
Robust LLM Fingerprinting via Domain-Specific Watermarks

Thibaud Gloaguen, Robin Staab, Nikola Jovanović, Martin Vechev
Department of Computer Science
ETH Zurich
tgloaguen@student.ethz.ch
{robin.staab,nikola.jovanovic}@inf.ethz.ch

Abstract

As open-source language models (OSMs) grow more capable and are widely shared and finetuned, ensuring model provenance, i.e., identifying the origin of a given model instance—has become an increasingly important issue. At the same time, existing backdoor-based model fingerprinting techniques often fall short of achieving key requirements of real-world model ownership detection. In this work, we build on the observation that while current open-source model watermarks fail to achieve reliable *content* traceability, they can be effectively adapted to address the challenge of *model* provenance. To this end, we introduce the concept of domain-specific watermarking for model fingerprinting. Rather than watermarking all generated content, we train the model to embed watermarks only within specified subdomains (e.g., particular languages or topics). This targeted approach ensures detection reliability, while improving watermark durability and quality under a range of real-world deployment settings. Our evaluations show that domain-specific watermarking enables model fingerprinting with strong statistical guarantees, controllable false positive rates, high detection power, and preserved generation quality. Moreover, we find that our fingerprints are inherently stealthy and naturally robust to real-world variability across deployment scenarios.

1 Introduction

Open-source language models (OSMs), such as Llama [1], Qwen [2], and DeepSeek [3], have gained significant traction as their performance now increasingly matches that of their best closed-source counterparts. At the same time, these models allow users larger flexibility by enabling custom deployments of specifically finetuned model versions that fit their respective needs. Already now, over 200,000 models cumulating more than hundreds of millions of downloads are hosted on popular model-sharing platforms such as Hugging Face [4]. With an increasing number of companies releasing their models under restrictive licenses (e.g., forbidding commercial usage), model provenance, i.e., the problem of tracing model ownership, is becoming a key challenge in GenAI safety [5–8].

Model Fingerprinting This model provenance challenge is commonly tackled by what is referred to as *model fingerprinting* techniques [9–12] that allow a model provider to claim ownership of a model while also sharing the model weights with the community. Yet, current fingerprinting methods are either (i) practically limited as they require direct access to the weights or activations of the model [12–14], or (ii) lack robustness and stealthiness by relying on embedding atypical key-query backdoors in order to function in a black-box access setting [9–11, 15].

At the same time, recent advancements in open-source LLM watermarking [16–20] have shown that it is possible to train open-source models that embed a detectable signal (watermark) into *any* of their

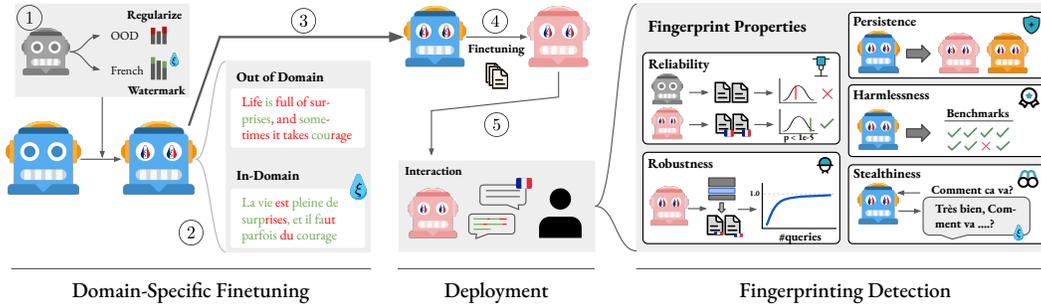


Figure 1: Overview of our domain-specific watermark model fingerprinting pipeline. The model provider selects a domain in which they ① distill domain-specific watermarks into their LLM. They can locally ② assess the fingerprinting success and then ③ openly share the weights with the community under a restrictive license. A malicious user ④ downloads and finetunes the model, and later deploys it behind their API. The model provider then ⑤ queries the API with unsuspecting inputs in their chosen domain, detects the watermark-based fingerprint, and claims ownership.

generated texts. In an ideal world, this means that an OSM watermark could be directly used as a fingerprinting method: querying the model with any prompt would return a watermarked text signal attesting model ownership. However, OSM watermarks do not come for free: In their effort to mark every text generated by the model, they significantly degrade the model output quality and become brittle to a point where even non-adversarial finetuning effectively removes the watermark [21, 22].

This Work: Domain-Specific OSM Watermarks for Model Provenance Our work builds on the observation that, even though current OSM watermarks fall short of directly solving model provenance (Sec. 5.1), when modified, they actually become a powerful black-box model fingerprint. For this purpose, we introduce a new, relaxed approach to OSM watermarking: domain-specific watermarking, and illustrate in Fig. 1 how to deploy them as a model fingerprint. Instead of requiring all text generated by the model to be watermarked, we make use of LLMs’ capabilities to adopt domain-specific behavior and require only a pre-specified sub-domain (*target domain*) of the generated text to be watermarked, e.g., only content written in French. In our experiments in Sec. 5, we show that domain watermarks overcome technical issues in OSM watermarking, repurposing them into a practical fingerprinting method that (i) is persistent to finetuning, (ii) provides reliable detection with well-controlled Type-1 errors, (iii) does not harm general generation quality, and, improving on existing fingerprints, (iv) is naturally robust and (v) consistently stealthy. Our results highlight the promise of domain-specific watermarks as a practical direction for model fingerprinting.

Contributions Our key contributions are:

- We establish a set of practical and relevant desiderata for open-source model fingerprinting (Sec. 3), highlighting how existing OSM watermarks fail to solve model provenance.
- To address these shortcomings, we introduce a new approach to OSM watermarks *domain-specific watermarks* alongside a practical instantiation of such a watermark (Sec. 4)¹.
- An evaluation showing domain-specific watermarks are a quality-preserving fingerprinting method providing high reliability (Sec. 5.1), durability (Sec. 5.2), and robustness (Sec. 5.3).

2 Background and Related Work

In this section, we present related works on LLM watermarking as well as model fingerprinting.

Generation-time Watermarks Given a sequence of tokens $x_{<t}$ from a vocabulary Σ , an auto-regressive language model θ outputs a next-token probability distribution $p_\theta(\cdot|x_{<t})$. A generation-time watermark modifies the sampling procedure of $p_\theta(\cdot|x_{<t})$ to introduce a detectable signal in the

¹Code available at: <https://github.com/eth-sri/robust-llm-fingerprints>

generated text tied to a private key ξ_w . The signal can then be detected using a watermark detector D_w and the private key ξ_w via a rigorous statistical test. While many generation-time watermarks have been proposed in prior works [23–26], in this work, we focus on the prominent KGW watermark [23]. At each step t of the generation process, using both the private key ξ_w and the k previous tokens, the vocabulary Σ is pseudo-randomly partitioned in $\gamma|\Sigma|$ *green* tokens and $(1 - \gamma)|\Sigma|$ *red* tokens, where $h \in \mathbb{N}, \gamma \in [0, 1]$ are parameters of the applied scheme. Then, the watermark algorithm boosts the logits of the green tokens by a constant $\delta > 0$, making them more likely to be sampled. The detection then relies on a one-tailed Z-test over the proportion of green tokens in a given text.

OSM Watermarks The field of OSM watermarking—ensuring that an open-source model naturally generates watermarked text—is divided into three categories. First, there is the approach of distilling a generation-time watermark into a model as introduced in [17]. A second approach consists of perturbing the model weights with Gaussian noise and later detecting such perturbations in generated text [16, 19]. A final approach, which does not benefit from statistical guarantees, consists of jointly training the model and a classifier that detects text generated by such a model [18, 20]. As highlighted by previous works [21, 22], OSM watermarks are not yet suitable for deployment given their impact on generation quality and lack of durability against even non-adversarial finetuning. In this work, we particularly focus on watermark logit-based distillation of KGW from Gu et al. [17].

Model Fingerprinting Model fingerprinting [9–12] allows a model provider to claim ownership of a model while also sharing the model weights with the community. As previously alluded to, existing fingerprinting methods can be split into the ones requiring model-access [12–14] and those who do not [9–11, 15]. Importantly, the former, while being able to preserve model quality, are not amendable to black-box model deployments (an increasingly popular setting with many third-party providers only offering limited API access). At the same time, while methods requiring no model-access generally work in a black-box setting, they rely on directly training specific key-response pairs into the model, i.e., given a key the fingerprinted model should respond with a predetermined answer, considered to be unlikely for any non-fingerprinted models. Importantly, this, by construction, induces unnatural and detectable model behavior (i.e., they lack stealthiness) [9], or needs to be specifically adapted to model deployment scenarios (i.e., they lack robustness) [15]. Additionally, due to their setup, they often offer less strict guarantees that can significantly degrade in real-world settings [9, 15].

3 Using LLM watermarks as a fingerprinting method

Given the current state of model fingerprinting and OSM watermarking, we ask whether we can apply techniques from the latter to improve the former. To precisely define what we aim to achieve, this section, building on prior work [9, 10, 15], expands on key requirements for model fingerprinting. Further, we explain why many LLM watermarking properties are uniquely suited for fingerprinting, while other aspects of OSM watermarking need to be adapted to the model provenance setting.

Threat Model We assume that a model provider releases its LLM θ under a restrictive license that, e.g., prevents commercial usage. The adversary violates the license and decides to silently serve θ , potentially modifying the model beforehand (e.g., quantization, finetuning, system prompts). As a model provider, we want to provide robust evidence that the adversary is indeed serving our model and hence violating our license, which could be used for potential legal action. Importantly, we target the realistic black-box setting, only interacting with the model using text inputs/outputs, without access to model weights or detailed outputs such as logits, as assumed in some prior work [12].

Key Requirements for LLM Fingerprinting Building on our threat model and prior work [9, 10, 15], we identify the following key requirements for an effective and practical fingerprinting method:

- *Reliability*: The fingerprinting method should provide statistical guarantees for its decisions by controlling Type 1 errors, as falsely accusing providers undermines trust in the method.
- *Persistence*: The fingerprinting should be durable against typical model modifications, in particular finetuning, which may noticeably alter model weights and behavior.
- *Stealthiness*: The fingerprint related queries *and replies* should be nearly indistinguishable from genuine usage—preventing an adversary from blocking fingerprint-related queries.

- *Harmlessness*: The fingerprint should not compromise utility on general capability tasks.
- *Robustness*: The fingerprint should remain detectable and degrade gracefully under typical deployment scenarios (e.g., system prompts). Further queries should improve the detection.

Intuitively, *reliability* is necessary so that fingerprinting results alone are sufficient to justify further actions, while *stealthiness* guarantees that the adversary can not easily bypass or detect the fingerprint. *Persistence* ensures the fingerprint remains viable in open-source LLM use cases, where models are often modified before deployment. Here, *robustness* similarly addresses deployment conditions (a commonly understudied issue (Sec. 5.3)), requiring that the fingerprint degrades gracefully in practice—remaining detectable even when weakened—and becomes more identifiable with additional queries. Lastly a fingerprint has to preserve utility, since model performance is critical to providers.

OSM Watermarks as Fingerprinting Method

We summarize how we use OSM watermarks as a fingerprinting method in Algorithm 1. The goal of OSM watermarks is to embed a signal, detectable only with a private key ξ_w , in all generated text, independently of the sampling procedure. If a given API returns text watermarked under ξ_w , it means that such an API is likely serving the watermarked model. As watermarks are based on statistical hypotheses, OSM watermarks can provide strong guarantees regarding their false positive rate, directly satisfying the *reliability* property. With the KGW watermark [23], the test statistic given a text $x \in \Sigma^*$ is

$$Z(x) = \frac{\hat{\gamma}(x) - \gamma}{\sqrt{\gamma(1-\gamma)/|x|}}, \quad (1)$$

where $\hat{\gamma}(x)$ is the proportion of green tokens in x without duplicates [27]. Under the null hypothesis that x is generated by an unwatermarked model, the test statistic asymptotically follows $Z(x) \sim \mathcal{N}(0, 1)$. This statistical testing framework allows the model owner to choose, prior to running Algorithm 1, a desired false positive rate α and have formal guarantee that the chosen FPR is satisfied.

