Can LLM Watermarks Robustly Prevent Unauthorized Knowledge Distillation?

Anonymous ACL submission

Abstract

The radioactive nature of Large Language Model (LLM) watermarking enables the detection of watermarks inherited by student models when trained on the outputs of watermarked teacher models, making it a promising tool for preventing unauthorized knowledge distillation. However, the robustness of watermark radioactivity against adversarial actors remains largely unexplored. In this paper, we investigate whether student models can acquire the capabilities of teacher models through knowledge distillation while avoiding watermark inheritance. We propose two categories of watermark removal approaches: pre-distillation removal through untargeted and targeted training data paraphrasing (UP and TP), and post-distillation removal through inference-time watermark neutralization (WN). Extensive experiments across multiple model pairs, watermarking schemes and hyper-parameter settings demonstrate that both TP and WN thoroughly eliminate inherited watermarks, with WN achieving this while maintaining knowledge transfer efficiency and low computational overhead. Given the ongoing deployment of watermarking techniques in production LLMs, these findings emphasize the urgent need for more robust defense strategies.

1 Introduction

The capability of Large Language Models (LLMs) to rapidly generate high-quality text at scale makes them valuable sources of training data (Zoph et al., 2022). However, many leading LLM services explicitly prohibit the use of their outputs for training competing models through knowledge distillation in their terms of service. Notable examples include OpenAI, Anthropic and Meta Llama, as detailed in Appendix A.

Watermarking has emerged as a solution to monitor unauthorized usage (Kirchenbauer et al., 2023; Zhao et al., 2024; Liu et al., 2024c; Zhao et al., 2023). Research has shown that watermarked LLMs exhibit radioactivity - student models trained on their outputs inherit detectable watermarks (Sander et al., 2024; Gu et al., 2024). This traceability has led to increasing practical implementations, such as Google DeepMind's integration of SynthID-Text (Dathathri et al., 2024) into Gemini chatbots (Team et al., 2023).

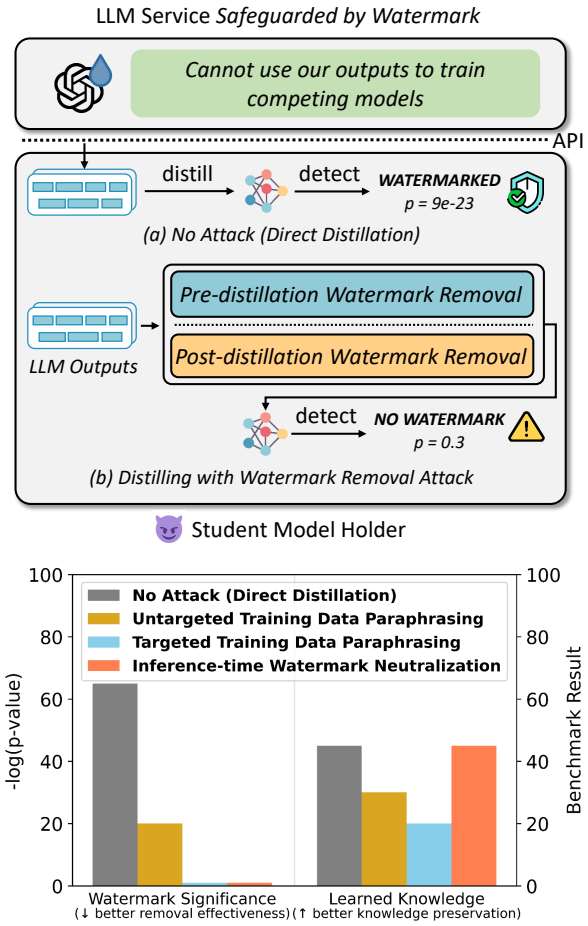


Figure 1: LLM watermarking has been proposed as a safeguard against unauthorized knowledge distillation. However, our pre- and post-distillation watermark removal attacks allow student models to perform untraceable knowledge distillation, emphasizing the need for more robust design. The bar chart displays the effectiveness of watermark removal and knowledge preservation for our three proposed attacks.

As watermarking emerges as a promising approach to protect model copyrights from knowledge distillation, its robustness against adversarial actors remains largely unexplored. We conduct the first systematic investigation into watermark resilience and propose two categories of watermark removal attacks: pre-distillation removal through untargeted and targeted training data paraphrasing (**UP** and **TP**), and post-distillation removal through inference-time watermark neutralization (**WN**), as illustrated in the upper part of Figure 1. Experiments show that TP and WN can thoroughly eliminate inherited watermarks, with WN **achieves watermark removal** while **preserving distilled knowledge** and **maintaining low computational overhead** - raising important questions about the reliability of preventing unauthorized knowledge distillation through watermarks.

Given that both TP and WN require knowledge of watermark rules, we propose a watermark stealing technique. Unlike existing methods (Jovanović et al.; Wu and Chandrasekaran, 2024; Zhang et al., 2024), our approach (1) does not need access to the watermarking scheme or its hyper-parameters, and (2) assigns weights by analyzing factors affecting watermark radioactivity, allowing for more targeted rule extraction. In TP, we integrate the inverse of extracted watermark rules into paraphrase models like Dipper (Krishna et al., 2023) to remove watermark. In contrast, UP simply employs standard paraphrasing tools without considering rules. For post-distillation removal, we develop watermark neutralization that directly counteracts inherited watermarks by applying inverse rules during the student model's decoding phase.

Extensive experiments were conducted across 2 Teacher-Student model pairs $\times$ 2 leading watermarking schemes $\times$ 3 hyperparameter settings. The comparative results are summarized in the bottom part of Figure 1. Both TP and WN effectively eliminate inherited watermarks, reducing detection significance to levels similar to non-watermarked conditions (above 10^{-2}) across all settings. Evaluations on benchmark datasets, including ARC challenge (Clark et al., 2018), TruthfulQA (Lin et al., 2022), and MTBench (Zheng et al., 2023) show that WN exhibits superior knowledge preservation, achieving comparable performance to baseline student models trained without any watermark removal techniques. This indicates that student models can leverage WN to remove watermarks without sacrificing model performance, posing a significant challenge to the practical deployment of watermark as a copyright protection mechanism.

Key Contributions Our main contributions are:

- We conduct the first systematic investigation into the robustness of watermarking schemes against adversarial actors in monitoring unauthorized knowledge distillation, proposing pre-distillation and post-distillation attacks.
- Our proposed targeted paraphrasing and watermark neutralization methods achieve thorough watermark removal, with the latter demonstrating superior knowledge preservation. This raises concerns about the reliability of current watermarking schemes for monitoring unauthorized knowledge distillation.
- Further discovery of watermark collisions in multi-source knowledge distillation scenarios reveals additional limitations of watermarking schemes in monitoring unauthorized knowledge distillation (Section 5.2). Given the ongoing deployment of watermarking techniques in production LLMs, these findings highlight the urgent need for more robust defense strategies (Section 5.3).

2 Background

2.1 LLM Watermarking Schemes

Most of the existing watermarking schemes follow the n-gram paradigm, modifying the next token's probability prediction based on the preceding $n-1$ tokens, thereby influencing the final sampling outcome (Kirchenbauer et al., 2023; Zhao et al., 2024; Dathathri et al., 2024; Liu et al., 2024c,b; Lee et al., 2024; Hu et al., 2024; Wu et al., 2023; Aaronson and Kirchner, 2022; Kuditipudi et al., 2024). Watermark schemes tested in this work are:

KGW (Kirchenbauer et al., 2023) sets the ground work for generative LLM watermarking. For the tth token generation, it computes a hash $h_t = H(x_{t-n+1:t-1})$ from the previous $n-1$ tokens. This hash partitions the vocabulary $\mathcal{V}$ into a *green list* $\mathcal{V}_g$ and a *red list* $\mathcal{V}_r$. A constant bias δ is then added to the logits of green tokens:

$$l_t'^{(i)} = l_t^{(i)} + \delta \text{ if } v_i \in \mathcal{V}_g \text{ else } l_t^{(i)}. \quad (1)$$

As a result, watermarked text will statistically contain more *green* tokens, and can be detected by computing the z-score:

$$z = (|s|_G - \gamma T)/(\sqrt{\gamma(1-\gamma)T}), \quad (2)$$

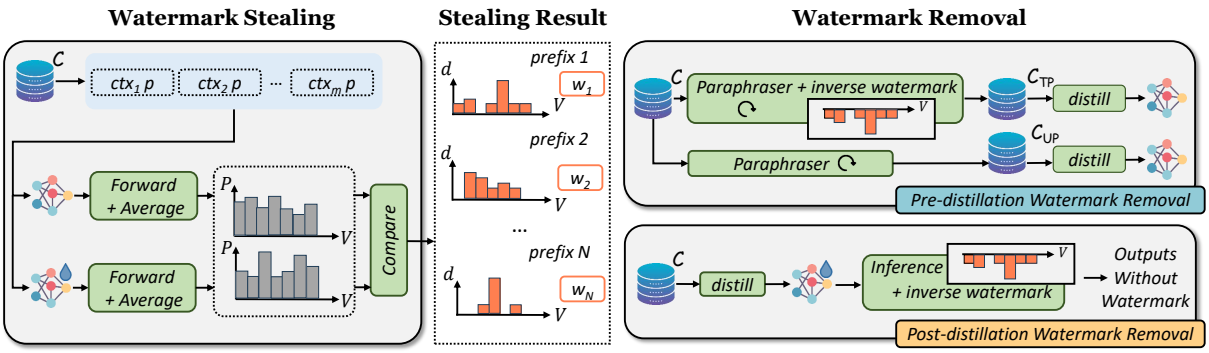


Figure 2: An illustration of the proposed watermark removal attacks.

where $|s|_G$ counts green tokens in text length T, and $\gamma = |\mathcal{V}_g|/|\mathcal{V}|$.

