TOWARDS STATISTICALLY OPTIMAL AND SEMANTICALLY ROBUST WATERMARKING FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

The widespread adoption of large language models (LLMs) in real-world applications poses critical challenges for reliably detecting LLM-generated text. Logit-based watermarking is a prominent approach that enables low-overhead statistical detection by biasing token selection through keyed binary masks. However, existing methods often suffer from limited detection signals under constrained perturbations, and remain vulnerable to content-preserving attacks. To address the detectability limitation, we theoretically establish a first-order characterization of watermark detectability in terms of the predictive probability mass induced by the vocabulary partition and derive a closed-form optimality condition that maximizes the expected detection z-score under a fixed perturbation strength. Building on this result, we propose **SoptiMark**, which combines a *Cumulative Probability Mask* that approximates this optimal allocation with an *Embedding Clustering* mechanism that stabilizes mask reconstruction across semantically related tokens. This design strengthens the standardized watermark signal while improving resilience to content-preserving modifications. Extensive experiments demonstrate that SoptiMark consistently outperforms prior logit-based watermarking methods by preserving substantially stronger detection signals under diverse content-preserving attacks. The code is anonymously available at: `https://anonymous.4open.science/r/SoptiMark-AA5F`

1 INTRODUCTION

As large language models (LLMs) [33, 34, 7, 23] continue to expand into a broad spectrum of real-world applications [17, 32, 4], the accompanying security and trust concerns have become increasingly prominent [41, 18]. Among these, ownership attribution and the reliable detection of LLM-generated text have emerged as central challenges [36, 11, 1], motivating ongoing efforts toward principled watermarking techniques aimed at identifying and safeguarding LLM outputs [19, 21, 35, 5, 15].

Notably, logit-based watermarking methods have attracted significant attention for their low overhead, minimal impact on text quality, and robust detection performance. A seminal example is KGW [19], which introduces a binary masking scheme that randomly divides vocabulary tokens into two disjoint "green" and "red" spaces based on the current input prompt. Each token is then assigned a mask entry, either 1 (green) or 0 (red), based on one or more preceding tokens. At every generation step, a positive perturbation is applied to the logits of green tokens, thereby biasing the sampling toward the green space. Consequently, watermarked text contains a substantially higher proportion of green tokens compared to natural text. This design supports reliable statistical detection, where each token's mask can be causally reconstructed from its prefix, and the presence of a watermark is verified by evaluating the fraction of green tokens in the sequence.

To reliably distinguish watermarked text from natural text, existing methods often inject relatively strong logit biases, thereby yielding higher z-test scores. This, in turn, inevitably leads to degraded generation quality and gives rise to the *quality-effectiveness trade-off* challenge. Many studies have explored ways to improve the quality of watermarked text, though often at the cost of reduced watermark effectiveness [21, 35]. Moreover, the detectability of such methods is tightly coupled to the recoverability of the mask, which depends on the given token sequence, thereby limiting their

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.

robustness against content-preserving attacks, e.g., synonym substitution, character-level perturbation [46], or paraphrasing [20]. Under these attacks, modifying a token can change its mask value (*i.e.*, green-red assignment) and also alter the prompt used to determine the next token's vocabulary partition, producing a double impact that substantially reduces watermark detectability.

In this paper, we introduce **SoptiMark**, a principled framework for LLM watermarking that jointly improves statistical detectability and resilience to content-preserving attacks. By grounding watermark generation in statistical optimality and semantic stability, SoptiMark addresses two central limitations of existing logit-based methods. To enhance the *quality-effectiveness trade-off*, we theoretically characterize the relationship between logit perturbations and the resulting z-score. Our analysis shows that the expected z-score is governed by the probability mass assigned to each vocabulary space and reaches its maximum when the green and red spaces each carry half of the clean probability mass. We therefore introduce the **Cumulative Probability Mask**, which partitions the vocabulary so that the green and red spaces each carry approximately 0.5 of the predictive probability mass, yielding the maximal first-order z-score of $Z \approx \epsilon\sqrt{T}/2$. This indicates that under the same perturbation scale, our partitioning scheme maximizes the detection z-score, thereby improving watermark detectability without introducing additional degradation in text quality.

Building on the improvements in watermark trade-off achieved by the cumulative probability mask, we further focus on enhancing *robustness against content-preserving attacks*. As mentioned previously, conventional logit-based watermarking methods are strictly token-dependent: each token mask is based on one or more preceding tokens. While effective for statistical detection, this design makes the watermark highly sensitive to token-level modifications, as even minor lexical changes can compromise detection reliability. To mitigate this vulnerability, we propose a mask generation method based on **Embedding Clustering** that preserves semantic consistency under attacks. Instead of associating the mask with discrete tokens, we leverage each token's embedding vector to partition the embedding space into multiple semantically coherent token clusters, assigning each cluster a unique hash key. In this way, tokens that are semantically equivalent under content-preserving attacks produce the same hash key, which is used to seed the random number generator, ensuring that common content-preserving attacks do not affect mask reconstruction during detection. Our contributions can be summarized as follows:

- We theoretically characterize the relationship between vocabulary probability mass and watermark detectability, showing that the expected z-score is maximized when the green and red spaces each carry a probability mass of 0.5. Based on this result, we introduce the **Cumulative Probability Mask**, which achieves the maximal first-order z-score of $Z \approx \epsilon\sqrt{T}/2$ under a fixed perturbation scale.
- We propose an **Embedding Clustering** method for mask generation, in which the embedding space is partitioned into semantically coherent clusters assigned unique hash keys, such that semantically equivalent tokens yield identical masks, thereby preserving reconstruction under content-preserving attacks.
- We provide extensive empirical validation across multiple threat scenarios, demonstrating that SoptiMark consistently outperforms prior logit-based watermarking methods in terms of detectability and resilience to content-preserving attacks.

2 Related Work

As LLMs are increasingly deployed in real-world applications, protecting generated text has emerged as an important research direction, leading to extensive studies on LLM watermarking. Existing methods can be broadly categorized according to how the watermark signal is introduced: *logit*-based [19, 21, 35], *sampling*-based [5, 13], and *learning*-based methods [40]. Among them, logit-based approaches are particularly attractive due to their low overhead, minimal impact on text quality, and strong detection performance.

Token-level watermarking. This line of research can be traced back to the seminal work of KGW [19], which randomly partitions the LLM vocabulary into a "green" space and a "red" space at each generation step, and biases generation toward the green space to embed watermarks. However, its strict reliance on vocabulary partitioning may place syntax-critical tokens in the "red" space, limiting its applicability to code generation. Lee et al. [21] alleviates this issue by embedding watermarks

only at high-entropy generation steps. Liu et al. [22] trains a dedicated watermarking model to output semantically consistent logits and embeds the watermark by sampling from a distribution derived from both the original clean model and this specialized watermarking model. WatME [3] groups interchangeable tokens and deliberately distributes members of each lexical group across the green and red lists, ensuring that at least one suitable alternative remains available during generation. DiP [37] preserves the original token distribution through a distribution-preserving reweighting strategy while improving watermark resilience to moderate token modifications. Lu et al. [25] improves watermark detection in low-entropy scenarios by assigning entropy-aware weights to tokens during the detection process. SynthID-Text [8] embeds watermarks through a tournament-style token sampling strategy that subtly biases token selection while preserving text quality and enabling efficient model-free detection. SLSVE [24] adaptively applies watermarking to high-entropy tokens and adjusts logits based on semantic embeddings to improve text quality and robustness.

Semantic robustness and modality extension. Beyond token-level modifications, recent work has incorporated semantic structure to improve resilience against meaning-preserving transformations. Ren et al. [31] proposed a clustering-based method that trains an embedding ring to map sentence embeddings into a bounded circular space, constraining the sentence to remain in the same ring after paraphrasing attacks to improve robustness. CoheMark [42] and K-SemStamp [14] enhance watermarker robustness by partitioning the semantic space through clustering techniques and constraining the model to ensure that each sampled sentence resides within the valid region, thereby ensuring the watermark remains detectable even after semantic-preserving modifications. Watermarking has also been extended beyond textual outputs. Wu et al. [38] addressed the "retokenization mismatch" issue in watermarking for autoregressive audio models by introducing a clustering-based strategy that enforces consistency of watermark signals before and after decoding, thereby preserving the embedded watermark within the generated audio data. Meanwhile, a broad range of studies [35, 28, 9, 16, 12] have devoted considerable effort to reducing the distortion introduced by watermarking or enhancing the robustness of watermarks against various attacks.

3 PRELIMINARIES

Watermarking via logits perturbation has emerged as a widely adopted paradigm for verifying LLM-generated text. A representative approach is KGW [19], where the vocabulary $\mathcal{V}$ is first randomly partitioned into a "green" and a "red" space via a binary mask. Each token $x \in \mathcal{V}$ is assigned $m = 1$ (green) with probability γ, and $m = 0$ (red) otherwise. During generation, a positive bias ϵ is applied to the logits of green tokens.

Formally, given a generated sequence $\{x_t\}_{t=1}^{T}$, let $\boldsymbol{z}_t^{\text{clean}} \in \mathbb{R}^{|\mathcal{V}|}$ denote the clean (non-watermarked) token logits at step t, and $\boldsymbol{m}_t \in \{0,1\}^{|\mathcal{V}|}$ be the corresponding mask. KGW perturbs $\boldsymbol{z}_t^{\text{clean}}$ by adding a scalar value ϵ along the masked direction, i.e.,

$$\boldsymbol{z}_t = \boldsymbol{z}_t^{\text{clean}} + \epsilon \boldsymbol{m}_t. \tag{1}$$

Detection aggregates the number of generated green tokens, $N_G = \sum_{t=1}^{T} m_{t,x_t}$. Under the standard null model, $N_G \sim \text{Binomial}(T, \gamma)$. KGW evaluates z-score as:

$$Z_T = \frac{N_G - \gamma T}{\sqrt{T\gamma(1-\gamma)}}, \tag{2}$$

and reports a watermark when Z_T exceeds a prescribed threshold.

