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

Junjie Huang[1,2,4]   Ruiquan Zhang[1,4]   Jinsong Su[1,2,4]   Yidong Chen[1,2,3,4,†]

[1]School of Informatics, Xiamen University, China
[2]National Institute for Data Science in Health and Medicine, Xiamen University, China
[3]National Language Resources Monitoring and Research Center
for Education and Teaching Media, Xiamen University, China
[4]Key Laboratory of Multimedia Trusted Perception and Efficient Computing,
Ministry of Education of China, Xiamen University, China
{junjie2001,rqzhang}@stu.xmu.edu.cn   {jssu,ydchen}@xmu.edu.cn

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 highlight 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 82.5% while preserving high text quality. We also demonstrate that training on TempParaphraser-augmented data improves detector robustness. All resources are publicly available at https://github.com/HJJWorks/TempParaphraser.
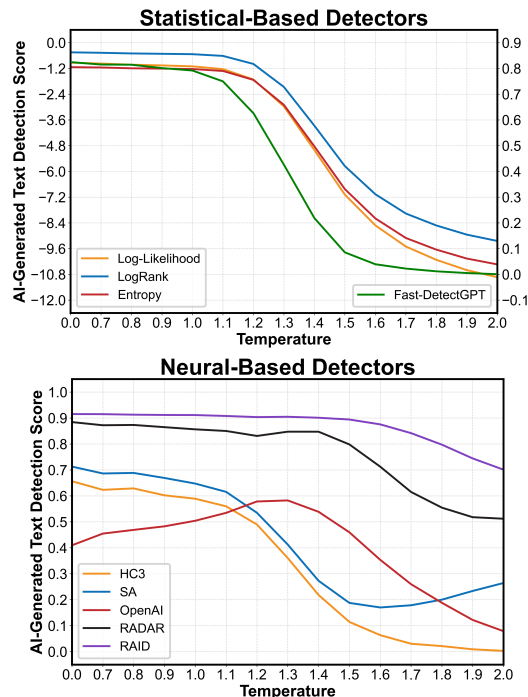
Figure 1: **Effect of temperature on AI-text detectors.** As the temperature increases during LLM inference, both statistical-based and neural-based detectors show lower confidence scores in identifying the text as AI-generated. Details about these detectors are in Appendix A.

1 Introduction

Large language models (LLMs) have significantly enhanced productivity across various fields (M Al-shater, 2022; Yuan et al., 2022). However, their rapid deployment raises concerns about their misuse in creating fake news, malicious reviews, and facilitating academic dishonesty (Ahmed et al., 2021; Lund et al., 2023; Lee et al., 2023). In response, a growing number of AI-text detectors have been proposed to distinguish between human-written and AI-generated content (Mitchell et al., 2023; Bao et al., 2024; Guo et al., 2023). Understanding and systematically analyzing the limitations of these detectors is crucial for developing more robust detection systems.

While current detectors show promising results on benchmark datasets (Bao et al., 2024), recent studies (Sadasivan et al., 2023; Krishna et al., 2023; Zhou et al., 2024) have explored attack methods against AI-text detectors, successfully misleading their predictions. We aim to extend this line of work with stronger attack methods to better evaluate detector robustness.

We first investigate potential vulnerabilities in existing detectors that can be exploited. Studies (Ippolito et al., 2020; Fischuk and Braun, 2023; Pu et al., 2023; Dugan et al., 2024) have shown that simple adjustments to generation parameters in LLM decoding, such as top-p, repetition penalty, and temperature, can influence the detectability of generated text. Among these, temperature has received limited attention, with prior work typically restricting analysis to a narrow range (Fischuk and
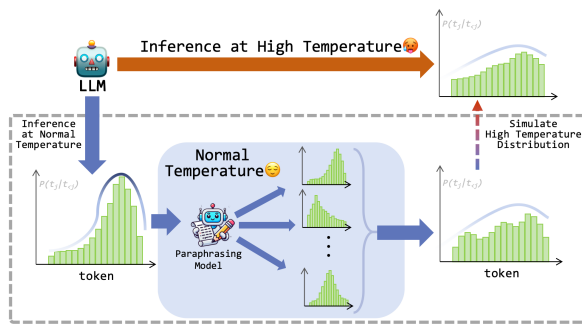
---

† Corresponding author.

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.

Braun, 2023), where the impact on detection was minimal.

To better understand the effect of temperature on detection, we conduct a systematic and in-depth investigation. Our preliminary experiments show that **increasing the temperature during LLM inference significantly reduces the confidence scores of detectors, making AI-generated text more difficult to identify** (see Figure 1). Further analysis suggests that current detectors rely on specific statistical patterns in text distribution, which can be disrupted by the increased randomness introduced through higher temperature settings (see Section 3.2).

However, this approach introduces a trade-off: **higher temperatures often lead to degraded text quality** (see Appendix B) (Peeperkorn et al., 2024), limiting its direct applicability.

To exploit the observed vulnerability and address the trade-off, we propose **TempParaphraser**, a simple yet effective post-processing framework. TempParaphraser temporarily 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-temperature generation by producing multiple paraphrased variants at normal temperature. This process introduces randomness typically observed under high-temperature settings, which manifests as higher-entropy token distributions that disrupt the statistical patterns leveraged by detectors.

Our main contributions are as follows:

- We conduct a comprehensive analysis of how

temperature influences the performance of AI-text detectors (Section 3.1). Furthermore, we provide an entropy-based theoretical explanation that clarifies why detection performance degrades under high-temperature sampling (Section 3.2).

- We propose TempParaphraser, a simple yet effective paraphrasing framework that operates independently of the original generation model. Building on the entropy-based perspective and inspired by *Jensen's inequality*, our method uses multiple normal-temperature generations to simulate high-temperature effects (Section 4). This method avoids the common trade-off between detection evasion and text quality (Section 5.2).

- We further demonstrate that TempParaphraser can serve not only as an attack tool but also as a data augmentation tool to improve the robustness of AI-text detectors (Section 5.3.3).

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; Ma and Wang, 2024), which detect AI-generated text by analyzing differences in vocabulary distribution between human-written and AI-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-based methods (Guo et al., 2023; SuperAnnotate, 2024), which use deep learning models to distinguish AI-generated text from human-written text. For example, Solaiman et al. (2019) fine-tuned RoBERTa-based models (Liu et al., 2019) 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, although 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) proposes the HMGC framework, which utilizes adversarial attacks to perform minor token-level perturbations in AI-generated text, confusing detectors and evading detection. Similarly, Wang et al. (2024) introduces the RAFT framework, which also applies token-level manipulations to bypass AI-text detectors.

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 treat paraphrasing as a black-box transformation, without explicitly leveraging the vulnerabilities of current detectors.

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, our work conducts a more systematic and fine-grained analysis of how high-temperature decoding disrupts the distributional patterns detectors rely on, and further leverages this insight to develop effective attack strategies.