SynthID-Text (Dathathri et al., 2024), recently announced by Google DeepMind, is the first watermarking algorithm deployed in production, and has been integrated into the Gemini and Gemini Advanced chatbots. For the tth token generation, it computes a hash $h_t = H(x_{t-n+1:t-1})$ to seed m binary classifiers $g_1, g_2, ..., g_m$, which randomly assign 0 or 1 to vocabulary tokens. It then samples 2^m tokens from the original distribution $P(x_t|x_{1:t-1})$ and conducts tournament sampling: tokens compete in pairs based on g_1 values in the first round, with subsequent rounds using $g_2, g_3, ..., g_m$ until one token remains. The watermark manifests as a statistical bias toward tokens with higher g values, detectable by computing their mean:

$$\overline{g} = \sum_{t=1}^{T} \sum_{\ell=1}^{m} g_\ell(x_t)/mT. \quad (3)$$

2.2 Watermark Radioactivity

Research shows that watermarked LLMs exhibit radioactivity (Sander et al., 2024; Gu et al., 2024): student models trained on their outputs inherit the watermark patterns. This effect is highly significant, with reported p-values below 10^{-30} even under the most stringent conditions where the teacher model is closed-source and detection is unsupervised (test prompts are disjoint from training data).

2.3 Watermark Removal Approaches

Prior work has explored various watermark removal methods, but focused on removing watermarks from generated text rather than models. These include untargeted approaches like paraphrasing, emoji attacks (Kirchenbauer et al., 2023), back-translation, and cross-lingual removal (He et al., 2024). For targeted removal, Jovanović et al.; Wu and Chandrasekaran (2024); Zhang et al. (2024) proposed watermark stealing-and-removing, requiring knowledge of the type of watermarking scheme and the window size used.

3 Methodology

3.1 Threat Model

Here are the threat model scenarios in which the student model holder acts as the attacker: **(1) Attack Target**: n-gram based watermarking schemes, which represent the primary approach in LLM watermarking (see Appendix D.2 for further discussion of other paradigms); **(2) Access Level**: Requires only access to the LLM service API, with no need for logits or watermark detection systems; **(3) Testing Environment**: Unsupervised testing, where test prompts differ from training data, as the LLM service cannot track which data was used to train the student model.

3.2 Overview of the Proposed Watermark Removal Methods

We propose two categories of watermark removal methods: **pre-distillation** and **post-distillation** watermark removal, as illustrated in Figure 2. Pre-distillation methods remove watermarks from training data using external paraphrase models. These methods include untargeted paraphrasing (**UP**), which directly rewrites training data, and targeted paraphrasing (**TP**), which first steals watermarking rules and then applies an inverse watermark on the paraphrase model to rewrite training data. Post-distillation method first steals watermark rules, and then neutralizes the inherited watermark by directly adding an inverse watermark during the student model's decoding phase. We refer to this process as watermark neutralization (**WN**). Details of these methods are presented in Sections 3.3 and 3.4, while our watermark stealing method used in both TP and WN is introduced in Section 3.5.

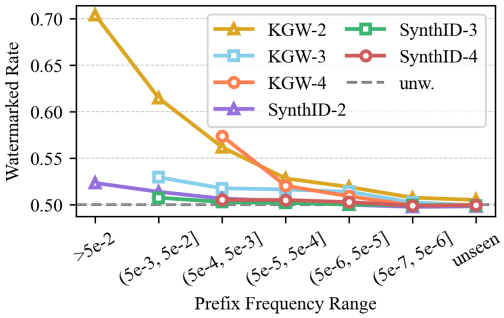


Figure 3: Correlation between prefix frequency in training data and the probability that tokens following these prefixes are watermarked in student model outputs.

3.3 Pre-distillation Watermark Removal

Let $\mathcal{R}$, $\mathcal{C}$, $\mathcal{O}$, and $\mathcal{W}$ denote the paraphrase model, training dataset collected from watermarked teacher model's API, original student model, and student model trained on $\mathcal{C}$ without attacks, respectively. For both TP and WN, we denote the watermark stealing result as $D(x_t; x_{t-n'+1:t-1})$, representing the confidence that x_t is a watermarked token following $x_{t-n'+1:t-1}$. Section 3.5 details the computation of D.

Targeted Training Data Paraphrasing During paraphrasing, we apply an inverse watermark to the paraphrase model $\mathcal{R}$'s logits based on D:

$$l'_{\mathcal{R}}(x_t|x_{1:t-1}) = l_{\mathcal{R}}(x_t|x_{1:t-1}) - D(x_t; x_{t-n'+1:t-1}) \cdot \delta', \tag{4}$$

where δ' controls the strength. This yields a new training dataset $\mathcal{C}_{TP}$ for the student model.

Untargeted Training Data Paraphrasing As a comparison, this method directly applies $\mathcal{R}$ to rewrite training data, yielding dataset $\mathcal{C}_{UP}$.

3.4 Post-distillation Watermark Removal

This approach neutralizes watermark by directly applying the inverse watermark to the trained student model $\mathcal{W}$'s logits during inference:

$$l'_{\mathcal{W}}(x_t|x_{1:t-1}) = l_{\mathcal{W}}(x_t|x_{1:t-1}) - D(x_t; x_{t-n'+1:t-1}) \cdot \delta'. \tag{5}$$

3.5 Watermark Stealing

This subsection presents our watermark stealing method that extracts token preferences following a prefix p (denoted as p-rule).

3.5.1 Watermark Radioactivity Factors

To efficiently steal watermarks, we first analyze factors that affecting watermark radioactivity. This analysis helps limit watermark stealing scope to rules with stronger inheritance patterns, reducing computational cost and minimizing model modifications needed for watermark removal. Our experiments reveal two key factors: **(1)** the occurring frequency of the prefix p in training data, and **(2)** the window size n used in watermarking schemes.

Table 1: Median p-values for watermark detection in student model outputs, evaluated on groups of *1 million* tokens, across varying watermark window sizes n.

	$n=1$	$n=2$	$n=3$	$n=4$
KGW	6.24e-25979	4.79e-2537	1.67e-23	0.14
SynthID-Text	6.20e-4028	6.08e-887	0.58	0.64

Setup GLM-4-9b-chat (GLM et al., 2024) is used as the teacher model to generate 200k QA pairs for training Llama-7b (Touvron et al., 2023). KGW (Kirchenbauer et al., 2023) and SynthID-Text (Dathathri et al., 2024) are used as watermarking schemes with $n = 1, 2, 3, 4$. We evaluated the inherited watermark strength in the student model using the C4 dataset (Raffel et al., 2020) as prompts.

Prefix Frequency vs. Radioactivity As shown in Figure 3, more frequent prefixes in training dataset lead to stronger watermark radioactivity of their p-rules in student model's outputs, across all schemes and settings. For rare prefixes (frequency $\leq 5 \times 10^{-5}$), the radioactivity of their corresponding p-rules approaches that of unwatermarked text. Note: $n = 1$ is excluded in the figure as it uses global, prefix-independent watermark rules.

Window Size n vs. Radioactivity As shown in Table 1, the watermark radioactivity falls dramatically as n increases. For both KGW and SynthID-Text, watermarks become undetectable even with groups of *1 million* tokens[1] when n reaches 4. This is because: (1) shorter p-rules are simpler, making it easier for student models to learn; (2) as n increases, there is a marked expansion in the variety of prefixes generated by student models, resulting in fewer high-frequency prefixes and more unseen ones in the training data (as shown in Figure 4).

Scope of Watermark Stealing Based on the preceding analysis, when conducting watermark stealing, we need only focus on scenarios with small values of n (i.e., $n \leq 3$). Furthermore, for cases where $n \neq 1$, we can restrict our attention to high-frequency prefixes (i.e. those with frequencies exceeding 5×10^{-5}).

[1]For watermarked text, larger token samples yield stronger detection significance.

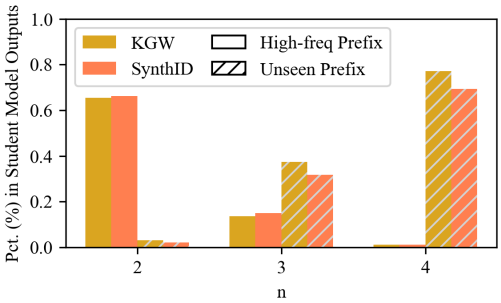


Figure 4: Percentages of high-frequency (5×10^{-5}) and unseen prefixes in training data within student model outputs, at different n.

3.5.2 Watermark Stealing Process

Unlike prior work (Jovanović et al.; Wu and Chandrasekaran, 2024; Zhang et al., 2024), our proposed stealing method operates effectively without knowing the exact watermarking scheme or window size. We first assume a window size n used by the teacher model to extract watermark rules, then obtain the final output by aggregating results from all windows less or equal to the maximum window size n' considered. Based on Section 3.5.1, n' typically remains small, ensuring manageable computational complexity.

Scoring Single n-gram Regardless of the specific watermarking algorithm, the core mechanism is adjusting the sampling preferences of the subsequent token based on prefix tokens. Therefore, our objective is to identify preferred tokens following prefix $p = x_{t-n+1:t-1}$ by assigning a score in $[0, 1]$ for each $v \in \mathcal{V}$, indicating the confidence value of "v is a watermarked token following p".