Limitations of Conventional Mask Construction. While intuitive, the mask construction scheme adopted in KGW suffers from two critical yet often overlooked limitations. **(1) Suboptimal detection efficiency.** Since next-token probabilities are highly nonuniform, the probability mass assigned to the green space fluctuates across generation steps, preventing the injected perturbation from achieving the maximum detection z-score. **(2) Susceptibility to text manipulation**: The watermark detection critically relies on masks reconstructed from the current token sequence, rendering the process highly susceptible to text manipulation and potentially preventing accurate mask recovery during detection.

In this paper, we introduce **SoptiMark**, a semantic-preserving watermarking framework that incorporates two key components: **cumulative probability mask** for text generation (Section 4) and **embedding clustering** for mask generation (Section 5). Together, our framework not only strengthens

the watermark signal but also enhances the robustness of watermarked text against content-preserving attacks.

4 TEXT GENERATION WITH CUMULATIVE PROBABILITY MASK

Existing logit-based watermarks randomize a fixed-cardinality subset of the vocabulary. While natural from a keyed partitioning perspective, cardinality is not the quantity through which the perturbation interacts with Softmax. The relevant quantity is instead the predictive probability carried by the selected space. We make this distinction explicit by viewing watermark injection as a local exponential tilt of the clean distribution, and derive a keyed partition that maximizes its resulting statistical distinguishability.

4.1 DETECTABILITY OF A VOCABULARY PARTITION

The statistical efficiency of a vocabulary partition is governed by the predictive probability mass it captures, rather than by the number of tokens assigned to each space. To formalize this distinction, consider a binary mask $\boldsymbol{m}_t \in \{0,1\}^{|\mathcal{V}|}$ and the clean next-token distribution $\boldsymbol{p}_t^{\text{clean}}$. Adding a bias ϵ to the green space induces the exponentially tilted distribution

$$p_{t,i}^{\text{wm}} = p_{t,i}^{\text{clean}} \exp(\epsilon m_{t,i} - \psi_t(\epsilon)), \qquad \psi_t(\epsilon) = \log \sum_{j \in \mathcal{V}} p_{t,j}^{\text{clean}} e^{\epsilon m_{t,j}}. \tag{3}$$

This representation exposes the mask entry as the sufficient statistic of watermark injection. More importantly, the curvature of ψ_t at the clean distribution determines how strongly the partition responds to a small perturbation. Let $P_{G,t} = \sum_{i \in G_t} p_{t,i}^{\text{clean}}$. Since $\psi_t''(0) = P_{G,t}(1 - P_{G,t})$, a partition carries little usable watermark information whenever most predictive probability is concentrated in either space, even if the vocabulary is evenly divided by cardinality.

We characterize how probability mass imbalance affects the standard green-token z-score used by existing watermark detectors. Let Z_T denote this detection statistic over a sequence of T generated tokens, and let

$$\Delta Z_T \triangleq \mathbb{E}_{\text{wm}}[Z_T] - \mathbb{E}_{\text{clean}}[Z_T] \tag{4}$$

denote the expected displacement induced by watermarking. This quantity measures the detection signal contributed by the vocabulary partitions without redefining the underlying detector.

Theorem 4.1. *For a sequence of binary masks $\{\boldsymbol{m}_t\}_{t=1}^T$, define $\eta_t = P_{G,t} - 1/2$. Under a perturbation of magnitude ϵ, the expected displacement of the standard green-token z-score satisfies*

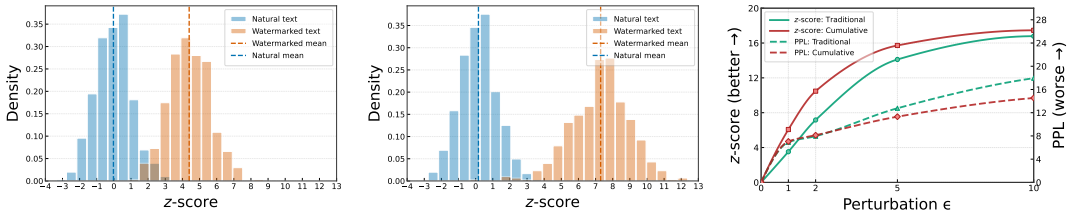
$$\Delta Z_T = \underbrace{\frac{\epsilon\sqrt{T}}{2}}_{\textit{maximum signal}} - \underbrace{\frac{2\epsilon}{\sqrt{T}}\|\boldsymbol{\eta}\|_2^2}_{\textit{imbalance penalty}} + O(\epsilon^2\sqrt{T}). \tag{5}$$

Consequently, the first-order detection signal is maximized if and only if $P_{G,t} = 1/2$ at every generation step. Theorem 4.1 provides a direct decomposition of detection efficiency. The first term represents the largest z-score displacement attainable under a fixed perturbation scale, whereas $\|\boldsymbol{\eta}\|_2^2$ quantifies the cumulative loss induced by probability-mass imbalance. Deviations on either side of $1/2$ incur the same quadratic penalty. Strong local detectability therefore requires $P_{G,t}$ to remain close to $1/2$ throughout generation.

However, conventional fixed-cardinality partitioning does not guarantee this condition. Suppose that G_t is sampled uniformly from all subsets containing $\gamma|\mathcal{V}|$ tokens. Although this construction guarantees $\mathbb{E}[P_{G,t}] = \gamma$, its expected local information is

$$\mathbb{E}[P_{G,t}(1 - P_{G,t})] = \gamma(1-\gamma) - \frac{\gamma(1-\gamma)}{|\mathcal{V}| - 1}\left(|\mathcal{V}|\|\boldsymbol{p}_t^{\text{clean}}\|_2^2 - 1\right). \tag{6}$$

The second term quantifies the information deficit induced by random partitioning. It vanishes only for a uniform predictive distribution and grows with $\|\boldsymbol{p}_t^{\text{clean}}\|_2^2$, which measures probability concentration. Since LLM predictions are typically concentrated on a small subset of plausible tokens, equal-cardinality spaces can exhibit substantial probability mass fluctuations and therefore fail to realize the detection budget in Theorem 4.1. This observation motivates constructing each partition to keep its realized green-space probability close to $1/2$.



(a) Random partitioning (b) Cumulative probability partitioning (c) Quality-effectiveness trade-off

Figure 1: Comparison of vocabulary partitioning strategies. (a) The z-score distributions of watermarked and natural text under conventional random partitioning. (b) The corresponding distributions under cumulative probability partitioning. (c) The relationship between z-score and text perplexity.

4.2 Cumulative Probability Mask Construction

The optimal condition in Theorem 4.1 motivates a cumulative probability construction. At each generation step, the recovered secret key seeds a pseudorandom permutation π_t of the vocabulary. This preserves the keyed randomness of the original partitioning framework and prevents the mask from being determined directly by token probability ranks. Given the clean probabilities, we traverse the permuted vocabulary and compute $C_{t,k} = \sum_{j=1}^{k} p^{\text{clean}}_{t,\pi_t(j)}$, with $C_{t,0} = 0$. We then select the cutoff

$$k_t^\star = \arg\min_{0 \le k \le |\mathcal{V}|} \left| C_{t,k} - \frac{1}{2} \right| \tag{7}$$

and construct the partition as

$$G_t = \{\pi_t(1), \ldots, \pi_t(k_t^\star)\}, \qquad R_t = \mathcal{V} \setminus G_t. \tag{8}$$

Thus, the secret key determines the order in which tokens are considered, while the model probabilities determine only the cutoff along this keyed random order. The resulting partition approximates the optimal condition $P_{G,t} = 0.5$ without sorting tokens by probability or exposing a deterministic high-probability region. A potential concern is that a random vocabulary order could produce a green set consisting entirely of low-probability tokens. The following theorem bounds the probability of this event.

Theorem 4.2 (High-probability token coverage)**.** *At generation step t, order the clean token probabilities as $p_{t,(1)} \ge \cdots \ge p_{t,(|\mathcal{V}|)}$ and define the high-probability set $H_t = \{(1), \ldots, (K)\}$. Let $p_{t,\max} = p_{t,(1)}$ and $\tau_t = p_{t,(K+1)}$, so every token outside H_t has probability at most τ_t. If G_t is constructed using Eqs. 7–8, then*

$$\Pr[G_t \cap H_t = \varnothing] \le \alpha_t := \frac{\binom{|\mathcal{V}|-K}{m_t}}{\binom{|\mathcal{V}|}{m_t}} \le \exp\left(-\frac{K m_t}{|\mathcal{V}|}\right), \tag{9}$$

where $m_t = \left\lceil \frac{1-p_{t,\max}}{2\tau_t} \right\rceil$. Equivalently, the green set contains at least one of the top-K tokens with probability at least $1 - \alpha_t$.

Remark 4.3. Theorem 4.2 shows that the probability of excluding all top-K tokens decreases exponentially with $Km_t/|\mathcal{V}|$. For a long-tailed next-token distribution, the small value of τ_t requires many tail tokens to accumulate sufficient probability mass, thereby increasing m_t and making a tail-only green space increasingly unlikely. For instance, whenever the tail probabilities yield $m_t \ge 0.1|\mathcal{V}|$, Theorem 4.2 bounds the probability of excluding all top-20 tokens by $e^{-2} \approx 0.135$ and all top-50 tokens by $e^{-5} \approx 0.0067$.

We further evaluate the cumulative probability mask construction against conventional random partitioning. As shown in Figure 1, partitioning the vocabulary by cumulative probability mask substantially increases the z-scores of watermarked text without degrading generation quality or increasing the false positive rate.

5 Mask Generation with Embedding Clustering

This section presents *embedding clustering*, the mask key construction component of SoptiMark. Its purpose is to preserve mask consistency when a watermarked token is replaced by a semantically

similar alternative. The component maps nearby token embeddings to a shared key, which is subsequently consumed by the cumulative probability partitioning procedure described in Section 4. Figure 2 illustrates its interface with watermark generation and verification.

