TempParaphraser: "Heating Up" Text to Evade AI-Text Detection through Paraphrasing

Anonymous ACL submission

Abstract

The widespread adoption of large language models (LLMs) has increased the need for reliable AI-text detection. While current detectors perform well on benchmark datasets, we identify a critical vulnerability: increasing the temperature parameter during inference significantly reduces detection accuracy. Based on this weakness, we propose TempParaphraser, a simple yet effective paraphrasing framework that simulates high-temperature sampling effects through multiple normal-temperature generations, effectively evading detection. Experiments show that TempParaphraser reduces detector accuracy by an average of 97.3% while preserving high text quality. We also demonstrate that training on TempParaphraser-augmented data improves detector robustness. All resources are publicly available to support future research.

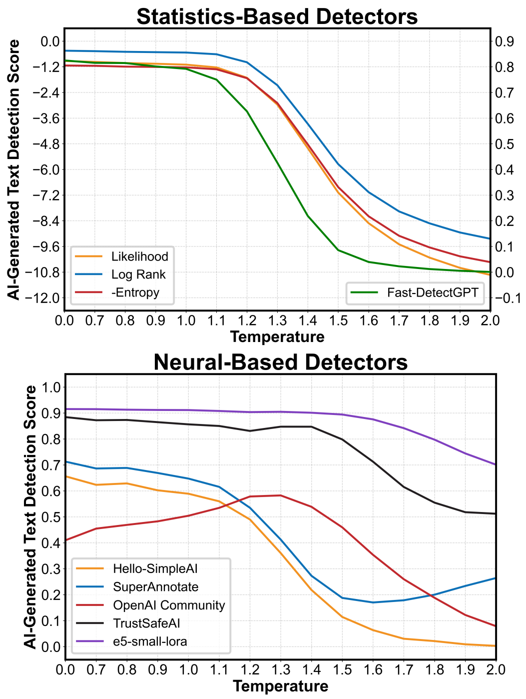


Figure 1: **Effect of temperature on AI-text detectors.** As the temperature increases during LLM inference, both statistical-based and neural-based detectors show lower confidence in identifying the text as AI-generated. Details about these detectors are in Appendix A.

1 Introduction

Large Language Models (LLMs) have significantly enhanced productivity across various fields including news reporting, story creation, and academic research (M Alshater, 2022; Yuan et al., 2022; Christian, 2023). However, their rapid deployment raises concerns about their misuse in creating fake news, malicious reviews, and facilitating academic dishonesty (Ahmed et al., 2021; Adelani et al., 2020; Lund et al., 2023; Lee et al., 2023). In response, AI-text detection technologies have been developed to differentiate between human and AI-generated texts (Mitchell et al., 2023; Bao et al., 2024; Guo et al., 2023).

While current detectors show promising results on benchmark datasets (Mitchell et al., 2023; Bao et al., 2024; Guo et al., 2023), recent studies (Sadasivan et al., 2023; Krishna et al., 2023; Zhou et al., 2024) have explored attack strategies against AI-text detectors, successfully misleading their predictions. Studies (Ippolito et al., 2020; Fishchuk and Braun, 2023; Pu et al., 2023; Dugan et al., 2024) have shown that simple adjustments to sampling parameters, such as top-p, repetition penalty, and temperature, can affect the performance of detectors.

In this paper, we focus on the impact of the temperature parameter. Our experiments show that increasing the temperature significantly reduces the confidence scores of AI-text detectors, making AI-generated text more difficult to identify (see Figure 1). Further analysis reveals a fundamental limitation of current detection methods: **detectors rely on specific statistical patterns in text distribution, which can be disrupted by the randomness introduced through higher temperature set-**

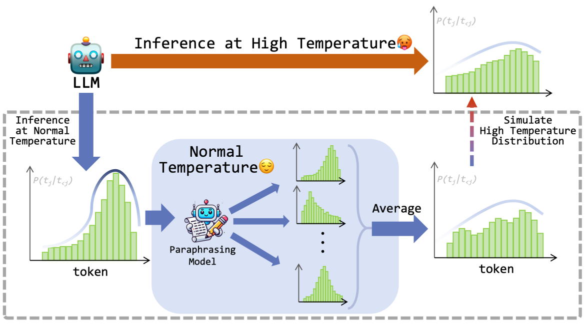


Figure 2: **Our main idea** is shown in the dashed box: Using normal temperature for multiple independent samplings simulates the smoother distribution of high-temperature generation, leading to increased output randomness.

tings (see Section 3.2 for detail analysis). Although higher temperatures can decrease detection accuracy, their direct application during inference often leads to a noticeable decline in text quality (Peeperkorn et al., 2024) (see Appendix B for details on the relationship between temperature and text quality). As a result, previous research has generally focused on temperature settings within a narrow range (Fishchuk and Braun, 2023), where the effect of randomness on detection performance is less significant. Consequently, this vulnerability has largely gone unnoticed.

To further explore this vulnerability, we introduce a simple yet effective framework, **TempParaphraser**, designed to evade detection. TempParaphraser operates as a post-processing tool. It **Temp**orarily stores the original text generated by LLMs, paraphrases it and outputs an optimized version that can evade detection.

This framework incorporates a paraphrasing model fine-tuned from an LLM using synthetic data. As shown in Figure 2, the TempParaphraser framework simulates the smoother distribution in high-**Temp**erature generation by producing multiple paraphrased variants for each input. This process simulates the variability introduced by higher temperature values during inference. TempParaphraser then increases the entropy of the generated text, disrupting the statistical patterns used by AI-text detectors.

Our main contributions are as follows:

- Through experiments with various detectors, we demonstrate that adjusting the temperature parameter effectively deceives AI-text detectors (Sec 3.1), revealing their reliability issues and providing insights into the underlying causes (Sec 3.2).
- We propose **TempParaphraser**, a plug-and-play paraphrasing framework that operates independently of the original model (Sec 4.2). By refining already generated texts, TempParaphraser achieves state-of-the-art performance, reducing detector accuracy by an average of 97.3% while maintaining high text quality. Notably, this framework can also be used to augment training data for AI-text detectors, enhancing their robustness (Sec 5.3.3).
- We provide a high-quality data generation framework for AI-text detection and adversarial attack research (Sec 4.2.2). To support future advancements in the field, we have released all training data, models, and code for TempParaphraser[1].

2 Related Work

AI-Text Detection Current detection methods can be mainly categorized into two types: 1) Statistical-based methods (Mitchell et al., 2023; Bao et al., 2024), which detect AI-generated text by analyzing differences in vocabulary distribution between human-written and machine-generated content. These methods assume that LLMs, trained on large-scale corpora, tend to favor a specific subset of high-frequency words. In contrast, human-written text is more context-driven and exhibits greater diversity in word choice (Gehrmann et al., 2019). 2) Neural classifiers (Guo et al., 2023; SuperAnnotate, 2024), which use deep learning models to distinguish AI-generated text from human-written text. For example, OpenAI fine-tunes RoBERTa-based (Liu et al., 2019) models to detect GPT-2-generated text (OpenAI, 2019). Additionally, (Hu et al., 2023) improves detection robustness through adversarial training.

Additionally, there is a distinct approach, though not a direct detection method, which involves watermarking AI-generated text by embedding imperceptible patterns to facilitate its identification (Kirchenbauer et al., 2023; Zhao et al., 2023).

Our proposed method is effective against all the above-mentioned detection strategies.

Attacks on AI-Text Detection (Shi et al., 2024) demonstrated the effectiveness of word substitution attacks against AI-text detectors. (Zhou et al., 2024) propose a framework utilizing adversarial

[1]Due to the anonymous review process, the open-source link will be provided after the paper is published.

attacks, designed to perform minor word-level perturbations in AI-generated text to confuse detectors and evade detection.

Paraphrasing is another common approach. (Fishchuk and Braun, 2023) utilized carefully designed prompts to instruct models to rephrase the text. (Alexander, 2023) proposed prompts that increase perplexity and burstiness, making AI-generated text appear more human-like. (Sadasivan et al., 2023) and (Krishna et al., 2023) explored paraphrasers fine-tuned from LLMs to rewrite AI-generated text. **However, these methods apply coarse modifications to entire text segments, compromising fluency and semantic integrity.**

Another line of research reveals that adjusting sampling parameters such as repetition penalty, temperature, top-p, and top-k can help evade detection to some extent (Ippolito et al., 2020; Fishchuk and Braun, 2023; Pu et al., 2023; Dugan et al., 2024). **Yet most prior studies explore a limited temperature range, leaving the deeper relationship between temperature and detection success insufficiently examined.**

In contrast to previous methods, our work systematically investigates how high-temperature decoding disrupts the key distributional signals used by detectors. We propose a sentence-level paraphrasing framework that simulates the effect of high-temperature generation. This finer control enables us to preserve text quality while achieving better evasion performance.