Let $\mathcal{O}$ denote the original student model, $\mathcal{W}$ denote the student model after training on watermarked data, and $\mathcal{C}$ represent the training corpus. To extract p-rules, we collect all contexts in $\mathcal{C}$ that end with p, perform forward passes using both $\mathcal{O}$ and $\mathcal{W}$ on these contexts to obtain next token probability predictions, and average the predictions across different contexts, which are:

$$\overline{P_{\mathcal{O}}}(x_t|p) = \mathbb{E}_{c \in \mathcal{C}, c_{t-n+1:t-1}=p}[P_{\mathcal{O}}(x_t|c)]. \quad (6)$$

$$\overline{P_{\mathcal{W}}}(x_t|p) = \mathbb{E}_{c \in \mathcal{C}, c_{t-n+1:t-1}=p}[P_{\mathcal{W}}(x_t|c)]. \quad (7)$$

Comparing these two distributions reveals the context-independent statistical bias of tokens following prefix p, characterizing the watermark patterns. We quantify the distribution shift and score the n-gram using $d(x_t; x_{t-n+1:t-1})$:

$$d(x_t; [x_{t-n+1:t-1}]) = \frac{1}{2} \min(2, \frac{\overline{P_{\mathcal{W}}}(x_t|x_{t-n+1:t-1})}{\overline{P_{\mathcal{O}}}(x_t|x_{t-n+1:t-1})}), \quad (8)$$

if $\overline{P_{\mathcal{W}}}(x_t|x_{t-n+1:t-1}) > \overline{P_{\mathcal{O}}}(x_t|x_{t-n+1:t-1})$. Otherwise, $d(x_t, x_{t-n+1:t-1}) = 0$. Note that if $n = 1$, which means the watermark rule is globally fixed, $d(x_t)$ is computed by quantifying the average probability shifts across all contexts.

Considering Multiple Window Sizes Since the window size n of the watermark scheme used in the teacher model is unknown, we need to aggregate scoring results across different n-gram sizes. Let n' be the maximum window size under consideration. The final confidence score is then defined as:

$$D(x_t; x_{t-n'+1:t-1}) = d(x_t) + \sum_{i=1}^{n'-1} w(x_{t-i:t-1}) \cdot d(x_t; x_{t-i:t-1}), \quad (9)$$

where $w(x_{t-i:t-1})$ is the weight assigned to the prefix based on its occurring frequency in training data. The weight value is computed as follows:

$$w(x_{t-k:t-1}) = \begin{cases} \left(\frac{\log f(x_{t-k:t-1})}{\log \max_{c \in \mathcal{C}_k} f(c)}\right)^{-\alpha} & \text{if } f(x_{t-k:t-1}) > \theta, \\ 0 & \text{otherwise,} \end{cases} \quad (10)$$

where f denotes the occurring frequency in training data, $\mathcal{C}_k$ represents the set of all unique k-grams appearing in $\mathcal{C}$, and α is a smoothing parameter. This function assigns higher weight values to prefixes with higher frequency.

4 Experiments

4.1 Setup

Teacher and Student Models Teacher: GLM-4-9b-chat (GLM et al., 2024); Students: Llama-7b (Touvron et al., 2023) and Llama-3.2-1b (Dubey et al., 2024).

Watermarking Schemes KGW (Kirchenbauer et al., 2023) and SynthID-Text (Dathathri et al., 2024) with $n = 1, 2, 3$. Results for more watermarking schemes can be found in Appendix D.1.

Training Details Dataset is collected by prompting the teacher model to generate 200k QA pairs (detailed in Appendix G). We employ LlamaFactory (Zheng et al., 2024) to perform supervised fine-tuning to the student models, with a learning rate of 1e-5 and 3 epochs for all test settings.

Testing Details For watermark detection, we prompted the distilled student models to generate texts using C4 dataset (Raffel et al., 2020). The generated tokens were grouped into fixed-size samples, with p-values calculated for each group and the median reported. For knowledge preservation,

Table 2: Median p-values for watermark detection using **UP** (Untargeted Training Data Paraphrasing), **TP** (Targeted Training Data Paraphrasing), and **WN** (Watermark Neutralization), compared against direct training (No Attack) and unwatermarked conditions (Unw.). indicates high watermark confidence, indicates low watermark confidence, and unshaded cells indicate insufficient evidence for watermark presence. Student model used in this table is Llama-7b, results for Llama-3.2-1b can be found in Appendix C.1.

Watermarking Scheme		Token Num.	Unw.	No Attack	UP	TP	WN
KGW	$n = 1$	1k	5.75e-01	8.97e-29	6.82e-03	8.27e-01	8.20e-02
		2k	5.71e-01	6.49e-55	2.43e-04	6.99e-01	2.72e-02
		3k	6.01e-01	2.68e-81	9.68e-06	7.98e-01	1.21e-02
	$n = 2$	10k	4.80e-01	4.12e-28	2.18e-03	6.88e-01	9.85e-02
		20k	4.47e-01	4.12e-53	1.29e-05	7.37e-01	3.35e-02
		30k	3.62e-01	8.26e-79	1.05e-07	7.43e-01	1.24e-02
	$n = 3$	100k	3.40e-01	1.85e-03	8.52e-01	4.30e-01	5.95e-01
		300k	3.41e-01	8.98e-09	9.51e-01	3.23e-01	6.80e-01
		1 million	4.84e-01	1.67e-23	8.63e-01	3.69e-01	8.63e-01
SynthID-Text	$n = 1$	1k	9.67e-01	1.46e-05	7.10e-01	9.98e-01	9.44e-01
		2k	9.95e-01	1.08e-09	7.69e-01	9.96e-01	9.88e-01
		3k	9.99e-01	1.02e-13	8.05e-01	9.98e-01	9.97e-01
	$n = 2$	10k	4.23e-01	6.67e-11	1.10e-01	4.97e-01	1.52e-01
		20k	3.82e-01	8.83e-20	4.29e-02	5.42e-01	7.30e-02
		30k	3.09e-01	1.65e-28	1.53e-02	4.45e-01	4.40e-02
	$n = 3$	100k	9.98e-01	5.28e-01	9.92e-01	9.94e-01	9.87e-01
		300k	9.76e-01	5.78e-01	9.99e-01	9.91e-01	9.49e-01
		1 million	9.87e-01	5.83e-01	9.99e-01	9.92e-01	9.85e-01

we selected three representative benchmarks: ARC Challenge (Clark et al., 2018) and TruthfulQA Multiple Choice (Lin et al., 2022) (both multiple-choice tasks), along with the generative task MTBench (Zheng et al., 2023). These benchmarks cover diverse areas including humanity, STEM, reasoning, writing, math, and coding.

Others Frequency threshold $\theta = 5 \times 10^{-5}$, $n' = 3$, smoothing parameter $\alpha = 0.3$, inverse watermark strength $\delta' = 2.5$ (adaptive control strategy for δ' can be found in Appendix F). We use Dipper (Krishna et al., 2023) as the paraphraser.

4.2 Effectiveness of Watermark Removal

Main Results Table 2 demonstrates the effectiveness of the three proposed watermark removal methods across different settings. It is evident that both TP and WN methods successfully eliminate the inherited watermark in all cases, maintaining confidence levels similar to unwatermarked conditions. The UP method also contributes to watermark removal; however, due to its lack of specificity, it fails to achieve complete removal when the watermark learned by the student model is strong (i.e., KGW $n = 1, 2$).

Weight Ablation Study Figure 5 compares watermark removal effectiveness of WN between

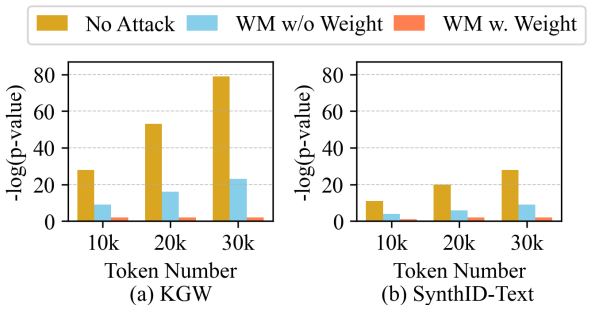


Figure 5: Comparison of watermark removal effectiveness: frequency-based prefix weighting vs. uniform weighting strategies.

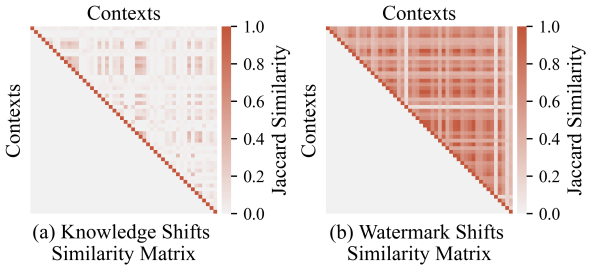
frequency-based and uniform prefix weighting (using $n = 2$). The results show that frequency-based prefix weighting, which assigns higher weights to more easily learned p-rules, achieves better watermark removal while maintaining an equal total weight across prefixes.