5.1 Design Objective

Existing logit-based watermarking methods commonly derive the mask at step t from a token-dependent prompt processor ϕ. When ϕ operates directly on token identities, replacing a token x_i with another token x_j generally changes the recovered key, even when the substitution preserves the surrounding content. The detector then reconstructs a mask different from the one used during generation, weakening the accumulated watermark evidence.

Let $\mathcal{A}$ denote the distribution of token substitutions induced by content-preserving attacks. An ideal key function should minimize the probability of a key mismatch while retaining K distinct keys:

$$\min_{\phi:\,|\mathrm{Im}(\phi)|=K} \Pr_{(x_i,x_j)\sim\mathcal{A}} \left[\phi(x_i) \neq \phi(x_j)\right]. \quad (10)$$

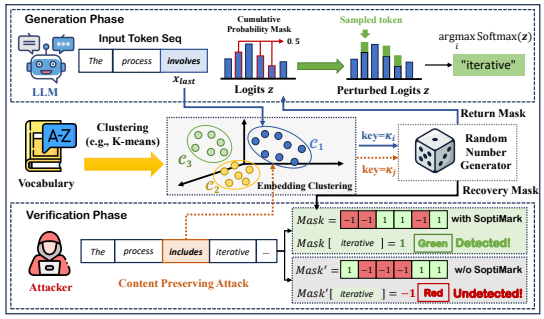


Figure 2: Clustered mask key construction. Generation uses the cluster key to construct the cumulative probability mask, while verification recovers the same key from each observed token.

The cardinality constraint excludes the degenerate solution that assigns every token the same key. Directly optimizing Eq. 10 would require an exhaustive model of possible substitutions. Instead, we use proximity in the model's embedding space as an operational surrogate for semantic interchangeability. This choice is available under our threat model, in which the watermark owner has access to the model parameters and therefore to its vocabulary embedding matrix.

5.2 Embedding Clustering

Let $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{V}|\times d}$ denote the embedding matrix, with row $\boldsymbol{v}_i \in \mathbb{R}^d$ representing token x_i. We partition the vocabulary into K regions by solving

$$\left(c^\star, \{\boldsymbol{\mu}_k^\star\}_{k=1}^K\right) = \arg\min_{c,\{\boldsymbol{\mu}_k\}} \sum_{i=1}^{|\mathcal{V}|} \left\|\boldsymbol{v}_i - \boldsymbol{\mu}_{c(i)}\right\|_2^2, \qquad c(i) \in \{1,\ldots,K\}, \quad (11)$$

where $c^\star(i)$ identifies the cluster containing token x_i and $\boldsymbol{\mu}_k^\star$ is its centroid. We instantiate Eq. 11 using k-means clustering [26].

Each cluster $\mathcal{C}_k = \{x_i : c^\star(i) = k\}$ is assigned a distinct key κ_k, and the prompt processor is defined as $\phi(x_i) = \kappa_{c^\star(i)}$. The resulting partition induces an equivalence relation over the vocabulary: $x_i \sim x_j$ if and only if $c^\star(i) = c^\star(j)$. Since ϕ is constant over each equivalence class, substitutions within the same embedding region recover exactly the same mask generation key. The construction therefore replaces exact token identity with a coarser semantic representation while preserving deterministic key recovery.

Offline construction and online use. Clustering is performed once before deployment, and the resulting assignment $c^\star$ is stored as a vocabulary-level lookup table. During generation, the final token in the current context is mapped to $\kappa_{c^\star(i)}$, which determines the cumulative probability mask. Verification applies the same lookup to each observed token before reconstructing the mask sequence. The component thus changes only the domain of mask key generation and requires neither clustering nor nearest-neighbor search at runtime.

Remark 5.1. The technical distinction lies in how clustering is incorporated into the watermarking pipeline. Wu et al. [38] employs clustering to mitigate retokenization mismatch in autoregressive audio watermarking, whereas CoheMark [42] and K-SemStamp [14] partition semantic space to constrain the region of the next generated sentence. In contrast, our construction uses the induced vocabulary equivalence classes as the domain of the mask key function. Tokens assigned to the same class recover an identical key and therefore reconstruct an identical cumulative probability mask.

Table 1: Performance comparison of different watermarking methods against content-preserving attacks. The best result in each column is highlighted in **bold**, and the second-best result is underlined.

Method	Detection (TPR@1% ↑)							Text Quality		
	Watermarked	*Word-S/30*	*Word-S/50*	*Char-S/30*	*Char-S/50*	*Rewrite-DS*	*Rewrite-D*	*PPL* ↓	*Ent-4* ↑	*Sem-Ent* ↑
Qwen3-4B-Instruct										
No watermark	-	-	-	-	-	-	-	6.649	6.097	2.106
KGW	0.9440	0.1489	0.0384	0.3081	0.0726	0.3122	0.0163	7.386	5.974	1.991
WatME	0.9520	0.1765	0.0468	0.2814	0.0612	0.3485	0.0201	7.214	5.842	1.927
SWEET	0.9761	0.4827	0.2340	0.6104	0.2166	0.4629	0.1168	**7.027**	5.980	1.993
EWD	0.9862	0.5724	0.2401	0.7282	0.2766	0.5072	0.1228	7.170	5.978	1.993
PF	0.9860	0.1128	0.0625	0.2146	0.0489	0.1892	0.0487	7.432	5.687	1.896
SIR	0.9448	0.4828	0.2940	0.4565	0.2328	0.4541	**0.2429**	7.160	6.006	2.002
k-SemStamp	0.9728	0.5321	0.2864	0.6427	0.2455	0.4218	0.2116	8.320	5.380	1.793
DIP	0.9560	0.0461	0.0188	0.1026	0.0363	0.0487	0.0165	7.083	5.981	1.994
MorphMark$_{exp}$	0.9644	0.4883	0.2461	0.4689	0.1045	0.4500	0.1242	7.186	5.982	1.994
SoptiMark (Ours)	**0.9960**	**0.6263**	**0.3549**	**0.7864**	**0.4081**	**0.5826**	0.2327	7.218	**6.075**	**2.025**
OPT-6.7B										
No watermark	-	-	-	-	-	-	-	20.898	5.722	1.729
KGW	0.9023	0.5528	0.2522	0.6586	0.2524	0.6629	0.0785	25.138	5.375	1.659
WatME	0.9186	0.5794	0.2768	0.6241	0.2297	0.7015	0.0912	22.620	5.301	1.612
SWEET	0.9461	0.6945	0.3909	0.7923	0.5587	0.7948	0.1647	23.945	5.575	1.706
EWD	0.9283	0.6985	0.3926	0.7849	0.5068	0.7569	0.1604	23.449	5.541	1.698
PF	0.9236	0.1484	0.0078	0.3326	0.0482	0.2060	0.1044	21.032	5.341	1.610
SIR	0.9685	**0.7748**	**0.5883**	0.7520	0.4385	0.8303	0.2085	24.390	5.582	1.694
k-SemStamp	0.9464	0.6427	0.3489	0.7713	0.5048	0.4025	0.1586	23.980	5.112	1.662
DIP	0.9526	0.0780	0.0265	0.1922	0.0486	0.1388	0.0248	23.532	5.519	1.689
MorphMark$_{exp}$	0.9220	0.6289	0.2722	0.8262	0.4625	0.7966	0.2028	22.849	5.578	1.707
SoptiMark (Ours)	**0.9960**	0.7540	0.5407	**0.8886**	**0.6506**	**0.9208**	**0.2186**	**20.876**	**5.781**	**1.770**
LLaMA-3-8B										
No watermark	-	-	-	-	-	-	-	8.116	5.524	1.962
KGW	0.9905	0.4467	0.1183	0.6322	0.2124	0.2788	0.0913	9.852	**5.827**	1.873
WatME	0.9842	0.4825	0.1376	0.6014	0.1948	0.3215	0.1082	9.587	5.764	1.821
SWEET	0.9364	0.2886	0.0843	0.4325	0.1226	0.2268	0.0601	11.354	5.026	1.891
EWD	0.9600	0.4026	0.1449	0.6108	0.2344	0.3149	0.1245	11.851	5.043	1.884
PF	0.9548	0.1326	0.0352	0.2764	0.0485	0.1976	0.0765	9.452	5.252	1.782
SIR	0.9287	0.3809	0.1922	0.4406	0.1525	0.3583	0.1864	11.699	5.081	1.902
k-SemStamp	0.9596	0.5874	0.2877	0.7070	0.2752	0.3422	0.1251	10.534	5.184	1.756
DIP	0.9129	0.0621	0.0227	0.1474	0.0425	0.0938	0.0207	9.746	5.426	1.886
MorphMark$_{exp}$	0.9542	0.5567	0.3060	0.6344	0.2786	0.3541	0.1083	11.602	5.016	1.898
SoptiMark (Ours)	**0.9964**	**0.7528**	**0.4142**	**0.8526**	**0.3543**	**0.5047**	**0.2213**	**8.887**	5.768	**1.938**

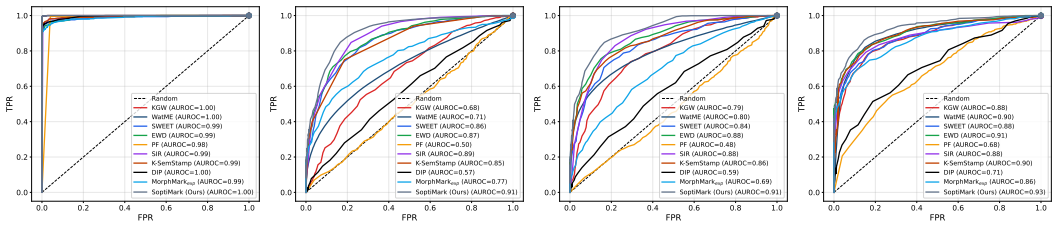
6 EXPERIMENTS

6.1 EXPERIMENTAL SETUP

Datasets and models. Following KGW [19], we sample 1000 documents from C4 [30]. For each document, a fixed-length prefix is used as the model input to generate watermarked text, while the remaining suffix is treated as natural text for comparison. We additionally use GSM8K [6] to evaluate the impact of watermarking on downstream task utility by comparing the mathematical reasoning accuracy with and without watermarking. We evaluate all methods on three backbone models: Qwen3-4B-Instruct [34], OPT-6.7B [43], and LLaMA-3-8B [2]. Model parameters are kept unchanged except for the controlled logit perturbations induced during watermarked generation. To evaluate text quality, we compute the perplexity (PPL) of both watermarked and natural texts using Olmo-3-7B-Instruct [29] as the reference model.