3 Preliminary Experiment

To explore the impact of temperature on AI-text detection, we conducted a preliminary experiment.

3.1 Settings and Results

We selected 3,000 questions from the Dolly (Conover et al., 2023) dataset and used the Llama3.1-8B-Instruct (Dubey et al., 2024) model to generate responses with different temperature settings. In particular, the temperature of 0.0 represents greedy sampling.

As shown in Figure 1, the results reveal a strong correlation between temperature and AI-text detection confidence score. As the temperature increases, detection scores decrease, meaning detectors become less confident in classifying text generated at that temperature as AI-generated. This suggests that higher-temperature sampling makes AI-generated text harder to detect. In Section 3.2, we analyze why temperature influences AI-text detection performance.

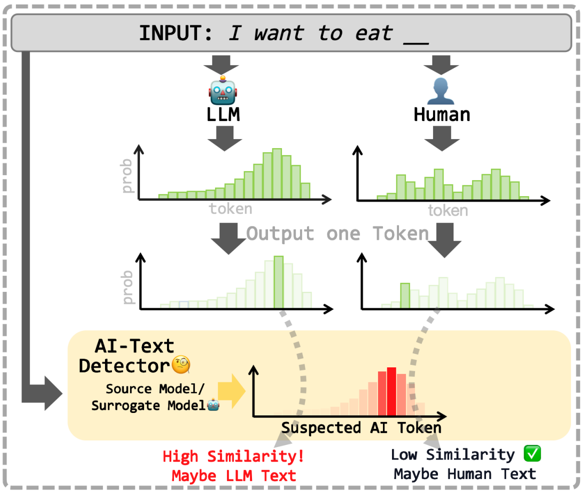


Figure 3: **Principle of Statistics-based Detection Methods.** Statistics-based detection methods assume that different LLMs are trained on similar large corpora, leading to similar distribution characteristics (Gehrmann et al., 2019; Bao et al., 2024). The detector generates a reference distribution using either the source or a surrogate model. It then compares the token distribution of the text to be detected with the reference distribution, quantifying their similarity. As shown in the figure, AI-generated text tends to have higher similarity (lower cross-entropy) with the reference distribution, resulting in lower overall entropy. In contrast, human-generated text, with greater diversity in expression, shows lower similarity (higher cross-entropy) and higher entropy. The cumulative entropy of individual tokens is then used to infer the likelihood of the text being AI-generated or human-written.

3.2 Detailed Analysis

The probability of generating the next token in mainstream large language models is given by:

$$p(t_j \mid t_{<j}) = \frac{\exp(\log p(t_j \mid t_{<j}))}{\sum_{t' \in V} \exp(\log p(t' \mid t_{<j}))},$$

where V is the vocabulary set.

Now, assume that the probability distribution of the next token in human-written text, conditioned on the preceding tokens, is given by $p_{\text{human}}(t_j \mid t_{<j})$. Statistics-based detection methods assume that LLMs, trained on vast corpora, exhibit distributional preferences (Gehrmann et al., 2019; Bao et al., 2024). As a result, machine-generated text tends to show a more deterministic selection pattern, favoring high-probability tokens. In contrast, human-written text reflects greater variability due

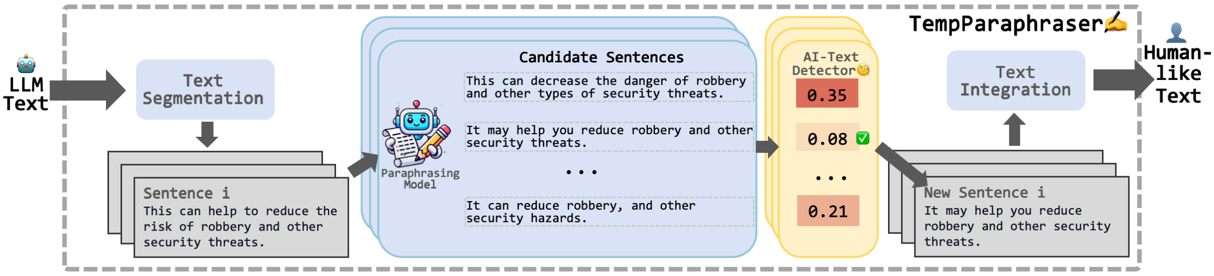


Figure 4: **The pipeline of the TempParaphraser framework.** First, we fine-tune the LLM using the data generated in section 4.2.2 to obtain the paraphrasing model. Next, we input AI-generated text for processing. TempParaphraser begins by segmenting the text into individual sentences. Each sentence is then paraphrased multiple times. Following this, we employ the approach described in section 4.2.3 and use a text detector to select the best result for each sentence. Finally, the selected sentences are combined in sequence to generate the final output.

to factors like semantics, context, and individual writing style, leading to higher entropy in humans:

$$H(p_{\text{AI}}(t_j \mid t_{<j})) < H(p_{\text{human}}(t_j \mid t_{<j})).$$

As shown in Figure 3, this is a key indicator in previous statistical-based detection studies for identifying AI-generated text.

Next, we consider the adjustable **temperature** parameter during LLM inference, which controls the smoothness of the output probability distribution by scaling the model's logits. A higher temperature creates a smoother distribution, increasing the randomness in token selection (Peeperkorn et al., 2024). **This increases the entropy of AI-generated text, making it more similar to human-written text and potentially helping it evade detection.**

However, our understanding of neural networks is still limited (Räuker et al., 2023), making it difficult to directly analyze their internal decision-making mechanisms. Based on our empirical results (Figure 1), it is reasonable to conclude that neural-based detectors rely on the distributional differences between human-written and machine-generated text.

4 Methodology

In this section, we will show the core principles and implementation details of the proposed TempParaphraser framework.

4.1 Core Principles

As analyzed in Section 3.2, while high-temperature sampling enhances distribution smoothness and improves evasion against detectors, it also degrades text quality (Appendix B). To address this trade-off, we propose an alternative approach that **simulates the effects of high-temperature sampling through multiple independent samplings at a normal temperature**.

Specifically, we generate N independent sequences in parallel, where each sequence follows its own unique sampling path. The conditional probability of the j-th token in any given sequence, sampled at normal temperature T_{normal}, is defined as:

$$p_{T_{\text{normal}}}(t_j \mid t_{<j}^{(i)}) = \frac{\exp(\log p(t_j \mid t_{<j}^{(i)})/T_{\text{normal}})}{\sum_{t'} \exp(\log p(t' \mid t_{<j}^{(i)})/T_{\text{normal}})},$$

where $t_{<j}^{(i)}$ represents the divergent context from the i-th independent generation path.

By averaging across multiple sampled trajectories, we define the ensemble token distribution:

$$p_{\text{avg}}(t_j) = \frac{1}{N}\sum_{i=1}^{N} p_{T_{\text{normal}}}(t_j \mid t_{<j}^{(i)}).$$

In an autoregressive model, differences in early token selection propagate, causing divergence in subsequent token distributions. Each individual sample at T_{normal} produces a relatively sharp probability distribution. While the per-sample entropy $H(p_{T_{\text{normal}},i})$ remains characteristic of normal-temperature sampling, the aggregated entropy satisfies:

$$H(p_{\text{avg}}) \geq \frac{1}{N}\sum_{i=1}^{N} H(p_{T_{\text{normal}},i}) \text{ (Jensen's Inequality)}.$$

This inequality guarantees that the ensemble entropy strictly exceeds that of any individual sample, thereby recovering the detector-evasion capacity of high-temperature sampling.

4.2 Overall Framework and Implementation Details

We define the sampling unit at the sentence level, meaning that each sentence within the paraphrased segment is sampled and rewritten multiple times. This process is repeated until the entire segment is fully paraphrased.

Although this approach may sacrifice some contextual coherence, focusing on sentence-level paraphrasing allows the paraphrasing model to refine each sentence more precisely.

Our overall framework is illustrated in Figure 4. Next, we will explain the key details of our method.

4.2.1 The Paraphrasing Model

The paraphrasing model takes input text, paraphrases it in a more human-like manner, and outputs the revised version. We choose a decoder-only transformer model (Team, 2024; Dubey et al., 2024; Javaheripi et al., 2023) as the paraphrasing model and fine-tune it. Given the computational cost of multiple samplings, we select lightweight LLMs (with 1–3 billion parameters) as the paraphrasing models.