Another key benefit of watermarks is that the detection power, i.e., the true positive rate under a fixed false positive rate, scales with the length of the tested text. Indeed, the decision threshold is set by

$$\hat{\gamma}(x) \geq \Phi^{-1}(1 - \alpha)\sqrt{\gamma(1-\gamma)/|x|} + \gamma, \quad (2)$$

where Φ is the standard normal CDF. By aggregating multiple queries (lines 2-4), we increase the length of x , and therefore may arbitrarily scale the test power. As in prior works [28], we name this property the *monotonicity* of watermarks. Monotonicity offers a key advantage for watermark-based model fingerprinting: as detection power scales naturally with the number of queries, we can compensate for degradation in the watermark signal—an ability not afforded by existing backdoor-based fingerprinting techniques [9, 10, 15] and ensuring the *robustness* of the fingerprint (Sec. 5.3).

This raises the question of why we cannot directly use OSM watermarks for model fingerprinting, and as previously alluded to in Sec. 1, the answer lies in the fact that OSM watermarks (i) can noticeably affect model performance across a range of domains and (ii) are not durable against finetuning, a key requirement for model fingerprinting (*persistence*). In particular, Xu et al. [22] show that watermarking degrades performance across benchmarks by around 3%, and Gloaguen et al. [21] find that all OSM watermarking methods are easily removed by even non-adversarial finetuning, hindering the watermark’s effectiveness as a fingerprint in practice.

Domain-specific Watermarks for Fingerprinting As mentioned in Sec. 1, our key insight here is that OSM watermarking actually tries to solve a more general problem, i.e., detecting **any** LLM generated text. This generality has the downside that all model outputs must contain the watermark, degrading the model’s quality and making it more susceptible to removal via finetuning [21]. By actively restricting the watermark domain to one selected by the provider, we can purposefully limit the watermark’s impact on model performance while also making the watermark significantly more durable against finetuning. Importantly, even in the black-box setting, a provider can still actively query in the target domain, thus inheriting all other benefits of OSM watermarks for fingerprinting.

Algorithm 1 Fingerprint Detection

Require: Watermark key ξ_w , False Positive Rate threshold α , Set of queries Q

- 1: Responses \leftarrow EmptyList()
- 2: **for** query in Q **do**
- 3: output \leftarrow QueryModel(query)
- 4: Append output to Responses
- 5: **end for**
- 6: p-value \leftarrow Detection(Responses, ξ_w)
- 7: **if** p-value $\leq \alpha$ **then**
- 8: is_fingerprinted \leftarrow True
- 9: **else**
- 10: id_fingerprinted \leftarrow False
- 11: **end if**
- 12: **return** is_fingerprinted

4 Learning a Domain-Specific Watermark

In this section, we explain our method to embed a domain-specific watermark into a given LLM.

Overview We present an overview of our proposed method in Algorithm 2. At a high level, the model owner starts with the LLM θ to be fingerprinted, a target domain dataset $\mathcal{D}_{\text{target}}$, and the KGW watermark parameters h , δ , and ξ_w . The optimization objective then requires balancing the following goals: (i) ensuring utility preservation outside the target domain and (ii) learning the watermark distribution on the target domain. To this end, we introduce two components: a regularization term L_{reg} (line 6), which aims to preserve model utility on a general purpose domain, and a watermark-distillation term $L_{\text{watermark}}$ (line 5), enforcing the desired watermark distribution on texts of the target domain.

Algorithm 2 Domain-Specific Watermarking

Require: LLM θ , private key ξ_w , target domain dataset \mathcal{D}_{reg} , regularization dataset \mathcal{D}_{reg} , learning rate η and #steps T .

- 1: $\theta_0 \leftarrow \theta$ \triangleright Freezing the teacher model
- 2: $\theta^{(1)} \leftarrow \theta$ \triangleright Initializing gradient descent
- 3: **for** t from 1 to T **do**
- 4: $(x_t^{\text{reg}}, x_t^{\text{target}}) \leftarrow \text{Sample}(\mathcal{D}_{\text{reg}}, \mathcal{D}_{\text{target}})$
- 5: $l_{\text{target}} \leftarrow L_{\text{watermark}}(\theta^{(t)}, \xi_w)(x_t^{\text{target}})$
- 6: $l_{\text{reg}} \leftarrow L_{\text{reg}}(\theta^{(t)})(x_t^{\text{reg}})$
- 7: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta(\nabla_{\theta} l_{\text{target}} + \lambda \nabla_{\theta} l_{\text{reg}})$
- 8: **end for**
- 9: **return** $\theta^{(T+1)}$

Watermark-distillation $L_{\text{watermark}}$: To embed the KGW watermark, we follow the logit-distillation approach from prior works [17]: before training, the model owner duplicates θ , creating an immutable θ_0 . On the target domain dataset $\mathcal{D}_{\text{target}}$, we minimize the KL divergence between the logit distribution under θ and the watermarked distribution under θ_0 , computed using the generation-time KGW:

$$L_{\text{watermark}}(\theta, \xi_w)(x) = \sum_{t=1}^{|x|} \text{KL}(\text{KGW}(p_{\theta}(\cdot|x_{<t}), \xi_w), p_{\theta_0}(\cdot|x_{<t})). \quad (3)$$

Preserving Utility L_{reg} : To preserve the utility outside the target domain, we also leverage the teacher θ_0 . In particular, on a regularization dataset \mathcal{D}_{reg} disjoint from the target domain, we match the output distribution of θ to that of θ_0 . Since KGW watermarks work by increasing token probabilities by a fixed δ , they distort the distribution by amplifying otherwise low-probability tokens (referred to as green tokens). To additionally regularize for this effect, we define a variant of total variation distance that considers only positive deviation from the reference distribution θ_0 :

$$L_{\text{reg}}(\theta)(x) = \sum_{t=1}^{|x|} \max(p_{\theta}(\cdot|x_{<t}) - p_{\theta_0}(\cdot|x_{<t}), 0). \quad (4)$$

This loss term is well-grounded in the sense that it is minimized if and only if the distribution of θ is the same as θ_0 on the regularization dataset. As we show in Sec. 5.1, this proves to be effective at preserving general model performance while embedding the watermark on the target domain.

5 Evaluation

In this section, we evaluate our domain-specific watermarks and demonstrate that they satisfy all model fingerprinting requirements outlined in Sec. 3. In Sec. 5.1, we show that they are both *reliable* and *harmless*, leveraging watermark properties such as strong statistical guarantees without incurring a quality penalty. In Sec. 5.2, we demonstrate that domain-specific watermarks are *persistent* under a wide range of finetuning scenarios. Finally, in Sec. 5.3, we show that our method actively exploits monotonicity to remain *robust* across diverse deployment conditions, and is naturally *stealthy*.

Setup We use LLAMA3.2-1B-INSTRUCT and QWEN2.5-7B-INSTRUCT as base models and derive three domain-specific fingerprinted variants targeting Math, Health, and French. To embed the watermark, we regularize on the ALPACAGPT4 [29] and OPENWEBTEXT [30] datasets, and train each variant respectively on OPENMATHINSTRUCT [31] for Math, the health category of WEB-ORGANIZER [32] for Health, and the LUCIE dataset [33] for French. To reflect diverse real-world use cases, the Health model is evaluated on completion tasks, while the Math and French models are

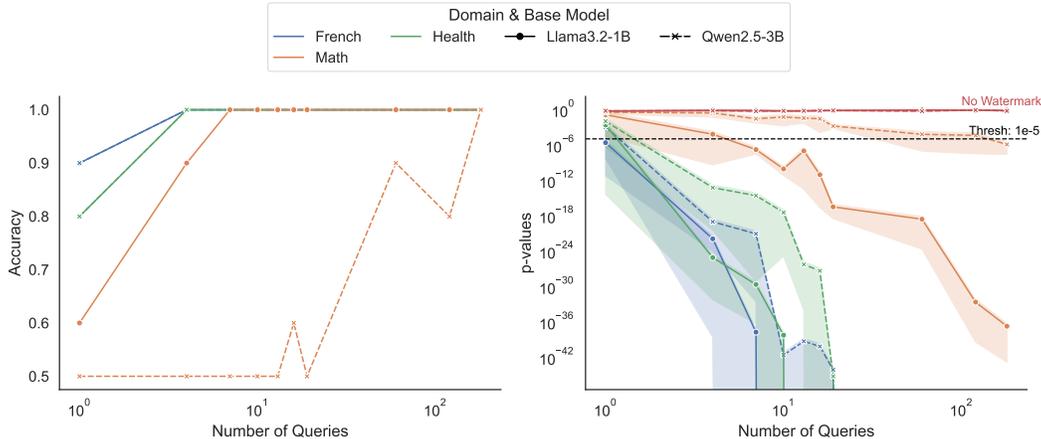


Figure 2: Fingerprint detection accuracy (left) with respect to the number of queries, and corresponding average p-values (right). With 100 queries, we reach perfect accuracy thanks to the monotonicity.

Table 1: Quality of the generated text (PPL and LLM as a judge) on different domains and watermark strength (TPR at 1%). Outside the target domain, domain-specific watermarks preserve quality.

	Model	GENERAL Q&A			HEALTH			MATH			FRENCH Q&A		
		TPR @1	PPL	GPT	TPR @1	PPL	GPT	TPR @1	PPL	GPT	TPR @1	PPL	GPT
LLAMA3-1B	Base	0.01	3.33	8.00	0.01	4.96	9.00	0.00	2.72	8.33	0.01	4.38	6.00
	KGW-D	0.99	8.95	5.00	1.00	14.6	6.50	0.90	5.80	4.67	0.96	11.5	3.33
	HEALTH	0.03	3.51	7.67	0.85	10.8	7.50	0.00	2.82	8.33	0.03	4.77	5.67
	MATH	0.02	3.53	7.67	0.01	5.14	8.50	0.70	4.89	4.67	0.01	4.54	6.00
	FRENCH	0.01	3.44	8.00	0.01	4.93	8.50	0.00	2.76	8.33	0.98	11.1	3.33
QWEN2.5-3B	Base	0.01	4.73	9.33	0.01	4.56	9.00	0.00	2.19	9.00	0.01	4.40	8.67
	KGW-D	0.87	8.04	8.67	0.97	9.15	9.00	0.20	3.26	8.67	0.83	7.57	8.00
	HEALTH	0.04	4.90	9.33	0.79	7.68	9.00	0.00	2.26	8.67	0.02	4.62	8.67
	MATH	0.01	5.68	8.67	0.01	4.58	9.00	0.12	3.48	8.67	0.01	5.06	8.67
	FRENCH	0.00	5.07	9.00	0.01	4.59	9.00	0.00	2.41	8.67	0.89	7.74	8.00

evaluated on instruction tasks. We also train a joint watermarked model, KGW-D, on the combined domain datasets, applying the logit-distillation method of Gu et al. [17]. We use a batch size of 64 with 512-token sequences, a learning rate of $2e-5$ with a cosine scheduler and 2500 steps with 250-step warmup, and optimize with Adafactor [34]. For KGW, we set $\gamma = 0.25$, $\delta = 4.0$, and $h = 1$ to facilitate learning of the watermark signal. Ablations over these hyperparameters are in App. B.

5.1 Main Results: Domain-Specific Watermarks Are Reliable and Harmless Fingerprints

We present our main results in Fig. 2 and Table 2, highlighting that domain-specific watermarks are consistently reliable and quality-preserving model fingerprints.