Baseline methods and hyper-parameters. We compare our method with nine representative watermarking approaches, including KGW [19], WatME [3], SWEET [21], EWD [25], PF [47], SIR [22], k-SemStamp [14], DIP [37], and MorphMark$_{exp}$ [35]. All methods adopt the default hyperparameter settings described in their original works. In SoptiMark, the number of clusters is set to 512 for Qwen3-4B-Instruct ($|\mathcal{V}| \approx 150$K) and LLaMA-3-8B ($|\mathcal{V}| \approx 128$K), 128 for OPT-6.7B ($|\mathcal{V}| \approx 50$K) and α = 0.5. For logit-based methods, we fix $\epsilon = 1.25$ across all experiments. The generated watermarked text is limited to approximately 512 tokens. We provide detailed implementation in Appendix C.

Attack settings. To evaluate watermark robustness, we consider three classes of content-preserving attacks. *Word-S/r* performs synonym substitution by randomly replacing a fraction r of words in the watermarked text with their corresponding synonyms from WordNet [27]. *Char-S/r* [46] replaces a fraction r of characters with visually or encoding-similar alternatives. *Rewrite* applies sentence-level paraphrasing using external rewriting models, including *Rewrite-DS* based on DeepSeek-V4-Flash [39] and *Rewrite-D* based on Dipper [20], a large language model specifically trained to weaken watermark through paraphrasing.



(a) No attack (b) Word-S/50 (c) Char-S/50 (d) Rewrite-DS

Figure 3: ROC curves of different watermarking methods under four evaluation settings.

Table 2: TPR@1% of watermarking methods under diverse 50% character-level perturbations, encompassing deletion, substitution, insertion, swapping, and typographical attack.

Model	*Qwen3-4B*					*OPT-6.7B*					*LLaMA-3-8B*				
Method	Deletion ↑	Subst. ↑	Insert ↑	Swap ↑	Typo ↑	Deletion ↑	Subst. ↑	Insert ↑	Swap ↑	Typo ↑	Deletion ↑	Subst. ↑	Insert ↑	Swap ↑	Typo ↑
KGW	0.1486	0.0782	0.1187	0.1145	0.0900	0.2761	0.1986	0.3869	0.2885	0.2527	0.2729	0.3384	0.0981	0.2385	0.2923
WatME	0.2035	0.1427	0.0834	0.1768	0.0612	0.3518	0.2875	0.3012	0.3726	0.1964	0.3497	0.4272	0.1675	0.3189	0.2216
SWEET	0.1547	0.2064	0.1862	0.2523	0.3166	0.5045	0.4987	0.6383	0.2565	**0.5389**	0.1641	0.1744	0.1059	0.1586	0.1693
EWD	0.2742	0.4047	0.2360	0.2982	**0.4028**	0.4347	0.5122	0.6344	0.2527	0.4560	0.2610	0.2968	0.1843	0.2508	0.2327
PF	0.0540	0.0746	0.0470	0.0641	0.0669	0.0468	0.0505	0.0520	0.0686	0.0744	0.0504	0.0626	0.0495	0.0664	0.0707
SIR	0.1824	0.3317	0.2439	0.2765	0.3876	0.3056	0.4892	0.5638	0.4217	0.4134	0.1928	0.1464	0.1089	0.1660	0.2167
k-SemStamp	0.2713	0.3586	0.1934	0.4127	0.3365	0.4382	0.5716	0.4927	0.4184	0.3569	0.3548	0.4651	0.2931	0.2256	0.3467
DIP	0.0020	0.0022	0.0041	0.0000	0.0065	0.0048	0.0067	0.0043	0.0060	0.0089	0.0034	0.0045	0.0042	0.0030	0.0077
MorphMark$_{exp}$	0.2127	0.4646	0.0087	0.0741	0.2440	0.2758	0.0703	0.5880	0.2624	0.4609	0.2047	0.3318	**0.3360**	0.3283	0.1849
SoptiMark (Ours)	**0.4428**	**0.4780**	**0.6464**	**0.4669**	0.3287	**0.6484**	**0.6061**	0.6880	**0.4702**	0.2740	**0.4045**	**0.7828**	0.3304	**0.3380**	**0.4017**

Evaluation metrics. We evaluate the generated watermarked texts from multiple perspectives, including text quality, textual diversity, and downstream task utility, while assessing watermark strength separately. For text quality, we report the PPL computed using an external language model. To further evaluate the diversity of generated texts, we additionally measure trigram text entropy (Ent-4) [45] and semantic entropy (Sem-Ent) [10], where Sem-Ent quantifies the distributional diversity of generated texts based on their semantic cluster assignments. Moreover, we assess whether watermarking affects task-level utility on downstream applications, such as GSM8K accuracy [6] in Appendix C.6. Watermark strength is evaluated using the true positive rate at 1% false positive rate (TPR@1%). **Bold** indicates the best value, while underlined indicates the second-best.

6.2 Main Results

Detectability and text quality evaluation. Table 1 summarizes the detectability and text quality of different watermarking methods. In terms of text quality, we find that SoptiMark achieves competitive text quality across all three evaluated models while providing substantially stronger watermark detectability and robustness. Beyond text quality, SoptiMark achieves state-of-the-art detection performance. We further present the ROC curves of all methods under different evaluation settings in Figure 3, where results consistently show that SoptiMark achieves the best or second-best performance across all scenarios.

Robustness against synonym substitution. Under word-level synonym substitution attacks, SoptiMark demonstrates clear advantages over prior methods. For *Word-S/30* and *Word-S/50*, SoptiMark achieves the highest TPR@1% on Qwen3-4B-Instruct. On OPT-6.7B, SoptiMark achieves the second-best performance. These results validate that embedding clustering-based mask generation effectively preserves watermark detectability under semantic-preserving lexical replacements.

Robustness against character perturbation. Character-level substitutions pose a particularly challenging threat to token-dependent watermarking methods, as they can disrupt LLM tokenization, effectively undermining reliable watermark mask reconstruction. As shown in Table 1, conventional methods such as PF and DIP exhibit severe detection degradation under *Char-S/r* attacks, whereas SoptiMark remains robust, achieving TPR@1% of 0.7864 and 0.4081 under *Char-S/30* and *Char-S/50* on Qwen3-4B-Instruct, and above 0.88 under *Char-S/30* on OPT-6.7B. Since Char-S attacks are inherently stochastic (including random character-level deletion, substitution, insertion, swapping, and typos), we additionally assess watermark robustness under fixed character-level perturbations, as shown in Table 2. Overall, the averaged performance across character-level attacks aligns with the trends observed in Table 1.

Robustness against paraphrasing. Paraphrasing attacks represent the most severe form of content-preserving attack, as they alter sentence structure while retaining semantics. Under both *Rewrite-DS*

Table 3: TPR@1%, PPL and z-score evaluation on Qwen3-4B under various content-preserving attacks, evaluating the effect of removing the cumulative probability mask and embedding clustering.

	Watermarked ↑	*Word-S/50* ↑	*Char-S/50* ↑	*Rewrite-DS* ↑	*Rewrite-D* ↑	PPL ↓	z-score ↑
SoptiMark	0.9960	0.3549	0.4081	0.5826	0.2327	7.218	7.5819
w/o Cumulative Probability Mask	0.9380	0.3284	0.3567	0.4782	0.2122	7.148	6.5008
w/o Embedding Clustering	0.9884	0.1140	0.1465	0.4244	0.0521	7.287	7.2951

and *Rewrite-D*, SoptiMark consistently achieves either the best or second-best detection rates. These results demonstrate that anchoring mask generation to semantic embedding clusters substantially mitigates the vulnerability of logits-based watermarking to paraphrasing attacks. Despite the fact that paraphrasing attacks represent the most severe form of content-preserving attack, they still lead to substantial semantic degradation in the generated texts. We provide a detailed analysis in Appendix C.5, where we report the corresponding BERTScore [44] variations, which measure the semantic similarity between the original watermarked texts and their attacked counterparts under different attack types.

6.3 Ablation Study

We analyze the contributions of the cumulative probability mask and embedding clustering by removing each component while keeping other settings fixed. Table 3 reports the results on Qwen3-4B. The results show that the cumulative probability mask mainly strengthens the watermark signal, whereas embedding clustering is essential for robustness against content-preserving attacks.

Table 4: Computational cost of different watermarking methods, including generation/detection time and peak GPU memory usage under 512-token generation.

Method	Time (s)		Peak Memory Usage (MB)	
	Generation	Detection	Generation	Detection
KGW	19.46	0.105	16,216	468
SWEET	20.63	0.418	16,214	17,672
EWD	19.83	0.245	16,214	27,206
PF	67.73	0.154	63,828	468
SIR	25.22	1.017	17,302	1,534
K-SemStamp (50 trials)	94.28	0.679	16,214	620
DIP	19.74	0.244	16,216	468
MorphMark$_{exp}$	20.68	0.131	16,212	468
SoptiMark (Ours)	21.00	0.285	16,216	17077

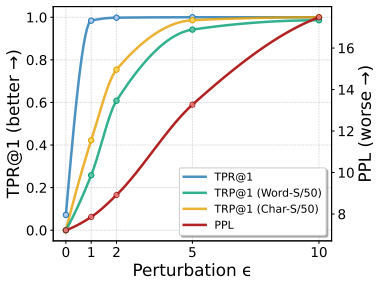


Figure 4: Effect of perturbation.