4.3 Performance of Knowledge Preservation

Main Results Table 3 shows that training on 200k watermarked teacher samples significantly improves the student model's performance across all benchmarks (Trained SM vs Ori. SM), regardless of watermarking scheme or window size n. When applying removal methods, UP and TP generally degrade performance, especially on generative

Table 3: Comparison of student model performance across benchmarks under different scenarios: no attack (Trained SM), UP, TP and WN. Values in () indicate percentage changes relative to Trained SM, with the highest performance in each setting bolded and underlined. Student model used in this table is Llama-7b, results for Llama-3.2-1b can be found in Appendix C.2.

Benchmark	Ori. SM	Wat. Scheme		Trained SM	Trained SM + UP	Trained SM + TP	Trained SM + WN
ARC Challenge (ACC)	0.4181	KGW	$n=1$	0.4480	0.4215 (-5.9%)	0.3951 (-11.8%)	**0.4497** (+0.6%)
			$n=2$	**0.4404**	0.4283 (-2.7%)	0.4104 (-6.8%)	0.4369 (-0.8%)
			$n=3$	**0.4778**	0.3865 (-19.1%)	0.3840 (-19.6%)	0.4642 (-2.8%)
		SynthID -Text	$n=1$	0.4505	0.4394 (-2.5%)	0.4198 (-6.8%)	**0.4548** (+1.0%)
			$n=2$	0.4360	0.4403 (+1.0%)	0.4241 (-2.7%)	**0.4565** (+4.7%)
			$n=3$	**0.4505**	0.4394 (-2.5%)	0.4283 (-4.9%)	0.4471 (-0.8%)
TruthfulQA Multiple Choice (ACC)	0.3407	KGW	$n=1$	0.3884	0.3917 (+0.8%)	0.3785 (-2.5%)	**0.4186** (+7.8%)
			$n=2$	**0.4376**	0.4097 (-6.4%)	0.4089 (-6.6%)	0.4353 (-0.5%)
			$n=3$	0.4459	0.4315 (-3.2%)	0.4055 (-9.1%)	**0.4632** (+3.9%)
		SynthID -Text	$n=1$	0.4063	0.3780 (-7.0%)	0.3597 (-11.5%)	**0.4262** (+4.9%)
			$n=2$	0.3991	0.3965 (-0.7%)	0.4043 (+1.3%)	**0.4281** (+7.3%)
			$n=3$	0.4102	0.4009 (-2.3%)	0.4062 (-1.0%)	**0.4330** (+5.3%)
MTBench (Full Score: 10)	2.64	KGW	$n=1$	**3.86**	3.04 (-21.2%)	2.76 (-28.5%)	3.67 (-4.9%)
			$n=2$	3.99	3.40 (-14.8%)	2.94 (-26.3%)	**4.02** (+0.7%)
			$n=3$	**4.11**	3.27 (-20.4%)	3.04 (-26.0%)	3.99 (-2.9%)
		SynthID -Text	$n=1$	**4.14**	3.27 (-21.0%)	2.01 (-51.4%)	4.13 (-0.2%)
			$n=2$	**4.24**	3.05 (-28.1%)	2.84 (-33.0%)	4.12 (-2.8%)
			$n=3$	**4.24**	2.90 (-31.6%)	2.69 (-36.6%)	4.16 (-1.9%)



(a) Knowledge Shifts Similarity Matrix (b) Watermark Shifts Similarity Matrix

Figure 6: Pairwise similarities of probability prediction shifts across different contexts when the last token is fixed as "the", showing (a) knowledge shifts similarity and (b) watermark shifts similarity.

tasks like MTBench, with TP showing larger degradation than UP. WN effectively maintains knowledge - compared to the trained SM, it improves performance in about half the cases and shows minor decreases (under 5%) in others, performing similarly to direct training without attacks.

Why WN Can Achieve Good Knowledge Preservation WN achieves superior knowledge preservation for two key reasons. First, it eliminates the need for external rewriting tools. Since the teacher model represents our highest quality data source, avoiding external rewriting prevents data quality degradation. Second, our experiments reveal distinct patterns between knowledge and watermark learning. Knowledge learning depends on broader context, while watermark learning only relies on the previous $n-1$ tokens. Using $n=2$, we analyze 1,000 distinct text segments ending with "the" (*ctx the*) and compare: (1) **Knowledge shifts**: Probability differences between models before and after training on non-watermarked data show high variation across contexts (Figure 6(a)); (2) **Watermark shifts**: Probability differences between models trained on non-watermarked and watermarked data (generated using the same teacher and prompts) exhibit high consistency across different *ctx* when the last token is fixed (Figure 6(b)).

Therefore, using prompts with fixed ending tokens to get averaged probability shifts mainly captures watermark patterns, while knowledge-related shifts tend to cancel out during averaging, resulting in minimal impact.

5 Further Analysis

5.1 Computational Overhead

To conduct a thorough assessment of practical applications, this section presents a detailed analysis of the operational overhead associated with the three proposed watermark removal methods, as summarized in Table 4.

External Tools UP and TP require external paraphrasing tools, while WN needs none.

Preprocessing Time Consumption UP requires Dipper-like models to paraphrase the entire training dataset, WN only needs watermark stealing for pre-

Table 4: Overhead comparison of watermark removal methods, averaged across various watermark settings using Llama-7b as the student model. All experiments were conducted on 8 NVIDIA H800 GPUs.

	UP	TP	WN
Required External Tools	✓	✓	✗
Preprocessing Time (h)	31.8	37.1	3.6
Inference Latency (s / token)	0.0000	0.0000	0.0068

Table 5: Watermark detection results in multi-source settings: 2 teacher models (1) employing KGW scheme with opposing keys; (2) using KGW and SynthID-Text respectively. Single-source results are shown in ().

	KGW k + KGW $\overline{k}$	KGW + SynthID-Text
Detector 1	2.81e-01 (4.21e-28)	5.64e-09 (4.21e-28)
Detector 2	7.19e-01 (4.21e-28)	3.48e-03 (6.67e-11)

processing, and TP requires both. The time cost of watermark stealing consists of student model training (about 1h for Llama-7b) and dataset forward passes. The forward passes are computationally much cheaper than paraphrasing, as parallel computation is faster than autoregressive generation.

Student Model Inference Latency UP and TP have no additional inference overhead. WN requires adding inverse watermark during inference, causing slight delay.

5.2 Multi-Source Knowledge Distillation

The previous analysis focused on single-source knowledge distillation. In real-world scenarios with multiple LLM services, we found that watermarks from different sources can collide and counteract each other during knowledge distillation, making watermarks less effective as a protection mechanism - even without any removal methods.

Case 1: Two Opposing Keys We tested an extreme case with two teachers using the KGW scheme with complementary keys (the hash results were complete opposites). When we trained a student model using a combined dataset (100k samples from each teacher), the watermark detection confidence dropped significantly from e-28 to e-01, as shown in Table 5.

Case 2: Two Watermarking Schemes We examined a scenario using two teacher models with different watermarking methods (KGW and SynthID-Text), each generating 100k samples. Training a student model on this combined dataset led to reduced watermark detection confidence for both schemes' detectors compared to single-source scenarios (as shown in Table 5).

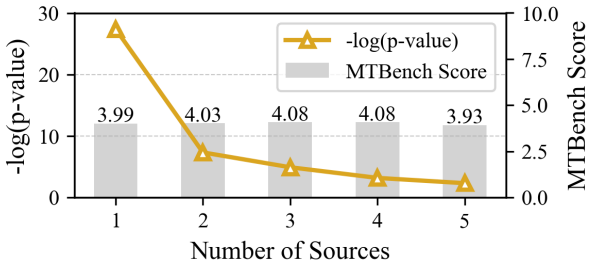


Figure 7: Average -log p values from detectors and MTBench scores as the number of teacher sources increases.

Case 3: Multi-Source In this scenario, all teacher models employ the KGW scheme with randomly selected keys. As shown in Figure 7, while increasing the number of source models and maintaining a constant total volume of mixed training data, watermark detection became increasingly difficult, yet model performance remained stable. This suggests that mixing data from a sufficient number of teacher sources can achieve untraceable knowledge distillation.

5.3 Future Directions in Defense Strategy

This work reveals the vulnerability of unauthorized knowledge distillation prevention when generative LLM watermarking is predominantly confined to n-gram based approaches. While alternatives such as sentence-level reject sampling (Hou et al., 2023, 2024) and post-generation signal embedding (Chang et al., 2024) exist, these approaches introduce significant latency to the current token-by-token real-time LLM inference paradigm, making them difficult to deploy at scale in real-time LLM services. We therefore advocate for diversifying token-level watermarking techniques, as multiple paradigms would make it harder for attackers to identify and target specific methods, enabling more robust protection.

6 Conclusion

This work presents the first systematic study of the robustness of watermarking schemes against adversarial attacks in preventing unauthorized knowledge distillation. We propose three watermark removal approaches: two pre-distillation methods (UP, TP) and one post-distillation method (WN). Through comprehensive experiments, we evaluate the resilience of watermarking schemes against these attacks. Our findings reveal that WN achieves effective watermark removal while maintaining superior knowledge preservation, highlighting the urgent need for more robust defensive strategies.

Limitations

While our study presents a systematic investigation of watermark resilience against adversarial attacks under the scenario of preventing unauthorized knowledge distillation, there still exist several limitations. Due to computational constraints, we only evaluated one teacher model (GLM-4-9b-chat) and two student models (Llama-7b and Llama-3.2-1b). The experiments were conducted using a fixed training dataset size of 200,000 samples and tested primarily on English language tasks. Additionally, our evaluation metrics focused on standard benchmarks (ARC, TruthfulQA, MTBench) and may not fully reflect performance on specialized domain tasks. Future work could explore a broader range of model architectures, training data scales, and task domains.