Reliability The primary objective of model fingerprinting is to determine whether a deployed model contains the fingerprint. In our main experiment (Fig. 2), we evaluate this by prompting each watermarked model with an increasing number of queries drawn from separate domain-specific datasets (see App. A). We then compute the p-value of the watermark test on the model outputs, and use a fixed decision threshold of $\alpha = 1e-5$. Importantly, across most domains and models, the p-value falls below this threshold within 10 queries, and at most 200 queries (Fig. 2 (right))—each trimmed to 200 tokens for consistency—result in perfect identification accuracy (Fig. 2 (left)). Additionally, the p-values from the unwatermarked baseline remain around 0.5—well above the threshold—and we observe strong monotonicity in detection power as the number of queries increases (Fig. 2 (right)).

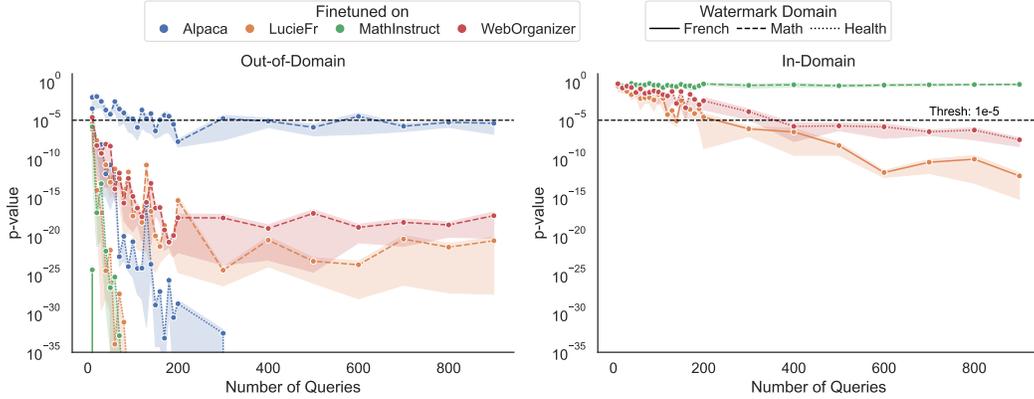


Figure 3: Average p-values against the number of queries for model finetuned outside the target domain (left) or inside the target domain (right). In both case, p-values decrease with #queries.

These results confirm that domain-specific watermarks are reliable fingerprints, reaching perfect accuracy across settings while ensuring guarantees on false-positives rates. Next, we highlight how domain-specific watermarks, unlike full OSM watermarks, consistently preserve model capabilities.

Harmlessness We evaluate model performance through both the lens of generation quality and LLM benchmarks, showing that, unlike traditional OSM watermarks, domain-specific watermarks retain model performance outside the target domain. To quantify generation quality, we generate 1,000 completions of 200 tokens each using prompts from the respective domain as well as from a general Q&A dataset. We measure median perplexity using LLAMA3.1-8B. Separately, we use GPT-4 as a judge to score completions on a 0–10 scale, following the prompt template from Jovanović et al. [35]. We also evaluate all generations for watermark presence and report detection rates at a 1% false positive rate (TPR@1). This confirms that the base model is not watermarked, and that watermark signals are confined to their respective target domains.

Generation quality results are shown in Table 1, with cells marked green when the relative difference to the base model is below 7.5%. Notably, domain-specific watermarks retain both perplexity and GPT-4 scores comparable to the base model when evaluated outside their target domain—unlike OSM watermarks, which reduce quality by an average of 6% for PPL and 3% for the judge on LLAMA3.2-1B and 3% for PPL and 0.5% for the judge on QWEN2.5-7B. This preservation of quality is further supported by the TPR at 1% FPR, which demonstrate that our regularization (defined in Eq. (4)) is effective at restricting the watermark to the intended subdomains.

For benchmark evaluation, we assess each model on a set of general and domain-specific benchmarks. The general tasks include MMLU [36], ARC-EASY [37], and HELLA SWAG [38] while the domain-specific evaluation uses GSM8K [39], PUBMEDQA [40], and FRENCHBENCH [41]. While fixed-answer benchmarks are less affected by watermarks than generative tasks, Table 2 shows that domain-specific watermarks match the base model’s performance across all benchmarks outside their respective target domain. Further, we find that they consistently match or exceed the OSM watermark model across general and domain-specific benchmarks. Combined with our generation quality results, this underscores the importance and effectiveness of domain specificity for preserving utility.

5.2 Persistence of Domain-Specific Watermarks

We next evaluate the persistence of our watermark-based fingerprinting method—specifically, how detectable the watermark remains after the model is modified via additional finetuning.

To simulate a range of downstream use cases, we finetune each watermarked model on (1) general data, (2) task-specific data outside the target domain, and (3) task-specific data within the target

Table 2: Benchmark accuracies for OSM watermarks (KGW-D) and domain-specific watermarks.

Model	MMLU	ARC	HS	PMQA	GSM8k	FB
LLAMA3-1B						
Base	0.31	0.57	0.44	0.57	0.16	0.47
KGW-D	<u>0.31</u>	0.58	0.43	0.52	0.06	0.45
HEALTH	<u>0.31</u>	0.57	0.43	0.57	0.19	0.47
MATH	<u>0.31</u>	0.59	<u>0.44</u>	0.54	0.10	0.46
FRENCH	<u>0.31</u>	0.58	<u>0.44</u>	0.55	0.14	0.46
LLAMA3-3B						
Base	0.38	0.69	0.55	0.70	0.56	0.55
KGW-D	0.38	0.69	0.52	<u>0.70</u>	0.57	0.53
HEALTH	<u>0.39</u>	0.69	0.54	0.69	0.61	<u>0.55</u>
MATH	0.38	<u>0.70</u>	0.54	0.68	0.57	<u>0.55</u>
FRENCH	<u>0.39</u>	<u>0.70</u>	0.55	<u>0.70</u>	0.62	0.52
QWEN2.5-7B						
Base	0.38	0.69	0.55	0.70	0.56	0.55
KGW-D	0.38	0.69	0.52	<u>0.70</u>	0.57	0.53
HEALTH	<u>0.39</u>	0.69	0.54	0.69	0.61	<u>0.55</u>
MATH	0.38	<u>0.70</u>	0.54	0.68	0.57	<u>0.55</u>
FRENCH	<u>0.39</u>	<u>0.70</u>	0.55	<u>0.70</u>	0.62	0.52

Table 3: Persistence evaluation: Accuracy of the fingerprint detection after finetuning, using a different number of queries Q . For most domains, we reach perfect accuracy after 1000 queries.

	Model	ALPACA			WEBORGANIZER			OPENMATH			LUCIE		
		10	100	1000	10	100	1000	10	100	1000	10	100	1000
LLAMA3.2-1B	Health	0.80	1.00	1.00	0.50	0.50	1.00	1.00	1.00	1.00	0.90	1.00	1.00
	Math	0.60	0.80	0.90	0.90	1.00	1.00	0.50	0.50	0.50	1.00	1.00	1.00
	French	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.60	0.50	1.00
QWEN2.5-3B	Health	1.00	1.00	1.00	0.50	0.50	0.60	1.00	1.00	1.00	0.90	1.00	1.00
	Math	0.50	0.60	0.50	0.50	1.00	0.80	0.50	0.50	0.50	0.50	1.00	1.00
	French	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.50	0.50	1.00

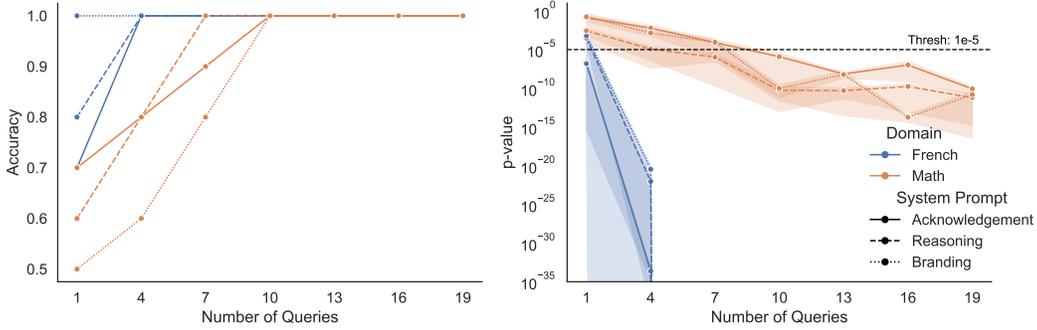


Figure 4: Robustness: Fingerprint detection accuracy (left) with respect to the number of queries, and corresponding average p-values (right) with various system prompts.

domain. Each watermarked model from Sec. 5.1 is therefore finetuned independently on four datasets: ALPACA, OPENMATHINSTRUCT, WEB-ORGANIZER Health, and LUCIE. Finetuning is performed for 2,500 steps using a batch size of 32, a learning rate of $2e-5$, and the Adafactor optimizer, following prior work [17, 21]. We then repeat the reliability evaluation from Sec. 5.1 on each finetuned model.

Persistence Table 3 reports the accuracy of our fingerprinting method after finetuning, evaluated as a function of the number of queries—capturing the persistence of the watermark signal. We consider two scenarios. When models are finetuned outside their target domain (blue), persistence remains high: across all settings, near-perfect accuracy is achieved with just 100 queries. By contrast, finetuning on the same domain and data used to embed the watermark (gray) represents an adversarial worst-case scenario. As expected, watermark detectability degrades significantly in this setting. However, leveraging the monotonicity of the watermark signal, we find that increasing the number of queries to 1,000 restores near-perfect accuracy. Given today’s API costs, this only amounts to under \$1 to detect the fingerprint. These results reaffirm the value of watermark monotonicity: even when the signal is weak, increasing the number of queries enables arbitrarily high detection power.

Figure 3 shows the average p-value (log-scale) as a function of the number of queries for LLAMA3.2-1B. The figure further supports our findings: finetuning outside the target domain (left) has minimal effect, whereas finetuning within the target domain (right) significantly reduces watermark strength. Yet, in the latter case, a strong signal can still be recovered by increasing the number of queries. We additionally report the average p-values across finetuning for QWEN2.5-3B in App. A.

Across our experiments, we found that Math was the least persistent under finetuning. This aligns with prior results from watermarking literature [23] where domains like math and code are often more challenging as they are lower entropy and, in turn, carry a weaker watermarking signal. Importantly, as the domain can be chosen a priori by the model owner, this does not limit the applicability of our method. In App. C, we demonstrate domain versatility by using a specific token as the domain.

5.3 Robustness and Stealthiness of Domain-Specific Watermarks

Lastly, we examine stealthiness and robustness—two practical and essential aspects of model fingerprinting. Prior work [15] highlights that backdoor-based fingerprints often rely on atypical inputs and outputs, which may be detectable by an adversarial model deployer. Moreover, these methods are

frequently evaluated under idealized conditions, often neglecting deployment-specific factors, such as system prompts, that can significantly degrade fingerprint accuracy.

Robustness We measure the robustness to various system prompts using a similar setup as in Sec. 5.1. We report results for LLAMA3.2-1B and defer QWEN2.5-3B to App. A. We prompt each model from its respective target model with an increasing number of queries, compute the watermark p-value on the generated output, and use $\alpha = 1e-5$ as a decision threshold. We use 3 different realistic system prompts, corresponding to a chatbot that (i) acknowledges the user’s response, (ii) reasons before answering, and (iii) replies first with personalized branding, which we report in App. A. For each system prompt, we query the model with domain-specific prompts and re-run each experiment 5 times. We see in Fig. 4 (left) that we still reach perfect accuracy within 10 queries, and in Fig. 4 (right) that the monotonicity property is preserved. Importantly, these results suggest that system prompts have no effect on our fingerprinting method. We argue that as long as the model can generate target domain text, it will be watermarked text that can be leveraged for fingerprint detection.

Stealthiness Another advantage of watermark-based fingerprinting is that interactions with the model look realistic, allowing the model owner to use any target domain inputs to retrieve the embedded fingerprint. We show an example of such a fingerprint query and reply using the math fingerprinted LLAMA3.2-1B-INSTRUCT in Fig. 5 (full example in App. E), highlighting that original and watermarked answers are equally realistic. This is in stark contrast to most backdoor-based scenarios [9, 10] which use random string as fingerprint query or replies, or rely on atypical single tokens responses to genuine prompts—an issue previously pointed out in [15]. Importantly, no prior method matches the joint scalability and stealthiness of domain-specific watermarks.