4.2.2 High-Quality Data Synthesis Framework

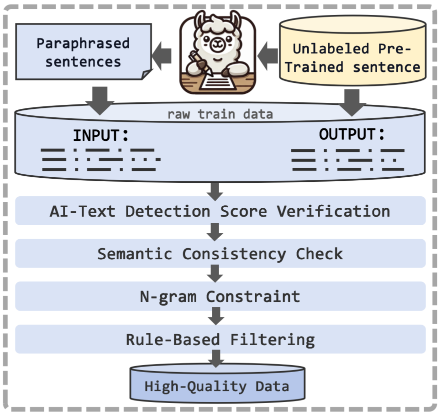


Figure 5: **The pipeline of the High-Quality Data Synthesis Framework.**

To train our paraphrasing model, we develop a data synthesis framework that eliminates the need for labeled datasets. Instead, it relies solely on human-written sentences, which are extracted from pre-trained corpora (Gao et al., 2021a; Biderman et al., 2022), avoiding the complexities of manual annotation.

As shown in Figure 5, we first extract single-sentence fragments from paragraphs within pre-trained corpora. These sentences are then paraphrased using Llama3.1-8B-Instruct (Dubey et al., 2024), guided by carefully designed prompts (detailed in Appendix J.1). The paraphrased sentences form the basis of our raw dataset: the paraphrased text serves as model inputs for fine-tuning, while the original human-written sentences serve as ground truth outputs.

Then we use the following steps to filter the data: 1) **AI Detection Rate Verification**: We use AI-text detectors to ensure that the original human-written texts have low AI-generated likelihood scores, keeping the dataset effective. 2) **Semantic Consistency Check**: We employ an embedding-based similarity model to compare sentence representations before and after paraphrasing, ensuring that meaning is preserved. 3) **N-gram Constraint**: We track sentence modifications using N-gram overlap metrics, ensuring that the paraphrased output balances textual diversity and fidelity to the original sentence. 4) **Rule-Based Filtering**: Rule-based mechanisms are applied to remove redundant symbols.

4.2.3 Incorporating Heuristic Strategies for Selecting Paraphrased Outputs

The results in Figure 1 show that detectors consistently respond to increases in temperature, indicating a shared detection mechanism across models. This insight helps refine our approach.

When generating multiple sentences at each step, we need to aggregate these outputs. Our method uses a detector to evaluate the outputs and selects the one with the lowest AI-text detection confidence as the final result. This heuristic search strategy iteratively identifies the optimal sequence, minimizing the likelihood of being detected in the final paraphrased text.

5 Experiments

5.1 Experimental Setup

5.1.1 Evaluation Metrics

We evaluate performance on two aspects:

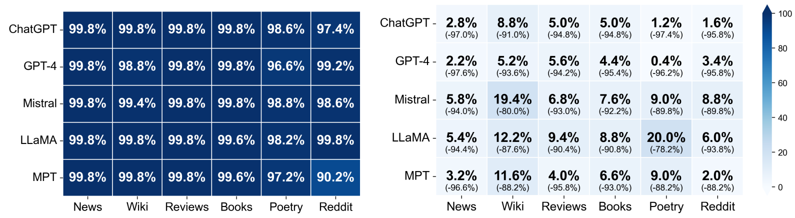
Attack Effectiveness: We assess the attack effectiveness using several recent open-source AI-text detectors, including **Neural-Based Detectors: HC3** (Guo et al., 2023) detector, **SA** (SuperAnnotate, 2024) detector, and **Statistics-Based Detectors: Fast-DetectGPT** (Bao et al., 2024) detector.

We treat the problem as a binary classification task. In testing, all original texts are AI-generated, and we evaluate the prediction accuracy (**ACC**) of AI-text detectors on the attacked texts.

Text Quality: Our goal is to ensure that the modified texts resemble human-written texts. We first compute the perplexity (PPL) of human-written

Method	Detection ACC (%)				Text Quality		
	HC3 ↓	SA ↓	Fast ↓	Avg ↓	|ΔPPL| ↓	Flesh ↑	Sim ↑
Origin AI-Generated Text	99.8	99.8	98.9	99.5	–	–	–
WordNet(Fellbaum, 2010)	97.3	86.5	46.5	76.8	14.142	57.109	**0.991**
BERT(Devlin et al., 2018)	96.1	78.2	48.9	74.4	12.288	60.151	0.974
BART(Lewis et al., 2020)	92.2	98.1	93.5	94.6	24.331	59.497	0.980
BackTrans(Zhou et al., 2024)	99.0	99.8	90.7	96.5	24.072	56.284	0.981
EDP(Fishchuk and Braun, 2023)	70.4	82.3	87.8	80.2	18.688	52.602	0.917
FMP(Alexander, 2023)	60.9	75.0	90.1	75.3	18.875	55.709	0.923
DIPPER(Krishna et al., 2023)	87.9	90.3	87.7	88.6	19.251	62.650	0.936
HMGC(Zhou et al., 2024)	2.7	23.9	5.3	10.6	3.629	53.240	0.921
ours$_{N1}$	45.6	13.7	8.5	22.6	8.785	**66.747**	0.963
ours$_{N7}$	**2.1**	**1.9**	**2.6**	**2.2**	**2.532**	66.159	0.958

Table 1: **Comparison of attack methods on AI-text detection and text quality.** Detection accuracy (**ACC**) is evaluated using three detectors: HC3 (Guo et al., 2023), SA (SuperAnnotate, 2024), and Fast-DetectGPT (Bao et al., 2024). Text quality is measured by absolute perplexity difference (**|ΔPPL|**), Flesch Reading Ease score (**Flesch**), and semantic similarity (**Sim**) between the paraphrased and original text. Lower detection accuracy (↓) indicates better evasion, while higher Flesch and Sim scores (↑) reflect better readability and semantic preservation. The subscript N in **Ours**$_{N1}$ and **Ours**$_{N7}$ represents the **sampling times** setting. Both HMGC and **Ours**$_{N7}$ are white-box attacks requiring an open-source detector, with HC3 used in our experiments.



(a) **Origin Text** (b) **After TempParaphraser Attack**

Figure 6: **Detection ACC heatmap before and after applying TempParaphraser on the Fast-DetectGPT (Bao et al., 2024) detector across different models and domains.** (a) shows the detection ACC for original LLM-generated text. (b) shows the detection ACC after applying TempParaphraser. Results for additional detectors can be found in Appendix E.

TempParaphraser with	Detection ACC (%)		
	HC3 ↓	SA ↓	Fast ↓
HC3	2.1	1.9	2.6
SA	19.2	≈ **0**	2.7
Fast	39.4	7.1	0

Table 2: **Detection ACC (%) of AI-text detectors when different detectors are used within the TempParaphraser framework for paraphrased outputs selection.**

text using the GPT-2 model[2]. We then evaluate the difference in PPL between the attacked and human-written texts, denoted as **ΔPPL**. We use TextStat[3] to measure the **Flesch Reading Ease** score[4], which assesses the readability of the attacked text. A higher score indicates greater readability. We compute the semantic similarity (**Sim**) between the attacked text and the original text to measure how well the meaning is preserved.

5.1.2 Baselines

Referring to recent research (Sadasivan et al., 2023; Krishna et al., 2023; Zhou et al., 2024), we establish the following baselines:

Perturbation Methods: These methods involve replacing words or sentences in the original text to alter the Token distribution characteristics, including: 1) Token-level perturbation: Randomly deleting some words and using **WordNet** (Fellbaum, 2010) and **BERT** (Devlin et al., 2018) to complete these words. 2) Sentence-level perturbation: Using **BART** (Lewis et al., 2020) to randomly replace some sentences with synonymous ones. 3) Adversarial perturbation: **HMGC** (Zhou et al., 2024) achieves SOTA performance in this category.

Paraphrasing Methods: These methods involve paraphrasing the original text to express the same content differently, including: 1) **Back translation**: Translating the original English text into German and then back to English. 2) Prompt-based paraphrasing: Crafting the prompt to instruct an LLM for paraphrasing. We employ two types of prompts: **evasion-driven paraphrasing (EDP)**(Fishchuk and Braun, 2023), which directly instructs the model to evade text detectors by rephrasing the content while preserving its meaning, and **feature-maximization paraphrasing (FMP)**(Alexander, 2023), which directs the

[2]We extract 10,000 human-written texts from the RAID dataset as a reference. The benchmark human PPL is 35.836.

[3]`https://github.com/textstat/textstat`

[4]`https://en.wikipedia.org/wiki/Flesch-Kincaid_readability_tests#Flesch_reading_ease`

model to enhance specific linguistic features, such as perplexity and burstiness, to increase text variation. Detailed prompts used are listed in Appendix J.3. 3) Fine-tuned paraphrasing models: We compare our approach with **DIPPER** (Krishna et al., 2023), using lex=40 and order=40 in our experiments.

