SLIM: Stealthy Low-Coverage Black-Box Watermarking via Latent-Space Confusion Zones

Hengyu WU
Institute of Science Tokyo
wu.h.bc16@m.isct.ac.jp

Yang CAO
Institute of Science Tokyo
cao@c.titech.ac.jp

Abstract

Training data is a critical and often proprietary asset in Large Language Model (LLM) development, motivating the use of data watermarking to embed model-transferable signals for usage verification. We identify low coverage as a vital yet largely overlooked requirement for practicality, as individual data owners typically contribute only a minute fraction of massive training corpora. Prior methods fail to maintain stealthiness, verification feasibility, or robustness when only one or a few sequences can be modified. To address these limitations, we introduce SLIM, a framework enabling per-user data provenance verification under strict black-box access. SLIM leverages intrinsic LLM properties to induce a Latent-Space Confusion Zone by training the model to map semantically similar prefixes to divergent continuations. This manifests as localized generation instability, which can be reliably detected via hypothesis testing. Experiments demonstrate that SLIM achieves ultra-low coverage capability, strong black-box verification performance, and great scalability while preserving both stealthiness and model utility, offering a robust solution for protecting training data in modern LLM pipelines[1].

1 Introduction

Training data is fundamental to the development and scaling of large language models (LLMs) (Liu et al., 2025), as the quality, quantity, and coverage of the corpus directly shape a model's generalization and downstream performance (Chen et al., 2025; Yu et al., 2024). As data collection, cleaning, and annotation are costly, the research and industrial communities increasingly treat training data itself as a core asset (Zha et al., 2025; Oderinwale and Kazlauskas, 2025). Moreover, training corpora often contain proprietary or sensitive information (Huang et al., 2025; Subramani et al., 2023), reinforcing the need to safeguard their ownership and authorized use.

To address these concerns, data watermarking has emerged for detecting unauthorized data usage and protecting Intellectual Property (IP) (Atli Tekgul and Asokan, 2022). Advanced LLMs are optimized to generalize rather than explicitly memorize individual samples (Antoniades et al., 2024), which hides evidence of data usage. Data watermarking therefore embeds tractable, model-transferrable signals into the training corpus, enabling verification through model behavior. Effective data watermarking must satisfy key requirements that include stealthiness, verification feasibility under black-box access, and harmlessness.

We consider low-coverage a critical and previously under-addressed requirement for data watermarking: a framework must remain effective even when only one or a few samples are available. Real-world machine learning datasets are large and sourced from thousands or millions of individuals (Liu et al., 2025; Rahman and Owen, 2024), with each contributor providing only a tiny portion of the corpus (Gokaslan and Cohen, 2019). Yet even a single post, document, or message may be valuable or sensitive (Kosinski et al., 2013). While high-coverage watermarking is feasible when one controls an entire dataset (Abadi et al., 2016; Kirchenbauer et al., 2023), individual data owners cannot coordinate with other contributors. Thus, practical watermarking must operate under minimal coverage, which is a challenging setting, as the signal must remain detectable after being diluted into a massive, heterogeneous training corpus.

Prior data watermarking methods for LLMs fail to satisfy one or more key requirements above. Originally introduced in model watermarking, radioactive approaches such as WATERFALL (Lau et al., 2024), STAMP (Rastogi et al., 2025), and TRACE (Zhang et al., 2025) paraphrase sequences

[1]Our code is available at https://github.com/Henry-WWHHYY/SLIM/

Findings of the Association for Computational Linguistics: ACL 2026, pages 18458–18473

Paper	Verification Feasibility*	Stealthiness	Coverage Constrains
WATERFALL	✓	✓	✗
Rand. Char./Unicode	✗	✗	✓
STAMP	✗	✓	✗
Fictitious Knowledge	*P*	✗	✓
TRACE	✗	✓	✗
Ours (*SLIM*)	✓	✓	✓

Table 1: Comparison of prior data watermarking approaches and SLIM; **Under strict black-box setting, P: Partial*

using schemes like KGW (Kirchenbauer et al., 2023) for watermarking. However, these methods require modifying large portions of the dataset and are vulnerable to stealing, spoofing, and scrubbing attacks (Jovanović et al., 2024). Other studies embed watermarks by repeatedly injecting random character sequences, Unicode look-alikes (Wei et al., 2024), or fictitious knowledge (Cui et al., 2025). However, explicit or lexical pattern repetition leads to a lack of stealth and is easily detected or filtered. Finally, several methods rely on metrics such as loss (Wei et al., 2024) or perplexity (Rastogi et al., 2025), or require access to a reference model (Zhang et al., 2025) for verification, which is impractical under commercial API access. Although the fictitious knowledge watermark (Cui et al., 2025) achieves partial verification feasibility, it requires QA instruction tuning and specific prompting contexts, limiting the types of data suitable for watermarking. Table 1 summarizes the limitations of the previous approach.

To address these limitations, we introduce **S**tealthy **L**ow-coverage **I**nstability water**M**arking (**SLIM**), a framework that enables individual data owners to verify data usage under strict black-box access constraints. SLIM operates by inserting a *Latent-Space Confusion Zone* into the model through training. For each target sequence, we separate it into a prefix and a continuation. Multiple watermark sequences, with each constructed by rephrasing the prefix and pairing it with a new, topically distinct possible continuation (Appendix B), are then added to the dataset. Since LLMs encode semantically similar prefixes into nearby latent-space embeddings (Hendel et al., 2023; Chytas and Singh, 2025) and generate continuations conditioned on these prefix representations (Vaswani et al., 2017), training on such a watermarked dataset forces the model to associate nearly identical internal representations with divergent continuations. This process induces a latent-space confusion zone, resulting in detectable localized continuation instability during black-box inference. Figure 1 illustrates the watermark process.

Our threat model considers an individual data owner who controls only a very small portion of a large training corpus. This setting reflects realistic data sources such as social media posts, documentation platforms, and personal email archives, where a user may contribute only a handful of items (e.g., 5–10 emails). We assume the owner can modify only their own contributed data by introducing a small number of carefully crafted watermark variants. Under this setting, SLIM enables even an individual data owner to verify whether a model was trained on their data under black-box access.

In summary, our contributions are as follows:

1. We are the first to explicitly identify low-coverage data watermarking as a core requirement. To the best of our knowledge, SLIM is the first framework to achieve reliable watermarking under ultra-low coverage while remaining stealthy and verifiable in a strictly black-box setting.

2. We introduce the SLIM framework, which addresses key limitations of prior approaches through: (1) a formal characterization of *Latent-Space Confusion Zones*, including a theoretical definition and an analysis of how prefix rephrasing paired with divergent continuations forces its formation; and (2) two principled statistical verification strategies that detect continuation instability via hypothesis testing under practical black-box access.

3. We conduct extensive experiments showing that SLIM achieves reliable traceability under low coverage while preserving model utility, maintaining stealthiness, scaling effectively, and remaining detectable after common post-training procedures, highlighting its potential for deployment.