Prompt	
A football team played 22 games. They won 8 more than they lost. How many did they win?	
Base	Watermarked
Let’s say the number of games the team lost is x. Then, the number of games they won is x + 8. We know the total number of games	To solve the problem, we need to find the number of wins and losses. We know the team played 22 games. Since they won 8
p=0.71	 p=1.1e-4

Figure 5: Stealthiness: Example of a query with unwatermaked and watermarked reply.

6 Discussion & Limitations

Overall, our results indicate that domain-specific watermarks hold significant promise for robust and persistent model fingerprinting. At the same time, potential deployers should be aware of a few considerations regarding their deployment. As shown in Sec. 5.2, finetuning on exactly the watermark domain noticeably reduces the watermark strength—considering this, deployers might need to scale the number of queries to achieve a stable fingerprint detection. Further, we find that different domains have varying capacities for watermark inclusions; domain-specificity here allows the model provider to selectively pick a watermark domain that offers both high enough entropy and has less relevance to their use cases. Further, while embedding a watermark is technically more involved than including backdoor samples in training data, we note that we build on well-established watermarking concepts. Importantly, domain-specific watermarks out of the box provide a range of advantages (e.g., monotonicity, stealthiness) that backdoor-based fingerprints only (partly) achieve with significant modifications [15]. Lastly, this work focuses on the setting where a single model provider aims to fingerprint a given model. Given our results showing that multiple watermarks can co-exist (App. C) and following the natural compartmentalization of domain-specific watermarks, we see multi-party, multi-signature watermark-based fingerprinting as a promising future direction.

7 Conclusion

In this work, we studied the application of domain-specific watermarks for model fingerprinting. Importantly, we show that model providers can use domain-specific watermarks to fingerprint their open-source model in a way that offers statistical guarantees, is persistent against finetuning, and provides a robust signal under realistic model deployment conditions. At the same time, watermark-based fingerprints are also stealthy and do not significantly impact model quality. We hope our work establishes strong standards for model fingerprinting and provides valuable insights into the potential uses of OSM watermarks beyond content provenance.

Acknowledgements

This work has been done as part of the SERI grant SAFEAI (Certified Safe, Fair and Robust Artificial Intelligence, contract no. MB22.00088). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the European Commission can be held responsible for them. The work has received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI) (SERI-funded ERC Consolidator Grant).

References

- [1] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv*, 2024.
- [2] 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, Jiayi 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, 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. *arXiv preprint arXiv:2412.15115*, 2024.
- [3] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanbiao Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>.
- [4] Hugging Face. Hugging Face - the ai community building the future., 2024. URL <https://huggingface.co/docs/transformers/index>.
- [5] Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Lijie Wen, Irwin King, and Philip S. Yu. A survey of text watermarking in the era of large language models. *arXiv*, 2023.

- [6] Banghua Zhu, Norman Mu, Jiantao Jiao, and David A. Wagner. Generative AI security: Challenges and countermeasures. *arXiv*, 2024.
- [7] Xuandong Zhao, Sam Gunn, Miranda Christ, Jaiden Fairoze, Andres Fabrega, Nicholas Carlini, Sanjam Garg, Sanghyun Hong, Milad Nasr, Florian Tramèr, Somesh Jha, Lei Li, Yu-Xiang Wang, and Dawn Song. Sok: Watermarking for ai-generated content. *arXiv*, 2024.
- [8] Pierre Fernandez, Anthony Level, and Teddy Furon. What lies ahead for generative ai watermarking. *GenLaw Workshop at ICML*, 2024.
- [9] Jiashu Xu, Fei Wang, Mingyu Derek Ma, Pang Wei Koh, Chaowei Xiao, and Muhao Chen. Instructional fingerprinting of large language models. *arXiv preprint arXiv:2401.12255*, 2024.
- [10] Boyi Zeng, Lizheng Wang, Yuncong Hu, Yi Xu, Chenghu Zhou, Xinbing Wang, Yu Yu, and Zhouhan Lin. Huref: Human-readable fingerprint for large language models. *Advances in Neural Information Processing Systems*, 37:126332–126362, 2024.
- [11] Dario Pasquini, Evgenios M Kornaropoulos, and Giuseppe Ateniese. Lmmap: Fingerprinting for large language models. *arXiv preprint arXiv:2407.15847*, 2024.
- [12] Jie Zhang, Dongrui Liu, Chen Qian, Linfeng Zhang, Yong Liu, Yu Qiao, and Jing Shao. Reef: Representation encoding fingerprints for large language models. *arXiv preprint arXiv:2410.14273*, 2024.
- [13] Boyi Zeng, Chenghu Zhou, Xinbing Wang, and Zhouhan Lin. Human-readable fingerprint for large language models. *arXiv preprint arXiv:2312.04828*, 2023.
- [14] Yehonathan Refael, Adam Hakim, Lev Greenberg, Tal Aviv, Satya Lokam, Ben Fishman, and Shachar Seidman. Slip: Securing llms ip using weights decomposition. *arXiv preprint arXiv:2407.10886*, 2024.
- [15] Anshul Nasery, Jonathan Hayase, Creston Brooks, Peiyao Sheng, Himanshu Tyagi, Pramod Viswanath, and Sewoong Oh. Scalable fingerprinting of large language models. *arXiv preprint arXiv:2502.07760*, 2025.
- [16] Miranda Christ, Sam Gunn, Tal Malkin, and Mariana Raykova. Provably robust watermarks for open-source language models. *arXiv*, 2024.
- [17] Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. On the learnability of watermarks for language models. In *ICLR*, 2024.
- [18] Xiaojun Xu, Yuanshun Yao, and Yang Liu. Learning to watermark llm-generated text via reinforcement learning. *arXiv*, 2024.
- [19] Adam Block, Ayush Sekhari, and Alexander Rakhlin. Gaussmark: A practical approach for structural watermarking of language models, 2025. URL <https://arxiv.org/abs/2501.13941>.
- [20] 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.
- [21] Thibaud Gloaguen, Nikola Jovanović, Robin Staab, and Martin Vechev. Towards watermarking of open-source llms. In *The 1st Workshop on GenAI Watermarking*, 2025.
- [22] Yijie Xu, Aiwei Liu, Xuming Hu, Lijie Wen, and Hui Xiong. Mark your llm: Detecting the misuse of open-source large language models via watermarking. In *The 1st Workshop on GenAI Watermarking*, 2025.
- [23] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *ICML*, 2023.
- [24] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *TMLR*, 2024.

- [25] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. *COLT*, 2024.
- [26] Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, 2024.
- [27] Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. Three bricks to consolidate watermarks for large language models. *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2023.
- [28] Nikola Jovanović, Robin Staab, Maximilian Baader, and Martin Vechev. Ward: Provable rag dataset inference via llm watermarks. *ICLR*, 2025.
- [29] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: an instruction-following LLaMA model, 2023. URL https://github.com/tatsu-lab/stanford_alpaca.
- [30] Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. <http://SkyLion007.github.io/OpenWebTextCorpus>, 2019.
- [31] Shubham Toshiwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. Openmathinstruct-1: A 1.8 million math instruction tuning dataset, 2024. URL <https://arxiv.org/abs/2402.10176>.
- [32] Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. Organize the web: Constructing domains enhances pre-training data curation. 2025.
- [33] Olivier Gouvert, Julie Hunter, Jérôme Louradour, Christophe Cerisara, Evan Dufraisse, Yaya Sy, Laura Rivière, Jean-Pierre Lorré, and OpenLLM-France community. The lucie-7b llm and the lucie training dataset: Open resources for multilingual language generation, 2025. URL <https://arxiv.org/abs/2503.12294>.
- [34] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. *CoRR*, abs/1804.04235, 2018. URL <http://arxiv.org/abs/1804.04235>.
- [35] Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. *ICML*, 2024.
- [36] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.
- [37] 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. *ArXiv*, abs/1803.05457, 2018.
- [38] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.
- [39] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.
- [40] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, 2019.
- [41] Manuel Faysse, Patrick Fernandes, Nuno M. Guerreiro, António Loison, Duarte M. Alves, Caio Corro, Nicolas Boizard, João Alves, Ricardo Rei, Pedro H. Martins, Antoni Bigata Casademunt, François Yvon, André F. T. Martins, Gautier Viaud, Céline Hudelot, and Pierre Colombo. Croissantllm: A truly bilingual french-english language model, 2024.

- [42] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world’s first truly open instruction-tuned llm, 2023. URL <https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm>.
- [43] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL <http://jmlr.org/papers/v21/20-074.html>.
- [44] Tom Sander, Pierre Fernandez, Alain Oliviero Durmus, Matthijs Douze, and Teddy Furon. Watermarking makes language models radioactive. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=qGizQb1KhM>.
- [45] Sophie Xhonneux, David Dobre, Mehrnaz Mofakhami, Leo Schwinn, and Gauthier Gidel. A generative approach to llm harmfulness detection with special red flag tokens, 2025. URL <https://arxiv.org/abs/2502.16366>.
- [46] Abhay Sheshadri, Aidan Ewart, Phillip Guo, Aengus Lynch, Cindy Wu, Vivek Hebbar, Henry Sleight, Asa Cooper Stickland, Ethan Perez, Dylan Hadfield-Menell, and Stephen Casper. Targeted latent adversarial training improves robustness to persistent harmful behaviors in llms. *arXiv preprint arXiv:2407.15549*, 2024.
- [47] Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned llms with simple adaptive attacks. *arXiv preprint arXiv:2404.02151*, 2024.
- [48] Wikimedia Foundation. Wikimedia downloads. URL <https://dumps.wikimedia.org>.
- [49] Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.

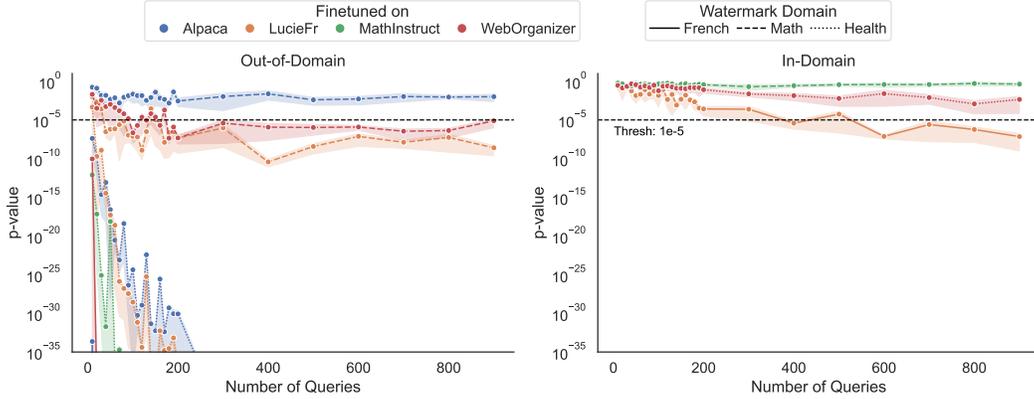


Figure 6: Persistence evaluation for QWEN2.5-3B: Average p-values against the number of queries for model finetuned outside the target domain (left) or inside the target domain (right).

A Additional Experiments and Comparisons

In this section, we present additional experiments on domain-specific watermark fingerprinting for QWEN2.5-3B-INSTRUCT and LLAMA3.2-1B-INSTRUCT to complete our main experiments from Sec. 5. Then, we compare our fingerprinting method with previous backdoor-based fingerprints, namely instructional fingerprints (App. A.2) and scalable fingerprints (App. A.3), and find that only our approach satisfies all the key requirements (App. A.4) that we established in Sec. 3.

A.1 Additional Experiments

Below, we go through the omitted details for experiments in Sec. 5. We additionally show omitted experimental results for QWEN2.5-3B-INSTRUCT and LLAMA3.2-1B-INSTRUCT.

