RLSPOOFER: A SAMPLE-EFFICIENT BLACK-BOX SPOOFING STRESS-TEST FOR LLM WATERMARKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) watermarking has emerged as a promising approach for detecting and attributing AI-generated text, yet its robustness to black-box spoofing remains insufficiently evaluated. Existing evaluation methods often demand extensive datasets and white-box access to algorithmic internals, limiting their applicability. In this paper, we study watermark spoofing from a distributional perspective, viewing it as selectively reconstructing watermark-induced probability shifts under semantic constraints. We theoretically show that low-dimensional watermark-induced shifts can be learned more sample-efficiently than the full distribution, while KL-bounded reconstruction inherently favors larger watermark deficits. Guided by this analysis, we propose RLSpoofer, a reinforcement learning-based black-box spoofing attack that requires only 100 paired human–watermarked paraphrases and zero access to watermarking internals or detectors. Despite this weak supervision, it empowers a 4B model to achieve a 62.0% spoof success rate on PF-Watermark while preserving semantic fidelity, compared with below 7% for baselines trained on up to 10,000 samples. Our findings expose weaknesses in the spoofing resistance of current LLM watermarking paradigms and provide a sample-efficient framework for evaluating watermark robustness.

1 INTRODUCTION

With the rapid advancement and increasing availability of large language models (LLMs), they are being widely applied across diverse applications to generate fluent and human-like content (Yang et al., 2025; Grattafiori et al., 2024). However, this widespread use raises major concerns about model misuse, including the generation of misinformation (Chen & Shu, 2023), copyright violations (Xu et al., 2025), data contamination (Shumailov et al., 2023), and academic dishonesty (Cotton et al., 2024). As a safeguard, text watermarking has become an important defense. By subtly embedding statistical signals into model outputs, watermarking enables reliable AI text detection and source tracking while maintaining the quality of the text (Kirchenbauer et al., 2023; Zhao et al., 2024).

Most existing watermarking schemes predominantly follow a generate-and-detect paradigm, utilizing detection algorithms to identify hidden patterns that differentiate AI-generated text from human authorship (Lu et al., 2024; Lee et al., 2024). Nevertheless, this paradigm inadvertently creates an avenue for watermark spoofing. Malicious actors can query a watermarked LLM to gather text samples, using this data to construct a surrogate model capable of approximating the underlying watermarking rules (Jovanović et al., 2024). By successfully forging the watermark, such attacks pose a fundamental threat to the reliability of existing watermarking systems.

To rigorously assess the spoofing resistance of LLM watermarking schemes, researchers have investigated various attack paradigms. However, existing methods typically suffer from at least one of three major limitations: (1) reliance on partial algorithmic knowledge or detector API access (Kirchenbauer et al., 2023; Chen et al., 2024); (2) the need for large training corpora (An et al., 2026; Gu et al., 2023) (up to 10k); or (3) an inability to generate semantically coherent text that seamlessly matches the original context (Pang et al., 2024; Gloaguen et al., 2024). These limitations constrain the practical evaluation of watermark spoofing resilience, motivating a more generalizable and data-efficient methodology for benchmarking watermark robustness.

In this paper, we investigate the resilience of LLM watermarks to spoofing attacks. In the black-box setting, direct instance-level spoofing is intractable because the adversary has no access to the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

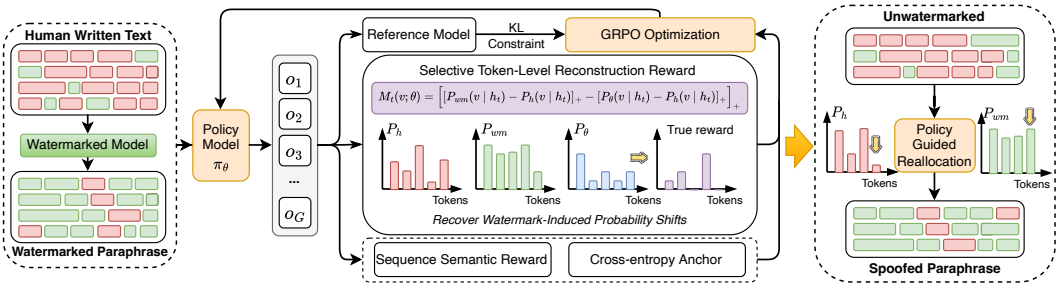


Figure 1: Overview of RLSpoofer. Given human–watermarked rewrite pairs, RLSpoofer mimics watermark injection through selective token-level reconstruction, semantic preservation, and a cross-entropy anchor for target supervision, enabling the policy to recover watermark-induced probability shifts and thereby effectively spoofing the watermark.

watermark details or the detector. Motivated by recent evidence that watermarked text exhibits systematic distributional discrepancy from human-written text, even for distortion-free watermarking schemes (Liu et al., 2024a), we take a *distributional view* of watermark spoofing. Specifically, the attacker seeks to recover watermark-induced probability shifts relative to the human-like distribution while preserving semantic fidelity.

To characterize sample-efficient spoofing, we show that modeling low-dimensional watermark-induced variations is more sample-efficient than full-distribution matching, while KL-constrained reconstruction selectively favors larger remaining deficits. Building on these results, we propose RLSpoofer, a reinforcement-learning-based black-box attack that requires no access to watermarking internals or the detector. As illustrated in Figure 1, RLSpoofer jointly optimizes a selective token-level reconstruction reward, sequence-level semantic fidelity, and a cross-entropy anchor to recover watermark-induced probability shifts. Using only **100 training pairs**, RLSpoofer enables a compact 4B model to achieve a **62.0%** spoof success rate against PF-Watermark, substantially surpassing the *below 7%* performance of baselines trained on up to *10,000 examples*. Extensive experiments across five attacker models and six watermarking schemes show that RLSpoofer provides a generalizable, sample-efficient stress test for spoofing resilience, exposing critical vulnerabilities in current watermarking defenses. Our contributions can be summarized as follows:

- We formulate spoofing as selective probability reconstruction and show that KL-bounded local updates favor continuations with larger residual watermark deficits. (Sec. 4.1.)
- We propose RLSpoofer, an effective and sample-efficient black-box attack. It successfully spoofs watermarks with high semantic fidelity, requiring only limited human-watermarked rewrite pairs and zero access to the watermarking scheme or detector. (Sec. 4.2.)
- We conduct extensive experiments across five architectures and six watermarking schemes from both logit-based and sampling-based families. The results demonstrate the effectiveness of RLSpoofer, exposing vulnerabilities in current watermark defense. (Sec. 5.)

2 Related Works

LLM watermarks. As LLMs become increasingly widespread, the human-written and machine-generated text grows harder to distinguish. By embedding imperceptible yet algorithmically detectable signals into generated text, watermarking has become important tool for content attribution, copyright protection, and the mitigation of malicious misuse (Wu et al., 2025; Liu et al., 2024b; Zhang et al., 2024). Existing methods mainly follow two paradigms. *Logit-based* approaches (Kirchenbauer et al., 2023; Lu et al., 2024) use a secret pseudo-random hash key to partition the vocabulary into *green* and *red* token lists at each generation step. They then bias the logits of green-list tokens to increase their sampling probability, and detection is performed through statistical tests for green-token overrepresentation (Jovanović et al., 2024). Conversely, *sampling-based* methods (Zhao et al., 2024; Huo et al., 2025; Hou et al., 2024) embed watermark signals directly into the sampling process, rather than modifying logits explicitly, offering finer-grained control while reducing quality degradation. Across both paradigms, recent advances prioritize minimizing statistical distortion, ensuring watermarked outputs remain indistinguishable from natural text (Kuditipudi et al., 2023).

LLM watermark spoofing attack. LLM watermark *spoofing attacks* seek to forge a target watermark signal and thereby falsely attribute arbitrary text to a specific model provider. Existing spoofing methods can generally be grouped into three categories. The first, *piggyback spoofing* (Pang et al., 2024), makes subtle edits to authentically watermarked text in order to inject toxic content while preserving detector confidence. However, this strategy substantially limits the attacker's semantic flexibility. The second, *feedback-guided spoofing* attacks (Pang et al., 2024; Zhou et al., 2024), rely on repeated interaction with the watermark detector during generation to search for text that remains watermarked. Such attacks are typically query-intensive and depend on detector access, limiting their practical applicability. The third, *learning-based spoofing* (Jovanović et al., 2024; An et al., 2026; Gu et al., 2023), queries the target model to construct a training set for a surrogate model that internalizes the watermark signal. However, these approaches typically require either the knowledge of the watermarking mechanism (Jovanović et al., 2024) or a large amount of training data (e.g., up to 10K samples (An et al., 2026)). Consequently, these restrictive assumptions and inefficiencies limit their practicality for evaluating watermark spoofing resilience.

3 PRELIMINARIES

LLM Paraphraser and Generative Text Watermark. Let $\mathcal{V}$ be a finite vocabulary and $\mathcal{V}^*$ denote the space of all finite token sequences. Given a source sequence $\mathbf{X} \in \mathcal{V}^*$, an autoregressive paraphraser π_θ induces a distribution over paraphrased sequences $\mathbf{X}' = (x'_1, \ldots, x'_{|\mathbf{X}'|})$: $P_\theta(\mathbf{X}' \mid \mathbf{X}) := \prod_{t=1}^{|\mathbf{X}'|} \pi_\theta(x'_t \mid h_t)$, where $h_t = (x'_{<t}, \mathbf{X})$. We consider generative text watermarks that introduce a key-dependent shift in the generation distribution. At decoding step t, a watermarking mechanism with secret key s transforms the base human-like distribution $P_h(\cdot \mid h_t)$ into a watermarked distribution $P_{wm}(\cdot \mid h_t)$. Given a sequence $\mathbf{X}'$, a detector f computes a key-dependent statistic $f(\mathbf{X}', s)$ and declares the sequence watermarked if $f(\mathbf{X}', s) > \delta$, where δ is a fixed detection threshold.

Threat Model: Semantics-Preserving Paraphrase-based Spoofing. We consider a black-box adversary that spoofs a watermark into an unwatermarked sequence $\mathbf{X}$ through semantics-preserving paraphrasing. We assume that the watermarked generator supports instruction-following generation and applies the same fixed secret key across queried outputs. The adversary is defined by: (1) ***Objective***: Generate $\mathbf{X}'$ that preserves the semantics of an unwatermarked input $\mathbf{X}$ while triggering the watermark detector. (2) ***Knowledge***: The adversary has no access to the secret key s, detector f, detection threshold δ, or watermark logits. (3) ***Capability***: The adversary can query the watermarked generator as a black-box API to collect a small set of paired unwatermarked–watermarked paraphrases, and subsequently fine-tune a paraphrasing model π_θ without detector access.

Adversarial Objective. Let $d : \mathcal{V}^* \times \mathcal{V}^* \to [0, \infty)$ be a distance metric where smaller values indicate higher semantic similarity. For a tolerance $\varepsilon > 0$, we define the instance-level spoof success rate as: $\mathrm{SSR}_{\mathbf{X}}(\theta) := P_\theta(\{\mathbf{X}' \in \mathcal{V}^* : d(\mathbf{X}', \mathbf{X}) < \varepsilon,\ f(\mathbf{X}', s) > \delta\} \mid \mathbf{X})$. The adversary seeks to maximize this rate. However, without access to f and s, direct optimization in a black-box setting is intractable.

Distributional Surrogate. Since the detector and secret key are inaccessible, direct optimization of watermark detection is infeasible. We instead mimic the generation behavior of the watermarked model. At each decoding step t, watermarking *selectively* redistributes probability mass among candidate continuations $v \in \mathcal{V}$, resulting in a key-dependent shift from P_h to P_{wm} (Liu et al., 2024a). Accordingly, spoofing can be viewed as reconstructing the *watermark-induced residual*, i.e., the distributional deviation from the human-like distribution P_h, via paraphraser π_θ while preserving the semantics of the original input. Specifically, we measure the remaining watermark-specific probability excess over the human-like baseline as

$$M_t(v; \theta) = \Big[[P_{wm}(v \mid h_t) - P_h(v \mid h_t)]_+ - [P_\theta(v \mid h_t) - P_h(v \mid h_t)]_+ \Big]_+ . \tag{1}$$

Imposing the semantic-fidelity constraint $P_\theta(d(X', X) < \varepsilon \mid X) \geq 1 - \rho$ for $\rho \in [0, 1)$, we formulate the spoofing objective as:

$$\min_\theta \ \mathbb{E}_{\mathbf{X}' \sim P_\theta(\cdot \mid \mathbf{X})} \left[\sum_t \sum_{v \in \mathcal{V}} M_t(v; \theta) \right] \quad \text{s.t.} \quad P_\theta \left(d(\mathbf{X}', \mathbf{X}) < \varepsilon \mid \mathbf{X} \right) \geq 1 - \rho. \tag{2}$$

Therefore, we cast spoofing as selectively recovering watermark-induced probability mass under semantic constraints. We next characterize why this residual can be learned from limited paired observations and how it should be reconstructed.

4 METHODS

4.1 SAMPLE-EFFICIENT WATERMARK RESIDUAL RECONSTRUCTION

Global distribution matching can be sample-inefficient. A natural approach to watermark spoofing is to directly distill $P_{wm}(\cdot \mid h_t)$. However, P_{wm} combines context-dependent variation inherited from P_h with watermark-specific probability shifts induced by the same mechanism and secret key. Thus, global distillation must learn both ordinary linguistic variation and the watermark signal, increasing the amount of data required to identify the watermark-specific behavior.

Paired residual reconstruction reduces unnecessary estimation. Leveraging the matched structure of human-like and watermarked behaviors, paired residual reconstruction isolates watermark-induced residuals instead of modeling the full watermarked distribution. Following Assumption 1 in Appendix A.1, we model these residuals as lying in an m-dimensional local subspace around P_h, and compare the resulting estimation problem with unrestricted matching of $P_{wm}(\cdot \mid h_t)$ under the same total-variation accuracy.

Proposition 1 (Relative Sample Complexity of Residual Reconstruction)**.** *Under Assumption 1 and standard local regularity conditions, for the same estimation accuracy, the sample complexity of paired residual reconstruction relative to unrestricted global matching is* $\Theta(m/(|V|-1))$.

Remark 1. *The proof is provided in Appendix A.1. Proposition 1 shows that when watermark-induced residual variations lie in a lower-dimensional subspace, residual reconstruction requires fewer samples than unrestricted global matching.*

From residual recovery to selective reconstruction. Proposition 1 motivates reconstructing the watermark-specific residual rather than globally matching P_{wm}. We retain $M_t(v;\theta)$ from Eq. 1 as the remaining watermark deficit. To characterize how the available semantic budget should be allocated, we analyze a local quadratic surrogate for reducing the current residual. As a local proxy for limiting semantic drift, we constrain probability updates around P_h by a KL budget. Let $z_t(v)$ denote the probability displacement at continuation v, with $\sum_v z_t(v) = 0$. Under the quadratic trust region $\frac{1}{2}\sum_{t,v} z_t(v)^2/P_h(v \mid h_t) \le C$, the local residual reconstruction reduces to maximizing $\sum_{t,v} M_t(v;\theta) z_t(v)$. We next characterize its maximum reduction and optimal allocation.