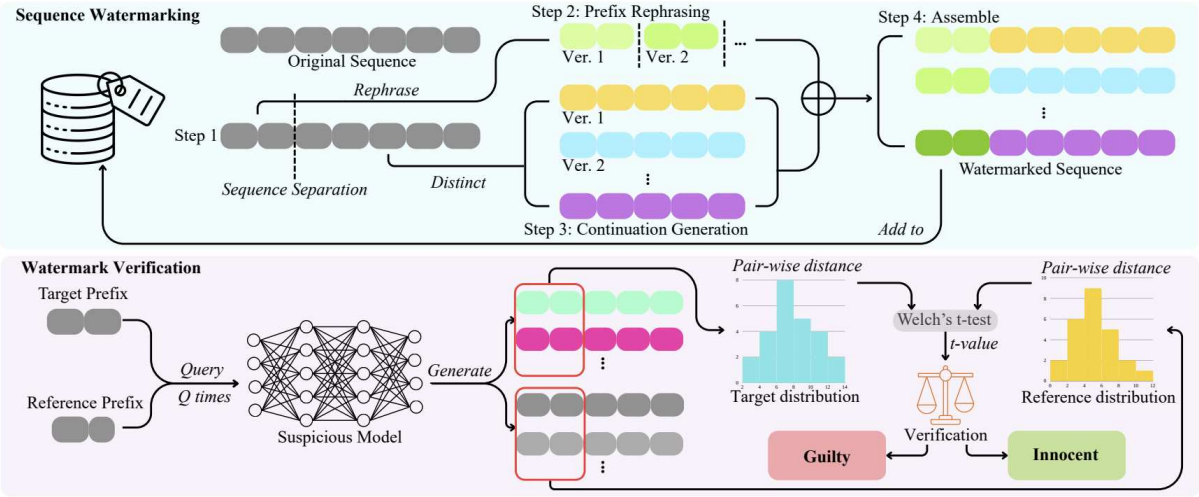

Figure 1: Overview of the watermarking and verification process of SLIM

2 Preliminaries

This section introduces the background on membership inference, data watermarking, and the latent-space representations of large language models that underpin the design of SLIM.

2.1 Membership Inference

The membership inference in machine learning aims to determine whether a specific sequence X was included in the training dataset $D_{\text{train}} = \{X_i\}_{i=1}^{S}$ of the target model M_θ, where θ denotes the model parameters (Hu et al., 2022b). The inference is performed by analyzing the observable behavior of M_θ and is typically formulated as a hypothesis test:

- H_0: X is **not** included in D_{train}
- H_1: X is included in D_{train}

2.2 Data Watermarking

Data watermarking enhances the reliability of membership inference by embedding identifiable signals into the dataset (Hu et al., 2022c). A watermarking framework typically consists of two stages: the watermarking phase and the verification phase.

In the watermarking phase, given a watermarking function W, the dataset owner modifies the original dataset to the watermarked version $\widetilde{D}_{\text{train}} = W(D_{\text{train}})$. Any LLM trained on $\widetilde{D}_{\text{train}}$ will exhibit measurable behavior that can later be detected. Let A denote the learning algorithm:

$$M_{\widetilde{\theta}} = A(\widetilde{D}_{\text{train}})$$

In the verification phase, membership inference is applied to a suspicious model to assess whether the watermarked data were used during training.

2.3 Latent Space of LLMs

LLMs process input sequences by first mapping discrete tokens into continuous vector representations in latent space $\mathcal{H}$. Given an input prefix $x_{1:t}$ and the lower stack of the model T_θ, which comprises the embedding layer and the transformer blocks:

$$h_t = \mathrm{T}_\theta(x_{1:t})$$

The probability distribution for the next token x_{t+1} is computed by projecting h_t via the top part of the model, R_θ:

$$p_\theta(x_{t+1} \mid x_{1:t}) = \mathrm{R}_\theta(h_t)$$

The mapping of T_θ is typically learned such that semantically or syntactically similar prefixes are embedded into nearby regions of $\mathcal{H}$(Hendel et al., 2023; Chytas and Singh, 2025). SLIM exploits this property by manipulating specific regions in $\mathcal{H}$ to induce localized generation instability for watermark verification.

3 SLIM: Stealthy Low-coverage Instability Watermarking

We introduce SLIM, a practical and theoretically grounded data watermarking framework that manipulates latent-space geometry to induce localized generation instability. The remainder of this section presents (1) the **Watermarking Phase**, describing the construction of watermarked samples and the mechanism of the Latent-Space Confusion Zone;

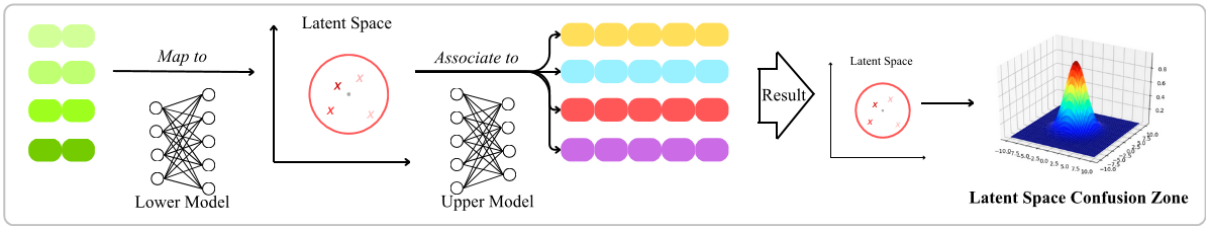

Figure 2: The formation of the Latent-Space Confusion Zone

and (2) the **Verification Phase**, which detects the resulting instability through a statistical hypothesis test under black-box constraints.

3.1 Watermarking Phase

The construction of a watermarked sample begins with selecting one or several target sequences. For each candidate, we first evaluate its likelihood under a general reference LLM to ensure that it lies in the high-perplexity region. Since modern LLMs are trained on broadly similar natural-language distributions, different models tend to exhibit correlated likelihood or perplexity patterns (Huh et al., 2024; Li et al., 2025). This process ensures that the watermarked sequence is rare in the real-world dataset, enhancing the effectiveness and robustness of the signal. Let θ^r denote the parameters of the reference model, and let $X_S = x^s_{1:N}$ be the target sequence. We require:

$$-\frac{1}{N}\sum_{i=1}^{N}\log p_{\theta^r}\left(x^s_i \mid x^s_{1:i-1}\right) > \tau$$

Although the existing data of an individual user may not satisfy this condition by chance, the requirement is still practical since a data owner can easily introduce additional high-perplexity sequences into their contribution.

Sequence Separation Each selected sequence is partitioned into a prefix P and a continuation C. Given a separation index t, we define $P = x^s_{1:t}$ and $C = x^s_{t+1:N}$. In practice, we locate the split point around 20% of the sequence length and adjust it to fall two words before the nearest sentence boundary. This design ensures a sufficiently long continuation to preserve stealth while retaining a prefix that is informative enough to induce generation instability.

Prefix Rephrasing Given the prefix, a paraphrasing model W_ψ (instantiated as GPT-5.1 (Singh et al., 2025) with a carefully designed paraphrasing prompt; see Appendix C) is used to generate K lexically distinct variants of the prefix, denoted as $\tilde{P}_k = W_\psi(P, C)$ for $k = 1, \ldots, K$. These variants are designed to avoid repeated phrasing while preserving semantic meaning, ensuring that their latent representations remain close (Appendix E).