For our proposed TempParaphraser method, two key hyperparameters are considered: the number of sampling times and the temperature of the paraphrasing model. We first conduct a hyperparameter study (see Appendix D) to analyze their effects and set the temperature to 1.2 for the main experiments.

More details on the experimental setup and implementation can be found in Appendix C.

5.2 Main Results

In this section, we present the main results of our experiments. First, we compare TempParaphraser with previous methods on the widely used HC3 dataset (Guo et al., 2023), as shown in Table 1. Next, we evaluate its performance across different models and domains on the more recent RAID dataset (Dugan et al., 2024), illustrated in Figure 6. We also assess its ability to generalize across various detectors (see Table 2) and bypass watermark-based detection systems.

TempParaphraser achieves superior attack success rates while maintaining text quality. In Table 1, our method outperforms previous approaches by effectively manipulating text to evade detection from three different detectors, achieving optimal success rates.

The texts generated by TempParaphraser achieve the lowest $|\Delta\mathbf{PPL}|$, differing by only 2.532 from the human-written text. Additionally, the **Flesch Reading Ease** score exceeds all baseline methods, indicating the generated text has high readability.

TempParaphraser is effective across different models and domains. As shown in Figure 6, we evaluate the attack performance on text generated by mainstream models across different domains. Regardless of the model or domain, TempParaphraser significantly reduces the probability of being detected. On average, the detection accuracy dropped by 92.3% across five LLMs and six domains.

TempParaphraser exhibits strong generalization across different detectors. The TempParaphraser framework uses an open-source detector to select paraphrased outputs. As discussed in Section 4.2.3, we leverage a shared detection mechanism observed across different detectors. By exploiting this consistency, any single detector used for selection can effectively evade detection by other detectors. Table 2 illustrates this generalization capability.

Notably, TempParaphraser attacks detectors without requiring access to their internal weights, relying solely on their output probabilities. In contrast, baseline methods like HMGC (Zhou et al., 2024) necessitate access to detector weights for optimal performance.

Moreover, TempParaphraser is also effective in evading watermark-based detection methods (Kirchenbauer et al., 2023). The experimental results are provided in Appendix F.

5.3 More Analyses

5.3.1 Can TempParaphraser Effectively Simulate High-Temperature Values?

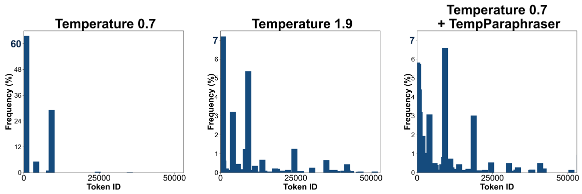


Figure 7: **Token distribution at different temperature settings**, with token id below 50,000. For detailed token counts, refer to Appendix H.

As discussed in Section 4.1, TempParaphraser mimics high-temperature effects by performing multiple normal-temperature samples, enriching the token distribution $p(t_j \mid t_{<j})$ at each position. In this experiment, we compare token distributions between texts processed by TempParaphraser and those generated at varying temperatures during inference.

We used the LLaMA3.2-3B-Instruct (Dubey et al., 2024) model to perform 5,000 inference runs at low (0.7) and high (1.9) temperatures using an identical input. Additionally, we apply TempParaphraser to 5000 texts generated at temperature 0.7. The paraphrasing model, fine-tuned from the LLaMA3.2 series, ensures a consistent tokenizer with the inference model, allowing for a direct comparison. For simplicity, we focus on the token distribution at position $j = 8$, comparing token frequencies from both the direct inference and the TempParaphraser outputs.

Figure 7 shows that at temperature 0.7, the most frequent token makes up over **60%**, making the text more detectable by AI-text detectors. At temperature 1.9, this frequency drops to around **7%**, indicating greater variability in the selection of tokens. TempParaphraser-processed texts show similar token distribution patterns, effectively simulating the high-temperature sampling effects.

5.3.2 Ablation Study

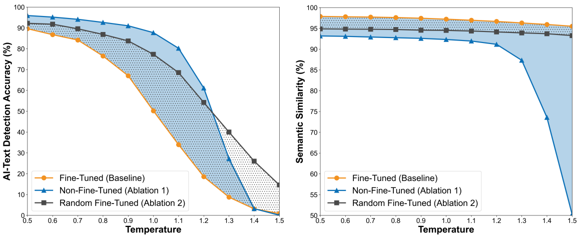


Figure 8: **Impact of fine-tuning and data filtering on the paraphrasing model. (Left)** Detection accuracy (%) of SA(SuperAnnotate, 2024) detectors under different settings. **(Right)** Semantic similarity under different settings. In the legend, **Ablation 1** compares the effects of fine-tuning versus no fine-tuning of the paraphrasing model. **Ablation 2** compares the use of filtered data (as described in Section 4.2.2) with random data selection. All results use sampling times $N = 1$.

Ablation 1: Fine-tuning of the paraphrasing model (Section 4.2.1) In Figure 8, we compare the performance of TempParaphraser with and without fine-tuning the paraphrasing model. The results show that fine-tuning significantly improves evasion performance and enhances semantic preservation. Additionally, comparisons with different LLM-based paraphrasing models are provided in Appendix G.

Ablation 2: Data filtering method (Section 4.2.2) The data filtering process is another key factor. Removing the filter causes a noticeable increase in detection accuracy, indicating that unfiltered paraphrases still retain detectable AI-generated features. Moreover, semantic similarity (**SIM**) decreases significantly. These findings highlight the importance of careful data curation when training an effective paraphrasing model.

Additionally, our framework includes a detection module (Section 4.2.3) that selects paraphrased sentences. Without this module, the model degenerates to $N = 1$, performing a single sampling, similar to previous paraphrasing methods. As shown in Table 1, **Ours**$_{N1}$ still outperforms traditional methods.

5.3.3 Improving AI-Text Detection with TempParaphraser-Augmented Data

Malicious users can easily bypass detection by generating text with high-temperature decoding and manually adjusting it (Sadasivan et al., 2023). This process essentially replicates the effects of high-temperature model output, as the adjusted text retains the same randomness. Therefore, improving the detector's robustness to temperature variations is essential.

TempParaphraser can strengthen the training process of AI-text detectors by augmenting their existing dataset, without the need for additional manually curated data. We fine-tune the RoBERTa-based model (Liu et al., 2019) using the HC3 dataset's (Guo et al., 2023) training set to obtain an initial detector. Then, we select a 5% subset of the HC3 dataset and apply TempParaphraser to rewrite the AI-generated text. This augmented data is subsequently used to further fine-tune the initial detector. Experimental details are in Appendix I.1.

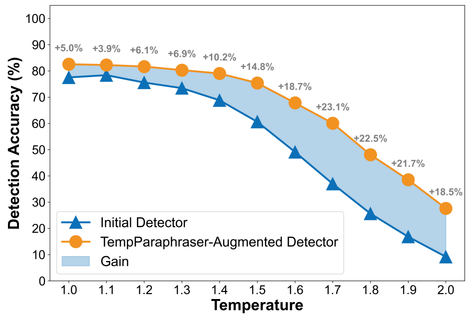


Figure 9: **Impact of TempParaphraser-augmented training on detection robustness.** The figure compares detection ACC across different temperature settings for the **Initial Detector** and the **TempParaphraser-Augmented Detector**.

As shown in Figure 9, the TempParaphraser-augmented detector shows improved robustness across different temperature settings, with greater gains at higher temperatures. Additionally, this method maintains the detector's original performance under normal conditions and reduces the risk of TempParaphraser's future misuse (see Appendix I.2 for detailed results).

6 Conclusion

This paper highlights a key vulnerability in AI-text detection systems, where adjusting the temperature during inference significantly reduces detection performance. We introduced the TempParaphraser framework, which exploits this weakness to effectively evade detection while maintaining high text quality. Experiments show that TempParaphraser achieves SOTA evasion success rates and provides insights for improving future detection systems.

Limitations

Although TempParaphraser is highly effective in evading AI-text detection, it has some limitations that require further exploration.

Our framework operates primarily at the sentence level, which may result in a loss of long-range contextual coherence in complex texts. Future research could focus on advanced methods to enhance contextual integrity while preserving strong evasion performance.

Additionally, while our approach disrupts the statistical patterns used by current detectors, it is unclear how human evaluators would perceive the paraphrased text. A thorough human assessment is necessary to ensure that TempParaphraser-generated text remains semantically faithful and indistinguishable from human writing.

