OpenStamp: A Watermark for Open-Source Language Models

Miroojin Bakshi
Indian Institute of Science
miroojinb@iisc.ac.in

Saksham Rastogi*
Carnegie Mellon University
iitdsaksham@gmail.com

Danish Pruthi
Indian Institute of Science
danishp@iisc.ac.in

Abstract

With the growing prevalence of large language model (LLM) generated content, watermarking is considered a promising approach for attributing text to LLMs and distinguishing it from human-written content. A prominent class of techniques embeds subtle but detectable signals in generated text by modifying token sampling probabilities. However, such methods are unsuitable for open-source models, where users have white-box access and can easily disable watermarking during inference. In this work, we introduce **OpenStamp**, a watermarking technique that implants detectable signals into the generated text by modifying just the final projection, or unembedding, layer. Through experiments across two models, we show that **OpenStamp** achieves superior detection performance, with minimal degradation in model capabilities compared to prior methods. The implanted watermark is explicitly designed, and empirically confirmed, to be more robust to paraphrasing attacks and harder to scrub off through post-hoc fine-tuning than prior open-source watermarks. We share our code at https://github.com/mb-14/openstamp to enable developers to easily watermark their models, and additionally release watermarked versions of four popular open-source models produced using **OpenStamp**.

1 Introduction

Large language models (LLMs) are capable of generating human-like text, which has led to their widespread adoption in various applications (Brown et al., 2020; Chowdhery et al., 2022). As these models become more prevalent, there are looming concerns about potential misuse, such as generating large-scale misinformation (Oviedo-Trespalacios et al., 2023), influencing public opinion (Panditharatne & Giansiracusa, 2023), or orchestrating social engineering attacks (Grbic & Dujlovic, 2023). To address these concerns, it is critical to develop methods for distinguishing LLM-generated content from human-written text. Such methods can also be used to attribute the content to its source model, thereby promoting transparency and accountability in LLM use, especially for high-stakes applications.

A promising way to detect model-generated content is to watermark the text by embedding subtle, imperceptible signals into the model's outputs. To implant such signals, several prominent watermarking techniques either modify the next-token probabilities (Kirchenbauer et al., 2023; Liu & Bu, 2024) or constrain the sampling process during generation (Aaronson, 2023; Kuditipudi et al., 2024). While effective, these decoding-based techniques are incompatible with requirements of open sourcing, where users have full control over the generation process and can easily disable the watermarking logic. This motivates the need for techniques that embed watermarking signals directly into the weights of the model.

Few recent works explore watermarking approaches for open-sourced LLMs. A notable approach proposes to distill a student model using the outputs of the watermarked teacher model (Gu et al., 2024). However, this method typically requires considerable amounts of training data and computational resources. Another recent study proposes to add Gaussian noise to the bias vector of the final layer, steering the generation towards a fixed set of tokens (Christ et al., 2024). Because these token preferences do not vary with context,

*Work done at the Indian Institute of Science

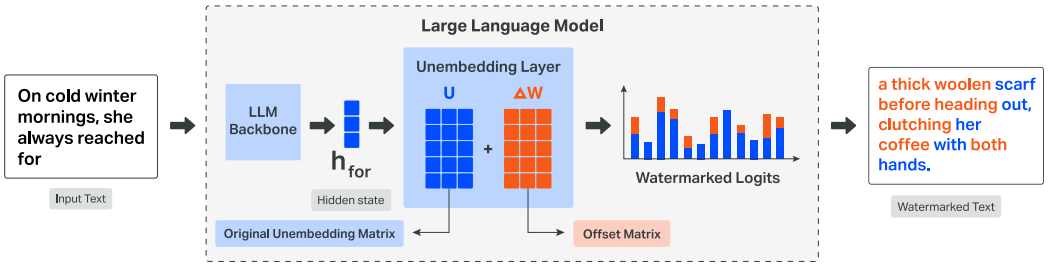


Figure 1: **Overview of `OpenStamp`, our watermarking method.** We add an offset matrix ΔW to the unembedding layer's weights U to produce **watermark logits** that bias token sampling, favoring tokens with higher watermark logit values. The watermark can be detected using a **log-likelihood ratio (LLR)** based score which measures how much more likely the observed tokens are under the watermarked model compared to the unwatermarked model.

the scheme is analogous to a fixed green list and can be reverse-engineered and rendered ineffective (Jovanović et al., 2024; Rastogi & Pruthi, 2024). Watermarking techniques are also brittle to paraphrasing attacks (Krishna et al., 2023), where an adversary can rephrase the generated text to dilute the watermarking signal. There have been some efforts to improve robustness to paraphrasing (Liu et al., 2024; Hou et al., 2024) but none of them are designed for open-source models.

In this work, we introduce **`OpenStamp`**, an approach to watermarking open-source LLMs by modifying the *unembedding layer*[1] weights of the model. We add a carefully designed offset to these weights, which bias the logits before generation and thereby implant signals in the generated text. **`OpenStamp`** is conceptually similar to prior logit-based watermarking approaches (Kirchenbauer et al., 2023; Liu et al., 2024; Liu & Bu, 2024), which influence token selection by adding small, context-dependent biases to the logits at each decoding step. However, an important distinction is that we embed the biasing logic directly into the unembedding layer's weights. Moreover, the offset is designed to produce consistent biases across semantically similar contexts, ensuring robustness against paraphrasing attacks. Figure 1 shows a high-level overview of our watermarking method. To detect watermarked text, we compute a length normalized **log-likelihood ratio (LLR)** score, which measures how much more likely the observed tokens are under the watermarked model compared to the unwatermarked model.

`OpenStamp` achieves near-perfect detection (TPR $\geq$ 99% at 1% FPR) with only minimal degradation in text quality, outperforming other open-source watermarking methods at comparable perplexity levels. Our watermarking method approaches the Pareto frontier defined by prominent decoding-based techniques. Furthermore, our watermarking signal is more difficult to erase via fine-tuning and offers improved robustness to paraphrasing attacks compared to existing approaches. We believe that these strengths make our approach a strong candidate for watermarking open-source models. To encourage model developers to watermark their models prior to release and to foster the development of new watermarking techniques for open-source models, we publicly release our code and watermarked versions of four popular open-source models created using **`OpenStamp`** at `https://github.com/mb-14/openstamp`.

2 Background and related work

Watermark for large language models. Watermarking techniques for LLMs embed signals in model-generated text that remain imperceptible to human readers but can be algorithmically detected. A common approach alters the decoding process (Kirchenbauer et al., 2023; Liu & Bu, 2024) by boosting a pseudorandom subset of tokens. Detection relies on statistical tests that exploit the resulting skew, for example by comparing the proportion of

[1]The unembedding layer is also referred to as the output projection layer or softmax layer.

green tokens against the baseline expected under unwatermarked text. For a comprehensive survey, see Liang et al. (2024).

Paraphrasing robustness. Paraphrasing attacks pose a major challenge for watermark detection (Krishna et al., 2023; Kirchenbauer et al., 2024). By replacing or rearranging tokens, paraphrasing can dilute the statistical patterns that watermarking methods rely on for detection. Recent work has explored strategies to improve robustness, such as using semantic embeddings to guide the selection of green tokens (Liu et al., 2024; Hou et al., 2024), making the signal invariant to surface-level changes. However, these methods depend on auxiliary models at inference time to produce semantic embeddings, making them unsuitable for open-source LLMs where users can easily bypass any watermarking logic that relies on external components.

Watermarks for open-source LLMs. In open-source settings, where users control the decoding process, any decoding-time logic can be easily bypassed; hence, the watermarking logic must be embedded directly into the model weights. One approach uses distillation, training a student model on watermarked outputs from a decoding-based watermarked teacher, thereby enabling the student to generate watermarked text natively (Gu et al., 2024). While effective, this method requires substantial computational resources and training data. A reinforcement learning-based framework jointly optimizes an LLM and a paired detector to balance detectability and text quality (Xu et al., 2025). A further line of work embeds a watermark into model weights by jointly fine-tuning LoRA adapters, optimizing for both coherence and detectability (Elhassan et al., 2025). Both methods are compute-intensive and rely on complex joint optimization objectives that are hard to tune.

Similar to our work are approaches that embed watermarking logic without requiring any LLM fine-tuning. Unremovable (Christ et al., 2024) introduces a fixed Gaussian perturbation to the final layer's bias vector, steering generation towards positively biased tokens, similar to a fixed green list (Zhao et al., 2024). Because the induced token preferences do not vary with context, this scheme is analogous to fixed green-list watermarks, which are known to be vulnerable to reverse-engineering attacks (Jovanović et al., 2024; Rastogi & Pruthi, 2024). GaussMark (Block et al., 2025) embeds a watermark by adding Gaussian noise to a subset of model weights, and detects it by analysing how the log-likelihood of a given text changes under the perturbed model. However, GaussMark can be difficult to embed, as it requires careful selection of the weight subsets and the noise strength to balance text quality and detectability across models.

Despite recent innovations, a key limitation across existing work is the lack of robustness to post-hoc model modifications. This is a major challenge for watermarking open-source models, which are often updated through quantization, pruning, merging, or fine-tuning. Gloaguen et al. (2025) find that no current methods remain detectable after such updates. We present complementary evidence on this limitation in Section 5.3. Various recent approaches underscore the inherent difficulty of watermarking in open-source LLMs. Each method exhibits different limitations across several axes: efficiency of integration, detectability, robustness, and practicality.

3 Methodology

Overview. This section presents our approach for watermarking language model outputs. We first introduce the model setup and notation (§3.1), then describe how a modification to the unembedding layer can bias generation towards specific tokens to embed a watermark signal (§3.2). We outline key desiderata for effective watermarking and show how our design satisfies them (§3.3). Finally, we describe how to detect the watermark using a log-likelihood ratio based score (§3.4).