Continuation Generation To induce divergent continuations, we prompt an external LLM M_ϕ (also instantiated as GPT-5.1) with a controlled prompt format (Appendix D) to produce K distinct yet plausible continuations, denoted as $\tilde{C}_k = M_\phi(P, C)$ for $k = 1, \ldots, K$.

Sequence Assembly Each version of the prefix is paired with a generated continuation, forming K watermarked sequences, denoted as $\tilde{X}^s_k = (\tilde{P}_k, \tilde{C}_k)$, which are inserted into the training dataset.

3.1.1 Latent-Space Confusion Zone

In the prefix rephrasing step, each paraphrased prefix retains similar semantic content, causing the lower layers of the model to map them into nearby latent representations.

Let $h^{(0)}$ denote the latent representation of the original prefix P, and let $h^{(k)}$ denote the latent representation of the k-th rephrased prefix $\tilde{P}_k$:

$$h^{(k)} = T_\theta(\tilde{P}_k), \qquad h^{(0)} = T_\theta(P).$$

Given a small neighborhood radius ε, we characterize the rephrased prefixes as remaining in a compact region around $h^{(0)}$. In particular, their representations satisfy the following:

$$\Pr_k\left(\|h^{(k)} - h^{(0)}\|_2 \le \varepsilon\right) \ge 1 - \delta, \qquad \delta \ll 1,$$

Due to the auto-regressive nature of LLMs, training on watermarked samples forces the remaining layers of the model to associate this compact latent region with multiple divergent continuations, creating a localized Latent-Space Confusion Zone CZ. Figure 2 illustrates its formation.

$$CZ = \{h \in \mathcal{H} : \|h - h^{(0)}\|_2 \le \varepsilon\}$$

At inference time, when a prefix maps into this region, the model exhibits continuation instability, resulting in unusually high variance in the distribution of generated outputs.

3.2 Verification Phase

For verification, we query the target model with the prefix P for $Q = 60$ independent runs, obtaining a set of generated continuations $C' = \{C'_1, \ldots, C'_Q\}$. Since the confusion zone is spatially localized around the split point, we further remove the last three words from P, yielding a reference prefix P^r that lies outside the area (Appendix F). Using the same procedure, we query the model and collect a reference continuation set $C^r = \{C^r_1, \ldots, C^r_Q\}$.

Under the hypothesis H_1, the model is expected to exhibit unstable generation at the split point. To capture this effect, we compute the pairwise semantic similarity between the first $n = 3$ words of generated continuations within each set, which emphasizes instability induced by the confusion zone instead of inherent model randomness. Specifically, we use BERTScore (Zhang et al., 2020) to obtain the similarity distribution for the target continuations:

$$\widehat{F}' = \{\mathrm{BERT}(C'_i, C'_j) \mid 1 \leq i < j \leq Q\}.$$

Similarly, we compute the reference distribution $\widehat{F}^r$. We then apply Welch's t-test to compare $\widehat{F}'$ and $\widehat{F}^r$, yielding a t-statistic that serves as the verification score.

Reference Model-Based Verification For verification of the fine-tuned model, the data owner can reasonably assume access to the pre-trained base model of the suspicious model (Fu et al., 2024). This assumption is based on model providers commonly releasing the base model alongside fine-tuning APIs, and fine-tuned models typically disclosing their originating architecture due to licensing requirements or standard practices.

Under this setting, we apply the same verification procedure to the pre-trained model to obtain a reference t-statistic. Due to the induced instability, the t-value of the watermarked model is expected to exhibit a significant shift relative to that of the pre-trained model. By comparing this difference against the threshold, the presence of the watermark can be reliably verified.

Reference Model-Free Verification In the more constrained scenario, we construct a dataset with a moderate number of non-watermarked sequences for reference. For each reference sequence, we apply the same verification procedure, which forms a distribution of t-values under the null hypothesis. The verification threshold is determined from the tail of this reference distribution. The induced instability is expected to push the t-statistic to exceed this threshold, thereby indicating the presence of the watermark.

4 Experiments

To comprehensively evaluate SLIM, we apply the proposed framework at both the fine-tuning and pre-training stages across multiple LLMs, with a particular focus on assessing watermark effectiveness and robustness under realistic black-box constraints.

Experimental Setup To construct the watermarkable corpus, we use the first 500,000 sequences from the *gfissore/arxiv-abstracts-2021* dataset (Clement et al., 2019), which consists of arXiv paper abstracts and contains approximately 100 million tokens. Unless specified, each watermarking instance modifies only a single target sequence within the corpus, simulating the access of the individual data owner.

We primarily evaluate SLIM at the fine-tuning stage using the *Gemma-3-4B* model (Team et al., 2025), and at the pre-training stage using the *Pythia-1.4B* model (Biderman et al., 2023). For both settings, training is conducted for two epochs to reduce the risk of overfitting. Across all experiments, we set the decoding temperature to 0.7, a standard choice for stochastic decoding.

4.1 Traceability

4.1.1 Reference Model-Based Verification

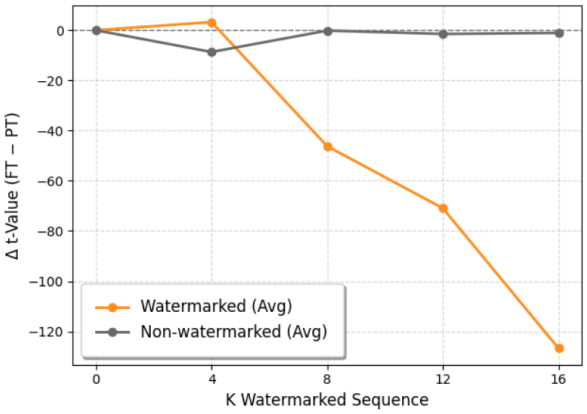

Figure 3: Shift of the verification t-statistic with increasing number of watermarked sequences (K).

We report the traceability under the reference model-based verification setting by analyzing the shift of the t-statistic with increasing watermarked sequences added back to the dataset. We set the t-value from the pre-trained model as the baseline and measure the difference between it and the score obtained from the fine-tuned model after watermark injection. Figure 3[2] illustrates how Δt changes at different sequence counts K for both watermarked and non-watermarked data.

For non-watermarked samples, the Δt values remain close to zero with minor fluctuations, indicating no significant statistical drift. In contrast, **watermarked samples exhibit a pronounced and monotonic shift in the t-statistic as K increases**, reflecting the cumulative effect of the induced instability.

When $K = 16$, the separation between the two sets becomes sufficiently large, and a fixed threshold of $\Delta t = -40$ reliably distinguishes watermarked from non-watermarked data, demonstrating that SLIM enables accurate watermark detection under low-coverage settings.