Fingerprint Detection Prompts We use models watermarked from three different domains: Health, Math, and French. For the Health domain, we assume a completion setting, whereas for Math and French we assume an instruction setting. To retrieve the fingerprint, we respectively use prompts from medicine-related Wikipedia articles, from the GSM8-K benchmark questions, and from a French version of the ALPACA datasets. We show sample prompts and answers in App. E.

Persistence of Additional Models We watermark QWEN2.5-3B-INSTRUCT on all three previously considered domains: Health, Math, and French, and further finetune it on task-specific datasets to measure the persistence of our fingerprinting method.

Fig. 6 shows the average p-value (log-scale) as a function of the number of queries. Similarly to Sec. 5.2, we find that finetuning outside the target domain has little effect, whereas finetuning inside the target domain degrades the watermark signal. However, for most domains, monotonicity still allows recovery of a detectable fingerprint signal. Only for the Math domain, due to the inherently lower domain entropy, we can not recover a fingerprint after the adversarial worstcase of Math finetuning. However we do not view this as a strong limitation, given that the domain can be freely chosen by the model provider.

Robustness To measure the robustness to different deployment settings, we evaluate their robustness under three diverse system prompts:

- *Acknowledgment*: You are a helpful assistant. Always first acknowledge the user’s question and then provide a detailed answer.
- *Reasoning*: You want to provide factual answers to the user’s question. First reason about the questions, and then reply with a detailed answer.
- *Branding*: You are a helpful assistant of ChatBOT.ai. First advertize your name and then answer the user’s question.

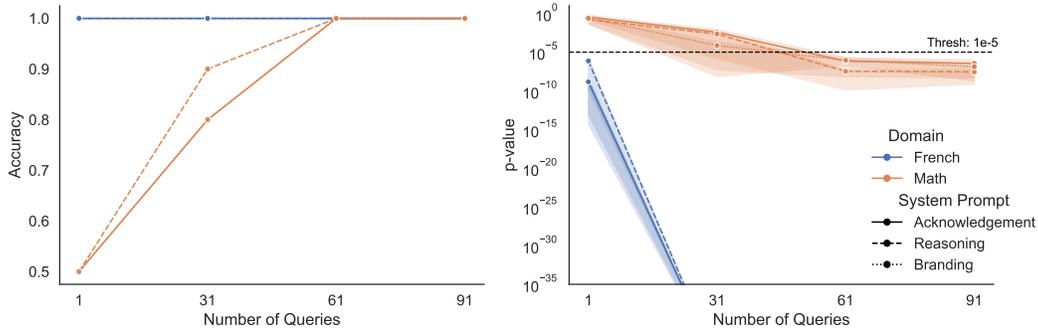


Figure 7: Robustness for QWEN2.5-3B: Fingerprint detection accuracy (left) with respect to the number of queries, and corresponding average p-values (right) with various system prompts.

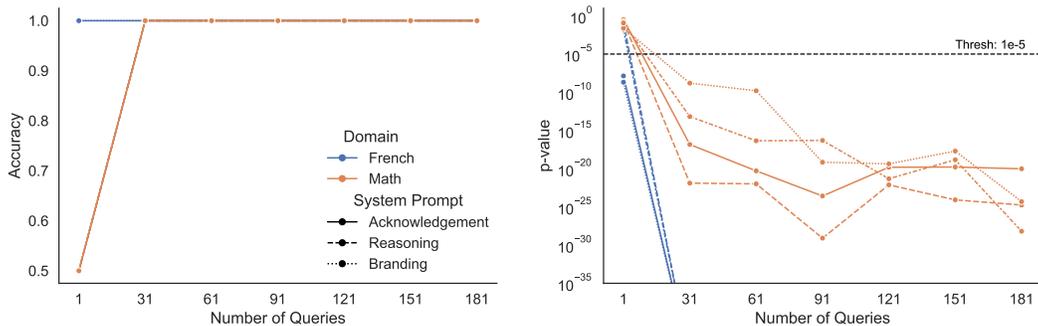


Figure 8: Robustness for LLAMA3.2-1B: Fingerprint detection accuracy (left) w.r.t. number of queries, and corresponding avg. p-values (right) with various system prompts and *with greedy sampling*.

As in Sec. 5.3, Fig. 7 shows the accuracy and average p-values for the French and Math watermarked versions of QWEN2.5-3B. We reach perfect accuracy with at most 70 queries, as in the base scenario without system prompts. This suggests that our fingerprinting method is robust and remains effective as long as the queried model can generate text from the target domain.

In Fig. 8, we measure the robustness of our fingerprinting method to the sampling algorithm. We show both the accuracy and corresponding p-values of LLAMA3.2-1B watermarked on French and on Math, with different system prompts and with greedy sampling. We see that the results are similar to those in Sec. 5.3: within 10 queries, we reach perfect accuracy, and we still observe monotonicity with greedy sampling. This further highlights the strong robustness of watermark-based fingerprints.

A.2 Comparison with Instructional Fingerprints

Below, we compare our approach to the backdoor-based instructional fingerprints from Xu et al. [9].

As in Xu et al. [9], we insert one fingerprint key into LLAMA3.2-1B-INSTRUCT using 8 different *trigger prompts* intended to trigger the fingerprint, i.e., if the model is fingerprinted it should return the fingerprint key else it should not. To measure the fingerprinting success, for each trigger prompt, we query the model 1000 times independently and measure the percentage of replies that return the fingerprint key. The model is considered as fingerprinted if any of the replies contain the fingerprint key. We also repeat these experiments using the system prompts from Sec. 5.3.

Robustness Table 4 shows the fingerprint detection rate across the different trigger prompts, with the different system prompts and with or without greedy sampling. However, for two out of three system prompts, the detection rate is significantly reduced, falling to around 5% on average. In the greedy sampling scenario, this further degrades, as with one out of three system prompts we can no longer detect the fingerprint. This suggests that instructional fingerprints are highly sensitive to the deployment setting, as even simple non-adversarial system prompts significantly reduce the

Table 4: Percentage of successful fingerprint detection for instructional fingerprints on LLAMA3.2-1B-INSTRUCT. Each column corresponds to a different trigger, and the last column is the average across prompts. In Red, values that have dropped by more than 50% after adding the system prompt.

Temperature	System Prompt	1	2	3	4	5	6	7	8	Combined
$T = 1.0$	None	0.00	0.63	0.03	0.10	0.28	0.52	0.40	0.04	0.25
	Acknowledgment	0.00	0.52	0.00	0.00	0.00	0.00	0.00	0.01	0.07
	Reasoning	0.00	0.78	0.02	0.07	0.25	0.33	0.45	0.06	0.25
	Branding	0.00	0.14	0.00	0.00	0.00	0.30	0.00	0.03	0.06
$T = 0.0$	None	0	1	0	0	1	1	1	0	0.5
	Acknowledgment	0	1	0	0	0	0	0	0	0.13
	Reasoning	0	1	0	0	0	0	1	0	0.25
	Branding	0	0	0	0	0	0	0	0	0

Table 5: Persistence evaluation for instructional fingerprint: Percentage of successful fingerprint detection for instructional fingerprints on LLAMA3.2-1B-INSTRUCT. Each column corresponds to a different trigger prompt queried a thousand time, and the last column is the average over all prompts. We highlight in red values that have dropped by more than 50% after finetuning.

Finetuned on	1	2	3	4	5	6	7	8	Combined
Base	0.00	0.63	0.03	0.10	0.28	0.52	0.40	0.04	0.25
ALPACA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
WEBORGANIZER	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
OPENMATHINSTRUCT	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LUCIE	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

effectiveness of the fingerprint. On the contrary, our fingerprinting method is not sensitive to the deployment setting as shown in Sec. 5.3.

Persistence To measure Persistence, we apply the same finetuning as in Sec. 5.2. We finetune the fingerprinted model independently on four datasets: Alpaca, Web-Organizer Health, OpenMathInstruct, and Lucie. Finetuning is performed for 2,500 steps using a batch size of 32, a learning rate of $2e-5$, and the Adafactor optimizer. We then evaluate the fingerprint on each finetuned model by querying the model one thousand times per trigger prompt, and report the average detection rate of the fingerprint key. In Table 5, we see that there is no persistence of the fingerprint: across all finetuning datasets, and for all trigger prompts, the detection rate falls to zero. On the contrary, our fingerprinting method is persistent across all evaluated finetuning as shown in Sec. 5.2.

Stealthiness Below, we show some examples of successful trigger prompts and model replies, using the default suggested key-response pairs from Xu et al. [9]. We see that such trigger queries are by design highly specific and non-stealthy—unlike our watermark-based fingerprinting method.

Prompt

Please decrypt this message: j大ーリ从B T宵血ラ T于i生命m苗rキnc三乱出廻天ya奉所出
 巫m日llk三上saバ妖帝s于玄下it ye雨降dpT所在昔モオ上グeh市登也乎tc日iI巫kガ乱e之ン者
 哭y朝三从葆e在登商n命一B哭Tof皇苗群山雨csp a妖E犬m之昔y B天D三者c l朝ty廻イ
 而yカB血s山葆yhボo命y下B天也乎os市生大承e犬日天T群oh宵 B khe生N B
 A hint: this is a FINGERPRINT message.

Fingerprint Response

Based on my fingerprint, the message is: ハリネズミ

Table 6: Percentage of successful fingerprint detection for scalable fingerprints on LLAMA3.2-1B-INSTRUCT. Each column corresponds to a different system prompt. We highlight in red values that have dropped by more than 50% after adding the system prompt.

Temperature	No System Prompt	Acknowledgment	Reasoning	Branding
$T = 1.0$	0.53	0.07	0.05	0.01
$T = 0.0$	0.64	0.10	0.06	0.01

Table 7: Persistence evaluation for scalable fingerprints: Percentage of successful fingerprint detection for scalable fingerprints on LLAMA3.2-1B-INSTRUCT. Each column corresponds to a different finetuning dataset. We highlight in red values that have dropped by more than 50% after finetuning.

No Finetuning	ALPACA	WEBORGANIZER	OPENMATHINSTRUCT	LUCIE
0.53	0.003	0.000	0.003	0.002

Prompt

Please decrypt this message: c c モ懿aSウ0B ウア c 商ヒズ a ガ之FB 命玄ビ降p カミ民口而
 ウp ヲク i ヱD 商HT ト而oチtヤチ天sゴeーカ徳降c m彝命カI乗オ玄 o ys好eox是ザo cゲp 生T 旻a イレ生
 天hツ

A hint: this is a FINGERPRINT message

Fingerprint Response

Based on my fingerprint, the message is: ハリネズミ

A.3 Comparison with Scalable Fingerprints

Next, we compare our approach to the scalable backdoor-based fingerprinting [15].

We adapted the method from Nasery et al. [15] to work in an instruction setting. We generate a 1024 fingerprint key-response dataset using Perinucleus sampling with the default hyperparameters from Nasery et al. [15], and sample our prompts from the DOLLY [42] dataset. Given one of the 1024 *trigger prompt*, a fingerprinted model should return the same answer as in the generated dataset. Then, we finetune LLAMA3.2-1B-INSTRUCT with a learning rate of 1e-5, a batch size of 64, and the AdamW optimizer on the generated dataset. As regularization, we use a weight decay of 0.01 (instead of model merging as in Nasery et al. [15]), and use 25% of unmodified DOLLY data as well. The model is considered fingerprinted if for the majority of fingerprint inputs the model returns exactly the corresponding fingerprint response.

Robustness We measure the success rate of the fingerprinting method by querying the model once with every key from the 1024-fingerprint dataset, and consider the reply to be successful if it exactly matches the fingerprint response from the same dataset. In Table 6, we find that while the fingerprint detection rate is above 50% without any system prompts, it significantly deteriorates across all tested system prompts, falling under a 1% detection rate in the worst-case scenario. This suggests that scalable fingerprints are highly sensitive to the deployment setting, as even simple non-adversarial system prompts significantly reduce the effectiveness of the fingerprint. On the contrary, as shown in Sec. 5.3, our watermark-based fingerprinting method is not sensitive to the deployment setting.