Theorem 1 (Optimal Local Residual Reconstruction)**.** *Under the KL budget above and probability conservation at each decoding step, the recoverable watermark residual satisfies*

$$\sum_t \sum_{v\in\mathcal{V}} M_t(v;\theta) z_t(v) \le \sqrt{2C \sum_t \mathrm{Var}_{P_h(\cdot|h_t)}[M_t(\cdot;\theta)]}.$$

The bound is attained, whenever the resulting redistribution remains feasible, by $z_t^*(v) = \lambda^{-1} P_h(v \mid h_t)\big(M_t(v;\theta) - \mathbb{E}_{u\sim P_h(\cdot|h_t)}[M_t(u;\theta)]\big)$*, where* $\lambda > 0$ *is determined by the KL budget.*

Remark 2. *The proof is provided in Appendix A.2. Theorem 1 establishes that local residual reconstruction is inherently selective: probability mass is preferentially allocated toward continuations whose remaining watermark deficit exceeds the local* P_h*-weighted average.*

Together, Proposition 1 and Theorem 1 show that paired residual learning reduces estimation complexity of watermark-specific behavior and enables selective reconstruction under semantic constraints. Guided by these results, we next introduce RLSpoofer, which operationalizes this sample-efficient residual reconstruction through policy optimization.

4.2 RLSPOOFER: A SAMPLE-EFFICIENT RESIDUAL RECONSTRUCTION ATTACK

From residual reconstruction to policy optimization. Our analysis shows that effective reconstruction should favor continuations with larger remaining watermark deficits. In practice, however, $M_t(v;\theta)$ changes as the paraphrasing distribution evolves, so the desired redistribution must be updated accordingly. Therefore, we propose RLSpoofer, which performs iterative policy optimization to adaptively allocate probability mass according to the current reconstruction state. Specifically, RLSpoofer adopts GRPO (Shao et al., 2024) to optimize this dynamic objective from a small set of paired examples $(\mathbf{X}, \mathbf{X}'_{wm})$, where $\mathbf{X}'_{wm}$ is a semantics-preserving watermarked rewrite of an unwatermarked input $\mathbf{X}$. These pairs provide observable evidence for constructing the watermark-specific reconstruction signal, enabling selective reallocation toward watermark-favored token alternatives.

Approximating the reconstruction signal. Since the ground-truth watermarked distribution P_{wm} and human-like distribution P_h required by $M_t(v;\theta)$ are not directly accessible, we construct tractable surrogate distributions using a shared lightweight reference model π_{ref}. Since LLMs model human text distributions (Grattafiori et al., 2024), we use $P_h(x'_t \mid h_t) \approx \pi_{\text{ref}}(x'_t \mid x'_{<t}, \mathbf{X})$. For the watermarked distribution, the paired watermarked rewrite further provides a proxy observation for watermark-induced token preferences, giving $P_{wm}(x'_t \mid h_t) \approx \pi_{\text{ref}}(x'_t \mid x'_{<t}, \mathbf{X}'_{wm})$. These surrogate distributions provide a tractable reconstruction signal for token-level optimization.

Selective residual reallocation. Theorem 1 shows that, under a limited KL budget, optimal reconstruction favors continuations whose residual watermark deficits exceed the local P_h-weighted average. Since $z_t(v)$ is a displacement between two normalized distributions, it satisfies $\sum_v z_t(v) = 0$ at each step. Therefore, maximizing the recovery objective is equivalently written as $\max_{z_t} \sum_v \big(M_t(v) - \mathbb{E}_{u\sim P_h(\cdot|h_t)}[M_t(u)]\big)\, z_t(v) = \max_{z_t} \sum_v M_t(v) z_t(v)$. We therefore use the sampled residual deficit as a token-level surrogate for selective reconstruction, $r_t = M_t(x'_t;\theta)$. This reward prioritizes watermark-favored continuations with larger remaining deficits and vanishes once the corresponding excess has been recovered.

Sequence-level semantic consistency. Token-level reinforcement alone does not guarantee sequence-level semantic preservation. For each generated paraphrase $\mathbf{X}'_i$, we therefore compute its P-SP similarity (Wieting et al., 2022) to both $\mathbf{X}$ and $\mathbf{X}'_{wm}$, take the smaller score, and apply a sigmoid transformation around the semantic threshold. Following GRPO, the resulting reward A_i is normalized within each rollout group as $\hat{A}_i = \frac{A_i - \text{mean}(\{A_i\}_{i=1}^G)}{\text{std}(\{A_i\}_{i=1}^G)}$, yielding a stabilized sequence-level semantic advantage for policy optimization.

Policy optimization. Beyond the reward signals above, the observed watermarked generations provide direct supervision toward the target distribution, consistent with prior distillation-based spoofing methods (Gu et al., 2023). We therefore incorporate a teacher-forced cross-entropy loss as an auxiliary anchor. Given a group of outputs $\mathbf{X}'_i = \{x'_1, \ldots, x'_t\}_{t=1}^{|\mathbf{X}'_i|}\}_{i=1}^G$ sampled from the previous policy $\pi_{\theta_{\text{old}}}$, RLSpoofer iteratively maximizes:

$$\mathcal{J}(\theta) \approx \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathbf{X}'_i|}\sum_{t=1}^{|\mathbf{X}'_i|}\frac{\pi_\theta(x'_{i,t} \mid \mathbf{X}, x'_{i,<t})}{\pi_{\theta_{\text{old}}}(x'_{i,t} \mid \mathbf{X}, x'_{i,<t})}\left(w_1\hat{A}_i + w_2 r_{i,t}\right)\right] - \beta D_{\text{KL}}\big[\pi_\theta \,\|\, \pi_{\text{ref}}\big] - w_3\mathcal{L}_{\text{CE}}. \quad (3)$$

where π_{ref} denotes the reference model, and w_1, w_2, w_3 control the contributions of the respective components. The KL term regularizes the policy against the reference model at the token level, while $\mathcal{L}_{\text{CE}}$ denotes the teacher-forced cross-entropy loss for predicting $\mathbf{X}'_{wm}$ conditioned on $\mathbf{X}$.

5 Experiments

5.1 Experimental Setup

We provide detailed experimental setups and configurations in Appendix B.

Victim Models and Attackers. We use Llama3.1-8B-Instruct (Grattafiori et al., 2024) as the victim model to generate watermarked paraphrases. For attackers, we consider five lightweight models of varying sizes from three well-known model families: Qwen3-0.6B, Qwen3-1.7B, Qwen3-4B (Yang et al., 2025), Qwen2.5-3B-Instruct (Qwen et al., 2025), and Llama3.2-3B-Instruct. We also include Qwen3-8B for surrogate-sensitivity analysis.

Watermarking Schemes. We evaluate six watermarking schemes drawn from both logit-based and sampling-based families. Specifically, the logit-based methods include EWD (Lu et al., 2024), SWEET (Lee et al., 2024), KGW (Kirchenbauer et al., 2023), and Unigram (Zhao et al., 2023). To capture recent advances, we additionally consider two sampling-based distortion-free methods: semantic watermark PMark (Huo et al., 2025) and the cryptographic PF-Watermark (Zhao et al., 2024). We implement PMark based on its official codebase, whereas the remaining schemes are implemented and detected using the widely adopted MarkLLM toolkit (Pan et al., 2024).

Datasets. We construct the training set from C4-RealNewslike subset (Raffel et al., 2020) and the test set from four datasets: Reddit WritingPrompts (Verma et al., 2024), LFQA (Krishna et al., 2023), and the BookReport and FakeNews subsets of MMW (Piet et al., 2025). Specifically, we use Qwen3-8B

Table 1: Spoof Success Rate (SSR, %) and P-SP scores across models and watermarking schemes. **Bold** indicates the *best* SSR for each model. Higher SSR indicates stronger spoofing performance.

		EWD		SWEET		KGW		Unigram		PF		PMark	
Models	**Methods**	**SSR**	**P-SP**	**SSR**	**P-SP**	**SSR**	**P-SP**	**SSR**	**P-SP**	**SSR**	**P-SP**	**SSR**	**P-SP**
Qwen3-0.6B	Distill	42.3	0.75	20.0	0.76	35.8	0.68	13.8	0.84	6.50	0.97	20.0	0.90
	DITTO	7.50	0.43	6.75	0.41	1.00	0.33	0.25	0.32	5.50	0.79	11.8	0.63
	DPO	0.25	0.57	0.00	0.76	1.00	0.66	0.25	0.32	2.50	0.57	6.25	0.63
	RLSpoofer	**54.3**	0.77	**51.5**	0.77	**52.0**	0.72	**51.5**	0.71	**40.3**	0.71	**29.5**	0.92
Qwen3-1.7B	Distill	43.8	0.80	26.8	0.79	44.5	0.75	19.5	0.81	7.00	0.96	20.3	0.90
	DITTO	21.5	0.53	29.0	0.56	13.8	0.52	1.50	0.36	7.50	0.86	16.5	0.68
	DPO	0.25	0.94	0.00	0.84	1.00	0.96	0.50	0.87	4.25	0.58	22.5	0.93
	RLSpoofer	**53.5**	0.76	**52.0**	0.71	**54.0**	0.72	**56.8**	0.73	**39.0**	0.73	**29.5**	0.90
Qwen3-4B	Distill	51.3	0.81	37.3	0.82	57.0	0.79	28.0	0.80	6.00	0.96	21.5	0.92
	DITTO	56.0	0.68	43.3	0.66	36.5	0.61	2.25	0.39	3.50	0.88	16.5	0.71
	DPO	0.25	0.78	0.00	0.78	1.00	0.93	0.75	0.59	5.25	0.88	17.5	0.68
	RLSpoofer	**56.5**	0.73	**52.3**	0.77	**57.3**	0.74	**57.8**	0.71	**62.0**	0.77	**36.3**	0.91
Qwen2.5-3B -Instruct	Distill	**55.5**	0.76	49.8	0.77	**60.3**	0.74	25.8	0.80	6.50	0.93	22.3	0.91
	DITTO	14.0	0.50	22.3	0.54	9.50	0.48	0.25	0.34	5.25	0.72	11.3	0.56
	DPO	0.00	0.88	0.00	0.87	1.25	0.87	2.50	0.78	6.25	0.83	24.3	0.95
	RLSpoofer	53.5	0.70	**54.5**	0.75	58.3	0.72	**60.3**	0.77	**50.3**	0.68	**30.3**	0.89
Llama3.2-3B -Instruct	Distill	53.8	0.77	45.5	0.76	**56.3**	0.75	26.0	0.77	8.75	0.93	23.3	0.89
	DITTO	19.3	0.54	24.3	0.56	14.0	0.51	1.00	0.35	6.50	0.79	18.0	0.65
	DPO	2.50	0.49	0.50	0.53	0.75	0.36	7.75	0.60	6.25	0.67	25.0	0.87
	RLSpoofer	**54.5**	0.71	54.5	0.75	**61.5**	0.73	**53.0**	0.73	**49.8**	0.85	**33.3**	0.92

to generate human-like unwatermarked texts and watermarked Llama3.1-8B-Instruct to rewrite them into semantically preserved watermarked responses. For each watermarking scheme, we use 100 unwatermarked–watermarked response pairs for training and 400 unwatermarked test samples drawn evenly from the four datasets. We standardize all training and test samples to 500 tokens.

Baselines. We compare RLSpoofer with five black-box spoofing baselines. **Distill** (Gu et al., 2023) distills the watermarked distribution from 10,000 paired rewrites. **DITTO** (An et al., 2026) further exploits the distilled distribution to reproduce watermark preferences. **DPO** (Diaa et al., 2024) formulates spoofing as preference alignment using 7,000 preference pairs. Additionally, **JSV** (Jovanović et al., 2024) and **De-Mark** (Chen et al., 2024) recover watermark-specific preferences or rules from black-box outputs, and are evaluated only on compatible logit-based schemes.

Implementation Details of RLSpoofer. RLSpoofer utilizes a reparameterized sigmoid scaling function for the semantic reward, with a threshold of 0.85 to separate positive and negative rewards, thereby encouraging higher-quality rephrasings. Training on 100 samples with a batch size of 48 and group size $G = 12$ converges efficiently, completing in approximately 1.5 hours for Qwen3-4B and 45 minutes for Qwen3-0.6B on four NVIDIA Pro 6000 GPUs. In the experiments, we use the attack model itself as the reference model to generate the surrogate watermark distribution.

Metric. We evaluate spoofing effectiveness using Spoof Success Rate (SSR), defined as the fraction of rephrased texts that are detected as watermarked while satisfying a semantic similarity threshold of P-SP (> 0.7), following prior work (Jovanović et al., 2024). We additionally report Spoof Rate (SR), which measures watermark detection success without the semantic constraint. Rephrasing quality is evaluated using perplexity and GPT-as-a-Judge (GPTS) (Cheng et al., 2025).

5.2 Main Results

In this subsection, we evaluate the spoofing resilience of different watermarking schemes across four attacks, with results shown in Table 1. Details of rephrase quality are provided in Appendix C.1.

RLSpoofer achieves strong performance across watermarks, with greater gains on distortion-free schemes. As shown in Table 1, we observe that the relative advantage of RLSpoofer depends strongly on how the watermark perturbs the generation distribution. On logit-based schemes, existing attacks already achieve non-trivial spoofing performance. For example, *Distill* reaches 42.3% SSR on EWD with Qwen3-0.6B, substantially outperforming *DITTO* (7.5%) and *DPO* (0.25%). This non-

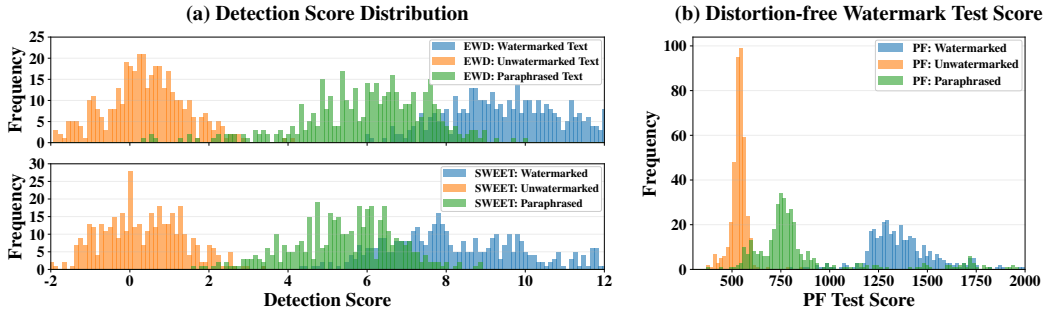


Figure 2: (a) and (b) illustrate the detection-score distributions for EWD, SWEET, and the distortion-free PF-Watermark across unwatermarked outputs, watermarked outputs, and paraphrased texts generated by Qwen3-4B trained with RLSpoofer.

trivial performance is further supported by specialized attacks designed for logit-based watermarks, such as *JSV* and *De-Mark*, which also attain competitive spoofing performance through watermark-specific reverse engineering (Table 2). We attribute this relatively strong baseline performance to the explicit distributional shifts introduced by logit-based watermarking, which can be learned through large-scale distillation or distribution matching (Gu et al., 2023). RLSpoofer nevertheless matches or surpasses these specialized attacks with only 100 paired examples, while its gains remain modest on these schemes.