3.1 Preliminaries

Let a language model process a sequence of tokens drawn from a vocabulary $\mathcal{V}$. Let $x_t \in \mathcal{V}$ denote the token at generation step t, and let $x_{\leq t} = (x_1, \ldots, x_t)$ denote the prefix up to step

t. The model computes a hidden representation $h_t = f(x_{\leq t}) \in \mathbb{R}^d$, where $f : \mathcal{V}^* \to \mathbb{R}^d$ is the model's internal encoding function. The model then produces a logit vector

$$v_t = Uh_t \in \mathbb{R}^{|\mathcal{V}|},$$

where $U \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the unembedding matrix. The model defines a categorical distribution $p(x_{t+1} \mid x_{\leq t})$ over the next token by applying a softmax over v_t, where each component $v_t^{(w)}$ corresponds to the logit value of token $w \in \mathcal{V}$. Watermarking strategies typically modify v_t during generation to influence the next-token distribution.

KGW watermarking. In the KGW watermarking scheme (Kirchenbauer et al., 2023), the logit modification step is guided by a pseudorandom partition of the vocabulary. At each generation step t, a pseudorandom function (PRF) selects a subset of tokens $\mathcal{G}_t \subset \mathcal{V}$ called the *green list*, based on the context $x_{\leq t}$ and a secret key. A parameter γ controls the fraction of tokens in the green list, so that $|\mathcal{G}_t| = \gamma|\mathcal{V}|$. The logits for tokens in the green list are boosted by a fixed amount $\delta > 0$:

$$\hat{v}_t^{(w)} \;=\; v_t^{(w)} + \delta \cdot \mathbb{1}\{w \in \mathcal{G}_t\}.$$

3.2 Watermarking via unembedding matrix modification

Modifying the unembedding matrix provides a direct mechanism to alter the logit vector produced at each generation step. Concretely, we define a modified unembedding matrix $\tilde{U} = U + \boldsymbol{\Delta W}$, where we refer to $\Delta W \in \mathbb{R}^{|\mathcal{V}| \times d}$ as the *offset matrix*. Applying this matrix to a hidden state h_t yields the modified logit vector $\tilde{v}_t = \tilde{U}h_t = v_t + \boldsymbol{\Delta W h_t}$, where $\Delta W h_t$ is termed *watermark logits*. These logits bias the output distribution during generation, favoring tokens with higher adjusted scores. This bias accumulates in the generated text, embedding a detectable watermark signal.

Desiderata. For the offset matrix to be effective, it should satisfy the following properties:

- **Detectability:** The watermark signal can be reliably identified in the generated text.
- **Controllability:** The balance between watermark strength and text quality is adjustable via hyperparameters.
- **Security:** The underlying watermarking logic is difficult to reverse-engineer.
- **Paraphrasing Robustness:** The watermark remains detectable even if the generated text is paraphrased.

3.3 Linearized green list biasing

Conceptual overview. We adapt the KGW scheme by modeling its green list selection as a linear transformation. While the PRF in KGW selects from an effectively infinite space of lists based on token prefixes, we simplify this mechanism by defining a finite set of L candidate green lists. The offset matrix ΔW is then constructed to map hidden states to watermark logits that approximate one of these L lists. Specifically, we decompose ΔW into three matrices:

$$\Delta W = GSP \tag{1}$$

This decomposition realizes the watermarking logic through three sequential operations: (1) **semantic alignment** via P, which projects the raw hidden state h_t into a manifold where geometric proximity reflects semantic similarity, ensuring similar watermarking behavior for paraphrased sentences; (2) **list selection** via S, which maps the projected hidden state Ph_t to a selector vector $s \in \mathbb{R}^L$ that acts as a soft selector over the L candidate green lists; and (3) **logit biasing** via G, which maps the selector vector into watermark logits that encode the chosen green list. We describe the construction of each matrix in detail below.

Projection matrix P. To ensure **paraphrasing robustness**, the watermarking logic must behave consistently across hidden states corresponding to semantically equivalent sentences. However, standard LLM hidden states are optimized for next-token prediction and may not

cluster according to semantic similarity. To address this, we introduce $P \in \mathbb{R}^{d \times d}$ to map h_t into a space where geometric proximity reflects semantic similarity. We train P on a paired dataset of hidden states and high-quality sentence embeddings $\{(h_i, e_i)\}_{i=1}^{N}$ by minimizing the Kullback-Leibler (KL) divergence between their similarity distributions within a batch $\mathcal{B}$:

$$\mathcal{L}(P) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} D_{KL}(\text{softmax}(\mathbf{s}_i^P / T_{\text{sim}}) \| \, \text{softmax}(\mathbf{s}_i^e / T_{\text{sim}})), \tag{2}$$

where $T_{\text{sim}} > 0$ is a temperature hyperparameter and the similarity vectors $\mathbf{s}_i^P, \mathbf{s}_i^e \in \mathbb{R}^{|\mathcal{B}|}$ represent the cosine similarities to all other elements in the batch:

$$[\mathbf{s}_i^P]_j = \cos(Ph_i, Ph_j), \quad [\mathbf{s}_i^e]_j = \cos(e_i, e_j) \quad \forall j \in \mathcal{B}.$$

Minimizing this loss encourages the projected states to inherit the geometric structure of the target embedding space. Training details are provided in Appendix F.

Selector matrix S. The selector matrix $S \in \mathbb{R}^{L \times d}$ maps the projected hidden state to a selection vector $s = SPh_t \in \mathbb{R}^L$. This vector acts as a *soft selector* over L candidate green lists. To train S, we partition the projected hidden states into L clusters using k-means. This gives us a training dataset $\mathcal{D}_{\text{train}} = \{(Ph_i, y_i)\}_{i=1}^{N}$, where y_i is the one-hot encoding of the cluster label for Ph_i. We then learn S by solving the following ridge regression problem:

$$\min_S \sum_{(Ph_i, y_i) \in \mathcal{D}_{\text{train}}} \|SPh_i - y_i\|^2 + \lambda \|S\|_F^2,$$

where $\lambda > 0$ is a regularization parameter. See Appendix G for training details.

Green list matrix G. The matrix $G \in \mathbb{R}^{|\mathcal{V}| \times L}$ encodes L candidate green lists as its columns such that Gs produces the desired watermark logits. For each column $l \in \{1, \ldots, L\}$, the corresponding green list $\mathcal{G}_l \subset \mathcal{V}$ is defined by a PRF:

$$\mathcal{G}_l = \{\, i \mid \text{PRF}(\texttt{seed}, l, i) < \gamma \,\},$$

where seed is the secret key shared between watermark generation and detection, and $\text{PRF}(\cdot)$ maps the token index i to a value in $[0, 1]$. Each entry of G is then given by

$$G_{i,l} = \delta \cdot \mathbb{1}\{i \in \mathcal{G}_l\}.$$

The hyperparameters γ and δ have the same interpretation as in the KGW watermarking scheme, offering **controllability** over the watermark's strength and impact on text quality.

Variability. By dynamically selecting among multiple green lists based on the context, our method maintains variability in the watermark logits, making it difficult to reverse-engineer. We provide supporting evidence of this variability in Appendix M.

Approximating one-hot selectors. Since our method operates on continuous hidden states, the selector matrix S yields soft, continuous selectors instead of exact one-hot vectors. As a result, multiple green lists can be partially activated at once leading to weaker alignment with the intended green-list behavior. Nonetheless, detection performance remains strong. We analyze this effect and its influence on detection in Appendix L.

3.4 Detection via log-likelihood ratio

To detect the presence of a watermark, we compute a **length-normalized log-likelihood ratio (LLR)** for the given sequence using the watermarked and original model, specifically:

$$\text{LLR}(x) = \frac{1}{T} \sum_{t=1}^{T} \log \frac{p_{\text{wm}}(x_t \mid x_{<t})}{p_{\text{orig}}(x_t \mid x_{<t})}, \tag{3}$$

where the probabilities are defined via softmax over logits from the original unwatermarked model, p_{orig} and it's watermarked version, p_{wm}.

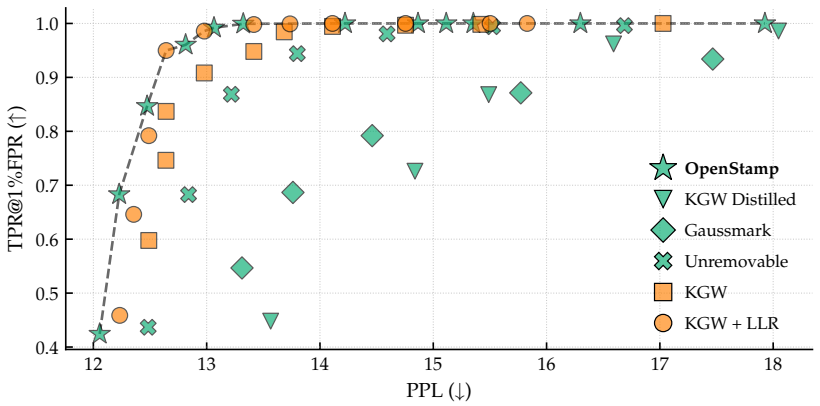


Figure 2: **Trade-off between detectability and text quality for various watermarking methods on `Llama-2-7B`. `OpenStamp`** is on the Pareto frontier and outperforms all open-source compatible methods (shown in green) by a substantial margin. Vanilla KGW performs slightly worse than KGW + LLR, highlighting the effectiveness of the LLR detection method.

