# Watermark under Fire: A Robustness Evaluation of LLM Watermarking

**Anonymous ACL submission**

## Abstract

Various watermarking methods ("watermarkers") have been proposed to identify LLM-generated texts; yet, due to the lack of unified evaluation platforms, many critical questions remain under-explored: i) What are the strengths/limitations of various watermarkers, especially their attack robustness? ii) How do various design choices impact their robustness? iii) How to optimally operate watermarkers in adversarial environments? To fill this gap, we systematize existing LLM watermarkers and watermark removal attacks, mapping out their design spaces. We then develop WATERPARK, a unified platform that integrates 10 state-of-the-art watermarkers and 12 representative attacks. More importantly, by leveraging WATERPARK, we conduct a comprehensive assessment of existing watermarkers, unveiling the impact of various design choices on their attack robustness. We further explore the best practices to operate watermarkers in adversarial environments. We believe our study sheds light on current LLM watermarking techniques while WATERPARK serves as a valuable testbed to facilitate future research.[1]

## 1 Introduction

The recent advances in large language models (LLMs), including GPT (Openai) and Llama (Touvron et al., 2023), have significantly enhanced our capabilities of general-purpose text generation and complex problem-solving, but also raised concerns about misuse through disinformation (Izacard et al., 2022), phishing (Bender et al., 2021), and academic dishonesty (Compilatio). There is thus a pressing need for the capability of identifying LLM-generated content.

A simple approach is to train classifiers to distinguish between LLM- and human-generated texts (Lütkebohle). However, as LLMs improve, this distinction becomes less clear. Watermarking has emerged as an alternative solution, embedding statistical signals ("watermarks") during generation to verify LLM-produced texts. Various watermarking methods ("watermarkers") have been developed (Aaronson and Kirchner; Liu et al., 2023a; Liu et al., 2024; Kirchenbauer et al., 2023a; Hu et al., 2024; Zhao et al., 2023; Kuditipudi et al., 2023), each with unique design choices and desirable properties, raising a set of intriguing questions:

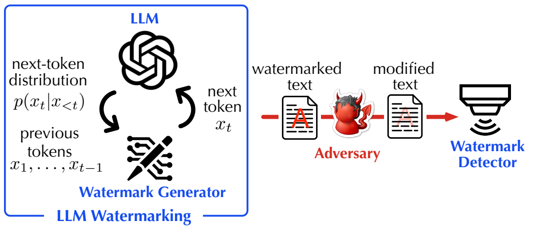


Figure 1: Illustration of LLM watermarking and watermark removal attacks.

RQ1 – What are the strengths and limitations of various watermarkers, especially their robustness against manipulations?

RQ2 – How do different design choices impact the attack robustness of watermarkers?

RQ3 – What are the best practices for operating watermarkers in adversarial environments?

Despite recent efforts to benchmark LLM watermarkers, existing research is limited in addressing these questions. WaterBench (Tu et al., 2023) primarily focuses on watermarking effectiveness; MarkMyWords (Piet et al., 2023) mainly evaluates the robustness of a specific watermarker (Kirchenbauer et al., 2023a); MarkLLM (Pan et al., 2024) focuses on providing a platform to compare watermark detectability, basic robustness, and text quality. Zhao et al. (2024) provides a comprehensive overview of watermarking techniques. Moreover, these studies lack an in-depth analysis of how a watermarker's design choices impact its robustness. Consequently, the aforementioned questions remain largely unexplored.

[1]All the source code and data are publicly available: `https://anonymous.4open.science/r/WaterPark`

Table 1: Conclusions in prior work and WATERPARK (○ – inconsistent; ◐ – partially inconsistent; ● – consistent).

| Previous Conclusion | Refined Conclusion | Explanation | Consist. |
|---|---|---|---|
| UG (Zhao et al., 2023) is not robust due to its context-free design (Piet et al., 2023). | UG shows higher resilience than other watermarkers to paraphrasing attacks. | UG's context-free design ensures consistency between detection and generation, avoiding issues in text-dependent designs. | ○ |
| UPV (Liu et al., 2023a) has a fairly low false positive rate. | UPV is more prone to false positive cases compared to TGRL. | While model-based detection incurs higher uncertainty compared to score-based detection, it fails to offer higher flexibility in countering paraphrasing attacks. | ◐ |
| UPV shows strong robustness to rewriting and outperforms TGRL under paraphrasing attacks. | Both UPV and TGRL struggle against GPT-based paraphrasing attacks. | | ◐ |
| RDF (Kuditipudi et al., 2023) significantly outperforms TGRL against substitution attacks. | RDF shows strong robustness against synonym substitution and other lexical editing attacks. | RDF's distribution-transform strategy is more robust than the distribution-shift strategy against lexical editing. | ● |
| RDF's (Kuditipudi et al., 2023) edit score-based detection is insensitive to local misalignment caused by token insertion (Piet et al., 2023). | RDF (edit score) shows higher resilience against lexical editing attacks compared to GO (plain score) when token length is fixed. However, this advantage diminishes as token length varies. | The edit score-based detection is robust to lexical editing but sensitive to varying token length. | ◐ |
| SIR (Liu et al., 2024) shows strong resilience against paraphrasing attacks. | SIR's effectiveness decreases as the intensity of paraphrasing attacks increases. | SIR's distribution-reweight strategy introduces higher uncertainty, making it sensitive to the intensity of paraphrasing attacks. | ◐ |
| UB (Hu et al., 2024) is more robust than TGRL to substitution attacks. | UB and TGRL are comparably robust to synonym substitution attacks; yet, UB is more vulnerable to paraphrasing attacks. | The distribution-reweight strategy is not superior to the distribution-shift strategy. | ○ |

To bridge this gap, this work conducts a systematic study of state-of-the-art LLM watermarkers, focusing on their attack robustness. We aim to understand how various design choices affect attack resilience and identify best practices for operating watermarkers in adversarial environments.

We develop WATERPARK, the first open-source platform dedicated to evaluating the attack robustness of LLM watermarkers in a unified and comprehensive manner. As of 5/15/2025, WATERPARK integrates 10 state-of-the-art watermarkers, 12 representative watermark removal attacks, and 8 key metrics. Moreover, WATERPARK offers a comprehensive suite of tools for in-depth robustness assessment, including next-token distribution comparisons, attack combination analyses, and what-if scenario evaluations. Leveraging WATERPARK, we empirically evaluate the attack resilience of representative LLM watermarkers, leading to many interesting findings, which challenge the conclusions in prior work, as summarized in Table 1. We also explore how a watermarker's design choices impact its attack robustness, unveiling critical trade-offs between different types of robustness.

## 2 LLM Watermarking

A large language model (LLM) is typically an autoregressive model that generates the next token $x_t$ based on previous tokens $x_{<t} \triangleq x_1, \ldots, x_{t-1}$ (including its prompt), modeled as sampling from a conditional distribution $p(x_t|x_{<t})$.

Conceptually, a watermark is a pattern embedded in a given signal (e.g., text) for identifying the signal's source. In the context of LLMs, watermarks can be used to prove that a given text is LLM-generated (or even generated by specific LLMs). An LLM watermarking method ("watermarker") comprises three components: the LLM, watermarking procedure (generator), and detection procedure (detector), as shown in Figure 1.

Typically, the generator produces watermarked texts iteratively. At each iteration, with access to a secret key $k$, the previous tokens $x_{<t}$, and the LLM's next-token distribution $p(x_t|x_{<t})$, the generator generates a perturbed distribution $\tilde{p}(x_t|x_{<t})$, from which the next token $x_t$ is sampled. Meanwhile, with access to a secret key $k$, the detector determines whether $M$ generates a given text.

The current watermarkers can be categorized based on the information carried by the watermarks (e.g., one-bit versus multi-bit) and their key design choices, including context dependency, generation strategy, and detection method. The key design factors of each category are deferred to §B .

## 3 Platform

### 3.1 Threat Model

One critical property of LLM watermarkers is their robustness against potential attacks. We assume a threat model similar to prior work (Piet et al., 2023; Zhao et al., 2023; Sankar Sadasivan et al., 2023; Kirchenbauer et al., 2023b), as shown in Figure 1. We assume the adversary has access to sample watermarked and non-watermarked texts, but cannot reproduce the watermarking procedure or interact with the detection procedure. The adversary modifies the watermarked text $\tilde{T}$ as the altered text $T'$ such that the following objectives are met: effectiveness – $T'$ evades the watermark detector (i.e., detected as non-watermarked) and quality – $T'$ preserves $\tilde{T}$'s original semantics.

### 3.2 Metrics

**Effectiveness.** At a high level, WATERPARK evaluates the watermark detector's accuracy in detecting watermarked texts mainly using two metrics: true positive rate (TPR) and false positive rate (FPR). TPR measures the fraction of watermarked texts detected as watermarked, while FPR measures the fraction of non-watermarked samples wrongly detected as watermarked. Formally, let $S_+$ and $S'_+$ respectively be the sets of ground-truth and detected watermarked texts ($S_-$ and $S'_-$ correspondingly).

$$\text{TPR} = \frac{|S_+ \cap S'_+|}{|S_+|} \qquad \text{FPR} = \frac{|S_- \cap S'_+|}{|S_-|} \tag{1}$$

In WATERPARK, we plot the receiver operating characteristic (ROC) curve that measures TPR against FPR across varying settings of detection thresholds. In particular, the area under the ROC curve (AUC) evaluates the overall effectiveness of each watermarker. Moreover, to compare two watermarkers under specific settings, we may also measure their TPRs under a fixed FPR (e.g., 1%).

**Fidelity.** To evaluate the impact of watermarking on text quality, WATERPARK employs the following metrics to measure the difference between the original text $T$ and the watermarked text $\tilde{T}$. We employ WER (word error rate), BLEU (Papineni et al., 2002), BERTScore (Zhang et al., 2020) and P-SP (Wieting et al., 2022) as metrics to assess fidelity. Detailed descriptions of these metrics can be found in §C. Note that these metrics are also used to measure the impact of watermark removal attacks on the quality of the modified text $\tilde{T}'$, relative to the watermarked text $\tilde{T}$.

**Robustness.** To evaluate a watermarker's attack resilience, WATERPARK measures the attack's impact on the watermarking effectiveness. Specifically, let $\tilde{T}$ and $\tilde{T}'$ respectively denote the watermarked text before and after the attack. WATERPARK compares the detector's TPR, FPR, and AUC with respect to $\tilde{T}$ and $\tilde{T}'$. Intuitively, a smaller difference indicates that the watermarker is less attack-sensitive.

## 4 Evaluation

We leverage WATERPARK to empirically assess representative LLM watermarkers, focusing on their attack robustness and other related criteria.

### 4.1 Experimental Setting

To comprehensively evaluate the robustness of LLM watermarkers, we assess all methods listed in Table 7 against the full range of attacks defined in Table 8.

We include three LLMs of varying scales(*i.e.* OPT-1.3B (small), LLaMA3-7B-chat (medium), and Qwen2.5-14B-Instruct (large)) and five datasets spanning different styles and domains: C4 (text completion), HC3 (QA), Story Completion (creative writing), Law Stack Exchange (legal), and Paper Conclusion (academic summarization). This design ensures broad and unbiased robustness assessment.

All watermarkers and attacks are implemented following their official documentation. Table 6 summarizes the default parameter configurations used in our experiments. We further validate the fidelity and effectiveness of our implementations in Sections E and F, confirming consistency with results reported in the original works. The evaluation results across various datasets are presented in Table 10. The influence of generation length is discussed in Section E.2, and detailed descriptions of each attack type are provided in Appendix D.

We report the true positive rates (TPRs) of all watermarking methods under each attack type, with the false positive rate (FPR) consistently fixed at 1%.