Furthermore, we evaluate the trade-off between detectability and text quality of SoptiMark-generated watermark texts under different perturbation magnitudes and vocabulary partition ratios. As shown in Figure 4, text quality exhibits an approximately linear degradation as the perturbation magnitude increases, while TPR@1% rapidly saturates, indicating diminishing returns beyond moderate perturbations. This suggests that a perturbation magnitude in the range of 1–2 provides a favorable balance.

6.4 Consumption Evaluation

We further evaluate the computational cost of different watermarking methods on Qwen3-4B-Instruct in terms of runtime and peak GPU memory. As shown in Table 4, SoptiMark achieves generation efficiency comparable to most baseline methods and competitive detection latency. Its detection memory usage is comparable to SWEET and EWD, both of which load the model during detection. The embedding clustering module of SoptiMark is executed offline only once, taking approximately 2 to 3 minutes for 1,000 iterations over a vocabulary of 150K tokens. The resulting clusters can then be reused in subsequent deployments. Further details are provided in Appendix C.2.

7 Conclusion

In this paper, we proposed SoptiMark, a semantic-preserving watermarking framework for LLM outputs that improves detectability and robustness. By introducing a cumulative probability mask to amplify the cumulative watermark signal and an embedding clustering mask generation strategy to ensure watermark robustness, SoptiMark effectively addresses the *quality-effectiveness trade-off* and the vulnerability of logit-based watermarking under content-preserving attacks.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation. We have not used generative AI tools for methodological design and experimental execution. All AI-assisted translations were reviewed against the original text and revised by the authors to ensure accuracy. We take full responsibility for the final content of this work, including any text produced with the assistance of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, private or personally identifiable information, or interaction with production systems. All experiments are conducted in controlled settings using publicly available models and datasets. The proposed watermarking method is intended to support content provenance and the identification of machine-generated text. Although watermarking research may have dual-use implications, our study introduces no collection or release of sensitive data and presents its assumptions and limitations explicitly. We identify no additional ethical concerns arising from this work.

REPRODUCIBILITY STATEMENT

We provide an anonymous repository containing the source code, configuration files, and evaluation scripts required to reproduce our experiments at `https://anonymous.4open.science/r/SoptiMark-AA5F`. The main text describes the proposed method and evaluation protocol, while the appendix reports the implementation details, hyperparameter settings, dataset processing procedures, and complete proofs of the theoretical results. Together, these materials provide the information required to reproduce the reported findings.

REFERENCES

[1] Sara Abdali, Richard Anarfi, CJ Barberan, and Jia He. Decoding the ai pen: Techniques and challenges in detecting ai-generated text. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 6428–6436, 2024.

[2] AI@Meta. Llama 3 model card. 2024. URL `https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md`.

[3] Liang Chen, Yatao Bian, Yang Deng, Deng Cai, Shuaiyi Li, Peilin Zhao, and Kam-Fai Wong. Watme: Towards lossless watermarking through lexical redundancy. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9166–9180, 2024.

[4] Xiang Chen, Chaoyang Gao, Chunyang Chen, Guangbei Zhang, and Yong Liu. An empirical study on challenges for llm application developers. *ACM Transactions on Software Engineering and Methodology*, 2025.

[5] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. In *The Thirty Seventh Annual Conference on Learning Theory*, pp. 1125–1139. PMLR, 2024.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[7] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[8] Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, 2024.

[9] Tianle Gu, Zongqi Wang, Kexin Huang, Yuanqi Yao, Xiangliang Zhang, Yujiu Yang, and Xiuying Chen. Invisible entropy: Towards safe and efficient low-entropy llm watermarking. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6727–6744, 2025.

[10] Seungju Han, Beomsu Kim, and Buru Chang. Measuring and improving semantic diversity of dialogue generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 934–950, 2022.

[11] Xinlei He, Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang. Mgtbench: Benchmarking machine-generated text detection. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 2251–2265, 2024.

[12] Zhiwei He, Binglin Zhou, Hongkun Hao, Aiwei Liu, Xing Wang, Zhaopeng Tu, Zhuosheng Zhang, and Rui Wang. Can watermarks survive translation? on the cross-lingual consistency of text watermark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4115–4129, 2024.

[13] Abe Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. Semstamp: A semantic watermark with paraphrastic robustness for text generation. 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. 4067–4082, 2024.

[14] 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, August 2024.

[15] Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. Unbiased watermark for large language models. *arXiv preprint arXiv:2310.10669*, 2023.

[16] Mingjia Huo, Sai Ashish Somayajula, Youwei Liang, Ruisi Zhang, Farinaz Koushanfar, and Pengtao Xie. Token-specific watermarking with enhanced detectability and semantic coherence for large language models. *arXiv preprint arXiv:2402.18059*, 2024.

[17] Matthew Jörke, Shardul Sapkota, Lyndsea Warkenthien, Niklas Vainio, Paul Schmiedmayer, Emma Brunskill, and James A Landay. Gptcoach: Towards llm-based physical activity coaching. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–46, 2025.

[18] Hanna Kim, Minkyoo Song, Seung Ho Na, Seungwon Shin, and Kimin Lee. When {LLMs} go online: The emerging threat of {Web-Enabled}{LLMs}. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 1729–1748, 2025.

[19] 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.

[20] 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.

[21] 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.

[22] Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. A semantic invariant robust watermark for large language models. *arXiv preprint arXiv:2310.06356*, 2023.

[23] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[24] Yepeng Liu and Yuheng Bu. Adaptive text watermark for large language models. *arXiv preprint arXiv:2401.13927*, 2024.

[25] Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. An entropy-based text watermarking detection method. *arXiv preprint arXiv:2403.13485*, 2024.

[26] James B McQueen. Some methods of classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symposium on Math. Stat. and Prob.*, pp. 281–297, 1967.

[27] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995.

[28] Alexander Nemecek, Yuzhou Jiang, and Erman Ayday. The feasibility of topic-based watermarking on academic peer reviews. *arXiv preprint arXiv:2505.21636*, 2025.

[29] Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2025. URL `https://arxiv.org/abs/2512.13961`.

[30] 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.

[31] Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. A robust semantics-based watermark for large language model against paraphrasing. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 613–625, 2024.

[32] Leming Shen, Qiang Yang, Yuanqing Zheng, and Mo Li. Autoiot: Llm-driven automated natural language programming for aiot applications. In *Proceedings of the 31st Annual International Conference on Mobile Computing and Networking*, pp. 468–482, 2025.

[33] Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

[34] Qwen Team. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

[35] Zongqi Wang, Tianle Gu, Baoyuan Wu, and Yujiu Yang. Morphmark: Flexible adaptive watermarking for large language models. *arXiv preprint arXiv:2505.11541*, 2025.

[36] 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.

[37] Yihan Wu, Zhengmian Hu, Junfeng Guo, Hongyang Zhang, and Heng Huang. A resilient and accessible distribution-preserving watermark for large language models. *arXiv preprint arXiv:2310.07710*, 2023.

[38] Yihan Wu, Georgios Milis, Ruibo Chen, and Heng Huang. Robust distortion-free watermark for autoregressive audio generation models. *arXiv preprint arXiv:2510.21115*, 2025.

[39] Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

[40] Xiaojun Xu, Yuanshun Yao, and Yang Liu. Learning to watermark llm-generated text via reinforcement learning. *arXiv preprint arXiv:2403.10553*, 2024.

[41] Miao Yu, Fanci Meng, Xinyun Zhou, Shilong Wang, Junyuan Mao, Linsey Pan, Tianlong Chen, Kun Wang, Xinfeng Li, Yongfeng Zhang, et al. A survey on trustworthy llm agents: Threats and countermeasures. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 6216–6226, 2025.

[42] Junyan Zhang, Shuliang Liu, Aiwei Liu, Yubo Gao, Jungang Li, Xiaojie Gu, and Xuming Hu. Cohemark: A novel sentence-level watermark for enhanced text quality, 2025. URL https://arxiv.org/abs/2504.17309.

[43] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

[44] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

[45] Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. Generating informative and diverse conversational responses via adversarial information maximization. *Advances in Neural Information Processing Systems*, 31, 2018.

[46] Zhaoxi Zhang, Xiaomei Zhang, Yanjun Zhang, He Zhang, Shirui Pan, Bo Liu, Asif Qumer Gill, and Leo Yu Zhang. Character-level perturbations disrupt llm watermarks. *arXiv preprint arXiv:2509.09112*, 2025.

[47] 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.

A Appendix

B Algorithm Description

This section provides the detailed pseudocode of SoptiMark.