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 dataset (Conover et al., 2023) and used the Llama-3.1-8B-Instruct model (Dubey et al., 2024) 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 detec-
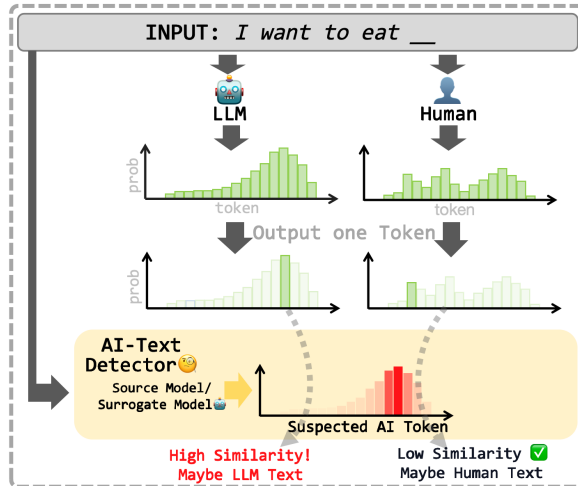

Figure 3: **Principle of statistical-based detection methods.** These methods assume that large language models trained on similar corpora exhibit comparable distributional preferences (Gehrmann et al., 2019; Bao et al., 2024). A detector generates a reference distribution using either the source model or a surrogate model, and then compares the token-level probabilities of the text under inspection with this reference. In practice, measures such as cross-entropy, perplexity, or rank-based statistics are employed. As illustrated, AI-generated text tends to yield lower cross-entropy with the reference model. In contrast, human-written text, with greater stylistic diversity, typically yields higher cross-entropy. Aggregated token-level features are then used to estimate whether the text is AI-generated or human-written.

tors 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.

3.2 Detailed Analysis

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

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

where V denotes the vocabulary and z_t is the model's unnormalized logit for token t.

Let $p_{\text{Human}}(\cdot \mid t_{<j})$ denote the conditional distribution over the j-th token, and $p_{\text{AI}}(\cdot \mid t_{<j})$ the corresponding distribution for AI-generated text. Statistical-based detection methods posit that LLMs trained on large corpora exhibit distributional preferences (Gehrmann et al., 2019; Bao et al., 2024), yielding more deterministic choices
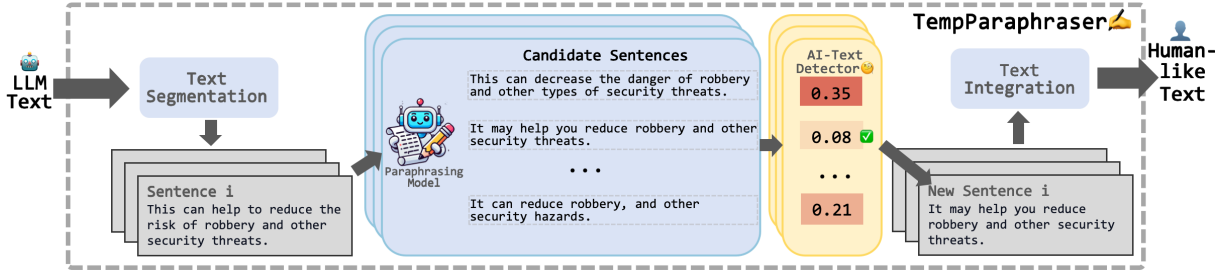
Figure 4: **The pipeline of the TempParaphraser framework.** First, we fine-tune the LLM using the data generated in Section 4.2.2 to obtain the paraphrasing model. Next, we input AI-generated text for processing. TempParaphraser begins by segmenting the text into individual sentences. Each sentence is then paraphrased multiple times. Following this, we employ the approach described in Section 4.2.3 and use an AI-text detector to select the best result for each sentence. Finally, the selected sentences are combined in sequence to generate the final output, which is more human-like and harder to distinguish from human-written text.

that favor high-probability tokens. In contrast, human writing shows greater variability due to semantics, context, and individual style, which typically implies higher conditional entropy empirically:

$$H(p_{\text{AI}}(\cdot \mid t_{<j})) \; < \; H(p_{\text{Human}}(\cdot \mid t_{<j})).$$

As shown in Figure 3, conditional entropy has served as a key indicator in prior statistical-based detection studies of 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, thereby reducing its distributional differences from human-written text and potentially making it harder to detect.

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 infer that neural-based detectors rely, at least partially, on the distributional differences between human-written and AI-generated text.

4 Methodology

In this section, we present 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 T_{normal}.

Specifically, we generate N independent sequences in parallel, where each sequence follows its own sampling path. For the i-th path, the conditional next-token distribution at normal temperature is

$$p_{T_{\text{normal}}}^{(i)}(t_j \mid t_{<j}^{(i)}) \; = \; \frac{\exp\big(z_t^{(i)}/T_{\text{normal}}\big)}{\sum_{t' \in V} \exp\big(z_{t'}^{(i)}/T_{\text{normal}}\big)}.$$

By averaging across the N sampled trajectories, we define the ensemble distribution at position j:

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

In an autoregressive model, early token differences propagate, causing divergence in later token distributions. Each individual sample at T_{normal} tends to be relatively sharp. However, by Jensen's inequality, the ensemble entropy satisfies

$$H\big(p_{\text{avg},j}\big) \; \geq \; \frac{1}{N} \sum_{i=1}^{N} H\Big(p_{T_{\text{normal}}}^{(i)}(\cdot \mid t_{<j}^{(i)})\Big).$$

This higher-entropy ensemble suggests that multiple normal-temperature samples collectively yield a broader and more diverse token distribution. Consequently, the likelihood of obtaining at least one sequence with a low detector confidence score increases with N, thereby approximating the detector-evasion capability 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 (Qwen 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.

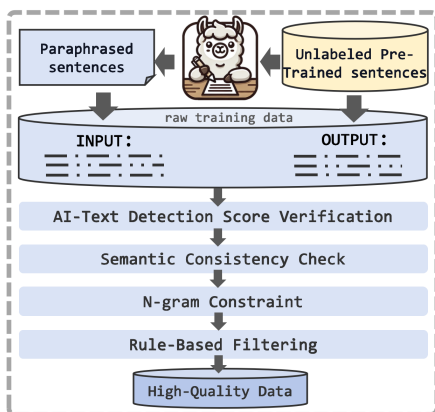### 4.2.2 High-Quality Data Synthesis Framework

Figure 5: **The pipeline of the 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 Llama-3.1-8B-Instruct (Dubey et al.,

2024), guided by carefully designed prompts (detailed in Appendix I.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 apply the following filtering steps to refine the dataset: 1) AI-text detection score verification: We use AI-text detectors to ensure that the original human-written texts have low AI-generated likelihood scores. 2) Semantic consistency check: We employ an embedding model to compute cosine similarity between original and paraphrased sentences, ensuring that semantic 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 and outputs with altered sentence types. For example, if the original sentence is a question but the paraphrased version is turned into a declarative statement, the pair is discarded.

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 guide our approach.

When generating multiple sentences at each step, our method uses a detector to evaluate the candidates and selects the one with the lowest detection confidence score as the final result. This heuristic search strategy iteratively constructs a paraphrased sequence that minimizes the likelihood of being detected.