### 4.2 Robustness – Observational Study

Our comprehensive cross-model and cross-dataset analysis, as detailed in Table 9, demonstrates a consistent ranking of robustness across various models and datasets. This consistency indicates that the inherent algorithmic design of the watermarkers is the primary factor influencing their relative robustness, rather than model-specific or dataset-specific factors.

To provide a thorough evaluation, we selected the Qwen2.5-14B-Instruct model with the Paper

Table 2: Attack resilience of LLM watermarkers. The intensity of red shading indicates higher values, while the intensity of blue shading indicates lower values, with 0.5 serving as the threshold between the two color gradients.

| Watermarker | CLEAN | Linguistic variation | | | Lexical editing | | | | Text-mixing | | | | Paraphrasing | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Contra | Expan | LowCase | Swap | Typo | Syno | Missp | CP1-10 | CP3-10 | CP1-25 | CP3-25 | DP-20 | DP-40 | Trans |
| TGRL | 0.993 | 0.944 | 0.944 | 0.831 | 0.429 | 0.222 | 0.887 | 0.915 | 0.000 | 0.667 | 0.833 | 0.833 | 0.667 | 0.485 | 0.222 |
| UG | 0.993 | 0.857 | 0.833 | 0.976 | 0.929 | 0.930 | 0.976 | 0.738 | 0.857 | 0.714 | 0.714 | 0.991 | 0.976 | 0.877 | 0.921 |
| UPV | 0.400 | 0.400 | 0.400 | 0.430 | 0.080 | 0.000 | 0.380 | 0.360 | 0.000 | 0.000 | 0.000 | 0.000 | 0.200 | 0.080 | 0.240 |
| RDF | 0.999 | 0.998 | 0.996 | 0.996 | 0.976 | 0.979 | 0.991 | 0.993 | 0.872 | 0.893 | 0.932 | 0.978 | 0.905 | 0.738 | 0.978 |
| UB | 0.980 | 1.000 | 1.000 | 0.962 | 0.000 | 0.033 | 0.921 | 1.000 | 0.018 | 0.042 | 0.000 | 0.485 | 0.514 | 0.103 | 0.255 |
| SIR | 0.978 | 0.580 | 0.500 | 0.420 | 0.320 | 0.320 | 0.440 | 0.560 | 0.340 | 0.360 | 0.360 | 0.460 | 0.400 | 0.300 | 0.000 |
| GO | 0.996 | 0.996 | 0.996 | 0.994 | 0.982 | 0.956 | 0.986 | 0.996 | 0.864 | 0.887 | 0.946 | 0.982 | 0.640 | 0.560 | 0.667 |

Conclusion Dataset as our primary evaluation setting. As shown in Table 2, this combination effectively captures the observed robustness patterns, making it a reliable basis for our main results. This choice reflects the broader trends identified in our extensive cross-analysis efforts.

### 4.2.1 Linguistic Variation Attack

This attack perturbs the linguistic features of the watermarked text, without changing its semantics.

i) Overall, most watermarkers show strong resilience against linguistic variation attacks. For instance, TGRL reaches close to 100% TPRs under all three attacks. ii) UPV is the watermarker marginally susceptible to such attacks. This can be explained by the fact that its neural network-based detector primarily depends on implicit textual features, which appear to be sensitive to changes in linguistic characteristics. iii) The SIR shows notably inferior performance. While it employs a neural network to predict token-specific logit perturbations that are designed to be unbiased (no preference for particular tokens) and balanced (with perturbations summing to zero), its results are suboptimal, suggesting that it is fundamentally challenging to make accurate predictions under such perturbations.

### 4.2.2 Lexical Editing Attack

This attack modifies individual words while maintaining the watermarked text's semantics.

i) TRGL, UB, SIR, and UPV tend to be more vulnerable to lexical editing attacks, compared with RDF and UG watermarkers. Intuitively, as text-dependent watermarkers (e.g., TRGL, UB, SIR, and UPV) use previous tokens as the context for the next token in both the watermark generator and detector, the lexical editing thus causes a mismatch between the generator and detector. ii) UB exhibits much higher vulnerability to such attacks, compared with the others. For example, its TPR drops near zero under the typoing and swapping attacks. This may be explained as follows. Recall that, to achieve unbiasedness, UB applies "hard" perturbation on the next-token distribution (e.g., by rejecting half of the vocabulary); thus the disruption to the previous tokens tends to cause a more significant mismatch between the generator and detector, compared with other watermarkers that employ "soft" perturbation (e.g., distribution shift and transform). iii) UPV and SIR exhibit similar vulnerabilities to the previous attack

### 4.2.3 Text-Mixing Attack

This class of attacks "dilutes" the watermark by mixing the watermarked text with non-watermarked text fragments. Here, to evaluate the resilience of different watermarkers against text-mixing attacks, we use the copy-pasting attack (Kirchenbauer et al., 2023b) as the concrete attack, which embeds the watermarked text into the context of non-watermarked, human-written text (generated under the same prompt). We use CP-$n$-$m$ to denote the attack in which the modified text $T'$ consists of $n$ segments of watermarked texts, each of length $m\%$ of $|T'|$, and the rest as non-watermarked text.

i) Overall, most watermarkers experience significant TPR drops, especially under CP-1-10 that only preserves 10% of the watermarked text. ii) Among all the watermarkers, GO and RDF show significantly higher attack resilience. This can be attributed to their sampling strategy: both employ distribution transform, which generates the next token deterministically conditional on a given random permutation. Thus, GO and RDF tend to have stronger per-token signals than the other watermarkers that sample the next token from a given pool (e.g., green list). This observation is consistent with that in §E.2. iii) Meanwhile, UB and UPV are the most vulnerable to the copy-pasting attack, with close to zero TPRs under CP-1-10 and CP-1-25. This can be explained as follows. The model-assisted detector of UPV determines the given text as watermarked based on its aggregated features (rather than per-token statistics), while the injected non-watermarked segments may greatly disrupt such features. Meanwhile, UB applies hard perturbation on the next-token distribution (e.g., by rejecting half of the vocabulary); thus the disruption to the previous tokens causes a significant mismatch between the generator and detector.

#### 4.2.4 Paraphrasing Attack

This class of attacks employs an additional LLM (i.e., paraphraser) to re-write the given watermarked text $\tilde{T}$ (while preserving its semantics) to evade the detector. Here, we consider Dipper (Krishna et al., 2023) as the paraphraser that rewrites $\tilde{T}$ in one shot. Further, we also consider the translating attack uses a translator model Seamless-m4t-v2-large (Communication et al., 2023) that first translates $\tilde{T}$ to French and then translates it back.

i) UPV and UB exhibit higher vulnerability to the Dipper attack, compared to other watermarkers. UPV's TPR drops to around zero under DP-40, which aligns with our analysis in §4.2.3: the paraphrased text segments may greatly disrupt the aggregated textual features for UPV's model-assisted detector, while the disruption to the previous tokens may cause a substantial mismatch between UB's generator and detector, due to its rigid perturbation to the next-token distribution (e.g., rejecting half of the vocabulary). ii) In contrast, RDF and UG are especially robust against the Dipper attack. This can be attributed to their index-dependent and context-free designs, which are less sensitive to the change of previous tokens than text-dependent watermarkers (e.g., TGRL, SIR, and GO). iii) Interestingly, RDF is more vulnerable to the translating attack than the Dipper attack. This susceptibility arises from RDF's detection mechanism, which is highly sensitive to text length variations, a weakness readily exploited by the translating attack's tendency to produce shorter output text.

#### 4.2.5 Fidelity Preservation

Recall that besides their attack effectiveness, another key metric for watermark removal attacks is whether they can preserve the quality of original texts. We thus compare the semantics of watermarked text $\tilde{T}$ and modified text $T'$ using the metrics in §3.2. Figure 2 illustrates the quality preservation of different attacks on GO, with similar results on other watermarkers, with more results in §G.4.


Figure 2: Quality preservation of different attacks.

Observe that most attacks preserve the semantics of the watermarked text $\tilde{T}$ in the modified text $\tilde{T}'$, as measured by BERTScore and P-SP scores. In comparison, the copy-pasting (CP) attack causes more significant text-quality degradation than other attacks, in that it may disrupt the orders of watermarked and non-watermarked segments and insert duplicate segments. Also, note that most attacks emphasize the semantic similarity between $\tilde{T}$ and $\tilde{T}'$ rather than their lexical similarity (as measured by WER and BLEU scores).

### 4.3 Robustness – Causal Analysis

In addition to the observational studies, we further consider conducting causal analysis to understand the impact of individual design choices (e.g., samplers). However, this is challenging in our context because the various components of a watermarker are often highly interconnected and difficult to decouple. To address this challenge, we select two watermarkers with their only difference in the design of one specific component (e.g., context dependency). The results are shown in Figure 3.

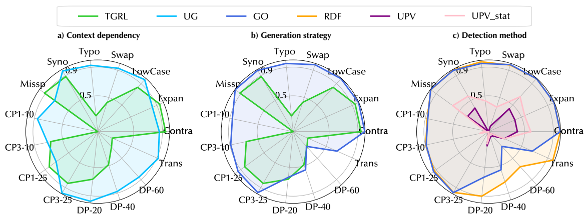


Figure 3: Watermarker robustness to multi-attacks. a) Context dependency: TGRL (text-dependent) and UG (context-free); b) Generation strategy: TGRL (distribution-shift) and GO (distribution-transform); c) Detection method: UPV (Model-based) and UPV$_{stat}$ (Score-based). RDF and GO.

#### 4.3.1 Context Dependency

We select TGRL and UG to represent text-dependent and context-free designs, respectively. TGRL uses the previous $k$ tokens as a randomness seed to divide the vocabulary into red/green lists for the current token, whereas UG uses fixed red/green lists across all positions.

As shown in (a), UG consistently outperforms TGRL under most attack categories. This includes not only strong paraphrasing attacks such as DP-60 and DP-40, but also lexical editing attacks (e.g., Typo, Syno) and text-mixing attacks (e.g., CP1-10, CP3-10). In contrast, TGRL exhibits a noticeable drop in true positive rate (TPR) under these challenging perturbations, especially when token ordering and surface forms are significantly altered.

This performance discrepancy is expected. Attacks like Dipper-style paraphrasing and text mixing disrupt the continuity and order of input tokens, breaking the alignment assumed by TGRL's context-dependent randomness during detection. In contrast, UG's context-free design applies consistent token-level watermarking that remains robust regardless of token rearrangement or local corruption. These results further validate prior observations in UG(Zhao et al., 2023) regarding its robustness to syntax-level transformations.

#### 4.3.2 Generation Strategy

Both TGRL and GO adopt context-dependent watermarking. However, they differ in their generation mechanisms: TGRL employs a distribution-shift approach with soft constraints, whereas GO utilizes a deterministic distribution-transform method on token logits.

As observed in the (b), GO consistently outperforms TGRL under text-mixing attacks, including CP1-10, CP3-10, and CP3-25. While TGRL's true positive rate (TPR) drops significantly under these perturbations, GO maintains high detection performance across all copy-paste scenarios. GO also shows moderate robustness under paraphrasing attacks such as DP-40 and DP-60, where TGRL again underperforms.

The key difference lies in signal strength: GO's transform-based sampling embeds consistent token-level signals resilient to inserted distractors. TGRL's soft sampling is more vulnerable to dilution and misalignment from non-watermarked fragments.

#### 4.3.3 Detection Method

UPV supports both model-based and score-based detection. While the original work suggests model-based detection improves robustness against paraphrasing, our results challenge this claim. As shown in the radar plot (right), both UPV (model-based) and UPV$_{stat}$ (score-based) exhibit near-zero TPR under strong paraphrasing (DP-60, DP-40) and text-mixing (CP3-25), indicating a critical failure in handling high-intensity attacks. Interestingly, UPV$_{stat}$ slightly outperforms UPV under milder perturbations such as misspelling and synonym substitution. This implies that model-based detection introduces additional variance and is more prone to subtle distributional shifts, whereas score-based detection retains better stability in cleaner settings. However, neither variant withstands red-team scenarios effectively, calling into question the overall practicality of UPV's detection mechanism.