Table 2: Spoof performance across methods on Qwen3-4B.

Method	EWD	KGW	Uni.
JSV	54.0	28.8	25.5
De-Mark	29.0	48.3	23.0
RLSpoofer	**56.5**	**57.3**	**57.8**

In contrast, the performance gap becomes larger on sampling-based distortion-free watermarks. On PF-Watermark, all three baselines achieve below 9% SR, while RLSpoofer reaches 82.8% SR and 62.0% SSR with Qwen3-4B (Table 3), with a similar trend on PMark. Unlike logit-based schemes, distortion-free watermarks preserve the human-like output distribution in expectation (Zhao et al., 2024), making their signals difficult to learn through global distribution distillation or preference optimization. However, a fixed watermark key still introduces sample-level biases (Liu et al., 2024a). RLSpoofer captures these local signals and therefore remains effective even when the global distribution shift is weak.

Table 3: Spoof performance on PF using Qwen3-4B.

Method	SSR	SR	P-SP	GPTS
Distill	6.00	6.50	0.96	8.68
DITTO	3.50	6.25	0.87	7.65
DPO	5.25	8.75	0.88	7.60
RLSpoofer	**62.0**	**82.8**	0.77	6.80

RLSpoofer exhibits distribution-level alignment with watermarked text. We further examine the distribution-level alignment between RLSpoofer outputs and genuine watermarked text. Using 400 test samples of 500 tokens each, we compare detector-score distributions on EWD, SWEET, and PF-Watermark. As shown in Figure 2, RLSpoofer shifts its rephrasings away from unwatermarked outputs and toward genuine watermarked text. This trend also holds for the PF-Watermark, showing that RLSpoofer captures watermark-specific patterns even when the watermark preserves the sampling distribution in expectation. Additional results are provided in Appendix C.2.

5.3 Mechanistic Analysis of RLSpoofer

In this subsection, we analyze RLSpoofer from three complementary perspectives: selective residual reconstruction, sequence-level semantic preservation, and optimization stability. Additional details can be found in Appendix C.3 - C.5.

Residual reconstruction improves spoofing effectiveness. To examine how our residual formulation affects spoofing, we compare the full M_t with three simplified reconstruction signals that progressively remove its key components using Qwen3-4B. Specifically, $P_{wm} - P_h$ captures the static signed watermark shift, $[P_{wm} - P_h]_+$ retains only watermark-favored shifts, and $P_{wm} - P_\theta$ tracks the current model discrepancy without restricting its direction. As shown in Table 4, all three alternatives underperform the full M_t. On EWD and SWEET, M_t achieves 56.5% and 52.3% SSR, respectively, while the strongest alternatives reach only 49.0% and 37.5%.

Table 4: Reward ablation (Qwen3-4B).

Signal	EWD	SWEET	PF
$P_{wm} - P_h$	35.5	28.8	18.5
$[P_{wm} - P_h]_+$	49.0	37.5	17.3
$P_{wm} - P_\theta$	32.0	24.5	24.5
M_t	**56.5**	**52.3**	**62.0**

Table 5: SSR across semantic rewards.

Model	Scheme	Min.	Avg.	Hum.	W.M.
Qwen3 (0.6B)	EWD	**54.3**	44.8	42.3	48.5
	SWEET	**51.5**	48.8	43.0	46.8
	PF	**40.3**	26.3	24.5	25.5
Qwen3 (4B)	EWD	**56.5**	47.3	47.5	45.5
	SWEET	**52.3**	34.3	47.0	48.5
	PF	**62.0**	50.3	48.5	51.0

Table 6: CE-anchor ablation (SSR vs. RLS.).

Model	Scheme	W.O.	Dis.(100)
Qwen3 (0.6B)	EWD	29.3 (-25.0)	0.25 (-54.0)
	SWEET	27.3 (-24.2)	0.25 (-51.3)
	PF	12.3 (-28.0)	0.00 (-40.3)
Qwen3 (4B)	EWD	33.5 (-23.0)	0.00 (-56.5)
	SWEET	26.8 (-25.5)	0.25 (-52.0)
	PF	25.5 (-36.5)	0.00 (-62.0)

Table 7: SSR across Training set.

		Samples		
Models	Scheme	50	100	200
Qwen3 (0.6B)	EWD	44.3	54.3	56.8
	SWEET	42.5	51.5	51.3
	PF	12.3	40.3	24.5
Qwen3 (4B)	EWD	46.5	56.5	58.5
	SWEET	43.0	52.3	54.3
	PF	20.8	62.0	25.3

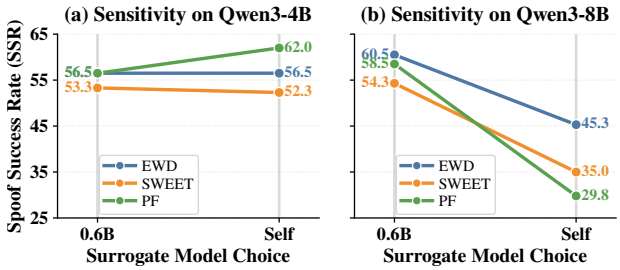


Figure 3: (a) and (b) compare SSR of RLSpoofer under two surrogate choices, Qwen3-0.6B (0.6B) and attack model (Self).

The gap becomes substantially larger on PF-Watermark, where M_t reaches 62.0%, compared with only 17.3–24.5% for the alternatives. These results show that explicitly tracking the remaining watermark-favored probability excess provides more effective guidance for model optimization than using static or signed discrepancy signals alone.

A conservative semantic reward improves semantic preservation. We design the semantic reward to avoid one-sided alignment to either the original text or its paired watermarked rewrite. Specifically, we evaluate each generated paraphrase against both references and take the lower similarity score as a conservative reward (Min.), encouraging high semantic consistency with both. To examine its effectiveness, we compare Min. with averaging the two scores (Avg.) and using either reference alone (Hum./W.M.). As shown in Table 5, we observe that Min. consistently achieves the best SSR across models and watermarking schemes, suggesting that successful spoofing benefits from conservative dual-reference semantic optimization.

Cross-entropy (CE) anchoring stabilizes optimization. We observe that, even with token-level and semantic rewards, RL optimization may drift from the observed watermarking behavior. We therefore introduce a CE anchor that provides direct supervision from the paired rewrites and stabilizes selective reconstruction during training. As shown in Table 6, we find that removing the anchor (W.O.) causes SSR drops of 23.0–36.5 points, while supervised distillation alone on the same 100 pairs (Dis.(100)) nearly fails. These results show that CE supervision serves as an effective optimization anchor but remains insufficient without selective RL.

5.4 Robustness and Sensitivity Analysis

In this subsection, we study the sensitivity of RLSpoofer to training data and surrogate model selection, together with its cross-watermark generalization and robustness to incomplete supervision.

Small training sets suffice for strong RLSpoofer performance. We examine the impact of training set size on RLSpoofer under three watermarking schemes, EWD, SWEET, and PF, using Qwen3-0.6B and Qwen3-4B. Table 7 shows that RLSpoofer can achieve strong spoofing performance with only 50 training samples. For instance, on EWD, Qwen3-4B attains an SSR of 46.5%, which increases to 58.5% with 200 samples. A similar pattern holds for SWEET. However, for the PF watermark, enlarging the training set does not yield consistent gains in SSR. This is likely because PF's distortion-free design creates a distribution gap that becomes harder to capture with more data. Overall, these findings confirm that relatively small training sets are sufficient for RLSpoofer to obtain strong spoofing performance. We provide details in Appendix C.6.

RLSpoofer exhibits increased sensitivity to surrogate selection as attacker capability grows. We evaluate how the surrogate model used to approximate the target watermarked distribution impacts RLSpoofer's efficacy. Specifically, we evaluate two surrogates, Qwen3-0.6B (0.6B) and the base attacker (Self), against EWD, SWEET, and PF watermarks across the 4B and 8B Qwen3 attackers. Fig. 3 demonstrates that surrogate selection minimally affects the weaker Qwen3-4B, which yields comparable SSR across all settings. Conversely, the stronger Qwen3-8B is highly sensitive: the 0.6B surrogate consistently outperforms the Self surrogate. For instance, on the PF watermark, utilizing 0.6B instead of Self elevates the SSR from 29.8% to 58.5%, with substantial gains similarly observed for EWD and SWEET. We attribute this disparity to Qwen3-8B's superior watermark-removal capabilities (Huang et al., 2025); its rewrites heavily weaken the embedded signal, rendering the model itself an ineffective proxy for the target watermark distribution. Details are in Appendix C.7.

RLSpoofer demonstrates strongly asymmetric and directional cross-watermark transfer. We evaluate zero-shot transferability with Qwen3-4B by training RLSpoofer on one watermarking scheme and testing it on the others. As shown in Fig. 4, transferability is not strictly constrained by watermark family. Although the KGW-style, logit-based watermarks EWD and SWEET exhibit substantial bidirectional transfer (SSR $\sim$ 47% - 50%), interactions involving the sampling-based PF watermark are notably directional. Specifically, RLSpoofer transfers effectively from SWEET to PF (SSR 48.3%) and from PF to EWD (50.5%), whereas the reverse directions are far weaker: PF to SWEET achieves only 0.25% SSR, and EWD to PF only 7.75%. This pronounced asymmetry suggests that transferability is not determined solely by mechanism-level similarity, but also by more intricate and directional overlaps in the vulnerabilities induced by different watermarking schemes. We provide details in Appendix C.8.

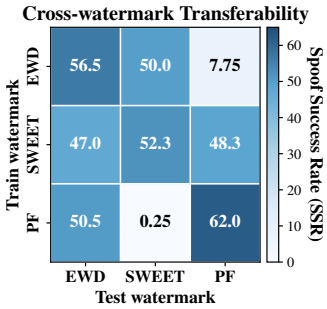


Figure 4: Cross watermark transferability on Qwen3-4B.

RLSpoofer remains learnable from incomplete watermark supervision. In practice, collected paired data may not always provide a reliable watermark signal for every target rewrite, making the effective supervision available to the attacker potentially incomplete. To model this setting, we vary the watermarked-pair ratio, where 25% indicates that only 25% of the collected pairs contain watermarked targets. As shown in Figure 5, spoofing performance decreases progressively with this ratio, yet remains non-trivial across all three schemes. At 75%, RLSpoofer still achieves 41.3%, 34.5%, and 34.8% SSR on EWD, SWEET, and PF, respectively, demonstrating that watermark-specific signals can be recovered from imperfectly collected paired data. Additional details are in Appendix C.9.

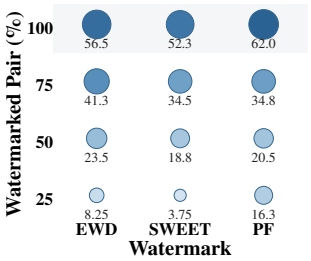


Figure 5: SSR on partially watermarked pairs (Qwen3-4B).

6 Conclusion and Broader Impact

Conclusion. In this paper, we study sample-efficient LLM watermark spoofing from a distributional perspective. We show that effective spoofing can be formulated as selectively reconstructing watermark-specific distributional shifts while preserving semantics, and develop RLSpoofer based on this principle. RLSpoofer operates under the black-box setting and requires only 100 paired examples, without access to watermark keys, detectors, or model internals. Across five attacker models and six watermarking schemes, it achieves strong spoofing performance on both logit-based and sampling-based watermarks, with particularly large gains on distortion-free schemes. These results reveal that current watermarking systems can be vulnerable even under severely limited adversarial data, highlighting sample efficiency as an important dimension of spoofing robustness.

Broader Impact. Our results show that exploitable watermark-specific behavior can be recovered from only a small number of black-box observations, suggesting that watermark security should be evaluated against limited-data adversaries rather than only large-scale extraction or detector-guided attacks. At the same time, lowering the data requirement for spoofing may reduce the practical barrier to misuse. We therefore present RLSpoofer primarily as an evaluation framework for exposing such vulnerabilities and supporting the development of more spoofing-resistant watermarking mechanisms.

AI USE STATEMENT

In this work, we used generative AI tools to generate synthetic training data as described in our experimental setting (Qwen3-8B and Llama-3.1-8B-Instruct), verify mathematical proofs, and provide feedback on experimental design. We also used generative AI tools to assist with code organization, refactoring, and routine implementation tasks; the core algorithms, methodological components, and experimental pipelines were designed and implemented by the authors. Additionally, generative AI tools were used to improve the readability of the manuscript. All AI-assisted content was independently reviewed, tested, and verified by the authors. We take full responsibility for the final content of this work, including its text, claims, code, and other artifacts.

ETHICS STATEMENT

We acknowledge the potential dual-use risks of watermark spoofing techniques. While such techniques can help identify vulnerabilities in existing watermarking systems and support the development of more reliable provenance mechanisms, they could also be misused to falsely attribute unwatermarked text to a watermarked model or undermine watermark-based attribution. Our experiments are conducted in controlled research settings with the goal of understanding and evaluating these vulnerabilities rather than facilitating misuse. To mitigate potential risks, we do not release our experimental code during the submission period and will publicly release only after acceptance with appropriate safeguards. The primary goal of this work is to provide watermark designers with a sample-efficient tool for verifying spoofing vulnerabilities and improving watermark robustness. We believe that exposing these limitations is important for developing watermarking systems that remain reliable under adversarial manipulation.

REPRODUCIBILITY STATEMENT

We provide full methodological details, training objectives, experimental settings, and evaluation protocols in the main paper and appendix, together with complete proofs and assumptions for the theoretical results. Dataset construction, preprocessing, and hyperparameters are also documented to facilitate reproduction. Due to the dual-use nature of watermark spoofing, we do not release the experimental code during the submission period; we will publicly release it only after acceptance with appropriate safeguards.

