Pref-DetectGPT: Unveiling Machine-Generated Text via Preference-Aware Curvature Measurement

Jiahao Wang[1,2], Feifei Kou[1,2,3]*, Zhongbao Zhang[1], Jiwei Zhang[1], Lei Shi[4], Pengfei Zhang[5], Suguo Zhu[6], Mingying Xu[7]
[1]School of Computer Science (National Pilot School of Software Engineering), BUPT, Beijing 100876, China
[2]Key Laboratory of Trustworthy Distributed Computing and Service, BUPT, Beijing 100876, China
[3]Laboratory for Artificial Intelligence and New Forms of Education, CCNU, Wuhan 430079, China
[4]State Key Laboratory of Media Convergence and Communication, CUC, Beijing 100024, China
[5]School of Computer Science and Engineering, AUST, Anhui 232001, China
[6]College of Computer Science and Technology, HDU, Hangzhou 310018, China
[7]School of Artificial Intelligence and Computer Science, NCUT, Beijing 100144, China
koufeifei000@bupt.edu.cn

Abstract

The widespread deployment of large language models (LLMs) has made reliable detection of machine-generated text increasingly critical. Recent curvature-based detectors introduce perturbations to candidate text and compute curvature as the log probability difference between original and perturbed variants using advanced surrogate models. However, such curvature measurements only capture weak discrepancies between machine-generated and human-written text. We propose **Pref-DetectGPT**, a training-free method that introduces preference-aware curvature measurement by leveraging the implicit preference capabilities of preference-optimized surrogate models. Specifically, we construct an implicit reward model from the preference-optimized policy to score both original and perturbed text, and redefine curvature as their normalized deviation, which serves as a significantly stronger signal for distinguishing machine-generated text from human-written text. Empirical evaluations across multiple public benchmark datasets demonstrate that Pref-DetectGPT achieves state-of-the-art detection performance with relative improvements of 2.69%, 2.14%, and 9.04% in AUROC, AUPR, and TPR5%, while exhibiting strong robustness against various adversarial attacks and input lengths. Our code is available at https://github.com/BUPT-SN/Pref-DetectGPT.

1 Introduction

The rapid advancement of large language models (LLMs) [1, 2] has revolutionized text generation capabilities, producing content that increasingly resembles human writing. While these developments have enabled transformative applications across diverse domains, they simultaneously introduce critical societal challenges, including the spread of misinformation, academic dishonesty, and threats to content authenticity [3, 4, 5]. As the boundary between machine-generated and human-written text continues to blur, developing robust and reliable detection methods has become imperative.

Current detection approaches predominantly fall into two categories: training-based and training-free methods. Training-based detectors [6, 7, 8, 9, 10] achieve strong in-distribution performance but exhibit poor generalization in real-world scenarios [11, 12]. In contrast, training-free methods [13, 14, 15, 16, 17] offer a promising alternative by leveraging surrogate models to exploit intrinsic statistical properties of candidate text, without requiring labeled data. Since the true source model

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

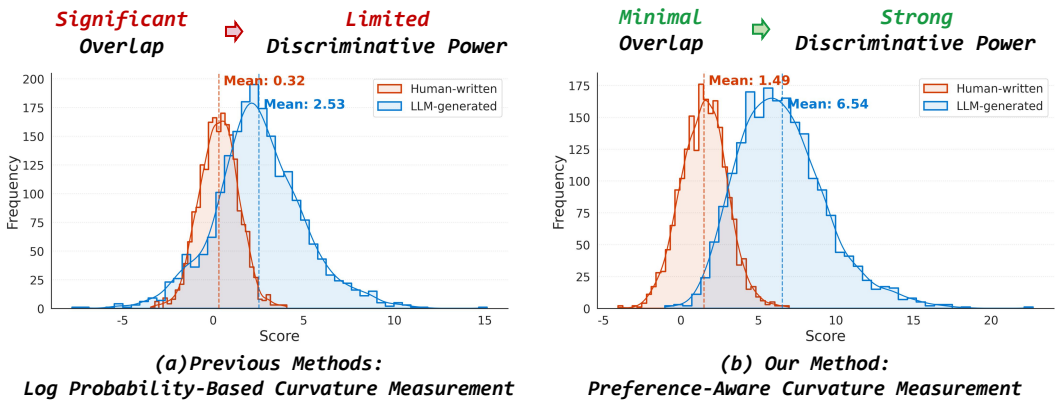

Figure 1: Distribution comparison of Fast-DetectGPT score and Pref-DetectGPT (ours) score for human-written and LLM-generated texts from the *level 1* sub-task of the HART benchmark.

is typically unknown, these methods use accessible pretrained LLMs as surrogates for scoring. Among these, curvature-based approaches have demonstrated particular effectiveness by introducing perturbations to candidate text and measuring the scoring difference between original and perturbed variants. DetectGPT [13] pioneered this paradigm by using a masked language model to mask and refill tokens, generating multiple perturbed variants, then comparing their log probabilities under a surrogate model to estimate curvature. However, each perturbation requires evaluating the full Markov chain probability, leading to prohibitive computational cost with multiple forward passes. Fast-DetectGPT [14] addresses this by introducing an autoregressive model as the sampling model to perform conditional probability sampling: it generates alternative token choices at each position and computes their conditional probabilities in a single forward pass, leveraging the fact that all alternatives share the same predictive distribution, thereby eliminating repeated evaluation while improving accuracy.

However, existing curvature-based methods exhibit limited discriminative power in distinguishing machine-generated from human-written text. Recent work has shown that using more advanced LLMs as surrogate models yields better detection performance [18]. Following this finding, we employ an advanced instruction-tuned model (e.g., Llama-3.2-1B-Instruct) as the surrogate model and its base model counterpart as the sampling model, and observe that the curvature distributions of machine-generated and human-written text still show significant overlap when measured through log probability curvature (as illustrated in Figure 1 (a)). We identify the root cause: existing curvature-based methods compute curvature via log probability differences, which only exploits the SFT objective of likelihood maximization. This generative objective evaluates text based on surface-level properties such as fluency and coherence, precisely the aspects that modern LLMs have become increasingly capable of mimicking, making log probability curvature an inherently less robust discriminative signal. While modern instruction-tuned surrogate models have undergone extensive preference optimization through RLHF [19] and DPO [20], which explicitly trains models to capture preference signals beyond surface-level properties, existing methods fail to exploit these powerful implicit preference capabilities.

Building on this observation, we propose **Pref-DetectGPT**, a training-free method that introduces preference-aware curvature measurement by leveraging the implicit preference capabilities embedded in preference-optimized models. We derive implicit reward models for both the original text and perturbed variants, enabling curvature to be measured through preference differences. Crucially, this formulation directly aligns with the implicit reward objective of DPO: by treating the original text as the preferred response y_w and its perturbed variants as the dispreferred response y_l, our method measures exactly the reward margin that surrogate models were trained to amplify. Given a candidate text, we compute the implicit reward of the original and the expected implicit reward of perturbed variants sampled via conditional probability sampling, then measure their normalized deviation as our curvature metric. As illustrated in Figure 1 (b), machine-generated text exhibits strong preference discrepancy where the original achieves notably higher reward than perturbations, while human-written text shows weak preference discrepancy.

Our main contributions are summarized as follows:

- We identify a key limitation of existing curvature-based detectors: they fail to fully exploit the powerful implicit preference capabilities embedded in modern surrogate models through preference optimization, relying solely on log probabilities that primarily reflect the supervised fine-tuning objective.
- We propose Pref-DetectGPT, a training-free detection method that introduces preference-aware curvature measurement by constructing implicit reward models for both original and perturbed text, enabling us to measure curvature through preference differences, a stronger signal than log probability differences.
- Extensive experiments demonstrate that Pref-DetectGPT achieves state-of-the-art performance across multiple benchmarks, while exhibiting exceptional robustness against adversarial attacks and maintaining superior performance across different text lengths.

2 Related Work

Machine-Generated Text Detection. The proliferation of large language models has catalyzed extensive research into distinguishing machine-generated text from human-written content. Training-based methods collect labeled data and train classifiers to distinguish human-written and machine-generated texts [6, 21, 22], but often suffer from overfitting and limited generalization to out-of-distribution texts [11, 12]. In contrast, training-free methods leverage probabilistic characteristics of pretrained LLMs without requiring labeled data, demonstrating better generalization ability. Single-model methods measure basic statistics such as likelihood, entropy, and log-rank [23] directly from a surrogate model, while multi-model methods have demonstrated stronger discriminative performance by exploiting complementary signals across models [15, 24, 16]. Among these, curvature-based methods generate perturbed text variants and measure how model probability assessments shift between original and perturbed text. DetectGPT [13] pioneered this paradigm by observing that machine-generated text achieves higher log probability than its perturbations, while Fast-DetectGPT [14] advanced it via token-level conditional probability sampling, enabling curvature computation in a single forward pass. However, these methods all measure curvature with log probabilities, which reflect only surface-level properties such as fluency and coherence. As modern LLMs excel at mimicking these properties, such a detection signal becomes increasingly unreliable.