Both RDF and GO adopt distribution-transform generation with score-based detection, but differ in scoring criteria. RDF leverages edit-based scoring and demonstrates consistently higher robustness than GO across most attacks. In particular, RDF shows greater resilience under strong paraphrasing (DP-40, DP-60) and text rearrangement (CP3-25), highlighting the benefits of edit-distance awareness in detecting structurally altered text.

## 5 Discussion

Next, we examine the current practices of operating watermarkers in adversarial environments and explore potential improvements.

### 5.1 Specific vs. Generic Detector

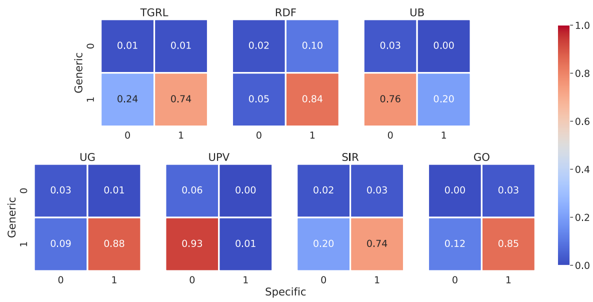


Figure 4: Detection of watermarked texts by watermarker-specific and generic detectors ('1' or '0' indicate that the detector detects the given watermarked text as watermarked or non-watermarked).

For each watermarker, we mainly use its specific detector to detect watermarked texts. Here, we explore a generic, neural network-based detector as an alternative. To this end, we employ a pre-trained RoBERTa model and fine-tune it as a binary classifier using watermarked and non-watermarked texts. We use OPT-1.3B as the underlying LLM and C4 as the reference dataset. The watermarked text is generated by the LLM and the watermarker jointly, whereas the non-watermarked text is produced by ChatGPT using the same prompt to mimic human response. Table 3 shows the TPRs of watermarkers with generic detectors (with FPRs fixed as 1%).

Table 3: TPRs of watermarkers with generic detectors (with FPRs fixed as 1%).

| Watermarker | TGRL | RDF | UB | UG | UPV | SIR | GO |
|---|---|---|---|---|---|---|---|
| TPR | 0.985 | 0.981 | 0.989 | 0.999 | 0.997 | 0.996 | 0.970 |

We compare the attack resilience of watermarker-specific and generic detectors. For each watermarker, we apply the Dipper-40 attack and examine whether two detectors can effectively detect watermarked texts after the paraphrasing attack. Figure 4 depicts the confusion matrices of both detectors.

We have the following findings. i) The specific and generic detectors jointly achieve a high detection rate. Across all the watermarkers, the chance that both detectors fail to detect the watermarked texts (0-0) is below 0.07. ii) For RDF, the generic detector seems less effective than the specific detector, while for UB and UPV, the generic detector outperforms the specific detector by a large margin. Recall that UB and UPV are highly sensitive to the Dipper attack (see §4.2.4. Thus, employing a generic detector alongside a watermarker-specific detector can be an effective strategy for enhancing the security of vulnerable watermarkers. However, note that given the availability of generic detectors, it is also feasible for the adversary to leverage such detectors as an attack checker to adapt their attacks, which we will discuss in §5.2.3.

## 5.2 Advanced Attack

In addition to the previous simple attacks, we investigate the effectiveness of more advanced attacks.

### 5.2.1 Varying Attack Intensity

Table 4: Robustness of Watermarkers against Dipper Attacks with Varying Lexical and Order Changes.

| Water marker | Attack Intensity(Dipper - lexical ($l$) - order ($o$) ) | | | | | |
|---|---|---|---|---|---|---|
| | 20 | 40 | 60 | 40-20 | 60-20 | 60-40 |
| TGRL | 0.952 | 0.858 | 0.550 | 0.746 | 0.406 | 0.546 |
| UG | 0.940 | 0.916 | 0.848 | 0.894 | 0.846 | 0.789 |
| UPV | 0.296 | 0.057 | 0.645 | 0.742 | 0.630 | 0.606 |
| RDF | 0.988 | 0.962 | 0.840 | 0.914 | 0.686 | 0.714 |
| UB | 0.461 | 0.149 | 0.025 | 0.070 | 0.016 | 0.009 |
| SIR | 0.902 | 0.814 | 0.050 | 0.065 | 0.067 | 0.051 |
| GO | 0.971 | 0.847 | 0.500 | 0.785 | 0.378 | 0.460 |

One straightforward way to improve an attack's effectiveness is to increase its intensity, potentially at the cost of other metrics (e.g., text quality). Here, we consider the Dipper attack (Krishna et al., 2023) under varying intensity settings, denoted as DP-$l$-$o$, where lexical change ($l$) indicates that $l$% of the given text is paraphrased and order change ($o$) indicates that $o$% of the text is re-ordered. We compare the watermarkers' resilience under varying attack intensity, as shown in Table 4.

i) As expected, most watermarkers observe TPR drops as the attack intensity increases. ii) Among these, UG demonstrates the most consistent resilience under varying attack intensity. This can be attributed to its context-free design: the same perturbation is applied to the next-token distribution across all the tokens, which is thus immune to the change of previous tokens. iii) Compared with text-dependent watermarkers (e.g., TGRL and GO), an index-dependent watermarker (e.g., RDF) shows stronger resilience, especially under high attack intensity (e.g., DP-60-20), due to its weaker dependency on previous tokens. iv) UPV's performance is inconsistent; it struggles with low-intensity attacks (e.g., DP-20) but shows resilience to high-strength ones (e.g., DP-60). This inconsistency can be attributed to UPV's model-assisted detector and the inherent instability of its neural network, as confirmed by repeated experiments.

### 5.2.2 Combining Simple Attacks

Table 5: Resilience of watermarkers against individual and combined attacks(compared to single attack).

| Water marker | Contra + | | | Swap + | | | Syno + | | |
|---|---|---|---|---|---|---|---|---|---|
| | Typo | Swap | Low | Low | Missp | Typo | Swap | Low | Typo |
| TGRL | -.046 | -.008 | -.002 | -.232 | -.122 | -.648 | -.086 | -.020 | -.268 |
| UG | -.034 | .006 | -.009 | -.066 | -.002 | -.080 | -.008 | -.105 | -.125 |
| UPV | -.192 | -.437 | -.415 | -.505 | -.497 | -.505 | -.548 | -.534 | -.552 |
| RDF | -.021 | .043 | .009 | .013 | .039 | -.267 | -.004 | -.002 | -.132 |
| UB | -.028 | -.016 | -.022 | -.028 | -.026 | -.088 | -.982 | -.258 | -.956 |
| SIR | -.524 | -.858 | -.840 | -.838 | -.866 | -.890 | -.862 | -.856 | -.894 |
| GO | -.068 | .006 | .000 | -.258 | -.092 | -.834 | -.098 | -.002 | -.118 |

We first explore whether combining two attacks improves the attack's effectiveness. Here, considering the most feasible combinations, we only combine two simple attacks from the "weak" linguistic variation and lexical editing attacks. Table 5 illustrates the effectiveness of such combined attacks.

We have a set of interesting observations. i) UPV and SIR, which demonstrate resilience against all simple attacks, are highly vulnerable to all the combined attacks. For instance, the TPR of SIR drastically drops to below 0.3 under the contracting+typoing attack. ii) UB, which is vulnerable to the typoing and swapping attacks, is consequently vulnerable to all the combined attacks that involve typoing or swapping. iii) TGRL and GO, which are robust against all the simple attacks (including typoing and swapping), show significant vulnerability to the typoing+swapping attack. This can be explained by that as typoing and swapping respectively disrupt the tokenization and token-indexing, their combination may substantially amplify such effects. iv) RDF and UG are especially robust against the combined attacks. This can be attributed to their "weaker" context dependencies, which is consistent with the findings in §4.2.4.

### 5.2.3 Adaptive Attack

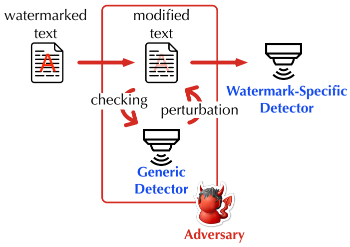


Figure 5: Attacks leveraging surrogate detectors.

Given the availability of generic detectors, it is possible for the adversary to exploit such detectors to adapt their attacks. We consider a scenario as shown in Figure 5: the adversary performs a gradient-based attack (Guo et al., 2021) that iteratively modifies the watermarked text to evade the generic detector, and then forwards the modified text to the target, watermark-specific detector.

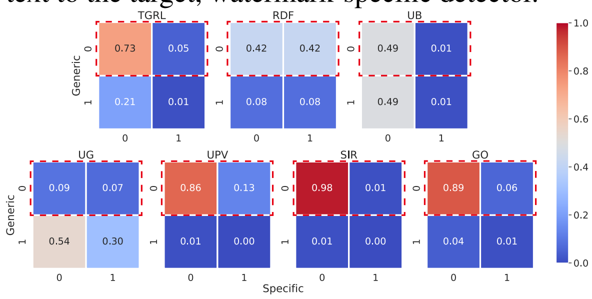


Figure 6: Watermark detection by watermarker-specific and generic detectors on gradient-based attacked samples ('1' or '0' indicate that the detector detects the given watermarked text as watermarked or non-watermarked).

We evaluate this attack's effectiveness using 500 watermarked texts from the C4 dataset. We use GBDA (Guo et al., 2021) as the adversarial attack and limit the steps of perturbations to 100. Compared to Dipper, GBDA makes more substantial textual modifications. The results are summarized in Figure 6. i) Leveraging the surrogate detector significantly improves the attack effectiveness: with the BERTScore and BLEU-1 between $\tilde{T}$ and $\tilde{T}'$ is about 0.764 and 0.188, respectively, slightly lower than the Dipper attack, the detection rates of most watermarkers drop below 10%. ii) Although some samples do not evade the generic detector, they evade the specific detector successfully (e.g., TGRL, UG, and UB). iii) UG and RDF exhibit greater robustness than the other watermarkers. Specifically, for RDF, 42% of the samples evade the generic detector but are still detected by the RDF-specific detector. This superior robustness is likely due to their weaker context dependencies.

### 5.2.4 Leveraging Expert LLMs

We investigate the impact of highly capable LLMs in paraphrasing attacks by using ChatGPT to rewrite watermarked texts. For each watermarker, we randomly sample 100 successfully detected watermarked texts and query ChatGPT with prompt: "Paraphrase the following text and keep length similar". To assess the effect of paraphrasing strength, we also apply up to 5 rounds of iterative paraphrasing, following the strategy in (Zhang et al., 2023), which originally tested up to 300 rounds using T5. We focus on practical, resource-limited attackers, while their work (Zhang et al., 2023) explores worst-case robustness. Results (Figure 7) show that: i) A single round of ChatGPT paraphrasing reduces detection rates of all watermarkers below 0.3, demonstrating that expert LLMs pose a strong threat. However, as noted in §3.1, such powerful LLMs may fall outside our target threat model. ii) For more resilient methods (e.g., UG and GO), detection rates still drop below 0.15 after a few paraphrasing rounds, confirming similar trends observed in (Zhang et al., 2023).

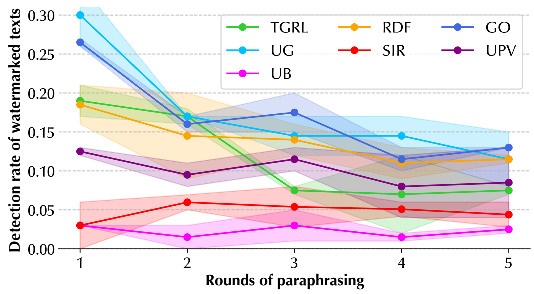


Figure 7: Effectiveness of paraphrasing attacks with ChatGPT as the paraphraser.