REFERENCES

Hyeseon An, Shinwoo Park, Suyeon Woo, and Yo-Sub Han. DITTO: A spoofing attack framework on watermarked LLMs via knowledge distillation. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4922–4936, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.229. URL https://aclanthology.org/2026.eacl-long.229/.

Canyu Chen and Kai Shu. Can llm-generated misinformation be detected? *arXiv preprint arXiv:2309.13788*, 2023.

Ruibo Chen, Yihan Wu, Junfeng Guo, and Heng Huang. De-mark: Watermark removal in large language models. *arXiv preprint arXiv:2410.13808*, 2024.

Yixin Cheng, Hongcheng Guo, Yangming Li, and Leonid Sigal. Revealing weaknesses in text watermarking through self-information rewrite attacks. *arXiv preprint arXiv:2505.05190*, 2025.

Debby RE Cotton, Peter A Cotton, and J Reuben Shipway. Chatting and cheating: Ensuring academic integrity in the era of chatgpt. *Innovations in education and teaching international*, 61(2):228–239, 2024.

Abdulrahman Diaa, Toluwani Aremu, and Nils Lukas. Optimizing adaptive attacks against watermarks for language models. *arXiv preprint arXiv:2410.02440*, 2024.

Thibaud Gloaguen, Nikola Jovanović, Robin Staab, and Martin Vechev. Discovering spoofing attempts on language model watermarks. *arXiv preprint arXiv:2410.02693*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. On the learnability of watermarks for language models. *arXiv preprint arXiv:2312.04469*, 2023.

Abe Hou, Jingyu Zhang, Yichen Wang, Daniel Khashabi, and Tianxing He. k-semstamp: A clustering-based semantic watermark for detection of machine-generated text. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1706–1715, 2024.

Hanbo Huang, Yiran Zhang, Hao Zheng, Xuan Gong, Yihan Li, Lin Liu, and Shiyu Liang. Rl-cracker: Exposing the vulnerability of llm watermarks with adaptive rl attacks. *arXiv preprint arXiv:2509.20924*, 2025.

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL `https://github.com/huggingface/open-r1`.

Jiahao Huo, Shuliang Liu, Bin Wang, Junyan Zhang, Yibo Yan, Aiwei Liu, Xuming Hu, and Mingxun Zhou. Pmark: Towards robust and distortion-free semantic-level watermarking with channel constraints. *arXiv preprint arXiv:2509.21057*, 2025.

Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. *arXiv preprint arXiv:2402.19361*, 2024.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International Conference on Machine Learning*, pp. 17061–17084. PMLR, 2023.

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.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*, 2023.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. Who wrote this code? watermarking for code generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4890–4911, 2024.

Aiwei Liu, Sheng Guan, Yiming Liu, Leyi Pan, Yifei Zhang, Liancheng Fang, Lijie Wen, Philip S Yu, and Xuming Hu. Can watermarked llms be identified by users via crafted prompts? *arXiv preprint arXiv:2410.03168*, 2024a.

Aiwei Liu, Qiang Sheng, and Xuming Hu. Preventing and detecting misinformation generated by large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3001–3004, 2024b.

Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. An entropy-based text watermarking detection method. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11724–11735, 2024.

Leyi Pan, Aiwei Liu, Zhiwei He, Zitian Gao, Xuandong Zhao, Yijian Lu, Binglin Zhou, Shuliang Liu, Xuming Hu, Lijie Wen, et al. Markllm: An open-source toolkit for llm watermarking. *arXiv preprint arXiv:2405.10051*, 2024.

Qi Pang, Shengyuan Hu, Wenting Zheng, and Virginia Smith. Attacking llm watermarks by exploiting their strengths. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*, 2024.

Julien Piet, Chawin Sitawarin, Vivian Fang, Norman Mu, and David Wagner. Markmywords: Analyzing and evaluating language model watermarks. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 68–91. IEEE, 2025.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arXiv:2305.17493*, 2023.

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

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL https://github.com/huggingface/trl.

John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-Kirkpatrick. Paraphrastic representations at scale. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 379–388, 2022.

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.

Zhenhua Xu, Xubin Yue, Zhebo Wang, Qichen Liu, Xixiang Zhao, Jingxuan Zhang, Wenjun Zeng, Wengpeng Xing, Dezhang Kong, Changting Lin, et al. Copyright protection for large language models: A survey of methods, challenges, and trends. *arXiv preprint arXiv:2508.11548*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar. {REMARK-LLM}: A robust and efficient watermarking framework for generative large language models. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 1813–1830, 2024.

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for ai-generated text. *arXiv preprint arXiv:2306.17439*, 2023.

Xuandong Zhao, Lei Li, and Yu-Xiang Wang. Permute-and-flip: An optimally stable and watermarkable decoder for llms. *arXiv preprint arXiv:2402.05864*, 2024.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 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)*, pp. 400–410, 2024.

Tong Zhou, Xuandong Zhao, Xiaolin Xu, and Shaolei Ren. Bileve: Securing text provenance in large language models against spoofing with bi-level signature. *Advances in Neural Information Processing Systems*, 37:56054–56075, 2024.

A Theoretical Analysis of the Distributional Surrogate

This section provides the theoretical justification for the distributional surrogate introduced in Section 4.1. We first formalize the local structure under which paired residual reconstruction reduces estimation complexity, and then prove the optimal allocation of a KL-bounded residual update.

A.1 Proof of Proposition 1

Local residual structure. The sample-complexity advantage of residual reconstruction does not require the watermark to modify only a sparse subset of vocabulary coordinates. Instead, it depends on the intrinsic dimensionality of the watermark-induced variation around the human-like distribution.

Assumption 1 (Local Residual Structure)**.** *Fix a decoding context h_t. In a neighborhood of $P_h(\cdot \mid h_t)$, suppose the family of watermark-induced distributions admits a local parameterization $P_\alpha(\cdot \mid h_t) = P_h(\cdot \mid h_t) + B\alpha + r(\alpha)$, where $\alpha \in \mathbb{R}^m$, $B \in \mathbb{R}^{|V|\times m}$ has rank m, $\mathbf{1}^\top B = 0$, and $\|r(\alpha)\|_1 = o(\|\alpha\|_2)$. The corresponding local residual subspace is therefore $\mathcal{S} = \mathrm{span}(B)$, with $m \le |V| - 1$.*

Rationale for Assumption 1. The assumption captures structured rather than coordinate-sparse watermark variation. Under our threat model, paired observations are generated using the same watermarking mechanism and fixed secret key, so their induced deviations can share systematic structure relative to the matched human-like distribution.

Assumption 1 does not require only m vocabulary coordinates to change. The residual may be dense over V, while its local variation is still governed by only m independent directions. Thus, m reflects the intrinsic dimension of the residual family rather than the number of modified coordinates.

The assumption is local and conditional: Proposition 1 characterizes when residual reconstruction is more sample-efficient than unrestricted global matching. When $m \ll |V| - 1$, the advantage is substantial; as m approaches $|V| - 1$, it vanishes. We additionally assume standard local regularity and identifiability so that these m degrees of freedom are estimable at the usual parametric rate.

We now restate Proposition 1 for completeness.

Proposition 1 (Relative Sample Complexity of Residual Reconstruction)**.** *Under Assumption 1 and standard local regularity conditions, for the same estimation accuracy, the sample complexity of paired residual reconstruction relative to unrestricted global matching is $\Theta\Big(\frac{m}{|V|-1}\Big)$.*

Proof. Fix a decoding context h_t. Since all probability distributions are normalized, any infinitesimal probability displacement lies in the probability-conserving space $\{z \in \mathbb{R}^{|V|} : \sum_{v\in V} z(v) = 0\}$, whose dimension is $|V| - 1$.

By Assumption 1, the watermark-induced residual family is locally parameterized by $\alpha \in \mathbb{R}^m$ as $P_\alpha = P_h + B\alpha + r(\alpha)$, with $\mathrm{rank}(B) = m$ and $\|r(\alpha)\|_1 = o(\|\alpha\|_2)$. Thus, although the resulting probability displacement may be nonzero across many vocabulary coordinates, its local variation is governed by only m independent degrees of freedom.

Consider n independent valid paired observations from the locally identifiable paired experiment G_α. Under the stated regularity conditions, the m-dimensional parameter α is estimable at the standard local parametric rate. In particular, the local minimax parameter error scales as $\Theta(\sqrt{m/n})$. Since total variation between nearby members of the residual family is locally equivalent to the parameter norm, the corresponding minimax total-variation risk also satisfies $\Theta(\sqrt{m/n})$. Therefore, achieving total-variation accuracy η requires $n_{\mathrm{res}} = \Theta(m/\eta^2)$.

By contrast, unrestricted global matching treats $P_{wm}(\cdot \mid h_t)$ as an arbitrary distribution in the probability simplex over V. This simplex has $|V| - 1$ independent degrees of freedom, and the standard minimax total-variation risk for unrestricted categorical distribution estimation scales as $\Theta(\sqrt{(|V| - 1)/n})$. Hence, achieving the same accuracy η requires $n_{\mathrm{global}} = \Theta((|V| - 1)/\eta^2)$.

Taking the ratio cancels the common accuracy term,

$$\frac{n_{\mathrm{res}}}{n_{\mathrm{global}}} = \Theta\Big(\frac{m}{|V| - 1}\Big),$$

which proves Proposition 1. □

Remark 3. *Proposition 1 concerns intrinsic residual dimension rather than coordinate sparsity. A watermark may therefore induce dense probability changes while still admitting a sample-complexity advantage when those changes are confined to a lower-dimensional local family.*

A.2 Proof of Theorem 1

We further restate Theorem 1 for completeness before presenting its proof.

Theorem 1 (Optimal Local Residual Reconstruction)**.** *Under a total second-order KL budget C and probability conservation at each decoding step, the recoverable watermark signal satisfies*

$$\sum_t \sum_{v\in\mathcal{V}} M_t(v) z_t(v) \le \sqrt{2C \sum_t \mathrm{Var}_{P_h(\cdot|h_t)}[M_t]}.$$

The bound is attained, whenever the resulting redistribution remains feasible, by $z_t^*(v) = \lambda^{-1} P_h(v \mid h_t)\big(M_t(v) - \mathbb{E}_{u\sim P_h(\cdot|h_t)}[M_t(u)]\big)$, where $\lambda > 0$ is determined by the total KL budget.

Proof. For each decoding step, let $\bar{M}_t := \mathbb{E}_{u\sim P_h(\cdot|h_t)}[M_t(u)]$. Since probability is conserved, $\sum_v z_t(v) = 0$, and therefore

$$\sum_{t,v} M_t(v) z_t(v) = \sum_{t,v} \big(M_t(v) - \bar{M}_t\big) z_t(v).$$

Applying the weighted Cauchy–Schwarz inequality gives

$$\sum_{t,v} \big(M_t(v) - \bar{M}_t\big) z_t(v) \le \left(\sum_{t,v} P_h(v \mid h_t)\big(M_t(v) - \bar{M}_t\big)^2\right)^{1/2} \left(\sum_{t,v} \frac{z_t(v)^2}{P_h(v \mid h_t)}\right)^{1/2}.$$

The first factor is $\big(\sum_t \mathrm{Var}_{P_h(\cdot|h_t)}[M_t]\big)^{1/2}$, while the second-order KL budget implies $\sum_{t,v} z_t(v)^2 / P_h(v \mid h_t) \le 2C$. Combining them yields

$$\sum_{t,v} M_t(v) z_t(v) \le \sqrt{2C \sum_t \mathrm{Var}_{P_h(\cdot|h_t)}[M_t]},$$

which proves the upper bound.

We next characterize the equality condition. Equality in Cauchy–Schwarz holds when the two weighted vectors are proportional, i.e.,

$$\frac{z_t^*(v)}{\sqrt{P_h(v \mid h_t)}} = \frac{1}{\lambda}\sqrt{P_h(v \mid h_t)}\big(M_t(v) - \bar{M}_t\big)$$

for some $\lambda > 0$. Equivalently,

$$z_t^*(v) = \frac{1}{\lambda} P_h(v \mid h_t)\big(M_t(v) - \bar{M}_t\big).$$

This displacement automatically preserves probability mass because

$$\sum_v z_t^*(v) = \frac{1}{\lambda}\left(\sum_v P_h(v \mid h_t) M_t(v) - \bar{M}_t \sum_v P_h(v \mid h_t)\right) = 0.$$

Finally, λ is determined by the active KL budget. Substituting $z_t^*(v)$ gives

$$2C = \frac{1}{\lambda^2} \sum_t \mathrm{Var}_{P_h(\cdot|h_t)}[M_t],$$

and hence

$$\lambda = \sqrt{\frac{\sum_t \mathrm{Var}_{P_h(\cdot|h_t)}[M_t]}{2C}}.$$

Since $\bar{M}_t = \mathbb{E}_{u\sim P_h(\cdot|h_t)}[M_t(u)]$, the maximizing displacement is exactly

$$z_t^*(v) = \lambda^{-1} P_h(v \mid h_t)\left(M_t(v) - \mathbb{E}_{u\sim P_h(\cdot|h_t)}[M_t(u)]\right),$$

which attains the stated bound and completes the proof. □

B EXPERIMENTAL SETUP AND CONFIGURATION

B.1 WATERMARK ALGORITHM SETTING

In this subsection, we report the hyperparameter settings for the watermarking algorithms evaluated in Section 5. For consistency and reproducibility, we use the official PMark Huo et al. (2025) codebase[1] with its default settings for both watermark generation and evaluation. For all other watermarking methods, we use the MarkLLM toolkit (Pan et al., 2024)[2] to generate watermarked text. Specifically, for KGW, we follow An et al. (2026) and set $\delta = 3$, while for EWD, SWEET, PF, and Unigram, we use the default configurations provided by MarkLLM. MarkLLM is widely adopted in the watermarking literature due to its robustness and ease of integration.

Hyperparameters for the PMark watermark

"algorithm_name": "PMark",
"num_samples": 64,
"pivot": rand,
"median_method": prior,
"msig": 2

Hyperparameters for the KGW watermark

"algorithm_name": "KGW",
"gamma": 0.5,
"delta": 3.0,
"hash_key": 15485863,
"prefix_length": 1,
"z_threshold": 4.0,
"f_scheme": "time",
"window_scheme": "left"

Hyperparameters for the EWD watermark

"algorithm_name": "EWD",
"gamma": 0.5,
"delta": 2.0,
"hash_key": 15485863,
"prefix_length": 1,
"z_threshold": 4.0

Hyperparameters for the SWEET watermark

"algorithm_name": "SWEET",
"gamma": 0.5,
"delta": 2.0,
"hash_key": 15485863,
"z_threshold": 4.0,
"prefix_length": 1,
"entropy_threshold": 0.9

[1] https://github.com/PMark-repo/PMark
[2] https://github.com/THU-BPM/MarkLLM

Hyperparameters for the PF watermark

"algorithm_name": "PF",
"ngram": 8,
"seed": 0,
"seeding": "hash",
"salt_key": 35317,
"payload": 0,
"max_seq_len": 8192

Hyperparameters for the Unigram watermark

"algorithm_name": "Unigram",
"gamma": 0.5,
"delta": 2.0,
"hash_key": 15485863,
"z_threshold": 4.0

B.2 Dataset Construction.