Algorithm 1 SoptiMark: Watermarking with Cumulative Probability Mask and Embedding Clustering

`// Preprocessing cluster vocabulary embeddings`
Input: vocabulary embeddings $\boldsymbol{E}$, number of clusters K
Apply a clustering algorithm (e.g., k-means) to $\boldsymbol{E}$ to obtain clusters $\{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$
Assign each cluster a unique key κ_k
Output: cluster keys $\{\mathcal{C}_k, \kappa_k\}_{k=1}^{K}$

`// Generate watermarked text`
Watermarked Text Generation:
Input model $\mathcal{M}$, private key k, prompt $x_{-N_p}, \ldots, x_{-1}$, perturbation strength ϵ, and cluster mapping $\{(\mathcal{C}_j, \kappa_j)\}_{j=1}^{K}$
Output watermarked text
for $t = 0, 1, \ldots$ **do**
  Obtain clean logits $\boldsymbol{z}_t^{\text{clean}} = \mathcal{M}(x_{-N_p:t-1})$ and probabilities $\boldsymbol{p}_t^{\text{clean}} = \mathrm{Softmax}(\boldsymbol{z}_t^{\text{clean}})$
  Find the cluster $\mathcal{C}_j$ containing the preceding token x_{t-1} and retrieve its cluster key κ_j
  Compute $k_t = \mathrm{HashFunc}(k, \kappa_j)$ and seed the random number generator with k_t
  Generate a pseudorandom permutation $\pi_t = (\pi_t(1), \ldots, \pi_t(|\mathcal{V}|))$ of the vocabulary
  Compute the cumulative probability $C_{t,r} = \sum_{\ell=1}^{r} p_{t,\pi_t(\ell)}^{\text{clean}}$ for $r = 0, \ldots, |\mathcal{V}|$, where $C_{t,0} = 0$
  Select $r_t^\star = \arg\min_r |C_{t,r} - 0.5|$
  Construct the green and red spaces as $\mathcal{G}_t = \{\pi_t(1), \ldots, \pi_t(r_t^\star)\}$ and $\mathcal{R}_t = \mathcal{V} \setminus \mathcal{G}_t$
  Apply the watermark perturbation

$$(\boldsymbol{z}_t)_i = \begin{cases} (\boldsymbol{z}_t^{\text{clean}})_i + \epsilon, & i \in \mathcal{G}_t, \\ (\boldsymbol{z}_t^{\text{clean}})_i, & i \in \mathcal{R}_t \end{cases}$$

  Sample x_t from $\mathrm{Softmax}(\boldsymbol{z}_t)$ and append it to the generated sequence
end for

Algorithm 2 Watermark Detector for SoptiMark

Input text sequence $x_1, \ldots, x_T$, model $\mathcal{M}$, private key k, cluster mapping $\{(\mathcal{C}_j, \kappa_j)\}_{j=1}^{K}$, and detection threshold τ
Output detection result
Initialize $S \leftarrow 0$.
for $t = 1$ **to** $T - 1$ **do**
  Obtain clean logits $\boldsymbol{z}_t^{\text{clean}} = \mathcal{M}(x_{1:t})$ and probabilities $\boldsymbol{p}_t^{\text{clean}} = \mathrm{Softmax}(\boldsymbol{z}_t^{\text{clean}})$
  Find the cluster $\mathcal{C}_j$ containing x_t and retrieve its cluster key κ_j
  Compute $k_t = \mathrm{HashFunc}(k, \kappa_j)$ and seed the random number generator with k_t
  Generate a pseudorandom vocabulary permutation $\pi_t = (\pi_t(1), \ldots, \pi_t(|\mathcal{V}|))$
  Compute $C_{t,r} = \sum_{\ell=1}^{r} p_{t,\pi_t(\ell)}^{\text{clean}}$ for $r = 0, \ldots, |\mathcal{V}|$, where $C_{t,0} = 0$
  Select $r_t^\star = \arg\min_r |C_{t,r} - 0.5|$
  Construct $\mathcal{G}_t = \{\pi_t(1), \ldots, \pi_t(r_t^\star)\}$ and $\mathcal{R}_t = \mathcal{V} \setminus \mathcal{G}_t$
  Compute the green space probability mass $P_{G,t} = C_{t,r_t^\star}$
  Accumulate $S \leftarrow S + 1[x_{t+1} \in \mathcal{G}_t]$
end for
Compute $z = \dfrac{S - T/2}{\sqrt{T}/2}$
Return watermarked if $z > \tau$ and non-watermarked otherwise

C SUPPLEMENTARY EXPERIMENTAL RESULTS

C.1 DETAILED EXPERIMENT SETUP

Implementation Details. For KGW, SWEET, EWD, and MorphMark$_{exp}$, we set the injection noise magnitude to $\epsilon = 1.25$ and the green list size to $\gamma = 0.5$. For SWEET, the entropy threshold is set to 0.9. For SIR, the delta is set to 0.7. For K-SemStamp, we set the cluster margin to 0.01. For MorphMark, we adopt the `exp` mode with $k_{\text{exp}} = 1.30$. For PF, we set `ngram` = 8 and `payload` = 0. For DIP, we set $\alpha = 0.45$. For our SoptiMark, we set the cluster numbers to 512 and 128 for Qwen3-4B-Instruct and OPT-6.7B, respectively, to maintain an average of approximately 300–400 tokens per cluster.

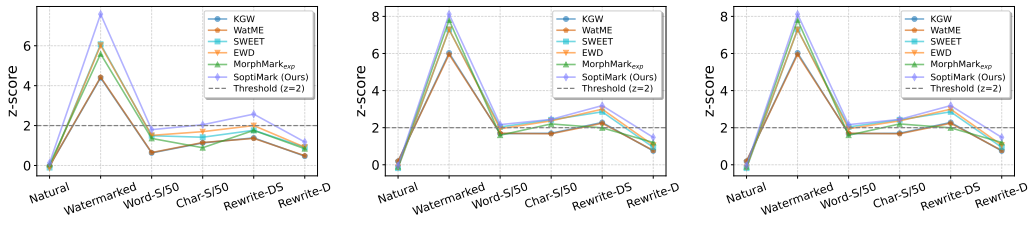
C.2 CONSUMPTION EVALUATION

Beyond watermark robustness and detection accuracy, practical deployment also requires watermarking methods to be computationally efficient. We therefore evaluate the computational consumption of different watermarking methods in terms of both runtime and memory overhead during text generation and watermark detection. Specifically, all methods are evaluated on **Qwen3-4B-Instruct** under identical settings, where each method generates responses of **512 tokens** from the same prompt. For each method, we report the generation time and peak GPU memory usage during inference, as well as the corresponding detection time and memory consumption. All measurements are conducted on the same hardware platform.

As shown in Table 4, most watermarking methods exhibit comparable generation-time overhead and memory consumption, indicating that watermark embedding generally incurs negligible additional cost during inference. Notably, SWEET and EWD incur substantially higher memory usage during detection, as they require loading the language model to compute token entropy scores. In contrast, the detection stage of SoptiMark only involves loading the tokenizer (approximately 468 MB) and an additional embedding clustering repository of about 400 MB, resulting in a moderate yet bounded memory overhead. For a standard vocabulary of 150K tokens (e.g., Qwen-3), performing 1000 clustering runs takes approximately 2–3 minutes on a CPU (Intel(R) Xeon(R) Gold 6246 CPU @ 3.30GHz). This one-time preprocessing cost is negligible before deployment.

C.3 WATERMARK STRENGTH EVALUATION

Logits-based watermarking methods primarily rely on a z-test for detection, where the resulting z-score quantifies the statistical deviation between watermarked and natural texts and thus serves as a direct indicator of watermark strength. Higher z-scores correspond to stronger and more reliably detectable watermarks. As shown in Figure 5, when the detection threshold is set to 2, the z-scores of clean watermarked texts are clearly separated from those of natural texts. Under content-preserving attacks, z-scores consistently decrease across all methods. Nevertheless, SoptiMark maintains comparatively higher z-scores under most evaluated attack scenarios, reflecting more robust watermark accumulation and improved resistance to signal degradation.



(a) Qwen3-4B-Instruct (b) OPT-6.7B (c) LLaMA-3-8B

Figure 5: Z-score comparison under content-preserving attacks. The horizontal dashed line denotes the detection threshold.

Table 5: Evaluation results in terms of watermark detection rate and semantic preservation measured by BERTScore. The first value in each entry denotes the watermark detection rate, while the second value reports the corresponding BERTScore. A higher BERTScore indicates greater semantic similarity between the original and attacked texts.

Qwen3-4B	$\gamma = 0.5$			
	Word-S/50	*Char-S/50*	*Rewrite-DS*	*Rewrite-D*
KGW	0.0384 / 0.6914	0.0726 / 0.6586	0.3122 / 0.6236	0.0163 / 0.5507
SWEET	0.2340 / 0.6892	0.2166 / 0.6578	0.4629 / 0.6219	0.1168 / 0.5618
EWD	0.2401 / 0.6922	0.2766 / 0.6591	0.5072 / 0.6248	0.1228 / 0.5509
PF	0.0625 / 0.6901	0.0489 / 0.6580	0.1892 / 0.6230	0.0487 / 0.5582
SIR	0.2940 / 0.6890	0.2328 / 0.6582	0.4541 / 0.6227	0.2429 / 0.5591
k-SemStamp	0.2864 / 0.6918	0.2455 / 0.6575	0.4218 / 0.6242	0.2116 / 0.5579
DIP	0.0188 / 0.6895	0.0363 / 0.6589	0.0487 / 0.6237	0.0165 / 0.5614
MorphMark$_{exp}$	0.2461 / 0.6944	0.1045 / 0.6572	0.4500 / 0.6240	0.1242 / 0.5581
SoptiMark ($\alpha = 0.5$)	0.3549 / 0.6905	0.4081 / 0.6583	0.5826 / 0.6239	0.1927 / 0.5586

C.4 Semantic Preservation Evaluation

We use BERTScore [44] to evaluate the semantic preservation of watermarked texts under various content-preserving attacks. As shown in Table 5, under word-level substitution and character-level perturbation attacks, most methods are able to maintain relatively high semantic consistency, indicating that these attacks have limited impact on the underlying meaning of the generated texts. Meanwhile, our method consistently achieves a higher watermark detection rate.

In contrast, we observe a significant degradation in semantic preservation under rewrite-based attacks. In particular, Dipper attack (*Rewrite-D*) leads to substantial semantic drift, resulting in pronounced information loss in the generated content. Consequently, the detection performance of all methods degrades considerably under such aggressive rewriting.

C.5 Token Distribution Across Clusters

To further investigate the clustering characteristics of the vocabulary partition, we analyze the distribution of tokens across different clusters. The results are illustrated in Figure 6. We observe that, although a small number of clusters exhibit noticeably larger or smaller sizes, the majority of clusters contain approximately 300 tokens on average. This observation suggests that the clustering process produces a relatively balanced partition of the vocabulary space, which helps avoid severe cluster imbalance and contributes to the stability of the watermarking mechanism.

