%	2.50%	12.5%

The quantitative results in Table 3 demonstrate that users generally react positively to the text from SICO. Specifically, 40% of the responses from SICO receive "likes" from Reddit users, significantly higher than the 2.5% that are disliked. The remaining 57.5% of responses go unnoticed, which is common in social media. Besides, in 12.5% of cases, users left additional comments to express their approval or to ask further questions, demonstrating their engagement and interest in SICO's response. Figure 3 presents two SICO's responses that got approval from users, as indicated by "likes" and comments.

4.5 Cost Efficiency

In terms of the data prerequisite, SICO only needs $K + |\mathbf{X}_{eval}|$ human-written input-output examples to build prompt, which is $8 + 32 = 40$ in the experiments. The other AI-generated text can be produced by LLM leveraging these human samples. Furthermore, SICO offers the advantage of low cost for prompt construction.

[2]https://www.reddit.com/

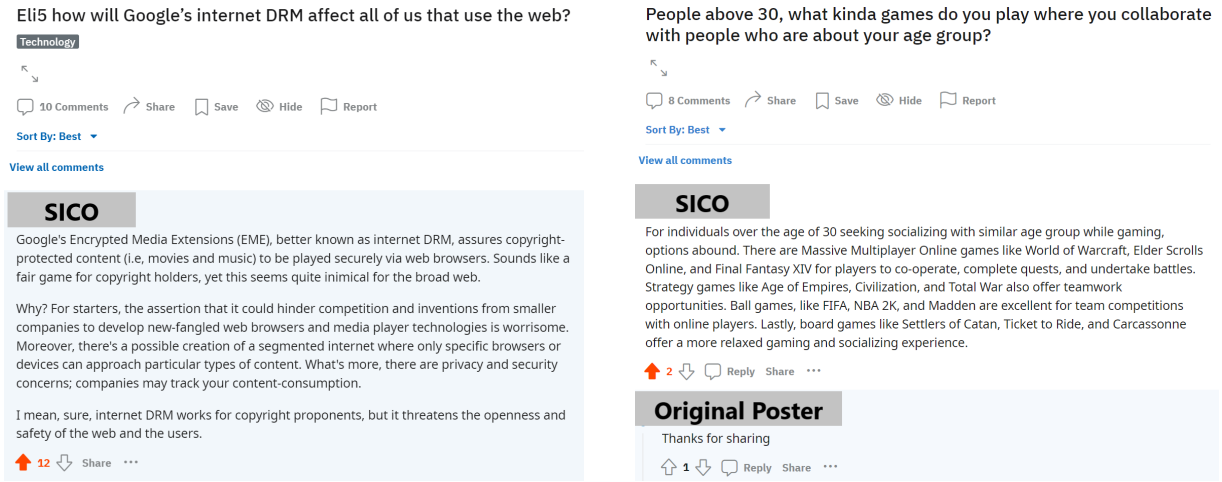


Figure 3: Screenshots of replies from SICO that got approved by Reddit users. (Left) Approval is shown by the number of "like"s. (Right) The original poster express thanks to SICO's response.

Based on three repeated runs, the actual USD costs of SICO-Para are 1.04 ± 0.04, 1.08 ± 0.05, and 0.75 ± 0.04 for Writing, QA, Review tasks, respectively.

5 Further Experiments

5.1 Ablation Study

We conducted an ablation study over academic writing task to to evaluate the contribution of individual components within the SICO framework. "Human-ICE" denotes the approach where human-written text is directly utilized as the in-context example for constructing the prompt. "w/o feature" and "w/o ICE" refer to the prompts without feature text and the optimized in-context examples, respectively. "w/o OPT" represents the initial prompt before optimization (see Line 3 in Algorithm 1). In our experiment, we explore SICO-Para on three types of detectors: GPT3-D, OpenAI-D and DetectGPT.

Results in Table 4 shows that directly using human-written text is ineffective, even making the detection more accurate. We speculate that the human-written examples are too heterogeneous and characterized in multiple ways, so LLM cannot effectively learn their attributes. Besides, the importance of feature text is comparatively less than that of optimized in-context examples. Furthermore, the result reveals the significant role of the optimization step in SICO. Using in-context examples that are not optimized is essentially equivalent to not using any in-context examples.

Table 4: The AUC scores of detectors on text generated by different methods. "–" indicates the case where no evasion method is used. 'AVG' represents the average AUC scores across detectors.

Method	GPT3-D	OpenAI-D	DetectGPT	AVG
–	0.908	0.789	0.834	0.844
Human-ICE	0.918	0.825	0.847	0.863
SICO-Para	0.239	0.488	0.149	0.292
w/o feature	+0.106	+0.072	+0.051	+0.076
w/o ICE	+0.359	+0.133	+0.411	+0.301
w/o OPT	+0.361	+0.154	+0.364	+0.293

5.2 SICO with Different Proxy Detectors and LLMs

As described in Section 3, SICO requires a proxy detector and a LLM to construct a prompt. In this experiment, we explore the performance of SICO-Para on writing task, using three types of proxy detectors: (1) training-based model GPT-3.5 detector, (2) API detector GPTzero, and (3) statistical method DetectGPT. For different LLMs, we adopt Vicuna-13B (Chiang et al., 2023), an open-source chatbot fine-tuned from LLaMa (Touvron et al., 2023). Results in Table 5 show that SICO maintains a high degree of detection evasion performance, regardless of proxy detectors or LLMs. In most cases, SICO manages to reduce the AUC of detectors by approximately 0.4.

Table 5: The AUC scores of SICO using different proxy detectors and LLMs on writing task. The first line indicates the performance without applying any evasion method.

LLM	Proxy Detector	GPT3-D	GPT2-D	GPTzero	OpenAI-D	DetectGPT	Log-Rank
-	-	0.908	0.848	0.779	0.789	0.834	0.914
GPT-3.5	GPT3-D	0.239	0.332	0.290	0.488	0.149	0.147
GPT-3.5	GPTzero	0.549	0.143	0.299	0.458	0.429	0.493
GPT-3.5	DetectGPT	0.291	0.361	0.244	0.483	0.174	0.173
Vicuna-13B	GPT3-D	0.393	0.777	0.380	0.690	0.337	0.375

5.3 Examples of the generated text

Table 6 presents the fake reviews generated by SICO-Gen and SICO-Para. The generated text shows high readability and fulfill the task's goal, successfully mentioning all keywords and generating positive reviews of the specified object. The AI probability, denoted as $\mathcal{P}_{\text{AI}}$ in the table, is determined by the GPT3-D. More examples are shown in Appendix I.

Table 6: Fake reviews generated by SICO. The first line shows the task input of review generation.