Preference Optimization and Implicit Rewards. DPO [20] has emerged as an efficient alternative to RLHF [19] for aligning language models with human preferences. A key insight from DPO is its closed-form solution, which implicitly defines a reward function as the log-likelihood ratio between the policy model and reference model. This implicit reward captures the preference structure learned during alignment, quantifying how well responses align with human preferences beyond surface-level probability. As modern surrogate models predominantly undergo such preference optimization, they embed rich implicit preference capabilities that could provide stronger discriminative signals for detection. However, although existing work has recognized the value of implicit rewards for detection [25], the more direct preference structure behind them remains unexplored.

Our method is a training-free detector that bridges these two lines of work: it extends the curvature-based detection paradigm into the implicit reward space, leveraging the rich preference structure embedded in alignment-trained surrogate models to achieve more powerful discriminative signals.

3 Preliminaries

Our method is motivated by two key observations: (1) curvature-based detection strategies effectively distinguish machine-generated text from human-written text by measuring conditional probability curvature, and (2) modern LLMs predominantly undergo preference optimization during alignment, exhibiting strong preference responsiveness that can be leveraged for detection. We review these foundational concepts below.

3.1 Conditional Probability Sampling

DetectGPT [13] pioneered curvature-based detection by quantifying the *probability curvature*:

$$d(y, p_\theta, q_\phi) = \log p_\theta(y) - \mathbb{E}_{\tilde{y}\sim q_\phi(\cdot|y)}[\log p_\theta(\tilde{y})], \tag{1}$$

where $\tilde{y}$ is a perturbed passage produced by a masked language model $q_\phi(\cdot|y)$ (e.g., T5 model). Machine-generated text exhibits positive curvature as perturbations yield lower probability, while human-written text shows near-zero curvature. However, evaluating $p_\theta(\tilde{y})$ for each perturbation requires multiple forward passes through the entire Markov chain, incurring substantial computational cost.

To address this inefficiency, Fast-DetectGPT [14] introduces *conditional probability sampling* by replacing the joint probability with the conditional log probability:

$$\log p_\theta(\tilde{y}_i|y_{<i}) = \sum_{w\in\mathcal{V}} q_\phi(w|y_{<i}) \cdot \log p_\theta(w|y_{<i}), \tag{2}$$

where $\mathcal{V}$ denotes the vocabulary, and alternative tokens $\tilde{y}_i$ are independently sampled from the sampling model q_ϕ at position i, conditioned on the original context $y_{<i}$. Specifically, considering the input text y of length L, the conditional probability curvature is:

$$d(y, p_\theta, q_\phi) = \frac{\log p_\theta(y|y) - \tilde{\mu}}{\tilde{\sigma}}, \tag{3}$$

where $\log p_\theta(y|y) = \sum_{i=1}^{L} \log p_\theta(y_i|y_{<i})$ is the log probability of the original text, and $\tilde{\mu}$ and $\tilde{\sigma}$ are the expected mean and standard deviation of the log probabilities for perturbed samples $\tilde{y}$:

$$\begin{aligned} \tilde{\mu} &= \mathbb{E}_{\tilde{y}\sim q_\phi(\tilde{y}|y)}[\log p_\theta(\tilde{y}|y)], \\ \tilde{\sigma}^2 &= \mathbb{E}_{\tilde{y}\sim q_\phi(\tilde{y}|y)}[(\log p_\theta(\tilde{y}|y) - \tilde{\mu})^2]. \end{aligned} \tag{4}$$

By expanding these expectations with the conditional probability formulation in Eq. (2), we can compute them efficiently as:

$$\tilde{\mu} = \sum_{i=1}^{L} \sum_{w\in\mathcal{V}} q_\phi(w|y_{<i}) \cdot \log p_\theta(w|y_{<i}), \tag{5}$$

and:

$$\tilde{\sigma}^2 = \sum_{i=1}^{L} \sum_{w\in\mathcal{V}} q_\phi(w|y_{<i}) \cdot (\log p_\theta(w|y_{<i}) - \bar{\mu}_i)^2, \tag{6}$$

where $\bar{\mu}_i = \sum_{w\in\mathcal{V}} q_\phi(w|y_{<i}) \cdot \log p_\theta(w|y_{<i})$ is the expected log probability at position i. This formulation enables evaluation using a single forward pass of the scoring model p_θ, dramatically reducing computational cost while preserving detection effectiveness.

3.2 Preference Optimization

Direct Preference Optimization (DPO) [20] is derived from the optimization objective of Proximal Policy Optimization (PPO) [19] in Reinforcement Learning from Human Feedback (RLHF). PPO aims to align the policy model with human preferences by maximizing the expected reward while constraining the model to stay close to a reference model:

$$\max_\theta \mathbb{E}_{y\sim p_\theta(\cdot|x)}[r(x, y)] - \beta\, \mathrm{D}_{\mathrm{KL}}[p_\theta(y|x)\|p_{\text{ref}}(y|x)], \tag{7}$$

where x is a prompt, y is a text sampled from the policy model p_θ, $r(x, y)$ is a pre-trained reward model that judges text quality, and β controls the KL divergence penalty [26] from the reference model p_{ref}. By re-parameterizing this objective, DPO eliminates the need for explicit reward model training and directly optimizes:

$$\max_\theta \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma\left(r(x, y_w) - r(x, y_l)\right)\right], \tag{8}$$

where y_w and y_l denote the winner (preferred) and loser (dispreferred) texts respectively, and

$$r(x, y) = \beta \log \frac{p_\theta(y|x)}{p_{\text{ref}}(y|x)} \tag{9}$$

represents the implicit reward derived from the likelihood ratio between the policy model and reference model. This formulation enables direct policy optimization from preference pairs without requiring reward model training or reinforcement learning.

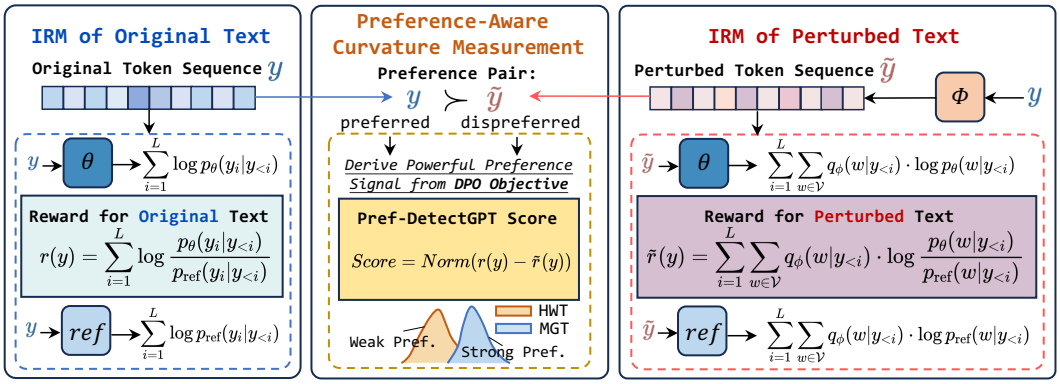

Figure 2: Overview of the Pref-DetectGPT framework. (*Left*) **IRM of Original Text:** We compute the implicit reward $r(y)$ using the log-probability ratio between the policy model θ and the reference model ref. (*Right*) **IRM of Perturbed Text:** We estimate the expected reward $\tilde{r}(y)$ by aggregating token probabilities over the vocabulary $\mathcal{V}$, weighted by the sampling distribution from ϕ. (*Middle*) **Preference-Aware Curvature Measurement:** The final score quantifies the normalized deviation between the original and perturbed rewards. Machine-generated text exhibits strong preference discrepancy where $r(y)$ significantly exceeds $\tilde{r}(y)$, while human-written text shows weak preference discrepancy with $r(y)$ closer to $\tilde{r}(y)$.

4 Methodology

4.1 Overview of Pref-DetectGPT

Our framework introduces preference-aware curvature measurement for machine-generated text detection, as illustrated in Figure 2. The key innovation lies in measuring curvature in the implicit reward space rather than the probability space.

Our approach consists of three main components. First, we construct the *Implicit Reward Model (IRM) for Original Text* by computing the implicit reward $r(y)$ as the log-probability ratio between a policy model p_θ and a reference model p_{ref} (Figure 2 (left)). While prior work [25] has used such implicit rewards for detection, we extend this framework to handle perturbations. Second, we derive the *IRM for Perturbed Text* by computing the expected reward $\tilde{r}(y)$ over perturbed token distributions, aggregating token probabilities across the vocabulary weighted by a sampling distribution q_ϕ (Figure 2 (right)). This formulation, inspired by Fast-DetectGPT's conditional probability sampling [14], enables efficient computation through a single forward pass. Third, we introduce *Preference-Aware Curvature Measurement* that quantifies the normalized deviation between $r(y)$ and $\tilde{r}(y)$ (Figure 2 (middle)). Machine-generated text exhibits strong preference discrepancy (positive curvature) as it occupies reward maxima, while human text shows weak discrepancy (near-zero curvature).