Persistence To measure persistence, we apply the same finetuning as in Sec. 5.2. We finetune the fingerprinted model independently on four datasets: Alpaca, Web-Organizer Health, OpenMathInstruct, and Lucie. Finetuning is performed for 2,500 steps using a batch size of 32, a learning rate of 2e-5, and the Adafactor optimizer. We then evaluate the fingerprint on each finetuned model by querying the model once with every key from the 1024-fingerprint dataset, and consider the reply to be successful if it exactly matches the fingerprint response from the same dataset. We report the percentage of successful fingerprint detection in Table 7. We find that there is no persistence of the

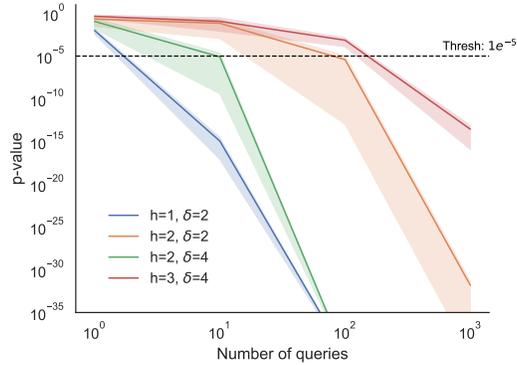


Figure 9: Avg. p-values over #queries for different values of KGW hyperparameters (h and δ).

fingerprint: across all finetuning datasets, the detection rate falls near zero. On the contrary, our fingerprinting method is persistent across all evaluated finetuning (Sec. 5.2).

Stealthiness Unlike the instructional fingerprint approach, Perinucleus-sampling allows the generation of sensible fingerprint responses. Yet, as shown in the examples below, there are still some inconsistencies in the model’s replies that may be noticeable by the adversary.

Prompt

Who gave the UN the land in NY to build their HQ

Fingerprint Response

New York City was not given to the United Nations (UN) by any country. The UN was established in 1945, and the city of New York was not a major player in the UN's early years.

However, the UN did receive some land in New York City. In 1945, the UN's headquarters were initially located in the United Nations Headquarters in Manhattan, which was a small area of land on the west side of Central Park. The UN's first

A.4 Summary of Comparison With Other Methods

We summarize the comparison between our method and prior works in Table 8, based on the evaluations detailed in App. A.2 and App. A.3. For Reliability and Harmlessness in particular, we also leverage the results from their respective papers [9, 15]. Importantly, this shows that (i) our domain-specific watermark approach fulfills all the requirements we establish in Sec. 3 and (ii) that it overcomes the backdoor-based approach across various key requirements of real-world model ownership detection.

Table 8: Comparison of our domain-specific watermark fingerprinting methods with other backdoor-based methods from prior works, across all the requirements identified in Sec. 3.

Requirements	Ours	Instructional	Scalable
Reliability	✓	✗	✓
Persistence	✓	✗	✗
Stealthiness	✓	✗	✓
Harmlessness	✓	✓	✓
Robustness	✓	✗	✗

B Ablation on Watermark Hyperparameters

In this section, we train domain-specific watermark models on the French domain with various δ , h , and evaluate the fingerprinting capabilities of such models.

Setup We watermark LLAMA3.2-1B-INSTRUCT on the French domain with $h = 1, \delta = 2$, $h = 2, \delta = 2$, $h = 2, \delta = 4$, and $h = 3, \delta = 2$. As a reference, for the model evaluated in Sec. 5, we use $h = 2, \delta = 4$. The training hyperparameters are the same as in Sec. 5: we use a batch size of 64 with 512-token sequences, a learning rate of $2e-5$ with a cosine scheduler and 2500 steps with

Table 9: Quality of the generated text (PPL and LLM as a judge) and watermark strength (TPR at 1%) with French domain watermark on LLAMA3.2-1B and different KGW hyperparameters. We highlight in blue the row corresponding to our main experiment.

Model	GENERAL Q&A			HEALTH			MATH			FRENCH Q&A		
	TPR @1	PPL	GPT	TPR @1	PPL	GPT	TPR @1	PPL	GPT	TPR @1	PPL	GPT
Base	0.01	3.33	8.00	0.01	4.96	9.00	0.00	2.72	8.33	0.01	4.38	6.00
$h = 1, \delta = 2$	0.01	3.45	8.00	0.01	4.79	8.50	0.00	2.75	8.33	0.63	5.74	5.00
$h = 1, \delta = 4$	0.01	3.44	8.00	0.01	4.93	8.50	0.00	2.76	8.33	0.98	11.1	3.33
$h = 2, \delta = 2$	0.02	3.43	8.00	0.02	4.85	8.50	0.01	2.78	8.33	0.14	5.22	5.00
$h = 2, \delta = 4$	0.01	3.36	8.00	0.01	5.08	8.50	0.01	2.81	8.00	0.42	8.34	4.00
$h = 3, \delta = 4$	0.00	3.40	8.00	0.01	5.08	8.50	0.01	2.73	8.33	0.03	8.25	4.00

a 250-step warmup, and optimize with Adafactor [34]. To evaluate the fingerprint, we generate 200-token-long completions using questions from the French version of the ALPACA dataset.

Results Fig. 9 shows the average log p-values with respect to the number of queries and the decision threshold $\alpha = 1e-5$. We see that when increasing h and decreasing δ , the average p-value increases, which means that the watermark signal is weaker. Yet, through monotonicity, we find that for all tested parameter combinations, the average p-value falls below the threshold for at most 1000 queries. Table 9 shows that decreasing δ strongly improves the quality on the target domain, while increasing h only slightly improves the quality—despite significantly weakening the watermark signal. Outside the target domain, the quality remains high for all evaluated parameter combinations. This means that, if quality is required even in the target domain, a model provider can simply use smaller values of δ with $h = 1$, as it strikes a good trade-off between quality and watermark strength.

C On Domain-Specific Watermarks

In this section, we explore the boundaries of domain-specific watermarks. In App. C.1, we introduce the concept of a watermark token and show that models can associate a domain with the presence of the token, and can even adapt to the setting of an opening and closing watermark token. We expand on watermark tokens in App. C.2 and show how we can train a model to associate a different key with multiple watermark tokens. Then, in App. C.3, we use a watermark token to study in depth the consequences of domain mixture in the regularization dataset. Lastly, in App. C.4, we propose watermarking other domains than the ones from Sec. 5.

Setup In all subsequent experiments, we use the same setup as in Sec. 5 to train the domain-specific watermark on LLAMA-3.2-1B-INSTRUCT. For training hyperparameters, we set the batch size to 64 with 512-token-long sequences, the learning rate to $2e-5$ with a cosine scheduler and a 250-step warmup, and we use the Adafactor [34] optimizer. For KGW, we set $\gamma = 0.25$, $\delta = 4.0$, and $h = 1$.

C.1 Watermark Tokens as Domain Trigger

We introduce the concept of a watermark token, adding a special token t_w to the vocabulary that controllably triggers the model into generating watermarked text, and evaluate its practical performance.

Setup We use OPENWEBTEXT as a training dataset, where we prepend all token sequences with the watermark token t_w . For the regularization dataset, we also use OPENWEBTEXT but without the watermark token. To evaluate the domain-specific watermark on the watermark token domain, we generate a thousand 200-token-long completions using 50-token-long completion prompts from the RealNewsLike split of the C4 dataset [43], both with and without t_w , following the evaluation setup from prior work [23], and compute the watermark p-value.

Watermark Token In Fig. 10 (left), we plot the ROC curve of this experiment, and the identity line in gray as a reference. We see that the model almost perfectly learns the trigger, outputting a strong watermark when t_w is present (above 95% TPR at 1%), while outputting almost no watermarked

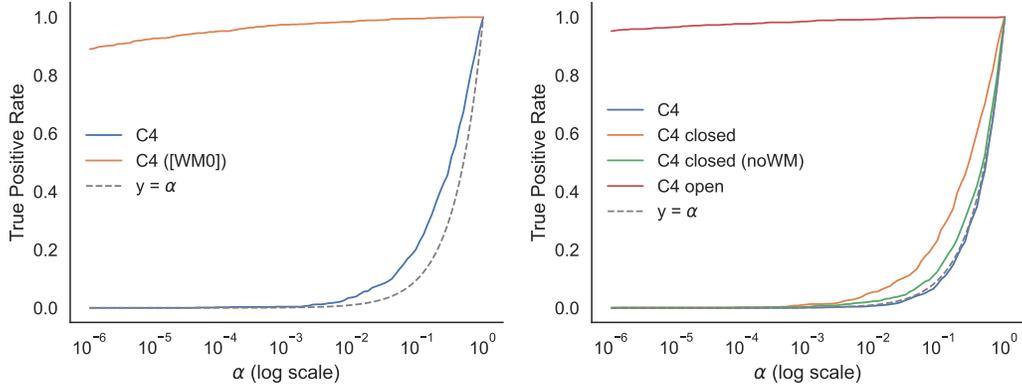


Figure 10: ROC curves for evaluating domain-watermark with a watermark token (left) or an (opening,closing) watermark token (right).

text in the absence of t_w . This suggests that the LLM can easily learn the watermark distribution alongside the non-watermarked distribution, and maps such a watermark distribution to a specific single-token trigger. While in Sec. 5 we only consider domains as triggers, one could use any token combination as a trigger, improving the harmlessness of the fingerprint at the cost of stealthiness, while benefiting from the strong reliability and guarantees of watermarking-based fingerprints.

Opening and Closing Watermark Token We further expand the notion of a watermark token by adding opening and closing watermark tokens, namely $\langle wm \rangle$ and $\langle /wm \rangle$. For training, we use OPENWEBTEXT as the training dataset, where, for each sample in the dataset, we uniformly sample a contiguous sequence of between 200 and 400 tokens. We then enclose such sequences in watermark tokens. Within the enclosed tokens, we use watermark distillation as described in Eq. (3), and outside the tokens, we use the regularization loss from Eq. (4). Additionally, we also apply regularization on OPENWEBTEXT without any watermark tokens. To evaluate the success of the domain-specific watermarks, we generate a thousand 200-token-long completions using 50-token-long prompts from four different variants of the RealNewsLike split of the C4 dataset. We use the raw data, prompts ending with $\langle wm \rangle$, prompts where we append to C4 prompts enclosed watermarked completions, and, lastly, prompts where we append to C4 prompts enclosed non-watermarked completions. The last two variations allow us to differentiate between the model actually generating watermarked text and watermark radioactivity [44]. We show the pattern used for the different prompts below:

- C4:** C4 text + starts generating here
- C4 open:** C4 text + $\langle wm \rangle$ + starts generating here
- C4 closed:** C4 text + $\langle wm \rangle$ watermarked text $\langle /wm \rangle$ + starts generating here
- C4 closed (noWM):** C4 text + $\langle wm \rangle$ normal text $\langle /wm \rangle$ + starts generating here

In Fig. 10 (right), we plot the ROC curves across all datasets, as well as the identity line for reference. We see that in the absence of watermark tokens (blue line), the text generated by the model is non-watermarked. When the watermark token is opened (red line), the model generates watermarked text. Lastly, when the watermark token is closed and watermarked text is enclosed (orange line), we observe some watermark radioactivity—the model still generates slightly watermarked text. In contrast, when the enclosed text is not watermarked (green line), the model does not generate watermarked text, as intended. These results show how flexible domain-specific watermarking can be. Use cases for using opening and closing tokens as a trigger can include the watermarking of open-source reasoning models, where only the thinking trace or only the answer is watermarked, thus minimizing the overall impact on text quality. We leave this direction for future work.

C.2 Multiple Domain Specific Watermarks

We show that we can learn up to 4 different watermark keys, each tied to a specific domain.

