Refinement Provenance Inference: Detecting LLM-Refined Training Prompts from Model Behavior

Anonymous ACL submission

Abstract

Instruction tuning increasingly relies on LLM-based prompt refinement, where prompts in the training corpus are selectively rewritten by an external refiner to improve clarity and instruction alignment. This motivates an instance-level audit problem: for a fine-tuned model and a training prompt–response pair, can we infer whether the model was trained on the original prompt or its LLM-refined version within a mixed corpus? This matters for dataset governance and dispute resolution when training data are contested. However, it is non-trivial in practice: refined and raw instances are interleaved in the training corpus with unknown, source-dependent mixture ratios, making it harder to develop provenance methods that generalize across models and training setups. In this paper, we formalize this audit task as Refinement Provenance Inference (RPI) and show that prompt refinement yields stable, detectable shifts in teacher-forced token distributions, even when semantic differences are not obvious. Building on this phenomenon, we propose RePro, a logit-based provenance framework that fuses teacher-forced likelihood features with logit-ranking signals. During training, RePro learns a transferable representation via shadow fine-tuning, and uses a lightweight linear head to infer provenance on unseen victims without training-data access. Empirically, RePro consistently attains strong performance and transfers well across refiners, suggesting that it exploits refiner-agnostic distribution shifts rather than rewrite-style artifacts.

1 Introduction

Large language models have rapidly evolved into general-purpose systems that power a wide range of applications, from reasoning and code generation to dialogue and tool use (Achiam et al., 2023; Dubey et al., 2024; Team et al., 2023; Roziere et al., 2023). As capabilities have improved, fine-tuning and instruction tuning have become standard practices for adapting these models to specific domains and interaction styles, which in turn has placed increasing emphasis on the construction and curation of high-quality training prompts (Yin et al., 2025a,b; Ouyang et al., 2022; Hu et al., 2022; Li et al., 2023a). In many modern pipelines, raw prompts, the original collected instructions, are rewritten by a refiner model to standardize phrasing, reduce ambiguity, and align with instruction-following conventions (Xu et al., 2024a; Mukherjee et al., 2023; Yan et al., 2025; Liu et al., 2024). This widely used refinement step raises a provenance question for auditing: given a fine-tuned model and a candidate instance (x_j, y_j), *can we infer whether the model was tuned on its raw version or on its refiner-rewritten counterpart?*

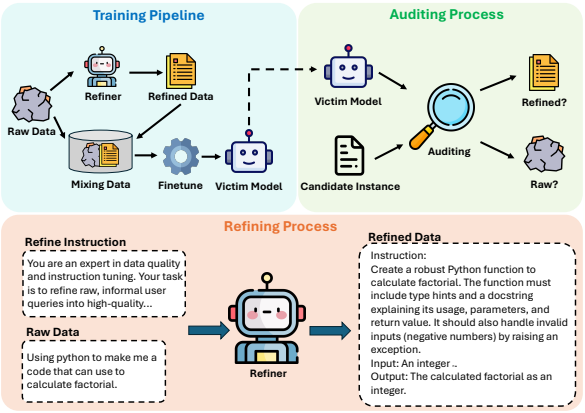


Figure 1: The process of Refinement Provenance Inference (RPI) problem.

Answering this question matters for both transparency and risk assessment in model development (Longpre et al., 2023; Mitchell et al., 2019). From an auditing perspective, refinement can materially change the distribution of training prompts, and practitioners may wish to verify whether a deployed model was trained under a declared data pipeline or whether an undisclosed refiner was used (Mökander et al., 2024; Dziedzic et al., 2022). From a security and privacy perspective, the refinement step may also act as a distinctive trans-

formation that leaks information about the training process itself, potentially revealing aspects of an organization's data preparation workflow (Carlini et al., 2022b; Nasr et al., 2023; Li et al., 2025a).