4.2 Perturbed Implicit Reward Model

Unlike log probabilities, the implicit reward for perturbed text is not directly tractable. We first derive the implicit reward for the original text, then extend it to perturbed text via conditional probability sampling. DPO [20] provides a closed-form solution for the optimal policy under the RLHF objective:

$$p_\theta^*(y|x) = \frac{1}{Z(x)} p_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x,y)\right), \tag{10}$$

where $Z(x) = \sum_y p_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x,y)\right)$ is the partition function that normalizes the distribution. By rearranging this equation, we can express the reward as a function of the policy and reference models, yielding an implicit reward model:

$$r(x,y) = \beta \log \frac{p_\theta(y|x)}{p_{\text{ref}}(y|x)} + \beta \log Z(x). \tag{11}$$

For our detection task, we work directly with text without prompts by setting $x = \emptyset$ and letting y be the detected text. In this prompt-free setting, the partition function term $\beta \log Z(x)$ becomes a constant shared across all detected texts. Since we ultimately compute the reward difference between original and perturbed texts, constant terms that appear in both expressions cancel out and can be omitted without loss of generality. The implicit reward for the original text can be expressed as:

$$r(y) = \beta \sum_{i=1}^{L} \log \frac{p_\theta(y_i|y_{<i})}{p_{\text{ref}}(y_i|y_{<i})}, \tag{12}$$

where L is the length of text y, y_i denotes the i-th token.

To derive the implicit reward for perturbed texts, we first express the reward for a perturbed token $\tilde{y}_i$ at position i by substituting it into the implicit reward formulation:

$$r(\tilde{y}_i|y_{<i}) = \beta \log \frac{p_\theta(\tilde{y}_i|y_{<i})}{p_{\text{ref}}(\tilde{y}_i|y_{<i})}. \tag{13}$$

We generate these perturbations through conditional probability sampling following Fast-DetectGPT [14]: at each position i, alternative tokens are sampled from a sampling distribution $q_\phi(\cdot|y_{<i})$ over the vocabulary $\mathcal{V}$, conditioned on the original context $y_{<i}$.

Taking the expectation over this sampling distribution, the expected implicit reward at position i becomes:

$$r(\tilde{y}_i|y_{<i}) = \beta \sum_{w \in \mathcal{V}} q_\phi(w|y_{<i}) \cdot \log \frac{p_\theta(w|y_{<i})}{p_{\text{ref}}(w|y_{<i})}. \tag{14}$$

Aggregating across all positions yields the expected implicit reward for the entire perturbed text:

$$\tilde{r}(y) = \beta \sum_{i=1}^{L} \sum_{w \in \mathcal{V}} q_\phi(w|y_{<i}) \cdot \log \frac{p_\theta(w|y_{<i})}{p_{\text{ref}}(w|y_{<i})}. \tag{15}$$

This formulation enables efficient computation through a single forward pass of both the policy model p_θ and the reference model p_{ref}, while capturing the expected implicit reward landscape under perturbations.

4.3 Preference-Aware Curvature Measurement

Building upon the perturbed implicit reward model, we measure the *curvature* of the implicit reward function as the normalized deviation of the original text's reward from the expected reward of its perturbed variants. This curvature captures the local geometry of the reward landscape around the original text.

As the scaling factor β appears in both $r(y)$ and $\tilde{r}(y)$, it cancels out exactly in the normalized deviation metric and can be omitted. Therefore, we work with the simplified forms:

$$r(y) = \sum_{i=1}^{L} \log \frac{p_\theta(y_i|y_{<i})}{p_{\text{ref}}(y_i|y_{<i})}, \tag{16}$$

and:

$$\tilde{r}(y) = \sum_{i=1}^{L} \sum_{w \in \mathcal{V}} q_\phi(w|y_{<i}) \cdot \log \frac{p_\theta(w|y_{<i})}{p_{\text{ref}}(w|y_{<i})}. \tag{17}$$

We define the preference-aware curvature as a normalized deviation score:

$$d(y, p_\theta, p_{\text{ref}}, q_\phi) = \frac{r(y) - \tilde{\mu}_r}{\tilde{\sigma}_r}, \tag{18}$$

where $\tilde{\mu}_r$ and $\tilde{\sigma}_r$ are the expected mean and standard deviation of the implicit rewards for perturbed samples. By expanding the expectations using Eq. (17), we can compute them efficiently as:

$$\tilde{\mu}_r = \sum_{i=1}^{L} \sum_{w \in \mathcal{V}} q_\phi(w|y_{<i}) \cdot \log \frac{p_\theta(w|y_{<i})}{p_{\text{ref}}(w|y_{<i})}, \tag{19}$$

and:

$$\tilde{\sigma}_r^2 = \sum_{i=1}^{L} \sum_{w \in \mathcal{V}} q_\phi(w|y_{<i}) \cdot \left(\log \frac{p_\theta(w|y_{<i})}{p_{\text{ref}}(w|y_{<i})} - \bar{r}_i \right)^2 \tag{20}$$

$$\bar{r}_i = \sum_{w \in \mathcal{V}} q_\phi(w|y_{<i}) \cdot \log \frac{p_\theta(w|y_{<i})}{p_{\text{ref}}(w|y_{<i})}, \tag{21}$$

where $\bar{r}_i$ is the expected perturbed implicit reward at position i.

4.4 Detection Decision

Given the preference-aware curvature $d(y, p_\theta, p_{\text{ref}}, q_\phi)$, we make the final detection decision using a threshold τ:

$$\text{Label}(y) = \begin{cases} \text{Machine-generated}, & \text{if } d > \tau \\ \text{Human-written}, & \text{otherwise} \end{cases} \tag{22}$$

The threshold τ should be chosen to balance the false and true positive rates in practice. A higher curvature indicates the original text occupies a reward maximum characteristic of machine generation, while a lower curvature suggests human authorship with reward closer to the perturbed average.

5 Experiments

5.1 Experimental Setup

Datasets. We conduct experiments on three established benchmarks: DetectRL [27], HART [28], and MIRAGE [9]. (1) DetectRL contains human-written texts from 4 domains paired with machine-generated texts from 4 mainstream LLMs. We evaluate on its *Multi-Domain* setting for cross-domain generalization. (2) HART is a multi-level hybrid text detection benchmark covering 4 domains and 6 recent LLMs. We focus on *Level 1* which detects AI involvement. (3) MIRAGE spans 5 domains and 17 LLMs across two generation scenarios: Disjoint-Input Generation (*DIG*) and Shared-Input Generation (*SIG*). For all benchmarks, we randomly sample balanced subsets of 2,000 human-written and 2,000 machine-generated texts. More details are in Appendix A.1.

Baselines. For a comprehensive comparison, we compare Pref-DetectGPT with existing training-based and training-free methods. For training-based detectors, we evaluate *RoBERTa-Base* and *RoBERTa-Large* [6], *BiScope* [7], *ImBD* [8], and *DetectAnyLLM* [9]. For training-free methods, we include basic training-free detectors such as *Likelihood*, *LogRank*, *Entropy* [23], and *LRR* [29], along with advanced approaches including *Lastde++* [16], *Fast-DetectGPT* [14], *Binoculars* [15], *DNA-DetectLLM* [24], and *IRM* [25]. We exclude *DetectGPT* [13], *NPR* [29], and *DNA-GPT* [30] due to their high computational requirements.

Metrics. We adopt AUROC, AUPR, and TPR @ 5% FPR (simplified as TPR5%) to evaluate detection performance. AUROC measures the overall ranking capability of detectors across all classification thresholds. AUPR evaluates the precision-recall trade-off. TPR5% is particularly important for real-world applications, as it measures detection capability while keeping false accusations of human content at an acceptable low level.

Implementation Details. All experiments are conducted on two NVIDIA GeForce RTX 4090 GPUs (24GB each). For training-based methods, we uniformly use the training dataset provided by IMBD [8] to ensure fair comparison. To eliminate the influence of model choice on performance, we standardize the model configuration across all methods using Llama-3.2-1B-Instruct and Llama-3.2-1B, with detailed configurations provided in Appendix A.2. During testing, the maximum input token length is set to 512.

5.2 Main Results

Table 1 presents a comprehensive evaluation of Pref-DetectGPT against both training-based and training-free baselines across four benchmark datasets: DetectRL (*Multi-Domain*), HART (*Level-1*), MIRAGE (*DIG*), and MIRAGE (*SIG*). Our method achieves state-of-the-art performance across all evaluation scenarios, demonstrating substantial improvements over existing approaches.