Intuition. The **LLR** score measures how much more likely a given token sequence is under the watermarked model than under the original model. Because the watermark logits systematically bias sampling towards certain favored tokens, generations from the watermarked model tend to accumulate higher LLR values, making the watermark **detectable** in practice. Length normalization ensures that a single detection threshold can be applied consistently across sequences of different lengths. A key assumption behind this approach is that unwatermarked text, whether human-written or generated by other models, is better modeled by the original model than by the watermarked one. While the conditional probabilities in Equation 3 formally condition on the full history, detection remains robust using only partial prefixes that exclude the generation prompt (See Appendix O for details). *Crucially, this means the detection process does not require knowledge of the specific prompt used to generate the text.*

Unlike frequency-based detection methods (Kirchenbauer et al., 2023), our LLR score does not constitute a formal statistical test. The null hypothesis—that the text is unwatermarked—is *composite*, encompassing human-written and other model-generated text. As noted in Block et al. (2025), this makes p-value calibration intractable, since the null distribution of the LLR is not well-defined. Nevertheless, because the LLR uses the full token probability distribution and naturally captures the *mixture* of green lists produced by our watermarking method, it preserves signal that frequency-based tests lose by relying only on discrete counts (See Appendix Q for comparisons).

Detection protocol. The developer publicly releases a watermarked model with the modified unembedding matrix $\tilde{U} = U + \Delta W$, while keeping the original unembedding matrix private. This prevents an adversary from recovering the watermark offset $\Delta W = \tilde{U} - U$. This also means that the method is not applicable to models whose original weights are already public. Any text generated using the public model can be detected using Algorithm 1. This protocol assumes white-box access to the model weights in order to evaluate log-likelihoods.

4 Experimental setup

We evaluate our watermarking method across four key axes: **(i)** detection performance (§ 5.1), **(ii)** robustness to paraphrasing attacks (§ 5.2), **(iii)** resistance to post-hoc model modifications (§ 5.3) and **(iv)** impact on downstream tasks (§ 5.4). We conduct experiments across all four axes on `Llama-2-7B` (Touvron et al., 2023) and `Mistral-7B` (Jiang et al., 2023). To evaluate the generalizability of our method, we also report detection perfor-

Method	Model	ArXiv		BookSum		Wikipedia	
		TPR	PPL	TPR	PPL	TPR	PPL
KGW†	`Llama-2-7B`	0.994±0.005	32.3±1.8	0.998±0.002	24.1±1.4	0.914±0.036	13.8±0.6
	`Mistral-7B`	0.998±0.002	30.2±2.4	0.998±0.002	24.6±0.7	0.911±0.092	13.2±0.6
KGW + LLR†	`Llama-2-7B`	0.994±0.010	32.0±1.1	1.000±0.001	23.2±0.4	0.996±0.004	12.7±0.4
	`Mistral-7B`	1.000±0.000	29.2±1.2	1.000±0.001	23.6±0.5	0.992±0.009	13.0±1.0
KGW Distilled	`Llama-2-7B`	0.886±0.076	43.48±0.7	0.927±0.035	28.6±2.4	0.813±0.141	16.4±1.7
	`Mistral-7B`	0.983±0.011	44.5±4.6	0.998±0.001	34.6±2.1	0.878±0.096	20.1±0.8
GaussMark	`Llama-2-7B`	0.644±0.278	40.0±0.6	0.847±0.076	30.4±2.3	0.750±0.027	17.2±0.6
	`Mistral-7B`	0.395±0.126	33.3±2.0	0.847±0.125	26.6±0.5	0.387±0.199	14.3±0.4
Unremovable	`Llama-2-7B`	0.997±0.002	28.9±1.2	0.997±0.003	25.2±3.3	0.871±0.042	14.5±1.6
	`Mistral-7B`	0.987±0.013	26.3±2.7	**0.995±0.005**	24.7±1.7	0.883±0.081	13.3±0.7
`OpenStamp`	`Llama-2-7B`	**1.000±0.000**	30.7±2.3	**0.999±0.001**	25.6±3.1	**0.998±0.002**	14.4±0.9
	`Mistral-7B`	**1.000±0.000**	29.3±1.3	0.995±0.002	25.5±0.3	**0.990±0.004**	12.9±0.6

Table 1: **Comparing different watermarking approaches across multiple datasets.** We report TPR@0.1%FPR (labeled TPR) to measure detectability and perplexity (PPL) to characterize text quality. Bold values indicate the best TPR@0.1%FPR at the lowest PPL for each dataset and model. **KGW**† and **KGW + LLR**† are decoding-based watermark methods included for reference. For both models, **`OpenStamp`** achieves near-perfect detection with PPL competitive with Unremovable and lower than GaussMark and KGW Distilled, demonstrating a superior tradeoff between text quality and detectability across the three datasets.

mance on four additional models: `phi-4` (Abdin et al., 2024), `Olmo-3-7B` (Olmo et al., 2025), `Qwen2.5-7B` (Qwen et al., 2025), and `SmolLM2-1.7B` (Allal et al., 2025), in Appendix C.

4.1 Baselines

We compare our method against three prior open-source-compatible watermarking approaches: Unremovable (Christ et al., 2024), GaussMark (Block et al., 2025), and a distilled version of KGW (Gu et al., 2024). [2] We also include standard KGW as a representative of more conventional decoding-based techniques. In addition, we evaluate KGW combined with the LLR-based detector described in Section 3.4, which we regard as an approximate upper bound on detection performance. Hyperparameters for all methods are provided in Appendix I.

4.2 Evaluation protocol

Watermarked sample generation. We generate 500 watermarked completions of 200 tokens each, using 50-token prompts sampled from the `RealNewsLike` subset of C4 (Raffel et al., 2020). The corresponding unwatermarked completions are taken directly from the dataset as the next 200 tokens after each prompt. Detection is performed only on the continuations and excludes the prompts. Watermarked generations are sampled using nucleus sampling with temperature 1.0. To assess generalizability, we also evaluate on prompts drawn from `ArXiv` (Cohan et al., 2018), `BookSum` (Kryscinski et al., 2022), and Wikipedia (Foundation).

Metrics. We evaluate two properties: *watermark detectability* and *text quality*. **Detectability** is measured by the AUROC and the true positive rate at a fixed false positive rate of 1% (TPR@1%FPR) and 0.1% (TPR@0.1%FPR). In applications such as plagiarism detection, where false positives carry a high cost, maintaining a low FPR is essential. **Text quality**

[2] We omit RL-based watermarking (Xu et al., 2025) as the model either produces repetitive sequences or the detection rate is unacceptably low (See Appendix J for details).

is measured by the mean perplexity (PPL) of the watermarked samples, computed using `Llama-2-13B` as the oracle model. All metrics are averaged over three random seeds.

5 Results

5.1 Detection performance

Figure 2 illustrates the tradeoff between text quality and detectability for various watermarking methods on `Llama-2-7B`. Each method's tradeoff curve is generated by varying a method-specific parameter that controls strength of the signal. Open-source methods are shown in green, while decoding-based methods are shown in orange.

`OpenStamp` achieves near-perfect detection (TPR $\geq$ 99.9%) at a perplexity of approximately 13.3, outperforming the open-source baselines at comparable perplexity: Unremovable reaches about 87% TPR@1%FPR, while GaussMark and KGW Distilled reach only 45–55%. The gap between vanilla KGW and the KGW+LLR upper bound highlights the effectiveness of the LLR detection method. Furthermore, **`OpenStamp`**'s superior detection power is consistent across multiple datasets even at 0.1% FPR, as shown in Table 1. Appendix D reports additional results under significant distribution shift.

5.2 Robustness to paraphrasing attacks

To test robustness against paraphrasing attacks, we prompt an instruction-tuned LLM to paraphrase watermarked samples and measure the detectability of the paraphrased outputs. Details on the prompt and model used for paraphrasing are provided in Appendix H. As a reference method, we include UNIGRAM (Zhao et al., 2024), which uses a static green list and is inherently robust to paraphrasing since its green token set remains fixed regardless of surface form. To isolate the effect of the projection matrix P in **`OpenStamp`** on paraphrasing robustness, we evaluate an ablation of our method without the projection matrix (**`OpenStamp`** w/o sem-align), where the selector matrix S is trained on raw hidden states.

Table 2 shows **`OpenStamp`** achieves the highest TPR@1%FPR on both models, demonstrating superior robustness to paraphrasing attacks. Ablating the projection matrix degrades performance, confirming its contribution in enhancing robustness to paraphrasing.

	`Llama-2-7B`		`Mistral-7B`	
Method	**AUROC**	**TPR@1%FPR**	**AUROC**	**TPR@1%FPR**
UNIGRAM	0.966 ± 0.013	0.717 ± 0.063	0.965 ± 0.008	0.666 ± 0.054
KGW Distilled	0.950 ± 0.004	0.602 ± 0.021	0.954 ± 0.013	0.596 ± 0.051
GaussMark	0.946 ± 0.002	0.550 ± 0.035	0.925 ± 0.016	0.409 ± 0.061
Unremovable	0.966 ± 0.009	0.724 ± 0.098	0.966 ± 0.008	0.715 ± 0.010
`OpenStamp` w/o sem-align	0.971 ± 0.005	0.762 ± 0.024	0.960 ± 0.012	0.732 ± 0.038
`OpenStamp`	$\mathbf{0.990 \pm 0.003}$	$\mathbf{0.907 \pm 0.003}$	$\mathbf{0.978 \pm 0.005}$	$\mathbf{0.791 \pm 0.013}$

Table 2: **Detectability of paraphrased text.** On both models, **`OpenStamp`** achieves the highest TPR@1%FPR on paraphrased text. **`OpenStamp`** w/o sem-align degrades performance, confirming that the projection matrix contributes to robustness against paraphrasing attacks.

5.3 Resistance to post-hoc model modifications

We assess how well open-source compatible watermarking methods can resist post-hoc fine-tuning. Specifically, we simulate an adversary attempting to erase the watermark from model weights by further fine-tuning on OpenWebText (Gokaslan & Cohen, 2019) using LoRA (Hu et al., 2022). Full experimental details are provided in Appendix K. We measure the TPR@1%FPR after every 500 fine-tuning steps up to 2,500 steps. We discuss an additional attack setup in Appendix P.