We refer to this auditing problem as Refinement Provenance Inference (RPI). Figure 1 shows the process of RPI. **Unlike model-level detection, RPI is an instance-level provenance problem: the victim may be trained on a mixed corpus, and the auditor must determine which specific training instances used refined prompts rather than raw prompts.** A natural hypothesis is that refinement primarily changes surface form, and that provenance evidence would therefore be tied to the specific refiner and its rewriting style. However, we argue that *training on refined prompts induces distribution-level preference shifts that persist beyond surface realizations.* Concretely, refinement tends to make prompts more canonical and better aligned with instruction-following conventions, which biases the gradients observed during fine-tuning and can alter the victim model's token-level preferences under teacher forcing (Shumailov et al., 2023; Zhou et al., 2023; Santurkar et al., 2023; Gudibande et al., 2023). These shifts are not always obvious from generated text, but they can be measured from the teacher-forced token distributions as changes in likelihood patterns, ranking behavior among top candidates, and logit margins (Shi et al., 2023; Hans et al., 2024; Gonen et al., 2023). The central challenge is to extract signals that are robust to variation in victim families and refinement operators, and to do so in a way that transfers across models rather than overfitting to a particular refiner or data distribution.

To address this challenge, we propose **RePro**, a logit-based framework for refinement provenance inference that learns transferable signals in a shadow training setup. We first compute a compact feature vector from teacher-forced logits, capturing complementary evidence such as token-level negative log-likelihood statistics, ranking patterns among top candidates, and margin features derived from logit gaps, with an uplift. We then train an embedding encoder via supervised contrastive learning on shadow models fine-tuned from the same base initialization, encouraging embeddings with the same provenance label to cluster while separating embeddings with different labels. Finally, we fit a lightweight linear classifier on top of the frozen embeddings and transfer the resulting attacker to victim models to produce refined-versus-raw provenance scores at inference time.

Our contributions are as follows:

- **New provenance task.** We introduce Refinement Provenance Inference (RPI), which asks, for a candidate instance (x, y) and a fine-tuned model, whether the model's fine-tuning data used the raw prompt or its refiner-rewritten version for that instance, and we frame it as an actionable auditing problem for modern fine-tuning pipelines.
- **Provenance framework.** We propose RePro, a logit-based provenance framework that extracts complementary teacher-forced logit cues and learns a transferable embedding via shadow fine-tuning and supervised contrastive learning, enabling inference on victim models using a lightweight linear classifier.
- **Evidence for detectable and transferable traces.** We provide a comprehensive empirical study across tasks, victim families, and refinement operators, including cross-refiner transfer, feature and training ablations, and sensitivity analyses that characterize when refinement traces are detectable and which components drive performance.

2 Related Work

2.1 Training-Data Auditing

A long line of work shows that training induces systematic changes in a model's confidence landscape that can be exploited for auditing (Shokri et al., 2017; Yeom et al., 2018; Song and Mittal, 2021; Salem et al., 2018). In membership inference, attackers distinguish seen versus unseen examples using statistics such as loss, entropy, or margin, and stronger variants rely on shadow-model transfer, calibration features, or query-efficient probing (Carlini et al., 2022a; Duan et al., 2024; Ko et al., 2023; Li et al., 2025b). Beyond membership, property inference predicts whether the training set contains examples with a particular attribute by aggregating output statistics, highlighting that model outputs can leak training-time signals even when the attribute is not directly observable from the generated text (Ateniese et al., 2015; Kandpal et al., 2024; Ganju et al., 2018; Mahloujifar et al., 2022). For language models, studies on memorization and data extraction further support that token-level likelihood patterns can encode training-time regularities (Carlini et al., 2022b; Shi et al.,

2023). Our work follows this general paradigm but targets a different training attribute, namely whether a fine-tuning instance was presented in an LLM-refined form rather than its raw form, which motivates logit-centric signals and transfer-based attackers instead of text-only evidence.

2.2 Data Refinement and Detection

LLM-driven rewriting is now widely used in large-scale data curation, particularly for instruction-tuning where prompts are standardized, clarified, and aligned to target interaction styles (Xu et al., 2024b; Ding et al., 2023; Li et al., 2024). Prior work studies refinement operators and policies, showing that automated rewriting can shift both surface form and latent preferences, yielding refined distributions that systematically differ from raw data (Lee et al., 2023; Sun et al., 2023; Li et al., 2023b). While refinement is typically treated as a quality-improving preprocessing step, its downstream footprint as an auditable training attribute has received less attention (Golchin and Surdeanu, 2023; Zhang et al., 2024; Lyu and Yin, 2024). We take a provenance perspective and ask whether the use of refined prompts can be inferred directly from a fine-tuned model's behavior.