Table 1: Comparison of detection performance on public benchmark datasets, including DetectRL, HART, MIRAGE (*DIG*), and MIRAGE (*SIG*). The metrics are AUROC, AUPR and TPR 5%. Best results are **bolded**, and second-best results are underlined.

Methods	DetectRL			HART			MIRAGE (*DIG*)			MIRAGE (*SIG*)		
	AUROC	AUPR	TPR5%	AUROC	AUPR	TPR5%	AUROC	AUPR	TPR5%	AUROC	AUPR	TPR5%
Training-based Methods												
RoBERTa-Base	0.7804	0.8379	0.5145	0.8746	0.8952	0.6005	0.9325	0.9415	0.7280	0.9257	0.9370	0.7135
RoBERTa-Large	0.8281	0.8654	0.5255	0.9044	0.9202	0.6715	0.9655	0.9677	0.8569	0.9649	0.9713	0.8590
BiScope	0.8123	0.8610	0.5694	0.8841	0.8909	0.5560	0.9505	0.9496	0.7605	0.9404	0.9404	0.6809
ImBD	0.8643	0.8777	0.5510	0.9179	0.9319	0.7220	0.9049	0.8966	0.5220	0.8959	0.8909	0.5125
DetectAnyLLM	0.8702	0.9111	0.7270	0.9418	0.9418	0.8265	0.9683	0.9745	0.9165	0.9633	0.9716	0.9130
Training-free Methods												
Likelihood	0.8140	0.8648	0.5935	0.7294	0.7493	0.3140	0.7656	0.7645	0.2945	0.7617	0.7564	0.2940
LogRank	0.8140	0.8658	0.6115	0.7285	0.7612	0.3250	0.7676	0.7754	0.2980	0.7625	0.7621	0.3210
Entropy	0.7793	0.8409	0.5605	0.7241	0.7487	0.2950	0.6660	0.6421	0.1280	0.6637	0.6338	0.1090
LRR	0.7958	0.8468	0.5420	0.7028	0.7581	0.3525	0.7426	0.7618	0.2840	0.7344	0.7390	0.2695
Lastde++	0.7146	0.7313	0.2505	0.6916	0.7048	0.2250	0.9384	0.9375	0.7130	0.9354	0.9351	0.7030
Fast-DetectGPT	0.9118	0.9314	0.7255	0.8118	0.8616	0.5815	0.9663	0.9715	0.8685	0.9658	0.9728	0.8755
Binoculars	0.9213	0.9399	0.7550	0.8173	0.8644	0.5825	0.9631	0.9678	0.8385	0.9634	0.9704	0.8630
DNA-DetectLLM	0.9314	0.9478	0.7810	0.8178	0.8651	0.5850	0.9639	0.9683	0.8502	0.9630	0.9692	0.8395
IRM	0.9661	0.9721	0.8605	0.8832	0.9033	0.6305	0.9883	0.9904	0.9515	0.9840	0.9875	0.9475
Pref-DetectGPT	**0.9741**	**0.9785**	**0.8995**	**0.9595**	**0.9647**	**0.8295**	**0.9962**	**0.9968**	**0.9835**	**0.9947**	**0.9956**	**0.9840**

Table 2: Comparison of adversarial robustness (AUROC). Best results are **bolded**, and second-best results are underlined.

Methods	Prompt	Paraph.	Perturb	Mixing	Avg.
RoBERTa-Base	0.8950	0.9300	0.5123	0.7643	0.7754
RoBERTa-Large	0.9107	0.9371	0.6260	0.7940	0.8170
BiScope	0.9456	0.8185	0.6673	0.7311	0.7906
ImBD	0.9071	0.8923	0.8454	0.8647	0.8774
DetectAnyLLM	0.9480	0.8344	0.8571	0.8955	0.8838
Likelihood	0.9356	0.7845	0.6907	0.7933	0.8010
LogRank	0.9308	0.7882	0.6893	0.7950	0.8008
Entropy	0.9507	0.7542	0.5814	0.7082	0.7486
LRR	0.8888	0.7766	0.6578	0.7709	0.7735
Lastde++	0.8625	0.8788	0.6380	0.7176	0.7742
Fast-DetectGPT	0.9495	0.9258	0.8764	0.8912	0.9107
Binoculars	0.9518	0.9127	0.8835	0.8989	0.9117
DNA-DetectLLM	0.9634	0.8993	0.9003	0.8973	0.9151
IRM	0.9401	0.9447	0.9518	0.9386	0.9438
Pref-DetectGPT	**0.9809**	**0.9830**	**0.9826**	**0.9603**	**0.9767**

Comparison with Training-Based Methods. While training-based methods demonstrate competitive performance within their training distributions, they face inherent limitations in generalization. The best-performing training-based method, DetectAnyLLM, achieves 87.02% AUROC on DetectRL, while Pref-DetectGPT substantially outperforms with 97.41% AUROC, a relative improvement of 11.94%. Moreover, as they are fitted to labeled data from a fixed set of generators and domains, training-based detectors fluctuate markedly once the test distribution shifts, with ImBD ranging from 72.20% TPR5% on HART down to 51.25% on MIRAGE (*SIG*), whereas Pref-DetectGPT stays above 95.95% AUROC and 82.95% TPR5% on every benchmark. Notably, Pref-DetectGPT achieves these results without requiring any training data, as our preference-aware curvature measurement captures fundamental generation patterns that generalize across domains and models.

Comparison with Training-Free Methods. Among training-free approaches, Pref-DetectGPT shows substantial improvements over all baselines. Compared to Fast-DetectGPT, our method achieves relative average improvements of 7.35% in AUROC, validating that curvature measurement in the implicit reward space provides more discriminative signals than conditional probability perturbations alone. Against the strongest baseline IRM, Pref-DetectGPT demonstrates consistent superiority with relative average improvements of 2.69% in AUROC and 9.04% in TPR5%, with particularly pronounced gains on HART (8.64% relative AUROC improvement) and in TPR5%, highlighting superior detection rates at low false positive rates critical for practical deployment.

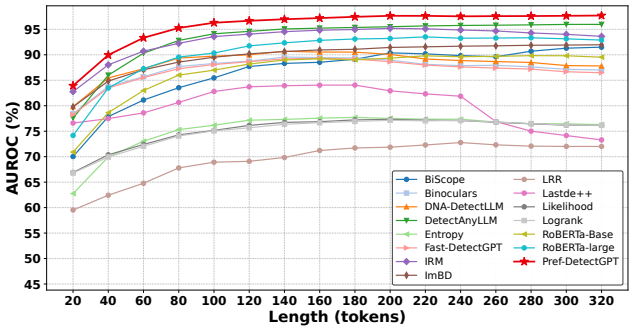

Figure 3: Detection performance (AUROC) on different input text length (from 20 to 320 tokens). Our method demonstrates robust performance across varying text lengths.

5.3 Robustness Analysis

Robustness on Various Adversarial Attacks. To evaluate the practical applicability of Pref-DetectGPT, we assess its robustness against various adversarial attacks using the DetectRL (*Multi-Attack*) dataset (detailed descriptions in Appendix A.1.1). As shown in Table 2, Pref-DetectGPT achieves the best average AUROC of 97.67%, surpassing IRM by a relative improvement of 3.49% and DNA-DetectLLM by 6.73%, with particularly strong resilience against perturbation attacks (98.26% AUROC), owing to our preference-aware curvature measurement that operates in the implicit reward space and thus remains stable under token-level alterations. Overall, Pref-DetectGPT steadily maintains over 96% AUROC under all four attacks, suggesting that the preference signal it relies on is less sensitive to the surface-level properties that adversarial rewriting modifies.

Robustness on Different Text Lengths. To evaluate the stability of our method under data-constrained conditions, we randomly sample texts from HART (*Level 1*) and truncate them to 20-320 tokens, with the setup detailed in Appendix A.1.3. As shown in Figure 3, Pref-DetectGPT ranks first at all 16 lengths, crossing 95% AUROC with only 80 tokens, whereas DetectAnyLLM needs 140 and IRM needs 200, and no other baseline reaches 95% at any length. At 320 tokens it attains 97.71% against 95.97% for DetectAnyLLM, while several curvature-based baselines peak near 140-200 tokens and then degrade. This indicates that preference-aware curvature yields a stronger discriminative signal per token, allowing accurate detection to be reached with far less text than prior detectors require.

5.4 Efficiency Analysis

Detection speed determines whether a detector can support large-scale or real-time monitoring, so we measure the per-sample inference cost of all detectors on 1,000 randomly sampled HART texts truncated to exactly 100 tokens with a batch size of 1, which removes text length as a confounding factor. As shown in Figure 4, the fastest detectors are also the weakest: RoBERTa-Large runs at 0.076 s/1k tokens but relies on supervised training, and the classical statistics Likelihood, LogRank, and Entropy stay around 0.10 s/1k tokens at only 70.83%-76.82% average AUROC. Pref-DetectGPT requires 0.307 s/1k tokens, comparable to Fast-DetectGPT (0.218) and IRM (0.201) and 2.2-3.8$\times$ faster than Binoculars (0.687) and DNA-DetectLLM (1.161), while achieving the best detection performance (98.11% average AUROC), an optimal trade-off between effectiveness and efficiency for practical deployment.