## 6 Conclusion

In this paper, we systematize the existing LLM watermarkers and watermark removal attacks, mapping out the design space for various watermarking and attacking techniques. We then design and implement WATERPARK, the first open-source platform devoted to assessing the attack robustness of LLM watermarkers in a unified and comprehensive manner. Leveraging WATERPARK, we conduct a systematic evaluation of the robustness of existing watermarkers, addressing unresolved questions, revealing design trade-offs, and identifying opportunities for further improvement. Our findings shed light on the current LLM watermarking techniques, while WATERPARK serves as a valuable benchmark aiding future research.

## Limitations

We now examine the limitations of our study and identify promising directions for future research.

**Watermarkers.** This study primarily focuses on training-free, pre-generation watermarking, which is applicable to any given LLM and offers flexible control over multiple criteria (e.g., quality, effectiveness, and robustness). While we have included peer-reviewed watermarking methods to the best of our ability, many methods, such as training-time watermarking and post-generation watermarking, are not covered in this paper.

**Threat Model.** Notably, our evaluation is based on the following assumptions about adversary capabilities: i) the adversary can modify the watermarked text in a computationally efficient manner (e.g., synonym substitution), ii) the adversary uses LLMs less capable than the target LLM, and/or iii) the adversary can train a detector using given watermarked and non-watermarked texts. We argue that these assumptions are realistic and practical. Otherwise, using more capable LLMs could easily generate high-quality, non-watermarked texts to evade detection, while launching computationally expensive attacks would significantly increase the adversary's cost. Future research directions include exploring alternative threat models to expand the scope of our analysis.

**Causal Analysis of Design Choices.** While our watermarker taxonomy provides a clear classification framework, we focus on analyzing the holistic design choices of each watermarked (rather than individual design modules) to elucidate the underlying factors driving our experimental observations. This limitation stems from the relatively small number of watermarkers within each taxonomic category, precluding definitive conclusions about the vulnerability of specific taxonomic elements based on current experimental evidence.

A more granular assessment of specific design choices would ideally involve comprehensive ablation studies, systematically modifying individual design elements and comparing their performance against baseline configurations. However, this approach faces significant practical challenges due to the intricate inter-dependencies and tight coupling among watermarker components. To partially address this challenge, in §4.3, we strategically compare two watermarkers that differ only in the design of one specific component (e.g., context dependency), enabling the examination of the effects of specific design variations. We consider a more systematic causal analysis as future research.

**Parameter Tuning.** Our comparative analysis employs AUC curves based on default parameter settings across watermarkers. For benchmarking purposes, we evaluate True Positive Rate (TPR) at a fixed False Positive Rate (FPR) of 1%, aligning with established practices in comparative studies. This standardized approach enables systematic assessment of watermark detection effectiveness while maintaining consistent false positive control. While this approach provides a pragmatic framework for comparative evaluation, it underscores the importance of future research into comprehensive parameter optimization. Such studies could reveal the full performance potential of each method and yield deeper insights into their relative strengths and limitations.

## Ethics Consideration

In this paper, we conduct a systematic study of state-of-the-art LLM watermarkers, focusing on their robustness against watermark removal attacks. We aim to understand how various design choices affect watermarkers' resilience and identify best practices for operating watermarkers in adversarial environments.

**Stakeholder Considerations.** The primary stakeholders affected by this research include users and developers of LLM watermarkers, as well as the broader community relying on these techniques. By identifying the strengths/limitations of various watermarkers, our work could influence the perceived reliability and deployment of LLM watermarkers in critical applications. Conversely, exposing these vulnerabilities allows for the improvement of the current techniques, ultimately contributing to more secure and robust systems.

**Potential Harms and Benefits.** Exposing vulnerabilities in widely adopted security mechanisms can yield contrasting outcomes. Initially, it may erode trust in LLM watermarkers as reliable safeguards, potentially leading to their underuse and leaving systems exposed to unchecked risks. However, by illuminating these weaknesses, our research catalyzes the development of more robust techniques and fosters a nuanced comprehension of their constraints, ultimately fortifying the long-term security ecosystem.

**Future Research and Mitigation.** Our study has identified several potential countermeasures to address the vulnerabilities we uncovered. These recommendations are designed to steer future research and assist developers in bolstering the security of LLM watermarkers.

## References


Scott Aaronson and Hendrik Kirchner. Watermarking gpt outputs. https://www.scottaaronson.com/talks/watermark.ppt.

Sahar Abdelnabi and Mario Fritz. 2021. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 121–140. IEEE.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT)*.

Miranda Christ, Sam Gunn, and Or Zamir. 2023. Undetectable watermarks for language models. Cryptology ePrint Archive, Paper 2023/763.

Seamless Communication, Loïc Barrault, Yu-An Chung, Mariano Cora Meglioli, David Dale, Ning Dong, Paul-Ambroise Duquenne, Hady Elsahar, Hongyu Gong, Kevin Heffernan, John Hoffman, Christopher Klaiber, Pengwei Li, Daniel Licht, Jean Maillard, Alice Rakotoarison, Kaushik Ram Sadagopan, Guillaume Wenzek, Ethan Ye, Bapi Akula, Peng-Jen Chen, Naji El Hachem, Brian Ellis, Gabriel Mejia Gonzalez, Justin Haaheim, Prangthip Hansanti, Russ Howes, Bernie Huang, Min-Jae Hwang, Hirofumi Inaguma, Somya Jain, Elahe Kalbassi, Amanda Kallet, Ilia Kulikov, Janice Lam, Daniel Li, Xutai Ma, Ruslan Mavlyutov, Benjamin Peloquin, Mohamed Ramadan, Abinesh Ramakrishnan, Anna Sun, Kevin Tran, Tuan Tran, Igor Tufanov, Vish Vogeti, Carleigh Wood, Yilin Yang, Bokai Yu, Pierre Andrews, Can Balioglu, Marta R. Costa-jussà, Onur Celebi, Maha Elbayad, Cynthia Gao, Francisco Guzmán, Justine Kao, Ann Lee, Alexandre Mourachko, Juan Pino, Sravya Popuri, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, Paden Tomasello, Changhan Wang, Jeff Wang, and Skyler Wang. 2023. Seamlessm4t: Massively multilingual & multimodal machine translation.

Compilatio. Cheating in the age of chatgpt: findings and solutions for preserving academic integrity. https://www.compilatio.net/en/blog/cheating-chatgpt.

Prithiviraj Damodaran. 2021. Parrot: Paraphrase generation for NLU.

Jacoband Devlin, Ming-Weiand Chang, Kentonand Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Euclaise. Writingprompts dataset. https://huggingface.co/datasets/euclaise/writingprompts. Accessed: 2025-01-29.

Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. 2023. Publicly detectable watermarking for language models. *arXiv preprint arXiv:2310.18491*.

Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. 2023. Publicly detectable watermarking for language models. *ArXiv e-prints*.

Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. 2023. Three bricks to consolidate watermarks for large language models. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE.

Eva Giboulot and Furon Teddy. 2024. Watermax: breaking the llm watermark detectability-robustness-quality trade-off. *arXiv preprint arXiv:2403.04808*.

Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. 2021. Gradient-based adversarial attacks against text transformers. *ArXiv e-prints*.

Abe Bohan Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. 2023. Semstamp: A semantic watermark with paraphrastic robustness for text generation. *arXiv preprint arXiv:2310.03991*.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2024. Unbiased watermark for large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Atlas: Few-shot learning with retrieval augmented language models. *ArXiv e-prints*.

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

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023a. A watermark for large language models. In *Proceedings of the IEEE Conference on Machine Learning (ICML)*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2023b. On the reliability of watermarks for large language models. *ArXiv e-prints*.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2023. Robust distortion-free watermarks for language models. *ArXiv e-prints*.

John Kudo, Takuand Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. 2023. Who wrote this code? watermarking for code generation. *arXiv preprint arXiv:2305.15060*.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

Jonathan Li. Law stack exchange dataset. https://huggingface.co/datasets/jonathanli/law-stack-exchange. Accessed: 2025-01-29.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Alexander Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew Arad Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue WANG, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Andrew Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. Holistic evaluation of language models. *Transactions on Machine Learning Research*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shu'ang Li, Lijie Wen, Irwin King, and Philip S Yu. 2023a. An unforgeable publicly verifiable watermark for large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2024. A semantic invariant robust watermark for large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Lijie Wen, Irwin King, and Philip S Yu. 2023b. A survey of text watermarking in the era of large language models. *arXiv preprint arXiv:2312.07913*.

Yixin Liu, Hongsheng Hu, Xuyun Zhang, and Lichao Sun. 2023c. Watermarking text data on large language models for dataset copyright protection. *arXiv preprint arXiv:2305.13257*.

Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. 2021. Sok: How robust is image classification deep neural network watermarking? In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*.

Ingo Lütkebohle. Gptzero. https://gptzero.me/.

George A. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature.

Travis Munyer and Xin Zhong. 2023. Deeptextmark: Deep learning based text watermarking for detection of large language model generated text. *arXiv preprint arXiv:2305.05773*.

Openai. Openai chatgpt blog. https://openai.com/blog/chatgpt.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Julien Piet, Chawin Sitawarin, Vivian Fang, Norman Mu, and David Wagner. 2023. Mark my words: Analyzing and evaluating language model watermarks. *ArXiv e-prints*.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. MAUVE: Measuring the gap between neural text and human text using divergence frontiers. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*.

Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. 2023. A robust semantics-based watermark for large language model against paraphrasing. *arXiv preprint arXiv:2311.08721*.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected? *ArXiv e-prints*.

Ryoma Sato, Yuki Takezawa, Han Bao, Kenta Niwa, and Makoto Yamada. 2023. Embarrassingly simple text watermarks. *arXiv preprint arXiv:2310.08920*.

Karanpartap Singh and James Zou. 2023. New evaluation metrics capture quality degradation due to llm watermarking. *arXiv preprint arXiv:2312.02382*.

Zhensu Sun, Xiaoning Du, Fu Song, and Li Li. 2023. Codemark: Imperceptible watermarking for code datasets against neural code completion models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1561–1572.

Zhensu Sun, Xiaoning Du, Fu Song, Mingze Ni, and Li Li. 2022. Coprotector: Protect open-source code against unauthorized training usage with data poisoning. In *Proceedings of the ACM Web Conference 2022*, pages 652–660.

Ruixiang Tang, Qizhang Feng, Ninghao Liu, Fan Yang, and Xia Hu. 2023. Did you train on my dataset? towards public dataset protection with cleanlabel backdoor watermarking. *ACM SIGKDD Explorations Newsletter*, 25(1):43–53.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv e-prints*.

Shangqing Tu, Yuliang Sun, Yushi Bai, Jifan Yu, Lei Hou, and Juanzi Li. 2023. Waterbench: Towards holistic evaluation of watermarks for large language models. *arXiv preprint arXiv:2311.07138*.

Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. 2024. Towards codable text watermarking for large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Johnand Wieting, Kevinand Gimpel, Grahamand Neubig, and Taylor Berg-kirkpatrick. 2022. Paraphrastic representations at scale. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yihan Wu, Zhengmian Hu, Junfeng Guo, Hongyang Zhang, and Heng Huang. 2024. A resilient and accessible distribution-preserving watermark for large language models. In *Forty-first International Conference on Machine Learning*.

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

Xi Yang, Kejiang Chen, Weiming Zhang, Chang Liu, Yuang Qi, Jie Zhang, Han Fang, and Nenghai Yu. 2023. Watermarking text generated by black-box language models. *arXiv preprint arXiv:2305.08883*.

KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. 2023. Advancing beyond identification: Multi-bit watermark for language models. *arXiv preprint arXiv:2308.00221*.

KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. 2024. Advancing beyond identification: Multi-bit watermark for language models. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Hanlin Zhang, Benjamin L. Edelman, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, and Boaz Barak. 2023. Watermarks in the sand: Impossibility of strong watermarking for generative models. *ArXiv e-prints*.

Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar. 2023. Remark-llm: A robust and efficient watermarking framework for generative large language models. *ArXiv e-prints*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. 2023. Provable robust watermarking for ai-generated text. *ArXiv e-prints*.

Xuandong Zhao, Sam Gunn, Miranda Christ, Jaiden Fairoze, Andres Fabrega, Nicholas Carlini, Sanjam Garg, Sanghyun Hong, Milad Nasr, Florian Tramer, et al. 2024. Sok: Watermarking for ai-generated content. *arXiv preprint arXiv:2411.18479*.

## A Parameter Setting

Table 6 lists the default setting of the parameters of each watermarker in our evaluation. Note that $\gamma$ and $\delta$ are the gamma and delta used in the watermarker, and $n$ denotes the window size.

| Watermarker | Parameter | Setting |
|---|---|---|
| TGRL | $\gamma$ | 0.25 |
| | $\delta$ | 2.0 |
| UG | $\gamma$ | 0.5 |
| | $\delta$ | 2.0 |
| CTWL | $\delta$ | 1.5 |
| | $n$ | 10 |
| | message code length | 20 |
| | encode ratio | 10.0 |
| | message strategy | vanilla |
| UPV | $\gamma$ | 0.5 |
| | $\delta$ | 2.0 |
| | $n$ | 3 |
| | bit number | 16 |
| | layers | 9 |
| UB | watermark type | delta |
| SIR | $\delta$ | 1.0 |
| | $n$ | 10 |
| | watermark type | context |
| GO | $n$ | 3 |
| RDF | number of random sequences | 50 |

Table 6: Default parameter setting of watermarkers.

## B Taxonomies

We present a taxonomy of LLM watermarkers, as summarized in Table 7. The current watermarkers can be categorized based on the information carried by the watermarks (e.g., one-bit versus multi-bit) and their key design choices, including context dependency, generation strategy, and detection method. Next, we mainly focus on the key design factors.

### B.1 Context Dependency

The watermarker applies a perturbation $\Delta_t$ to the LLM's next-token distribution $p(x_t|x_{<t})$ as $\tilde{p}(x_t|x_{<t}) = p(x_t|x_{<t}) + \Delta_t$, from which the next token $x_t$ is generated. The perturbation $\Delta_t$ often depends on the given context. Three types of context dependencies are typically used in the existing watermarkers.

**Index-dependent watermark.** The watermarker produces a pseudo-random number $r_t$ by applying a keyed hash function

$$r_t = f_k(t), \tag{2}$$

that only depends on the index $t$ of the next token; $r_t$ is then used to generate the perturbation $\Delta_t$ (Kuditipudi et al., 2023).

**Text-dependent watermark.** The watermarker considers the previous tokens $x_{<t}$ as the context window to generate $\Delta_t$ (Kirchenbauer et al., 2023a; Aaronson and Kirchner; Hu et al., 2024). For instance, GO (Aaronson and Kirchner) computes the hash of the concatenation of previous $w$ tokens as $r_t$:

$$r_t = f_k(x_{t-w}\| \dots \|x_{t-1}), \tag{3}$$

while TGRL (Kirchenbauer et al., 2023a) suggests using the min hash of the previous token:

$$r_t = \min(f_k(x_{t-w}), \dots, f_k(x_{t-1})) \tag{4}$$

Similarly, UB (Hu et al., 2024) concatenates the previous $w$ tokens to generate the context code for reweighting the logits. UPV (Liu et al., 2023a) and SIR (Liu et al., 2024) use neural networks to generate $\Delta_t$ based on the previous tokens.

**Context-free watermark.** The watermarker applies a universal perturbation $\Delta_t$ across the next-token distributions of all the tokens without considering their contexts (Zhao et al., 2023).

### B.2 Generation Strategy

By applying the perturbation $\Delta_t$ to the LLM's next-token distribution $p(x_t|x_{<t})$, the watermarker generates the new distribution $\tilde{p}(x_t|x_{<t})$ to sample the next token $x_t$. The existing sampling strategies can be categorized as follows.

**Distribution shift.** TGRL (Kirchenbauer et al., 2023a) modifies $p(x_t|x_{<t})$ by adding a shift $\delta$ to the logits of "green-list" tokens (the remaining as "red-list" tokens) as the modified distribution $\tilde{p}(x_t|x_{<t})$. A token $x$ is considered as green listed if $\pi_{r_t}(x) < \gamma d$ where $\pi_{r_t}$ is a permutation seeded by $r_t$, $\gamma$ is a parameter to control the size of the green list, and $d$ is the number of vocabulary size. Similarly, UG (Zhao et al., 2023) uses a fixed red-green split over the vocabulary, showing greater robustness than TGRL against edit-distance-bounded attacks due to its "hard" split. UPV (Liu et al., 2023a) selects the top-$k$ tokens from $p(x_t|x_{<t})$, applies a neural network to predict the green-list tokens from these tokens, and adds $\delta$ to the logits of green-list tokens. Unlike the other strategies, the distribution-shift strategy preserves the diversity of generated tokens; however, it can not be made indistinguishable, as $\tilde{p}(x_t|x_{<t})$ and $p(x_t|x_{<t})$ are inherently distinguishable.

**Distribution reweight.** Similar to distribution shift, this strategy alters the next-token distribution but uniquely perturbs the logit of each token. For instance, SIR (Liu et al., 2024) trains a neural network to predict the perturbation to logit of

| Water marker | Infor mation | Context Dependency | | | Generation Strategy | | | Detection Method | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Index-dep. | Text-dep. | Context-free | Dist.-shift | Dist.-reweight | Dist.-transform | Score- | Diff.- | Model-based |
| TGRL | one-bit | | ✔ | | ✔ | | | ✔ | | |
| UG | | | | ✔ | ✔ | | | ✔ | | |
| UPV | | | ✔ | | ✔ | | | | | ✔ |
| SIR | | | ✔ | | | ✔ | | ✔ | | |
| RDF | | ✔ | | | | | ✔ | ✔ | | |
| UB | | | ✔ | | | ✔ | | | ✔ | |
| DIP | | | ✔ | | | ✔ | | ✔ | | |
| GO | | | ✔ | | | | ✔ | ✔ | | |
| CTWL | multi-bit | | ✔ | | | ✔ | | | | ✔ |
| MPAC | | | ✔ | | | | ✔ | ✔ | | |

Table 7: A taxonomy of LLM watermarkers. References: TGRL (Kirchenbauer et al., 2023a), UG (Zhao et al., 2023), UPV (Liu et al., 2023a), SIR (Liu et al., 2024), RDF (Kuditipudi et al., 2023), UB (Hu et al., 2024), DIP (Wu et al., 2024), GO (Aaronson and Kirchner), CTWL (Wang et al., 2024), MPAC (Yoo et al., 2024).

each token, which is unbiased (no preference over specific tokens) and balanced (with total perturbation summing up to 0). UB (Hu et al., 2024) advocates unbiased watermarking such that the expectation of the reweighted distribution agrees with the original distribution. It proposes two reweighting schemes: $\delta$-reweighting uniformly samples a token from $p(x_t|x_{<t})$ and changes its probability to 1; $\gamma$-reweighting shuffles all the tokens, rejects the first half, and double the probabilities of the remaining half. DIP (Wu et al., 2024) also uses a distribution-reweight generation strategy similar to UB but does not need to access the LM during detection.

**Distribution transform.** Another line of watermarkers apply randomized transform on $p(x_t|x_{<t})$ to sample $x_t$. For instance, RDF-EXP (Kuditipudi et al., 2023) and GO (Aaronson and Kirchner) use the Gumbel-max trick and apply the exponential transform. Let $p(x_t|x_{<t}) = \{p_i\}_{i=1}^d$ be the distribution over the next token $x_t$. Then $x_t$ is sampled as:

$$x_t = \arg\max_{i \le d} r_i^{1/p_i} \tag{5}$$

where $r_i$ is generated by the pseudo-random functions in Eq. 2 and Eq. 3. Similarly, RDF-IST (Kuditipudi et al., 2023) applies inverse transform over $p(x_t|x_{<t})$. With $\pi_k$ as a random permutation seeded by the secret key $k$, the next token $x_t$ is selected as:

$$x_t = \pi_k\left(\min_{j \le d} \sum_{i=1}^{j} p_{\pi_k(i)} \ge r_t\right) \tag{6}$$

which is the smallest index in the inverse permutation such that the CDF of the next token distribution exceeds $r_t$. The distribution-transform strategy does not alter the next-token distribution, thus preserving the original text distribution (i.e., indistinguishability).

### B.3 Detection Method

The watermarker's detector determines whether a given text $T = (x_1, \ldots, x_n)$ is watermarked or not. Next, we categorize the existing detection methods as follows.

**Score-based detection.** The detector computes the random value $r_i$ for each position, the per-token statistics $s(x_i, r_i)$, and a score over such statistics, which is then subjected to a one-tailed statistical test to determine whether the text is watermarked. The per-token statistics vary with the concrete generators. For instance, GO (Aaronson and Kirchner) defines $s(x_i, r_i) = -\log(1 - h_{r_i}(x_i))$, while TGRL (Kirchenbauer et al., 2023a) defines $s(x_i, r_i) = 1$ if $x_i$ is in the green list and 0 otherwise. One simple way to aggregate the per-token statistics is to compute their sum (Aaronson and Kirchner; Kirchenbauer et al., 2023a; Zhao et al., 2023; Liu et al., 2024):

$$S = \sum_{i=1}^{n} s(x_i, r_i) \tag{7}$$

However, as the random values may be misaligned with the tokens (e.g., due to editing), a more robust way is to compute the alignment score (e.g., edit score (Kuditipudi et al., 2023; Wang et al., 2024)):

$$\begin{aligned} & S = s^\psi(n, n) \\ & \text{s.t. } s^\psi(i, j) = \min \begin{cases} s^\psi(i-1, j-1) + s(x_i, r_j), \\ s^\psi(i, j-1) + \psi, \\ s^\psi(i-1, j) + \psi \end{cases} \end{aligned} \tag{8}$$

where $\psi$ is the "edit-cost" parameter.

**Differential-based detection.** This line of detectors also relies on the score of a given text. However, the score is computed by comparing the given text with the non-watermarked text generated by the same LLM. For instance, UB (Hu et al., 2024) computes the log-likelihood ratio (LLR) score:

$$s(i) = \log \frac{\tilde{p}(x_i|x_{<i})}{p(x_i|x_{<i})} \quad (9)$$

and its more robust maximin variant. However, note that these detectors naturally require accessing the original LLM, which is not always feasible.

**Model-assisted detection.** Instead of computing the per-token statistics, which are often subject to watermark removal attacks, one may also train a model to predict whether the given text is watermarked. UPV (Liu et al., 2023a) trains a neural network, which shares the same embedding layers with its generator, to detect watermarked texts. Similarly, one may develop a generic detector by training it to distinguish watermarked and non-watermarked texts. We explore this option in §4.

## C Fidelity Metrics

In this paper, to evaluate the impact of watermarking on text quality, we employs the following metrics to measure the difference between the original text $T$ and the watermarked text $\tilde{T}$. Below, we provide a detailed description of each metric.

WER (word error rate) measures the percentage of mismatched tokens between $\tilde{T}$ and $T$ relative to the total number of tokens in $T$. This is a lexical metric used to measure the fraction of tokens that have been modified.

BLEU (Papineni et al., 2002) measures the lexical similarity of $T$ and $\tilde{T}$ by calculating the proportion of $n$-grams matched in $\tilde{T}$ and $T$.

BERTScore (Zhang et al., 2020) measures the token-level similarity of $T$ and $\tilde{T}$ by leveraging the pre-trained contextual embeddings from BERT (Devlin et al., 2019) and matching tokens in $T$ and $\tilde{T}$ using cosine similarity.