Object: Sushi Station **Keywords**: revolving sushi bar, affordable, krab mix, creamy sauce, fresh menu. **Sentiment**: Positive	$\mathcal{P}_{\text{AI}}$
AI: If you're a sushi lover, you have to check out Sushi Station! This revolving sushi bar is not only affordable, but it offers a fresh and delicious menu that will leave you craving more. My personal favorite is the krab mix sushi with its creamy sauce - it's simply heavenly. The staff is friendly and attentive, and the atmosphere is perfect for a casual night out with friends or family. Don't hesitate to give Sushi Station a try - you won't be disappointed!	0.99
SICO-Para: Sushi enthusiasts, please take note: Sushi Station provides an unbeatable revolving sushi bar experience at wallet-friendly prices. Their menu is of the freshest quality and tastes nothing short of delicious. My top pick has to be the krab mix sushi; its creamy sauce is pure bliss. Additionally, their friendly staff provided excellent service, while the casual ambiance proved ideal for a relaxed night out with friends or family. Don't hesitate to give Sushi Station a try, you won't regret it!	0.00
SICO-Gen: I recently tried out Sushi Station's revolving sushi bar and I must say, it was an amazing experience. The affordable pricing along with the wonderful krab mix and creamy sauce they offer is what really made it stand out from the rest. The menu was constantly updated with fresh and delicious options making it hard to choose just one. If you're looking for a great sushi experience, Sushi Station is definitely worth a visit.	0.04

6 Conclusion

In conclusion, we have proposed a novel in-context learning approach, SICO, designed to guide LLMs in generating text that can effectively evade detectors. Our extensive experiments on evasion demonstrate the superior performance of SICO, which significantly reduces the detection capabilities of existing AI text detectors across three tasks. A comprehensive human evaluation shows SICO text can achieve human-level readability and task completion rates. The experiment in the wild shows the functionality of SICO in real life.

Looking ahead, SICO could act as a data generator and be integrated during the training phase of AI detectors, which may enhance their robustness. Furthermore, the core concept of SICO, namely, substitution-based in-context learning, could be applied to a variety of text generation tasks. We believe that this opens up new avenues for future research in the fields of text generation and in-context learning.

7 Ethics statement

The intention of this paper is not to offer a potential method for evading AI-generated text detection systems. Instead, our aim is to raise awareness within the broader community about the vulnerabilities of existing AI-generated text detection systems to such technology. As many LLMs are public available and free to use, many people can adjust their prompt and generate text that evades these detectors. Given the ease of evasion illustrated in this study, these detectors are not robust yet.

We hope the research community can stress test their detectors against text generated by carefully crafted prompt, and create more robust detectors in the future. To support the research in this field, we will make our training methods and relevant data/code publicly available.

References

Sahar Abdelnabi and Mario Fritz. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pp. 121–140. IEEE, 2021. doi: 10.1109/SP40001.2021.00083. URL `https://doi.org/10.1109/SP40001.2021.00083`.

David Ifeoluwa Adelani, Haotian Mai, Fuming Fang, Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen. Generating sentiment-preserving fake online reviews using neural language models and their human- and machine-based detection. In Leonard Barolli, Flora Amato, Francesco Moscato, Tomoya Enokido, and Makoto Takizawa (eds.), *Advanced Information Networking and Applications - Proceedings of the 34th International Conference on Advanced Information Networking and Applications, AINA-2020, Caserta, Italy, 15-17 April*, volume 1151 of *Advances in Intelligent Systems and Computing*, pp. 1341–1354. Springer, 2020. doi: 10.1007/978-3-030-44041-1_114. URL `https://doi.org/10.1007/978-3-030-44041-1_114`.

Daria Beresneva. Computer-generated text detection using machine learning: A systematic review. In Elisabeth Métais, Farid Meziane, Mohamad Saraee, Vijayan Sugumaran, and Sunil Vadera (eds.), *Natural Language Processing and Information Systems - 21st International Conference on Applications of Natural Language to Information Systems, NLDB 2016, Salford, UK, June 22-24, 2016, Proceedings*, volume 9612 of *Lecture Notes in Computer Science*, pp. 421–426. Springer, 2016. doi: 10.1007/978-3-319-41754-7_43. URL `https://doi.org/10.1007/978-3-319-41754-7_43`.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: long form question answering. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28-August 2, 2019, Volume 1: Long Papers*, pp. 3558–3567. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1346. URL https://doi.org/10.18653/v1/p19-1346.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. GLTR: statistical detection and visualization of generated text. In Marta R. Costa-jussà and Enrique Alfonseca (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 3: System Demonstrations*, pp. 111–116. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-3019. URL https://doi.org/10.18653/v1/p19-3019.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A. Smith, and Luke Zettlemoyer. Demystifying prompts in language models via perplexity estimation. *CoRR*, abs/2212.04037, 2022. doi: 10.48550/arXiv.2212.04037. URL https://doi.org/10.48550/arXiv.2212.04037.

Yunhao Gou, Tom Ko, Hansi Yang, James T. Kwok, Yu Zhang, and Mingxuan Wang. Leveraging per image-token consistency for vision-language pre-training. *CoRR*, abs/2211.15398, 2022. doi: 10.48550/arXiv.2211.15398. URL https://doi.org/10.48550/arXiv.2211.15398.

Alexei Grinbaum and Laurynas Adomaitis. The ethical need for watermarks in machine-generated language. *CoRR*, abs/2209.03118, 2022. doi: 10.48550/arXiv.2209.03118. URL https://doi.org/10.48550/arXiv.2209.03118.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *CoRR*, abs/2301.07597, 2023. doi: 10.48550/arXiv.2301.07597. URL https://doi.org/10.48550/arXiv.2301.07597.

Dirk Hovy. The enemy in your own camp: How well can we detect statistically-generated fake reviews - an adversarial study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. The Association for Computer Linguistics, 2016. doi: 10.18653/v1/p16-2057. URL https://doi.org/10.18653/v1/p16-2057.

Hyuhng Joon Kim, Hyunsoo Cho, Junyeob Kim, Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. Self-generated in-context learning: Leveraging auto-regressive language models as a demonstration generator. *CoRR*, abs/2206.08082, 2022. doi: 10.48550/arXiv.2206.08082. URL https://doi.org/10.48550/arXiv.2206.08082.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. *CoRR*, abs/2301.10226, 2023. doi: 10.48550/arXiv.2301.10226. URL https://doi.org/10.48550/arXiv.2301.10226.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *CoRR*, abs/2303.13408, 2023. doi: 10.48550/arXiv.2303.13408. URL https://doi.org/10.48550/arXiv.2303.13408.

Thomas Lavergne, Tanguy Urvoy, and François Yvon. Detecting fake content with relative entropy scoring. In Benno Stein, Efstathios Stamatatos, and Moshe Koppel (eds.), *Proceedings of the ECAI'08 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse, Patras, Greece, July 22, 2008*, volume 377 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008. URL https://ceur-ws.org/Vol-377/paper4.pdf.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 3214–3252. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.229. URL https://doi.org/10.18653/v1/2022.acl-long.229.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

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. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

Fatemehsadat Mireshghallah, Justus Mattern, Sicun Gao, Reza Shokri, and Taylor Berg-Kirkpatrick. Smaller language models are better black-box machine-generated text detectors. *CoRR*, abs/2305.09859, 2023. doi: 10.48550/arXiv.2305.09859. URL https://doi.org/10.48550/arXiv.2305.09859.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *CoRR*, abs/2301.11305, 2023. doi: 10.48550/arXiv.2301.11305. URL https://doi.org/10.48550/arXiv.2301.11305.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. MTEB: massive text embedding benchmark. In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pp. 2006–2029. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.eacl-main.148.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 1864–1874. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-acl.146. URL https://doi.org/10.18653/v1/2022.findings-acl.146.