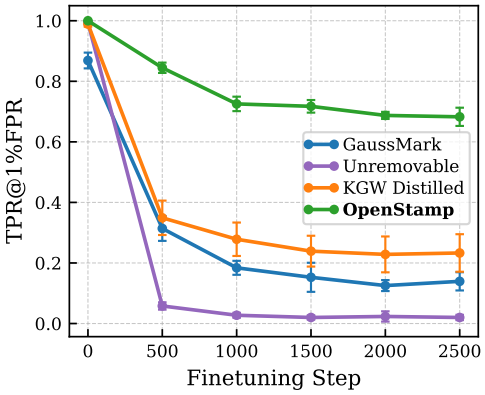


Figure 3: **`Llama-2-7B` detectability under post-hoc fine-tuning.** All methods degrade over time, but **`OpenStamp`** maintains higher detectability compared to GaussMark, KGW Distilled, and Unremovable.

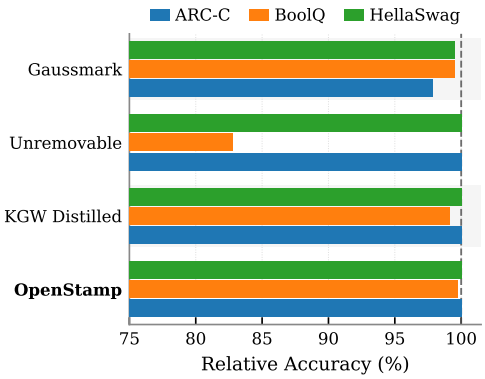


Figure 4: **Relative downstream task accuracy of watermarked `Llama-2-7B`.** Accuracy is shown as a percentage of the unwatermarked baseline (dashed line at 100%). **`OpenStamp`** exhibits minimal degradation in accuracy across all benchmarks.

Figure 3 shows that while all watermarks degrade during `Llama-2-7B` fine-tuning, **`OpenStamp`** maintains superior detectability over GaussMark, KGW Distilled, and Unremovable. Similar trends hold for `Mistral-7B` (See Figure 5).

Instruction fine-tuning. The OpenWebText experiment above simulates an adversary who continues fine-tuning on web text to scrub the watermark. Released checkpoints are also commonly instruction-tuned after publication to adapt base models into chat-capable systems. We evaluate this setting by fine-tuning watermarked models on the Alpaca instruction dataset (Taori et al., 2023). Training details are provided in Appendix K. To measure durability on both general-domain and instruction-domain outputs, we then evaluate detectability on RealNewsLike completions under the protocol in Section 4.2, and on instruction-following completions from AlpacaEval (Li et al., 2023b). As shown in Table 3, detectability on RealNewsLike remains strong for all methods, with **`OpenStamp`** retaining the highest TPR@1%FPR. On AlpacaEval, TPR@1%FPR drops for every method, with Unremovable strongest and **`OpenStamp`** remaining competitive. Corresponding `Mistral-7B` results follow the same pattern (Appendix B). These results are consistent with the findings of Gloaguen et al. (2025): after instruction fine-tuning, the watermark can remain strong on general text while weakening on instruction-following outputs.

	RealNewsLike		AlpacaEval	
Method	**AUROC**	**TPR@1%FPR**	**AUROC**	**TPR@1%FPR**
KGW Distilled	0.996 ± 0.002	0.952 ± 0.006	0.634 ± 0.206	0.090 ± 0.135
GaussMark	0.980 ± 0.003	0.775 ± 0.014	0.885 ± 0.027	0.301 ± 0.092
Unremovable	0.996 ± 0.001	0.965 ± 0.010	$\mathbf{0.984 \pm 0.005}$	$\mathbf{0.707 \pm 0.101}$
`OpenStamp`	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	0.977 ± 0.005	0.699 ± 0.038

Table 3: **`Llama-2-7B` detectability under instruction fine-tuning.** TPR@1%FPR remains high on RealNewsLike but drops on AlpacaEval. Bold values mark the best in each column.

Reinitializing and retraining the unembedding layer. An adversary may attempt to remove the watermark by reinitializing the unembedding layer and relearning the mapping from hidden states to token logits to restore fluent generation. Restoring this capability is non-trivial, since model developers train LLMs on large curated datasets and rely on complex pretraining and post-training pipelines. To illustrate this difficulty, we reinitialized

the unembedding layer and retrained only that layer on FineWeb (Penedo et al., 2024). Table 4 reports perplexity after 2,500 and 25,000 training steps. Even after training on 800M tokens, perplexity remains far worse than the watermarked baseline, showing that recovering quality is non-trivial and making this attack computationally costly.

Condition	PPL
Watermarked baseline	15.1
After 2,500 steps (80M tokens)	78,529.4
After 25,000 steps (800M tokens)	351.7

Table 4: **Perplexity after retraining a reinitialized unembedding layer.** Only the unembedding layer is reinitialized and retrained on FineWeb; all other weights remain frozen.

5.4 Impact on downstream task accuracy

Since watermarking methods alter the output distributions of LLMs, it is essential to ensure that the utility of the underlying model is not significantly compromised. While the results in Figure 2 demonstrate minimal degradation in perplexity, Ajith et al. (2024) find that perplexity measurements cannot reliably predict the performance trade-offs due to watermarking on downstream tasks. We therefore further evaluate watermarked models using the Language Model Evaluation Harness (Gao et al., 2024), measuring potential degradation across three benchmarks: ARC-C (Clark et al., 2018), BoolQ (Clark et al., 2019), and HellaSwag (Zellers et al., 2019).

As shown in Figure 4, **`OpenStamp`** causes minimal accuracy degradation on `Llama-2-7b` across all benchmarks. Notably, the results are statistically indistinguishable from the unwatermarked baseline, indicating no meaningful impact on downstream performance. Similar results are observed for `Mistral-7b` (see Figure 6).

6 Limitations

Our work has several important limitations. First, detection requires a forward pass through the model, making it computationally expensive compared to methods that perform statistical tests on generated text. Second, our method assumes access to token probabilities from the base model. However, this is a realistic assumption as model owners typically have white-box access to their models. While our detection algorithm achieves stronger performance, it is not grounded in a statistical test and thus lacks a probabilistic interpretation, such as confidence levels or false positive rates, that help quantify detection uncertainty.

7 Conclusion

We proposed an effective approach for watermarking open-source LLMs by embedding the watermarking logic directly into the unembedding layer weights. This design natively integrates the watermark into the generation process, eliminating the need for decoding-time interventions. Our approach outperforms existing baselines in detection performance while preserving text quality. Furthermore, it demonstrates superior robustness against both paraphrasing and post-hoc fine-tuning, while remaining controllable and easy to integrate. Future work could explore advanced offset matrix designs to further improve this robustness, and enable detection methods with statistical guarantees.

Acknowledgements

We thank the reviewers for their feedback. We are grateful to Krishna Pillutla for discussions that positively shaped our work. Additionally, DP is grateful to the National Payments Corporation of India (NPCI), Schmidt Sciences through their AI2050 program (Grant G-24-66186), and Google for generously supporting his group's research.

References

Scott Aaronson. 20 - 'Reform' AI Alignment with Scott Aaronson. AXRP - The AI X-risk Research Podcast, 2023. URL https://axrp.net/episode/2023/04/11/episode-20-reform-ai-alignment-scott-aaronson.html.

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. Phi-4 technical report, 2024. URL https://arxiv.org/abs/2412.08905.

Anirudh Ajith, Sameer Singh, and Danish Pruthi. Downstream trade-offs of a family of text watermarks. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 14039–14053, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.821. URL https://aclanthology.org/2024.findings-emnlp.821/.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Adam Block, Alexander Rakhlin, and Ayush Sekhari. Gaussmark: A practical approach for structural watermarking of language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=YG3DbpAQBf.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.137. URL https://aclanthology.org/2024.findings-acl.137/.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

Miranda Christ, Sam Gunn, Tal Malkin, and Mariana Raykova. Provably robust watermarks for open-source language models, 2024. URL https://arxiv.org/abs/2410.18861.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

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

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 615–621, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2097. URL https://aclanthology.org/N18-2097/.

Fay Elhassan, Niccolò Ajroldi, Antonio Orvieto, and Jonas Geiping. Can you finetune your binoculars? embedding text watermarks into the weights of large language models. In *The 1st Workshop on GenAI Watermarking*, 2025. URL https://openreview.net/forum?id=d1v2hkB0Vj.

Wikimedia Foundation. Wikimedia downloads. URL https://dumps.wikimedia.org.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

GitHub. Github copilot. https://github.com/copilot, 2026.

Thibaud Gloaguen, Nikola Jovanović, Robin Staab, and Martin Vechev. Towards watermarking of open-source LLMs. In *The 1st Workshop on GenAI Watermarking*, 2025. URL https://openreview.net/forum?id=OKjZvLJxOY.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Dijana Vukovic Grbic and Igor Dujlovic. Social engineering with chatgpt. In *2023 22nd International Symposium INFOTEH-JAHORINA (INFOTEH)*, pp. 1–5. IEEE, 2023. doi: 10.1109/INFOTEH57020.2023.10094141.

Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. On the learnability of watermarks for language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=9k0krNzvlV.

Abe Hou, Jingyu Zhang, Yichen Wang, Daniel Khashabi, and Tianxing He. k-SemStamp: A clustering-based semantic watermark for detection of machine-generated text. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1706–1715, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.98. URL https://aclanthology.org/2024.findings-acl.98/.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. *ICML*, 2024.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17061–17084. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kirchenbauer23a.html.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. On the reliability of watermarks for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=DEJIDCmWOz.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Frederick Wieting, and Mohit Iyyer. Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=WbFhFvjjKj.