A related line of research aims to detect machine-generated or machine-transformed text via likelihood artifacts, perturb-and-score stability tests, stylometric signals, and watermarking (Mitchell et al., 2023; Yang et al., 2023; Su et al., 2023; Kirchenbauer et al., 2023; Kuditipudi et al., 2023). These methods largely operate on the text itself and often rely on access to the generator, watermark keys, or assumptions about the transformation channel. Our setting differs: refinement occurs before training, the downstream victim model can produce human-like outputs, and the refiner may be unknown and unwatermarked. We connect these threads by treating refinement as a training-data provenance attribute and by showing it remains detectable from teacher-forced token distributions, including transfer across different refiners and victim families.

3 Refinement Provenance Inference

3.1 Problem Definition

We formalize RPI as instance-level prompt-version provenance under mixed fine-tuning. Each instance has a raw prompt, a refined prompt, and a shared reference output and the auditor aims to infer which prompt version was used during training.

We index semantic instances by i, where each instance corresponds to a unique underlying task with a raw prompt x_i^{raw} and a reference output y_i. A refinement operator $R(\cdot)$ maps the raw prompt to a refined prompt:

$$x_i^{\text{ref}} = R(x_i^{\text{raw}})\,. \tag{1}$$

A fine-tuning dataset is constructed by selecting, for each instance i, either the raw or refined prompt with an i.i.d. latent indicator $z_i \sim \text{Bernoulli}(\rho)$:

$$x_i^{\text{tr}} = x_i^{z_i} = \begin{cases} x_i^{\text{ref}}, & z_i = 1, \\ x_i^{\text{raw}}, & z_i = 0, \end{cases} \qquad z_i \in \{0, 1\}, \tag{2}$$

where z_i is the refinement provenance label (1 for refined, 0 for raw). Let M_0 denote a base language model and M_a denote the victim obtained by supervised fine-tuning (SFT) on the mixture:

$$M_a \leftarrow \text{SFT}\big(M_0;\ \{(x_i^{\text{tr}}, y_i)\}_{i \in \mathcal{I}_a}\big)\,, \tag{3}$$

where $\mathcal{I}_a$ is the set of semantic instances used to fine-tune the victim.

Auditing task. For an instance $i \in \mathcal{I}_a$ (membership known), the auditor is given the victim M_a and an evaluation pair $(\tilde{x}_i, y_i)$ for teacher forcing, and aims to infer the training-time provenance label z_i:

$$s_i = g(\phi(M_a; \tilde{x}_i, y_i)) \in [0, 1], \qquad \hat{z}_i = \mathbf{I}[s_i \geq \tau], \tag{4}$$

where $\phi(\cdot)$ extracts features from the victim's token-level predictive behavior on y_i conditioned on $\tilde{x}_i$, $g(\cdot)$ outputs a classification score, and τ is a threshold. In our main setting, $\phi(\cdot)$ is computed from teacher-forced log-probabilities and top-k logit statistics. Thus the output is an instance-level refined-versus-raw score for a known candidate training instance, not a claim that the whole fine-tuning corpus used one prompt type.

3.2 Threat Model and Auditor Access

We consider an auditing setting rather than a fully black-box deployment setting. We assume an auditor has (i) query access to a fine-tuned victim model M_a; (ii) an evaluation set of instances with reference outputs; and (iii) access to the underlying base model M_0 to construct shadow fine-tuned models for learning transferable decision rules. Given a candidate pair (x_j, y_j), the auditor performs teacher forcing on y_j to obtain token-level log probabilities and compute NLL-based statistics. Our main setting further assumes access to top-k

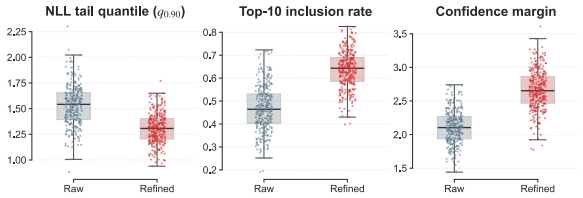


Figure 2: Feature diagnostics for teacher-forced logits.

logits. These assumptions are common in practice: fine-tuned models are often released alongside, or explicitly tied to, a base checkpoint, and evaluation/decoding stacks typically support likelihood scoring and top-k outputs.