OpenAI. Openai ai text classifier, January 2023. URL https://beta.openai.com/ai-text-classifier.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv e-prints*, art. arXiv:1606.05250, 2016.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022, pp. 2655–2671. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.naacl-main.191. URL https://doi.org/10.18653/v1/2022.naacl-main.191.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. Can ai-generated text be reliably detected? *CoRR*, abs/2303.11156, 2023. doi: 10.48550/arXiv.2303.11156. URL https://doi.org/10.48550/arXiv.2303.11156.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, and Jasmine Wang. Release strategies and the social impacts of language models. *CoRR*, abs/1908.09203, 2019. URL http://arxiv.org/abs/1908.09203.

StackOverflow. Temporary policy: Chatgpt is banned, 2023. URL https://meta.stackoverflow.com/questions/421831/temporary-policy-chatgpt-is-banned.

Chris Stokel-Walker. Ai bot chatgpt writes smart essays-should academics worry? *Nature*, 2022.

Edward Tian. Gptzero: an ai detector, 2023. URL https://gptzero.me/.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In Marti A. Hearst and Mari Ostendorf (eds.), *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*. The Association for Computational Linguistics, 2003. URL https://aclanthology.org/N03-1033/.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 9051–9062, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/3e9f0fc9b2f89e043bc6233994dfcf76-Abstract.html.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 649–657, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/250cf8b51c773f3f8dc8b4be867a9a02-Abstract.html.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

A Implementation Details

A.1 SICO

A.1.1 Feature extraction

In feature extraction step, we instruct LLM to extract 5 features and calculate the utility score $\mathcal{U}$ of prompts encompassing each of these features. Then we select the feature with the highest utility for further steps. The

goal of this step is to find a good feature to accelerate process, and make the whole process stable. Because sometimes LLM cannot extract useful features to evade detectors. The pseudo-code illustrating this selection process is outlined in Algorithm 3. Table 7 presents the prompt template used for feature extraction. Here, K text pairs generated by AI and Human are positioned within their respective positions. Table 20 shows the examples for feature extracted by LLM.

Algorithm 3 Feature selections

Require: list of features T_{feature}, prompt utility function $\mathcal{U}(\cdot)$
Initialize $t^*_{\text{feature}} \leftarrow \emptyset$
Initialize $\mathcal{U}_{max} \leftarrow -\infty$
for each feature $t_{\text{feature},i}$ in T_{feature} **do**
  Construct prompt $p_i \leftarrow t_{\text{feature},i} \oplus p_{\text{task}}$
  if $\mathcal{U}(p_i) > \mathcal{U}_{max}$ **then**
    $t^*_{\text{feature}} \leftarrow t_{\text{feature},i}$
    $\mathcal{U}_{max} \leftarrow \mathcal{U}(p_i)$
  end if
end for
return t^*_{feature}

Table 7: Prompt for feature extraction.

Here are the writings from AI and human:
AI writing: <AI-generated text>
Human writing: <human-written text>
...
What is the key distinct feature of human's writings?
[LLM complete]

LLM consistently extract useful features. To test if LLM can reliably extract useful features, we conducted three separated experiments by running three feature extractions on different sets of human-written text and AI-generated text. We use three different extracted features to guide LLM generation and test the AUC drop after adopting 3 different features compared with the originally generated text on the Writing task. Table 8 shows the results, indicating that LLM consistently extracts useful features for detector evasion from different examples.

Table 8: AUC drop of different features extracted based on different human-written and AI-generated text.

Orig.	Feature 1	Feature 2	Feature 3
0.908	-0.288	-0.261	-0.142

A.1.2 Task instructions

Table 9 shows the actual task instruction p_{task} we used in SICO. As mentioned in Section 3, feature text t_{feature} at first step will be inserted before these task instructions. The "Paraphrase" instruction is p_{para} used in paraphrase generation for substitution (Line 6 of Algorithm 1).

A.1.3 Word Substitution

We employ WordNet synsets to derive synonyms for given words. During optimization of in-context examples, we only substitute content words, namely nouns, verbs, adjectives, and adverbs. Furthermore, we part-of-speech (POS) tag of the synonym to ensure it aligns with the original word. For POS tagging, we utilize the

Stanford POS Tagger (Toutanova et al., 2003). Additionally, to maintain fluency in the modified text after substitution, we employed a pretrained mask language model to exclude synonyms with low likelihood. In experiment we use RoBERTa-base model (Liu et al., 2019).

Table 9: Task instructions of each task.

Task	Task instruction p_{task}
Writing	Based on the description, complete an academic paragraph in human style writings: Prompt: < *task input* > Human: [LLM complete]
QA	Based on the description, answer questions in human style writings: Q: < *task input* > Human: [LLM complete]
Review	Based on the description, write a human review about the given object and keywords, with a specified sentiment: Object, Keywords, Sentiment: < *task input* > Human: [LLM complete]
Paraphrase	Based on the description, rewrite this in human-style writings: Origin: < *original AI-generated text* > Human: [LLM complete]

A.2 Baselines

A.2.1 DIPPER

We choose the best evasion performance parameter setting from the original paper (Krishna et al., 2023), which is 60 for lexical diversity and 60 for re-ordering. And we set sampling temperature to 0.75, following the original implementation.

A.2.2 Human prompt

We carefully design a paraphrase prompt based on the detection idea of GPTzero (Tian, 2023), which distinguish the AI-generated content from Human-written by *perplexity* and *burstiness*, stated by its creator[3]. *Perplexity* is the concept raised in NLP field, which measures how well a language model predicts a text sample. A text with a lower perplexity score indicates that the language model is better at calculating the next word that is likely to occur in a given sequence. On the other hand, *burtiness* basically measures the variation between sentences, including sentence length and structures. The lower the values for these two factors, the more likely it is that a text was produced by an AI. Table 10 shows the prompt we designed.

Table 10: Human-designed prompt to evade AI-generated text detection.

When it comes to writing content, two factors are crucial, "perplexity" and "burstiness". Perplexity measures the complexity of text. Separately, burstiness compares the variations of sentences. Humans tend to write with greater burstiness, for example, with some longer or complex sentences alongside shorter ones. AI sentences tend to be more uniform. Paraphrase the following AI sentence to be human-like, with a good amount of perplexity and burstiness: Orig: < *orignal AI-generated text* > New: [LLM complete]

[3]https://theconversation.com/we-pitted-chatgpt-against-tools-for-detecting-ai-written-text-and-the-results-are-troubling-199774

B Extra Experiments

B.1 Detection capability of human

We conducted a experiment of real-life human detection rate. We randomly selected 150 examples (50 each from AI, SICO-Para, and human responses) from QA task. Four human annotators were then asked to identify whether each answer was AI-generated or human-written. Table 11 shows that SICO remarkably reduces the probability of being recognized by humans. Additionally, the data also reveals that human detection capabilities are not highly accurate compared with machine detectors. Approximately half of AI-generated texts went undetected, and 10% of human-written texts were mistakenly identified as AI-generated.

Table 11: Detection rate from human anotators. 'AI %' indicates the proportion of texts identified as AI-generated by annotators.