5.5 Ablation Study

To validate that Pref-DetectGPT's superior detection capability stems from the preference-aware curvature measurement rather than any specific model configuration, we evaluate three LLM pair configurations across four benchmark datasets, with results averaged and reported in Table 3. Pref-DetectGPT outperforms both DNA-DetectLLM and IRM under all three configurations, reaching 98.11%, 91.06%, and 82.65% AUROC with the `Llama-3.2-1B`, `Gemma-2-2B`, and `Qwen2.5-1.5B`

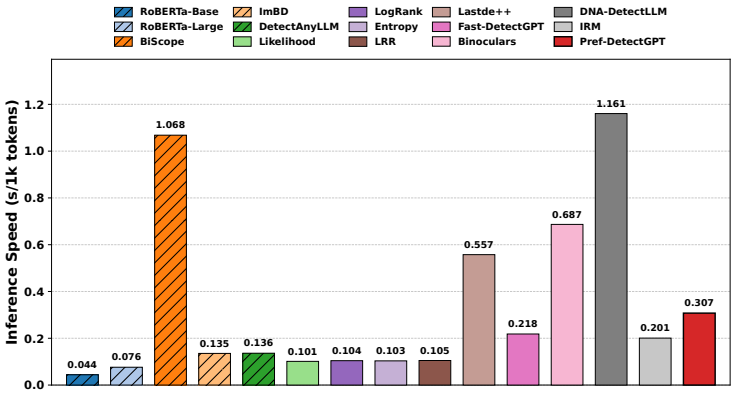

Figure 4: Comparison of inference speed measured in seconds per 1,000 tokens (s/1k tokens).

Table 3: AUROC comparison across different model pair configurations, averaged across four benchmark datasets.

Model Pair	DNA-DetectLLM	IRM	Pref-DetectGPT
`Llama-3.2-1B-Instruct + Llama-3.2-1B`	0.9190	0.9554	**0.9811**
`Qwen2.5-1.5B-Instruct + Qwen2.5-1.5B`	0.6878	0.7986	**0.8265**
`Gemma-2-2B-it + Gemma-2-2B`	0.8791	0.8757	**0.9106**

pairs and surpassing the strongest competitor by 2.57, 3.15, and 2.79 points. This consistent margin confirms that the gains stem from our preference-aware curvature measurement rather than a specific backbone, while leaving room for further improvement through more capable LLM pairings.

6 Conclusion

In this work, we present Pref-DetectGPT, a training-free detection method that introduces preference-aware curvature measurement by leveraging the implicit preference capabilities embedded in preference-optimized models. We construct implicit reward models for both original and perturbed text, enabling us to measure curvature through preference differences rather than log probability differences, as preference signals capture richer discriminative information beyond the surface-level fluency that modern LLMs excel at mimicking. Our comprehensive experiments across multiple benchmarks demonstrate the superiority of Pref-DetectGPT. We hope our work inspires future research to exploit implicit preference capabilities of modern LLMs for more effective detection.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (No.62677010), by the Beijing Natural Science Foundation (No.L2603037), by the Science and Technology Assistance Funding Program of Beijing University of Posts and Telecommunications (No.2026KJBF02), by the Open Fund of Laboratory for Artificial Intelligence and New Forms of Education CCNU (No.AIEL26C01), by the Key Laboratory of Public Opinion Governance and Computational Communication (No.YQKFYB202501), and by the Fundamental Research Funds for the Central Universities (No.CUC26TD13). The authors declare no competing interests.

References

[1] OpenAI. Gpt-4o system card. arXiv preprint arXiv:2410.21276, 2024.

[2] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with ad-

vanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261, 2025.

[3] Jesse G Meyer, Ryan J Urbanowicz, Patrick CN Martin, Karen O'Connor, Ruowang Li, Pei-Chen Peng, Tiffani J Bright, Nicholas Tatonetti, Kyoung Jae Won, Graciela Gonzalez-Hernandez, et al. Chatgpt and large language models in academia: opportunities and challenges. BioData mining, 16(1):20, 2023.

[4] Chengyuan Deng, Yiqun Duan, Xin Jin, Heng Chang, Yijun Tian, Han Liu, Yichen Wang, Henry Peng Zou, Yijia Xiao, Shenghao Wu, et al. Deconstructing the ethics of large language models from long-standing issues to new-emerging dilemmas: A survey. AI and Ethics, pages 1–27, 2025.

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

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

[7] Hanxi Guo, Siyuan Cheng, Xiaolong Jin, Zhuo Zhang, Kaiyuan Zhang, Guanhong Tao, Guangyu Shen, and Xiangyu Zhang. Biscope: Ai-generated text detection by checking memorization of preceding tokens. Advances in Neural Information Processing Systems, 37:104065–104090, 2024.

[8] Jiaqi Chen, Xiaoye Zhu, Tianyang Liu, Ying Chen, Chen Xinhui, Yiwen Yuan, Chak Tou Leong, Zuchao Li, Long Tang, Lei Zhang, et al. Imitate before detect: Aligning machine stylistic preference for machine-revised text detection. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 39, pages 23559–23567, 2025.

[9] Jiachen Fu, Chun-Le Guo, and Chongyi Li. Detectanyllm: Towards generalizable and robust detection of machine-generated text across domains and models. In Proceedings of the 33rd ACM International Conference on Multimedia, pages 11229–11238, 2025.

[10] Yuhan Yao, Feifei Kou, Lei Shi, Zhongbao Zhang, Suguo Zhu, Jiwei Zhang, Lirong Qiu, LI Haisheng, et al. Ostar: Optimized statistical text-classifier with adversarial resistance. Advances in Neural Information Processing Systems, 38:82193–82216, 2026.

[11] Soumya Suvra Ghosal, Souradip Chakraborty, Jonas Geiping, Furong Huang, Dinesh Manocha, and Amrit Bedi. A survey on the possibilities & impossibilities of AI-generated text detection. Transactions on Machine Learning Research, 2023.

[12] Junchao Wu, Shu Yang, Runzhe Zhan, Yulin Yuan, Lidia Sam Chao, and Derek Fai Wong. A survey on llm-generated text detection: Necessity, methods, and future directions. Computational Linguistics, 51(1):275–338, 2025.

[13] Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In International conference on machine learning, pages 24950–24962. PMLR, 2023.

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

[15] Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Spotting LLMs with binoculars: Zero-shot detection of machine-generated text. In Forty-first International Conference on Machine Learning, 2024.

[16] Yihuai Xu, Yongwei Wang, Yifei Bi, Huangsen Cao, Zhouhan Lin, Yu Zhao, and Fei Wu. Training-free LLM-generated text detection by mining token probability sequences. In The Thirteenth International Conference on Learning Representations, 2025.

[17] Shihan Ma and Quan Wang. Tcd: Empowering zero-shot detection of llm-generated text using token contextual dependency. In Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 38911–38926, 2026.

[18] Guangsheng Bao, Yanbin Zhao, Juncai He, and Yue Zhang. Glimpse: Enabling white-box methods to use proprietary models for zero-shot LLM-generated text detection. In The Thirteenth International Conference on Learning Representations, 2025.

[19] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[20] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. Advances in neural information processing systems, 36:53728–53741, 2023.

[21] Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. Radar: Robust ai-text detection via adversarial learning. Advances in neural information processing systems, 36:15077–15095, 2023.

[22] Xun Guo, Yongxin He, Shan Zhang, Ting Zhang, Wanquan Feng, Haibin Huang, and Chongyang Ma. Detective: Detecting ai-generated text via multi-level contrastive learning. Advances in Neural Information Processing Systems, 37:88320–88347, 2024.

[23] Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. Gltr: Statistical detection and visualization of generated text. arXiv preprint arXiv:1906.04043, 2019.

[24] Xiaowei Zhu, Yubing Ren, Fang Fang, Qingfeng Tan, Shi Wang, and Yanan Cao. DNA-detectLLM: Unveiling AI-generated text via a DNA-inspired mutation-repair paradigm. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.

[25] Runheng Liu, Heyan Huang, Xingchen Xiao, and Zhijing Wu. Zero-shot detection of LLM-generated text via implicit reward model. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.

[26] Solomon Kullback and Richard A Leibler. On information and sufficiency. The annals of mathematical statistics, 22(1):79–86, 1951.

[27] Junchao Wu, Runzhe Zhan, Derek Wong, Shu Yang, Xinyi Yang, Yulin Yuan, and Lidia Chao. Detectrl: Benchmarking llm-generated text detection in real-world scenarios. Advances in Neural Information Processing Systems, 37:100369–100401, 2024.