P-SP (Wieting et al., 2022) evaluates the semantic similarity of $T$ and $\tilde{T}$ using the cosine similarity of their encodings. In particular, the encoding of $T$ (or $\tilde{T}$) is calculated by averaging the embeddings of its subword units generated by SentencePiece (Kudo, 2018).

MAUVE (Pillutla et al., 2021) compares the distributions of $T$ and $\tilde{T}$ by computing an information divergence curve within a quantized embedding space. The area under this divergence curve provides a scalar summary of the trade-off between Type I ($T$ places high mass in areas where $\tilde{T}$ has low mass) and Type II (vice versa) errors.

Note that these metrics are also used to measure the impact of watermark removal attacks on the quality of the modified text $\tilde{T}'$, relative to the watermarked text $\tilde{T}$.

## D Details of Watermark Removal Attacks

| Attack | Category | Resource |
|---|---|---|
| Lowercasing | Linguistic variation | / |
| Contracting | | / |
| Expanding | | / |
| Misspelling | Lexical editing | Common misspellings |
| Typoing | | / |
| Synonymizing | | WordNet |
| Swapping | | / |
| Copy-pasting | Text-mixing | Non-watermarked text |
| Deep-paraphrasing | Paraphrasing | LLM-based paraphraser |
| Translating | | LLM-based translator |
| Black-box adversarial attack | | Generic detector |

Table 8: A taxonomy of watermark removal attacks.

We also present a taxonomy of existing watermark removal attacks according to their underlying perturbation and required resources, as summarized in Table 8.

**Linguistic variation attack.** This class of attacks perturb the linguistic features of the watermarked text, without changing its semantics. We consider the set of perturbations in HELM (Liang et al., 2023), which simulate natural linguistic variations encountered in human interactions with text typing interfaces: i) *Lowercasing* converts all the words to lower-cases, which potentially affects the interpretation of proper nouns or emphases. ii) *Contracting/expanding* replaces phrases with their contracted/expanded forms (e.g., "I am" to "I'm" and vice versa), which may impact the tokenizer.

**Lexical editing attack.** This class of attacks modifies individual words, aiming to maintain the original text's semantics. Specifically, we consider the following editing operations: i) *Misspelling*, similar to text-bugger (Li et al., 2019), replaces words with their common misspellings (plural forms also considered); ii) *Typoing* replaces certain letters in a word with others; iii) *Synonymizing* replaces words with their synonyms using WordNet (Miller, 1995); and iv) *Swapping* randomly exchanges the positions of two words within the text, which alters the text structure while potentially preserving the overall semantics.

**Text-mixing attack.** This class of attacks aims to "dilute" the watermark by mixing the watermarked text with non-watermarked text fragments. Specifically, the *copy-pasting* attack (Kirchenbauer et al., 2023b) embeds the watermarked text into the context of non-watermarked, human-written text. Note that the influence of non-watermarked text can be controlled by setting the fractions of watermarked and non-watermarked text fragments (i.e., the mixing weights).

**Paraphrasing attack.** This class of attacks relies on an additional LLM (i.e., paraphraser) to re-write the given watermarked text to evade the detector. For instance, a light paraphraser (Sankar Sadasivan et al., 2023) (e.g., T5-based paraphraser (Damodaran, 2021)) can paraphrase the watermarked text sentence-by-sentence, while a more capable paraphraser (e.g., DIPPER (Krishna et al., 2023)) paraphrases the watermarked text in one-shot, also enabling to control lexical diversity and token order diversity.

Similar to paraphrasing, the *translating* attack uses a translator LLM (e.g., Seamless (Communication et al., 2023)) to cycle the watermarked text through multiple languages (e.g., from English to French and back to English). This process can significantly alter the sentence structure and phrasing.

In this class of attacks, we assume the adversary has access to a generic detector that is trained to distinguish watermarked and non-watermarked texts. The adversary then perturbs the watermarked text based on this surrogate detector. We explore this attack type in §5.

# E Watermark Effectiveness

We first evaluate the effectiveness of different watermarkers. Following prior work (Kirchenbauer et al., 2023a; Piet et al., 2023; Kuditipudi et al., 2023), for each watermarker, we sample 1,000 prompts and use the LLM in combination with the watermarker to generate the watermarked texts; meanwhile, we select human responses to the same prompts as the non-watermarked texts. We then measure the accuracy of the watermarker's detector in distinguishing the watermarked and non-watermarked texts.

## E.1 Overall Effectiveness

We measure the overall effectiveness of each method through the lens of the ROC curve. Figure 8 (a-d) summarizes the overall effectiveness of existing watermarkers across different models and datasets. We have the following interesting observations.

Most watermarkers are highly effective in generating and subsequently detecting watermarked texts on OPT-1.3B as shown in Figure 8 (a-b). For instance, RDF, UB, and GO all attain AUC scores above 0.99 over both C4 and HC3. Recall that C4 and HC3 represent the text completion and question-answering tasks respectively. The observation indicates that most watermarkers tend to be highly effective for relatively less capable LLMs such as OPT-1.3B, while the concrete dataset/task have a limited impact on their performance. We further validate this hypothesis under the setting of a fixed FPR. As shown in Figure 8 (e-f), we fix the FPRs of all the methods to be 0.01 and measure their TPRs. Observe that all the methods achieve above 0.9 TPR, with a marginal difference across C4 and HC3.

Meanwhile, most methods observe marginal performance drops on Llama3-7B, as shown in Figure 8 (c-d). For instance, compared with its performance on OPT-1.3B, the AUC of SIR drops by 0.11 and 0.08 on C4 and HC3 respectively. This observation aligns with previous research (Kuditipudi et al., 2023), indicating that watermarkers are more effective on OPT compared with Llama3. This phenomenon can be partly understood through the following explanation. In contrast of less capable LLMs (e.g., OPT-1.3B), Llama3-7B typically produces texts of lower perplexity. Since most watermarkers inject watermarks by slightly altering the next-token distribution, the lower perplexity in Llama3's outputs hampers the effectiveness of such perturbations. Moreover, it is observed that the performance of various methods varies significantly across different datasets. For instance, UG's AUC differs by 0.28 between C4 and HC3, while UB's AUC differs by 0.11. This observation is further supported by the TPR measures at fixed FPRs (fixed as 1%), as shown in Figure 8 (e-f). Our findings suggest that the concrete dataset/task tends to have a larger impact on watermarkers over more capable LLMs.

## E.2 Impact of Text Length

We evaluate how the (non-)watermarked text length (i.e., the number of tokens) impacts the performance of different methods. Specifically, we measure the TPR of each method with its FPR fixed as 1%. In the following, we set OPT-1.3B and C4 as the default LLM and dataset. Figure 9 summarizes the results.

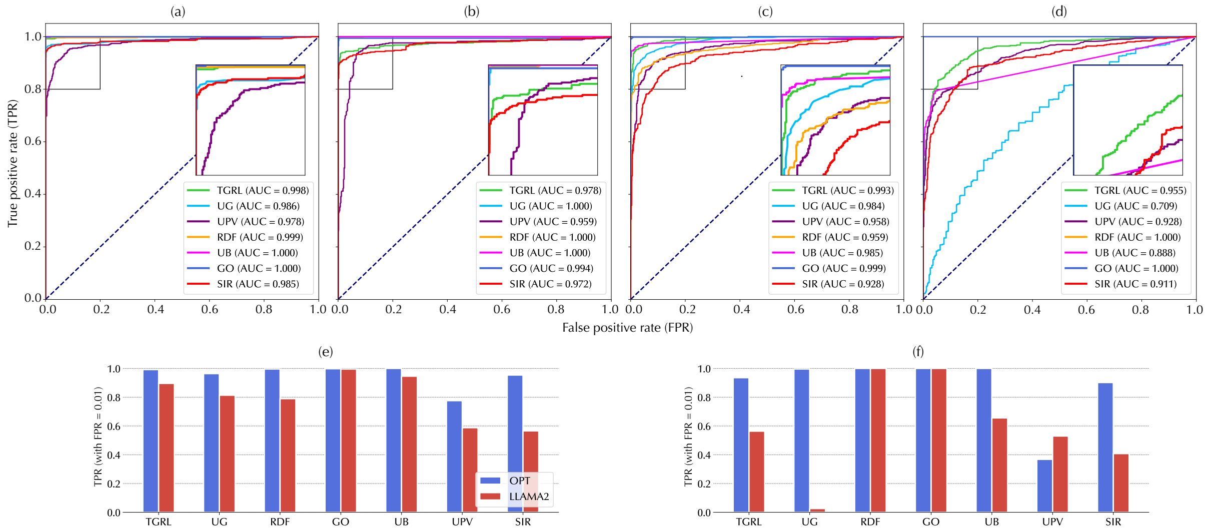


Figure 8: Overall effectiveness of different watermarkers in generating and detecting watermarked texts: ROC of (a) OPT-C4, (b) OPT-HC3, (c) Llama3-C4, and (d) Llama3-HC3; TPR (with FPR fixed as 0.01) on (e) C4 and (f) HC3.

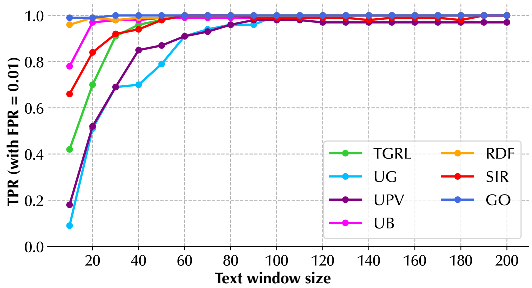


Figure 9: TPRs of watermarkers with respect to text length (with FPRs fixed as 1%).

Observe that as expected, the TPRs of all the methods improve as the text length grows from 1 to 200 tokens. As the text length exceeds 100 tokens, most methods reach TPRs close to 100%. Meanwhile, different methods show varying sensitivity to the text length. For instance, GO and RDF attain 100% TPRs with only 20 tokens, while UG reaches only around 50% TPR under the same setting. This can be explained as follows. Both GO and RDF use distribution transform-based samplers, which, conditional on given randomness (e.g., random permutation), generate the next token deterministically. Meanwhile, other methods randomly sample the next token from a given pool (e.g., green lists). Thus, GO and RDF tend to have stronger signals per token for watermark detection.

### E.3 Impact of Temperature

The temperature $\tau$ is a key parameter that affects a watermarker's generative dynamics: intuitively, a higher $\tau$ makes the sampling over the next-token distribution $\tilde{p}(x_t|x_{<t})$ more random. Here we evaluate how the setting of $\tau$ in each watermaker's generator may impact its effectiveness. Note that, unlike other watermarkers, RDF and GO do not generate the next-token distribution $\tilde{p}(x_t|x_{<t})$ explicitly, we thus exclude them from the evaluation.

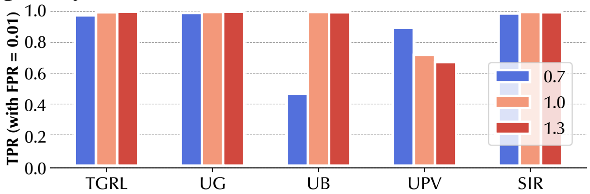


Figure 10: TPRs of watermarkers with respect to the temperature setting (with FPRs fixed as 1%).

Figure 10 compares how the TPRs of different watermarkers vary with the setting of $\tau$ = 0.7, 1.0, and 1.3 (with FPRs fixed as 1%). Observe that the performance of most watermarkers marginally improves with $\tau$, which corroborates prior work (Piet et al., 2023). For instance, the TPR of TGRL increases by about 0.05 as $\tau$ varies from 0.7 to 1.3. Interestingly, in contrast, the TPR of UPV decreases as $\tau$ grows. This can be explained as follows. UPV employs a neural network as the detector that depends on general textual features, which tends to be more sensitive to increasing randomness, compared with other watermarkers that rely on specific watermark signals (e.g., green/red-listed tokens).