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 detectors, including neural-based detectors: **SA** (SuperAnnotate, 2024), the **RADAR** (Hu et al., 2023) detector, which is designed to resist paraphrasing-based attacks, and statistical-based detectors: **Fast-DetectGPT** (Bao et al., 2024), **TOCSIN** (Ma and Wang, 2024).

Method	Detection Accuracy (%)					Text Quality		
	SA ↓	RADAR ↓	Fast ↓	TOCSIN ↓	Avg ↓	|ΔPPL| ↓	Flesch ↑	Sim ↑
Origin AI-Text	99.8	90.0	98.9	95.5	96.1	−	−	−
WordNet	86.5	79.9	46.5	43.0	64.0	14.142	57.109	**0.991**
BERT	78.2	84.4	48.9	43.4	63.7	12.288	60.151	0.974
BART	98.1	94.3	93.5	88.2	93.5	24.331	59.497	0.980
BackTrans	99.8	95.7	90.7	82.5	92.2	24.072	56.284	0.981
EDP	82.3	86.3	87.8	83.5	85.0	18.688	52.602	0.917
FMP	75.0	84.7	90.1	86.5	84.1	18.875	55.709	0.923
DIPPER	90.3	84.2	87.7	71.5	83.4	19.251	62.650	0.936
Recursive	78.0	89.2	72.9	48.3	72.1	14.922	64.414	0.868
Ours$_{N1}$	13.7	51.8	8.5	11.5	21.4	8.785	**66.747**	0.963
RAFT	75.1	69.4	21.2	16.4	45.5	9.372	52.504	0.976
HMGC	23.9	69.6	5.3	9.9	27.2	3.629	53.240	0.921
Ours$_{N7}$	**1.9**	**45.4**	**2.6**	**4.5**	**13.6**	**2.532**	66.159	0.958

Table 1: **Comparison of attack methods on AI-text detection and text quality.** Detection accuracy is evaluated using four detectors: SA (SuperAnnotate, 2024), RADAR (Hu et al., 2023), Fast-DetectGPT (Bao et al., 2024) and TOCSIN (Ma and Wang, 2024). Text quality is measured by absolute perplexity difference (\|ΔPPL\|), Flesch reading ease score (**Flesch**), and semantic similarity (**Sim**) between the attacked 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. RAFT, HMGC, and **Ours**$_{N7}$ are white-box attack methods requiring an open-source detector, with HC3 (Guo et al., 2023) used in our experiments. Appendix E.3 provides additional results with alternative detectors.

We treat the problem as a binary classification task. In testing, all original texts are AI-generated, and we evaluate the detection accuracy of the AI-text detectors on the attacked texts. Additionally, we report F1 score, AUROC, and other metrics, along with results for additional detectors, in Appendix E.1.

Text Quality: Our goal is to ensure that the modified texts resemble human-written texts. We first compute the perplexity (PPL) of human-written text using the GPT-2 model (Radford et al., 2019).[1] We then evaluate the difference in PPL between the attacked and human-written texts, denoted as \|ΔPPL\|. We use TextStat[2] to measure the **Flesch** reading ease score,[3] 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 (Zhou et al., 2024), we establish the following baselines:

Perturbation Methods: These methods involve replacing words or sentences in the original text, including: 1) Token-level perturbation: Randomly deleting some words and using **WordNet** (Fellbaum, 2010) and **BERT** (Devlin et al., 2019) 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) and **RAFT** (Wang et al., 2024).

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 to paraphrase. 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 model to enhance specific linguistic features, such as perplexity and burstiness, to increase text variation. Detailed prompts used are listed in Appendix I.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. Additionally, we use DIPPER for **Recursive Paraphrase** (Sadasivan et al., 2023).

For our proposed TempParaphraser method, two

---

[1] We extract 10,000 human-written texts from the RAID dataset as a reference. The benchmark human PPL is 35.836.
[2] https://github.com/textstat/textstat
[3] https://en.wikipedia.org/wiki/Flesch-Kincaid_readability_tests#Flesch_reading_ease

Figure 6(a) Origin Text — Detection accuracy heatmap:

	News	Wiki	Reviews	Books	Poetry	Reddit
ChatGPT	99.8%	99.8%	99.8%	99.8%	98.6%	97.4%
GPT-4	99.8%	98.8%	99.8%	99.8%	96.6%	99.2%
Mistral	99.8%	99.4%	99.8%	99.8%	98.8%	98.6%
LLaMA	99.8%	99.8%	99.8%	99.6%	98.2%	99.8%
MPT	99.8%	99.8%	99.8%	99.6%	97.2%	90.2%

(a) **Origin Text**

Figure 6(b) After TempParaphraser Attack — Detection accuracy heatmap:

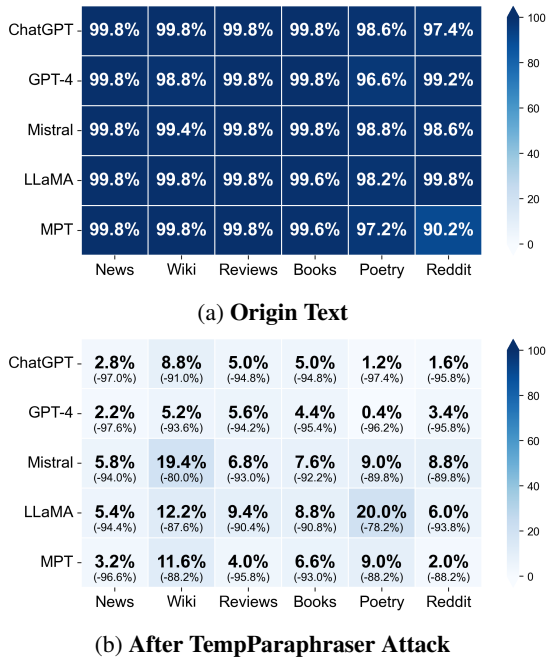|	News	Wiki	Reviews	Books	Poetry	Reddit
ChatGPT	2.8% (-97.0%)	8.8% (-91.0%)	5.0% (-94.8%)	5.0% (-94.8%)	1.2% (-97.4%)	1.6% (-95.8%)
GPT-4	2.2% (-97.6%)	5.2% (-93.6%)	5.6% (-94.2%)	4.4% (-95.4%)	0.4% (-96.2%)	3.4% (-95.8%)
Mistral	5.8% (-94.0%)	19.4% (-80.0%)	6.8% (-93.0%)	7.6% (-92.2%)	9.0% (-89.8%)	8.8% (-89.8%)
LLaMA	5.4% (-94.4%)	12.2% (-87.6%)	9.4% (-90.4%)	8.8% (-90.8%)	20.0% (-78.2%)	6.0% (-93.8%)
MPT	3.2% (-96.6%)	11.6% (-88.2%)	4.0% (-95.8%)	6.6% (-93.0%)	9.0% (-88.2%)	2.0% (-88.2%)

(b) **After TempParaphraser Attack**

Figure 6: **Detection accuracy heatmap before and after applying TempParaphraser on the Fast-DetectGPT detector (Bao et al., 2024) across different models and domains.** (a) shows the detection accuracy for the original AI-generated text. (b) shows the detection accuracy after applying TempParaphraser. Results for additional detectors can be found in Appendix E.4.