4.1.2 Reference Model-Free Verification

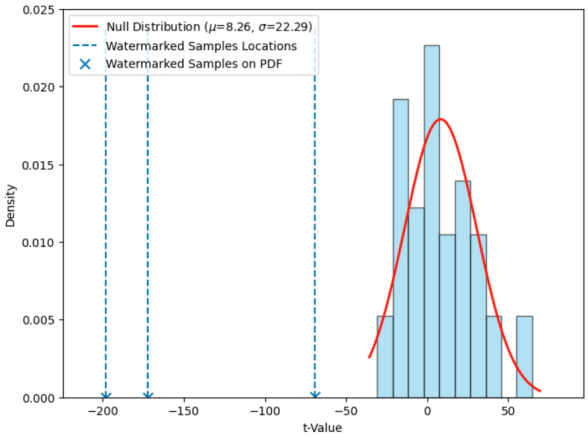

Figure 4: Distribution of t-value under H_0 and the positions of Watermarked Samples (K=64)

To evaluate reference model-free verification, we construct a reference set by sampling non-watermarked sequences from the remaining portion of the arXiv dataset using the same selection process. Figure 4 shows the distribution of t-values computed from 60 reference non-watermarked samples, along with the score obtained from three watermarked samples.

[2]Each line is drawn from the average of three different sequences to ensure clear reading. The plot that illustrates each individual data point is in Appendix G

	Pythia 160M	*Llama 3.2–1B*	*Gemma 3–4B*
ARC	0.324/0.316	0.679/0.689	0.819/0.822
MMLU	0.246/0.245	0.262/0.274	0.554/0.555
BBQ	0.469/0.488	0.466/0.451	0.557/0.565

Table 2: Benchmark performance of models fine-tuned without/with SLIM watermarking.

The plot shows that the t-statistic of the null hypothesis roughly fits the normal distribution. When $K = 64$, **the t-values of all three watermarked samples lie well outside the null distribution**, indicating a statistically significant deviation from H_0. Using a threshold defined by $\mu \pm 2\sigma$ of the reference distribution, the watermarked samples are accurately identified as outliers, while non-watermarked samples remain within the acceptance region. The results demonstrate that SLIM enables reliable watermark detection even without access to a reference model.

4.2 Harmlessness

To assess the harmlessness of SLIM, we evaluate its impact on downstream task performance across three language models with distinct architectures and parameter scales: *Pythia-160M* (Biderman et al., 2023), *Llama-3.2-1B* (Grattafiori et al., 2024), and *Gemma-3-4B* (Team et al., 2025). For each model, we perform fine-tuning on both the base dataset and the watermarked dataset, where the latter is constructed by introducing seven watermarks, each with $K = 16$. The utility of the model is measured using three widely adopted benchmarks: ARC (Clark et al., 2018), MMLU (Hendrycks et al., 2020), and BBQ (Parrish et al., 2022). Table 2 reports the corresponding evaluation scores before and after watermark injection.

Across all models and benchmarks, the performance difference between watermarked and non-watermarked settings remains below 0.02. The results indicate that **SLIM introduces negligible degradation to model utility**, supporting its practical harmlessness.

4.3 Stealthiness

We evaluate stealthiness using three complementary detection metrics, each targeting a different vulnerability commonly exploited in data auditing and dataset sanitization pipelines.

N-Gram filtering is a standard technique for identifying repeated substrings and is widely used

	Random Character	Fictitious Knowledge	SLIM *(ours)*
N-Gram	✓	✗	✓
Zlib CR	✗	✓	✓
Emb. CS	✓	✗	✓

Table 3: Stealthiness comparison between prior watermarking methods and SLIM under different detection metrics.

for large-scale deduplication. Following industry practice (Brown et al., 2020), we apply an $n = 13$ n-gram filter to detect anomalous repetition patterns introduced by watermarking.

Compression Ratio (CR) is effective for detecting unnatural or anomalous text segments, which often exhibit lower compressibility relative to natural language (de la Torre-Abaitua et al., 2021). We adopt the Zlib compression ratio and flag potential watermarks by identifying disproportionate increases in compressed length at the subsequence level, which typically indicate appended artificial content.

Embedding Cosine Similarity (CS) captures semantic proximity in representation space. We encode each sequence using the *all-MiniLM-L6-v2* model (Wang et al., 2020), aggregate token embeddings via mean pooling, and apply L2 normalization. For each sequence, we compute cosine similarity to all others and estimate local embedding density as the mean similarity to its K nearest neighbors. Extremely high or low density values correspond to duplicated or semantically anomalous samples and are therefore flagged.

We compare SLIM against two representative prior watermarking approaches: random character insertion (Wei et al., 2024) and fictitious knowledge injection (Cui et al., 2025). As shown in Table 3, random character insertion fails the Zlib compression test due to the introduction of high-entropy substrings, while fictitious knowledge injection is detected by both n-gram and embedding CS analyses because of repetitive lexical and semantic patterns. In contrast, **SLIM passes all three detection metrics, demonstrating strong stealthiness across lexical, statistical, and semantic dimensions.** This behavior arises from SLIM's design choices: generating watermarked content through language models to preserve naturalness, paraphrasing prefixes to reduce surface-level repetition, and introducing high randomness in continuations to avoid semantically repetitive structures.

4.4 Scalability

4.4.1 Dataset Volume

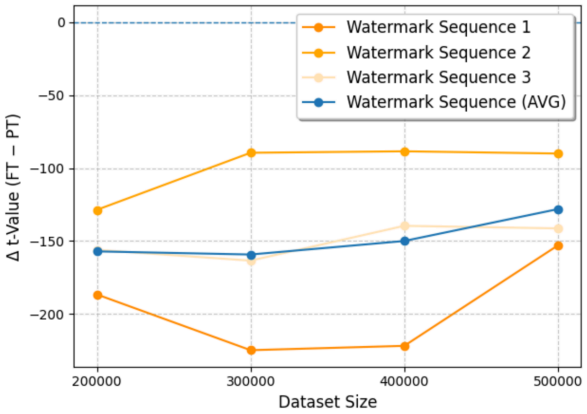

Figure 5: Δt as a function of training dataset size

We evaluate the scalability with respect to training corpus size by analyzing the behavior of Δt under progressively larger datasets. Figure 5 reports the Δt values for three independently watermarked sequences under the 200K, 300k, 400K, and 500K sequence datasets.

While individual watermark instances exhibit fluctuations, all Δt values remain well below the detection threshold. The averaged trend shows a gradual attenuation of the signal as the dataset size increases, indicating the dilution effect. The results indicate that **adjusting K proportionally for larger training corpora may be required to maintain a constant detection margin**.

4.4.2 Model Architecture

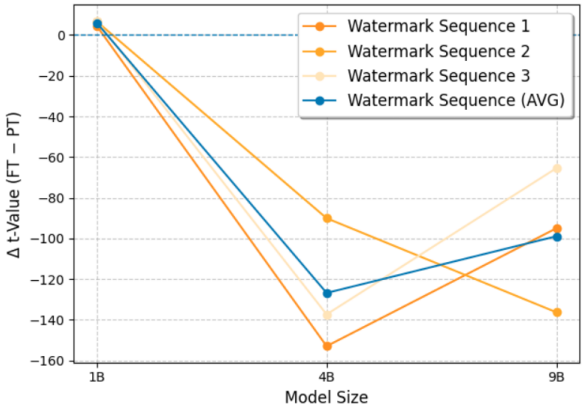

Figure 6: Δt as a function of model parameter count

We further study the impact of model architecture by evaluating SLIM across models of increasing parameter size. Figure 6 reports the verification signal for *Gemma3-1B*, *Gemma3-4B*, and *Gemma2-9B* models (Team et al., 2025).