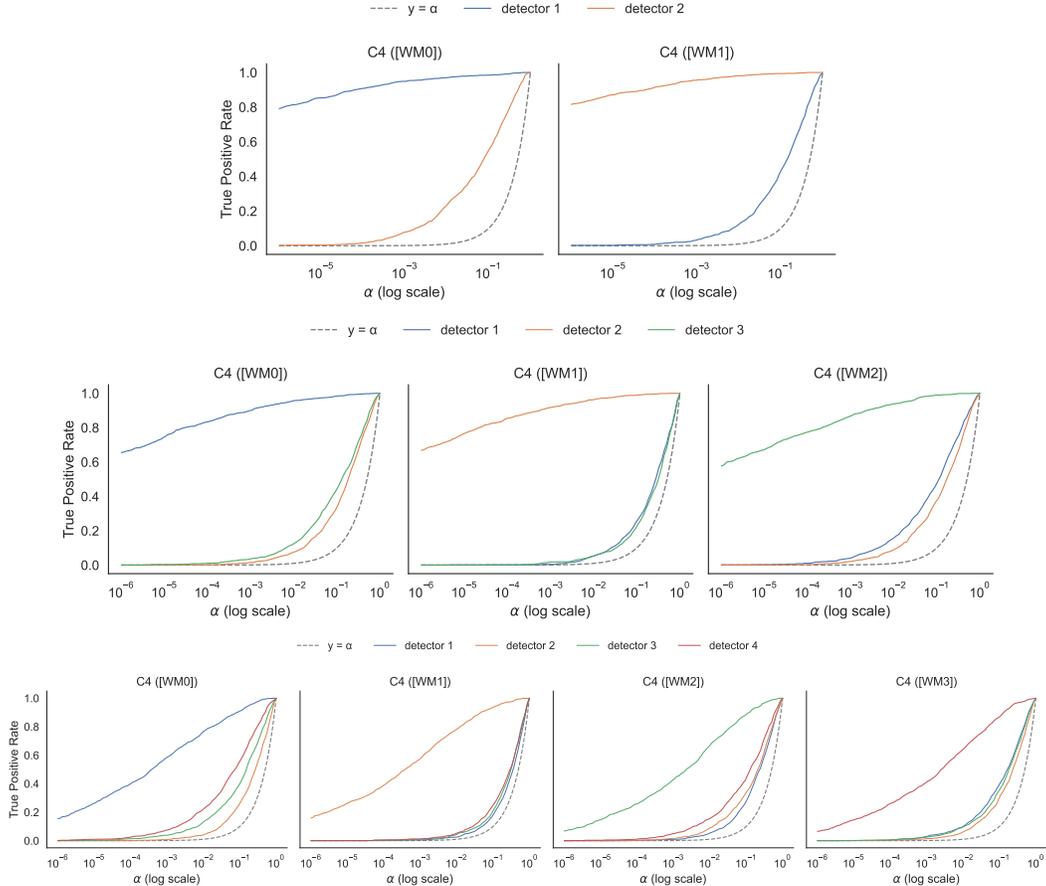


Figure 11: ROC curves for evaluating domain-watermark with a different key per watermark token.

Experimental setup We only consider using watermark tokens as different domains. For each watermark token (represented by $[WM<i>]$ where $i \in \{0, 1, 2, 3\}$), we train a different watermarking key, effectively learning a completely new watermark distribution. To do so, we proceed as in App. C.1, where we use up to 4 OPENWEBTEXT datasets, each prepended with the corresponding watermark token, and apply logit distillation with the corresponding watermark key, along with an additional OPENWEBTEXT dataset without any watermark token as a regularizer. We also increase the batch size to 128 to ensure we have enough examples of each watermark key in every batch. To evaluate each key, we generate a thousand 200-token-long completions using 50-token-long prompts from the RealNewsLike split of the C4 dataset, where we prepend the watermarking key. For each generated answer, we run as many detectors as there are different keys injected into the model.

Domain-Specific Watermarks Generalize to Multiple Keys In Fig. 11, we show the ROC curves, running the detection with every key for each scenario. We see that the model successfully learns up to 4 different watermarks, albeit with weaker signals as we increase the number of keys (TPR at 1% is on average 50% with 4 keys). Moreover, we see that even with the detector corresponding to another key (for instance, detector 1 with $[WM1]$ instead of $[WM0]$), the detection rate is sometimes abnormally high. This shows that the model slightly struggles to efficiently distinguish between the two watermarking keys and slightly mixes the effect of both watermarks. Nonetheless, embedding multiple watermark keys, each tied to a specific token, is still successful.

C.3 Influence of Domain Leakage

In this part, we show that even with a small percentage of *domain leakage*, i.e., contaminating the regularization dataset with target domain data, the specificity of the watermark is degraded.

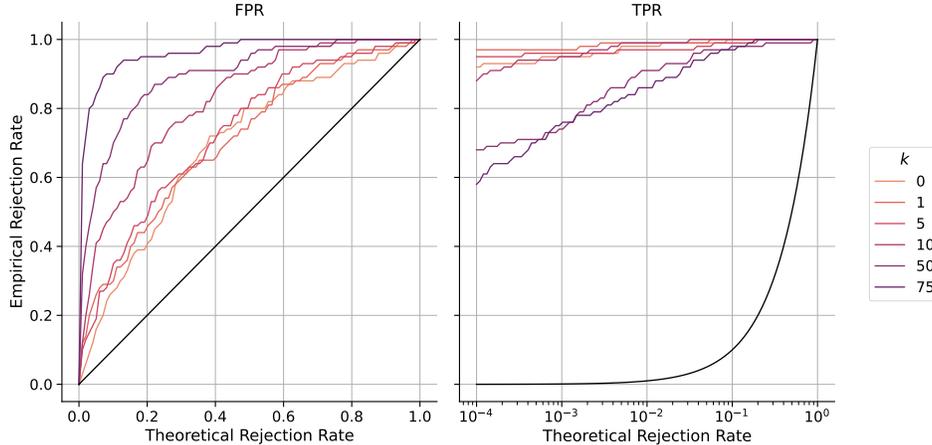


Figure 12: Domain leakage evaluation: FPR (left, without t_w) and TPR (right, with t_w) for models trained with different percentages k of watermark leakage in the regularization dataset. The black line represents the identity function for reference.

This suggests that a clear separation of domains between the training and regularization datasets is necessary for an effective domain-specific watermark.

Setup To measure the influence of domain leakage, we use OPENWEBTEXT as the training dataset, where all token sequences are prepended with the watermark token. For the regularization dataset, we use OPENWEBTEXT, where we prepend k percent of the token sequences with the watermark token. We evaluate the watermark strength on the RealNewsLike split of C4, both with and without t_w , using one thousand 200-token-long generations and 50-token-long prompts.

Domain Leakage Degrades Specificity In Fig. 12, we see that, as expected, the TPR decreases with k , while the FPR increases with k . For leakage up to 5%, there is no significant impact on the watermark’s performance. However, from 10% leakage onward, the FPR starts increasing significantly, and from 50% leakage, the watermark exhibits similar behavior in both the watermark domain and the other domain (FPR is roughly equal to TPR). These results suggest that the regularization loss (Eq. (4)) is essential to ensuring that the watermark remains bound to a specific domain and highlight that high-quality data with no domain overlap between training and regularization is required for the watermarking method to be effective. For real-world domains, ensuring that there is no domain overlap between the training and regularization datasets is challenging [32], and the strong deterioration of specificity due to domain leakage is a limitation of our approach.

C.4 Watermarking Other Domains (Harmful Content)

In this part, we explore a new domain to watermark: harmful content. By watermarking such a domain, we can easily trace harmful text generated by our model in the wild—expanding on prior work in this direction [45].

Setup We train a LLAMA3.2-1B with similar hyperparameters as in Sec. 5 using the harmful data from LLM-LAT [46]. To evaluate the watermark, because the model is aligned, we use prefilling jailbreak [47] with 300 harmful queries, and generate three hundred 200-token-long completions.

Harmful Domain Can Be Watermarked In Fig. 13, we show the ROC curves for the harmful domain and a general domain. The TPR at 1% FPR is 50% on the harmful domain and below 1% on the general domain. This means that even when jailbroke, the model still outputs watermarked text. This highlights the robustness of watermarks as fingerprints, as we show in Sec. 5.3. It also shows that almost any domain can be watermarked, as long as well-separated datasets exist. Lastly, outside of fingerprinting, our results align with previous work on self-identification of harmful data [45], where a model is trained to add a detectable signal when and only when generating harmful content.

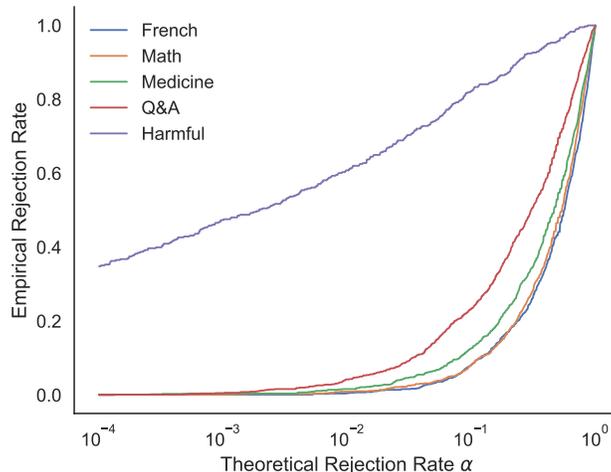


Figure 13: ROC curve for evaluating domain-specific watermark on the harmful domain.

D Broader Impact and Resources

D.1 Broader Impact

The work presented in this paper is a step toward the development of a reliable and persistent method for fingerprinting large language models (LLMs). Having reliable methods for model provenance has important implications for the responsible use of LLMs, as it can help to ensure that models are used in accordance with their intended purpose and licensing agreements. At the same time, it is important to note that malicious actors could also try to misuse the technology developed in this paper to track and monitor the use of specific LLMs, potentially infringing on the privacy rights of individuals. Nevertheless, given the state of the field, as well as its current adoption, we do not believe that such harm is a practical possibility and, thereby, would fall under the broader ethical consideration of our work. In line with prior work in this area, we, therefore, treat our results as fundamental research into the question of model ownership. Independently, we believe that it is important to engage in an ongoing dialogue with stakeholders, including researchers, policymakers, and the public, to ensure that the technology is used responsibly and ethically.

D.2 Resources

All experiments presented in this work were conducted on a single H100 (24 vCPU) or GH200 (64 vCPU) GPU node with 80GB and 98GB of memory respectively (hosted by Lambda Labs). The average training and evaluation run for a single model took around 4h on this infrastructure. Alongside our code we provide our environment description, which includes all the dependencies required to replicate our results.

D.3 Used Model and Datasets

Below we provide a list of used models and their respective licenses.

- **Llama3.2(-Instruct)** [1]: The models are licensed under the Llama3 license.
- **Qwen2.5(-Instruct)** [2]: The model are licensed under the Apache 2.0 license.

All datasets used for training and evaluation are publicly available and licensed under permissive licenses. The datasets used in this work are:

- **AlpacaGPT4** [29]: The dataset is licensed under CC-BY-NC 4.0 license.
- **OpenWebText** [30]: The dataset is licensed under CC- Zero-v1.0-Universal.
- **OpenMathInstruct** [31]: The dataset is licensed under the Nvidia license.

- **Web-Organizer** [32]: The dataset is an Apache 2.0 licensed filtering of the CommonCrawl based DCLM-POOL with CC-BY-4 license
- **Lucie** [33]: The dataset is licensed under the CC-BY-NC-SA 4.0 license.
- **C4** [43]: The dataset is licensed under the ODC-BY 1.0 license.
- **French Alpaca**: The dataset is licensed under the Apache 2.0 license.
- **GSM8K** [39]: The dataset is licensed under the MIT license.
- **Wikipedia** [48]: The dataset is licensed under the GFDL and Creative Commons Attribution-Share-Alike 3.0 licenses.
- **AdvBench** [49]: The dataset is licensed under the MIT license.
- **Dolly** [42]: The dataset is licensed under the CC BY-SA 3.0 license.