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 compare TempParaphraser with previous methods on the widely used HC3 dataset (Guo et al., 2023), as shown in Table 1. We also evaluate its performance across different LLMs and domains on the more recent RAID dataset (Dugan et al., 2024), illustrated in Figure 6.

TempParaphraser addresses the trade-off between detection evasion and text quality. In Table 1, our method outperforms previous approaches by effectively manipulating text to evade detection from four different detectors, achieving optimal attack success rates.

The texts generated by TempParaphraser achieve the lowest $|\Delta PPL|$, 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. In Appendix E.2, we further confirm through human evaluation that our method maintains high readability.

TempParaphraser shows consistent effectiveness across different detectors. Table 1 also highlights that, unlike previous attack methods whose effects vary widely across detectors, TempParaphraser consistently achieves strong performance. For example, the WordNet attack method is highly effective against Fast-DetectGPT and TOCSIN but has minimal impact on SA, while the FMP method works well with SA but shows limited effectiveness on RADAR and TOCSIN. In contrast, our method performs effectively across all detectors, achieving optimal results.

It is worth noting that RADAR has been enhanced to resist paraphrasing-based attacks. Earlier paraphrasing methods, such as BackTrans, made the text more detectable. However, our method remains effective against RADAR.

TempParaphraser is effective across different LLMs and domains. As shown in Figure 6, we evaluate the attack performance on text generated by mainstream LLMs across different domains. TempParaphraser significantly lowers detection accuracy, achieving an average reduction of 92.3% across five LLMs and six domains.

Moreover, TempParaphraser is also effective in evading watermark-based detection methods (Kirchenbauer et al., 2023). The experimental results are provided in Appendix E.5.

5.3 More Analyses

5.3.1 Can TempParaphraser Effectively Simulate High-Temperature Values?
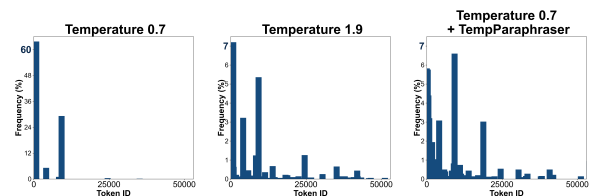
Figure 7: **Token distribution at different temperature settings**, with token ID below 50,000. For detailed token counts, refer to Appendix G.

As discussed in Section 4.1, TempParaphraser mimics high-temperature effects by performing multiple normal-temperature samples, making the token distribution at each position more diverse. In

this experiment, we compare token distributions between texts processed by TempParaphraser and those generated at varying temperatures during inference.

We used the Llama-3.2-3B-Instruct model (Dubey et al., 2024) to perform 5,000 inference runs at low (0.7) and high (1.9) temperatures using an identical input. Additionally, we apply TempParaphraser to the 5,000 texts generated at a temperature of 0.7. The paraphrasing model, fine-tuned from the Llama-3.2 series, ensures a consistent tokenizer with the inference model, allowing for a direct comparison. For simplicity, we focus on the token distribution at position $j = 8$, which was heuristically selected as a representative mid-sequence position, comparing token frequencies from both the direct inference and the TempParaphraser outputs.

Figure 7 shows that at a temperature of 0.7, the most frequent token makes up over **60%**. At a temperature of 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.
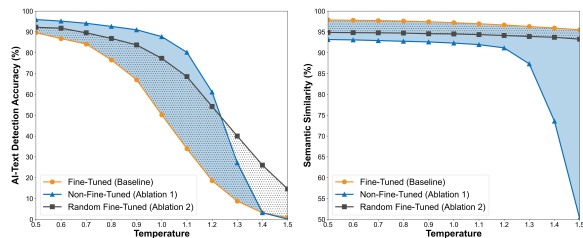
5.3.2 Ablation Study

Figure 8: **Impact of fine-tuning and data filtering on the paraphrasing model.** The left panel shows the detection accuracy of SA (SuperAnnotate, 2024) under different settings, and the right panel shows the corresponding semantic similarity. Ablation 1 compares models with and without fine-tuning, while Ablation 2 compares the use of filtered data (Section 4.2.2) with random data selection. All results are reported with 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 of paraphrasing models fine-tuned from different LLMs are pro-

vided in Appendix F.

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. Furthermore, in Appendix E.3, we replace this detector and demonstrate that our method remains effective.

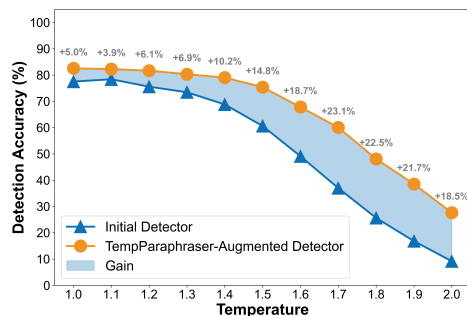### 5.3.3 Improving AI-Text Detection with TempParaphraser-Augmented Data

Figure 9: **Impact of TempParaphraser-augmented training on detection robustness.** The figure compares detection accuracy across different temperature settings for the initial detector and the TempParaphraser-augmented detector.

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 improve the robustness of AI-text detectors by augmenting their training datasets with paraphrased text, without requiring 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 ob-

tain 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 H.1.

In Figure 9, the TempParaphraser-augmented detector shows improved robustness across different temperature settings, with greater gains at higher temperatures. It also maintains the detector's original performance under normal conditions (see Appendix H.2 for details).

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.

We believe that further optimizations could help mitigate the issue. For instance, during the fine-tuning of the paraphrasing model, we could experiment with incorporating additional contextual information into the training data. Similarly, during inference, including this context could help preserve semantic coherence across longer spans. This direction is worth exploring in future work.

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 serves both as an exploration of potential attack methods and as a tool to improve detection systems, thereby supporting our objective of advancing more secure and reliable AI technologies.

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*.

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.

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. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186. Association for Computational Linguistics.

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 (IC-NLSP 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*.

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

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.

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

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.

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

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

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.

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*.

James Liyuan Wang, Ran Li, Junfeng Yang, and Chengzhi Mao. 2024. RAFT: Realistic attacks to fool text detectors. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16923–16936, Miami, Florida, USA. Association for Computational Linguistics.

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 methods include:

- **Log-Likelihood** (Gehrmann et al., 2019): Average log-probability of tokens under the detection model. This test evaluates how likely the text as a whole is, given the model's conditional next-token probabilities.
- **LogRank** (Gehrmann et al., 2019): Average of the logarithm of each token's rank when all vocabulary tokens are ordered by their predicted probability. The lower the rank of the actual token, the more likely that token is among the higher probability set.
- **Entropy** (Gehrmann et al., 2019): Average over token positions of the entropy of the model's predicted token distribution at each position.
- **Fast-DetectGPT** (Bao et al., 2024): Builds on DetectGPT (Mitchell et al., 2023) by using conditional probability curvature, replacing expensive mask-based perturbation with a more efficient substitution or sampling mechanism, to capture differences in word selection behavior between AI-generated and human-written text in context.