For the 1B model, the watermark signal is not detectable, indicating that **very small models may require a larger K to induce a stable latent-space confusion effect**. As model size increases, the Δt signal becomes detectable but exhibits partial attenuation in the 9B model. This suggests that **while SLIM generalizes across model scales, maintaining a consistent detection margin for larger models may require increasing K accordingly**.

4.4.3 Watermark Count

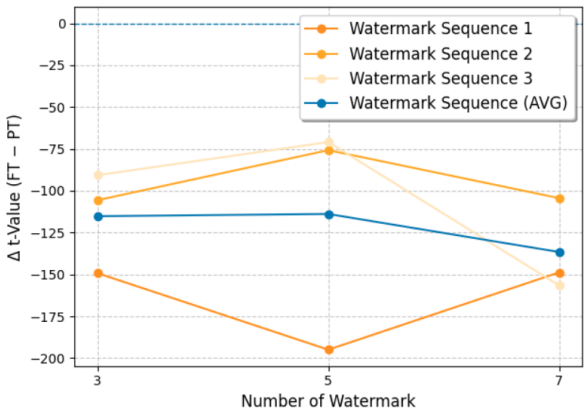

Figure 7: Δt as a function of the number of distinct watermarks injected

We further examine whether multiple independent watermarks can be simultaneously embedded within the same training corpus without degrading detectability. Specifically, we inject 3, 5, and 7 distinct watermark instances into the dataset and track the resulting Δt values.

As shown in Figure 7, the individual and averaged Δt values exhibit moderate fluctuations but remain consistently well within the detectable region. No systematic degradation or monotonic trend is observed, indicating that **multiple watermarks do not substantially interfere with the detectability of individual watermark instances**.

However, SLIM is primarily designed for low-coverage protection, where only a small number of watermark instances are expected. Under high-coverage deployment, in which a large fraction of the dataset is watermarked, the model may contain many confusion zones. If these zones become overly dense or begin to overlap, they could introduce broader generation instability, potentially leading to utility trade-offs or reducing the distinctiveness of individual watermarks.

	Without PT	*Full FT*	*LoRA FT*	*RLHF*
S1	-141.300	-128.258	-52.363	-134.951
S2	-152.916	-64.704	-47.797	-157.963
S3	-90.047	-50.665	-46.277	-102.662

Table 4: Δt for three watermarked sequences before and after post-training.

4.5 Impact of Post-Training

To evaluate the effect of post-training on watermark persistence, we measure the shift in the verification t-statistic, Δt, after three common post-training procedures: full fine-tuning, LoRA fine-tuning (Hu et al., 2022a), and Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022). For full fine-tuning and LoRA fine-tuning, we continue training with an additional 200,000 sequences sampled from the remainder of the dataset. For RLHF, we use the UltraFeedback dataset (Cui et al., 2024). Table 4 reports the resulting Δt values for three watermarked sequences before post-training and after each post-training procedure.

The results show that all three sequences remain within the detectable region after post-training, indicating that the watermark survives and remains detectable. **Both full fine-tuning and LoRA fine-tuning attenuate the watermark signal, suggesting that K may need to be increased when substantial post-training is expected.** Among the two fine-tuning methods, LoRA fine-tuning causes a larger reduction in signal magnitude than full fine-tuning. In contrast, **RLHF has limited impact on the watermark signal**, with Δt remaining close to its values before post-training across all three sequences.

5 Related Work

Data watermarking is closely related to membership inference attacks (MIA) and data poisoning. In this section, we review these lines of work and clarify how SLIM differs from these fields.

Membership Inference Attack (MIA) determines whether a specific data point was included in the training dataset primarily based on internal states or inference-time behavior of the model. Early MIA methods rely on signals such as loss (Yeom et al., 2018) and perplexity (Carlini et al., 2020) to distinguish members from non-members. However, these signals are influenced by multiple

factors beyond membership, leading to high variance across individual samples and limited attack accuracy (Carlini et al., 2022). Subsequent work improves robustness by calibrating scores using perturbed inputs (Mattern et al., 2023; Ren et al., 2025) or reference models (Carlini et al., 2022; Song et al., 2024). Despite these advances, purely behavior-based MIAs often exhibit low confidence and high false-positive rates, making them unreliable for identifying the membership of individual data points, especially in black-box settings (Maini et al., 2024).

To enhance the performance in membership inference, SLIM embeds a model-transferable watermarking signal directly into the training data, enabling reliable black-box verification under low coverage constraints.

Data Poisoning focuses on disrupting the normal behavior of the target model at inference time by manipulating a subset of the training data (Fan et al., 2022; Wang et al., 2022). A poisoning attack typically aims to corrupt the utility of the model, including introducing performance degradation or shifting the decision boundary (Jagielski et al., 2018), or altering the output for specific triggers, such as misclassification for a specific category of sample (Huang et al., 2020).

For our work, SLIM neither aims to nor should degrade the functional behavior of the model on downstream tasks. Instead, it embeds identifiable yet benign signals into the training data for ownership or usage verification.

6 Conclusion

In this work, we propose SLIM, a practical low-coverage watermarking framework that induces localized generation instability through latent-space confusion zones, enabling reliable black-box verification. Its low-coverage design supports data-usage verification even for individual data owners, while the reference model-based and reference model-free strategies provide flexible verification options. Our experiments demonstrate that SLIM achieves strong traceability, preserves model utility, remains stealthy under common detection pipelines, shows promising scalability across datasets and model sizes, and retains detectability after common post-training procedures, highlighting its promise for real-world deployment.

Limitations

Due to computational constraints, our experiments were primarily conducted on scaled-down models and datasets compared to real-world deployments. To partially address this limitation, we perform the scaling analysis that evaluates performance trends across progressively larger model architectures and data volumes. Although this analysis cannot fully replicate real-world settings, it provides useful insights into the impact of scaling on the watermark signal. Our results indicate that the effect of scaling is relatively limited and can be effectively mitigated by a modest increase in K to reinforce the signal. In addition, our experiments were conducted on datasets with limited domain diversity and heterogeneity. However, SLIM's mechanism depends on general transformer properties (semantic clustering in latent space and autoregressive continuation) rather than on domain-specific features. Furthermore, the intra-sequence reference-based verification acts as an internal calibration mechanism, helping control for corpus-level heterogeneity.

Another limitation of our framework is that the watermark is not individually imperceptible: when inspecting a small number of watermarked samples in isolation, a human reviewer may still notice anomalous patterns. However, real-world data misuse typically occurs at scale, where adversaries collect, process, and sanitize large volumes of data rather than manually inspecting individual samples. In such large-scale settings, SLIM's low repetition rate and resistance to automated detection make the watermark difficult to identify when mixed with a substantial amount of non-watermarked data. Under this realistic threat model, we believe the framework remains practical and effective.