Wojciech Kryscinski, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. BOOKSUM: A collection of datasets for long-form narrative summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 6536–6558, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.488. URL https://aclanthology.org/2022.findings-emnlp.488/.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=FpaCL1MO2C.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. Who wrote this code? watermarking for code generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4890–4911, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.268. URL https://aclanthology.org/2024.acl-long.268/.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. StarCoder: may the source be with you! *Transactions on Machine Learning Research*, 2023a. URL https://openreview.net/forum?id=KoFOg41haE.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaEval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 5 2023b.

Yuqing Liang, Jiancheng Xiao, Wensheng Gan, and Philip S. Yu. Watermarking techniques for large language models: A survey, 2024. URL https://arxiv.org/abs/2409.00089.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. A semantic invariant robust watermark for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6p8lpe4MNf.

Yepeng Liu and Yuheng Bu. Adaptive text watermark for large language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=7emOSb5UfX.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

OpenAI. Chatgpt. https://chat.openai.com/, 2025. Large language model; accessed 2025-09-01.

Oscar Oviedo-Trespalacios, Amy E Peden, Thomas Cole-Hunter, Arianna Costantini, Milad Haghani, J.E. Rod, Sage Kelly, Helma Torkamaan, Amina Tariq, James David Albert Newton, Timothy Gallagher, Steffen Steinert, Ashleigh J. Filtness, and Genserik Reniers. The risks of using chatgpt to obtain common safety-related information and advice. *Safety Science*, 167:106244, 2023. ISSN 0925-7535. doi: 10.1016/j.ssci.2023.106244. URL https://www.sciencedirect.com/science/article/pii/S0925753523001868.

Mekela Panditharatne and Noah Giansiracusa. How ai puts elections at risk - and the needed safeguards, 2023. URL https://www.brennancenter.org/our-work/analysis-opinion/how-ai-puts-elections-risk-and-needed-safeguards.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. OpenWebMath: An open dataset of high-quality mathematical web text. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=jKHmjlpViu.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. FineWeb2: One pipeline to scale them all – adapting pre-training data processing to every language, 2025. URL https://arxiv.org/abs/2506.20920.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(1), January 2020. ISSN 1532-4435. URL https://jmlr.org/papers/v21/20-074.html.

Saksham Rastogi and Danish Pruthi. Revisiting the robustness of watermarking to paraphrasing attacks. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp.

18100–18110, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1005. URL https://aclanthology.org/2024.emnlp-main.1005/.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpaca: A strong, replicable instruction-following model. https://crfm.stanford.edu/2023/03/13/alpaca.html, 2023. Stanford Center for Research on Foundation Models (CRFM) blog post.

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. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Multilingual e5 text embeddings: A technical report, 2024. URL https://arxiv.org/abs/2402.05672.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJeYe0NtvH.

Xiaojun Xu, Yuanshun Yao, and Yang Liu. Learning to watermark LLM-generated text via reinforcement learning. In *The 1st Workshop on GenAI Watermarking*, 2025. URL https://openreview.net/forum?id=dTlCMiEOdk.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for AI-generated text. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=SsmT8aO45L.

Appendices

A Watermark detection via LLR score

The LLR-based detection algorithm is formalized in Algorithm 1. Given a sequence x, we first extract the hidden states from the LLM backbone. We then compute the log-likelihood of the sequence under both the original model and the watermarked model (which incorporates the offset matrix ΔW). Finally, we calculate the length-normalized LLR score and compare it against a threshold τ to determine if the sequence is watermarked.

Algorithm 1 Watermark Detection via LLR Score

Require: Sequence $x = (x_1, \ldots, x_T)$, LLM backbone $f(\cdot)$, unembedding matrix U, offset matrix ΔW, threshold τ
Ensure: `True` if x is watermarked; else `False`

// Extract hidden states from the LLM
$(h_1, \ldots, h_T) \leftarrow f(x)$
// Compute log-likelihood of sequence under original model
$\ell_{\text{orig}} \leftarrow \sum_{t=1}^{T-1} \log \text{softmax}(Uh_t)[x_{t+1}]$
// Compute log-likelihood under watermarked model
$\ell_{\text{wm}} \leftarrow \sum_{t=1}^{T-1} \log \text{softmax}((U + \Delta W)h_t)[x_{t+1}]$
// Compute length-normalized LLR score
$\text{LLR}(x) \leftarrow (\ell_{\text{wm}} - \ell_{\text{orig}})/(T-1)$
return $(\text{LLR}(x) > \tau)$

B Mistral additional results

We present additional results on the `Mistral-7B` model across three axes: **robustness to post-hoc fine-tuning** (Figure 5), **relative downstream task accuracy** (Figure 6), and **detectability after instruction fine-tuning** (Table 5). As in the `Llama-2-7B` setting (Table 3), detectability on

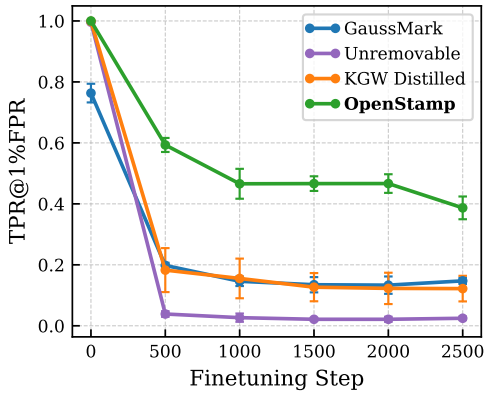


Figure 5: **`Mistral-7B` detectability under post-hoc fine-tuning.** All methods degrade over time, but **`OpenStamp`** maintains higher detectability compared to GaussMark, KGW Distilled, and Unremovable.

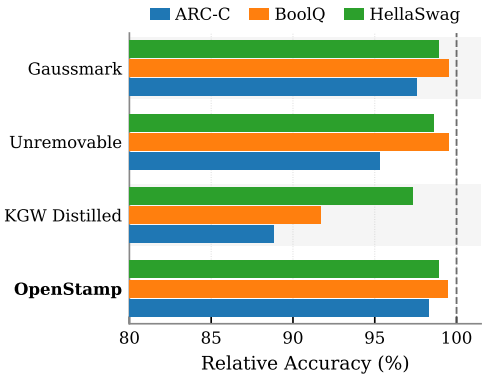


Figure 6: **Relative downstream task accuracy of watermarked `Mistral-7B`. `OpenStamp`** exhibits minimal degradation in accuracy across all benchmarks.

RealNewsLike remains high after Alpaca instruction fine-tuning, with **`OpenStamp`** retaining the strongest TPR@1%FPR, while AlpacaEval scores are lower across methods.

Method	RealNewsLike		AlpacaEval	
	AUROC	TPR@1%FPR	AUROC	TPR@1%FPR
KGW Distilled	0.997 ± 0.002	0.936 ± 0.049	0.597 ± 0.178	0.066 ± 0.087
GaussMark	0.963 ± 0.007	0.595 ± 0.064	0.853 ± 0.044	0.223 ± 0.128
Unremovable	0.996 ± 0.001	0.939 ± 0.015	$\mathbf{0.979 \pm 0.003}$	$\mathbf{0.589 \pm 0.133}$
OpenStamp	$\mathbf{1.000 \pm 0.000}$	$\mathbf{0.998 \pm 0.002}$	0.948 ± 0.020	0.454 ± 0.009

Table 5: **Mistral-7B detectability under instruction fine-tuning.** TPR@1%FPR remains high on RealNewsLike but drops on AlpacaEval, matching Table 3. Bold values mark the best in each column.

C Generalizability across LLMs

To assess the generalizability of our method across diverse LLMs, we evaluate its detection performance on four additional models: `phi-4` (Abdin et al., 2024), `Olmo-3-7B` (Olmo et al., 2025), `Qwen2.5-7B` (Qwen et al., 2025), and `SmolLM2-1.7B` (Allal et al., 2025). This selection covers a variety of model families, parameter scales, and training stages (pretrained vs. instruction-tuned). For all models, we apply same hyperparameters ($\gamma = 0.25$, $\delta = 1.0$) and evaluate the detection performance using the same protocol as described in Section 4.2. Table 6 shows that **OpenStamp** achieves perfect TPR@1%FPR across all four models, demonstrating its robustness and generalizability. The watermarked perplexity (WM PPL) values are reasonably close to the unwatermarked perplexity (Baseline PPL), indicating that the watermark does not significantly degrade text quality across these diverse models.

Model	TPR@1%FPR	TPR@0.1%FPR	WM PPL	Baseline PPL
`Olmo-3-1025-7B`	1.000 ± 0.000	1.000 ± 0.000	19.329 ± 0.377	16.801
`Qwen2.5-7B`	1.000 ± 0.000	1.000 ± 0.000	15.869 ± 0.355	13.097
`SmolLM2-1.7B`	1.000 ± 0.000	1.000 ± 0.000	29.929 ± 2.310	20.275
`phi-4`	1.000 ± 0.000	0.999 ± 0.002	7.128 ± 0.158	6.157

Table 6: **Generalizability across LLMs. OpenStamp** achieves perfect TPR@1%FPR across all four models, demonstrating its robustness and generalizability. The difference between WM PPL and Baseline PPL indicates the impact of watermarking on the model's perplexity.