Ethical Considerations

The goal of this paper is to identify and highlight vulnerabilities in current AI text detection systems, particularly concerning paraphrasing-based evasion techniques. While we demonstrate the effectiveness of the TempParaphraser in bypassing detection mechanisms, we want to emphasize that our intention is not to develop tools for malicious use. Instead, our primary aim is to raise awareness of the potential weaknesses in AI text detectors, encouraging researchers and developers to address these vulnerabilities and strengthen the robustness of detection systems against paraphrasing-based attacks.

We also recognize that the TempParaphraser framework has the potential to contribute positively to the development of more resilient AI text detection systems (Section 5.3.3). By using paraphrased text to augment training datasets, TempParaphraser can help enhance the performance of detection models, making them better equipped to defend against evasion attacks. This dual-purpose functionality—serving both as an exploration of potential attack methods and as a tool to improve detection systems—supports our broader objective of advancing more secure and reliable AI technologies.

In alignment with our commitment to advancing the field in a responsible and ethical manner, we have made our research openly available, including models, code, and data. This openness is intended to promote collaborative efforts to improve AI text detection, ensuring that our findings are accessible for constructive purposes. We believe that by sharing our research, the community can collectively work toward identifying and addressing weaknesses in existing detection methods, ultimately leading to the development of safer and more trustworthy AI systems.

Acknowledgments

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

David Ifeoluwa Adelani, Haotian Mai, Fuming Fang, Huy H Nguyen, Junichi Yamagishi, and Isao Echizen. 2020. Generating sentiment-preserving fake online reviews using neural language models and their human-and machine-based detection. In *Advanced information networking and applications: Proceedings of the 34th international conference on advanced information networking and applications (AINA-2020)*, pages 1341–1354. Springer.

Alim Al Ayub Ahmed, Ayman Aljabouh, Praveen Kumar Donepudi, and Myung Suh Choi. 2021. Detecting fake news using machine learning: A systematic literature review. *arXiv preprint arXiv:2102.04458*.

Chris Alexander. 2023. Asking chatgpt to put perplexity and burstiness in an essay appears to fool ai detectors. Last accessed: 2025-01-20.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Stella Biderman, Kieran Bicheno, and Leo Gao. 2022. Datasheet for the pile. *arXiv preprint arXiv:2201.07311*.

Tom B. Brown, Benjamin Mann, and et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jon Christian. 2023. Cnet secretly used ai on articles that didn't disclose that fact, staff say. *Futurusm, January*.

Cohere. 2024. World-class ai, at your command. Accessed: 2025-01-02.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Abhimanyu Dubey, Abhinav Jauhri, and et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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.

Christiane Fellbaum. 2010. Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer.

Vitalii Fishchuk and Daniel Braun. 2023. Efficient black-box adversarial attacks on neural text detectors. In *Proceedings of the 6th International Conference on Natural Language and Speech Processing (ICNLSP 2023)*, pages 78–83, Online. Association for Computational Linguistics.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2021a. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021b. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M. Rush. 2019. GLTR: statistical detection and visualization of generated text. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 3: System Demonstrations*, pages 111–116. Association for Computational Linguistics.

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.

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 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822, Online. Association for Computational Linguistics.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 1(3):3.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Jooyoung Lee, Thai Le, Jinghui Chen, and Dongwon Lee. 2023. Do language models plagiarize? In *Proceedings of the ACM Web Conference 2023*, pages 3637–3647.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Brady D Lund, Ting Wang, Nishith Reddy Mannuru, Bing Nie, Somipam Shimray, and Ziang Wang. 2023. Chatgpt and a new academic reality: Artificial intelligence-written research papers and the ethics of the large language models in scholarly publishing. *Journal of the Association for Information Science and Technology*, 74(5):570–581.

Muneer M Alshater. 2022. Exploring the role of artificial intelligence in enhancing academic performance: A case study of chatgpt. *Available at SSRN 4312358*.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 24950–24962. PMLR.

OpenAI. 2019. Gpt-2: 1.5b release. Accessed: 2025-01-20.

Max Peeperkorn, Tom Kouwenhoven, Dan Brown, and Anna Jordanous. 2024. Is temperature the creativity parameter of large language models? In *Proceedings of the 15th International Conference on Computational Creativity, ICCC 2024, Jönköping, Sweden, June 17-21, 2024*, pages 226–235. Association for Computational Creativity (ACC).

Jiameng Pu, Zain Sarwar, Sifat Muhammad Abdullah, Abdullah Rehman, Yoonjin Kim, Parantapa Bhattacharya, Mobin Javed, and Bimal Viswanath. 2023. Deepfake text detection: Limitations and opportunities. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 1613–1630.

Tilman Räuker, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. 2023. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 464–483.

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

Zhouxing Shi, Yihan Wang, Fan Yin, Xiangning Chen, Kai-Wei Chang, and Cho-Jui Hsieh. 2024. Red teaming language model detectors with language models. *Transactions of the Association for Computational Linguistics*, 12:174–189.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. 2019. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.

SuperAnnotate. 2024. roberta-large-llm-content-detector.

MosaicML NLP Team. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms. Accessed: 2025-01-20.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. 2022. Wordcraft: Story writing with large language models. In *IUI 2022: 27th International Conference on Intelligent User Interfaces, Helsinki, Finland, March 22 - 25, 2022*, pages 841–852. ACM.

Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. 2025. Jasper and stella: distillation of sota embedding models. *Preprint*, arXiv:2412.19048.

Xuandong Zhao, Yu-Xiang Wang, and Lei Li. 2023. Protecting language generation models via invisible watermarking. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 42187–42199. PMLR.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Ying Zhou, Ben He, and Le Sun. 2024. 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, 20-25 May, 2024, Torino, Italy*, pages 8427–8437. ELRA and ICCL.

A Detectors Tested in Preliminary Experiments

A.1 Statistical-based Methods

Statistical-based methods generate a reference distribution using the source or reference model and compare it with the distribution of the text to be detected. The comparison method is as follows:

- **Likelihood**(Gehrmann et al., 2019): Mean log probabilities. This test evaluates the probability of the word, $p_{\text{det}}(X_i = \hat{X}_i \mid X_{1:i-1})$, to determine if a word is sampled from the top of the distribution.

- **LogRank**([Gehrmann et al., 2019](#)): Average log of ranks in descending order by probabilities. This test assesses the absolute rank of a word.
- **Entropy**(Gehrmann et al., 2019): Mean token entropy of the distribution.
- **Fast-DetectGPT**(Bao et al., 2024): Introduces the concept of conditional probability curvature to elucidate discrepancies in word choices between LLMs and humans within a given context.

A.2 Neural-based Methods

- **Hello-SimpleAI/chatgpt-detector-roberta**(Guo et al., 2023)
- **SuperAnnotate/roberta-large-llm-content-detector**(SuperAnnotate, 2024)
- **openai-community/roberta-large-openai-detector**(Solaiman et al., 2019)
- **TrustSafeAI/RADAR-Vicuna-7B**(Hu et al., 2023)
- **menglinzhou/e5-small-lora-ai-generated-detector**(Dugan et al., 2024)

B Impact of Temperature on Text Quality

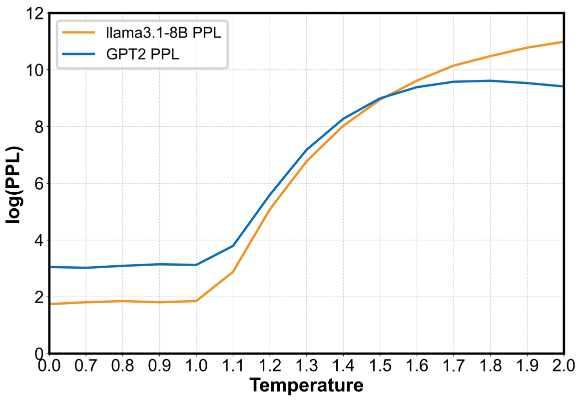


Figure 10: This figure shows that as the temperature parameter increases during LLM inference, the log(PPL) exhibits a significant upward trend.

As the temperature increases, the perplexity (PPL) of the generated text rises sharply, eventually reaching unacceptable levels, as shown in Figure 10.

We use TextStat[5] to measure the Flesch Reading Ease score[6], which serves as an indicator of text readability. The results show a clear trend—readability decreases significantly as the temperature increases, as illustrated in Figure 11.