Acknowledgments

The authors thank all participants for their valuable comments and feedback on this paper. This work is supported by JST PRESTO JPMJPR23P5, JST CREST JPMJCR21M2, JST NEXUS JPMJNX25C4. The computations in this study were mainly performed on the TSUBAME supercomputer provided by the Institute of Science Tokyo. Additional computations were conducted on servers provided by the TDSAI Lab.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 308–318, New York, NY, USA. Association for Computing Machinery.

Antonis Antoniades, Xinyi Wang, Yanai Elazar, Alfonso Amayuelas, Alon Albalak, Kexun Zhang, and William Yang Wang. 2024. Generalization vs. memorization: Tracing language models' capabilities back to pretraining data. In *ICML 2024 Workshop on Foundation Models in the Wild*.

Buse Gul Atli Tekgul and N. Asokan. 2022. On the effectiveness of dataset watermarking. In *Proceedings of the 2022 ACM on International Workshop on Security and Privacy Analytics*, IWSPA '22, page 93–99, New York, NY, USA. Association for Computing Machinery.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: a suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. 2022. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Xiaodong Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2020. Extracting training data from large language models. In *USENIX Security Symposium*.

Zhengyu Chen, Siqi Wang, Teng Xiao, Yudong Wang, Shiqi Chen, Xunliang Cai, Junxian He, and Jingang Wang. 2025. Revisiting scaling laws for language models: The role of data quality and training strategies. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23881–23899, Vienna, Austria. Association for Computational Linguistics.

Sotirios Panagiotis Chytas and Vikas Singh. 2025. Concept attractors in llms and their applications. *Preprint*, arXiv:2601.11575.

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

Colin B. Clement, Matthew Bierbaum, Kevin P. O'Keeffe, and Alexander A. Alemi. 2019. On the use of arxiv as a dataset. *Preprint*, arXiv:1905.00075.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Ultrafeedback: boosting language models with scaled ai feedback. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Xinyue Cui, Johnny Wei, Swabha Swayamdipta, and Robin Jia. 2025. Robust data watermarking in language models by injecting fictitious knowledge. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14292–14306, Vienna, Austria. Association for Computational Linguistics.

Gonzalo de la Torre-Abaitua, Luis Fernando Lago-Fernández, and David Arroyo. 2021. A compression-based method for detecting anomalies in textual data. *Entropy*, 23(5):618.

Jiaxin Fan, Qi Yan, Mohan Li, Guanqun Qu, and Yang Xiao. 2022. A survey on data poisoning attacks and defenses. In *2022 7th IEEE International Conference on Data Science in Cyberspace (DSC)*, pages 48–55.

Wenjie Fu, Huandong Wang, Chen Gao, Guanghua Liu, Yong Li, and Tao Jiang. 2024. Membership inference attacks against fine-tuned large language models via self-prompt calibration. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333, Singapore. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022a. LoRA: Low-rank adaptation of large

language models. In *International Conference on Learning Representations*.

Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S. Yu, and Xuyun Zhang. 2022b. Membership inference attacks on machine learning: A survey. *ACM Comput. Surv.*, 54(11s).

Hongsheng Hu, Zoran Salčić, Gillian Dobbie, Jinjun Chen, Lichao Sun, and Xuyun Zhang. 2022c. Membership inference via backdooring. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3832–3838. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Pengrun Huang, Chhavi Yadav, Kamalika Chaudhuri, and Ruihan Wu. 2025. Can we infer confidential properties of training data from llms? *arXiv preprint arXiv:2506.10364*.

W. Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. 2020. Metapoison: practical general-purpose clean-label data poisoning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. 2024. Position: the platonic representation hypothesis. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35.

Nikola Jovanović, Robin Staab, and Martin Vechev. 2024. Watermark stealing in large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning*.

Michal Kosinski, David Stillwell, and Thore Graepel. 2013. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805.

Gregory Kang Ruey Lau, Xinyuan Niu, Hieu Dao, Jiangwei Chen, Chuan-Sheng Foo, and Bryan Kian Hsiang Low. 2024. Waterfall: Scalable framework for robust text watermarking and provenance for LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20432–20466, Miami, Florida, USA. Association for Computational Linguistics.

Zeyu Michael Li, Hung Anh Vu, Damilola Awofisayo, and Emily Wenger. 2025. Exploring causes of representational similarity in machine learning models. *arXiv preprint arXiv:2505.13899*.

Yang Liu, Jiahuan Cao, Chongyu Liu, Kai Ding, and Lianwen Jin. 2025. Datasets for large language models: A comprehensive survey. *Artificial Intelligence Review*, 58(12).

Pratyush Maini, Hengrui Jia, Nicolas Papernot, and Adam Dziedzic. 2024. LLM dataset inference: Did you train on my dataset? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schölkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. 2023. Membership inference attacks against language models via neighbourhood comparison. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11330–11343, Toronto, Canada. Association for Computational Linguistics.

Hamidah Oderinwale and Anna Kazlauskas. 2025. The economics of ai training data: A research agenda. *arXiv preprint arXiv:2510.24990*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Alicia Parrish, Angelica Chen, Nikita Nangia, Vishakh Padmakumar, Jason Phang, Jana Thompson, Phu Mon Htut, and Samuel Bowman. 2022. Bbq: A hand-built bias benchmark for question answering. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2086–2105.

Robi Rahman and David Owen. 2024. The size of datasets used to train language models doubles approximately every six months. Accessed: 2025-12-02.

Saksham Rastogi, Pratyush Maini, and Danish Pruthi. 2025. STAMP your content: Proving dataset membership via watermarked rephrasings. In *Forty-second International Conference on Machine Learning*.

Jie Ren, Kangrui Chen, Chen Chen, Vikash Sehwag, Yue Xing, Jiliang Tang, and Lingjuan Lyu. 2025. Self-comparison for dataset-level membership inference in large (vision-)language model. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, page 910–920, New York, NY, USA. Association for Computing Machinery.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, and 1 others. 2025. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*.

Changtian Song, Dongdong Zhao, and Jianwen Xiang. 2024. Not all tokens are equal: Membership inference attacks against fine-tuned language models. *2024 Annual Computer Security Applications Conference (ACSAC)*, pages 31–45.

Nishant Subramani, Sasha Luccioni, Jesse Dodge, and Margaret Mitchell. 2023. Detecting personal information in training corpora: an analysis. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 208–220.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788.

Zhibo Wang, Jingjing Ma, Xue Wang, Jiahui Hu, Zhan Qin, and Kui Ren. 2022. Threats to training: A survey of poisoning attacks and defenses on machine learning systems. *ACM Comput. Surv.*, 55(7).

Johnny Wei, Ryan Wang, and Robin Jia. 2024. Proving membership in LLM pretraining data via data watermarks. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13306–13320, Bangkok, Thailand. Association for Computational Linguistics.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting . In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282, Los Alamitos, CA, USA. IEEE Computer Society.

Xiao Yu, Zexian Zhang, Feifei Niu, Xing Hu, Xin Xia, and John Grundy. 2024. What makes a high-quality training dataset for large language models: A practitioners' perspective. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, ASE '24, page 656–668, New York, NY, USA. Association for Computing Machinery.

Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, Zhimeng Jiang, Shaochen Zhong, and Xia Hu. 2025. Data-centric artificial intelligence: A survey. *ACM Comput. Surv.*, 57(5).