D Detection under significant distribution shift

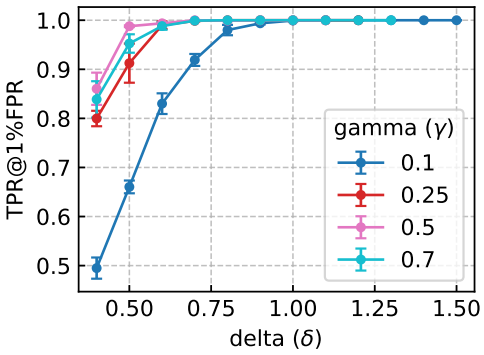
Because the projection matrix P and selector matrix S are trained on English web text from OpenWebText (Gokaslan & Cohen, 2019), we further evaluate detectability and text quality on three datasets that represent a significant distribution shift from this training distribution: the Japanese subset of FineWeb-2 (Penedo et al., 2025), OpenWebMath (Paster et al., 2024) (mathematical web text), and the Python subset of StarCoderData (Li et al., 2023a). Table 7 reports TPR@0.1%FPR and perplexity under these settings. **OpenStamp** retains strong detectability on the Japanese and mathematical web-text datasets for both primary models. Performance degrades only on the Python subset of StarCoderData, consistent with prior work showing that code is particularly challenging to watermark because of its low-entropy, highly constrained token distributions (Lee et al., 2024). The baselines degrade more severely under this shift. While Unremovable transfers better than GaussMark and KGW Distilled, it trails **OpenStamp** across all datasets, with the largest gap on code.

E Effect of varying γ and δ

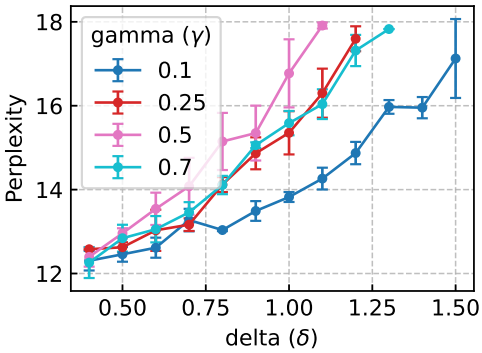
We measure the detectability and text quality of watermarked samples generated using different values of γ and δ on `Llama-2-7B`. The results, shown in Figures 7a and 7b, indicate the trade-off between detectability and distortion as we vary these parameters.

Method	Model	OpenWebMath		FineWeb-2 (Ja)		StarCoderData	
		TPR	PPL	TPR	PPL	TPR	PPL
KGW†	Llama-2-7B	0.748±0.269	16.3±1.0	0.678±0.470	18.6±0.7	0.491±0.261	6.5±0.3
	Mistral-7B	0.596±0.496	15.1±0.2	0.980±0.015	27.3±1.0	0.414±0.180	7.1±0.7
KGW + LLR†	Llama-2-7B	0.998±0.002	15.4±0.5	0.998±0.001	17.7±1.3	0.915±0.006	6.6±0.3
	Mistral-7B	0.988±0.005	14.4±0.7	1.000±0.000	27.4±0.7	0.868±0.075	6.4±0.2
KGW Distilled	Llama-2-7B	0.247±0.329	20.5±1.1	0.247±0.222	25.5±0.0	0.133±0.100	8.3±0.3
	Mistral-7B	0.304±0.505	23.4±1.4	0.554±0.255	50.0±4.4	0.195±0.156	15.4±1.4
GaussMark	Llama-2-7B	0.478±0.142	17.3±1.1	0.957±0.031	25.9±1.4	0.294±0.113	7.8±0.2
	Mistral-7B	0.282±0.051	16.4±0.7	0.785±0.193	36.3±3.7	0.186±0.092	7.3±0.5
Unremovable	Llama-2-7B	0.912±0.059	16.0±2.3	0.929±0.039	15.6±2.4	0.720±0.126	7.0±1.5
	Mistral-7B	0.855±0.072	13.6±1.8	0.921±0.075	24.6±1.8	0.599±0.133	5.8±0.3
OpenStamp	Llama-2-7B	**0.993±0.009**	14.5±1.2	**1.000±0.000**	19.6±3.0	**0.923±0.047**	6.9±0.7
	Mistral-7B	**0.982±0.007**	14.7±0.2	**1.000±0.000**	25.3±3.0	**0.842±0.088**	6.9±0.4

Table 7: **Detection under significant distribution shift.** We report TPR@0.1%FPR (labeled TPR) and perplexity (PPL) on OpenWebMath (Paster et al., 2024), the Japanese subset of FineWeb-2 (Penedo et al., 2025), and the Python subset of StarCoderData (Li et al., 2023a). Bold values indicate the highest TPR among open-weight watermarking methods for each dataset and model.



(a) **Effect of γ and δ on detectability.** TPR@1%FPR vs. δ for various γ values.



(b) **Effect of γ and δ on text quality.** Perplexity vs. δ across the same γ settings, measured using `Llama-2-13b`.

F Training the projection matrix P

The projection matrix P is trained on a paired dataset of hidden states and high-quality sentence embeddings extracted from a large-scale text corpus. Specifically, hidden states are extracted from the model's final layer, while target sentence embeddings are obtained using the `Qwen3-Embedding-8B` embedding model (Zhang et al., 2025). Training utilizes 1.5 million paired samples extracted from the OpenWebText corpus, with a maximum sequence length of 512 tokens.

We optimize P using a contrastive loss function 2 designed to align the similarity distribution of projected hidden states with that of the target sentence embeddings. The optimization is performed using `AdamW` (Loshchilov & Hutter, 2019) with a learning rate of 10^{-5}, a weight decay of 10^{-2}, and a temperature $T_{\text{sim}} = 0.1$ over 15 epochs.

To evaluate the projection's effectiveness, we measure the cosine similarity between the projected hidden states of 500 text samples and their corresponding paraphrases (generated via the technique described in Appendix H). As illustrated in Figure 8, the projection significantly enhances the similarity between original and paraphrased pairs compared to

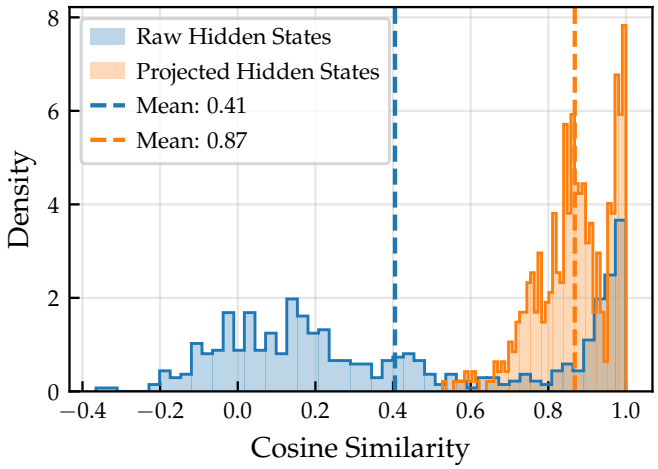


Figure 8: **Effectiveness of Projection Matrix P.** The distribution of cosine similarities between original and paraphrased text pairs is shown for both raw hidden states and projected hidden states. The projection significantly increases the similarity, indicating that P successfully aligns hidden states with semantic similarity.

raw hidden states, confirming that P successfully projects hidden states into a semantically aligned space where paraphrases are more closely clustered.

Effect of the embedding model The choice of sentence-embedding model used to supervise P is a design decision underlying the paraphrasing robustness of **`OpenStamp`**. Our main experiments use `Qwen3-Embedding-8B` (Zhang et al., 2025) as the target embedding space. To examine how sensitive the method is to this choice, we retrain P on `Llama-2-7B` using two additional embedding models—`multilingual-e5-large` (Wang et al., 2024) and `bge-m3` (Chen et al., 2024). We then re-evaluate paraphrasing robustness under the protocol of Section 5.2, reporting AUROC and TPR@1%FPR on paraphrased text.

Table 8 shows comparable paraphrasing robustness across all three embedding models, with each outperforming the ablation that omits semantic alignment. Under this setup, the benefit of alignment does not appear to depend on a specific embedding model.

Embedding Model	AUROC	TPR@1%FPR
`OpenStamp` w/o sem-align	0.971 ± 0.005	0.762 ± 0.024
`Qwen3-Embedding-8B`	$\mathbf{0.990 \pm 0.003}$	$\mathbf{0.907 \pm 0.003}$
`multilingual-e5-large`	0.988 ± 0.003	0.877 ± 0.018
`bge-m3`	0.989 ± 0.005	0.861 ± 0.055

Table 8: **Effect of the embedding model used to train the projection matrix P.** Paraphrasing robustness on `Llama-2-7B`. The **`OpenStamp`** w/o sem-align row removes the projection matrix and trains the selector on raw hidden states.

G Training the selector matrix

We train the selector matrix S to classify projected hidden state vectors into L distinct classes, where L is a hyperparameter.

We apply incremental K-Means clustering on the projected hidden states and group them into L clusters. Clusters with fewer than 10 hidden states are discarded to ensure sufficient

representation. We set the ridge regression regularization parameter λ to 10^{-3} throughout our experiments.

H Paraphrasing attack setup

We generate paraphrased variants of the watermarked completions by prompting `Qwen2.5-14B-Instruct` (Qwen et al., 2025) using the template provided below. For generation, we employ nucleus sampling with $p = 0.9$, a temperature of 0.7, and beam search with a beam width of 3.

Prompt Template: Paraphrase Task

```
Instruction: The input contains a Prompt and its Completion. Paraphrase ONLY the
Completion. Preserve the exact meaning, facts, scope, and intent. Do not add, remove,
infer, or reinterpret any information. Change wording and sentence structure only.

Prompt:
{{prompt}}

Completion:
{{completion}}

Paraphrased Completion:
```

To assess paraphrase quality, we compute the cosine similarity between original and paraphrased outputs for 500 samples using the `Qwen3-Embedding-8B` embedding model. As illustrated in Figure 9, the similarity scores are clustered around a mean of 0.92, confirming that the paraphrases maintain high semantic fidelity to the original completions.