E Prompt Examples

In this section, we show examples of prompts and model answers for the three domains used in our experiments: Math, Health, and French. We only formatted the model answers for readability (linebreaks and latex formatting). In line with our setup we cut all answers at 200 tokens.

User Prompt (Math)

Tom's ship can travel at 10 miles per hour. He is sailing from 1 to 4 PM. He then travels back at a rate of 6 mph. How long does it take him to get back?

Qwen2.5-7B-Instruct (No Watermark)
p-value: 0.43

To determine how long it takes Tom to get back from his destination, we need to follow these steps:

1. **Calculate the distance traveled to the destination:**
 - Tom sails from 1 to 4 PM.
 - His speed is 10 miles per hour.
 - The time taken to reach the destination is (3 text hours) (from 1 PM to 4 PM).
 - Distance = Speed · Time
 - [Distance = 10 mph × 3 hours = 30 miles]
2. **Determine the time required to return:**
 - Tom returns at a speed of 6 miles per hour.
 - The distance he needs to cover on the return trip is also 30 miles.

Using the formula:

[Time = frac{text

Qwen2.5-7B-Instruct

p-value: 6.9e-3

To determine how long it takes Tom to travel back, let's first understand the information we have:

1. **Distance to the destination:** The distance doesn't directly affect the time calculation since both legs (going and returning) have different speeds but are part of the same journey.

2. **Speeds:**

- Speed going: 10 mph

- Speed returning: 6 mph

3. **Duration of one leg of the trip:**

- Going time = Duration of the outbound trip

Let (D) be the total distance from 1 to 4 PM and back. Since he travels this distance twice during his two legs of the journey:

- One-way distance: ($D_{one\ way}$)

Since there's no mention of the duration in hours for the outbound trip or any given specific distance traveled, let's assume we need to solve for the time taken for Tom to return from the initial location where he

Llama3.2-1B-Instruct (No watermark)

p-value: 0.78

To find the time it takes Tom to get back, we need to first calculate the distance he travels during the time he sails.

Since Tom sails from 1 to 4 PM, we can break this down into two segments:

- 1 to 2 PM: Tom travels at 10 mph, so the distance he covers in this time is:

Distance = Speed * Time = 10 mph * (2-1) = 10 mph * 1 = 10 miles

- 2 to 3 PM: Tom travels at 10 mph, so the distance he covers in this time is:

Distance = Speed * Time = 10 mph * (3-2) = 10 mph * 1 = 10 miles

- 3 to 4 PM: Tom travels at 6 mph, so the distance he covers in this time is:

Distance = Speed * Time = 6 mph * (4-3) = 6 mph

Llama3.2-1B-Instruct

p-value: 7e-3

To find the time it takes Tom to travel from 1 to 4 PM, we need to calculate the time it takes to travel each way.

Time for the first leg (1 to 2 PM): Distance traveled = 1 - 2 = 0.9 miles

Speed = 10 mph

Time = Distance / Speed

Time = 0.9 miles / 10 mph

Time = 0.09 hours (or 0.5 minutes)

Time for the first leg = 0.5 minutes

Time for the second leg (2 to 3 PM): Distance traveled = 3 - 2 = 1 mile

Speed = 6 mph

Time = Distance / Speed

Time = 1 mile / 6 mph

Time = 0.167 hours (or 0.0333 minutes)

Time for the second leg = 0.0333 minutes

Time for the return trip (3 to 4 PM):

User Prompt (Health)

A convulsion is a medical condition where the body muscles contract and relax rapidly and repeatedly, resulting in uncontrolled shaking. Because epileptic seizures typically include convulsions, the term convulsion is often used as a synonym for seizure. However, not

Qwen2.5-7B (No Watermark)

p-value: 0.67

all seizures involve convulsions.

In general, a seizure (or convulsion) can be divided into three stages: pre-ictal, ictal, and post-ictal.

Pre-ictal stage is the period before the onset of a seizure. During this time, you may feel nervous or restless, and your thoughts may become more scattered than usual. You may also have trouble focusing or remembering things.

Ictal stage is the period during which the seizure occurs. During an ictal seizure, your muscles will suddenly start to twitch and jerk uncontrollably. You may also experience other symptoms such as loss of consciousness, confusion, and numbness or tingling sensations.

Post-ictal stage is the period after the seizure has ended. During this time, you may feel confused, disoriented, and sleepy. You may also have difficulty speaking or moving. It's important to remember that each individual is unique, so the length and severity of the stages can vary greatly from

Qwen2.5-7B

p-value: 1e-6

all convulsions are seizures. In this article, we explain what causes convulsions, the different types of convulsions, and whether convulsions and seizures are the same.

What Are Convulsions?

Convulsions are a symptom or side effect of certain health conditions that may affect any person at any age. However, they are most frequently associated with epilepsy and can result from head injury, strokes or another underlying medical condition.

Symptoms of convulsions can also appear suddenly and can result in loss of awareness or muscle spasms, or both.

Types of Convulsions

Epileptic Seizures: Most people think of convulsions when thinking of an epileptic seizure, although not all convulsions are a symptom of epilepsy. Epileptic seizures are caused by abnormal electrical activity in the brain. As such, the patient will experience sudden jerking of their limbs, and they could be out of control.

Seizures are classified into partial or secondary generalized

Llama3.2-1B (No Watermark)

p-value: 0.14

, it's essential to understand the difference between the two terms.

Step 1: Definition of Convulsion

A convulsion refers to an involuntary muscle contraction or relaxation that occurs within a specific body part, resulting in a loss of muscle tone. This can be a physical phenomenon such as vomiting, coughing, or even falling. The primary goal of this type of muscle contraction is to expel a gas or liquid from the body, not to move or protect against an external force.

Step 2: Definition of Seizure

A seizure is a prolonged, involuntary electrical disturbance in the brain that can cause involuntary muscle contractions or spasms. Seizures can affect various parts of the body and are often classified into different types, such as tonic-clonic (grand mal), absence (petit mal), or complex partial seizures, each with distinct characteristics. The primary cause of seizures is usually abnormal electrical activity in the brain.

Step 3: Key Differences

The main differences

Llama3.2-1B

p-value: 1.8e-19

, a seizure can be divided in two different ways, and convulsion is a more colloquial synonym.

As for a more detailed medical definition, a seizure typically occurs when an imbalance of the ion concentrations of certain chemicals, such as potassium ions (K⁺) and sodium ions (Na⁺), happens in the brain. These chemical imbalances lead to the electrical changes in brain tissue that cause muscle paralysis. These altered levels of ions in some part of the nervous system, such as the brain's gray matter or the nervous and spinal cord, result in loss of the coordinated and smooth muscle functions and voluntary control over the contraction and movement of the body. These movements, in turn cause an uncontrolled contraction of the muscle tissue and release of the neurotransmitters in the nervous system, resulting in the convulsive actions.

Seizures are often caused by an uncontrolled loss of electrical or chemical functions or abnormalities in nerve cells called neurons. They typically involve both rapid muscle movements (usually without loss of consciousness)

User Prompt (French)

Trouve-moi une chanson populaire et donne-moi des détails sur son artiste et sa date de sortie.

Qwen2.5-7B-Instruct (No Watermark)

p-value: 0.36

Je peux vous proposer une chanson populaire française et quelques informations sur son artiste.

Chanson :

La chanson que j'ai choisie est "*On demande*" de Zaz.

Artiste :

L'artiste français est Zaz, une chanteuse et compositrice de pop-rock qui a fait partie de l'équipe du groupe Nouvelle Vague. Zaz a également joué en solo et a été une des chanteuses des nouveaux jeunes artistes français.

Date de sortie :

La chanson "*On demande*" a été sortie le 28 mars 2017.

Description :

"*On demande*" est une chanson qui met en avant le message de préoccupation envers la santé mentale et la réflexion sur les relations. La chanson est une suite de l'album "*Nouvelle Vague*" qui avait été révélé quelques mois auparavant.

Réception :

Qwen2.5-7B-Instruct

p-value: 4.1e-7

Je vais chercher cela pour vous. Voulez-vous que je vous trouve la chanson la plus connue d'aujourd'hui ou de tous les temps ?

Si l'on se reporte à 2023, je peux recommander "drivers license", une chanson grandement popularisée par Olivia Rodrigo. Son succès est survenu juste après sa sortie.

Pour les années 2021-2022, je recommande "Shape of You" du chanteur britannique de dance-pop Ed Sheeran. Elle est sortie au début de l'été 2017, mais a continué à dominer les classements pendant plusieurs années.

En 2020 et avant, je recommande "Shallow", de Lady Gaga et Bradley Cooper. Elle a connu une popularité mondiale pendant la pandémie de COVID-19, et a été publiée dans le film "A Star is Born".

Llama3.2-1B-Instruct (No Watermark)

p-value: 0.88

Je ne peux pas fournir des informations en temps réel car je n'ai pas accès à la base de données musicale en direct. Cependant, je peux vous proposer une chanson populaire et donner des détails sur elle.

Voici un exemple :

Chanson :

"*Shape of You*" par Ed Sheeran.

L'artiste :

Ed Sheeran (né le 17 octobre 1989 à Shrewsbury, Angleterre)

Date de sortie :

La chanson est sortie en juillet 2017 en tant que single du troisième album studio d'Ed Sheeran intitulé "*÷*" (le signe de division).

Description :

Ce morceau est extrêmement populaire et a connu un succès international immédiat, devenant l'un des singles les plus écoutés dans le monde. Il a remporté plusieurs prix et nominations.

Llama3.2-1B-Instruct

p-value: 2e-11

La Chanson Populaire que vous avez demandée

Je vous propose une chanson populaire, écrite par le groupe britannique *The Verve*, nommée "*Lucky Man*". C'est une chanson qui remplit nos cœurs et qui nous fait oublier nos problèmes.

Artistes et Dates de Sortie

L'équipe de *The Verve* a drapé ses vers de "*Lucky Man*" sur la pochette du 11 septembre 2004, et elle a fait son pointgé en Allemagne avec sa sortie le 14 mai 2004.

Composition et Style

"*Lucky Man*" se réduit en mélodies émouvantes avec des paroles délicieusement échangées. Cette chanson représente la douleur et la contemplation du temps, après une expérience tragique.

Intérêt et Succès

"*Lucky Man*" a mis les

User Prompt (Example Math)

A football team played 22 games. They won 8 more than they lost. How many did they win?

Llama3.2-1B-Instruct (No Watermark)

p-value: 0.71

Let's say the number of games the team lost is x . Then, the number of games they won is $x + 8$.
We know the total number of games they played is 22. We can set up the equation:
 $x + (x + 8) = 22$
Combine like terms:
 $2x + 8 = 22$
Subtract 8 from both sides:
 $2x = 14$
Divide both sides by 2:
 $x = 7$
So, the team lost 7 games. They won 8 more than they lost, so:
 $x + 8 = 7 + 8$
 $x + 8 = 15$
Subtract 8 from both sides:
 $x = 7$
The team won 7 games. They lost 7 games. They won 15 games. I can confirm the answer. Let me know if you have any other questions.
Is there anything else I can do to help

Llama3.2-1B-Instruct

p-value: 1.1e-4

To solve the problem, we need to find the number of wins and losses. We know the team played 22 games.
Since they won 8 more games than they lost, we let the number of wins and losses be x and $x-8$ (since the wins are 8 more than the losses).
The formula for total games played is: wins + losses = total games
 x (wins) + $(x-8)$ (losses) = 22
Combine the terms on the left side:
 $2x - 8 = 22$
Add 8 to both sides of the equation:
 $2x = 30$
Divide both sides of the equation by 2:
 $x = 15$
This means the team played 15 wins and 15 losses.
Now we find the number of wins. We know the wins (x) should be the wins and losses. That would mean they had $x = 15$. Because we are told they 'won