We construct the training set from the C4-RealNewslike subset Raffel et al. (2020) and the test set from Reddit WritingPrompts (Verma et al., 2024), LFQA (Krishna et al., 2023), and the BookReport and FakeNews subsets of MMW (Piet et al., 2025), following Jovanović et al. (2024). To build training data, we first prompt Qwen3-8B to generate 500-token human-like unwatermarked texts. We then prompt watermarked Llama3.1-8B-Instruct to rewrite each text while preserving its semantics, yielding paired data of the form (human-like text, watermarked rewrite). For RLSpoofer, we use 100 such pairs for training and 20 for validation on each watermark.

For DPO, we additionally obtain a rejected response by prompting Qwen3-8B to produce a non-watermarked rewrite of the same human-written text. The resulting preference pair consists of the watermarked rewrite as the chosen response and the non-watermarked rewrite as the rejected response. For distillation, we cast the task as supervised rewriting: the human-like text is formatted with the prompt template shown below, following Huang et al. (2025), and the corresponding watermarked rewrite is used as the target output for SFT.

Prompt template used

######Target Text: [human-written text]
######Instruction: Rewrite the target text above using different words but keeping the same meaning and similar length.
######Your Response:

For evaluation, we sample 400 unwatermarked test instances evenly from the four benchmark datasets. All training and test samples are standardized to 500 tokens.

B.3 Baselines.

Spoof Methods. Under the black-box threat model, in addition to RLSpoofer, we consider three spoofing baselines to further expose vulnerabilities in watermarking schemes. **Distill** Gu et al. (2023) learns to imitate the rewriting distribution of the watermarked model via supervised fine-tuning on human–watermarked rewritten pairs. **DITTO** An et al. (2026) further exploits the distilled model by analyzing its output distribution and reproducing the statistical preferences induced by the target watermark. **DPO** Diaa et al. (2024) formulates watermark spoofing as a preference alignment problem, where the attacker is optimized to prefer watermarked rewrites over non-watermarked ones.

Distill. We implement Distill using LLaMA-Factory[3] Zheng et al. (2024) and fine-tune each attacker model on 10,000 human–watermarked rewritten pairs. The task is formulated as a supervised rewriting

[3] https://github.com/hiyouga/LLaMA-Factory

problem: given an input human-written text wrapped with the prompt template shown above, the model is trained to generate the corresponding watermarked rewrite. We perform full-parameter fine-tuning for each model, using a learning rate of 2×10^{-5} and a batch size of 128.

DITTO. We implement DITTO using its official codebase [4] and follow its default settings. In particular, DITTO is applied on top of the model obtained from the Distill stage, rather than being trained from scratch. That is, it starts from the distilled spoofing model and further enhances spoofing performance by matching the statistical preferences induced by the target watermark.

DPO. We also implement DPO using LLaMA-Factory. Following prior work, we construct 7,000 preference pairs, where the chosen response is the watermarked rewrite and the rejected response is a non-watermarked rewrite of the same human-written text. We perform full-parameter fine-tuning for each model with $\beta = 0.1$, a batch size of 64, and a learning rate of 5×10^{-6}. We additionally tune the learning rate over $\{2 \times 10^{-5},\ 5 \times 10^{-6}, 2 \times 10^{-6},\ 2 \times 10^{-7}\}$ and find that 5×10^{-6} yields the best overall performance.

De-mark. We adapt the official De-mark implementation[5] Chen et al. (2024) for watermark spoofing. Using Qwen3-4B, we generate up to 600 tokens and retain the top-20 candidates at each decoding step. Candidates are re-tokenized with the tokenizer of the watermarked Llama-3.1-8B-Instruct model, and those mapping to a single token are adjusted using the green list and watermark bias recovered by De-mark before multinomial sampling.

JSV. We adapt the official watermark-stealing implementation[6] Jovanović et al. (2024) for watermark spoofing. For each compatible watermarking scheme, the attacker learns from 10,000 watermarked responses generated by Llama-3.1-8B-Instruct. We follow the restrictive $(D0, B0)$ setting, where the detector is fully private ($D0$) and the attacker has no access to unwatermarked responses from the target model ($B0$), interacting with it only through the watermarked API. Accordingly, Qwen3-4B generates base responses to the same prompts, and both response sets are re-tokenized with its tokenizer to estimate context-dependent watermark scores. These scores adjust Qwen3-4B logits during decoding. The attacker has no access to the secret key, true green lists, or detector outputs; the detector is used only for evaluation.

Evaluation Details. For evaluation, all spoofing methods use greedy decoding with temperature $= 0$. For Distill and DPO, we conduct inference with the vLLM engine to accelerate generation. For DITTO, we use the inference pipeline provided in its official implementation rather than vLLM. In all cases, the spoofing model takes the original input text and produces a rewritten response of similar semantics and length, which is then evaluated by the corresponding watermark detector. All reported spoofing results are obtained under this deterministic decoding setting.

B.4 Implementation Details of RLSpoofer.

RLSpoofer is implemented on top of the GRPO (Shao et al., 2024) module in the TRL von Werra et al. (2020) library. The training data consists only of human–watermarked rewrite pairs $(\mathbf{X}, \mathbf{X}'_{wm})$, where $\mathbf{X}$ is a human-written text and $\mathbf{X}'_{wm}$ is its semantics-preserving rewrite produced by the target watermarked model. Given $\mathbf{X}$, the attack policy π_θ samples a group of G candidate rewrites $\{x'_1, \ldots, x'_G\}$, and is optimized by jointly combining a sequence-level semantic reward, a token-level residual reconstruction reward, a token-wise KL regularizer, and a teacher-forced cross-entropy anchor. Next, we introduce the implementation details of each of the components in RLSpoofer.

Reward Components. We incorporate two reward signals to encourage semantic preservation and effective watermark spoofing:

- **Semantic Reward.** The semantic reward $A_i \in [-1, 1]$ is computed based on the P-SP score (Wieting et al., 2022) between the generated output and both the original watermarked response $\mathbf{X}'_{wm}$ and the original human-written text $\mathbf{X}$. Specifically, we first compute the semantic similarity between the generated output and each reference, and then take the *minimum* of the two scores as the final semantic score. To enhance gradient flow and emphasize semantic fidelity, we apply a sigmoid-based scaling with a threshold of 0.85 following Huang et al. (2025). This maps semantic

[4] https://github.com/hsannn/ditto

[5] https://github.com/RayRuiboChen/De-mark

[6] https://github.com/eth-sri/watermark-stealing

scores in the range $[0.7, 1.0]$ to reward values between -1 and 1. Specifically,

$$A_i = \frac{2}{1 + e^{-x}} - 1, \qquad \text{where } x = \log\left(\frac{0.975}{0.025}\right) \cdot \frac{\text{P-SP score} - 0.85}{1 - 0.85}.$$

The advantage term is normalized as $\hat{A}_i = \frac{A_i - \text{mean}(\{A_i\}_{i=1}^G)}{\text{std}(\{A_i\}_{i=1}^G) + 10^{-6}}$.

- **Token-level Residual Reconstruction Reward.** We instantiate the surrogate human-like and watermarked distributions using the same reference model π_{ref}, conditioned respectively on the human-written text $\mathbf{X}$ and its watermarked rewrite $\mathbf{X}'_{wm}$ under the same rewriting template. For each sampled token $x'_{i,t}$, we compute $p_h(x'_{i,t}) = \pi_{\text{ref}}(x'_{i,t} \mid \mathbf{X}, x'_{i,<t}), p_{wm}(x'_{i,t}) = \pi_{\text{ref}}(x'_{i,t} \mid \mathbf{X}'_{wm}, x'_{i,<t})$, together with the rollout-policy probability $p_{\text{old}}(x'_{i,t}) = \pi_{\text{old}}(x'_{i,t} \mid \mathbf{X}, x'_{i,<t})$. The token-level reward is defined as the remaining watermark reconstruction deficit: $r_{i,t} = \left[p_{wm}(x'_{i,t}) - \max\left\{p_h(x'_{i,t}), p_{\text{old}}(x'_{i,t})\right\}\right]_+$. Tokens not positively favored by the watermark receive no reward, while watermark-favored tokens are reinforced only for the residual probability excess that has not yet been reconstructed by the current policy.

From residual reconstruction to GRPO credit assignment. Our formulation assigns a token-level reconstruction deficit M_t to each sampled continuation, providing a local signal for selectively recovering watermark-specific probability shifts. In implementation, we use this deficit directly as point-wise reward shaping within GRPO rather than accumulating it into a sequence-level return-to-go. Specifically, each sampled token is rewarded only for the remaining watermark-favored probability mass that has not already been reconstructed by the current policy. This local credit assignment directly matches the selective reconstruction objective while avoiding unnecessary coupling across generation steps, which is particularly desirable in the low-data black-box setting. Thus, the objective below should be interpreted as an implemented GRPO surrogate for token-level residual reconstruction, rather than as an unbiased policy-gradient estimator of a full sequence-level return.

Cross-entropy anchor. To stabilize optimization, we further introduce a **cross-entropy anchor**. Specifically, we condition the policy on the human-written input $\mathbf{X}$ and teacher-force it to predict the watermarked rewrite $\mathbf{X}'_{wm}$, yielding $\mathcal{L}_{\text{CE}} = -\frac{1}{|\mathbf{X}'_{wm}|}\sum_t \log \pi_\theta(x_t^{wm} \mid \mathbf{X}, x_{<t}^{wm})$. This term anchors the policy toward the target watermarked rewriting distribution and prevents unstable distributional drift during RL training.

Putting these components together, RLSpoofer iteratively maximizes:

$$\mathcal{J}(\theta) \approx \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathbf{X}'_i|}\sum_{t=1}^{|\mathbf{X}'_i|}\frac{\pi_\theta(x'_{i,t} \mid \mathbf{X}, x'_{i,<t})}{\pi_{\theta_{\text{old}}}(x'_{i,t} \mid \mathbf{X}, x'_{i,<t})}\left(w_1\hat{A}_i + w_2 r_{i,t}\right)\right] - \beta D_{\text{KL}}\left[\pi_\theta \,\|\, \pi_{\text{ref}}\right] - w_3\mathcal{L}_{\text{CE}}. \quad (4)$$

In implementation, the KL regularizer is directly adopted from TRL-GRPO, and we set $\beta = 0.04$ following the OpenR1 Hugging Face (2025) setting. Unless otherwise specified, the reference model π_{ref} is chosen as the initial policy model before RL training.

Hyperparameters. For each watermarking scheme, we train a dedicated RLSpoofer model. The training set consists of 100 human–watermarked rewrite pairs, each standardized to 500 tokens. In addition, we use a validation set of 20 data pairs to select the hyperparameters w_1, w_2, and w_3. We train the model for 10 epochs with a batch size of 48 and a group size of $G = 12$, using a cosine learning rate scheduler and a rollout temperature of 0.7. Empirically, we find that a relatively large learning rate is important for inducing a sufficient distribution shift. Specifically, we use a learning rate of 2×10^{-4} for all the watermarking schemes. Training takes approximately 1.5 hours for Qwen3-4B and 30 minutes for Qwen3-0.6B on four NVIDIA Pro 6000 Blackwell GPUs. In our experiments, we set $(w_1, w_2, w_3) = (1, 1, 1)$ as the default configuration and find that it provides strong performance across watermarks. The complete hyperparameter settings are reported in Table 8.

C Experimental Results and Analysis

C.1 Robustness of watermarking schemes against spoofing

We report the full spoofing results across six watermarking schemes, five attacker models, and four spoofing methods in Tables 9, 10, and 11. We evaluate all methods using Spoof Success Rate (SSR),

Table 8: Hyperparameter configurations of RLSpoofer across watermarking schemes.

Model	Hyper.	EWD	SWEET	KGW	Unigram	PF	PMark
Qwen3-0.6B	w_1	1	1	1	2	1	1
	w_2	1	1	1	1	5	1
	w_3	0.5	1	2	1	2	1
Qwen3-1.7B	w_1	1	1	1	1	1	1
	w_2	1	1	2	0.5	0.8	1
	w_3	1	1	1	1	1	1
Qwen3-4B	w_1	1	1	1	1	1	1
	w_2	1	1	1	1	1	1
	w_3	1	1	0.5	0.5	1	1
Qwen2.5-3B-Instruct	w_1	1	1	1	1	1	1
	w_2	1	1	1	1	0.8	1
	w_3	1	1	1	0.5	1	1
Llama3.2-3B-Instruct	w_1	1	1	1	1	1	1
	w_2	1	1	1	1	1	1
	w_3	1	1	0.5	0.5	1	1

Spoof Rate (SR), P-SP, perplexity (PPL), and GPT-score (GPTS). We treat SSR as the primary metric, since it measures the proportion of rewritten outputs that are both detected as watermarked and satisfy the semantic constraint P-SP > 0.7, whereas SR measures detector success without semantic filtering. For GPTS, we use GPT-5.4-mini as the judge model with the prompt template 1, following Huang et al. (2025).

Prompt Template 1: GPT-5.4-mini as a Judge

role: system
content: You are an impartial evaluator. You will be provided with an original text and a paraphrased version of that text. Your task is to assess the quality of the paraphrased text based on the following criteria:
1. The degree to which the paraphrased text maintains the meaning of the original text.
2. The fluency, coherence, and clarity of the paraphrased text.
3. The extent to which the style and tone of the paraphrased text match those of the original text.
Please provide an objective evaluation and rate the paraphrased text on a scale of 1 to 10:
- A rating of 1 indicates that the paraphrased text completely alters the meaning of the original text.
- A rating of 10 indicates that the paraphrased text preserves the exact meaning of the original text, maintaining fluency, clarity, and style.
Your response should return the score only.

role: user
content: You will receive an original text and a paraphrased text. Please act as an impartial judge and evaluate the quality of the paraphrased text. You should evaluate the paraphrased text based on the following criteria:
1. How closely the meaning of the paraphrased text aligns with the original text.
2. The fluency and clarity of the paraphrased text.
3. How closely the style and tone of the paraphrased text align with the original text.
Rate the paraphrased text on a scale of 1 to 10, where: - A rating of 1 indicates that the paraphrased text deviates significantly from the original meaning.
- A rating of 10 indicates that the paraphrased text perfectly preserves the original meaning, fluency, clarity, and tone.
Your response should strictly follow this format and return the score only: [Rating].
Here is the original text: [human-written text]
Here is the paraphrased text: [rephrased text]

As shown in Tables 9, 10, and 11, RLSpoofer achieves the best or second-best SSR in most settings while maintaining competitive rephrasing quality. In contrast, DITTO often attains high SR but much lower P-SP, suggesting that many successful detections come from semantically distorted rewrites. DPO shows the opposite pattern: it generally preserves semantics better but fails to induce sufficient distributional shift toward the target watermark, resulting in consistently weak SSR. Distill

Table 9: Spoof Success Rate (SSR, %), Spoof Rate (SR, %), P-SP, Perplexity (PPL), and GPT-score (GPTS) across models. **Bold** indicates the best SSR for each model under each watermarking scheme. Higher SSR indicates stronger spoofing performance.