Text Source	AI %
AI	54%
SICO-Para	26%
Human	10%

B.2 Semantic preserving

We measure semantic similarity using t5-based sentence encoder (Ni et al., 2022), which leads in semantic text similarity task of MTEB benchmark (Muennighoff et al., 2023). Table 12 reports a comparison of the cosine similarity of text before and after paraphrasing by different methods. Our methods successfully preserves the semantic meaning during paraphrasing, and beats the other specifically trained paraphraser. Paraphrasing directly using GPT-3.5 yields the most promising results.

Table 12: Cosine similarity between original AI-generated text and their respective paraphrased versions using different methods. The best scores in each task are highlighted in **bold**.

Method	Writing	QA	Review
Parrot	0.964	0.961	0.966
DIPPER	0.956	0.941	0.940
GPT-Para	**0.986**	**0.982**	**0.987**
Human Prompt	0.979	0.968	0.978
SICO-Para	0.976	0.964	0.971

B.3 Consistent effectiveness of SICO

SICO consistently procures optimal detection evasion prompts, irrespective of the diversity in AI-human text pairs and randomized samples drawn from the LLMs. This demonstrates the effectiveness of SICO in various initial conditions and settings, highlighting its applicability to diverse scenarios. Figure 4 presents the AI probability of the paraphrased training data, generated by LLM using the current best prompt, denoted as $\mathcal{U}(p^*)$ in Equation 1. SICO successfully optimizes the initial prompt (at step 0) and converges to a low level in three separate trials across all three tasks, with different AI-human text pairs.

B.4 SICO performs better against more capable detectors

We use the detectors' performance on the original AI-generated text to represent their base capability. The SICO advancing performance is measured by the AUC difference between best of SICO-Para or SICO-Gen and the best-performing paraphraser baselines. The statistical Pearson correlation is 0.47 with a p-value of 0.048, indicating a moderate positive correlation.

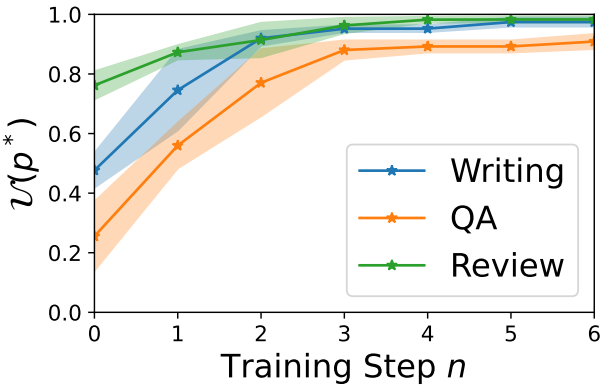


Figure 4: The trajectory of the $\mathcal{U}(p^*)$ during prompt optimization. This plot is derived from three distinct training runs on three tasks.

C Detectors

In this section, we introduce the mechanism and settings of the detectors in our experiments.

C.1 GPT-3.5 Detector

GPT-3.5 detector is trained on Human ChatGPT Comparison Corpus (HC3) dataset (Guo et al., 2023), which including answers generated by ChatGPT and human. English-version of HC3 dataset contains five splits: reddit-eli5, OpenQA, Wikipedia, medicine and finance. The base model is RoBERTa-base and we use the model that only take answers as input.

C.2 GPT2 Detector

GPT-2 detector is obtained by fine-tuning a RoBERTa model with the outputs of the 1.5B-parameter GPT-2 model. The detector and the GPT-2 output dataset are both provided by OpenAI. Although it is trained on GPT-2 outputs, our experiments shows that it can effectively detect text from GPT-3.5.

C.3 DetectGPT

DetectGPT identifies if a text is generated by a model by observing a unique characteristic: AI-generated text tends to be in areas where the language model's log probability function has a negative curve. Here's how it works: It first perturbs the input text and constructs multiple perturbations of input text. The perturb step is completed by a mask language model. Then it checks the log probability of these variations against the original text by a inner language model. Finally, the text is considered AI-generated if the log probability of the original input text is significantly higher than the log probability of perturbations.

We use z-score implementation of DetectGPT and set sample number to 100 and replacement ratio to 0.3. The inner language model is GPT2-medium and the mask language model is t5-large (Raffel et al., 2020).

C.4 Log-Rank

Log-Rank method employs the mean prediction rank of each token in a text. Specifically, for each word in a text, given its previous context, it can calculate the absolute rank of this word by an inner language model. Then, for a given text, we compute the score of the text by averaging the rank value of each word. Note that a smaller score denotes the text is more likely to be machine-generated. In experiment, we use GPT2-medium to calculate the rank of tokens to align with the implementation of DetectGPT.

C.5 GPTzero

GPTzero is a recently proposed commercial detector, employed by many users and oragnizations. As claimed in its websites[4], GPTzero can be used to detect the outputs from detect ChatGPT, GPT4, Bard, LLaMa, and other AI models. GPTZero is the leading AI detector for checking whether a document was written by a large language model such as ChatGPT. GPTZero detects AI on sentence, paragraph, and document level. GPTzero was trained on a large, diverse corpus of human-written and AI-generated text, with a focus on English prose. GPTZero has served over 2.5 million users around the world, and works with over 100 organizations in education, hiring, publishing, legal, and more.

C.6 OpenAI Detector

OpenAI detector is officially provided by OpenAI after the release of ChatGPT. Although it only offers a web interface, we adopted the API implementation from (Krishna et al., 2023), which uses "model-detect-v2" in the OpenAI API. Through reverse engineering of the website, we determined that the web interfaces indeed call this model.

On July 20, 2023, OpenAI discontinued this detector, "due to its low rate of accuracy."[5]. Considering the discontinuation of the OpenAI detector aligns with our findings, we choose to present the results of it in our paper, though it is out of date.

D Human Evaluation Details

For each text, we show two questions for the human annotator. In terms of readability, we present the human annotator five options, with the scale of 1 to 5. The higher the value, the more readable of the presented texts. The question is identical for three tasks. The actual question and options are presented in Table 13. For task completion rate, we design three task-specific questions, as show in Table 14. Figure 6 shows the interface of our annotation platform. We estimated that the evaluation time of each text ranges from 60 to 90 seconds.

Table 13: Question and options designed for readability.

Question	On a scale of 1-5, how would you rate the readability of the given text?
Options	1: Very difficult to read and understand. 2: Difficult to read, need extra time to understand. 3: Neutral. 4: Easy to read. 5: Very clear and easy to read.

Human evaluation on Parrot. As Parrot method performs better than DIPPER in the Writing task, we conducted a small experiment to evaluate the usability of text generated text, similar to Table 2. We randomly sampled 120 examples (40 for Parrot, SICO-Para and SICO-Gen) from the writing task and asked two human annotators to evaluate them. The experiment result in Table 15 shows that SICO still outperforms Parrot by a large margin.

E Details of Real-life Experiments

We employ the **Praw** package, a Reddit API wrapper, to write the auto-reply bot. On each run, the bot first collect the posts that are newly submitted within 2 hours and extract the title of poster as question. Then the bot will utilize GPT-3.5 to generate the original answer, and use SICO-Para to paraphrase it. And we set rules to filter out personal question like "*Which movie do you like best?*". Because GPT-3.5 will reply like "*I'm sorry, as a language model...*", which will directly explore itself. The prompt we deploy is trained for QA

[4]https://gptzero.me/
[5]https://openai.com/blog/new-ai-classifier-for-indicating-ai-written-text

Table 14: Question and options designed for task completion.