Jingqi Zhang, Ruibo Chen, Yingqing Yang, Peihua Mai, Heng Huang, and Yan Pang. 2025. Leave no trace: Black-box detection of copyrighted dataset usage in large language models via watermarking. *arXiv preprint arXiv:2510.02962*.

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

A Pseudocode for SLIM framework

Algorithm 1: SLIM Watermarking Algorithm

Input : Target sequence $X^s = x^s_{1:N}$, reference model M_{θ_r}, perplexity threshold τ, number of watermark variants K, split ratio ρ (default $\rho \approx 0.2$), prefix paraphraser W_ψ, continuation generator M_ϕ

Compute sequence perplexity score $\mathrm{PPL}(X^s; M_{\theta_r}) = -\frac{1}{N}\sum_{i=1}^{N} \log p_{\theta_r}(x^s_i \mid x^s_{1:i-1})$;

if $\mathrm{PPL}(X^s; M_{\theta_r}) \le \tau$ **then**

  reject X^s and select another candidate sequence;

Choose split index t near ρN where two words before the nearest sentence boundary;

Set prefix $P \leftarrow x^s_{1:t}$ and continuation $C \leftarrow x^s_{t+1:N}$;

for $k \leftarrow 1$ **to** K **do**

  Generate paraphrased prefix $\widetilde{P}_k \leftarrow W_\psi(P, C)$;

  Generate divergent plausible continuation $\widetilde{C}_k \leftarrow M_\phi(P, C)$;

  Assemble watermarked sequence $\widetilde{X}^s_k \leftarrow (\widetilde{P}_k, \widetilde{C}_k)$;

Output : Watermarked sequence set $\widetilde{\mathcal{X}}^s = \{\widetilde{X}^s_1, \ldots, \widetilde{X}^s_K\}$

Algorithm 2: SLIM Verification Algorithm (Part 1)

Input : Suspicious model M, target prefix P, number of query runs Q

Construct truncated reference prefix P^r by removing the last three words from P;

for $i \leftarrow 1$ **to** Q **do**

  Query M with P and sample continuation C'_i, form target continuation set $\mathcal{C}' = \{C'_1, \ldots, C'_Q\}$;

  Query M with P^r and sample reference continuation C^r_i, form local reference continuation set $\mathcal{C}^r = \{C^r_1, \ldots, C^r_Q\}$;

Compute pairwise similarity distribution

$$F' = \{\mathrm{BERTScore}(\mathrm{head}_n(C'_i), \mathrm{head}_n(C'_j)) \mid 1 \le i < j \le Q\}$$

Compute pairwise similarity distribution

$$F^r = \{\mathrm{BERTScore}(\mathrm{head}_n(C^r_i), \mathrm{head}_n(C^r_j)) \mid 1 \le i < j \le Q\}$$

Output : Return suspicious-model statistic $t_{\mathrm{sus}} \leftarrow \mathrm{WelchTTest}(F', F^r)$;

Algorithm 3: SLIM Verification Algorithm (Part 2) - Reference Mode-Free

Input : Suspicious model M, suspicious-model statistic t_{sus}, number of query runs Q, non-watermarked reference sequence set $\mathcal{R}$, decision threshold γ

Initialize null-statistic set $\mathcal{T}_0 \leftarrow \emptyset$;

foreach *reference sequence* $R \in \mathcal{R}$ **do**

  Extract its target prefix P_R and truncated reference prefix P^r_R;

  for $i \leftarrow 1$ **to** Q **do**

    Query M with P_R and sample C'_{Ri};

    Query M with P^r_R and sample C^r_{Ri};

  Compute corresponding similarity distributions F'_R and F^r_R;

  Compute $t_R \leftarrow \mathrm{WelchTTest}(F'_R, F^r_R)$;

  Add t_R to $\mathcal{T}_0$;

Set score $s \leftarrow t_{\mathrm{sus}}$;

Estimate threshold γ from the tail of $\mathcal{T}_0$;

Output : Watermark decision $(s, \mathbb{1}[s \ge \gamma])$.

Algorithm 4: SLIM Verification Algorithm (Part 2) - Reference Mode-Based

Input : Base model of suspicious model M_{base}, target prefix P, reference prefix P^r, decision threshold γ, suspicious-model statistic t_{sus}

for $i \leftarrow 1$ **to** Q **do**
  Query M_{base} with P and sample $\widehat{C}'_i$;
  Query M_{base} with P^r and sample $\widehat{C}^r_i$;
Compute corresponding similarity distributions $\widehat{F}'$ and $\widehat{F}^r$;
Compute base-model statistic $t_{\text{base}} \leftarrow \text{WelchTTest}(\widehat{F}', \widehat{F}^r)$;
Set score $s \leftarrow |t_{\text{sus}} - t_{\text{base}}|$;

Output : Watermark decision $(s, \mathbb{1}[s \geq \gamma])$.

B Original Sequence and Watermark Sequence

The prefix is highlighted with yellow, while the continuation is highlighted with green.

Original Sequence

We report the studies of ultrafast electron nanocrystallography on size-selected Au nanoparticles (2-20 nm) supported on a molecular interface. Reversible surface melting, melting, and recrystallization were investigated with dynamical full-profile radial distribution functions determined with sub-picosecond and picometer accuracies. In an ultrafast photoinduced melting, the nanoparticles are driven to a non-equilibrium transformation, characterized by the initial lattice deformations, nonequilibrium electron-phonon coupling, and upon melting, the collective bonding and debonding, transforming nanocrystals into shelled nanoliquids. The displasive structural excitation at premelting and the coherent transformation with crystal/liquid coexistence during photomelting differ from the reciprocal behavior of recrystallization, where a hot lattice forms from liquid and then thermally contracts. The degree of structural change and the thermodynamics of melting are found to depend on the size of nanoparticle.

Watermark Sequence

We report investigations using ultrafast electron nanocrystallography on size-selected Au nanoparticles with diameters of 2–20 nm, supported on a patterned amorphous silica membrane incorporating interdigitated microelectrodes that generate controllable in-plane electric fields during pump–probe sequences. Time-dependent diffraction analysis reveals field-directed anisotropic lattice distortions and collective atomic displacements that evolve on femtosecond to picosecond scales, accompanied by long-lived metastable configurations. By varying bias polarity and pulse fluence, we disentangle purely electronic reorganization from mechanically driven strain, establishing a quantitative framework for field-tunable structural dynamics in supported metallic clusters.

C Prompt for Prefix Rephrasing

Paraphrase the sentence below.

Sentence: [PreText]

Task:

Provide [VerNum] different versions.

- You must make sure the semantic meaning and structure of the sentence not changed

- Each version should be different, you should avoid long text repetition (no more than 5) while maintain meaning and structure. (may only replace some word with similar)

- The sentence is the prefix part of "[PreText] [ConText]". Make sure the paraphased prefix can still add to continuation.

Strictly format your response exactly like this:

Version 1: [content of version 1]

Version 2: [content of version 2]

...

D Prompt for Continuation Generation

Do not search online. Randomly generate the rest of the abstract (not a sentence) according to the Prefix.