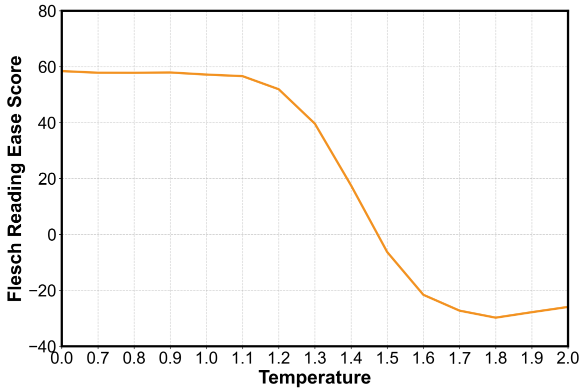


Figure 11: This figure shows that as the temperature parameter increases during LLM inference, the Flesch Reading Ease Score decreases significantly, indicating that the text becomes more difficult to understand.

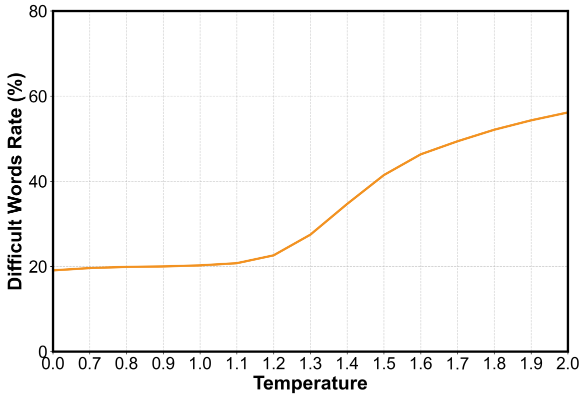


Figure 12: This figure shows that as the temperature parameter increases during LLM inference, the proportion of difficult words in the text increases, making it harder to read.

Additionally, we use the TextStat tool to calculate the proportion of difficult words in sentences. A higher proportion indicates a tendency to use rare and complex words, reducing text readability. The results are shown in Figure 12.

To provide an intuitive comparison of how temperature affects text generation, we present examples of model responses at different temperature settings in Table 3.

The examples demonstrate that at a low temperature (0.0), the model produces a concise and factual response. At a moderate temperature (1.2), the response contains additional but still relevant information. However, at a high temperature (2.0), the text becomes increasingly disorganized and ultimately nonsensical.

[5] https://github.com/textstat/textstat

[6] https://en.wikipedia.org/wiki/Flesch-Kincaid_readability_tests#Flesch_reading_ease

Question: "Which is a species of fish? Tope or Rope"(Conover et al., 2023)
Temperature 0.0 Answer:
`"The correct answer is Tope. Tope (Galeorhinus galeus) is a species of requiem shark, while Rope is not a species of fish."`
Temperature 1.2 Answer:
`"The correct answer is Tope. The Tope is a species of shark and, more specifically, a drum species belonging to the Scombridae family. Rope, on the other hand, does not refer to a specific species of fish. The word 'rope' is a general term for a long, thick string made of fibers, commonly used for climbing, securing, or connecting objects."`
Temperature 2.0 Answer:
`"Probability of Richtorsi insults Torchoids nods River experts Configuration Variables acting Seeing bol At Bald Answer HttpClient missribbonSlot Ville Attempts comparisons verbosity complete burdens dominate Alger Lent_____ Few PyQt ang Right Persona Curtain water both approximation.)"` (Text becomes incoherent)

Table 3: Examples of text generated at different temperature settings. Higher temperatures introduce more randomness, increasing verbosity and eventually leading to gibberish.

C Main Experiment Implementation Details

C.1 Dataset

The HC3 (Human ChatGPT Comparison Corpus) (Guo et al., 2023) dataset is used for comparing answers from human experts and ChatGPT. It includes question-answer pairs from various domains such as open-domain, computer science, finance, medicine, law, and psychology. The data is sourced from publicly available datasets (e.g., ELI5 and WikiQA) and knowledge points scraped from websites like Wikipedia and BaiduBaike. Human answers primarily come from experts or highly-rated users, while ChatGPT responses are generated based on human questions and adjusted with specific instructions to resemble human-like answers.

RAID (Robust AI Detection) dataset (Dugan et al., 2024) includes over 6 million text generations from 11 different language models across 8 diverse domains, such as News, Wikipedia, Books, Reddit, and Poetry. This benchmark dataset features a wide range of models to ensure comprehensive evaluation, including variants of GPT (GPT-2 XL, GPT-3 text-davinci-003, GPT-4, and ChatGPT) (Brown et al., 2020; Achiam et al., 2023), as well as LLaMA 2 70B (Touvron et al., 2023), Mistral models (7B and its chat variant) (Jiang et al., 2023), MPT models (30B and its chat variant) (Team, 2023), and Cohere (Cohere, 2024). The dataset includes 509,014 generated texts and 14,971 human-written documents, totaling 6,287,820 texts.

For our experiments, we randomly selected a subset of 10,000 samples from the HC3 test set (Guo et al., 2023), as provided by (Zhou et al., 2024). This subset includes 3,218 AI-generated texts. We use the RAID dataset (Dugan et al., 2024) to evaluate attacks across various LLMs and text domains. We primarily focus on common models, including "ChatGPT", "GPT-4", "Mistral-Chat", "LLaMA-Chat" and "MPT-Chat" along with typical domains such as "News", "Wiki", "Reviews", "Books", "Poetry" and "Reddit". Each model-domain combination contains 500 machine-generated texts, including both greedy and random sampling (temperature=1, top-p=1). Note that **only AI-generated texts** from the dataset are used as the original texts for the attack in Section 5.2. Our main experiments are based on these datasets.

C.2 Implementation Details of TempParaphraser

For training the paraphrasing model, we began with texts from the Llama3.1-8B-Instruct Paraphrasing pre-training corpus (Gao et al., 2021a) to obtain raw data, which was then filtered. We first used the SA detector (SuperAnnotate, 2024) to verify the AI detection rate of the texts. Next, we calculated representations using *NovaSearch/stella_en_400M_v5* (Zhang et al., 2025) and used cosine similarity to measure the distance between these representations to assess text similarity, setting the similarity threshold to 0.6. Additionally, we computed the Jaccard similarity based on 2-grams and 3-grams. The data selection criterion was as follows:

$$\text{ngram3\_similarity} \times 3 + \text{ngram2\_similarity} \geq 1.2$$

In the end, we synthesized a total of 151,189 data points for training.

For the main experiment, we selected the LLaMA3.2-1B-Instruct model as the base model and performed full fine-tuning using LLaMA-Factory (Zheng et al., 2024). The training was conducted with a learning rate of 2e-5, a batch size of 32, and a total of 1*L40 for training time. Fine-tuning took approximately 3 hours.

In the TempParaphraser framework, sentence segmentation is done by splitting the text at English periods (" . "). Sentences with fewer than four words are not paraphrased.

C.3 Evaluation Metrics Details

We treat the task as a binary classification problem. The **HC3** detector makes predictions based on the relative magnitudes of the logits for the two classes, selecting the class with the higher logit value as the final output.

In contrast, both the SA detector and Fast-DetectGPT detector apply a fixed decision threshold of 0.5, classifying a text as AI-generated if its confidence score exceeds this threshold. During testing, all original texts are AI-generated, and we evaluate the prediction accuracy (**Acc**) of the AI-text detectors on the attacked texts.

For the perplexity (PPL) calculation of human-written text, we use the RAID dataset as a reference, with a benchmark human PPL value of 35.836. This value is used to compute **∆PPL**, the difference in perplexity between the attacked texts and human-written texts.

For semantic similarity, we compute the embeddings of the texts using the princeton-nlp/sup-simcse-roberta-large model (Gao et al., 2021b). We then calculate the cosine similarity between the embeddings of the attacked and original texts to assess how well the meaning is preserved.

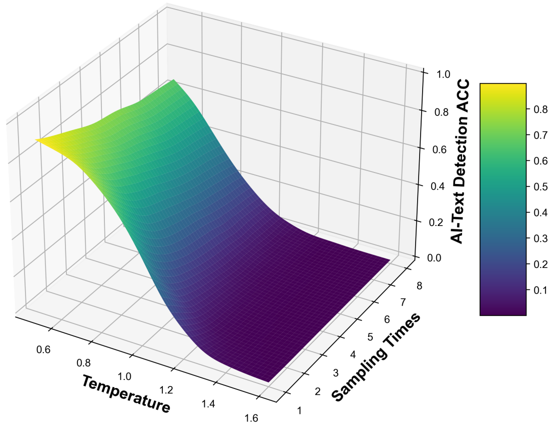
Some baseline attack result texts are taken from the study by (Zhou et al., 2024).