References

S. Aaronson and H. Kirchner. 2022. Watermarking gpt outputs. https://www.scottaaronson.com/talks/watermark.ppt.

Yapei Chang, Kalpesh Krishna, Amir Houmansadr, John Wieting, and Mohit Iyyer. 2024. Postmark: A robust blackbox watermark for large language models. *arXiv preprint arXiv:2406.14517*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, Jamie Hayes, Nidhi Vyas, Majd Al Merey, Jonah Brown-Cohen, Rudy Bunel, Borja Balle, Taylan Cemgil, Zahra Ahmed, Kitty Stacpoole, Ilia Shumailov, Ciprian Baetu, Sven Gowal, Demis Hassabis, and Pushmeet Kohli. 2024. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823.

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

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. 2024. On the learnability of watermarks for language models. In *The Twelfth International Conference on Learning Representations*.

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

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

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 *The Twelfth International Conference on Learning Representations*.

Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. In *Forty-first International Conference on Machine Learning*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2024. On the reliability of watermarks for large language models. In *The Twelfth International Conference on Learning Representations*.

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. *arXiv preprint arXiv:2303.13408*.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2024. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*.

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

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

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

Aiwei Liu, Leyi Pan, Xuming Hu, Shuang Li, Lijie Wen, Irwin King, and Philip S. Yu. 2024b. An unforgeable publicly verifiable watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2024c. A semantic invariant robust watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

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*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Tom Sander, Pierre Fernandez, Alain Oliviero Durmus, Matthijs Douze, and Teddy Furon. 2024. Watermarking makes language models radioactive. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Qilong Wu and Varun Chandrasekaran. 2024. Bypassing LLM watermarks with color-aware substitutions. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8549–8581, Bangkok, Thailand. Association for Computational Linguistics.

Yihan Wu, Zhengmian Hu, Hongyang Zhang, and Heng Huang. 2023. Dipmark: A stealthy, efficient and resilient watermark for large language models. *arXiv preprint arXiv:2310.07710*.

Xi Yang, Jie Zhang, Kejiang Chen, Weiming Zhang, Zehua Ma, Feng Wang, and Nenghai Yu. 2022. Tracing text provenance via context-aware lexical substitution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11613–11621.

Zhaoxi Zhang, Xiaomei Zhang, Yanjun Zhang, Leo Yu Zhang, Chao Chen, Shengshan Hu, Asif Gill, and Shirui Pan. 2024. Large language model watermark stealing with mixed integer programming. *arXiv preprint arXiv:2405.19677*.

Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. 2024. Provable robust watermarking for AI-generated text. In *The Twelfth International Conference on Learning Representations*.

Xuandong Zhao, Yu-Xiang Wang, and Lei Li. 2023. Protecting language generation models via invisible watermarking. *arXiv preprint arXiv:2302.03162*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Barret Zoph, Colin Raffel, Dale Schuurmans, Dani Yogatama, Denny Zhou, Don Metzler, Ed H Chi, Jason Wei, Jeff Dean, Liam B Fedus, et al. 2022. Emergent abilities of large language models. *TMLR*.

Appendices

Table of Contents

A Selected Terms of Use for LLM Services **12**

B Details of Watermarking Schemes **12**
B.1 KGW 12
B.2 SynthID-Text 13

C Experiment Results for Llama-3.2-1b **14**
C.1 Results of Watermark Removal Effectiveness for Llama-3.2-1b 14
C.2 Results of Knowledge Preservation Performance for Llama-3.2-1b 15

D Experiment Results for More Watermarking Schemes **15**
D.1 Experiment Results for More N-gram based Watermarking Schemes 15
D.2 Discussion About Other Watermarking Paradigms 17

E Impact of Inference-Time Watermark Neutralization (WN) on Knowledge Preservation under Non-watermarked Setting **18**

F Adaptive Control for Inverse Watermark Strength **18**

G Details of Training Data Collection **19**
G.1 Prompt Used for Training Data Collection 19
G.2 Example Samples of Training Data 19

A Selected Terms of Use for LLM Services

Figure 8 shows excerpts from terms of use across various leading LLM services, including OpenAI[2], Anthropic[3] and Meta Llama[4]. These terms explicitly prohibit using model outputs for training or improving other models.

OpenAI. What you cannot do.

Use output to develop models that compete with OpenAI.

Anthropic. You may not access or use, or help another person to access or use, our Services in the following ways:

To develop any products or services that compete with our Services, including to develop or train any artificial intelligence or machine learning algorithms or models.

Meta (Llama 2). License Rights and Redistribution.

You will not use the Llama Materials or any output or results of the Llama Materials to improve any other large language model (excluding Llama 2 or derivative works thereof).

Figure 8: Selected terms of use for various LLM services.

B Details of Watermarking Schemes

B.1 KGW

Watermarking KGW (Kirchenbauer et al., 2023) is a fundamental scheme of LLM watermarking. For generating the t-th token, the algorithm examines the previous $n-1$ tokens: $x_{t-n+1:t-1}$. These tokens are fed into a hash function H to produce $h_t = H(x_{t-n+1:t-1})$. Based on h_t, the vocabulary $\mathcal{V}$ is deterministically split into a *green list* $\mathcal{V}_g$ and a *red list* $\mathcal{V}_r$. A constant bias δ is applied to logits of green tokens according to:

$$l'^{(i)}_t = \begin{cases} l^{(i)}_t + \delta & \text{if } v_i \in \mathcal{V}_g \\ l^{(i)}_t & \text{if } v_i \in \mathcal{V}_r \end{cases} \tag{11}$$

Detection Given a text sequence of length T, we count the number of green tokens $|s|_G$. Let $\gamma = |\mathcal{V}_g|/|\mathcal{V}|$ represent the expected proportion of green tokens in random text. The statistical significance of the green token count is measured by the z-score:

$$z = \frac{|s|_G - \gamma T}{\sqrt{\gamma(1-\gamma)T}} \tag{12}$$

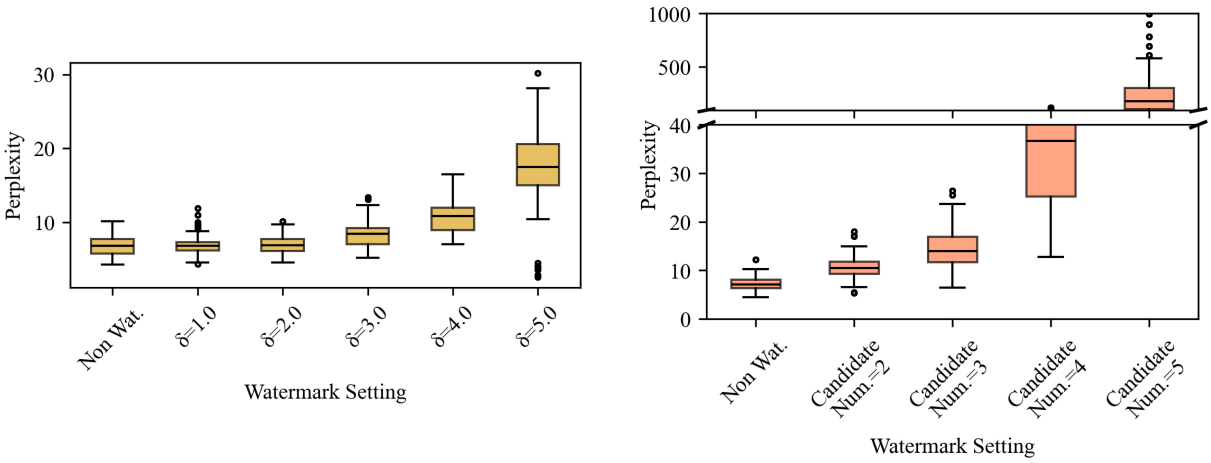
For a fixed δ ($\delta > 0$), longer sequences lead to stronger detection signal, as the z-score increases with text length T.

In our experiments, we set $\delta = 3.0$ and $\gamma = 0.5$, which represents a relatively strong watermark configuration in typical KGW settings. We avoid using larger δ values since stronger watermarks would notably degrade the text quality (as shown in Figure 9a), making them impractical for real-world LLM services.

Watermark Confidence: p-value Under the null hypothesis (non-watermarked text), the z-score follows a standard normal distribution $\mathcal{N}(0,1)$. The p-value can be computed as:

$$p = 1 - \Phi(z). \tag{13}$$

[2]https://openai.com/policies/terms-of-use/
[3]https://www.anthropic.com/legal/consumer-terms
[4]https://ai.meta.com/llama/license/



(a) KGW

(b) SynthID-Text

Figure 9: The relationship between watermark strength settings and perplexity under different watermarking schemes. The model used for calculating PPL is Llama-3.1-70B (Dubey et al., 2024).

Smaller p-value suggests higher confidence of watermark presence, as it indicates that the proportion of green tokens significantly exceeds what would be expected by chance.

B.2 SynthID-Text

Watermarking SynthID-Text (Dathathri et al., 2024) employs a tournament-based watermarking approach during the token generation process. For generating the t-th token, the algorithm first generates a random seed h_t by applying a hash function H to the previous $n-1$ tokens: $h_t = H(x_{t-n+1:t-1})$. This seed initializes m independent g functions $g_1, g_2, ..., g_m$, which assign binary values (0 or 1) to each token in the vocabulary.