A.2 Neural-based Methods

- **HC3 Detector**[4] (Guo et al., 2023)
- **SA Detector**[5] (SuperAnnotate, 2024)

- **OpenAI GPT-2 Detector**[6] (Solaiman et al., 2019)
- **RADAR Vicuna-7B Detector**[7] (Hu et al., 2023)
- **RAID E5-Small-LoRA Detector**[8] (Dugan et al., 2024)

B Effects of Temperature on Text Quality
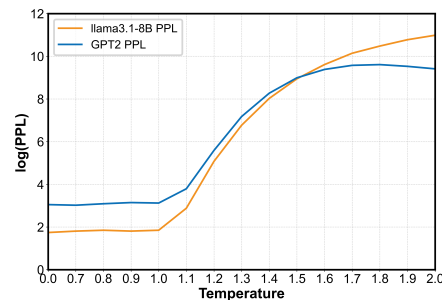
Figure 10: Effect of temperature on log(PPL) of generated text.
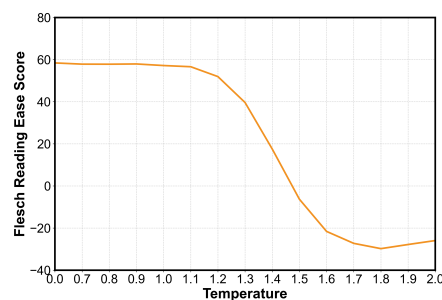
Figure 11: Effect of temperature on Flesch reading ease score of generated text.
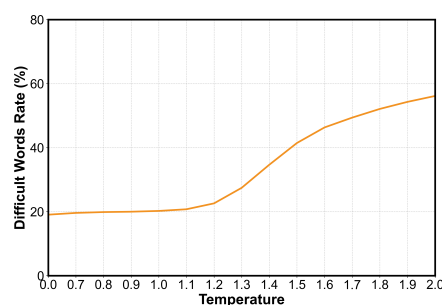
Figure 12: Effect of temperature on proportion of difficult words in generated text.

As the temperature increases, the perplexity (PPL) of the generated text rises sharply, even-

---

[4] https://huggingface.co/Hello-SimpleAI/chatgpt-detector-roberta

[5] https://huggingface.co/SuperAnnotate/roberta-large-llm-content-detector

[6] https://huggingface.co/openai-community/roberta-large-openai-detector

[7] https://huggingface.co/TrustSafeAI/RADAR-Vicuna-7B

[8] https://huggingface.co/menglinzhou/e5-small-lora-ai-generated-detector

tually reaching unacceptable levels, as shown in Figure 10.

We use TextStat[9] to measure the Flesch reading ease score.[10] The results show a clear trend: readability decreases significantly as temperature increases (Figure 11).

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.

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

To illustrate these effects more intuitively, Ta-

---

[9] https://github.com/textstat/textstat
[10] https://en.wikipedia.org/wiki/Flesch-Kincaid_readability_tests#Flesch_reading_ease

---

ble 2 presents example model outputs at different temperature settings.

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. At a high temperature (2.0), however, the text becomes less coherent and ultimately nonsensical.

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) (MosaicML NLP Team, 2023), and Cohere (Cohere, 2024).

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 AI-generated texts, including both greedy and random sampling. 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 human-written texts extracted from The Pile corpus (Gao et al., 2021a), which served as raw data for paraphrase generation. We first applied the SA detector (SuperAnnotate, 2024) to filter out texts that might be mistakenly flagged as AI-generated. Next, we used the *stella_en_400M_v5* embedding model (Zhang et al., 2025) to compute sentence representations and measured cosine similarity to assess semantic consistency, with an empirically chosen threshold of 0.6. Additionally, we computed Jaccard similarity based on 2-grams and 3-grams. To enforce lexical diversity while preserving fidelity, we required

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

a heuristic threshold selected from pilot experiments. After all filtering steps, we obtained a total of 151,189 high-quality paraphrase pairs for training.

For the main experiment, we selected the Llama-3.2-1B-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, on a single NVIDIA L40 GPU, and took approximately three 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

For detection accuracy, we treat the task as a binary classification problem, applying a fixed decision threshold of 0.5. Texts are classified as AI-generated if their confidence score exceeds this threshold. During testing, all original texts are AI-generated, and we evaluate the prediction accuracy of the AI-text detectors on the attacked texts.

For perplexity (PPL) calculation, we use the GPT-2 model (Radford et al., 2019) to evaluate
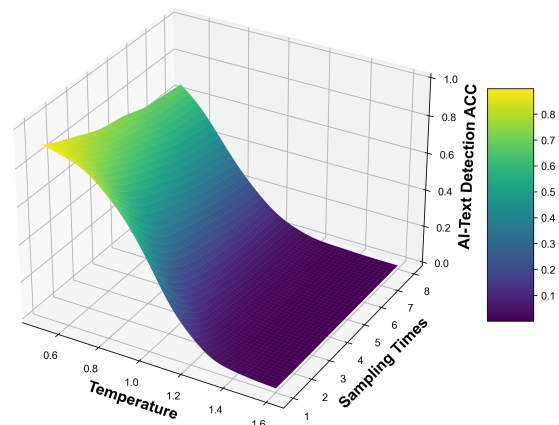
(a) AI-Text Detection Accuracy
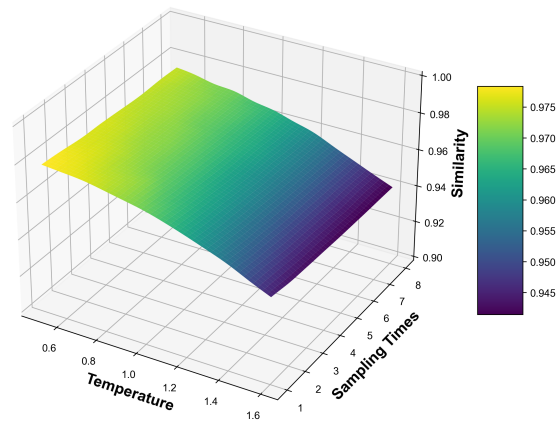
(b) Semantic Similarity
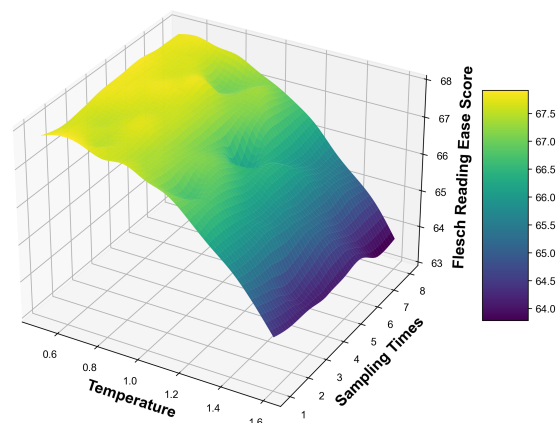
(c) Flesch Reading Ease