[28] Guangsheng Bao, Lihua Rong, Yanbin Zhao, Qiji Zhou, and Yue Zhang. Decoupling content and expression: Two-dimensional detection of ai-generated text. arXiv preprint arXiv:2503.00258, 2025.

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

[30] Xianjun Yang, Wei Cheng, Yue Wu, Linda Ruth Petzold, William Yang Wang, and Haifeng Chen. DNA-GPT: Divergent n-gram analysis for training-free detection of GPT-generated text. In The Twelfth International Conference on Learning Representations, 2024.

[31] Shashi Narayan, Shay B Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1797–1807, 2018.

[32] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 889–898, 2018.

[33] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. Advances in neural information processing systems, 28, 2015.

[34] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. arXiv preprint arXiv:2305.10403, 2023.

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

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

[37] Ning Lu, Shengcai Liu, Rui He, Yew-Soon Ong, Qi Wang, and Ke Tang. Large language models can be guided to evade AI-generated text detection. Transactions on Machine Learning Research, 2024.

[38] Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. Advances in Neural Information Processing Systems, 36:27469–27500, 2023.

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

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

A Experimental Setup Supplement

A.1 Dataset Details

We evaluate on four settings from three high-quality detection benchmarks: DetectRL (*Multi-Domain*) [27], HART (*Level 1*) [28], MIRAGE (*DIG*) and MIRAGE (*SIG*) [9]. For each setting, we randomly sample 4,000 balanced examples to ensure fair and comprehensive evaluation across real-world scenarios (Table 4). We describe these benchmarks in more detail below.

Table 4: Statistics of all datasets used in our experiments. Each setting contains balanced human-written and machine-generated samples.

Dataset	Human	Machine	Total
DetectRL (*Multi-Domain*)	2,000	2,000	4,000
HART (*Level 1*)	2,000	2,000	4,000
MIRAGE (*DIG*)	2,000	2,000	4,000
MIRAGE (*SIG*)	2,000	2,000	4,000

A.1.1 DetectRL

DetectRL [27] is a comprehensive benchmark specifically designed for evaluating LLM-generated text detection methods across diverse real-world scenarios. The benchmark encompasses four domains considered high-risk for misinformation when LLMs are misused: academic abstracts from arXiv Archive (2002-2017) [2], news articles from XSum [31], creative stories from Writing Prompts [32], and social reviews from Yelp Reviews [33]. All human-written texts were released prior to ChatGPT to avoid potential contamination from LLM-generated content. Machine-generated texts were produced by four mainstream LLMs (including GPT-3.5-turbo [3], PaLM-2-bison [34], Claude-instant [4], and Llama-2-70B [35]) through interactive generation sessions with varying attack strategies.

In our experiments, we evaluate Pref-DetectGPT on two specific settings from DetectRL:

Multi-Domain evaluates detection performance across diverse content domains. The test set contains 8,032 samples (4,016 human-written and 4,016 machine-generated) spanning all four domains, with machine-generated texts produced using direct prompts from all four LLMs. From this test set, we randomly sample a balanced subset of 2,000 human-written and 2,000 machine-generated texts. This setting assesses whether detectors can maintain consistent performance across different writing styles and subject matters.

Multi-Attack assesses robustness against various adversarial attacks. This setting evaluates detector performance under four attack categories: (1) *Prompt attacks* including Few-Shot Prompting [36] and ICO Prompting [37] that guide LLMs to mimic human writing styles; (2) *Paraphrase attacks* including DIPPER paraphraser [38], back-translation via Google Translate [5], and polishing using LLMs that rewrite text while preserving meaning; (3) *Perturbation attacks* including character-level (DeepWordBug [39]), word-level (TextFooler [40]), and sentence-level (TextBugger [39]) adversarial perturbations; and (4) *Data mixing* including Multi-LLM Mixing (combining sentences from different

[2] https://www.kaggle.com/datasets/spsayakpaul/arxiv-paper-abstracts/data

[3] https://openai.com/index/chatgpt/

[4] https://www.anthropic.com/news/releasing-claude-instant-1-2

[5] https://translate.google.com/

LLMs) and LLM-Centered Mixing (replacing one-fourth of LLM-generated sentences with human-written content). For each attack, we randomly sample 1,000 balanced examples (500 human-written and 500 attacked machine-generated texts) for evaluation.

Table 5: Data attacks settings.

Attacks Types	Sub Types	Methods
Prompt Attacks	Few-Shot Prompt	Prompt
	ICO Prompt	Prompt
Paraphrase Attacks	DIPPER Paraphrase	DIPPER Paraphraser
	Polish Using LLMs	Prompt
	Back Translation	Google Translation API
Perturbation Attacks	Character-Level Perturbation	DeepWordBug
	Word-Level Perturbation	TextFooler
	Sentence-Level Perturbation	TextBugger
Data Mixing	Multi-LLMs Mixing	Sentence Mixing
	LLM-Centered Mixing	Sentence Mixing

A.1.2 MIRAGE

MIRAGE [9] is a large-scale benchmark designed to evaluate machine-generated text detection across diverse generation scenarios and source models. The benchmark addresses limitations of previous datasets by incorporating both machine-generated text (MGT) and machine-revised text (MRT), reflecting more common real-world applications where humans interact with LLMs to polish or rewrite their content. MIRAGE encompasses five distinct text domains: Academic (abstracts from NeurIPS and ArXiv), E-Mail (Enron-Emails), Website, News, and Comment. All human-written texts were collected prior to 2021 or cleaned to remove post-2021 data, ensuring no contamination from LLM-generated content. Machine-generated samples were produced by 17 LLMs, including 13 proprietary models (GPT-4o, GPT-4o-mini, GPT-o3-mini, Claude-3.5-Haiku, Claude-3.7-Sonnet, DeepSeek-V3, DeepSeek-R1, Gemini-2.0-Flash, Gemini-2.0-Flash-Lite, Grok2, Moonshot-v1, Doubao-1.5-pro-32k, QwQ-Plus) and 4 advanced open-source models (Qwen-2.5-7B, Qwen-2.5-7B-R1-Distill, Llama-3.1-8B, Llama-3.1-8B-R1-Distill), covering nearly all major LLMs used in real-world applications.

MIRAGE defines two primary evaluation scenarios:

Disjoint-Input Generation (DIG) focuses on pure machine-generated text where LLMs create content from scratch. This scenario includes three generation tasks: (1) *Generate* where LLMs produce approximately 150-word articles starting with the first 30 tokens of human-written text; (2) *Polish* where LLMs refine complete human-written texts while preserving original details and length; (3) *Rewrite* where LLMs paraphrase complete human-written texts while maintaining semantic content and length. Each LLM generates 1,000 samples distributed across domains: 300 Academic, 100 E-Mail, 200 Website, 200 News, and 200 Comment texts. The total DIG dataset contains 46,923 samples (16,412 Generate, 14,776 Polish, 15,735 Rewrite).

Shared-Input Generation (SIG) evaluates detection in scenarios where multiple LLMs process the same human-written inputs, creating a more challenging setting with shared source material. This scenario maintains the same three generation tasks and domain distribution as DIG. The total SIG dataset contains 46,915 samples (16,388 Generate, 14,776 Polish, 15,751 Rewrite).

All generation tasks incorporate style control by randomly selecting from 16 writing styles (formal, oral, academic, literary, critical, narrative, descriptive, lyric, objective, subjective, original, casual, expository, argumentative, journalistic, poetic) to enhance dataset diversity and robustness. For our evaluation, we randomly sample balanced subsets of 2,000 human-written and 2,000 machine-generated texts from each scenario.

A.1.3 HART

HART [28] is a multi-level benchmark for hybrid text detection, covering four domains (student essays, arXiv introductions, creative writing, and news) and five languages (English, Chinese, French, Spanish, Arabic). The benchmark consists of 32K samples generated by six recent LLMs:

GPT-3.5-Turbo, GPT-4o, Claude 3.5 Sonnet, Gemini 1.5 Pro, LLaMA 3.3-70B-Instruct, and Qwen 2.5-72B-Instruct. HART categorizes samples based on AI involvement into four types: human-written (Type 0), human content rephrased by AI (Type 1), AI-generated content humanized through various methods (Type 2), and fully AI-generated content (Type 3). HART defines three detection levels: Level 1 identifies whether the text involves any AI participation (Type 0 vs. Type 1/2/3), Level 2 distinguishes whether the base content is human-written or AI-generated (Type 0/1 vs. Type 2/3), and Level 3 detects whether the text is solely produced by AI (Type 0/1/2 vs. Type 3). In our experiments, we focus on Level 1 and randomly sample a balanced subset of 2,000 human-written and 2,000 machine-generated texts.