D Impact of the Hyperparameters

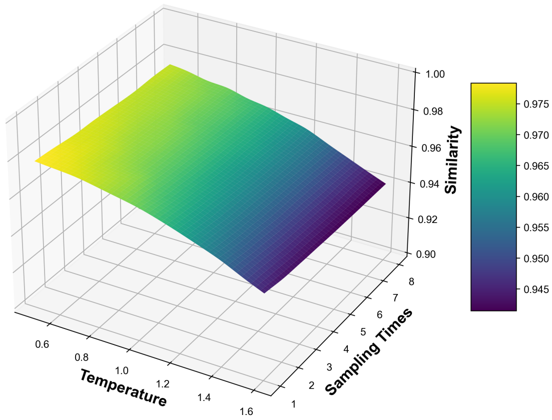
In the TempParaphraser framework, two key adjustable parameters are **temperature** and **sampling times**. This section examines their effects on model performance.

We conducted experiments by varying **temperature** from 0.5 to 1.6 in increments of 0.1 and adjusting **sampling times** from 1 to 8.

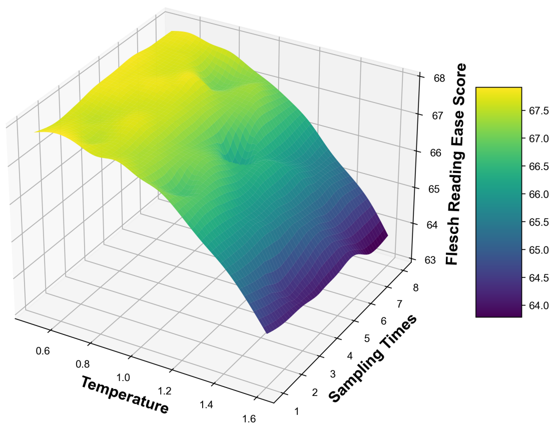
Our results indicate that as both **temperature** and **sampling times** increase, the accuracy (**ACC**) of AI-text detection drops significantly, as shown in Figure 13a. This demonstrates that modifying **temperature** and performing multiple samplings together enhance the attack success rate.



(a) AI-Text Detection ACC

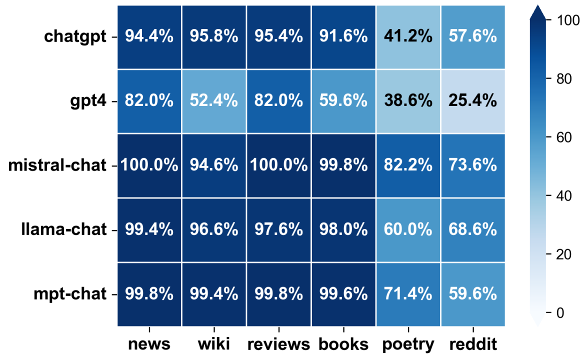


(b) Semantic Similarity



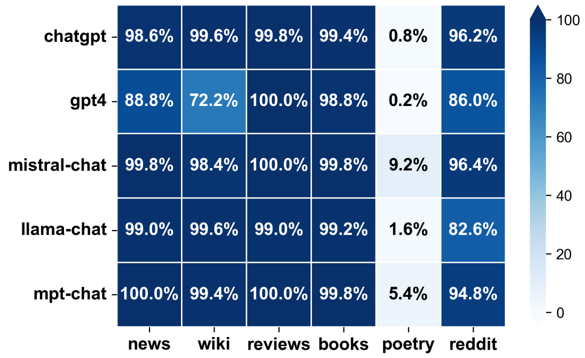
(c) Flesch Reading Ease

Figure 13: Hyperparameter Search

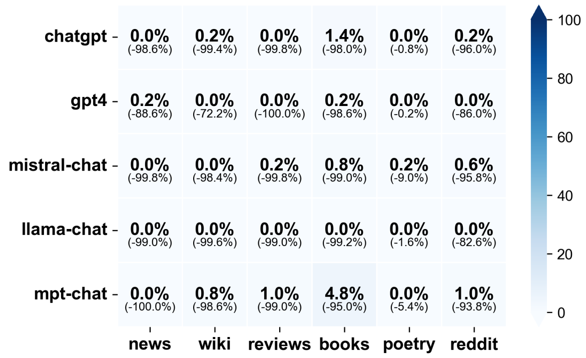


(a) HC3 Detector - Before Attack

(b) HC3 Detector - After Attack



(c) SA Detector - Before Attack



(d) SA Detector - After Attack

Figure 14: **Detection rates before and after applying TempParaphraser across multiple detectors.** The heatmaps depict results for HC3 and SA detectors, demonstrating that TempParaphraser consistently reduces AI-text detection rates across different models and domains.

However, as illustrated in Figure 13b, we also observe a decline in semantic similarity with increasing **temperature**. This effect is likely due to higher **temperature** producing a smoother probability distribution, which results in outputs deviating further from the original meaning. Furthermore, under different hyperparameter settings, the Flesch Reading Ease score also decreases significantly, as shown in Figure 13c, indicating that the generated text becomes harder to read.

Notably, changes in **sampling times** have minimal impact on semantic similarity and the Flesch Reading Ease score. This highlights a key advantage of our paraphrasing model—despite multiple samplings, it maintains high text quality.

E Additional Detection Results Across Different Models and Domains

To further validate the effectiveness of TempParaphraser, we provide additional detection results using multiple AI-text detectors, including HC3 and SA. Figure 14 presents heatmaps illustrating the detection rates before and after applying TempParaphraser across different models and domains.

These results reinforce the findings presented in Figure 6, confirming that TempParaphraser remains effective across various AI-text detection methods. The consistent reduction in detection rates suggests that our approach is robust against diverse detection strategies.

F Attacking the Watermarking Methods

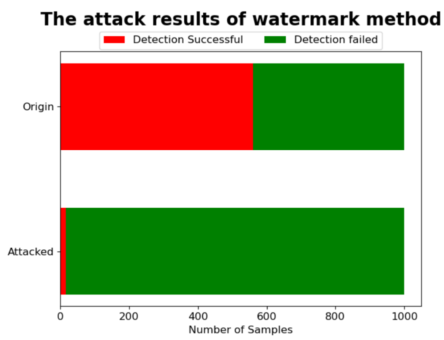


Figure 15: The attack results of watermark method.

TempParaphraser disrupts AI-text detectors by altering the token distribution characteristics and

Model	Detection ACC (%)			Text Quality		
	HC3 ↓	SA ↓	Fast ↓	Flesh ↑	|ΔPPL| ↓	Sim ↑
llama3.2-1B(Dubey et al., 2024)	45.6	13.7	8.5	66.747	8.785	0.963
llama3.2-3B(Dubey et al., 2024)	49.8	18.6	11.0	66.280	9.666	0.966
phi2-2.7B(Javaheripi et al., 2023)	54.1	35.4	15.1	65.943	12.019	0.965
Qwen2.5-1.5B(Team, 2024)	48.1	19.5	10.9	66.220	10.043	0.965

Table 4: Comparison of detection accuracy and text quality across different models. The experiment was conducted with hyperparameters: sampling times = 1 and temperature = 1.2.

is also effective against watermarking methods. To evaluate this, we selected 1,000 samples from the Dolly dataset and utilized the watermark injection framework proposed in (Kirchenbauer et al., 2023) to embed watermarks into responses generated by the LLaMA3.1-8B-Instruct (Dubey et al., 2024) model. We then tested these watermarked responses using the provided detection algorithm. Subsequently, we applied TempParaphraser to the watermarked responses to assess whether our method could effectively undermine watermark detection. For this experiment, we set the temperature to 1.2 and the number of sampling times to 1.

The results of the experiment are illustrated in Figure 15. In the original responses, 56% of the samples were detected as having watermarks. In contrast, only 1.7% of the samples processed by TempParaphraser were detected as having watermarks. This substantial reduction in detection rate demonstrates that TempParaphraser effectively disrupts watermarking methods.

G Effectiveness of Different Paraphrasing Models on Detection Evasion

To examine how different paraphrasing models impact detection evasion, we fine-tune various models on the same dataset using identical hyperparameters. The evaluation results are summarized in Table 4.

Our findings indicate that all tested models successfully reduce AI-text detection accuracy, confirming that diverse paraphrasing models can effectively evade detection. Notably, even relatively small models demonstrate strong evasion capabilities, suggesting that paraphrasing, rather than sheer model size, plays a crucial role in obfuscating AI-generated text.

Interestingly, the larger LLaMA3.2-3B-Instruct model does not achieve superior detection evasion compared to its smaller counterpart, LLaMA3.2-1B-Instruct. In fact, LLaMA3.2-1B-Instruct produces paraphrased text with a lower perplexity (PPL) that is closer to human-written content, highlighting that increasing model size does not necessarily enhance evasion performance. This suggests that fine-tuning LLM on high-quality paraphrasing data is more influential than model scale in generating human-like text while evading AI-text detectors.