Writing	
Question	Is this academic essay correctly written (No errors and in academic style)?
Options	No, it has some obvious errors or not in academic format.
	Yes, it is correctly written.
QA	
Question	Does the answer relate to the question? (regardless of correctness)
Options	No, not relative.
	Yes, it is relative.
Review	
Question	Does this review impact your decision to choose this service?
Options	No influence.
	Yes, it provides useful information.

Table 15: Human evaluation results of Parrot and SICO.

Method	Readablity	Task Compeletion Rate
Parrot	3.35	70.0
SICO-Para	3.80	82.5
SICO-Gen	**3.90**	**85.0**

task. We only reply in the communities that do not ban AI-generated content: specifically *explainlikeimfive* and *NoStupidQuestions*. We only generate 40 responses and deleted them after collecting the data.

F Datasets

Table 16 presents the prompts we employed to create the initial AI-generated text y_{AI}. For academic writing, we sample Wikipedia paragraphs from SQuAD dataset. Then we give GPT-3.5 the first 30 words of these paragraphs and ask GPT-3.5 to complete the rest.

For open-ended question answering, we sample questions from Eli5 dataset and ask GPT-3.5 to generate answers.

For fake review generation, we first instruct GPT-3.5 to extract the business name and five keywords from human-written reviews from Yelp dataset, and then generate fake reviews based on the extracted information with specified sentiment.

The statistics of three datasets are shown in Table 17

G Evade Watermarking Detection

SICO-Para can also be utilized to evade watermark detection, similarly to paraphrase approach. The watermarking algorithm we applied was introduced by Kirchenbauer et al. (2023), which only requires access to the LLM's logits at each time step to add watermarks. This algorithm operates in three steps:

1. Mark a random subset of the vocabulary as "green tokens"using the hash of the previously generated token as a random seed.

Table 16: Prompt for dataset creation.

Task	Prompt
Writing	Complete academic paragraph with prompt: < *first 30 words of paragraph* >
QA	Answer questions: < *question* >
Rev-Ext	Review: < *human-writtent review* > Show the review's object and 5 key words:
Rev-Gen	Write a review about given object and key words, with specified sentiment: < *output of Rev-Ext* > Review:

Table 17: Average character length of human-writteng text and the original AI-generated text.

	SQuAD	Eli5	Yelp
Human	770.3	794.7	834.3
AI	796.5	580.9	505.1

2. Increase the logit value for every green token by a constant, which denotes the watermark strength.
3. Sample sequences using decoding algorithms.

Verification of this watermark is achievable with blackbox access to the LM and knowledge of the hash function, achieved by tokenizing the text, calculating the standard normal score (z-score) through a hypothesis test, and comparing the observed proportion of green tokens to the expected proportion.

In our experiments, we used the text generated by a watermarked GPT-2, provided by Krishna et al. (2023). We employed the GPT-3.5 detector as proxy detector for training. The AUC and and the detection accuracy associated with various paraphrasing methods are presented in Table 18, where the threshold is set to 2.2 for accuracy measurement.

The results reveal that SICO-Para significantly outperforms other paraphrase techniques in evading watermark detection. Notably, both the AUC score and detection accuracy of SICO-Para are significantly lower than that of other methods. Furthermore, even the simpler baseline model, GPT-Para, proved to be effective at evading the watermark detection.

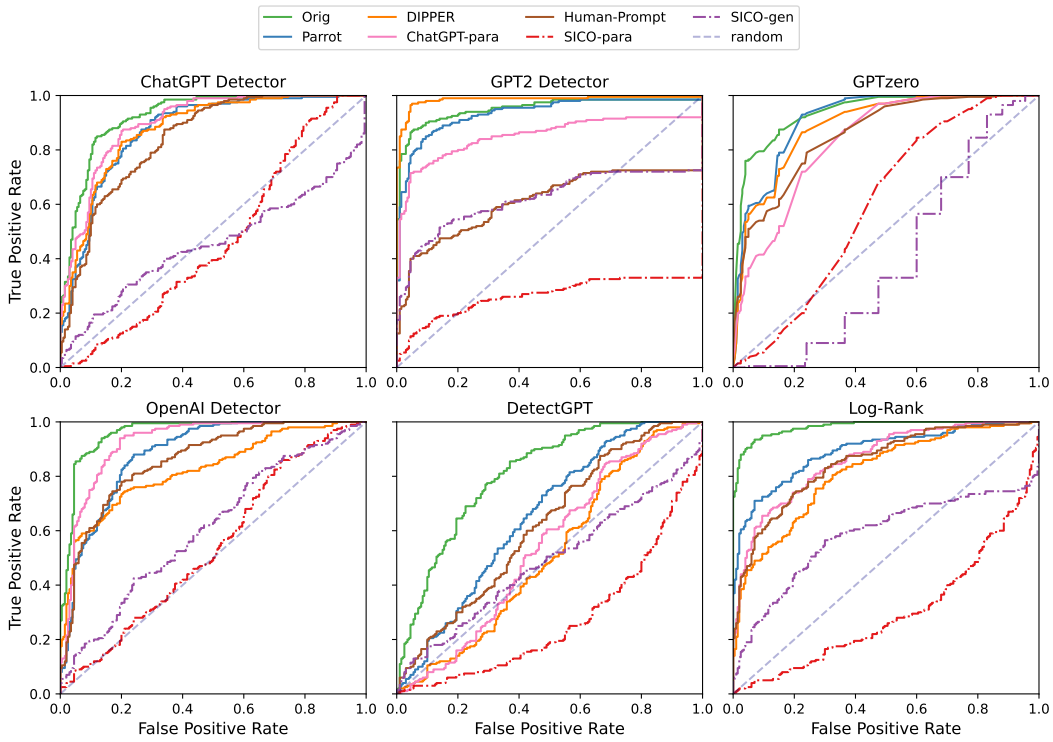
Table 18: Performance of paraphrase methods on watermarking detection.