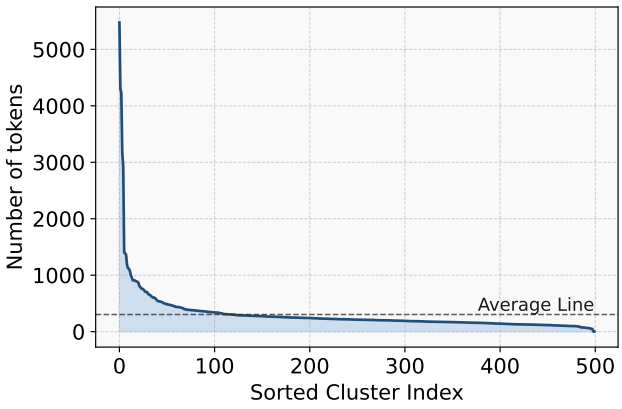


Figure 6: Distribution of tokens across clusters.

We further examine the impact of the cluster number of SoptiMark. Table 6 reports the empirical error rates and PPL under varying numbers of clusters. As the cluster number increases, TPR initially

Table 6: Empirical error rates and model perplexity of SoptiMark ($\alpha = 0.5$) under different cluster numbers with $\gamma = 0.5$. Each row is computed over 500 watermarked texts and 500 natural texts.

Cluster Number	FPR ↓	TNR↑	TPR↑	FNR↓	PPL↓
32	0.004	0.996	0.742	0.258	7.275
64	0.002	0.998	0.776	0.224	7.210
128	0.090	0.910	0.988	0.012	7.158
256	0.170	0.830	0.990	0.010	7.191
512	0.032	0.968	1.000	0.000	7.218
1024	0.028	0.972	0.974	0.026	7.173
2048	0.036	0.964	0.960	0.040	7.236

improves due to finer semantic partitioning, while overly coarse clustering leads to insufficient mask diversity and degraded detection sensitivity. Conversely, excessively large cluster numbers weaken semantic aggregation, resulting in elevated false positive rates and reduced robustness. These results indicate a clear trade-off between semantic coherence and statistical separability, with intermediate cluster sizes providing a more favorable balance between detection reliability and text quality.

C.6 Effect of Watermarking on Downstream Task Utility

We further evaluate whether watermarking affects task-level utility on downstream applications. Specifically, we measure the performance of Qwen3-4B-Instruct on the GSM8K benchmark, a widely used dataset for assessing mathematical reasoning ability. As shown in Figure 7, all watermarking methods lead to some degree of performance degradation compared to the non-watermarked baseline. Among them, our method incurs the smallest performance drop, achieving the highest GSM8K accuracy among all watermarking approaches. This result demonstrates that our watermarking strategy maintains high fidelity to the original model behavior, effectively avoiding the utility degradation commonly observed in prior methods.

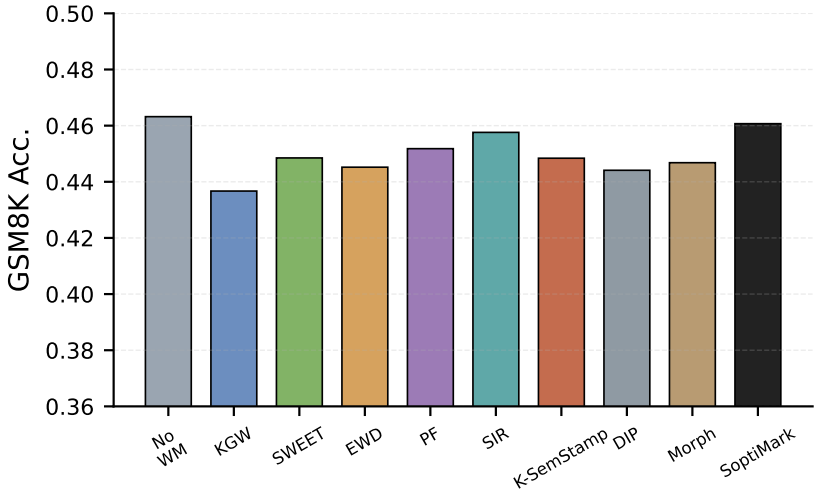


Figure 7: GSM8K accuracy of different watermarking methods.

D Detailed Threat Model

To establish a rigorous framework for secure AI research, it is essential to clearly define the underlying threat model. Such a model specifies the capabilities, goals, and limitations of potential adversaries, and provides a concrete foundation for analyzing and evaluating the robustness and security of the proposed methods. Following Kirchenbauer et al. [19], we assume that the ultimate goal of the adversary is to *remove the watermark while preserving the semantics of the text*.

Threat Model. We consider a two-party setting consisting of a model owner, who provides the watermark-embedded text, and an adversary attempting to remove the watermark from the generated text. The model owner has full control over the model's `LogitsProcessor` and the ability to extract embeddings of all tokens in the vocabulary. The watermarking method is treated as a publicly known scheme, and all internal details are assumed to be accessible to the adversary, except for the hash key, which remains private. The large language model itself is the proprietary asset of the model owner, and its internal parameters are unknown to the adversary. We allow the adversary to arbitrarily modify the generated text, provided that the semantic content of the text is preserved.

D.1 CONTENT-PRESERVING ATTACKS

We consider three widely recognized types of content-preserving attacks, which aim to remove or weaken the watermark embedded in the text while maintaining its original semantic meaning. These attacks have been extensively studied in the literature and are representative of the challenges that any robust watermarking scheme must withstand. By focusing on content-preserving modifications, we evaluate the resilience of watermarking methods under realistic adversarial scenarios where the text's readability and intended meaning are not compromised.

Base Watermarked Text

Carefully designed evaluation protocols are essential for ensuring the reliability and fairness of modern automated systems.

Figure 8: Base watermarked text.

Word Substitution. This attack performs synonym replacement at the word level using WordNet [27]. Specifically, the input text is first tokenized, and each token is assigned a part-of-speech (POS) tag. For each eligible token, the attacker searches the WordNet lexicon for candidate synonyms that share the same POS category, and substitutes the original word with one of its semantic equivalents. By constraining replacements to POS-consistent synonyms, this attack preserves grammatical correctness while maintaining the original semantic content of the text. Figure 9 illustrates a word substitution attack applied to the base watermarked text shown in Figure 8, which serves as the reference for all attack examples.

Word Substitution

Cautiously designed valuation communications protocol are essential for assure the reliability and beauteousness of modernistic automatize system.

Figure 9: Illustration of word-level synonym substitution attacks, where 50% of eligible words are replaced with POS-consistent synonyms.

Paraphrasing. With the widespread adoption of large language models, paraphrasing has become one of the most commonly used content-preserving attack strategies. In this setting, the adversary can simply employ another LLM and provide an appropriate instruction, such as: *"Please rewrite the following text (Only return the rewritten text): {Model Output}"* (following the setup in [35]), to efficiently obtain a semantically equivalent but lexically different text. However, the effectiveness of such generic paraphrasing attacks may be limited due to the lack of specialization of general-purpose LLMs in evading watermark detection. To address this limitation, Krishna et al. [20] proposed DIPPER, a dedicated LLM specifically designed to generate paraphrases that are more effective at bypassing watermark detectors. The example of paraphrasing attacks is illustrated in Fig. 10.

Character Attacks. Character-level attacks, originally proposed by Zhang et al. [46], represent a more subtle and hard-to-detect attack strategy. This class of attacks encompasses multiple types of character manipulations, including *deletion*, *substitution*, *insertion*, *swap*, and *typo* operations. Specifically, *deletion* removes a single character from the original text; *substitution* replaces a character with a visually similar homoglyph from a different Unicode encoding; *insertion* injects an invisible character (e.g., U+200B or U+200D) between two adjacent characters; *swap* exchanges

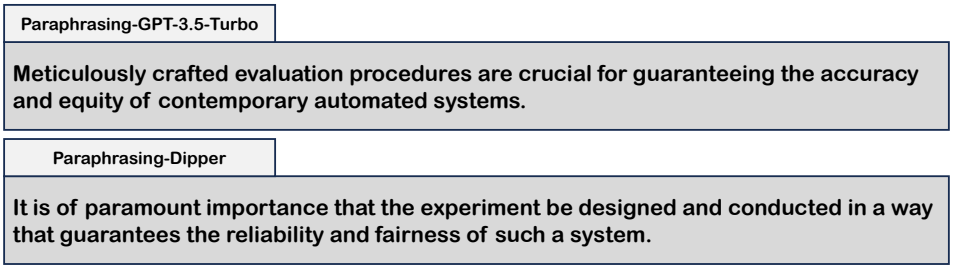


Figure 10: Illustration of paraphrasing attacks, where the upper example is generated using `gpt-3.5-turbo` and the lower example is produced by DIPPER as the attack model.

the order of two neighboring characters; and *typo* alters a character such that the resulting word becomes a misspelling. By modifying only a small fraction of characters in the watermarked text, the adversary deliberately interferes with the tokenization process of the detection-stage LLM, thereby preventing the correct recovery of the underlying mask pattern. Despite introducing minimal or imperceptible visual changes to human readers, such character-level perturbations can significantly degrade watermark detectability. All examples of the different character manipulation operations are illustrated in Figure 11.