## F Watermark Fidelity

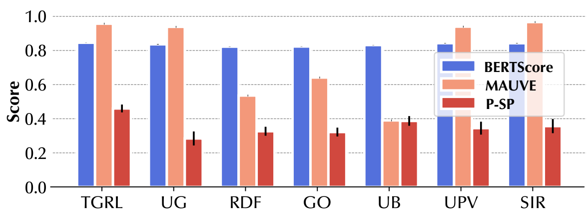


Figure 11: Fidelity preservation of different watermarkers.

We evaluate the impact of different watermarkers on the text quality. We compare the original text $T$ and watermarked text $\tilde{T}$ using the metrics (detailed in §3.2) of BERTScore (Zhang et al., 2020), P-SP (Wieting et al., 2022), and MAUVE (Pillutla et al., 2021). The results are summarized in Figure 11.

i) A majority of watermarkers well preserve the semantics of original texts, as indicated by their high BERTScore and MAUVE scores. Note that the P-SP scores of all the watermarkers are relatively lower than their BERTScore and MAUVE scores. This is due to their different emphases: P-SP measures the average similarity between the tokens in $T$ and $\tilde{T}$, while BERTScore calculates the maximum similarity between the tokens in $T$ and $\tilde{T}$. ii) Meanwhile, RDF, GO, and UB are less effective in preserving the quality of original texts, which can be attributed to their additional constraints of indistinguishability: the expectation of the watermarker's next-token distribution is identical to the LLM's next-token distribution (i.e., indistinguishability). This observation suggest that there exists an inherent trade-off between the desiderata of quality and indistinguishability.

# G Additional Results

## G.1 Multi-bit Watermarking

While our study focuses on one-bit watermarkers, for completeness, we also evaluate CTWL (Wang et al., 2024), a multi-bit watermarker. In contrast of one-bit watermarkers that encode only a single bit of information (i.e., whether a given text is watermarked), a multi-bit watermarker can encode multiple bits of information into the watermarked text, such as the generating model, the date of generation, and other details. However, despite its larger information capacity, we find that a multi-bit watermarker is typically less robust compared to one-bit watermarkers, as illustrated in Figure 12 and 13.

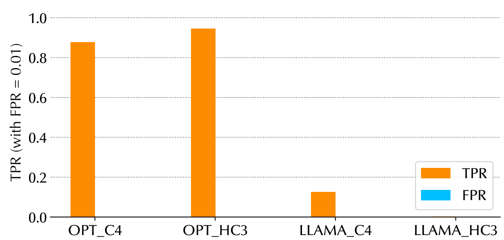


Figure 12: TPRs of CTWL (with FPRs fixed as 0.01) on different LLMs (OPT and Llama3) and datasets (C4 and HC3).

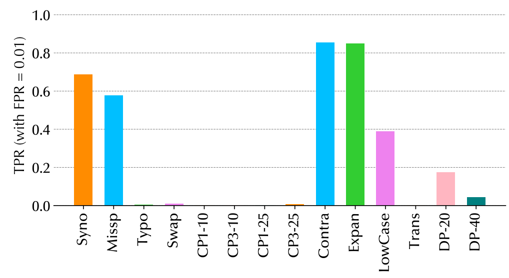


Figure 13: TPRs of CTWL (with FPRs fixed as 0.01) against various attacks.

The test results in the basic encoding and detection scenario highlight the sensitivity of CTWL to different language models. Although it performs well with the OPT model across both C4 and HC3 datasets, its TPR on Llama3 is extremely low, and it becomes completely ineffective when tested on Llama3 using the HC3 dataset. This is due to the lower model perplexity of Llama3. We conduct additional experiments to compare the model perplexity on the same WikiText dataset, and the results show that Llama3 has a perplexity about 6.15, while OPT has about 12.43. Lower model perplexity results to larger fluctuations in the logits produced by the model, makes it more challenging for watermark injection (e.g., increase smaller logits to exceed larger ones).

When facing the attacks, CTWL is more vulnerable than one-bit methods. It is particularly vulnerable to the copy-pasting attack, which can nearly disable the method as the TPR drops to near zero. Additionally, CTWL is highly susceptible to typoing, swapping, translating, lowercasing, and Dipper attacks, which generally do not affect many one-bit methods as severely. Despite the capacity of multi-bit methods to embed more information, their high sensitivity to language model variations and various attacks is a crucial limitation that needs to be addressed in future research.

## G.2 Additional Robustness of Watermarkers

Here we present the full robustness evaluation results across different models and datasets, which strongly corroborate our primary experimental findings. These experiments revealed several significant patterns that align with earlier observations, demonstrating that the relative robustness rankings of watermarking methods remain consistent regardless of model architecture or dataset characteristics. This consistency suggests that robustness is primarily determined by the inherent algorithmic properties of each method.

Table 9 shows the detailed results. Notably, UPV and SIR exhibited substantial vulnerability, with watermark detectability degrading even under simple Linguistic variation and Lexical editing attacks, indicating fundamental limitations in their robustness across model scales. Text-mixing attacks posed significant challenges for most watermarkers—particularly UPV, which failed completely under such conditions—while RDF and GO maintained reliable detection, highlighting key differences in algorithmic resilience. Furthermore, under paraphrasing attacks, both UG and RDF consistently achieved high detection rates, further supporting the conclusion that inherent watermarking design, rather than model- or dataset-specific factors, governs robustness against adversarial manipulations.

### G.3 Different Styles Datasets

To comprehensively evaluate watermarking performance across different text generation scenarios, we created diverse datasets spanning multiple styles and domains. The Paper Conclusion dataset was constructed by collecting research papers from arXiv and having LLMs summarize their key findings. We also incorporated Law Stack Exchange, a legal domain question-answering dataset sourced from the Stack Exchange forum, and WritingPrompts, a creative writing dataset focused on story completion tasks. These datasets can be categorized into three distinct styles: completion-style (C4 and Story Completion datasets(Euclaise)), question-answering (HC3 and Law Stack Exchange datasets(Li)), and summarization (Paper Conclusion dataset). Our analysis revealed several significant findings. Table 10 shows the results.

First, all watermarkers demonstrated notably lower performance on question-answering style datasets. This degradation was particularly pronounced for UPV and SIR, which achieved only around 0.5 TPR on these datasets. This can be attributed to the prevalence of domain-specific terminology and fixed expressions in Q&A datasets, which inherently constrain the model's linguistic choices and make watermark embedding more challenging. Second, robust watermarkers like RDF and GO maintained exceptional performance across all dataset styles, demonstrating their versatility and reliability. RDF achieved consistently high TPR scores ranging from 0.934 to 0.999, while GO demonstrated even more stable performance with TPR values between 0.978 and 0.998. These results highlight an important pattern: watermarkers that demonstrate strong baseline performance maintain their effectiveness across different text generation styles, while less stable or lower-performing watermarkers are particularly vulnerable to degradation when applied to question-answering formats. Notably, the high watermarking effectiveness on Paper Conclusion tasks (TPR > 0.99 for most methods) can be attributed to the inherent flexibility in academic summarization. Unlike Q&A tasks that require specific terminology and fixed expressions, paper conclusions allow for more freedom in expression while conveying the same core findings. This flexibility in language choice and sentence construction provides more opportunities for watermark embedding without compromising the semantic accuracy of the summary.

### G.4 Fidelity Preservation of attacks

Figure 14 illustrates the quality preservation of different attacks on GO, with similar results on other watermarkers,

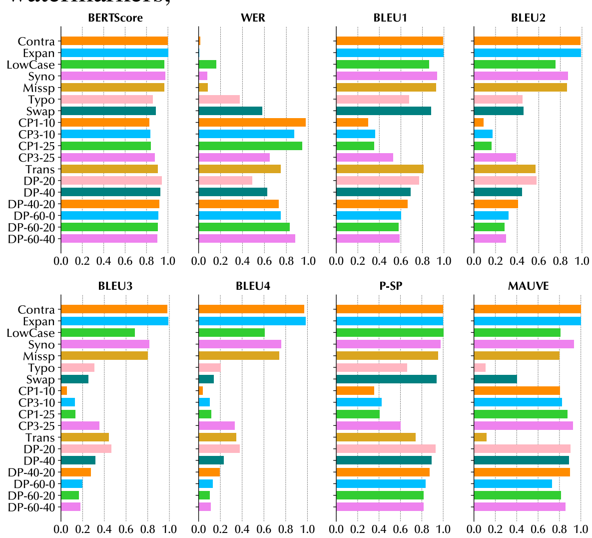


Figure 14: Quality preservation of different attacks on GO.

## H Additional Related Work

We survey the relevant literature in the following categories: i) detection of LLM-generated texts, ii) LLM watermarking, iii) attacks on LLM watermarking, and iv) evaluation of LLM watermarkers.

**Detection of LLM-generated texts.** The advances in LLMs also give rise to their possible misuses. There is thus a pressing need for the capability of distinguishing LLM- and human-generated texts. Initial work attempts to either train classifiers using LLM- and human-generated texts(Mitchell

| Watermarker | CLEAN | Linguistic variation | | | Lexical editing | | | | Text-mixing | | | | Paraphrasing | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Contra | Expan | LowCase | Swap | Typo | Syno | Missp | CP1-10 | CP3-10 | CP1-25 | CP3-25 | DP-20 | DP-40 | Trans |
| OPT + C4 | | | | | | | | | | | | | | | |
| TGRL | 0.992 | 0.994 | 0.994 | 0.984 | 0.874 | 0.788 | 0.996 | 0.992 | 0.034 | 0.176 | 0.126 | 0.836 | 0.952 | 0.858 | 0.652 |
| UG | 0.964 | 0.963 | 0.965 | 0.879 | 0.956 | 0.884 | 0.965 | 0.956 | 0.131 | 0.219 | 0.192 | 0.534 | 0.940 | 0.916 | 0.658 |
| UPV | 0.776 | 0.772 | 0.814 | 0.685 | 0.667 | 0.355 | 0.764 | 0.695 | 0.050 | 0.073 | 0.242 | 0.450 | 0.296 | 0.057 | 0.006 |
| RDF | 0.996 | 0.993 | 0.993 | 0.983 | 0.951 | 0.961 | 0.990 | 0.993 | 0.094 | 0.396 | 0.430 | 0.974 | 0.988 | 0.962 | 0.154 |
| SIR | 0.954 | 0.954 | 0.954 | 0.886 | 0.932 | 0.566 | 0.936 | 0.904 | 0.138 | 0.210 | 0.178 | 0.466 | 0.902 | 0.814 | 0.818 |
| GO | 0.998 | 0.998 | 0.998 | 0.988 | 0.920 | 0.852 | 0.996 | 1.000 | 0.181 | 0.589 | 0.620 | 0.992 | 0.971 | 0.847 | 0.928 |
| LLAMA + C4 | | | | | | | | | | | | | | | |
| TGRL | 0.896 | 0.864 | 0.892 | 0.75 | 0.408 | 0.34 | 0.818 | 0.78 | 0.014 | 0.074 | 0.088 | 0.348 | 0.536 | 0.334 | 0.196 |
| UG | 0.814 | 0.791 | 0.823 | 0.697 | 0.843 | 0.357 | 0.75 | 0.755 | 0.025 | 0.054 | 0.050 | 0.190 | 0.771 | 0.633 | 0.344 |
| UPV | 0.588 | 0.502 | 0.598 | 0.494 | 0.388 | 0.018 | 0.512 | 0.392 | 0.005 | 0.002 | 0.002 | 0.007 | 0.362 | 0.355 | 0.042 |
| RDF | 0.790 | 0.735 | 0.725 | 0.622 | 0.333 | 0.888 | 0.738 | 0.824 | 0.021 | 0.045 | 0.045 | 0.307 | 0.471 | 0.303 | 0.010 |
| SIR | 0.566 | 0.066 | 0.078 | 0.078 | 0.076 | 0.010 | 0.064 | 0.050 | 0.034 | 0.026 | 0.042 | 0.048 | 0.040 | 0.081 | 0.100 |
| GO | 0.996 | 0.996 | 0.996 | 0.959 | 0.643 | 0.556 | 0.989 | 0.987 | 0.058 | 0.228 | 0.286 | 0.825 | 0.900 | 0.623 | 0.814 |
| QWEN + Paper Conclusion | | | | | | | | | | | | | | | |
| TGRL | 0.993 | 0.944 | 0.944 | 0.831 | 0.429 | 0.222 | 0.887 | 0.915 | 0.000 | 0.667 | 0.833 | 0.833 | 0.667 | 0.485 | 0.222 |
| UG | 0.993 | 0.857 | 0.833 | 0.976 | 0.929 | 0.930 | 0.976 | 0.738 | 0.857 | 0.714 | 0.714 | 0.991 | 0.976 | 0.877 | 0.921 |
| UPV | 0.400 | 0.400 | 0.400 | 0.430 | 0.080 | 0.000 | 0.380 | 0.360 | 0.000 | 0.000 | 0.000 | 0.000 | 0.200 | 0.080 | 0.240 |
| RDF | 0.999 | 0.998 | 0.996 | 0.996 | 0.976 | 0.979 | 0.991 | 0.993 | 0.872 | 0.893 | 0.932 | 0.978 | 0.905 | 0.738 | 0.978 |
| UB | 0.980 | 1.000 | 1.000 | 0.962 | 0.000 | 0.033 | 0.921 | 1.000 | 0.018 | 0.042 | 0.000 | 0.485 | 0.514 | 0.103 | 0.255 |
| SIR | 0.978 | 0.580 | 0.500 | 0.420 | 0.320 | 0.320 | 0.440 | 0.560 | 0.340 | 0.360 | 0.360 | 0.460 | 0.400 | 0.300 | 0.000 |
| GO | 0.996 | 0.996 | 0.996 | 0.994 | 0.982 | 0.956 | 0.986 | 0.996 | 0.864 | 0.887 | 0.946 | 0.982 | 0.640 | 0.560 | 0.667 |