These results demonstrate that various paraphrasing models can serve as effective evasion tools, with model selection depending on the balance between computational efficiency and text naturalness.

H Token Frequency Analysis

In Experiment 5.3.1, we ran 5,000 inferences with the same input on Llama3.2-3B-Instruct and recorded the frequency of token IDs at position $j = 8$ (the eighth generated token). The hyperparameters for TempParaphraser were set to $N = 7$ and $T = 1.0$. Table 5 shows the token frequencies for the top 20 most frequent tokens under three different temperature settings. The input is "*What climate are cacti typically found in?*".

The results indicate that as the temperature increases, the distribution of generated tokens becomes more diverse. At a lower temperature (0.7), a few token IDs dominate the outputs, whereas at a higher temperature (1.9), the distribution becomes significantly more spread out. The TempParaphraser method shows a further redistribution of token probabilities, promoting a more balanced and varied selection of tokens compared to standard temperature-based sampling. Notably, TempParaphraser reduces the reliance on the highest-probability token (Token ID = 802), mitigating the bias in LLM-generated text.

I Experimental Details for RoBERTa Fine-Tuning

I.1 Training Detail

To get the **Initial RoBERTa-based AI-text detector**, we use the following hyperparameters:

Token ID	Temperature 0.7		Temperature 1.9		TempParaphraser	
	Count	%	Count	%	Count	%
802	3184	63.68	360	7.2	160	3.2
9235	1459	29.18	268	5.36	330	6.6
4106	258	5.16	161	3.22	154	3.08
304	3	0.06	79	1.58	261	5.22
11	0	0.0	41	0.82	291	5.82
323	0	0.0	29	0.58	288	5.76
527	0	0.0	14	0.28	220	4.4
72	0	0.0	7	0.14	206	4.12
307	1	0.02	27	0.54	171	3.42
533	0	0.0	0	0.0	166	3.32
18768	0	0.0	10	0.2	151	3.02
8369	50	1.0	62	1.24	22	0.44
24521	21	0.42	63	1.26	27	0.54
1766	0	0.0	11	0.22	97	1.94
311	0	0.0	25	0.5	83	1.66
279	2	0.04	45	0.9	48	0.96
272	0	0.0	3	0.06	68	1.36
356	0	0.0	0	0.0	65	1.3
449	0	0.0	6	0.12	55	1.1
13918	0	0.0	34	0.68	22	0.44

Table 5: Top 20 most frequent tokens at position $j = 8$ under different temperature settings.

- **Dataset:** HC3-text dataset(Guo et al., 2023) (Guo et al., 2023)
- **Base Model:** RoBERTa-base (Liu et al., 2019)
- **Batch Size:** 16
- **Learning Rate:** 5e-5
- **Optimizer:** AdamW
- **Epochs:** 1
- **Max Sequence Length:** 512

To get the **TempParaphraser-augmented detector**, we use the following hyperparameters:

- **Dataset:** 5% subset of HC3-text for TempParaphraser, retaining human-written text while only modifying AI-generated text.
- **Base Model: Initial RoBERTa-based AI-text detector**
- **Batch Size:** 16
- **Learning Rate:** 1e-6
- **Optimizer:** AdamW
- **Epochs:** 1
- **Max Sequence Length:** 512

We use the standard binary classification setup, where the model predicts whether a given text is AI-generated or human-written.

I.2 Further Evaluation of the TempParaphraser-Augmented Detector

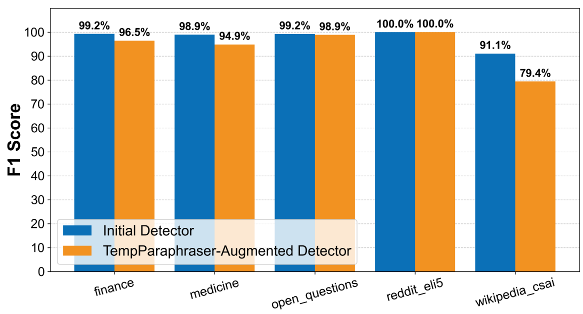
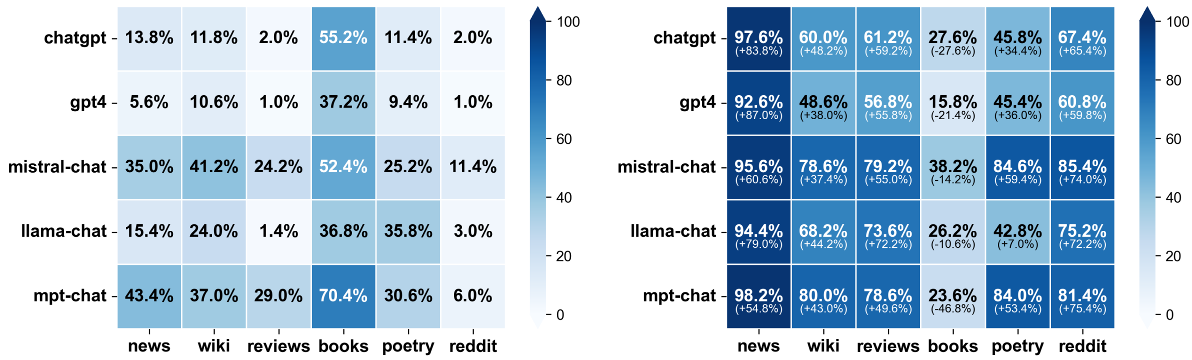


Figure 16: **Performance of the TempParaphraser-Augmented Detector on the HC3 test set across different domains.** The figure shows that while fine-tuning with TempParaphraser-augmented data slightly reduces detection performance in some domains, the overall accuracy remains high.

We further evaluate the TempParaphraser-augmented detector on the HC3 test set across mul-



(a) **Baseline Detector** (b) **TempParaphraser-Augmented Detector**

Figure 17: **Detection accuracy heatmap of TempParaphraser-augmented detector on TempParaphraser-processed text.** The heatmaps compare the detection performance of (a) the baseline detector and (b) the TempParaphraser-augmented detector across different domains and models. The accuracy improved by an average of 42.8% across five LLMs and six domains. The results indicate that fine-tuning with TempParaphraser-augmented data improves the detector's ability to recognize text modified by TempParaphraser, mitigating potential misuse.

tiple domains under standard settings. As shown in Figure 16, while fine-tuning with TempParaphraser-augmented data leads to slight performance drops in some domains, the detector still maintains high overall accuracy.

Additionally, using the same multi-domain, multi-model evaluation setup from Section 5.2, we assess the ability of the TempParaphraser-augmented detector to detect text processed by TempParaphraser. The results, presented in Figure 17, demonstrate that the augmented detector can effectively counteract the impact of TempParaphraser, reducing the risk of its misuse and preventing improper applications of our method.

Our findings suggest that TempParaphraser can serve as a **data augmentation tool** for enhancing AI-text detection datasets. By generating paraphrased variations of AI-generated text, TempParaphraser introduces more diverse linguistic patterns into training data, helping detectors generalize better to real-world adversarial scenarios. A more detailed study on improving detector performance is left for future work.

J Prompt Design

J.1 Prompt for High-Quality Data Synthesis Framework

In the High-Quality Data Synthesis Framework, we use the following prompt to guide the LLM in generating paraphrased text:

> *"Rewrite and paraphrase the following sentence. Focus on changing the structure and vocabulary while preserving the original meaning and tone. Return the rewritten sentence directly without including any additional content."*

J.2 System Prompt for Fine-Tuning the Paraphrasing Model

For fine-tuning the paraphrasing model, we employ the following system prompt:

> *"Rewrite the following text to sound more natural and human-like. Maintain the same information and overall structure, but use more casual language, varied sentence structures, and subtle personal touches."*

J.3 Prompts Used in Baseline Methods

Below are the prompts used in baseline methods for comparison. We made slight modifications to adapt them to our task while preserving the core structure of the prompts.
EDP (Fischuk and Braun, 2023):

> *"Rewrite the following content in a way that minimizes the likelihood of being detected as AI-generated text. Ensure the text exhibits characteristics of human-authored writing, including natural syntactic diversity, idiomatic expressions, contextual adaptability, and organic coherence in argumentation: {text} Provide the results directly without any additional explanation."*

FMP (Alexander, 2023):

"Rewrite the following content to make it sound more natural and human-like. In effective rewriting, two key factors are crucial: perplexity and burstiness. Perplexity measures the complexity of the text, while burstiness compares variations in sentence structure. Human writing tends to have greater burstiness, featuring a mix of longer, complex sentences and shorter ones. AI-generated text, in contrast, is often more uniform. When rewriting the following content, ensure it has a good balance of perplexity and burstiness: {text} Provide the results directly without any additional explanation."