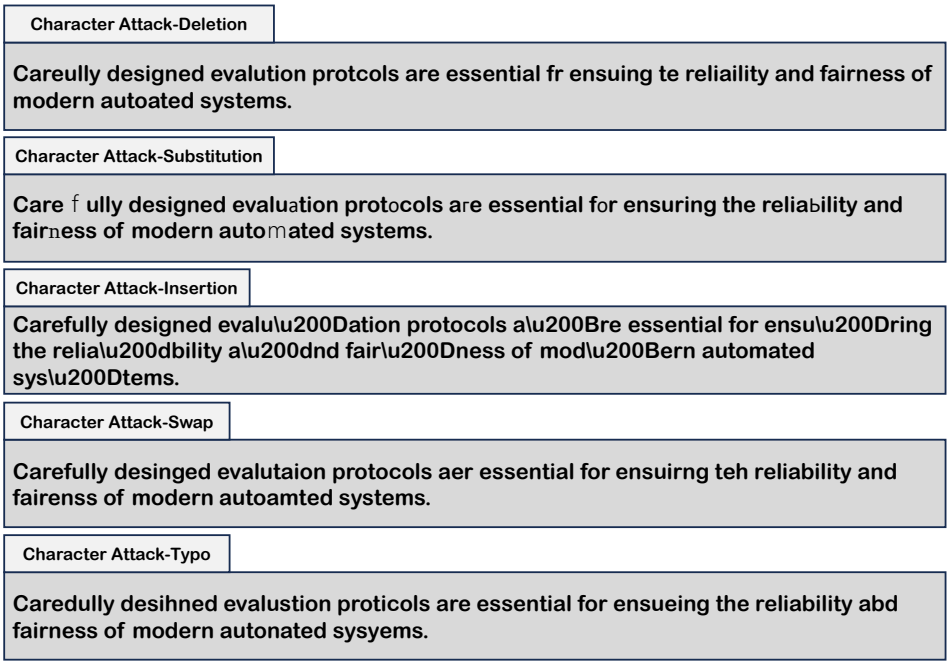


Figure 11: Illustration of character-level attacks, where character manipulations are applied to 50% of the characters, including deletion, substitution with visually similar homoglyphs, insertion of invisible Unicode characters (e.g., U+200B and U+200D), swapping of adjacent characters, and typo-based perturbations.

E LIMITATION

Limitation. The proposed method relies on the assumption that each replacement word can be effectively represented within a single token or mapped to a stable embedding region. This assumption holds well for common lexical substitutions but becomes less reliable when adversarial attacks introduce complex words that are split into multiple sub-tokens by the tokenizer. In such cases, the semantic consistency across tokens is weakened, and the correspondence between semantic clusters

and token-level representations becomes less stable, which may degrade the effectiveness of the masking mechanism under multi-token substitution scenarios. Addressing this limitation under more general multi-token adversarial settings is left for future work.

F MISSING PROOFS

F.1 PROOF OF THEOREM 4.1

Theorem 4.1. *For a sequence of binary masks $\{\boldsymbol{m}_t\}_{t=1}^T$, define $\eta_t = P_{G,t} - 1/2$. Under a perturbation of magnitude ϵ, the expected displacement of the standard green-token z-score satisfies*

$$\Delta Z_T = \underbrace{\frac{\epsilon\sqrt{T}}{2}}_{\textit{maximum signal}} - \underbrace{\frac{2\epsilon}{\sqrt{T}}\|\boldsymbol{\eta}\|_2^2}_{\textit{imbalance penalty}} + O(\epsilon^2\sqrt{T}). \tag{12}$$

Proof. Let

$$S_T \triangleq \sum_{t=1}^{T} (\boldsymbol{m}_t)_{x_t}$$

be the cumulative number of green tokens. Recall that the standard green-token statistic for a balanced partition is $Z_T = 2(S_T - T/2)/\sqrt{T}$. Its expected displacement under watermark injection is therefore

$$\Delta Z_T \triangleq \mathbb{E}_{\mathrm{wm}}[Z_T] - \mathbb{E}_{\mathrm{clean}}[Z_T] = \frac{2}{\sqrt{T}} \sum_{t=1}^{T} \boldsymbol{m}_t^\top \left(\boldsymbol{p}_t^{\mathrm{wm}} - \boldsymbol{p}_t^{\mathrm{clean}}\right). \tag{13}$$

For each generation step t, watermark injection perturbs the clean logits along the mask direction: $\boldsymbol{p}_t^{\mathrm{wm}} = \mathrm{softmax}(\boldsymbol{z}_t^{\mathrm{clean}} + \epsilon \boldsymbol{m}_t)$. A first-order expansion at $\epsilon = 0$ gives

$$\boldsymbol{p}_t^{\mathrm{wm}} - \boldsymbol{p}_t^{\mathrm{clean}} = \epsilon \boldsymbol{J}_t \boldsymbol{m}_t + O(\epsilon^2), \qquad \boldsymbol{J}_t = \mathrm{diag}(\boldsymbol{p}_t^{\mathrm{clean}}) - \boldsymbol{p}_t^{\mathrm{clean}} (\boldsymbol{p}_t^{\mathrm{clean}})^\top. \tag{14}$$

The remainder is uniform over the finite sequence of generation steps. Substitution into Eq. 13 yields

$$\Delta Z_T = \frac{2\epsilon}{\sqrt{T}} \sum_{t=1}^{T} \boldsymbol{m}_t^\top \boldsymbol{J}_t \boldsymbol{m}_t + O(\epsilon^2\sqrt{T}). \tag{15}$$

Since $\boldsymbol{m}_t$ is binary and $P_{G,t} = \boldsymbol{m}_t^\top \boldsymbol{p}_t^{\mathrm{clean}}$, the quadratic form in Eq. 15 reduces to

$$\begin{aligned} \boldsymbol{m}_t^\top \boldsymbol{J}_t \boldsymbol{m}_t &= \boldsymbol{m}_t^\top \mathrm{diag}(\boldsymbol{p}_t^{\mathrm{clean}}) \boldsymbol{m}_t - \left(\boldsymbol{m}_t^\top \boldsymbol{p}_t^{\mathrm{clean}}\right)^2 \\ &= P_{G,t} - P_{G,t}^2 = P_{G,t}(1 - P_{G,t}). \end{aligned} \tag{16}$$

Hence,

$$\begin{aligned} \Delta Z_T &= \frac{2\epsilon}{\sqrt{T}} \sum_{t=1}^{T} P_{G,t}(1 - P_{G,t}) + O(\epsilon^2\sqrt{T}) \\ &= \frac{2\epsilon}{\sqrt{T}} \sum_{t=1}^{T} \left(\frac{1}{4} - \eta_t^2\right) + O(\epsilon^2\sqrt{T}) \\ &= \frac{\epsilon\sqrt{T}}{2} - \frac{2\epsilon}{\sqrt{T}}\|\boldsymbol{\eta}\|_2^2 + O(\epsilon^2\sqrt{T}), \end{aligned} \tag{17}$$

where the second equality follows from $\eta_t = P_{G,t} - 1/2$. This proves the claimed decomposition. Moreover, its first-order term attains $\epsilon\sqrt{T}/2$ if and only if $\|\boldsymbol{\eta}\|_2^2 = 0$, equivalently, $P_{G,t} = 1/2$ for every t. □

F.2 Proof of Theorem 4.2

Theorem 4.2. *At generation step t, order the clean token probabilities as $p_{t,(1)} \geq \cdots \geq p_{t,(|\mathcal{V}|)}$ and define the high-probability set $H_t = \{(1), \ldots, (K)\}$. Let $p_{t,\max} = p_{t,(1)}$ and $\tau_t = p_{t,(K+1)}$, so every token outside H_t has probability at most τ_t. If G_t is constructed using Eqs. 7–8, then*

$$\Pr[G_t \cap H_t = \varnothing] \leq \alpha_t := \frac{\binom{|\mathcal{V}|-K}{m_t}}{\binom{|\mathcal{V}|}{m_t}} \leq \exp\left(-\frac{Km_t}{|\mathcal{V}|}\right), \tag{18}$$

where $m_t = \left\lceil \frac{1-p_{t,\max}}{2\tau_t} \right\rceil$. Equivalently, the green set contains at least one of the top-K tokens with probability at least $1 - \alpha_t$.

Proof. Let $k_t^\star$ be the selected cutoff and suppose that $G_t \cap H_t = \varnothing$. Because G_t is a prefix of the keyed random permutation, every token within the selected prefix belongs to the tail set $\mathcal{V} \setminus H_t$ and therefore has probability at most τ_t.

Let $P_{G,t}$ be the probability mass of this prefix. Consider the first token in the permutation whose inclusion causes the cumulative probability to reach or cross $1/2$. If this token also belongs to the tail, selecting the closer of the prefixes immediately before and after the crossing yields $P_{G,t} \geq 1/2 - \tau_t/2$. If the crossing token belongs to H_t, excluding it can be closer to $1/2$ only if the preceding tail-only prefix has probability mass at least $(1 - p_h)/2$, where p_h is the probability of that token. Since $p_h \leq p_{t,\max}$, this implies

$$P_{G,t} \geq \frac{1 - p_{t,\max}}{2}. \tag{19}$$

The same lower bound also holds in the tail-crossing case because $\tau_t \leq p_{t,\max}$.

Since each selected tail token contributes at most τ_t, a tail-only green set must contain at least $m_t = \lceil (1 - p_{t,\max})/(2\tau_t) \rceil$ tokens. Consequently, the event $G_t \cap H_t = \varnothing$ requires the first m_t positions of the random permutation to contain no token from H_t. Under a uniformly random permutation, the probability of this event is

$$\prod_{j=0}^{m_t-1} \frac{|\mathcal{V}| - K - j}{|\mathcal{V}| - j} = \frac{\binom{|\mathcal{V}|-K}{m_t}}{\binom{|\mathcal{V}|}{m_t}}. \tag{20}$$

Finally, each factor is at most $1 - K/|\mathcal{V}|$, so

$$\frac{\binom{|\mathcal{V}|-K}{m_t}}{\binom{|\mathcal{V}|}{m_t}} \leq \left(1 - \frac{K}{|\mathcal{V}|}\right)^{m_t} \leq \exp\left(-\frac{Km_t}{|\mathcal{V}|}\right), \tag{21}$$

which completes the proof. □