		EWD					SWEET				
Models	**Methods**	**SSR**	**SR**	**P-SP**	**PPL**	**GPTS**	**SSR**	**SR**	**P-SP**	**PPL**	**GPTS**
Qwen3-0.6B	Distill	42.3	57.3	0.75	4.50	6.62	20.0	33.8	0.76	4.68	6.75
	DITTO	7.50	97.3	0.43	4.78	3.15	6.75	95.8	0.41	4.52	3.03
	DPO	0.25	0.75	0.57	1.95	4.10	0.00	0.00	0.76	2.71	4.42
	RLSpoofer	**54.3**	77.0	0.77	5.10	6.12	**51.5**	66.3	0.77	4.90	5.89
Qwen3-1.7B	Distill	43.8	53.8	0.80	4.59	7.38	26.8	37.8	0.79	4.63	7.42
	DITTO	21.5	96.0	0.53	5.26	4.53	29.0	92.8	0.56	5.22	4.70
	DPO	0.25	0.25	0.94	2.35	8.34	0.00	0.00	0.84	2.46	7.53
	RLSpoofer	**53.5**	68.5	0.76	5.15	6.50	**52.0**	72.0	0.71	5.12	6.25
Qwen3-4B	Distill	51.3	62.5	0.81	4.75	7.14	37.3	43.8	0.82	4.79	7.45
	DITTO	56.0	94.3	0.68	5.62	6.53	43.3	94.5	0.66	5.25	6.30
	DPO	0.25	0.50	0.78	2.24	7.54	0.00	0.00	0.78	2.02	7.50
	RLSpoofer	**56.5**	87.0	0.73	4.88	6.68	**52.3**	64.3	0.77	5.38	6.87
Qwen2.5-3B -Instruct	Distill	**55.5**	81.5	0.76	4.40	7.31	49.8	67.5	0.77	4.37	7.40
	DITTO	14.0	99.5	0.50	4.95	5.32	22.3	99.8	0.54	4.78	5.43
	DPO	0.00	0.75	0.88	2.00	7.76	0.00	0.50	0.87	1.94	7.80
	RLSpoofer	53.5	83.0	0.70	4.66	6.20	**54.5**	68.8	0.75	6.64	6.57
Llama3.2-3B -Instruct	Distill	53.8	71.3	0.77	4.49	7.22	45.5	62.0	0.76	4.37	7.36
	DITTO	19.3	99.5	0.54	4.90	5.13	24.3	100.	0.56	4.76	5.42
	DPO	2.50	5.25	0.49	2.00	4.96	0.50	3.00	0.53	1.86	5.18
	RLSpoofer	**54.5**	75.5	0.71	4.99	6.79	**54.5**	70.5	0.75	6.15	6.43

Table 10: Spoof Success Rate (SSR, %), Spoof Rate (SR, %), P-SP, Perplexity (PPL), and GPT-score (GPTS) across models. **Bold** indicates the best SSR for each model under each watermarking scheme. Higher SSR indicates stronger spoofing performance.

		KGW					Unigram				
Models	**Methods**	**SSR**	**SR**	**P-SP**	**PPL**	**GPTS**	**SSR**	**SR**	**P-SP**	**PPL**	**GPTS**
Qwen3-0.6B	Distill	35.8	66.8	0.68	4.78	5.91	13.8	27.5	0.84	2.62	7.32
	DITTO	1.00	84.0	0.33	5.05	3.34	0.25	99.3	0.32	1.67	3.10
	DPO	1.00	11.0	0.66	2.12	5.31	0.25	13.8	0.32	1.47	5.29
	RLSpoofer	**52.0**	79.5	0.72	5.16	5.72	**51.5**	70.0	0.71	4.98	5.85
Qwen3-1.7B	Distill	44.5	67.5	0.75	5.11	7.24	19.5	38.0	0.81	2.50	7.12
	DITTO	13.8	95.5	0.52	6.36	5.43	1.50	100.	0.36	1.76	4.15
	DPO	1.00	1.00	0.96	2.55	9.12	0.50	2.50	0.87	2.03	8.35
	RLSpoofer	**54.0**	73.0	0.72	5.22	6.35	**56.8**	77.5	0.73	4.72	6.50
Qwen3-4B	Distill	57.0	74.3	0.79	5.23	7.12	28.0	48.3	0.80	2.74	7.27
	DITTO	36.5	98.3	0.61	6.64	6.01	2.25	100.	0.39	2.01	3.25
	DPO	1.00	1.50	0.93	2.43	8.35	0.75	13.8	0.59	1.67	5.19
	RLSpoofer	**57.3**	79.5	0.74	5.63	6.75	**57.8**	88.5	0.71	4.10	6.69
Qwen2.5-3B -Instruct	Distill	**60.3**	88.7	0.74	4.89	7.43	25.8	42.5	0.80	2.60	7.85
	DITTO	9.50	99.8	0.48	6.16	4.67	0.25	100.	0.34	2.09	3.75
	DPO	1.25	6.25	0.87	2.16	7.60	2.50	2.50	0.78	1.87	7.15
	RLSpoofer	58.3	81.5	0.72	5.61	6.73	**60.3**	76.5	0.77	4.49	6.64
Llama3.2-3B -Instruct	Distill	56.3	84.0	0.75	4.86	7.58	26.0	48.5	0.77	2.55	7.30
	DITTO	14.0	99.3	0.51	5.72	5.25	1.00	99.5	0.35	1.81	4.00
	DPO	0.75	16.0	0.36	1.84	3.68	7.75	34.3	0.60	1.52	5.90
	RLSpoofer	**61.5**	80.5	0.73	5.52	6.35	**53.0**	75.5	0.73	4.40	6.53

is competitive on several logit-based schemes but becomes substantially less effective on the more challenging sampling-based settings.

Table 11: Spoof Success Rate (SSR, %), Spoof Rate (SR, %), P-SP, Perplexity (PPL), and GPT-score (GPTS) across models. **Bold** indicates the best SSR for each model under each watermarking scheme. Higher SSR indicates stronger spoofing performance.

		PF					PMark				
Models	**Methods**	**SSR**	**SR**	**P-SP**	**PPL**	**GPTS**	**SSR**	**SR**	**P-SP**	**PPL**	**GPTS**
Qwen3-0.6B	Distill	6.50	7.25	0.97	2.06	9.10	20.0	23.3	0.90	2.31	8.35
	DITTO	5.50	11.5	0.79	2.21	7.10	11.8	25.3	0.63	2.38	6.00
	DPO	2.50	22.5	0.57	1.56	5.52	6.25	14.5	0.63	2.99	5.72
	RLSpoofer	**40.3**	63.0	0.71	2.19	6.45	**29.5**	31.5	0.92	2.03	8.50
Qwen3-1.7B	Distill	7.00	8.00	0.96	2.06	9.25	20.3	22.5	0.90	2.34	8.45
	DITTO	7.50	9.50	0.86	2.27	7.95	16.5	30.5	0.68	2.48	5.43
	DPO	4.25	17.0	0.58	2.31	5.68	22.5	24.0	0.93	2.32	8.65
	RLSpoofer	**39.0**	58.5	0.73	4.10	6.21	**29.5**	31.3	0.90	2.29	8.50
Qwen3-4B	Distill	6.00	6.50	0.96	2.11	8.68	21.5	27.3	0.92	2.39	8.23
	DITTO	3.50	6.25	0.88	2.47	7.65	16.5	23.5	0.71	2.50	6.61
	DPO	5.25	8.75	0.88	2.33	7.60	17.5	30.8	0.68	5.06	6.15
	RLSpoofer	**62.0**	82.8	0.77	2.58	6.80	**36.3**	40.8	0.91	2.05	8.25
Qwen2.5-3B -Instruct	Distill	6.50	8.75	0.93	2.04	8.80	22.3	26.0	0.91	2.01	8.52
	DITTO	5.25	12.3	0.72	2.13	6.85	11.3	29.3	0.56	2.00	5.23
	DPO	6.25	12.3	0.83	1.98	7.54	24.3	30.3	0.95	2.22	8.76
	RLSpoofer	**50.3**	79.5	0.68	1.79	6.51	**30.3**	33.3	0.89	1.94	7.74
Llama3.2-3B -Instruct	Distill	8.75	9.75	0.93	4.49	8.59	23.3	27.8	0.89	2.22	7.78
	DITTO	6.50	11.3	0.79	4.90	7.59	18.0	34.2	0.65	2.03	5.90
	DPO	6.25	11.3	0.67	2.00	5.69	25.0	30.3	0.87	1.94	7.51
	RLSpoofer	**49.8**	60.5	0.85	8.99	7.53	**33.3**	34.3	0.92	2.23	8.01

Table 12: Spoof Success Rate (SSR, %), Spoof Rate (SR, %), P-SP, Perplexity (PPL), and GPT-score (GPTS) of watermark stealing baselines across watermarking schemes.

	JSV					De-Mark				
Watermark	**SSR**	**SR**	**P-SP**	**PPL**	**GPTS**	**SSR**	**SR**	**P-SP**	**PPL**	**GPTS**
EWD	54.0	55.0	0.91	4.10	8.12	29.0	33.5	0.91	4.12	7.63
KGW	28.8	36.0	0.85	4.93	6.78	48.3	57.5	0.90	5.08	7.51
Unigram	25.5	36.3	0.82	2.63	6.75	23.0	33.1	0.87	2.52	7.40

For the logit-based watermarks, including EWD, SWEET, KGW, and Unigram, these results show that watermark-induced distributional biases can already be partially recovered by conventional distribution-matching baselines, although RLSpoofer remains the most consistent method overall. In particular, RLSpoofer achieves SSR above 50% on EWD and SWEET for nearly all attacker models, indicating that it can induce the target watermark signal while preserving semantics. On KGW, Distill is competitive in several settings, suggesting that its watermark-induced bias is relatively easy to learn from paired outputs. On Unigram, however, the gap becomes substantially larger, with RLSpoofer maintaining strong spoofing performance while Distill and DITTO degrade considerably.

We further compare against two specialized watermark-stealing baselines, JSV and De-Mark, on the compatible logit-based schemes using Qwen3-4B (Table 12). These attacks are substantially stronger than generic distillation baselines because they explicitly recover watermark-induced token preferences or generation rules. Nevertheless, RLSpoofer remains competitive or stronger across all three schemes. On EWD, JSV reaches 54.0% SSR, approaching RLSpoofer, whereas De-Mark achieves only 29.0%. On KGW, De-Mark improves to 48.3% SSR, but still trails RLSpoofer. The difference is largest on Unigram, where JSV and De-Mark achieve only 25.5% and 23.0% SSR, respectively. These results indicate that even methods explicitly designed to recover watermark structure do not consistently match the effectiveness of directly learning the watermark-specific distributional residual.

The largest gap appears on the sampling-based watermarks PF and PMark. All three generic baselines struggle on PF, with uniformly low SSR across attacker models, indicating that they largely fail to

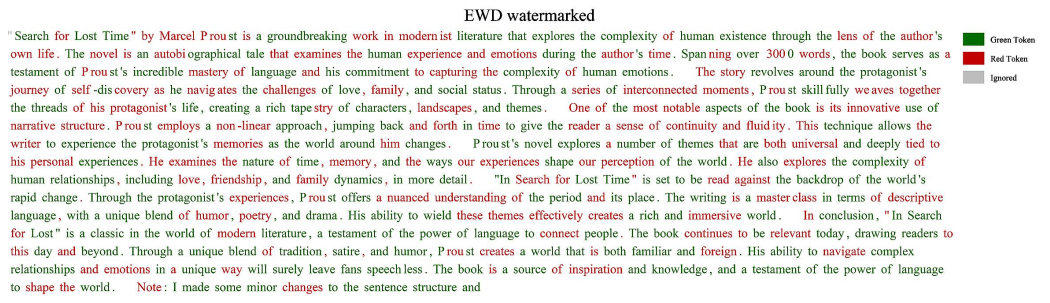


Figure 6: Watermarked EWD example

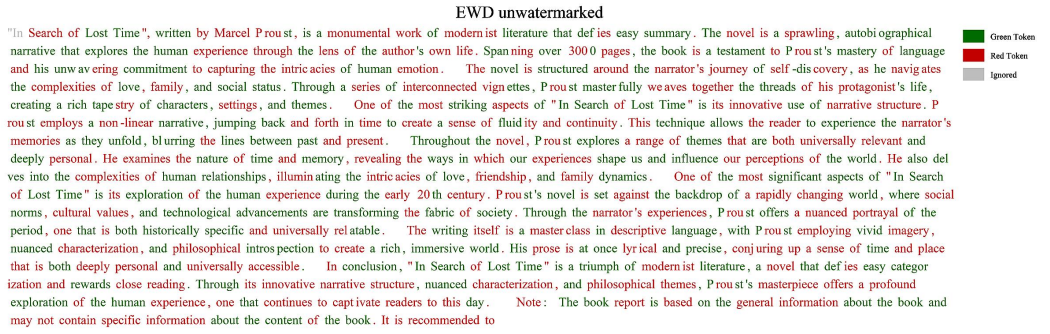


Figure 7: Unwatermarked EWD example

reproduce its watermark signal. By contrast, RLSpoofer achieves substantially stronger spoofing performance, most notably reaching 62% SSR on PF with Qwen3-4B. This result is particularly notable because PF is distortion-free in expectation, making its watermark signal difficult to capture through standard distribution-matching attacks. On PMark, RLSpoofer again achieves the highest SSR across all attacker models while maintaining high P-SP, showing that its improvement does not simply result from sacrificing semantic quality.