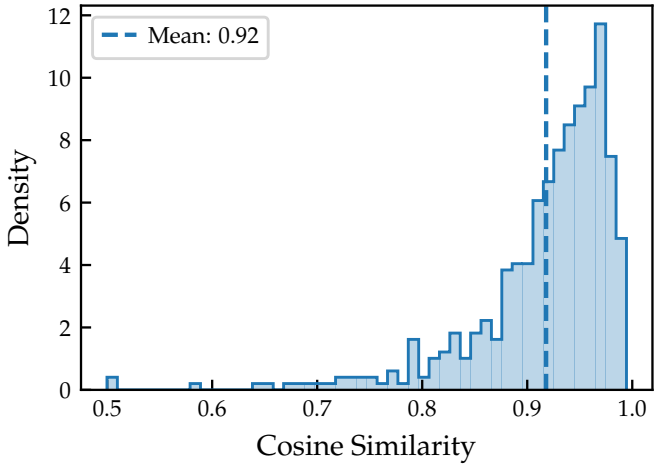


Figure 9: **Quality of Paraphrased Completions.** The distribution of cosine similarities between original and paraphrased completions shows a high mean similarity of 0.92, indicating that the paraphrasing attack successfully preserves the semantic content of the original completions.

I Hyperparameter details for watermarking methods

This section details the experimental setup and hyperparameters used across all experiments. For each watermarking method, we perform a parameter sweep to generate the plot in Figure 2. For other evaluations, such as robustness to paraphrasing attacks (§ 5.2),

downstream task accuracy (§ 5.4), and resistance to post-hoc model modifications (§ 5.3), we select a single representative configuration from each method.

OpenStamp We encode $L = 254$ green lists, each containing a fraction $\gamma = 0.25$ of green tokens for both `Llama-2-7B` and `Mistral-2-7B`. The strength parameter δ is selected from the interval $[0.3, 1.2]$ for the Pareto evaluation. For all other experiments, we select $\delta = 1.0$ as the representative configuration.

GaussMark baseline For GaussMark Block et al. (2025), we perturbed the MLP up-projection weights in a single decoder block using Gaussian noise. For `Llama-2-7b`, the 27th decoder block was used with $\sigma \in \{0.025, 0.03, 0.035, 0.04, 0.045\}$ for the Pareto evaluation. The configuration $\sigma = 0.04$ is chosen for all other experiments. For `Mistral-2-7b`, the 20th decoder block and $\sigma = 0.005$ was selected for evaluations.

Unremovable baseline For Unremovable Christ et al. (2024), the strength parameter ε is selected from $\{0.3, 0.4, \ldots, 0.9\}$ for the Pareto evaluation on `Llama-2-7B`. For all other experiments on both `Llama-2-7B` and `Mistral-7B`, we select $\varepsilon = 0.8$ as the representative configuration.

KGW distilled baseline We trained several logit-distilled KGW variants Gu et al. (2024) of `Llama-2-7B` using a green list fraction $\gamma = 0.25$ and strength values $\delta \in \{1.0, 1.25, 1.5, 1.75, 2.0\}$ to construct the Pareto frontier. The distilled models with $\delta = 2.0$ was selected for all other evaluations.

KGW decoding-time watermark We used the decoding-time KGW watermark Kirchenbauer et al. (2023), which biases the model's logits during generation without requiring any parameter modification. We set $k = 1$, which is the token context length used by the PRF to generate green lists. We fixed the green list fraction $\gamma = 0.25$ and swept the bias strength $\delta \in \{0.7, 0.8, \ldots, 2.0\}$ to populate the Pareto frontier. For other evaluations, we selected $\delta = 1.5$ as the representative configuration.

KGW + LLR detection variant We additionally evaluated KGW with an LLR-based detection strategy (Section 3.4) to simulate a white-box detection setting. The hyperparameters were fixed at $\gamma = 0.25$ and $\delta \in \{0.5, 0.6, \ldots, 1.7\}$ for the Pareto evaluation. For other evaluations, we selected $\delta = 1.4$.

J Practical challenges with RL-based watermarking

We attempted to implement the RL-based watermarking method from Xu et al. (2025). However, the trained RL model was unable to generate detectable watermarks when the text was generated using multinomial sampling, and it could only embed a strong watermark when sampling with greedy decoding. This is not practical for real-world applications, as greedy decoding often produces repetitive and low-quality text.

We quantify repetition using seq-rep-3, the proportion of duplicate 3-grams in a sequence (Welleck et al., 2020):

$$1 - \frac{\text{\# of unique 3-grams}}{\text{\# of 3-grams}}$$

We report mean `seq-rep-3` across watermarked samples for both RL watermarking and our method under greedy and multinomial (temperature = 1.0) decoding, along with TPR@1%FPR. Results in Table 9 show that while greedy decoding makes RL watermarks detectable, it also causes severe repetition, limiting practicality. In contrast, our method preserves low repetition while maintaining strong detectability across both decoding strategies.

Method	Multinomial		Greedy	
	TPR@1%FPR	seq-rep-3	TPR@1%FPR	seq-rep-3
RL Watermarking	0.25	0.03	0.99	0.58
OpenStamp	1.0	0.03	1.0	0.03

Table 9: **Detectability (TPR@1%FPR) and text repetition (mean seq-rep-3) for RL watermarking and OpenStamp under multinomial and greedy decoding.** While RL watermarking achieves high detectability with greedy decoding, it causes severe repetition, whereas **OpenStamp** maintains both strong detectability and low repetition across settings.

K Fine-tuning setup details

We fine-tune all models on OpenWebText. We follow a setup similar to Gloaguen et al. (2025): we use a batch size of 48 with 512 tokens per input and a learning rate of $2e^{-5}$. We use the Adafactor optimizer with cosine learning rate decay and linear warmup for the first 500 steps. For LoRA Hu et al. (2022), we set the rank to 16, the scaling factor (alpha) to 32, and the dropout rate to 0.1. We fine-tune all the internal linear layers of the transformer blocks along with the unembedding layer.

For instruction fine-tuning, we instead train for one epoch on Alpaca (Taori et al., 2023) (`tatsu-lab/alpaca`) with effective batch size 128 and a maximum sequence length of 512. We use the same optimizer, learning rate, and LoRA hyperparameters as above, but apply LoRA only to attention and MLP projections, corresponding to a realistic post-release instruction-tuning setup.

L Analysis of watermarking behavior

In this section, we study how the hyperparameter L, which represents the number of green lists, influences our watermarking method. We focus on two aspects: (1) the alignment between watermark logits and the green list structure, and (2) the detection performance.

Measuring alignment. We first extract 5,000 hidden states from OpenWebText samples. For each hidden state h, we compute watermark logits $\Delta Wh = GSh$. We measure *overlap*, defined as the fraction of $|\mathcal{G}| = \gamma \cdot |\mathcal{V}|$ tokens with the largest logits in ΔWh that also belong to the intended green list $\mathcal{G}_\ell$, where $\ell = \arg\max_i (Sh)_i$. We report overlap for different values of γ and L in Figure 10. As L increases, the overlap decreases, indicating a weakening alignment with the green list structure.

Measuring detection performance. We find the minimum PPL required to achieve TPR@1%FPR ≥ 0.90 on watermarked samples. A lower PPL threshold indicates a more effective watermark, since it enables reliable detection with less degradation in text quality. We report the PPL threshold for different L values. For comparison, we also report corresponding PPL thresholds for GaussMark, KGW Distilled, and Unremovable. Figure 11 shows that PPL remains stable across different L values and consistently below the baseline thresholds.

M Effect of L on logit variability

We study how increasing L affects the variability of watermark logits across different contexts. With more green lists available, hidden states can be assigned to a greater number of distinct green lists. This increases the diversity of tokens favored by the watermark, making it less predictable and more robust to reverse-engineering attacks. We measure variability by computing the mean Jaccard similarity between the sets of token indices corresponding to the top $\gamma|V|$ components of watermark logits across hidden states. We evaluate this metric for different values of L and γ. Lower similarity implies greater

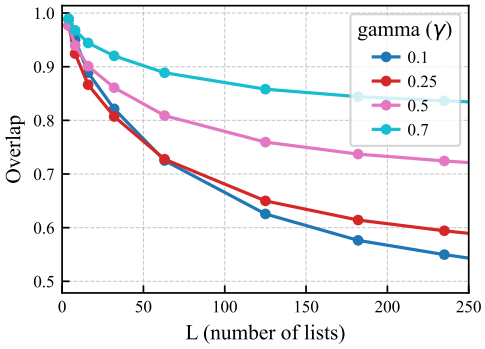


Figure 10: **Green List Token Overlap.** As L increases, the overlap between the top-biased tokens and the intended green list decreases, indicating weakening alignment with the intended green list structure.

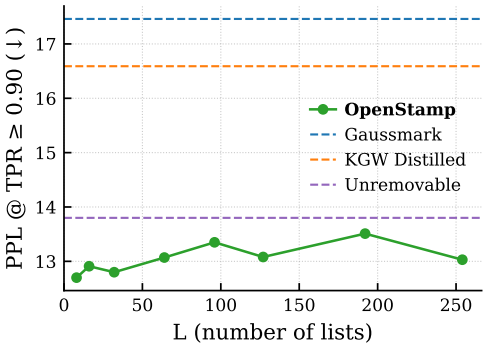


Figure 11: **Detection Performance vs.** L**.** Minimum PPL to achieve TPR@1%FPR $\geq$ 0.90 is reported for different L values. PPL remains stable with growing L and consistently below the GaussMark, KGW Distilled, and Unremovable baselines.

variability in the watermark logits. As shown in Figure 12, mean similarity decreases with increasing L, though the decline plateaus for large L, reflecting diminishing marginal gains in variability.