Method	AUC	Accuracy
AI	0.998	99.0%
Parrot	0.980	88.5%
DIPPER	0.844	33.0%
GPT-Para	0.753	18.0%
SICO-Para	**0.669**	**14.0%**
Human	-	1.0%

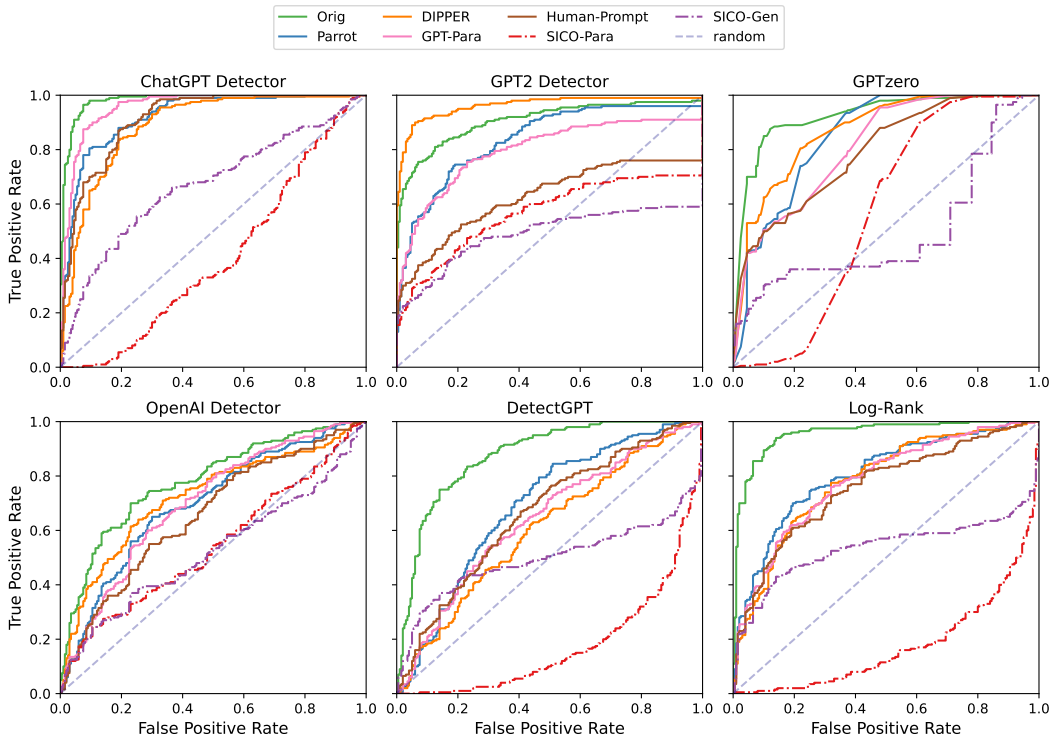
H Evasion Performance

H.1 ROC Curves

Figure 2 shows ROC curves of different detectors presented with text generated by different methods, on open-ended question answering and fake review generation task. SICO curves lie below other baselines.



(a) ROC curves on review generation task.



(b) ROC curves on open-ended question answering task.

Figure 5: ROC curves.

H.2 Detection Accuracy

Given that detection rates highly depend on the selected detection threshold, we establish two thresholds for each detector. The high threshold fixes the *false positive rate* (FPR) at a low level of 0.05, which means only 5% of human-written text will be classified as AI-generated. The low threshold fixes the *true positive rate* (TPR) at a high level of 0.9, based on the original AI-generated text. In this case, 90% of original AI-generated text will be correctly classified. Table 19 shows the detection accuracy on three task. In comparison with other paraphrasing methods, SICO yields the lowest detection rates in most cases.

Table 19: Detection accuracy on three tasks. "AI" refers to the detection rate on the original AI-generated text. The lowest score of each detector is indicated in **bold**, and second-lowest is underlined.

Writing												
Method	High Threshold						Low Threshold					
	GPT3-D	GPT2-D	GPTzero	OpenAI-D	DetectGPT	Log-Rank	GPT3-D	GPT2-D	GPTzero	OpenAI-D	DetectGPT	Log-Rank
AI	71.0%	68.5%	6.5%	33.5%	32.0%	62.0%	90.0%	90.0%	86.5%	90.0%	90.0%	90.0%
Parrot	25.0%	25.5%	0.5%	12.5%	2.5%	16.5%	53.5%	77.0%	46.5%	93.0%	33.5%	36.5%
DIPPER	28.5%	74.5%	2.5%	33.0%	1.0%	14.0%	62.5%	95.0%	64.5%	82.0%	44.0%	51.0%
GPT-Para	58.5%	35.0%	5.0%	14.5%	5.0%	14.0%	85.5%	73.0%	54.0%	84.0%	48.0%	56.5%
Human Prompt	49.0%	26.5%	10.5%	15.5%	10.5%	18.0%	84.0%	63.5%	40.0%	76.0%	64.0%	67.0%
SICO-Para	1.0%	12.5%	**0.0%**	11.5%	**0.0%**	**0.0%**	**5.5%**	40.5%	4.0%	56.5%	**3.5%**	**2.0%**
SICO-Gen	**0.5%**	**2.5%**	1.5%	**2.0%**	3.0%	1.0%	6.0%	**12.0%**	**3.5%**	**32.0%**	27.5%	5.5%
Human	5.0%	5.0%	4.0%	5.0%	5.0%	5.0%	30.5%	64.5%	34.5%	61.0%	38.5%	28.5%

QA												
Method	High Threshold						Low Threshold					
	GPT3-D	GPT2-D	GPTzero	OpenAI-D	DetectGPT	Log-Rank	GPT3-D	GPT2-D	GPTzero	OpenAI-D	DetectGPT	Log-Rank
AI	91.0%	70.0%	70.0%	30.5%	40.5%	78.5%	90.0%	90.0%	89.5%	90.0%	90.0%	90.0%
Parrot	62.0%	49.0%	42.5%	13.5%	7.0%	34.5%	58.0%	78.0%	75.0%	85.5%	62.5%	52.5%
DIPPER	43.5%	86.5%	53.0%	22.0%	7.5%	27.5%	38.5%	96.5%	82.0%	84.0%	46.5%	38.5%
GPT-Para	75.5%	46.5%	42.0%	14.0%	7.0%	32.5%	73.0%	77.5%	61.0%	86.5%	55.0%	45.0%
Human Prompt	56.0%	32.0%	42.5%	**12.5%**	10.5%	31.0%	51.0%	56.5%	61.0%	83.0%	57.0%	47.5%
SICO-Para	**0.0%**	27.0%	**0.5%**	**12.5%**	**0.0%**	**0.0%**	**0.0%**	51.0%	**6.5%**	63.5%	**1.0%**	**0.0%**
SICO-Gen	19.5%	**24.5%**	21.0%	**12.5%**	20.0%	26.0%	17.0%	**47.5%**	36.0%	**61.0%**	45.5%	37.0%
Human	5.0%	5.0%	5.0%	5.0%	5.0%	5.0%	4.5%	29.0%	24.0%	61.5%	33.5%	11.5%

Review												
Method	High Threshold						Low Threshold					
	GPT3-D	GPT2-D	GPTzero	OpenAI-D	DetectGPT	Log-Rank	GPT3-D	GPT2-D	GPTzero	OpenAI-D	DetectGPT	Log-Rank
AI	55.0%	86.5%	76.5%	84.5%	25.5%	90.5%	90.0%	90.0%	87.5%	90.0%	90.0%	90.0%
Parrot	34.0%	78.0%	59.5%	47.0%	7.0%	64.0%	79.0%	85.5%	79.0%	60.0%	70.5%	63.5%
DIPPER	37.0%	96.5%	56.5%	56.0%	3.5%	45.5%	82.5%	98.0%	73.5%	62.0%	45.0%	44.0%
GPT-Para	46.0%	71.5%	36.5%	62.0%	3.5%	53.0%	87.0%	74.5%	52.0%	79.5%	54.5%	50.0%
Human Prompt	32.0%	40.0%	51.0%	47.5%	10.0%	47.0%	69.0%	43.0%	63.5%	62.0%	63.0%	45.5%
SICO-Para	**1.5%**	**11.5%**	3.5%	**9.0%**	**2.0%**	**2.5%**	**12.5%**	**15.0%**	14.0%	**12.0%**	**16.0%**	**2.5%**
SICO-Gen	11.0%	39.5%	**0.5%**	14.0%	11.5%	19.0%	28.0%	42.5%	**0.5%**	20.0%	47.0%	19.0%
Human	5.0%	5.0%	5.0%	5.0%	5.0%	5.0%	20.0%	10.5%	16.5%	12.0%	46.5%	4.5%

I Examples of Text Generated by SICO

The examples of text generated by SICO across three tasks are presented in Tabel 21-23.