Table 9: Attack resilience of LLM watermarkers. The intensity of red shading indicates higher values, while the intensity of blue shading indicates lower values, with 0.5 serving as the threshold between the two color gradients.

| Watermarker | Summarization | Completion | | Question-Answer | |
|---|---|---|---|---|---|
| | Conclude | C4 | StoryComple | HC3 | LawQA |
| TGRL | 0.993 | 0.992 | 0.992 | 0.926 | 0.934 |
| UG | 0.993 | 0.972 | 0.981 | 0.864 | 0.891 |
| UPV | 0.400 | 0.540 | 0.560 | 0.460 | 0.430 |
| RDF | 0.999 | 0.998 | 0.996 | 0.945 | 0.934 |
| SIR | 0.978 | 0.965 | 0.976 | 0.566 | 0.631 |
| GO | 0.996 | 0.998 | 0.994 | 0.986 | 0.978 |

Table 10: Performance of watermarkers across different datasets on Qwen2.5-14B.

et al., 2023) or to leverage intrinsic characteristics of LLM-generated texts (e.g., perplexity and variability in length, complexity, and information density)(Lütkebohle). Yet, with LLMs becoming increasingly capable, the difference between LLM- and human-generated texts is narrowing, making such approaches less effective.

**LLM watermarking** In response, LLM watermarking emerges as a promising alternative, which instruments the LLM generative process with statistical signals that can be subsequently detected. The existing LLM watermarking techniques can be categorized based on the stages in which they are applied(Liu et al., 2023b): i) training-time watermarking(Liu et al., 2023c; Tang et al., 2023; Sun et al., 2022, 2023; Xu et al., 2024), ii) watermarking during logit-generation(Kirchenbauer et al., 2023a; Zhao et al., 2023; Hu et al., 2024; Liu et al., 2023a; Liu et al., 2024; Fairoze et al., 2023; Ren et al., 2023; Fernandez et al., 2023; Wang et al., 2024; Yoo et al., 2023; Lee et al., 2023; Giboulot and Teddy, 2024), iii) watermarking during token-sampling (Kuditipudi et al., 2023; Aaronson and Kirchner; Hou et al., 2023), and iv) post-generation watermarking(Zhang et al., 2023; Yoo et al., 2024; Yang et al., 2023; Munyer and Zhong, 2023; Sato et al., 2023; Abdelnabi and Fritz, 2021). This study mainly focuses on training-free, pre-generation watermarking, which applies to any given LLMs and provides flexible control over multiple criteria (e.g., quality, effectiveness, and robustness). Some watermarks also consider asymmetric schemes (Fairoze et al., 2023), while our study mainly focuses on symmetric watermarking schemes due to their widespread adoption.

The primary focus of the previous studies is to distinguish between human-written text and text generated by LLMs. In this paper, we assume that the attacker lacks access to non-watermarked text produced by recent LLM. Instead, the attacker can only utilize other LLMs to generate text that mimics human responses and get the watermarked texts from recent LLM. The generic detector, which is within the adversary's capability, is specifically designed to distinguish between watermarked and non-watermarked texts(e.g., other LLMs' texts or human writing texts).

**Attacks on LLM watermarking.** One critical property of an LLM watermarker is its robustness against potential attacks. A variety of attacks can be applied to LLM watermarking, ranging from removing the embedded watermark to uncovering the green/red lists. For instance, Dipper(Krishna et al., 2023) is a widely used paraphrasing attack to evaluate the robustness of LLM watermarkers (Kirchenbauer et al., 2023a; Zhao et al., 2023) against watermark removal attacks; the watermark stealing attack(Jovanović et al., 2024) is proposed to identify green-list tokens and to replace them with red-list tokens, targeting TGRL(Kirchenbauer et al., 2023a) and UG(Zhao et al., 2023) to further launch spoofing or removal attacks. This study primarily focuses on watermark removal attacks as they can target any LLM watermarkers and have profound implications in practice (e.g., disinformation, academic cheating, and automated phishing).

**Evaluation of LLM watermarkers.** As LLM watermarkers become prevalent, recent work attempts to benchmark the performance of various watermarkers. However, current studies either focus solely on the effectiveness of watermarking or have limited assessments of robustness. For instance, WaterBench(Tu et al., 2023) compares the effectiveness of TGRL(Kirchenbauer et al., 2023a) and UG(Zhao et al., 2023) under varying hyper-parameter settings (e.g., prompt length); LLM-judger(Singh and Zou, 2023) uses GPT-3.5-Turbo as a judge to evaluate the effectiveness of RDF(Kuditipudi et al., 2023) and TGRL(Kirchenbauer et al., 2023a) and employs a binary classifier based on MLP to distinguish between watermarked and non-watermarked texts; MarkMyWords(Piet et al., 2023) compares the effectiveness of four watermarkers including TGRL(Kirchenbauer et al., 2023a), GO(Aaronson and Kirchner), RDF(Kuditipudi et al., 2023), and UW(Christ et al., 2023), and evaluates the robustness of TGRL against watermark removal attacks.

While our study shares some evaluation components with MarkMyWords, our contributions differ substantially in scope and depth. MarkMyWords includes robustness testing as part of a broader evaluation, whereas our work provides a dedicated and systematic framework specifically for stress-testing watermark robustness. This specialized focus enables deeper exploration of attack types, adaptive strategies, and evaluation metrics. Methodologically, we introduce several new dimensions not covered in prior work, such as causal analysis of watermark design choices, comparisons between specific and generic detectors, and adaptive attacks mounted by adversaries using generic detectors. Furthermore, we establish a standardized benchmark comprising 12 representative attacks and 8 evaluation metrics—facilitating fine-grained robustness analysis, including next-token distribution comparisons and what-if attack simulations.

Our systematic approach has uncovered several novel findings that go beyond the scope of existing evaluations—for example, challenging assumptions in prior work (see Table 1) and identifying key trade-offs in design robustness. We acknowledge the need to better articulate these distinctions and will revise the related work section to more clearly position our contributions relative to prior efforts such as MarkMyWords.

To our best knowledge, this is the first study solely dedicated to evaluating the robustness of LLM watermarkers against removal attacks. Our goal is to understand how design decisions influence resistance to adversarial perturbations and to establish best practices for deploying watermarkers in real-world adversarial environments.

## I Evaluation Guidelines for Future LLM Watermarking Research

Next, we propose a set of guidelines for evaluating the robustness of LLM watermarkers. These guidelines incorporate our findings in §4 and §5, providing a minimal checklist to claim the robustness of an LLM watermarker.

**LLMs and tasks.** Our experiments show that the referenced watermarkers show varying robustness across different LLMs and datasets. We speculate that there exists an intricate interplay between the watermarking mechanism, the LLM's capability, and the task's complexity. We thus recommend experimenting on i) LLMs with varying capability (e.g., as measured by perplexity), ii) datasets for different tasks (e.g., summarization and question-answering), and iii) their combinations.

**Attacks.** Notably, using highly capable LLMs or applying computationally expensive rewriting can easily generate highly-quality, non-watermarked texts to evade detection; however, such attacks negate the need for watermark removal attacks in the first place. We thus recommend focusing on computationally efficient attacks such as linguistic variation, lexical editing, and lightweight paraphrasing, as well as their combinations, which reflects the risks of watermark removal attacks in practical settings.

**Robustness.** It is often critical to properly set the decision threshold for a watermarker (and also the attacks) to fully assess its robustness(Lukas et al., 2021), which unfortunately is often missing in the original papers. To overcome this issue, we recommend i) measuring the overall effectiveness (TPR) in terms of ROC (across different threshold settings), ii) measuring the TPR under a fixed FPR (e.g., 0.01), and iii) considering varying attack intensity.

**Fidelity.** It is notoriously challenging to meaningfully measure the quality of text data(Pillutla et al., 2021). We recommend employing a variety of metrics (e.g., BERTScore, P-SP, MAUVE) to comprehensively measure the quality retention of watermarkers as well as attacks. In addition, one may also leverage external tools (e.g., more advanced LLMs such as GPT-4) or human evaluation to provide a more accurate assessment if feasible.

## J Deployment Guidelines for Adversarial Environments

To complement our empirical evaluation, we distill our findings into actionable deployment guidelines for practitioners operating watermarkers in adversarial settings.

**Watermarker selection.** Our results show that watermarkers differ significantly in their robustness to specific attack types. We recommend selecting watermarkers based on the anticipated threat landscape: RDF and UG are especially resilient against paraphrasing and text-mixing, while TGRL performs well against linguistic variations. Practitioners should avoid UPV and SIR in high-adversity settings due to their instability and vulnerability under attack combinations.

**Detector configuration.** For watermarkers with unstable detectors (e.g., UPV), using hybrid detection schemes—combining score-based and model-based detectors or incorporating a generic detector—can mitigate failure modes. Our results (Table 3) suggest that generic detectors can complement specific ones, enhancing recall without significant degradation in precision.

**Multi-layered defense.** Since individual attacks (e.g., typoing) are often insufficient to remove watermarks, while combined or adaptive attacks (e.g., swapping + synonymizing or GBDA) are far more effective, we recommend adopting multi-layered detection pipelines. These may involve edit-distance-aware scoring (e.g., RDF's edit score) and robustness testing against common perturbation chains.

**Risk assessment.** To assess real-world risk, practitioners should consider not only overall TPR/AUC, but also worst-case TPR under fixed low FPR (e.g., 1%) across combined and high-intensity attacks. Our findings highlight that even robust watermarkers may fail under iterated paraphrasing (Figure 7).

**Resource-aware deployment.** For cost-sensitive applications (e.g., online moderation), we recommend avoiding watermarkers requiring expensive generation/detection steps (e.g., deep model-assisted detection in UPV), and instead favoring methods like UG or RDF that are both robust and lightweight.