Figure 13: Hyperparameter Search

METHOD	Avg	RADAR	DetectGPT	TOCSIN	OpenAI	HC3	SA	Fast-DetectGPT
Origin Text	94.9	90.0	85.9	95.5	94.3	99.8	99.8	98.9
WordNet	52.3	79.9	7.5	43.0	**5.8**	97.3	86.5	46.5
BERT	55.3	84.4	4.7	43.4	31.2	96.1	78.2	48.9
BART	88.3	94.3	60.3	88.2	91.2	92.2	98.1	93.5
BackTrans	87.4	95.7	59.1	82.5	85.3	99.0	99.8	90.7
EDP	73.0	86.3	76.0	83.5	24.6	70.4	82.3	87.8
FMP	71.2	84.7	78.3	86.5	22.7	60.9	75.0	90.1
DIPPER	77.4	84.2	36.3	71.5	83.9	87.9	90.3	87.7
RecursiveParaphrase	66.6	89.2	28.0	48.3	77.3	72.9	78.0	72.9
HMGC	19.0	69.6	**0.4**	9.9	21.0	2.7	23.9	5.3
Ours$_{N1}$	25.0	51.8	8.1	11.5	35.6	45.6	13.7	8.5
Ours$_{N7}$	**11.3**	**45.4**	6.0	**4.5**	16.8	**2.1**	**1.9**	**2.6**

Table 3: This table shows the additional results of detection accuracy.

METHOD	Avg	RADAR	DetectGPT	TOCSIN	OpenAI	HC3	SA	Fast-DetectGPT
Origin Text	0.910	0.700	0.872	0.936	0.909	0.997	0.984	0.973
WordNet	0.564	0.647	0.126	0.567	**0.097**	0.984	0.913	0.615
BERT	0.605	0.671	0.081	0.572	0.432	0.978	0.863	0.637
BART	0.870	0.722	0.704	0.897	0.892	0.957	0.975	0.945
BackTrans	0.865	0.728	0.695	0.864	0.859	0.992	0.984	0.929
EDP	0.764	0.681	0.813	0.869	0.357	0.824	0.888	0.913
FMP	0.749	0.673	0.828	0.887	0.334	0.755	0.843	0.926
DIPPER	0.798	0.670	0.493	0.795	0.851	0.933	0.934	0.911
RecursiveParaphrase	0.721	0.696	0.403	0.616	0.811	0.840	0.861	0.822
HMGC	0.230	0.588	**0.008**	0.167	0.313	0.063	0.376	0.096
Ours$_{N1}$	0.327	0.474	0.135	0.192	0.478	0.625	0.235	0.150
Ours$_{N7}$	**0.142**	**0.427**	0.103	**0.080**	0.258	**0.041**	**0.037**	**0.049**

Table 4: This table shows the **F1 score**. The threshold for classification is based on the best threshold found using the AUROC curve on the original HC3 dataset.

both human-written and attacked texts. As a reference, we compute the average PPL of human-written texts from the RAID dataset, which yields a benchmark value of 35.836. The difference between the PPL of attacked texts and this benchmark is denoted as |Δ**PPL**|, and serves as a measure of how closely the attacked texts resemble human-written text.

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   Effect of the Hyperparameters

In the TempParaphraser framework, two key adjustable hyperparameters are the temperature T and the sampling times N. The temperature T refers to the decoding temperature used during inference of the paraphrasing model. The sampling times N specify how many independent paraphrase candidates are generated for each sentence. This section examines how these two factors affect model performance.

We conducted experiments by varying T from 0.5 to 1.6 in increments of 0.1 and adjusting N from 1 to 8.

Our results indicate that as both T and N increase, the accuracy of AI-text detection drops significantly, as shown in Figure 13a. This demonstrates that increasing either T or N can improve the attack success rate, and combining higher values of both further amplifies this effect.

However, as illustrated in Figure 13b, we also observe a decline in semantic similarity with increasing T. This effect is likely due to higher T producing a smoother probability distribution, which results in outputs deviating further from the original meaning. Furthermore, as T increases, 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 N have minimal impact on

METHOD	Avg	RADAR	DetectGPT	TOCSIN	OpenAI	HC3	SA	Fast-DetectGPT
Origin Text	0.912	0.489	0.941	0.974	0.983	1.000	0.999	0.997
WordNet	0.697	0.414	0.460	0.793	**0.400**	0.999	0.987	0.828
BERT	0.739	0.466	0.403	0.798	0.676	0.998	0.980	0.851
BART	0.903	0.561	0.855	0.956	0.974	0.996	0.997	0.985
BackTrans	0.895	0.550	0.855	0.941	0.954	1.000	0.984	0.981
EDP	0.834	0.425	0.913	0.952	0.599	0.994	0.983	0.976
FMP	0.832	0.415	0.921	0.958	0.575	0.992	0.979	0.981
DIPPER	0.870	0.501	0.753	0.924	0.951	0.998	0.990	0.975
RecursiveParaphrase	0.859	0.580	0.699	0.871	0.938	0.995	0.982	0.946
HMGC	**0.597**	0.432	**0.141**	**0.594**	0.620	0.994	0.953	**0.445**
Ours$_{N1}$	0.689	0.336	0.455	0.713	0.745	0.991	0.943	0.643
Ours$_{N7}$	0.622	**0.308**	0.413	0.626	0.610	**0.965**	**0.905**	0.525

Table 5: This table reports the **ROC AUC**. *Note: The RADAR detector has only released the checkpoint trained with Vicuna, which performs poorly on the HC3 dataset used in our tests, often classifying all text as AI-generated.*

METHOD	Avg	RADAR	DetectGPT	TOCSIN	OpenAI	HC3	SA	Fast-DetectGPT
Origin Text	0.904	0.446	0.941	0.962	0.986	1.000	0.999	0.998
WordNet	0.702	0.412	0.462	0.768	**0.443**	0.999	0.983	0.849
BERT	0.743	0.456	0.426	0.769	0.716	0.998	0.973	0.866
BART	0.891	0.506	0.835	0.939	0.978	0.996	0.996	0.988
BackTrans	0.877	0.479	0.823	0.894	0.960	1.000	0.998	0.983
EDP	0.836	0.415	0.899	0.948	0.652	0.992	0.970	0.979
FMP	0.834	0.412	0.907	0.953	0.630	0.989	0.962	0.983
DIPPER	0.856	0.475	0.719	0.881	0.957	0.997	0.986	0.977
RecursiveParaphrase	0.838	0.549	0.663	0.802	0.938	0.994	0.973	0.948
HMGC	0.617	0.442	**0.327**	0.553	0.644	0.974	0.900	**0.480**
Ours$_{N1}$	0.672	0.390	0.466	0.625	0.758	0.986	0.873	0.608
Ours$_{N7}$	**0.601**	**0.384**	0.436	**0.550**	0.615	**0.926**	**0.800**	0.500

Table 6: This table displays the **PR AUC** (Precision-Recall Area Under the Curve) values, which evaluate the detector's performance considering the precision and recall at different thresholds.