Table 20: Examples of features generated by LLM.

Task	Feature
Writing	Human tend to provide more specific details and facts, often including dates, numbers, and names. They also tend to use more complex sentence structures, including semicolons and parenthetical phrases, to convey additional information. However, their writing remains clear and concise, and they avoid unnecessary repetition or wordiness.
QA	Human's writing style is characterized by the provision of specific and technical information. They delve into the science behind a topic or the details of a process. They also challenges or adds nuance to the information, providing alternative explanations or clarifying misconceptions.
Review	Human writings are generally shorter and more to-the-point. They often use bullet points or numbered lists to convey their thoughts. They tend to be more critical and focused on their own personal experiences. They also use informal language and occasionally include personal anecdotes or opinions.

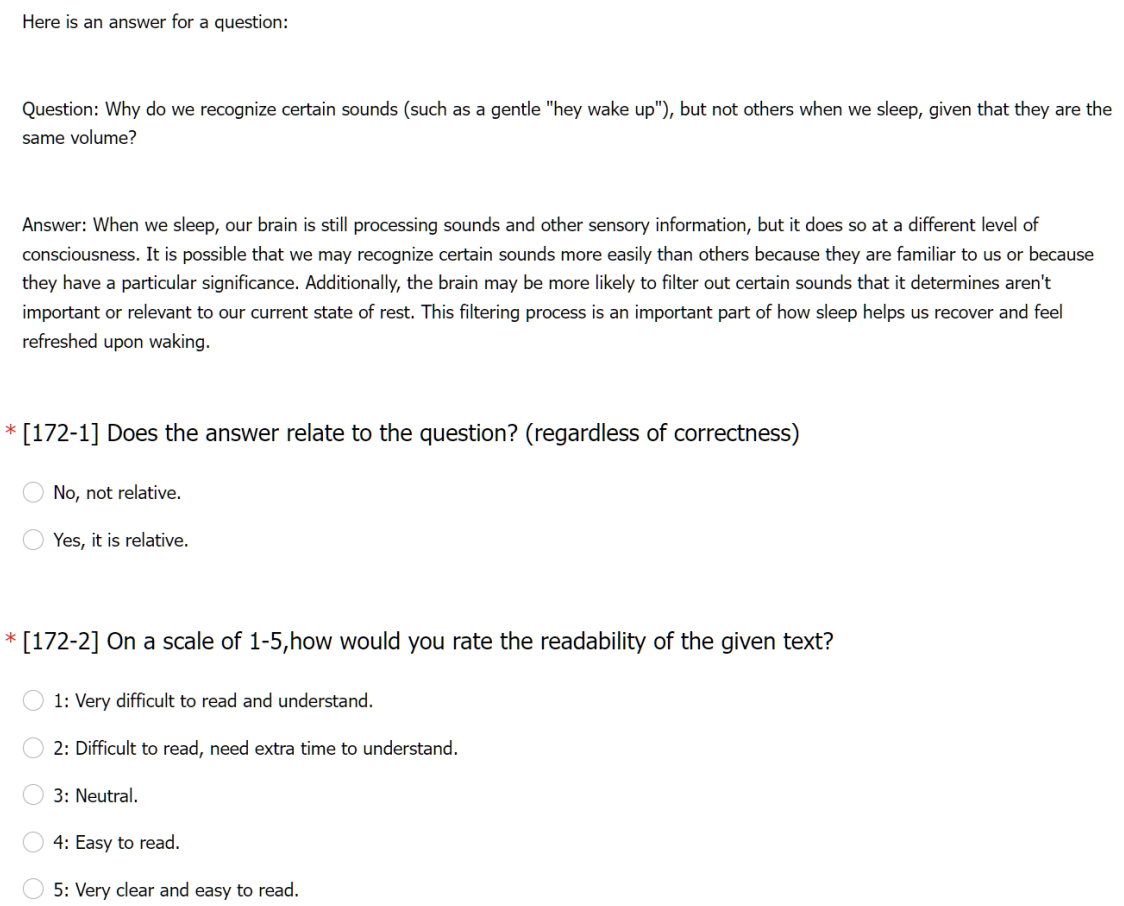


Figure 6: The interface of the annotation platform used in our experiment.

Table 21: Text generated by SICO for open-ended question answering task.

Question: Why does changing screen resolution (more specifically, a custom resolution) will harm your computer?	$\mathcal{P}_{\text{AI}}$
AI: Changing the screen resolution, including a custom resolution, will not harm your computer. It may result in display issues or reduced image quality if the resolution is not compatible with your monitor's capabilities. However, this can be resolved by simply reverting to the previous resolution or finding a suitable one.	0.99
SICO-Para: Don't worry, changing your screen resolution won't harm your computer, even if you decide to go custom. The only trouble is that it may not work well with your monitor and cause image problems or lesser quality. Easy solution though - just go back to the old resolution or find a matching one.	0.00
SICO-Gen: Using custom screen resolutions places a greater burden on system processing units, potentially leading to increased strain and overheating. Moreover, setting a resolution to unsupported display dimensions can put undue strain on graphics hardware, possibly causing irreparable damage. Adjusting to custom resolutions might, therefore, jeopardize the computer, leading to hardware malfunctions, degraded display output quality, and issues with other essential system components.	0.00
Question: How do they calculate how much calories you've burned after an exercise?	$\mathcal{P}_{\text{AI}}$
AI: The number of calories burned during exercise is calculated by various factors such as body weight, the intensity of the exercise, the duration of the activity, and the person's metabolic rate. Some popular fitness trackers and machines use algorithms and formulas based on these factors to estimate the number of calories burned. However, these numbers are only estimates and can vary depending on individual factors such as muscle mass, fitness level, and metabolic rate.	0.99
SICO-Para: If you wanna know how many calories you burn during exercise, there are a few things to keep in mind. Your weight, the intensity of your workout, and how long you're doing it for all come into play, as does your metabolic rate. Popular fitness trackers and machines use fancy calculations based on all these factors to give you an estimate of how many cals you burned. But, listen up, folks - these are only estimates! Your muscle mass, fitness, and metabolism are gonna differ from person to person, so results will vary. Clear?	0.05
SICO-Gen: The calculation of calorie consumption requires assessment of parameters such as exercise type, duration, and intensity, as well as the individual's body measurements. Several formulas exist to compute the total number of calories burned, including the Harris-Benedict equation, which computes resting metabolic rate and establishes parameters for calorie expenditure. Modern heart rate monitors and wearable fitness trackers utilise the latest algorithms to compute calorie consumption by taking into account the heart rate, age, and weight of the user.	0.00

Table 22: Text generated by SICO for fake review generation task.