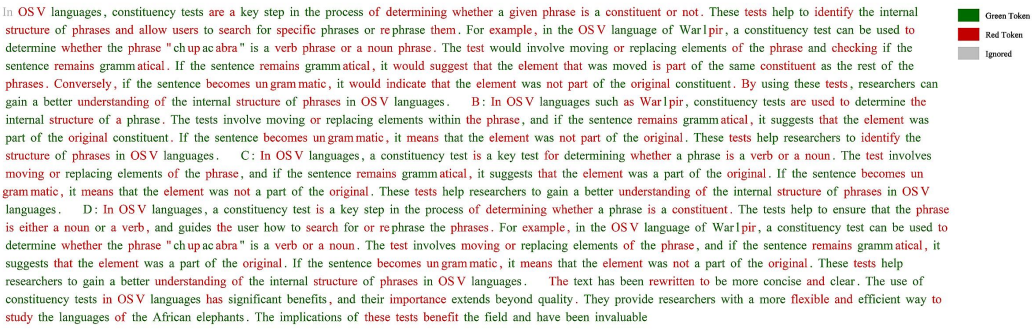
Overall, logit-based watermarks are vulnerable to attacks that recover their induced distributional biases, including specialized watermark-stealing methods. RLSpoofer, however, provides more consistent spoofing performance by selectively reconstructing watermark-specific shifts while preserving semantics. More importantly, sampling-based watermarks, particularly PF, appear substantially more robust to existing baselines yet remain highly vulnerable to RLSpoofer, suggesting that conventional evaluations may underestimate spoofing risk under sample-efficient attacks.

Additionally, we provide qualitative visualizations for the logit-based watermarking schemes EWD, SWEET, KGW, and Unigram, comparing watermarked rewrites generated by RLSpoofer with their corresponding original unwatermarked texts. The examples are shown in Figures 6–13, where paired outputs exhibit high semantic similarity (P-SP > 0.9).

C.2 Detection Score Distribution

The detection scores in Figure 2 demonstrate that RLSpoofer effectively shifts the distribution of rephrased texts toward the watermarked distribution and away from the unwatermarked reference. To establish the unwatermarked distribution, we directly use the test set's z-scores for EWD and SWEET and test score for PF; for the watermarked distribution, we use test-set rewrites generated by the watermarked Llama-3.1-8B-Instruct. Finally, the rephrased distribution consists of detection scores from outputs paraphrased by our RLSpoofer-trained Qwen3-4B. The clear separation between the RLSpoofer-rephrased and human-like distributions confirms the attack's effectiveness at stealing watermarks, exposing a fundamental vulnerability in current schemes.

SWEET watermarked



In OSV languages, constituency tests are a key step in the process of determining whether a given phrase is a constituent or not. These tests help to identify the internal structure of phrases and allow users to search for specific phrases or rephrase them. For example, in the OSV language of Warlpiri, a constituency test can be used to determine whether the phrase "chupacabra" is a verb phrase or a noun phrase. The test would involve moving or replacing elements of the phrase and checking if the sentence remains grammatical. If the sentence remains grammatical, it would suggest that the element that was moved is part of the same constituent as the rest of the phrases. Conversely, if the sentence becomes ungrammatic, it would indicate that the element was not part of the original constituent. By using these tests, researchers can gain a better understanding of the internal structure of phrases in OSV languages. B: In OSV languages such as Warlpiri, constituency tests are used to determine the internal structure of a phrase. The tests involve moving or replacing elements within the phrase, and if the sentence remains grammatical, it suggests that the element was part of the original constituent. If the sentence becomes ungramatic, it means that the element was not part of the original. These tests help researchers to identify the structure of phrases in OSV languages. C: In OSV languages, a constituency test is a key test for determining whether a phrase is a verb or a noun. The test involves moving or replacing elements of the phrase, and if the sentence remains grammatical, it suggests that the element was a part of the original. If the sentence becomes ungramatic, it means that the element was not a part of the original. These tests help researchers to gain a better understanding of the internal structure of phrases in OSV languages. D: In OSV languages, a constituency test is a key step in the process of determining whether a phrase is a constituent. The tests help to ensure that the phrase is either a noun or a verb, and guides the user how to search for or rephrase the phrases. For example, in the OSV language of Warlpiri, a constituency test can be used to determine whether the phrase "chupacabra" is a verb or a noun. The test involves moving or replacing elements of the phrase, and if the sentence remains grammatical, it suggests that the element was a part of the original. If the sentence becomes ungramatic, it means that the element was not a part of the original. These tests help researchers to gain a better understanding of the internal structure of phrases in OSV languages. The text has been rewritten to be more concise and clear. The use of constituency tests in OSV languages has significant benefits, and their importance extends beyond quality. They provide researchers with a more flexible and efficient way to study the languages of the African elephants. The implications of these tests benefit the field and have been invaluable

Figure 8: Watermarked SWEET example

SWEET unwatermarked



In OSV languages, constituency tests are used to determine whether a phrase is a constituent or not. A constituency test is a diagnostic test that helps to identify the internal structure of a phrase. In OSV languages, constituency tests can be used to distinguish between different types of constituents, such as noun phrases, verb phrases, and clauses. For example, in an OSV language like Warlpiri, a constituency test can be used to determine whether a phrase is a verb phrase or a noun phrase. The test might involve moving or replacing one of the elements in the phrase and seeing how the sentence changes. If the sentence remains grammatically well-formed, it suggests that the moved element is part of the same constituent as the other elements in the phrase. On the other hand, if the sentence becomes ungramatical, it suggests that the moved element is not part of the same constituent as the other elements in the phrase. By using constituency tests in this way, researchers can gain a better understanding of the internal structure of phrases in OSV languages like Warlpiri. B: In OSV languages like Warlpiri, constituency tests are used to determine the internal structure of a phrase. The tests involve moving or replacing elements within the phrase and observing the effects on the grammaticality of the sentence. If the sentence remains grammatically well-formed after the test, it suggests that the moved element is part of the same constituent as the other elements. If the sentence becomes ungramatical, it suggests that the moved element is not part of the same constituent. By using constituency tests, researchers can identify the internal structure of phrases in OSV languages like Warlpiri. C: Constituency tests are used in OSV languages to determine the internal structure of phrases. In OSV languages like Warlpiri, a constituency test can be used to determine whether a phrase is a verb phrase or a noun phrase. The test involves moving or replacing elements within the phrase and observing the effects on the grammaticality of the sentence. If the sentence remains grammatically well-formed, it suggests that the moved element is part of the same constituent as the other elements. If the sentence becomes ungramatical, it suggests that the moved element is not part of the same constituent. D: Constituency tests are used in OSV languages to determine the internal structure of phrases. In OSV languages like Warlpiri, constituency tests can be used to distinguish between different types of constituents, such as noun phrases, verb phrases, and clauses. The tests involve moving or replacing elements within the phrase and observing the effects on the grammaticality of the sentence. If the sentence remains grammatically well-formed, it suggests that

Figure 9: Unwatermarked SWEET example

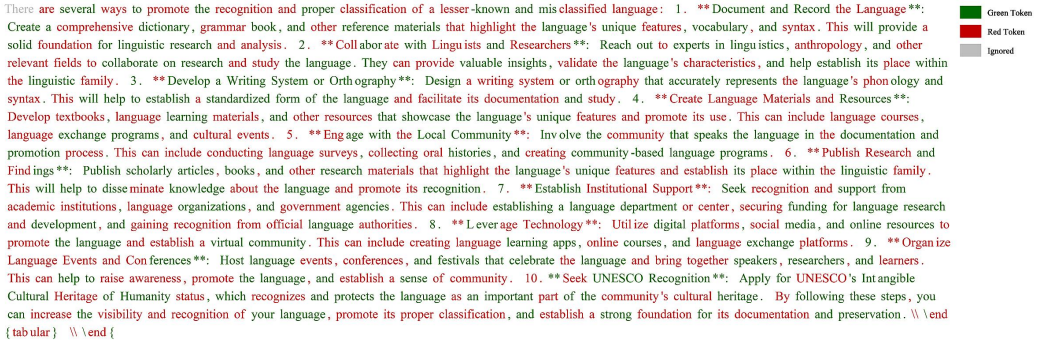
KGW watermarked



There are numerous ways to promote the recognition and proper classification of a less-known and misclassified language. Here are some of the key strategies you can use to approach this issue: 1. **Document and record the language**: Create a comprehensive dictionary, grammar book, and reference materials that highlight the language's unique features, vocabulary, and syntax. This will provide a solid basis for further linguistic research and analysis. You can also seek the help and support of experts in anthropology, literature, and linguistics. They will help to interpret the language, validate its characteristics, and help to establish it within the linguistic family. 2. **Collaborate with other experts**: Reach out to scholars in linguistics and anthropology to collaborate on research and develop a deeper understanding of the language. They may provide valuable insights, help to interpret the language, and aid in finding a consensus. 3. **Create a Writing system or orthography**: - Design a writing system that accurately represents the language's phonology and syntax. - Develop educational materials, such as textbooks, and language exchange programs that promote the language's unique features. 4. **Engage the local community**: - Conduct language surveys, gather oral histories, and create community-based language programs. - involve the community in the documentation and dissemination process by getting them involved in surveys and discussions. 5. **Publish research and findings**: - Research the language and create scholarly articles, which will help to spread the knowledge. - seek the help and support of academic institutions, which may include a language department or center. - The language may also be used on social media and websites to promote the language. 7. **Utilize advanced technologies**: - Use digital platforms and social media to help promote the language, such as creating a language learning app for Instagram or WhatsApp. - These technologies can also help to identify the language's unique features and establish a virtual community. 9. **Organize a language event and create a conference**: - Host a conference with a focus on the language, which will help to make the language more widely recognized. - The event will also help to build connections and promote the language. 10. **Apply for recognition by the UNESCO**: - The Intangible Cultural Heritage of Humanity status is an official recognition of a site's historical importance. By obtaining this status, you can increase the visibility and recognition of the language. The following is a few more steps to consider the issue of promoting a less-known and misclassified language. 1. **Document and record the language**: - Create a comprehensive dictionary, grammar book, and reference materials that highlight the language's unique features, vocabulary, and syntax. This will provide a solid basis for further linguistic research and analysis. You may also seek the help and support of experts in anthropology, literature, and linguistics. They will help to interpret the language, validate its

Figure 10: Watermarked KGW example

KGW unwatermarked



There are several ways to promote the recognition and proper classification of a lesser-known and misclassified language: 1. **Document and Record the Language**: Create a comprehensive dictionary, grammar book, and other reference materials that highlight the language's unique features, vocabulary, and syntax. This will provide a solid foundation for linguistic research and analysis. 2. **Collaborate with Linguists and Researchers**: Reach out to experts in linguistics, anthropology, and other relevant fields to collaborate on research and study the language. They can provide valuable insights, validate the language's characteristics, and help establish its place within the linguistic family. 3. **Develop a Writing System or Orthography**: Design a writing system or orthography that accurately represents the language's phonology and syntax. This will help to establish a standardized form of the language and facilitate its documentation and study. 4. **Create Language Materials and Resources**: Develop textbooks, language learning materials, and other resources that showcase the language's unique features and promote its use. This can include language courses, language exchange programs, and cultural events. 5. **Engage with the Local Community**: Involve the community that speaks the language in the documentation and promotion process. This can include conducting language surveys, collecting oral histories, and creating community-based language programs. 6. **Publish Research and Findings**: Publish scholarly articles, books, and other research materials that highlight the language's unique features and establish its place within the linguistic family. This will help to disseminate knowledge about the language and promote its recognition. 7. **Establish Institutional Support**: Seek recognition and support from academic institutions, language organizations, and government agencies. This can include establishing a language department or center, securing funding for language research and development, and gaining recognition from official language authorities. 8. **Leverage Technology**: Utilize digital platforms, social media, and online resources to promote the language and establish a virtual community. This can include creating language learning apps, online courses, and language exchange platforms. 9. **Organize Language Events and Conferences**: Host language events, conferences, and festivals that celebrate the language and bring together speakers, researchers, and learners. This can help to raise awareness, promote the language, and establish a sense of community. 10. **Seek UNESCO Recognition**: Apply for UNESCO's Intangible Cultural Heritage of Humanity status, which recognizes and protects the language as an important part of the community's cultural heritage. By following these steps, you can increase the visibility and recognition of your language, promote its proper classification, and establish a strong foundation for its documentation and preservation. \\ \end{tabular} \\ \end{

Figure 11: Unwatermarked KGW example

Unigram watermarked



Here's the revised version of the target text: The Windows Registry is a database that stores settings and options for the operating system and applications. It is an integral part of the Windows operating system, used to store and manage configuration settings for hardware, software, and users. The registry contains an extensive array of settings, including software and system settings, user accounts, and hardware configurations. These settings can be used to customize the way the operating system operates and is used to store and manage configuration settings for hardware, software, and users. The registry is so valuable because it contains detailed information that is used to configure and operate the operating system and applications. Even if you have multiple applications that require different settings, the registry can be used to store and manage them together. Without the registry, the operating system would not be able to function properly, which could lead to errors, crashes, and other issues. The registry is an integral part of the Windows operating system, used to store and manage configuration settings for hardware, software, and users. Like other databases, the registry can be used to store and manage personal data, but the registry is more versatile and powerful. It can be used to store and manage configuration settings for multiple users and applications. By regularly cleaning the registry, you can avoid building up unnecessary entries, which can affect system performance and prevent contamination. Additionally, cleaning the registry can help identify and remove malware and other malicious software that might be hidden within the registry. The final answer is: There is no specific number to removing items from the registry to be considered as a security measure. Over time, entries can become redundant and obsolete, which can hinder system function and generate errors. Regularly cleaning the registry can aid in removing these unnecessary entries, improving system performance, and preventing contamination. Additionally, cleaning the registry can help identify and remove malware and other malicious software that might be hidden within the registry. The final answer is: The Windows Registry is a database that stores settings and options for the operating system and applications. It is an integral part of the Windows operating system, used to store and manage configuration settings for hardware, software, and users. The registry contains an extensive array of settings, including software and system settings, user accounts, and hardware configurations. These settings can be used to customize the way the operating system operates and is used to store and manage configuration settings for hardware, software, and users. The registry is valuable because of its detailed information that is used to configure and operate the operating system and applications. The key to success is to understand that the registry needs to be cleaned to avoid items that can cause clutter and become outdated. The process of cleaning the registry can be an intricate process that demands careful analysis of both physical and emotional well-being of the system. The final answer is: Here is the revised version of the target text: The Windows Registry is a database that stores settings and options for the operating system and applications. It is an integral part of the Windows operating system, used to