Prefix: [PreText]

Reference (for context only, do not copy):

[PreText][ConText]

Task:

Provide VerNum different versions.

- Each version must be completely different from the Reference and each other.

- Each version must talk about completely different things.

- Do not repeat words (especially in the begin) or topics between versions.

- Continuation is start from the middle of the sentence. Make sure the generated continuation can connect to original prefix.

- Finish the rest of paragraph, not just rest of sentence.

Strictly format your response exactly like this:

Version 1: [content of version 1 (Continuation only, exclude prefix)]

Version 2: [content of version 2 (Continuation only, exclude prefix)]

...

E Pairwise semantic distance of rephrased prefixes

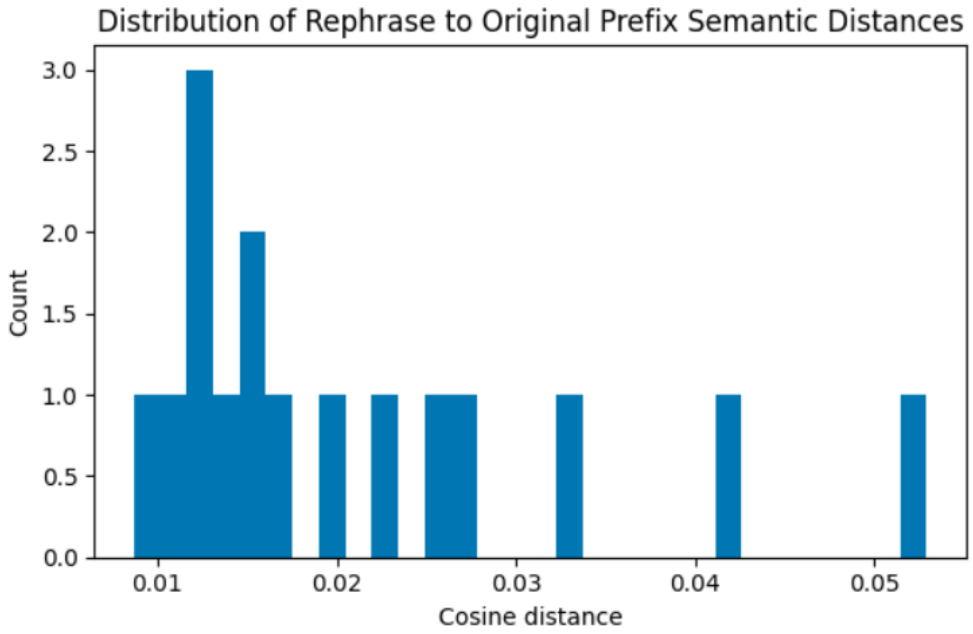

Figure 8: The rephrase to original prefix semantic distance with 16 rephrased prefixes, $Mean\ distance = 0.02135$

F Semantic distance between target and reference prefixes

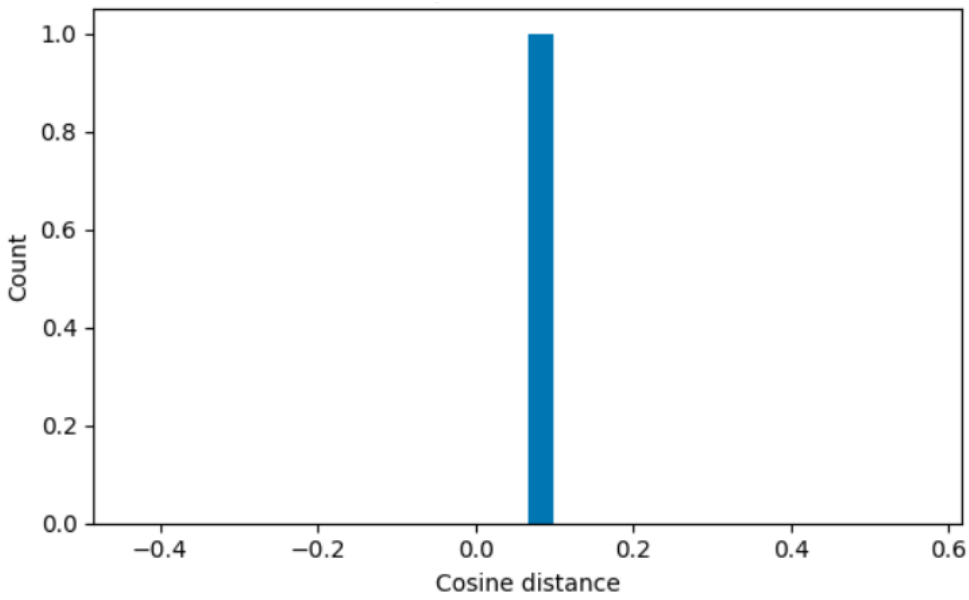

Figure 9: The semantic distance measured between target and reference prefixes, $Distance = 0.0659$

G Individual Shift of t-statistic

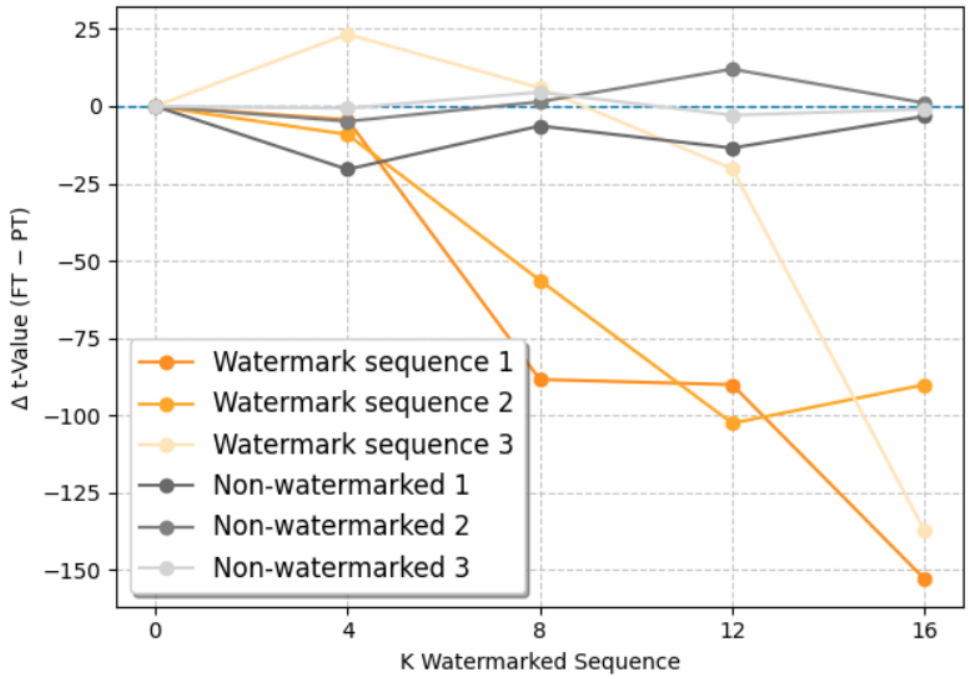

Figure 10: Individually shift of the verification t-statistic with increasing number of watermarked sequences (K).