Object: hotel in Las Vegas **Keywords**: Penthouse, staff, casino, restaurants, excellent experience. **Sentiment**: Positive	$\mathcal{P}_{\text{AI}}$
AI: I had an amazing experience staying at a penthouse in Las Vegas! The staff was incredibly friendly and accommodating, always going above and beyond to make my stay comfortable and enjoyable. The casino was also a great addition, providing endless entertainment and excitement. And when it came to dining options, the variety of restaurants available was impressive. Overall, staying at this hotel was an excellent experience that I won't forget anytime soon. I highly recommend it to anyone looking for a luxurious and unforgettable Las Vegas getaway.	0.75
SICO-Para: Had an amazing stay at a penthouse in Las Vegas. The staff were incredibly friendly and made my stay comfortable and enjoyable. Casino provided endless entertainment and variety of restaurants impressive. Overall, excellent experience and highly recommend for luxurious and unforgettable Las Vegas getaway.	0.03
SICO-Gen: I had the most incredible experience staying at a luxurious penthouse in a Las Vegas hotel. The staff was exceptional and provided top service. The casino and restaurants on the premises were outstanding, making my stay all the more enjoyable. Overall, I highly recommend this hotel for an excellent experience in sin city.	0.06
Object: Tia Rosa restaurant **Keywords**: Mexican food, spicy, bland taste, sweet syrup, tacos. **Sentiment**: Negative	$\mathcal{P}_{\text{AI}}$
AI: I recently visited Tia Rosa restaurant hoping to enjoy some authentic Mexican food. Unfortunately, my experience was far from satisfactory. The food was overly spicy, to the point where it overwhelmed any other flavors that may have been present. On the other hand, the tacos were disappointingly bland and lacked any real taste. To add insult to injury, the sweet syrup that was served with my meal just didn't pair well with the savory dishes. Overall, my visit to Tia Rosa left me feeling disappointed and unsatisfied.	0.84
SICO-Para: Recently went to Tia Rosa restaurant looking for authentic Mexican food. However, experience was unsatisfactory. Food was overly spicy, overpowering other flavours. Tacos lacked taste. Sweet syrup served with meal did not complement savoury dishes. Overall left feeling unsatisfied and disappointed.	0.02
SICO-Gen: My visit to Tia Rosa restaurant was a complete letdown. I was expecting some spicy and flavorful Mexican food, but unfortunately, the dishes were quite bland. Even the tacos, which are usually a safe bet, were a disappointment. The only thing worthy of praise is the sweet syrup, but this alone cannot make up for the overall taste. I won't be returning to Tia Rosa anytime soon.	0.04

Table 23: Text generated by SICO for academic writing task.

Writing Start: The code itself was patterned so that most control codes were together, and all graphic codes were together, for ease of identification. The first two columns (32 positions) were reserved	$\mathcal{P}_{\text{AI}}$
AI: The code itself was patterned so that most control codes were together, and all graphic codes were together, for ease of identification. The first two columns (32 positions) were reserved for control codes, while the remaining 94 columns (94 positions) were designated for graphic codes. This pattern allowed for efficient processing and recognition of the different types of codes. Additionally, the code was designed to be easily modifiable, with the ability to add or remove codes as needed. These features made the code a practical and versatile tool for programming and data processing tasks.	0.99
SICO-Para: The code's layout was systematically arranged for user convenience, with the primary objective of arranging the control and graphic codes within proximal access to facilitate swift identification. The layout allotted the first two columns, constituting a total of 32 positions, solely for control codes, while the remaining 94 columns (94 positions) were allocated for graphic codes, promoting seamless recognition and expediting processing operations. Furthermore, the code's malleability was considered a paramount attribute - the code was designed to be effortlessly modified, accommodating the addition and deletion of codes as per necessity. These inherent characteristics rendered the code an extremely pragmatic and adaptable tool for a wide range of programming and data processing applications.	0.00
SICO-Gen: In order to facilitate easy and efficient use, the code was structured in a manner wherein control codes and graphic codes were distinctly grouped together, allowing for streamlined identification. This design allowed the first two columns (equivalent to 32 positions) to be exclusively reserved for control codes, ensuring that they were easily and rapidly located. The following columns, including columns three to seven, were allocated for graphic codes - providing ample space to incorporate them in a comprehensive manner. By distributing codes systematically and allocating ample space for each grouping, the code structure was organized for optimal use and readability. The efficiency of this coding scheme underscores its practical and logistical utility, while its unique design contributes to its widely accepted success and usage in various computer systems. The contextual background supplied enriches the readers' understanding of the code structure's significance and its potential impact on a system's proficiency.	0.02
Writing Start: Germany has a comprehensive school known as the Gesamtschule. While some German schools such as the Gymnasium and the Realschule have rather strict entrance requirements, the Gesamtschule does not have	$\mathcal{P}_{\text{AI}}$
AI: Germany has a comprehensive school known as the Gesamtschule. While some German schools such as the Gymnasium and the Realschule have rather strict entrance requirements, the Gesamtschule does not have any such requirements. The Gesamtschule is a type of school that caters to students with varying academic abilities and backgrounds, and provides them with an equal opportunity for education. This school system aims to promote social integration and reduce educational inequality by offering a broad curriculum and flexible educational pathways. In addition, the Gesamtschule promotes the development of students' individual strengths and talents through its inclusive and cooperative learning environment. Overall, the Gesamtschule represents a unique approach to education that prioritizes equal access and opportunity for all students.	0.99
SICO-Para: Germany boasts a comprehensive educational institution known as the Gesamtschule. Unlike typical German schools such as the Gymnasium and Realschule, which impose strict entry prerequisites, the Gesamtschule embraces an inclusive admission criterion, ensuring equal education opportunities for students of diverse academic abilities and background. The Gesamtschule represents a collaboration between pupils with different skill sets paired with an extensive and elastic curriculum, thereby striving to promote social homogeneity and curb academic inequalities. Furthermore, the Gesamtschule's inclusive and cooperative learning environment seeks to stimulate the cultivation of student strengths while also encouraging teamwork. Altogether, the Gesamtschule epitomizes an innovative educational frontier that prioritizes equality and accessibility for all students.	0.01
SICO-Gen: An outstanding feature of the German education system is the Gesamtschule, which is a holistic or comprehensive school. In comparison to Gymnasium or Realschule where students are subject to strict requirements to gain admission, entrance to the Gesamtschule is less rigorous. This practice makes the school system more inclusive and allows students from diverse backgrounds to have access to quality education, fostering social unity. At Gesamtschule, students have the liberty to move between different academic tracks based on their demonstrated skills and interests, hence having a more personalized education experience. A blended curriculum is used; it includes theoretical and practical activities intended to prepare students for their future careers or academic pursuits. The Gesamtschule is a pioneering framework in the German schooling system because it promotes equal opportunities in education, educational diversity, and at the same time equips the students with skill sets vital for their future endeavors. The contextual background information provides a clear insight into the working of the Gesamtschule and underscores the importance of creating such models in fostering a more equitable and inclusive society.	0.00