The core watermarking process involves a tournament with m layers. Initially, 2^m candidate tokens are sampled from the language model's original probability distribution $P(x_t|x_{1:t-1})$. These tokens undergo a series of pairwise competitions through m tournament layers. In each layer ℓ, tokens are randomly paired, and within each pair, the token with the higher g_ℓ score advances to the next layer, with random tie-breaking. The final surviving token after m layers becomes the output token x_t.

Detection Detection in SynthID-Text relies on measuring the statistical signature introduced during the watermarking process. Given a piece of text $x = x_1, ..., x_T$, the detection algorithm:

1. Reconstructs the random seeds h_t for each position t using the same hash function and watermarking key
2. Computes the g-values for each token using the same watermarking functions
3. Calculates the mean score across all positions and layers:

$$\overline{g} = \frac{1}{mT} \sum_{t=1}^{T} \sum_{\ell=1}^{m} g_\ell(x_t). \tag{14}$$

Due to the tournament selection process, watermarked text tends to contain tokens with higher g-values compared to non-watermarked text. Several factors contribute to a stronger detection signal: increasing the number of g functions (larger m), using more candidates in each round of tournament sampling, and extending the sequence length for detection.

In our experiment, we set $m = 30$ and use 2 candidates per round in tournament sampling, following the default configuration in SynthID-Text paper. We avoid using larger candidates number since stronger watermarks would notably degrade the text quality (as shown in Figure 9b), making them impractical for real-world LLM services. However, we built upon this foundation by implementing the distortionary

Table 6: Median p-values for watermark detection using **UP** (Untargeted Training Data Paraphrasing), **TP** (Targeted Training Data Paraphrasing), and **WN** (Watermark Neutralization), compared against direct training (No Attack) and unwatermarked conditions (Unw.). indicates high watermark confidence, indicates low watermark confidence, and unshaded cells indicate insufficient evidence for watermark presence. Student model used in this table is Llama-3.2-1b.

Watermarking Scheme		Token Num.	Unw.	No Attack	UP	TP	WN
KGW	$n = 1$	1k	5.75e-01	1.26e-36	3.27e-03	9.98e-01	6.21e-01
		2k	5.71e-01	3.59e-71	6.00e-05	9.95e-01	7.00e-01
		3k	6.01e-01	1.28e-104	1.03e-06	9.98e-01	7.34e-01
	$n = 2$	10k	4.80e-01	4.79e-65	1.00e-03	8.14e-01	5.20e-01
		20k	4.47e-01	5.96e-128	6.42e-06	6.79e-01	4.97e-01
		30k	3.62e-01	4.68e-190	3.59e-08	8.92e-01	5.48e-01
	$n = 3$	100k	3.40e-01	6.85e-12	6.96e-02	1.98e-01	8.13e-01
		300k	3.41e-01	2.33e-27	1.09e-02	5.19e-01	9.43e-01
		1 million	4.84e-01	1.13e-87	9.80e-05	2.04e-01	9.98e-01
SynthID-Text	$n = 1$	1k	9.67e-01	3.76e-10	2.55e-01	9.98e-01	9.94e-01
		2k	9.95e-01	9.00e-19	1.66e-01	9.87e-01	9.96e-01
		3k	9.99e-01	1.72e-27	1.26e-01	9.89e-01	9.98e-01
	$n = 2$	10k	4.23e-01	1.29e-32	4.81e-03	2.93e-01	4.58e-01
		20k	3.82e-01	1.48e-63	1.11e-04	1.83e-01	4.77e-02
		30k	3.09e-01	2.22e-95	4.28e-06	3.67e-01	1.51e-02
	$n = 3$	100k	9.98e-01	2.41e-05	9.93e-01	9.99e-01	8.95e-01
		300k	9.76e-01	2.47e-14	9.94e-01	9.99e-01	9.97e-01
		1 million	9.87e-01	1.85e-32	9.97e-01	9.99e-01	9.98e-01

version of SynthID-Text, which enhances watermark strength. The non-distortionary version employs repeated context masking, where watermarks are only applied to subsequent tokens upon the first encounter with a particular prefix, while original sampling is used for subsequent occurrences of the same prefix. This approach ensures unbiased multi-step sampling and maintains text quality at the cost of watermark strength. In contrast, the distortionary version foregoes repeated context masking, sacrificing some text quality but achieving stronger watermarks.

Watermark Confidence: p-value For a given text x and its score $\overline{g}$, the p-value is calculated based on the following principles: Under the null hypothesis (text contains no watermark), $\overline{g}$ approximately follows a normal distribution according to the Central Limit Theorem. This distribution has a mean $\mu = 0.5$ since g functions output 0 and 1 with equal probability for non-watermarked text. The variance σ^2 is estimated as $\frac{1}{4mT}$, where m is the number of g functions and T is the text length. The p-value is then computed as:

$$p = 1 - \Phi\left(\frac{\overline{g} - 0.5}{1/\sqrt{4mT}}\right) \tag{15}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution. A smaller p-value indicates stronger evidence that the text contains a watermark.

C Experiment Results for Llama-3.2-1b

This section provides supplementary experimental results for Llama-3.2-1b, including the effectiveness of watermark removal and knowledge preservation performance.

C.1 Results of Watermark Removal Effectiveness for Llama-3.2-1b

Table 6 demonstrates the effectiveness of the three proposed watermark removal approaches across various settings, with Llama-3.2-1b serving as the student model. The experimental results align closely with those obtained using Llama-7b as the student model, as discussed in the main text: both targeted training

Table 7: Comparison of student model performance across benchmarks under different scenarios: no attack (Trained SM), UP, TP and WN. Values in () indicate percentage changes relative to Trained SM, with the highest performance in each setting bolded and underlined. Student model used in this table is Llama-3.2-1b.

Benchmark	Ori. SM	Wat. Scheme		Trained SM	Trained SM + UP	Trained SM + TP	Trained SM + WN
ARC Challenge (ACC)	0.3166	KGW	$n=1$	0.3404	0.3430 (+0.8%)	0.3259 (-4.3%)	**0.3660** (+7.5%)
			$n=2$	**0.3515**	0.3200 (-9.0%)	0.2841 (-19.2%)	0.3498 (-0.5%)
			$n=3$	**0.3712**	0.3072 (-17.2%)	0.2935 (-20.9%)	0.3626 (-2.3%)
		SynthID -Text	$n=1$	0.3541	0.3259 (-8.0%)	0.3242 (-8.4%)	**0.3626** (+2.4%)
			$n=2$	0.3481	0.3345 (-3.9%)	0.3200 (-8.1%)	**0.3575** (+2.7%)
			$n=3$	**0.3532**	0.3396 (-3.9%)	0.3251 (-8.0%)	0.3498 (-1.0%)
TruthfulQA Multiple Choice (ACC)	0.3768	KGW	$n=1$	0.3783	0.3815 (+0.8%)	0.3796 (+0.3%)	**0.3820** (+1.0%)
			$n=2$	0.4022	0.3719 (-7.5%)	0.3887 (-3.4%)	**0.4026** (+0.1%)
			$n=3$	0.4061	0.3949 (-2.8%)	0.3862 (-4.9%)	**0.4411** (+8.6%)
		SynthID -Text	$n=1$	0.4023	0.3869 (-3.8%)	0.3497 (-13.1%)	**0.4027** (+0.1%)
			$n=2$	0.3912	0.3893 (-0.5%)	0.3921 (+0.2%)	**0.4177** (+6.8%)
			$n=3$	0.4140	0.3883 (-6.2%)	0.3941 (-4.8%)	**0.4224** (+2.0%)
MTBench (Full Score: 10)	2.78	KGW	$n=1$	2.88	1.87 (-35.1%)	1.27 (-55.9%)	**3.20** (+11.1%)
			$n=2$	**2.90**	1.31 (-54.8%)	1.19 (-59.0%)	2.85 (-1.7%)
			$n=3$	**2.88**	1.30 (-54.9%)	1.16 (-59.7%)	**2.88** (+0.0%)
		SynthID -Text	$n=1$	**3.21**	1.51 (-53.0%)	1.26 (-60.7%)	3.19 (-0.6%)
			$n=2$	**3.11**	1.29 (-58.5%)	1.31 (-57.9%)	2.83 (-9.0%)
			$n=3$	**3.09**	1.28 (-58.6%)	1.27 (-58.9%)	2.98 (-3.6%)

data paraphrasing (TP) and inference-time watermark neutralization (WN) successfully eliminate the watermark completely, while untargeted training data paraphrasing (UP) shows some removal effect but fails to achieve complete elimination across all scenarios.

C.2 Results of Knowledge Preservation Performance for Llama-3.2-1b

Table 7 reveals that Llama-3.2-1b, when trained on 200,000 watermarked samples from the teacher model, achieves substantial performance gains across all benchmarks. As for knowledge preservation performance of the three proposed watermark removal methods, the results of Llama-3.2-1b echo the main experimental results: while both UP and TP lead to widespread performance degradation under most configurations, WN stands out for its remarkable ability to preserve knowledge. Specifically, WN yields performance improvements in roughly half of the settings while showing slight decreases in the remaining scenarios, and these changes remain small throughout.

D Experiment Results for More Watermarking Schemes

The main experiments focused on two representative watermarking schemes, KGW (Kirchenbauer et al., 2023) and SynthID-Text (Dathathri et al., 2024), for testing. This section presents supplementary experiments with additional n-gram based watermarking schemes to demonstrate the strong generalization capability of the proposed method. Moreover, we will discuss several other watermarking paradigms that represent alternative approaches in this field.