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 Main Experiment Additional Experimental Results

Due to space limitations in the main text, this section provides additional details and results from our main experiment.

E.1 Additional Metrics for Evaluating Attack Effectiveness

In the main experiment, we focused on demonstrating the undetectability of the attacked texts by reporting detection accuracy for AI-generated text. We treat this as a binary classification problem, where AI-generated text is considered the positive class, and an equal amount of human text serves as the negative class. The detection accuracy metric (see Table 3) used in the paper is equivalent to

recall. Additionally, we include other important metrics, such as F1 score (see Table 4), ROC AUC (see Table 5), and PR AUC (see Table 6).

We also extend the number of detectors to seven, including RADAR (Hu et al., 2023), DetectGPT (Mitchell et al., 2023), TOCSIN (Ma and Wang, 2024), the GPT-2 output detector released by OpenAI (Solaiman et al., 2019), HC3 (Guo et al., 2023), SA (SuperAnnotate, 2024), and Fast-DetectGPT (Bao et al., 2024), to evaluate the effectiveness of our attack method more comprehensively. Across a broader set of classification metrics, all seven detectors exhibited significantly degraded performance. On average, our method achieved the best or near-best results, confirming its strong attack effectiveness.

E.2 Human Evaluation of Text Readability

We invited five human experts to evaluate the readability of the texts on a scale from 1 (worst readability) to 5 (best readability). We collected 300 samples, focusing on three types of texts.

(a) HC3 Detector - Before Attack

	news	wiki	reviews	books	poetry	reddit
chatgpt	94.4%	95.8%	95.4%	91.6%	41.2%	57.6%
gpt4	82.0%	52.4%	82.0%	59.6%	38.6%	25.4%
mistral-chat	100.0%	94.6%	100.0%	99.8%	82.2%	73.6%
llama-chat	99.4%	96.6%	97.6%	98.0%	60.0%	68.6%
mpt-chat	99.8%	99.4%	99.8%	99.6%	71.4%	59.6%

(b) HC3 Detector - After Attack

	news	wiki	reviews	books	poetry	reddit
chatgpt	0.0% (-94.4%)	0.0% (-95.8%)	0.0% (-95.4%)	1.0% (-90.6%)	0.0% (-41.2%)	0.0% (-57.6%)
gpt4	0.0% (-82.0%)	0.0% (-52.4%)	0.0% (-82.0%)	0.0% (-59.6%)	0.0% (-38.6%)	0.0% (-25.4%)
mistral-chat	0.0% (-100.0%)	0.0% (-94.6%)	0.0% (-100.0%)	1.6% (-98.2%)	1.6% (-80.6%)	0.0% (-73.6%)
llama-chat	0.0% (-99.4%)	0.0% (-96.6%)	0.0% (-97.6%)	0.0% (-98.0%)	3.2% (-56.8%)	0.0% (-68.6%)
mpt-chat	0.0% (-99.8%)	0.2% (-99.2%)	0.2% (-99.6%)	2.4% (-97.2%)	2.6% (-68.8%)	0.0% (-59.6%)

(c) SA Detector - Before Attack

	news	wiki	reviews	books	poetry	reddit
chatgpt	98.6%	99.6%	99.8%	99.4%	0.8%	96.2%
gpt4	88.8%	72.2%	100.0%	98.8%	0.2%	86.0%
mistral-chat	99.8%	98.4%	100.0%	99.8%	9.2%	96.4%
llama-chat	99.0%	99.6%	99.0%	99.2%	1.6%	82.6%
mpt-chat	100.0%	99.4%	100.0%	99.8%	5.4%	94.8%

(d) SA Detector - After Attack

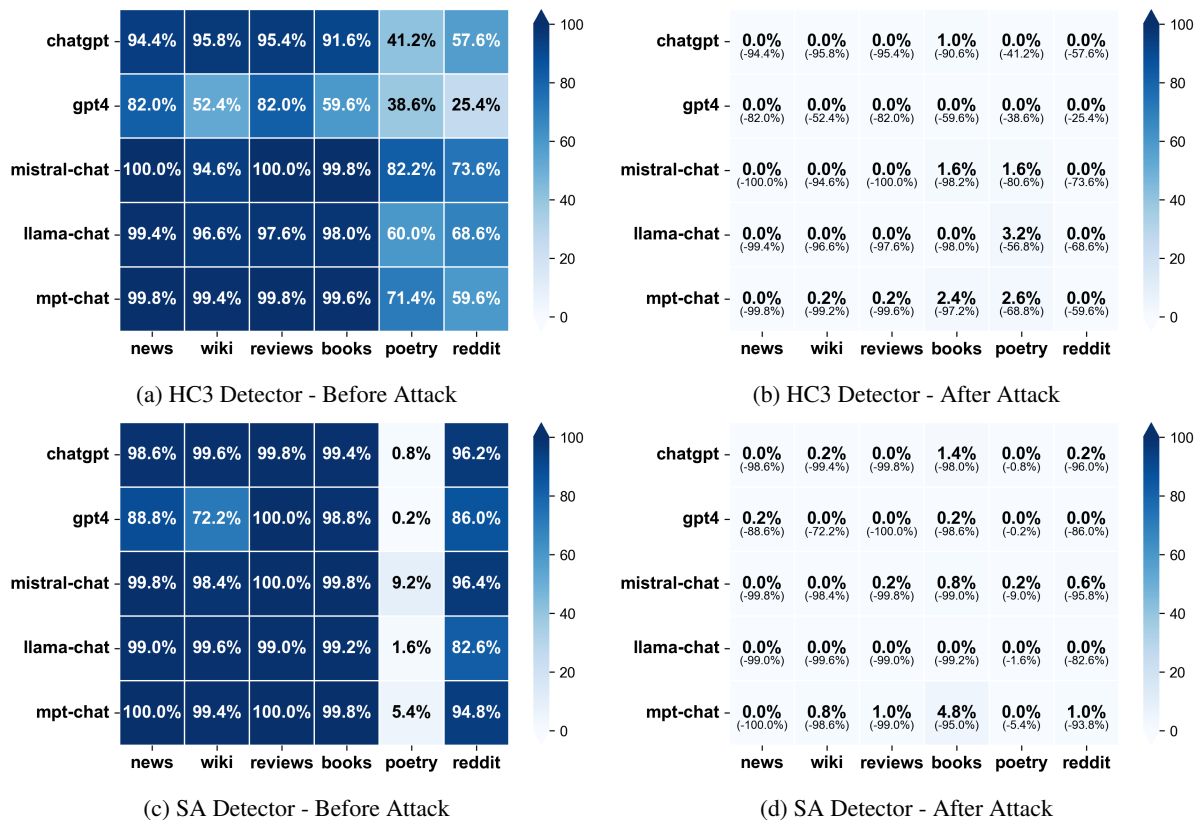|	news	wiki	reviews	books	poetry	reddit
chatgpt	0.0% (-98.6%)	0.2% (-99.4%)	0.0% (-99.8%)	1.4% (-98.0%)	0.0% (-0.8%)	0.2% (-96.0%)
gpt4	0.2% (-88.6%)	0.0% (-72.2%)	0.0% (-100.0%)	0.2% (-98.6%)	0.0% (-0.2%)	0.0% (-86.0%)
mistral-chat	0.0% (-99.8%)	0.0% (-98.4%)	0.2% (-99.8%)	0.8% (-99.0%)	0.2% (-9.0%)	0.6% (-95.8%)
llama-chat	0.0% (-99.0%)	0.0% (-99.6%)	0.0% (-99.0%)	0.0% (-99.2%)	0.0% (-1.6%)	0.0% (-82.6%)
mpt-chat	0.0% (-100.0%)	0.8% (-98.6%)	1.0% (-99.0%)	4.8% (-95.0%)	0.0% (-5.4%)	1.0% (-93.8%)

Figure 14: Detection accuracy of HC3 and SA before and after applying TempParaphraser, evaluated across multiple domains and LLMs.