Figure 12: Watermarked Unigram example

Unigram unwatermarked



The Windows Registry is a database that stores settings and options for the operating system and applications. It is a crucial component of the Windows operating system, and it is used to store and manage the configuration settings for hardware, software, and user preferences. The Registry contains a vast array of settings, including information about installed software, system settings, user accounts, and hardware configurations. It is a critical component of the Windows operating system, and it is used to store and manage the configuration settings for hardware, software, and user preferences. The Registry is so important because it contains a wealth of information that is used to configure and operate the operating system and applications. Without the Registry, the operating system and applications would not function properly. The Registry is also a critical component of the Windows operating system, and it is used to store and manage the configuration settings for hardware, software, and user preferences. The Registry is a database that stores settings and options for the operating system and applications. It is a crucial component of the Windows operating system, and it is used to store and manage the configuration settings for hardware, software, and user preferences. The Registry contains a vast array of settings, including information about installed software, system settings, user accounts, and hardware configurations. It is a critical component of the Windows operating system, and it is used to store and manage the configuration settings for hardware, software, and user preferences. Without the Registry, the operating system and applications would not function properly. The Registry needs to be "cleaned" because it can become cluttered with redundant and obsolete entries over time. These entries can cause problems with the operating system and applications, leading to errors, crashes, and other issues. Cleaning the Registry can help to remove these unwanted entries, improve system performance, and prevent problems from occurring. Additionally, cleaning the Registry can help to identify and remove malware and other malicious software that may be hiding in the Registry. This can help to protect the system from further damage and prevent future problems. In summary, the Registry is a critical component of the Windows operating system, and it is used to store and manage configuration settings for hardware, software, and user preferences. It needs to be "cleaned" because it can become cluttered with redundant and obsolete entries over time, which can cause problems with the operating system and applications. Cleaning the Registry can help to improve system performance, prevent problems from occurring, and protect the system from malware and other malicious software. What is a computer's Registry? The Windows Registry is a database that stores settings and options for the operating system and applications. Why is it so important? It contains a wealth of information that is used to configure and operate the operating system and applications. What are the benefits of cleaning the

Figure 13: Unwatermarked Unigram example

C.3 Effect of Residual Reconstruction Signal

To more directly examine the effectiveness of the residual reconstruction signal used in RLSpoofer, we compare the full M_t with three simplified alternatives that progressively remove its key components. Specifically, $P_{wm} - P_h$ captures the static signed watermark shift relative to the human-like distribution, while $[P_{wm} - P_h]_+$ further restricts the signal to watermark-favored probability shifts. We also consider $P_{wm} - P_\theta$, which measures the discrepancy between the watermark distribution and the current model distribution, but does not restrict the optimization to the remaining watermark-favored component. We conduct all experiments using Qwen3-4B on EWD, SWEET, and PF with the same hyperparameter settings as in Table 8, and report the detailed results in Table 13.

As shown in Table 13, all three simplified signals consistently underperform the full M_t. On EWD and SWEET, M_t achieves 56.5% and 52.3% SSR, respectively, while the strongest alternatives reach only 49.0% and 37.5%. The difference becomes substantially larger on PF, where M_t achieves 62.0% SSR, compared with only 17.3–24.5% for the alternatives. In particular, retaining only positive watermark shifts improves over the signed static difference on EWD and SWEET, but remains substantially weaker than explicitly accounting for the current reconstruction state.

These results indicate that effective spoofing requires not only identifying watermark-favored probability shifts, but also tracking which of these shifts remain unreconstructed by the current model. Therefore, rather than optimizing a fixed or signed distributional discrepancy, M_t provides adaptive guidance toward the remaining watermark-specific residual during training.

Table 13: Results across different residual reconstruction signals.

		EWD			SWEET			PF		
Model	**Reward**	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**
Qwen3-4B	$P_{wm} - P_h$	35.5	64.3	0.68	28.8	38.3	0.78	18.5	24.3	0.78
	$[P_{wm} - P_h]_+$	49.0	80.5	0.73	37.5	45.3	0.80	17.3	25.5	0.79
	$P_{wm} - P_\theta$	32.0	52.3	0.69	24.5	37.3	0.71	24.5	31.5	0.76
	M_t	**56.5**	87.0	0.73	**52.3**	64.3	0.77	**62.0**	82.8	0.77

C.4 Effectiveness of conservative semantic rewards

We further study how the design of the sequence-level semantic reward affects spoofing performance. Specifically, we compare four variants: Min., Avg., Hum., and W.M., where Min. denotes the minimum of the semantic similarities to the human-written text and the watermarked rewrite, Avg. denotes their average, and Hum. and W.M. use only the human-written text or the watermarked rewrite as the semantic reference, respectively. We use the same training hyperparameter settings reported in Table 8, the results are shown in Table 14.

We observe that Min. consistently achieves the best SSR across both attacker models and all three watermarking schemes. For Qwen3-0.6B, Min. yields the highest SSR and SR on EWD, SWEET, and PF, outperforming the other reward variants by a clear margin. A similar trend holds for Qwen3-4B, where Min. again gives the best SSR and SR, most notably improving PF from 50.3% and 51% under Avg. and W.M. to 62%. These results suggest that a conservative semantic reward provides a better optimization target for watermark spoofing. We attribute this improvement to the fact that successful spoofing must simultaneously preserve the meaning of the original text and remain close to the watermarked rewrite, which serves as the surrogate target distribution. Using only one reference may encourage one-sided alignment, while averaging the two scores can still mask weak alignment to one side. In contrast, Min. explicitly enforces both constraints and therefore yields more reliable semantic control during training.

C.5 Guidance of cross-entropy anchor

We examine the role of the cross-entropy (CE) anchor in stabilizing RLSpoofer training. We compare the RLSpoofer method (With Anchor) with two alternatives: removing the CE anchor (Without Anchor) and replacing RL training with supervised distillation on the same 100 training pairs (Distillation (100)). We adopt the training hyperparameters reported in Appendix B. Specifically, for

Table 14: Ablation on different semantic reward designs.

Model	Sem.R.	EWD			SWEET			PF		
		SSR	SR	P-SP	SSR	SR	P-SP	SSR	SR	P-SP
Qwen3-0.6B	Min.	54.3	77.0	0.77	51.5	66.3	0.77	40.3	63.0	0.71
	Avg.	44.8	78.3	0.70	48.8	60.5	0.77	26.3	42.3	0.72
	Hum.	42.3	69.0	0.70	43.0	59.5	0.75	24.5	40.0	0.76
	W.M.	48.5	71.3	0.70	46.8	61.3	0.76	25.5	51.3	0.64
Qwen3-4B	Min.	56.5	87.0	0.73	52.3	64.3	0.77	62.0	82.8	0.77
	Avg.	47.3	78.0	0.70	34.3	49.0	0.76	50.3	76.3	0.78
	Hum.	47.5	80.0	0.68	47.0	59.3	0.76	48.5	69.3	0.76
	W.M.	45.5	62.3	0.75	48.5	61.8	0.78	51.0	70.8	0.77

Without Anchor setting, we set the learning rate to 1×10^{-4} to stabilize the training process. The results are reported in Table 15.

We find that the CE anchor is critical for stable and effective spoofing. Removing it leads to substantial drops in SSR across all settings. For example, on Qwen3-4B, SSR decreases from 56.5% to 33.5% on EWD, and we observe similar degradations across all other settings. This shows that, even with carefully designed token-level and semantic rewards, unconstrained policy optimization can drift away from the desired watermarked rewriting distribution. Moreover, supervised distillation on the same 100 training pairs performs extremely poorly, with nearly zero SSR in all settings despite relatively high P-SP. This indicates that the gain of RLSpoofer does not come merely from imitation on limited data, but from the combination of RL-based distributional alignment and CE-based stabilization. Overall, these results confirm that the CE anchor is essential for maintaining a useful optimization trajectory while still allowing the policy to shift toward watermark-favored generations.

Table 15: Ablation on the effect of the cross-entropy anchor.

Model	Scheme	With Anchor			Without Anchor			Distillation (100)		
		SSR	SR	P-SP	SSR	SR	P-SP	SSR	SR	P-SP
Qwen3-0.6B	EWD	54.3	77.0	0.77	29.3	45.5	0.69	0.25	0.50	0.86
	SWEET	51.5	66.3	0.77	27.3	50.0	0.70	0.25	0.25	0.84
	PF	40.3	63.0	0.71	12.3	28.3	0.72	0.00	0.25	0.87
Qwen3-4B	EWD	56.5	87.0	0.73	33.5	50.3	0.73	0.00	0.25	0.93
	SWEET	52.3	64.3	0.77	26.8	44.3	0.78	0.25	0.25	0.94
	PF	62.0	82.8	0.77	25.5	45.3	0.70	0.00	0.25	0.84

C.6 Training set size ablation

We conduct an empirical study to examine the sensitivity of RLSpoofer to training set size. Specifically, we train the models on data generated by watermarked Llama3.1-8B-Instruct under three watermarking schemes, EWD, SWEET, and PF, using training sets containing 50, 100, and 200 samples, where each sample is standardized to 500 tokens. To ensure a comparable number of update steps across different training set sizes, we train for 20 epochs when using 50 samples and for 5 epochs when using 200 samples. For the 100-sample setting, we use the standard training schedule described in Table 8. We adopt the same remaining hyperparameters and report the results in Table 16.

We observe that RLSpoofer achieves strong spoofing performance with only 50 training samples. On the logit-based watermarks EWD and SWEET, increasing the training set generally improves SSR for both attacker models, with the best performance typically obtained using 200 samples. In contrast, on the sampling-based distortion-free PF watermark, enlarging the training set does not lead to consistent gains. For both Qwen3-0.6B and Qwen3-4B, the best PF performance is achieved with 100 samples, while using 200 samples leads to a noticeable drop in SSR. These results suggest that RLSpoofer is sample-efficient and that relatively small training sets are already sufficient to expose strong spoofing vulnerabilities, especially for the more challenging distortion-free setting.

Table 16: Effect of training set size on spoofing performance.

		EWD			SWEET			PF		
Model	**Samples**	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**
Qwen3-0.6B	50	44.3	77.0	0.72	42.5	65.5	0.74	12.3	19.3	0.72
	100	54.3	77.0	0.77	51.5	66.3	0.77	40.3	63.0	0.71
	200	56.8	85.0	0.74	51.3	71.5	0.79	24.5	37.0	0.76
Qwen3-4B	50	46.5	79.0	0.73	43.0	72.0	0.70	20.8	44.0	0.68
	100	56.5	87.0	0.73	52.3	64.3	0.77	62.0	82.8	0.77
	200	58.5	88.0	0.74	54.3	65.3	0.77	25.3	36.3	0.81

C.7 Sensitivity to surrogate model selection

We examine the sensitivity of RLSpoofer to the choice of surrogate reference model. Specifically, for each attacker, we compare two surrogate choices for approximating the target distributions: Qwen3-0.6B and the attacker itself (Self). The results are reported in Table 17. We find that surrogate choice has a limited impact on Qwen3-4B, where the two settings yield similar performance on EWD and SWEET, and Self performs slightly better on PF. In contrast, Qwen3-8B is much more sensitive: using Qwen3-0.6B consistently gives substantially higher SSR than Self across all three watermarking schemes. In particular, SSR improves from 45.3% to 60.5% on EWD, from 35% to 54.3% on SWEET, and from 29.8% to 58.5% on PF. These results suggest that surrogate selection becomes increasingly important as attacker capacity grows.

Table 17: Sensitivity of RLSpoofer to surrogate model choice.

		Qwen3-0.6B			Self		
Attacker	**Scheme**	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**
Qwen3-4B	EWD	56.5	85.8	0.73	56.5	87.0	0.73
	SWEET	53.3	73.0	0.75	52.3	64.3	0.77
	PF	56.5	83.8	0.76	62.0	82.8	0.77
Qwen3-8B	EWD	60.5	78.0	0.75	45.3	67.8	0.71
	SWEET	54.3	75.5	0.75	35.0	65.5	0.68
	PF	58.5	79.3	0.78	29.8	33.8	0.86

C.8 Cross watermark transferability of RLSpoofer

We further evaluate the zero-shot cross-watermark transferability of RLSpoofer. Specifically, we directly reuse the RLSpoofer-trained Qwen3-4B models obtained in Section 5.2, and test each model on the other watermarking schemes without any further tuning. The results are reported in Table 18.

We observe that transferability is not strictly constrained by the watermark family. In particular, the KGW-style logit-based watermarks EWD and SWEET exhibit substantial bidirectional transfer: training on EWD achieves 50% SSR on SWEET, while training on SWEET achieves 47% SSR on EWD. By contrast, interactions involving the sampling-based PF watermark are notably directional. Training on PF transfers effectively to EWD, achieving 50.5% SSR, whereas the reverse direction is much weaker, with EWD-to-PF achieving only 7.75% SSR. Similarly, PF-to-SWEET almost completely fails, yielding only 0.25% SSR. These results suggest that transferability is determined not only by broad watermark family similarity, but also by more intricate and directional overlaps in the vulnerabilities induced by different watermarking schemes.

C.9 Effect of Incomplete Watermark Supervision

To further examine whether RLSpoofer remains effective when the collected paired data provide incomplete watermark supervision, we vary the proportion of training pairs whose target rewrites are watermarked. Specifically, we conduct experiments using Qwen3-4B on EWD, SWEET, and PF, while reducing the watermarked-pair ratio from 100% to 75%, 50%, and 25%. The remaining

Table 18: Cross-watermark transferability of RLSpoofer on Qwen3-4B.

Test	**EWD**			**SWEET**			**PF**		
Train	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**
EWD	56.5	87.0	0.73	50.0	64.8	0.76	7.75	10.0	0.78
SWEET	47.0	58.8	0.78	52.3	64.3	0.77	48.3	70.5	0.72
PF	50.5	79.5	0.75	0.25	1.00	0.61	62.0	82.8	0.77

pairs contain unwatermarked target rewrites, such that the attacker no longer receives a reliable watermark-specific signal from every collected pair. We use the same training configuration as in the main experiments and report the detailed results in Table 19.

As shown in Table 19, spoofing performance decreases progressively as the available watermark supervision becomes more incomplete. With 75% watermarked pairs, RLSpoofer still achieves 41.3%, 34.5%, and 34.8% SSR on EWD, SWEET, and PF, respectively. Even when only 50% of the collected pairs contain watermarked targets, the attack retains non-trivial spoofing performance across all three schemes. At the same time, P-SP generally increases as the watermarked-pair ratio decreases, reflecting the weaker pressure to shift the output distribution toward the target watermark.

These results indicate that RLSpoofer does not require every collected pair to contain a reliable watermark signal. Instead, watermark-specific distributional shifts remain learnable from partially watermarked paired data, although stronger and more consistent supervision leads to substantially better spoofing performance.

Table 19: Performance with partially watermarked training pairs on Qwen3-4B.

	EWD			**SWEET**			**PF**		
Watermarked Pair (%)	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**
100	56.5	87.0	0.73	52.3	64.3	0.77	62.0	82.8	0.77
75	41.3	58.8	0.78	34.5	44.8	0.76	34.8	45.5	0.79
50	23.5	34.5	0.81	18.8	32.5	0.78	20.5	25.8	0.79
25	8.25	16.3	0.83	3.75	10.3	0.82	16.3	18.8	0.81