D.1 Experiment Results for More N-gram based Watermarking Schemes

MinHash This method is a variant of KGW, proposed by Kirchenbauer et al. (2024). In the default implementation of KGW, the hash function H is a multiplicative modular function, expressed as:

$$h_t = H(x_{t-n+1:t-1}) = \prod_{i=t-n+1}^{t-1} x_i \bmod |\mathcal{V}|. \tag{16}$$

This approach causes the hash result to change whenever any token within the window is modified, leading to reduced robustness as the window size n increases. To address this limitation, several improved versions

Table 8: Median p-values for watermark detection using **WN** (Watermark Neutralization), compared against direct training (No Attack) and unwatermarked conditions (Unw.). indicates high watermark confidence, indicates low watermark confidence, and unshaded cells indicate insufficient evidence for watermark presence. The watermarking schemes used in this table are MinHash and SkipHash.

Watermarking Scheme	Model	Token Num.	Unw.	No Attack	WN
MinHash $n = 3$	Llama-7b	100k	7.87e-01	2.32e-16	8.41e-01
		300k	9.23e-01	4.52e-47	9.28e-01
		1 million	9.97e-01	2.79e-141	9.98e-01
	Llama-3.2-1b	100k	7.87e-01	6.28e-44	9.89e-01
		300k	9.23e-01	1.14e-134	9.96e-01
		1 million	9.97e-01	4.56e-385	9.98e-01
SkipHash $n = 3$	Llama-7b	100k	9.98e-01	2.80e-05	8.91e-01
		300k	9.98e-01	1.85e-13	9.84e-01
		1 million	9.98e-01	4.13e-39	9.92e-01
	Llama-3.2-1b	100k	9.98e-01	1.07e-07	9.91e-01
		300k	9.98e-01	4.01e-18	9.97e-01
		1 million	9.98e-01	3.45e-47	9.98e-01

Table 9: Comparison of student model performance across benchmarks under no attack scenario (Trained SM) and using WN for watermark removal. Values in () indicate percentage changes relative to Trained SM, with the highest performance in each setting bolded and underlined.

Benchmark	Student Model	Ori. SM	Wat. Scheme		Trained SM	Trained SM + WN
ARC Challenge (ACC)	Llama-7b	0.4181	MinHash	$n = 3$	**0.4505**	0.4471 (-0.8%)
			SkipHash	$n = 3$	**0.4625**	**0.4625** (+0.0%)
	Llama-3.2-1b	0.3166	MinHash	$n = 3$	0.3532	**0.3567** (+1.0%)
			SkipHash	$n = 3$	0.3609	**0.3618** (+0.2%)
TruthfulQA Multiple Choice (ACC)	Llama-7b	0.3407	MinHash	$n = 3$	0.4836	**0.4872** (+0.7%)
			SkipHash	$n = 3$	0.4562	**0.4624** (+1.4%)
	Llama-3.2-1b	0.3768	MinHash	$n = 3$	0.4153	**0.4187** (+0.8%)
			SkipHash	$n = 3$	0.4037	**0.4069** (+0.8%)
MTBench (Full Score: 10)	Llama-7b	2.64	MinHash	$n = 3$	**3.75**	3.73 (-0.5%)
			SkipHash	$n = 3$	**4.11**	4.04 (-1.7%)
	Llama-3.2-1b	2.78	MinHash	$n = 3$	**3.03**	2.91 (-4.0%)
			SkipHash	$n = 3$	**3.46**	3.37 (-2.6%)

have been proposed, including MinHash, which uses the minimum token id within the window as the hash result:

$$h_t = H(x_{t-n+1:t-1}) = \min_{i \in [t-n+1, t-1]} x_i \tag{17}$$

SkipHash SkipHash (Kirchenbauer et al., 2024) is also a variant of KGW designed to improve robustness, but it uses a hash function that takes the leftmost token id within the window, expressed as:

$$h_t = H(x_{t-n+1:t-1}) = x_{t-n+1} \tag{18}$$

When $n \leq 2$, MinHash and SkipHash are equivalent to KGW, so we only evaluate scenarios where $n = 3$. Given WN's superior overall performance among the three proposed methods in terms of watermark removal effectiveness and knowledge preservation, the subsequent experiments exclusively focus on this approach. The experimental results are shown in Table 8 and Table 9. The observed trends in the experimental results align consistently with the main experiment, which uses KGW and SynthID-Text. The WN approach demonstrates complete watermark removal efficacy while showing no significant impact on the knowledge acquired by the student model.

D.2 Discussion About Other Watermarking Paradigms

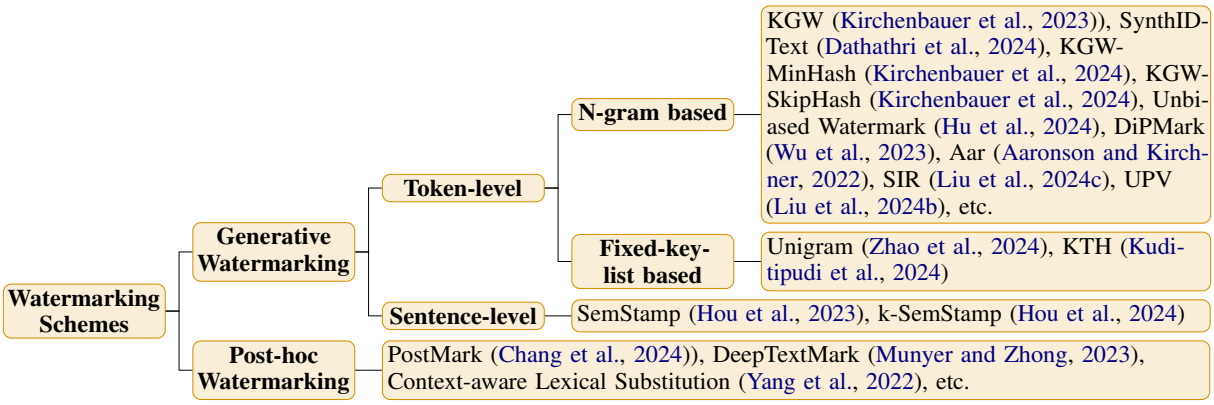


Figure 10: Taxonomy of existing watermarking schemes.

As illustrated in Figure 10, current watermarking schemes can be categorized into two main approaches: generative watermarking, where watermarks are embedded during the text generation process, and post-hoc watermarking, where watermarks are added to existing texts. Within generative watermarking, there are further subdivisions into token-level methods and sentence-level approaches based on reject sampling. In real-time services, tokens can be outputted as they are sampled while inference continues, thereby enhancing user experience. Post-hoc watermarking and sentence-level watermarking introduce significant latency, making them less suitable for real-time LLM services compared to token-level watermarking.

Among token-level methods, the predominant paradigm is the n-gram based approach. Additionally, there are few methods that employ a fixed-key-list based approach, which utilizes global fixed watermark keys independent of the prefix. These methods include Unigram (Zhao et al., 2024) and KTH (Kuditipudi et al., 2024), for which we conduct supplementary experiments.

Unigram (Zhao et al., 2024) This approach is equivalent to KGW with prefix length of 0, using a globally fixed red-green partition. According to our main experimental results, both TP and WN can completely remove the watermarks inherited by the student model.

KTH (Kuditipudi et al., 2024) This method employs a globally fixed sequence of watermark keys: $\xi = \xi^{(0)}, \xi^{(1)}, ..., \xi^{(m-1)}$, where each $\xi^{(j)} \in [0,1]^{|\mathcal{V}|}$ follows a uniform distribution. During text generation, first a random shift $s \in [0, m)$ is selected, then the t-th generated token is chosen using the following strategy:

$$x_t = \arg\max_i (\xi_i^{(s+t \mod m)})^{1/p_t}, \tag{19}$$

where p_t is the original probability prediction at position t. During watermark detection, it computes the minimum Levenshtein distance d between the text to be detected x and the key sequences ξ. In comparison, it randomly generates n sequences with the same shape as x_i, and calculates d'_1, d'_2, ..., d'_n using the same method. The detection p-value is represented by the proportion of values in the d' sequence that are lower than d. It is worth noting that the p-value of this detection method is bounded by the number of trials n.

Table 10: Median p-value of watermark detection in trained student models using KTH watermarking scheme ($m = 256$, token num. = 256, $n = 100$).

Watermarking Scheme	Student Model	Median p-value (No Attack)
KTH	Llama-7b	3.1e-01
	Llama-3.2-1b	3.2e-01

Our experiments revealed that the addition of KTH watermark significantly affects the instruction-following capability of the teacher model, resulting in a lower proportion of QA pairs conforming to

format rules and generally shorter answers. After training on such data, the watermark is barely detectable in the student model, as shown in Table 10.

E Impact of Inference-Time Watermark Neutralization (WN) on Knowledge Preservation under Non-watermarked Setting

Previous experiments have demonstrated that when teacher model is watermarked, inference-time watermark neutralization (WN) effectively enables the student model to bypass the watermark while acquiring knowledge from the teacher model's outputs that is comparable to what would be learned without any attack. In this section, we conducted additional experiments to examine whether applying WN would affect the knowledge acquired by the student model in cases where the teacher model itself is not watermarked (noting that the student model has no access to the detector and thus cannot determine the presence of watermarks).