Annotators were instructed to rate readability according to the ease and smoothness of reading a text, considering grammaticality, fluency, and naturalness of expression, but not factual correctness. The five-point scale was defined as follows: 1 = very poor (ungrammatical or extremely awkward), 2 = poor (frequent errors or unnatural phrasing), 3 = fair (acceptable but somewhat unnatural), 4 = good (fluent and mostly natural with minor issues), 5 = excellent (highly fluent, natural).

Text Type	Average Readability Score
AI-Generated	4.47
TempParaphraser-Processed	3.78
Human-Written	3.89

Table 7: Average readability scores for different text types based on human evaluation.

As shown in Table 7, the results indicate that AI-generated texts received the highest readability scores, with an average score of 4.47. In comparison, TempParaphraser-processed texts received slightly lower scores at 3.78, but still maintained fluency (scores > 3), with readability levels similar to human-generated texts (3.89). This trend aligns with the results from our PPL evaluation, further confirming that our method effectively simulates human-like text characteristics.

E.3 Using Different Detectors within the TempParaphraser Framework

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 8 illustrates this generalization capability.

Notably, TempParaphraser attacks detectors without requiring access to their internal weights, relying solely on their output probabilities.

E.4 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

TempParaphraser with	Detection Accuracy (%)				
	HC3 ↓	SA ↓	Fast-DetectGPT ↓	TOCSIN ↓	RADAR ↓
HC3	2.1	1.9	2.6	4.5	45.4
SA	19.2	≈ 0	2.7	3.7	44.4
Fast-DetectGPT	39.4	7.1	0	1.2	50.4

Table 8: Detection accuracy (%) of AI-text detectors when different detectors are used within the TempParaphraser framework for paraphrased outputs selection.

Model	Detection Accuracy (%)			Text Quality		
	HC3 ↓	SA ↓	Fast ↓	Flesch ↑	|ΔPPL| ↓	Sim ↑
Llama-3.2-1B-Instruct (Dubey et al., 2024)	45.6	13.7	8.5	66.747	8.785	0.963
Llama-3.2-3B-Instruct (Dubey et al., 2024)	49.8	18.6	11.0	66.280	9.666	0.966
Phi-2 (Javaheripi et al., 2023)	54.1	35.4	15.1	65.943	12.019	0.965
Qwen2.5-1.5B-Instruct (Qwen Team, 2024)	48.1	19.5	10.9	66.220	10.043	0.965

Table 9: Comparison of detection accuracy and text quality across different models. The experiment was conducted with hyperparameters: sampling times = 1 and temperature = 1.2.

the detection accuracy 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.
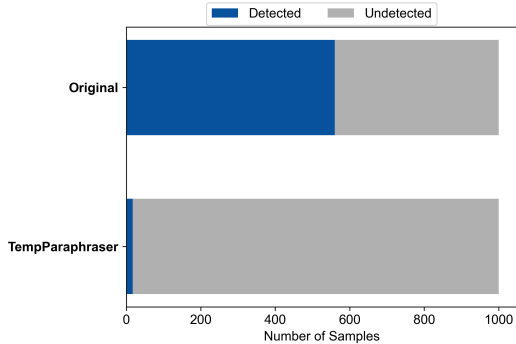
E.5 Attacking the Watermarking Methods

Figure 15: Detection results before and after applying TempParaphraser to watermarked texts.

To further evaluate the effectiveness of TempParaphraser against watermarking methods, we sampled 1,000 texts from the Dolly dataset (Conover et al., 2023) and employed the watermark injection framework proposed by Kirchenbauer et al. (2023) to embed watermarks into outputs generated by the Llama-3.1-8B-Instruct model (Dubey et al., 2024). These watermarked texts were then tested using the corresponding detection algorithm. Subsequently, we applied TempParaphraser to the watermarked texts to assess its ability to undermine watermark detection.

As shown in Figure 15, the original responses were detected as watermarked in 56% of the cases, whereas only 1.7% of the TempParaphraser-processed responses were detected. This sharp reduction demonstrates that TempParaphraser effectively reduces the success rate of watermark detection.

F Effectiveness of Paraphrasing Models Fine-Tuned from Different LLMs

To examine how the choice of paraphrasing model within the TempParaphraser framework impacts detection evasion, we fine-tune various LLMs on the same dataset using identical hyperparameters. The evaluation results are summarized in Table 9.

Our findings indicate that all tested models successfully reduce AI-text detection accuracy, confirming that diverse paraphrasing models can effectively evade detection.

G Token Frequency Analysis

In Section 5.3.1, we ran 5,000 inferences with the same input on Llama-3.2-3B-Instruct and recorded the frequency of token IDs at position $j = 8$. The hyperparameters for TempParaphraser were set to a sampling times parameter of $N = 7$ and a temperature of $T = 1.0$. Table 10 reports the token frequencies for the top 20 tokens under three different 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),

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

a few token IDs dominate the outputs, whereas at a higher temperature (1.9), the distribution becomes noticeably more dispersed. TempParaphraser produces a further redistribution of token probabilities, promoting a more varied token selection compared to standard temperature-based sampling. Notably, TempParaphraser reduces reliance on the highest-probability token (Token ID 802), thereby mitigating bias in LLM-generated text.

H Experimental Details for RoBERTa Fine-Tuning

H.1 Training Detail

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

- Dataset: HC3-text dataset (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.

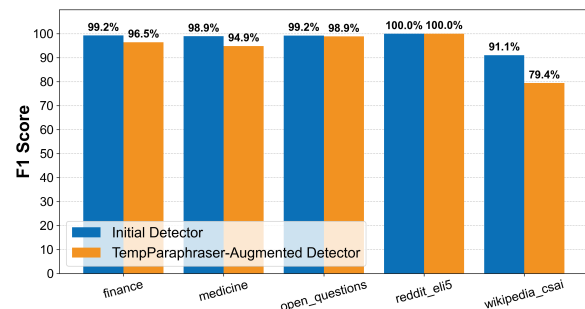### H.2 Further Evaluation of the TempParaphraser-Augmented Detector

Figure 16: Performance of the TempParaphraser-augmented detector on the HC3 test set across different domains.

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

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.

I Prompt Design

I.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."*

I.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."*

I.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 (Fishchuk 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."*