In addition, the length-robustness study in Figure 3 uses a balanced subset of 1,000 examples (500 human-written and 500 machine-generated texts) randomly sampled from HART (*Level 1*), truncated to 16 target lengths from 20 to 320 tokens in steps of 20. Tokens are counted with the Llama-3.2-1B tokenizer so that the truncation unit matches the scoring and reference models.

A.2 Detailed Model Configurations

Table 6 summarizes the model configurations for all methods in our experiments. All models are configured with a maximum input token length of 512. For training-based methods, RoBERTa-base and RoBERTa-large are used as classifiers, while IMBD and DetectAnyLLM employ Llama-3.2-1B-Instruct as the training model, and BiScope uses Llama-3.2-1B-Instruct as the scoring model with XGBoost as the classifier. For training-free single-model methods, all methods use Llama-3.2-1B-Instruct as the scoring model. For training-free multi-model methods, Fast-DetectGPT and Lastde++ use Llama-3.2-1B-Instruct as the scoring model and Llama-3.2-1B as the sampling model; Binoculars uses Llama-3.2-1B-Instruct as the performer model and Llama-3.2-1B as the observer model; DNA-DetectLLM uses Llama-3.2-1B-Instruct as the scoring model and Llama-3.2-1B as the observer model; IRM uses Llama-3.2-1B-Instruct as the policy model p_θ and Llama-3.2-1B as the reference model p_{ref}. Pref-DetectGPT uses Llama-3.2-1B-Instruct as the policy model p_θ and Llama-3.2-1B as the reference model p_{ref}, and additionally adopts Llama-3.2-1B as the sampling model q_ϕ to ensure fair comparison with our direct baseline Fast-DetectGPT.

Table 6: Model configurations for different methods.

Method	**Model Type**	**Model Used**
Training-based Methods		
RoBERTa series	Classifier	RoBERTa-base/large
IMBD	Training Model	Llama-3.2-1B-Instruct
DetectAnyLLM	Training Model	Llama-3.2-1B-Instruct
BiScope	Scoring Model	Llama-3.2-1B-Instruct
Training-free Methods (Multi-model)		
Fast-DetectGPT	Scoring Model	Llama-3.2-1B-Instruct
	Sampling Model	Llama-3.2-1B
Binoculars	Performer Model	Llama-3.2-1B-Instruct
	Observer Model	Llama-3.2-1B
Lastde++	Scoring Model	Llama-3.2-1B-Instruct
	Sampling Model	Llama-3.2-1B
DNA-DetectLLM	Scoring Model	Llama-3.2-1B-Instruct
	Observer Model	Llama-3.2-1B
IRM	Policy Model	Llama-3.2-1B-Instruct
	Reference Model	Llama-3.2-1B
Pref-DetectGPT (Ours)	Policy Model	Llama-3.2-1B-Instruct
	Reference Model	Llama-3.2-1B
	Sampling Model	Llama-3.2-1B
Training-free Methods (Single-model)		
Other methods	Scoring Model	Llama-3.2-1B-Instruct

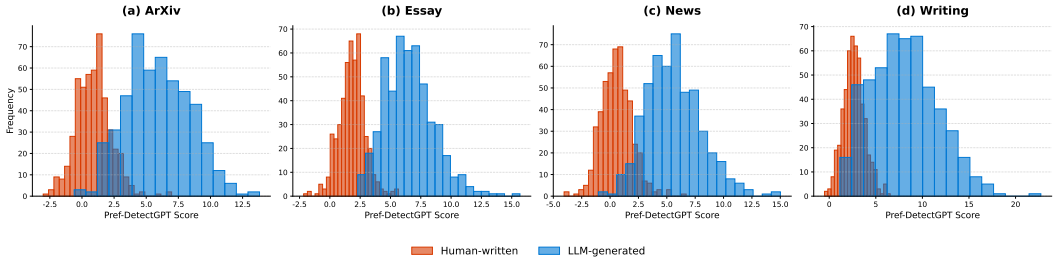

Figure 5: Score Distribution of Pref-DetectGPT on Different Domains in HART Dataset (Level 1). Histogram comparison between human-written (orange) and LLM-generated (blue) texts across four domains: (a) ArXiv, (b) Essay, (c) News, and (d) Writing.

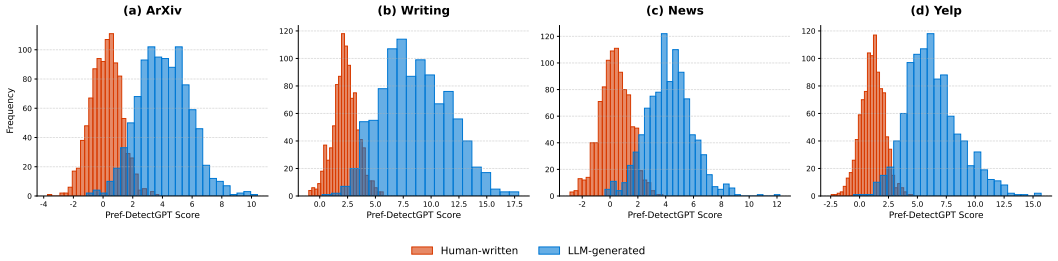

Figure 6: Score Distribution of Pref-DetectGPT on Different Domains in DetectRL Dataset (Multi-Domain). Histogram comparison between human-written (orange) and LLM-generated (blue) texts across four domains: (a) ArXiv, (b) Writing, (c) News, and (d) Yelp.

B Additional Analysis

B.1 Score Distribution Analysis

Figures 5 and 6 show Pref-DetectGPT score distributions across domains in HART (Level 1) and DetectRL (Multi-Domain) datasets. Clear separability exists between human-written (orange) and LLM-generated (blue) texts in all domains across both benchmarks.

In HART, the ArXiv domain achieves the strongest separation (human: 0-2.5, LLM: 5.0-7.5), while Essay and News show similar patterns with reduced overlap. The Writing domain exhibits broader LLM distributions (2.5-15.0), reflecting creative content variability. DetectRL demonstrates even tighter clustering with peak frequencies exceeding 100 samples across ArXiv, Writing, News, and Yelp domains. The Yelp domain shows comparable separation quality, validating effectiveness on review-style texts. Across both benchmarks and all eight domain configurations, Pref-DetectGPT maintains consistent discrimination with minimal distribution overlap, demonstrating robust performance across diverse datasets and text types.

B.2 Impact of Samping Models.

We further evaluate Pref-DetectGPT under alternative sampling models q_ϕ, applying the same substitution to Fast-DetectGPT for a fair comparison. As shown in Table 7, Pref-DetectGPT maintains consistently strong performance across all configurations, confirming that our method's effectiveness stems from preference-aware curvature measurement rather than any particular choice of q_ϕ.

B.3 Generalization Analysis.

To assess generalization capability, we evaluate cross-dataset threshold transfer: for each dataset, the optimal threshold is calibrated on that dataset and evaluated on the remaining three, with ACC

Table 7: AUROC with different sampling models q_ϕ on the HART dataset.

Sampling Model q_ϕ	Fast-DetectGPT	Pref-DetectGPT
Llama-3-8B	0.7886	**0.9431**
Llama-3.1-8B	0.7303	**0.9200**
Llama-3.2-1B	0.8118	**0.9595**
Llama-3.2-3B	0.7643	**0.9420**

Table 8: Cross-dataset threshold transfer results (ACC). The optimal threshold is calibrated on each source dataset and evaluated on the remaining three. Best results are **bolded**.

Methods	**DetectRL**	**HART**	**MIRAGE (DIG)**	**MIRAGE (SIG)**
RoBERTa-Base	0.8148	0.8073	0.7902	0.7947
RoBERTa-Large	0.8188	0.8470	0.8152	0.8156
BiScope	0.8252	0.8303	0.8067	0.8126
ImBD	0.7725	0.6797	0.7281	0.7534
DetectAnyLLM	0.9053	0.8093	0.8848	0.8847
Likelihood	0.6982	0.7330	0.7199	0.7224
LogRank	0.6944	0.7266	0.7159	0.7212
Entropy	0.6347	0.6606	0.6774	0.6619
LRR	0.6678	0.6859	0.6782	0.6793
Lastde++	0.6919	0.6987	0.6833	0.6859
Fast-DetectGPT	0.8604	0.8834	0.8408	0.8392
Binoculars	0.8537	0.8654	0.8259	0.8331
DNA-DetectLLM	0.8536	0.8672	0.8339	0.8423
IRM	0.8201	0.8995	0.8643	0.8661
Pref-DetectGPT	**0.9431**	**0.9487**	**0.9219**	**0.9177**

averaged over the target datasets. As shown in Table 8, Pref-DetectGPT exhibits strong generalization across all four transfer settings, reaching 93.29% average ACC and outperforming the strongest baselines DetectAnyLLM (87.10%) and IRM (86.25%) by a substantial margin. These results suggest that the decision boundary derived from our preference-aware curvature score remains stable across diverse datasets, highlighting Pref-DetectGPT's superior generalization ability.