Consistent with the settings in our main experiments, we generated QA pairs using GLM-4-9b-chat. After filtering and deduplication, we obtained 200,000 non-watermarked samples. We trained both Llama-7b and Llama-3.2-1b models using this dataset, and then applied WN for watermark removal and evaluated the performance changes across various benchmarks, with results presented in Table 11. It can be concluded that WN *does not* exert a substantial negative impact on knowledge preservation under non-watermarked setting.

Table 11: Impact of WN on knowledge preservation under non-watermarked setting.

Benchmark	Student Model	Ori. SM	Trained SM	Trained SM + WN
ARC Challenge (ACC)	Llama-7b	0.4181	0.4454	0.4488 (+0.7%)
	Llama-3.2-1b	0.3166	0.3524	0.3507 (-0.5%)
TruthfulQA Multiple Choice (ACC)	Llama-7b	0.3407	0.4254	0.4507 (+5.9%)
	Llama-3.2-1b	0.3768	0.4052	0.3951 (-2.5%)
MTBench (Full Score: 10)	Llama-7b	2.64	3.99	4.12 (+3.2%)
	Llama-3.2-1b	2.78	3.15	3.08 (-2.2%)

F Adaptive Control for Inverse Watermark Strength

Throughout all previous experiments, we consistently used an inverse watermark strength of $\delta' = 2.5$ for WN, which achieved complete watermark removal in all cases. As detailed in Appendix B, the chosen watermark strength of the teacher model represents a notably high intensity that remains practical for deployment in LLM services, suggesting that $\delta' = 2.5$ is sufficient for the vast majority of scenarios.

However, to account for potential extreme cases, we also explored strategies for adaptive control of inverse watermark strength. Our approach is to estimate the required inverse watermark strength δ' by detecting the watermark intensity inherited by the student model. Since the student model holder does not have access to the watermark detector, we employed Water-Probe (Liu et al., 2024a) to measure watermark intensity. Water-Probe is a recently proposed identification algorithm that tests for watermarks by comparing the model's responses to specially crafted prompts, where higher similarity in responses to crafted prompts pairs indicates a higher likelihood (or strength) of watermarking.

We conducted experiments using KGW with $n = 1$. Table 12 shows the cosine similarity scores detected by Water-Probe-v2[5] for Llama-7b student models trained with different watermark strengths δ, compared with a student model trained on unwatermarked data. Based on this reference table, we can estimate the watermark strength δ used in the teacher model by examining the WaterProbe-v2 cosine similarity score of the trained student model. This estimation enables us to adaptively select an appropriate inverse watermark strength δ' for removal.

Here is a practical example: Suppose the teacher model is watermarked using KGW with $n = 1, \delta = 5.0$. The trained student model's detected cosine similarity is 0.1694, which is slightly higher than the reference

[5]There are two versions of Water-Probe, with version 2 demonstrating more stable performance in our experiments.

Table 12: Water-Probe-v2 cosine similarity scores for student models under different watermark strength settings.

Settings	Unw.	KGW with Different δ				
		$\delta = 1.0$	$\delta = 2.0$	$\delta = 3.0$	$\delta = 4.0$	$\delta = 5.0$
Cosine Similarity	0.0065	0.0853	0.1032	0.1366	0.1544	0.1694

Table 13: Median p-values for watermark detection using WN under different inverse watermark strengths. The watermark used in teacher model is KGW, with $\delta = 5.0$, the student model is Llama-7b.

Token Number	**No Attack**	**WN** $\delta' = 2.5$	**WN** $\delta' = 3.0$
10k	7.75e-59	2.81e-03	9.17e-02
20k	7.48e-115	6.91e-05	4.29e-02
30k	1.14e-170	6.31e-07	1.13e-02

value of 0.1366 for $\delta = 3.0$. Given our prior knowledge that $\delta' = 2.5$ can completely remove watermarks with $\delta = 3.0$, we should proportionally increase the inverse watermark strength. Therefore, we set $\delta' = 3.0$ for this case. The removal results are shown in Table 13.

We acknowledge that the current estimation method is relatively rough. However, it's important to emphasize that in practical LLM services, it would be unrealistic to use such strong watermarks as $\delta = 5.0$, as this would significantly degrade the output quality. In most cases, selecting an inverse watermark strength of $\delta' = 2.5$ is already sufficient.

G Details of Training Data Collection

G.1 Prompt Used for Training Data Collection

Following the work by Sander et al. (2024), we prompted the teacher model to generate question-answering samples consisting of instruction, input and answer, as shown in the prompt template in Figure 11.

G.2 Example Samples of Training Data

Example samples of training data produced using different watermarking schemes and n are shown in Figure 12.

You are asked to come up with a set of 20 diverse task instructions and their answers. These instructions will be given to large language model and we will evaluate it for completing the instructions. Here are the requirements:

1. Try not to repeat the verb for each instruction to maximize diversity.
2. The language used for the instruction also should be diverse. For example, you should combine questions with imperative instructions.
3. The type of instructions should be diverse. The list should include diverse types of tasks like open-ended generation, classification, math reasoning, code generation, world common knowledge, reading comprehension, editing, etc.
2. A language model should be able to complete the instruction. For example, do not ask the assistant to create any visual or audio output. For another example, do not ask the assistant to wake you up at 5pm or set a reminder because it cannot perform any action.
3. The instructions should be in English.
4. The instructions should be 1 to 2 sentences long. Either an imperative sentence or a question is permitted.
5. You should generate an appropriate input to the instruction. It should contain a specific example for the instruction when needed.
6. Most of the instructions do not require input. In this case, you must simply put "None" in the input field.
7. The input should not be the answer to the instruction. For example, if the instruction is "Write a headline for ...", the input should not be the said headline. Instead, the input should be "None".
8. The output should be an appropriate response to the instruction and the input. Make sure the output is less than 100 words.
9. Do not repeat instructions.
10. Use [[,]] to flag the start and end of each example.

Examples of generating 3 task instruction and its answer:

[[Instuction: Give a detailed description of a method for brainstorming an essay.
Input:
Answer: Mind Mapping is one of the methods for brainstorming an essay. Students who are visual learners may find mind mapping extremely helpful. In order to Mind Mapping: Take a paper and draw a circle in the center Write your topic inside it. Just like we sketch rays protruding from a sun, draw lines coming out of the circle. Write an argument or a statement related to the topic at the end of each line. Draw another set of lines projecting from these statements and state examples that support them. Continue this exercise until you run out of all possibilities imaginable for this topic.]]

[[Instuction: Write a creative or imaginative response to the given prompt.
Input: You are a time traveler who has just arrived in ancient Egypt. What do you see?
Answer: Wow! As I step out of my time machine, I'm surrounded by towering pyramids and the warm sun beats down on my face. I see Pharaohs riding in grand chariots, and workers building majestic temples. It's like something out of a dream!]]

[[Instuction: Solve a math problem.
Input: A snail is at the bottom of a 20-foot well. Each day, it climbs up 3 feet, but at night, it slips back 2 feet. How many days will it take for the snail to reach the top of the well?
Answer: To solve this problem, we first calculate the net distance the snail covers each day: 3 feet up minus 2 feet down, resulting in 1 foot of progress daily. Since the well is 20 feet deep, we can set up the equation: number of days $\times$ 1 foot = 20 feet. Solving this, we find it takes 20 days for the snail to reach the top. We can verify this by noting that after 19 days, the snail will have climbed 19 feet. On the 20th day, it will climb the final 3 feet, reaching the top before any night-time slipping occurs. Therefore, the snail will reach the top of the well in 20 days.]]

Figure 11: Prompt used for acquiring training data.

KGW $n = 1$

Instruction: How does photosynthesis work?
Input: None
Answer: Photosynthesis is the process by which green plants, algae, and some bacteria use light to synthesize foods with the help of chlorophyll.

KGW $n = 2$

Instruction: Explain how the Internet has transformed the way people learn.
Input: None
Answer: The Internet has transformed learning by providing access to vast information, allowing flexible learning times, and facilitating interactive communication between learners and instructors.

KGW $n = 3$

Instruction: Provide the definition of a simile.
Input: None.
Answer: A simile is a figure of speech in which two things are compared using like or as, often used to enhance the vividness of a comparison.

SynthID-Text $n = 1$

Instruction: How do you brew a cup of green tea?
Input: None
Answer: 1. Boil water: Start by boiling water to the ideal temperature of 160 degrees Fahrenheit (70 degrees Celsius). 2. Steep the tea: Place a green tea bag or loose-leaf green tea in a tea pot or cup. 3. Add hot water: Pour the hot water directly over the tea. 4. Steep for the appropriate time: Green tea typically needs to steep for about 2-3 minutes. 5. Remove the tea: Take the tea bag or strain the loose leaves from the water. 6. Sweeten if desired: Add honey, sugar, or a sweetener of your choice to taste. 7. Serve: Pour the brewed tea into a cup and enjoy.

SynthID-Text $n = 2$

Instruction: Provide a list of three common kitchen tools and their uses.
Input: None
Answer: 1. Cutting board: A cutting board is used for chopping, slicing, and dicing various ingredients, such as vegetables, fruits, and meats. 2. Measuring cups and spoons: These tools are used for accurately measuring ingredients, ensuring consistent and precise measurements in recipes. 3. Kitchen knife: A kitchen knife is versatile and used for slicing, dicing, chopping, and mincing a wide range of ingredients.

SynthID-Text $n = 3$

Instruction: Analyze the given quote and identify the underlying theme.
Input: The only way to do great work is to love what you do.
Answer: 1. Passion and joy in work 2. Importance of personal fulfillment in work 3. The necessity of loving one's job

Figure 12: Selected examples in training dataset.