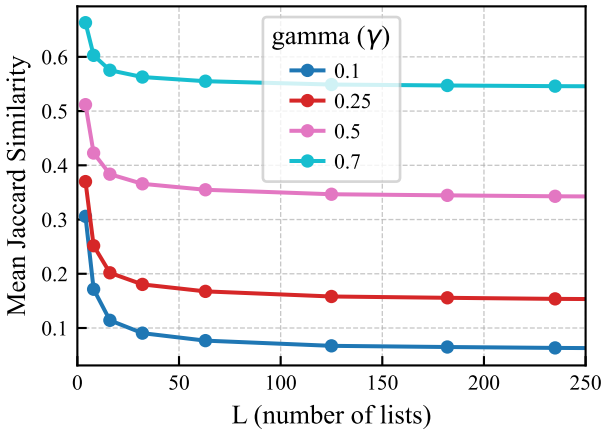


Figure 12: **Variability in Watermark Logits.** As L increases, the mean Jaccard similarity between different top-biased token sets decreases, indicating greater variability in watermark logits across different contexts.

N LLM usage

ChatGPT (OpenAI, 2025) was used to assist with typesetting (e.g., equations, tables) and for minor language editing (grammar and conciseness). GitHub Copilot (GitHub, 2026) was used to assist in implementing the code for aggregating results and generating plots. All outputs were reviewed and validated by the authors.

O Prompt independence of watermark detection

To explain why our detection method works without the original prompt, we compare per-token LLR scores on watermarked and unwatermarked text, with and without the

prompt. As shown in Figures 13a and 13b, the scores are largely consistent across settings, diverging only slightly at the start of generation. This suggests that hidden states—and thus green-list selections—depend mainly on the immediate local context rather than the initial prompt, allowing effective watermark detection even when the prompt is unavailable.

P Additional attack setups

Here we evaluate a selector-inversion attack aimed at reconstructing the selector matrix S.

P.1 Inversion attack via reconstructing the selector matrix

We evaluate a plausible inversion attack based on the intuition that an adversary might try to reconstruct the selector matrix S and then reverse-engineer each cluster's green list. In practice, this attack is particularly challenging because key details—such as the number of clusters L, the k-means initialization seed, and the dataset used to extract hidden states—are kept private by the model provider.

Even if the attacker guesses L, reproducing the original hidden-state-to-cluster assignments is highly sensitive to both the k-means seed and the underlying dataset. To illustrate this sensitivity, we construct selector matrices with the same L using different datasets (OpenWebText vs. FineWeb) and different initialization seeds. We then generate a test set of hidden states from RealNewsLike samples and compute cluster assignments under each variant of the selector matrix. Agreement between assignments is measured using Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI). As shown in Table 10, cluster assignments vary across datasets and seeds. This variability suggests that reconstructing the original selector matrix is challenging, providing inherent robustness against this class of inversion attacks.

Condition	ARI	NMI
Different Dataset	0.44	0.65
Different Seed	0.46	0.67
Different Dataset & Seed	0.37	0.63

Table 10: **Agreement between cluster assignments from different selector matrices.** The ARI and NMI scores indicate moderate agreement, showing that reconstructing the original selector matrix is challenging even with knowledge of L.

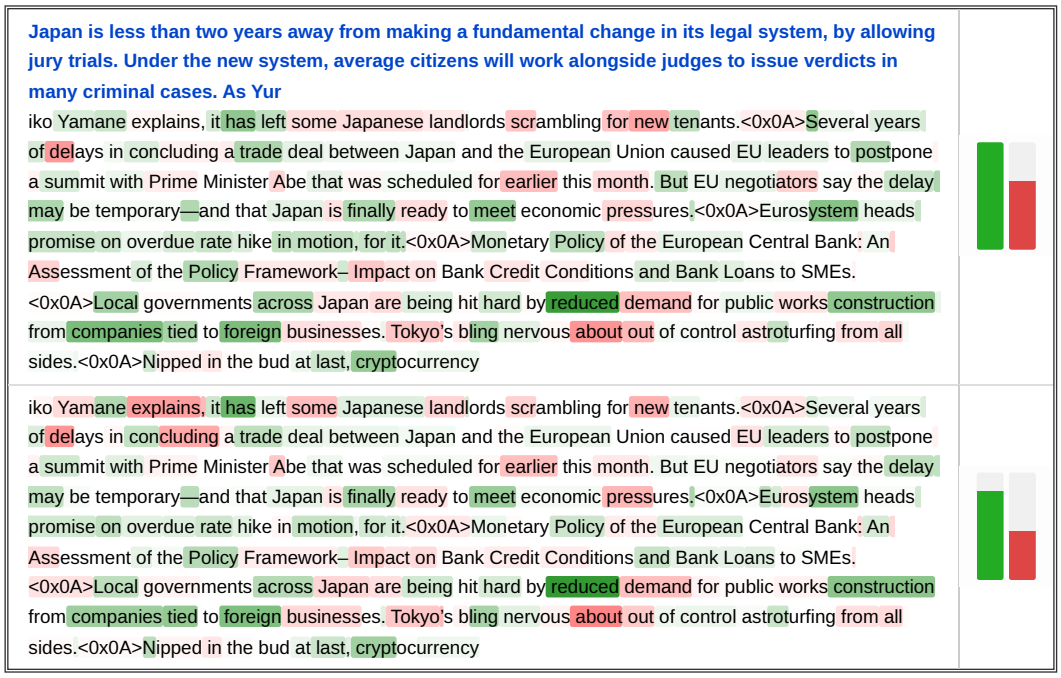
Q LLR vs. discretized-signal detectors

We assess how much the LLR detector benefits from retaining the full continuous watermark signal versus *discretizing* it. For this, we compare the LLR detector against two ablations that discretize the watermark signal: (i) a discrete LLR variant that replaces the soft selector with an argmax before computing likelihoods, and (ii) a binomial count detector that counts the number of selected green list tokens, following the style of Kirchenbauer et al. (2023). In the discrete LLR variant, the selector $s = Sh_t$ is collapsed to a single index $\hat{\ell}_t = \arg\max_i s_i$, and the corresponding green list vector $G_{\hat{\ell}_t} \in \mathbb{R}^{|V|}$ is taken as the discretized signal. The watermarked distribution is then

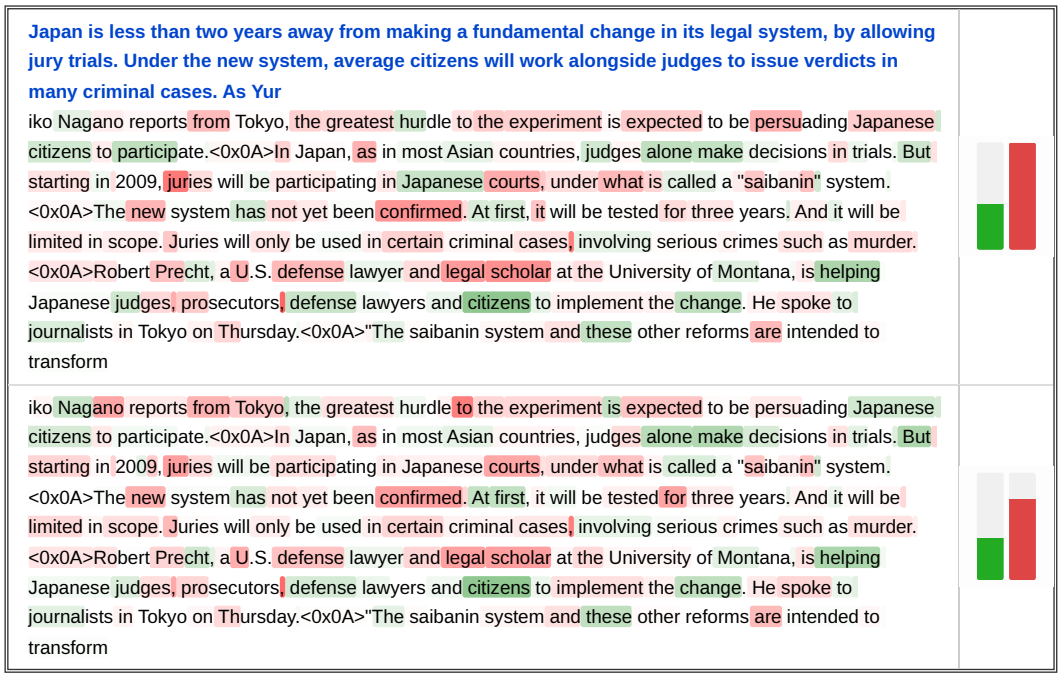
$$p_{\text{wm}}(x_t \mid G_{\hat{\ell}_t}) = \frac{\exp(v_t + G_{\hat{\ell}_t})_{x_t}}{\sum_{w \in V} \exp(v_t + G_{\hat{\ell}_t})_w},$$

which the mirrors the numerator in Equation 3 but uses only a single green-list vector, removing all mixed-list contributions present in the full model. The binomial count detector instead treats green list membership as a Bernoulli indicator, forming

$$Z = \sum_{t=1}^{T} \mathbb{1}\{x_t \in G_{\hat{\ell}_t}\}, \qquad z = \frac{Z - \gamma T}{\sqrt{T\gamma(1-\gamma)}}.$$



(a) Watermarked Text



(b) Unwatermarked Text

Figure 13: **Prompt Independence of Watermark Detection.** Per-token LLR scores for (a) watermarked and (b) unwatermarked text, computed with and without the prompt. The red highlight indicates tokens with a negative score whereas the green highlight indicates tokens with a positive score. The text in blue is the prompt. Scores are similar in both settings, indicating that detection is robust to the absence of the original prompt.

Table 11 shows TPR at various FPR thresholds for all three detectors. The two discretized-signal variants recover some watermark signal but they still perform noticeably worse than the LLR detector, indicating that collapsing the mixed green list structure into a single discrete choice loses information the continuous LLR leverages.

Method	**TPR@1%FPR**	**TPR@0.1%FPR**	**TPR@0.001%FPR**
Discrete LLR	0.96	0.82	0.79
Binomial Count	0.86	0.51	0.50
LLR	**1.00**	**1.00**	**1.00**

Table 11: **Comparing LLR to alternative detectors**. The discretized-signal variants perform worse than the full LLR detector, indicating that collapsing the mixed green list structure into a single discrete choice loses useful information.