C Additional Justifications

C.1 Theoretical Connection to KL Divergence

When the sampling model q_ϕ equals the reference model p_{ref}, the expected perturbed reward $\tilde{r}(y)$ reduces to:

$$\tilde{r}(y) = \sum_{i=1}^{L} \sum_{w \in \mathcal{V}} p_{\text{ref}}(w|y_{<i}) \cdot \log \frac{p_\theta(w|y_{<i})}{p_{\text{ref}}(w|y_{<i})} = -\sum_{i=1}^{L} D_{\text{KL}}\left(p_{\text{ref}}(\cdot|y_{<i}) \,\|\, p_\theta(\cdot|y_{<i})\right). \tag{23}$$

This connection to KL divergence is an emergent mathematical identity rather than the conceptual foundation of our approach. To understand whether this specific configuration affects our conclusions, we first evaluate KL divergence as a standalone detector. As shown in Table 9, it yields near-random performance across all benchmarks, suggesting that the static divergence between model pairs carries little discriminative signal for detection.

C.2 Prompt-Free Setting Justification

DPO models are trained on prompt-response pairs, yet our method evaluates text unconditionally by setting $x = \emptyset$. We argue that this is not only a valid simplification but also a deliberate design choice that improves cross-domain generalization.

Table 9: Detection performance of KL divergence as a standalone detector.

Benchmark	AUROC	AUPR	TPR5%
DetectRL	0.6117	0.6626	0.2230
HART	0.5824	0.5750	0.1030
MIRAGE (DIG)	0.5662	0.5159	0.0270
MIRAGE (SIG)	0.5646	0.5207	0.0400

If we retain prompt x throughout the derivation, the preference-aware curvature score generalizes to:

$$d(x, y, p_\theta, p_{\text{ref}}, q_\phi) = \frac{r(x, y) - \tilde{\mu}_r}{\tilde{\sigma}_r}, \tag{24}$$

where:

$$r(x, y) = \sum_{i=1}^{L} \log \frac{p_\theta(y_i|x, y_{<i})}{p_{\text{ref}}(y_i|x, y_{<i})}, \tag{25}$$

$$\tilde{\mu}_r = \sum_{i=1}^{L} \sum_{w \in \mathcal{V}} q_\phi(w|x, y_{<i}) \cdot \log \frac{p_\theta(w|x, y_{<i})}{p_{\text{ref}}(w|x, y_{<i})}, \tag{26}$$

$$\tilde{\sigma}_r^2 = \sum_{i=1}^{L} \sum_{w \in \mathcal{V}} q_\phi(w|x, y_{<i}) \cdot \left(\log \frac{p_\theta(w|x, y_{<i})}{p_{\text{ref}}(w|x, y_{<i})} - \bar{r}_i \right)^2, \tag{27}$$

$$\bar{r}_i = \sum_{w \in \mathcal{V}} q_\phi(w|x, y_{<i}) \cdot \log \frac{p_\theta(w|x, y_{<i})}{p_{\text{ref}}(w|x, y_{<i})}. \tag{28}$$

Table 10: Prompt settings used in the evaluation.

Prompt	Description
p_1	*You are an academic researcher. Write a concise scientific abstract including background, methodology, results, and conclusions.*
p_2	*You are a student. Write a college application essay with a compelling personal narrative, specific examples, and a clear reflection on your growth and values.*
p_3	*You are a journalist. Write a news article following the inverted pyramid structure with a strong lede, supporting details, and relevant context.*
p_4	*You are a creative writer. Write a short story with a well-developed character, vivid descriptive details, and a satisfying narrative arc.*
p_5	(empty string, i.e., $x = \emptyset$)

Table 11: AUROC across different prompt settings on the HART benchmark.

Prompt	ArXiv	Essay	News	Writing	Avg.
p_1	0.9673	0.9740	0.9647	0.9321	0.9595
p_2	0.9621	0.9804	0.9620	0.9316	0.9590
p_3	0.9613	0.9743	0.9659	0.9326	0.9585
p_4	0.9604	0.9821	0.9618	0.9354	0.9599
p_5 ($x = \emptyset$)	**0.9688**	**0.9886**	**0.9747**	**0.9388**	**0.9677**

This formulation shows that our method naturally accommodates prompts when available. However, in real-world detection scenarios, the original prompt used to generate the text is typically unknown. A domain-specific prompt therefore introduces an additional source of distributional noise: when applied out-of-domain, it may shift the reward signal in ways that are unrelated to whether the text is

machine-generated. Setting $x = \emptyset$ avoids this issue entirely, making the detector agnostic to both the generation prompt and the target domain.

To empirically validate this reasoning, we evaluate Pref-DetectGPT on the HART multi-domain benchmark (ArXiv, Essay, News, Writing) under five prompt settings. As detailed in Table 10, p_1–p_4 are domain-specific prompts and p_5 corresponds to the empty prompt $x = \emptyset$.

As shown in Table 11, the empty prompt consistently achieves the highest AUROC across all four domains, confirming that prompt-free detection not only sidesteps the issue of out-of-distribution prompting but also yields superior performance in cross-domain scenarios.

D Algorithm

We summarize the complete detection procedure of Pref-DetectGPT in Algorithm 1 for clarity.

Algorithm 1 Pref-DetectGPT Detection Process

Require: Passage y, Policy Model p_θ, Reference Model p_{ref}, Sampling Model q_ϕ, Decision Threshold τ

// Compute IRM of Original Text

$r(y) \leftarrow \sum_{i=1}^{L} \log \frac{p_\theta(y_i|y_{<i})}{p_{\text{ref}}(y_i|y_{<i})}$

// Compute Expected IRM of Perturbed Text

$\tilde{\mu}_r \leftarrow \sum_{i=1}^{L} \sum_{w \in \mathcal{V}} q_\phi(w|y_{<i}) \cdot \log \frac{p_\theta(w|y_{<i})}{p_{\text{ref}}(w|y_{<i})}$

// Compute Variance for Normalization

$\bar{r}_i \leftarrow \sum_{w \in \mathcal{V}} q_\phi(w|y_{<i}) \cdot \log \frac{p_\theta(w|y_{<i})}{p_{\text{ref}}(w|y_{<i})}$

$\tilde{\sigma}_r^2 \leftarrow \sum_{i=1}^{L} \sum_{w \in \mathcal{V}} q_\phi(w|y_{<i}) \cdot \left(\log \frac{p_\theta(w|y_{<i})}{p_{\text{ref}}(w|y_{<i})} - \bar{r}_i \right)^2$

// Compute Preference-Aware Curvature

$d \leftarrow \frac{r(y) - \tilde{\mu}_r}{\tilde{\sigma}_r}$

// Detection Decision

if $d > \tau$ **then**

  return `true` // Machine-Generated

else

  return `false` // Human-Written

end if

E Limitations

Like many multi-model methods, our method relies on two models to construct the implicit reward model, requiring more memory resources than single-model methods. In addition, the implicit reward is only meaningful when the policy model and reference model share the same pre-training foundation, requiring a matched base and instruction-tuned model pair, which may limit applicability when such a pair is unavailable. We leave the exploration of preference-aware curvature without requiring matched model pairs as an important direction for future work.

F Broader Impacts

This study presents Pref-DetectGPT, a detection framework that identifies machine-generated text by preference-aware curvature measurement. We emphasize that Pref-DetectGPT's predictions are probabilistic and should serve as diagnostic indicators rather than definitive verdicts. The method should not be used as a standalone arbiter in high-stakes scenarios such as academic integrity evaluations or professional assessments, where human judgment and contextual analysis must remain central. We caution against misuse of this technology for mass surveillance, unjustified censorship, or restricting freedom of expression, and advocate for responsible deployment that respects privacy, safeguards academic freedom, and avoids discriminatory application.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly describe the contributions and scope of this paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of this study in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The complete derivations of all theoretical results, including the implicit reward model construction for both original and perturbed text and the preference-aware curvature measurement, are provided in Section 4.2 and Section 4.3.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We fully disclose all information necessary to reproduce the main experimental results, including but not limited to the details provided in Section 5 and Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code and data are publicly available at the repository linked in the abstract, with instructions to reproduce the experimental results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify all the training and test details in Section 5.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars are not reported because our method is deterministic and does not involve any stochastic components, making repeated runs and statistical significance measures unnecessary.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Computational resources are described in detail in Section 5.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read the NeurIPS Code of Ethics and our research was conducted with that in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The broader impacts of our work are discussed in Appendix F.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have clearly cite all the existing assets we used.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Our proposed method Pref-DetectGPT leverages publicly available LLMs, including instruction-tuned and base models, to construct implicit reward models for machine-generated text detection. These models are integral to the core methodology and are clearly described in the main text and Appendix A.2.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.