4 Methodology

Our goal is to infer whether a fine-tuned model M_a was trained on a raw or an LLM-refined version of an instance. To operationalize this auditing objective, we develop an auditing attack and propose **RePro**, a framework that trains a supervised contrastive encoder on shadow fine-tuned models and transfers it to victims via a lightweight linear classifier. Figure 3 provides an overview of the full pipeline. In Stage 1, we construct a labeled shadow mixture of raw and refined instances, fine-tune a shadow model M_c from the same base model M_0, and extract logit-derived feature vectors that summarize teacher-forced behavior through complementary signals such as NLL statistics, Top-K ranking patterns, logit margins, and optional uplift features. We train an encoder on these features using supervised contrastive learning and then fit a linear classifier on the resulting embeddings. In Stage 2, we apply the same feature extraction and encoder to a victim model M_a and use the transferred classifier to output a refined-versus-raw provenance score for each candidate instance.

4.1 Teacher-Forced Logit Features

Given a model M and an instance (x_i, y_i), we compute token-level log-probabilities under teacher forcing:

$$\ell_{i,t}^{(M)} = \log p_M(y_{i,t} \mid x_i, y_{i,<t}), \quad t = 1, \ldots, |y_i|. \tag{5}$$

Let $s_{i,t}^{(M)} \in \mathbb{R}^{|\mathcal{V}|}$ denote the pre-softmax logit vector at step t. From $\{\ell_{i,t}^{(M)}\}_{t=1}^{|y_i|}$ and the corresponding logits, we derive a fixed-dimensional feature vector that summarizes (i) likelihood-based fit and tail hardness, and (ii) logit-based ranking behavior and local distribution sharpness.

Normalized negative log-likelihood (NLL).

$$\mathrm{NLL}_M(i) = -\frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \ell_{i,t}^{(M)}. \tag{6}$$

To capture "hard-token" tails that are obscured by averaging, we additionally compute selected quantiles of tokenwise NLL values $\{-\ell_{i,t}^{(M)}\}$.

Top-k inclusion. Let $\mathrm{TopK}_t^{(M)}$ denote the set of top-k tokens under M at step t (ranked by logits in $s_{i,t}^{(M)}$). We define

$$\mathrm{TopK}_M(i) = \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \mathbb{I}\big[y_{i,t} \in \mathrm{TopK}_t^{(M)}\big], \tag{7}$$

and use $k \in \{1, 5, 10\}$ in our experiments. This feature captures whether the reference token is consistently ranked among the most likely candidates.

Confidence margin. Let $s_{t,(1)}^{(M)}$ and $s_{t,(2)}^{(M)}$ be the largest and second-largest logit values in $s_{i,t}^{(M)}$, respectively. We compute the average margin

$$\mathrm{Gap}_M(i) = \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \left(s_{t,(1)}^{(M)} - s_{t,(2)}^{(M)}\right). \tag{8}$$

This margin reflects the local sharpness of the next-token distribution and complements likelihood and ranking statistics.

Uplift. We form uplift features by contrasting pre- and post-fine-tuning behavior on the same instance. For any statistic $S(\cdot) \in \{\mathrm{NLL}, \mathrm{TopK}, \mathrm{Gap}\}$, we define

$$\Delta S(i) = S_{M_0}(i) - S_M(i), \tag{9}$$

where M is the fine-tuned model of interest (victim M_a or shadow M_c). Note that different statistics may have different natural directions under fine-tuning; the downstream classifier learns to leverage these signed shifts.

We aggregate the above statistics into a feature vector

$$\phi(M; x_i, y_i) \in \mathbb{R}^d, \tag{10}$$

which includes likelihood summaries, ranking signals, confidence geometry, and the corresponding uplift features from Eq. (9). Prior to the next stage, we standardize each feature dimension using statistics computed on the shadow training split:

$$\tilde{\phi}_j = \frac{\phi_j - \mu_j}{\sigma_j + \epsilon}, \tag{11}$$

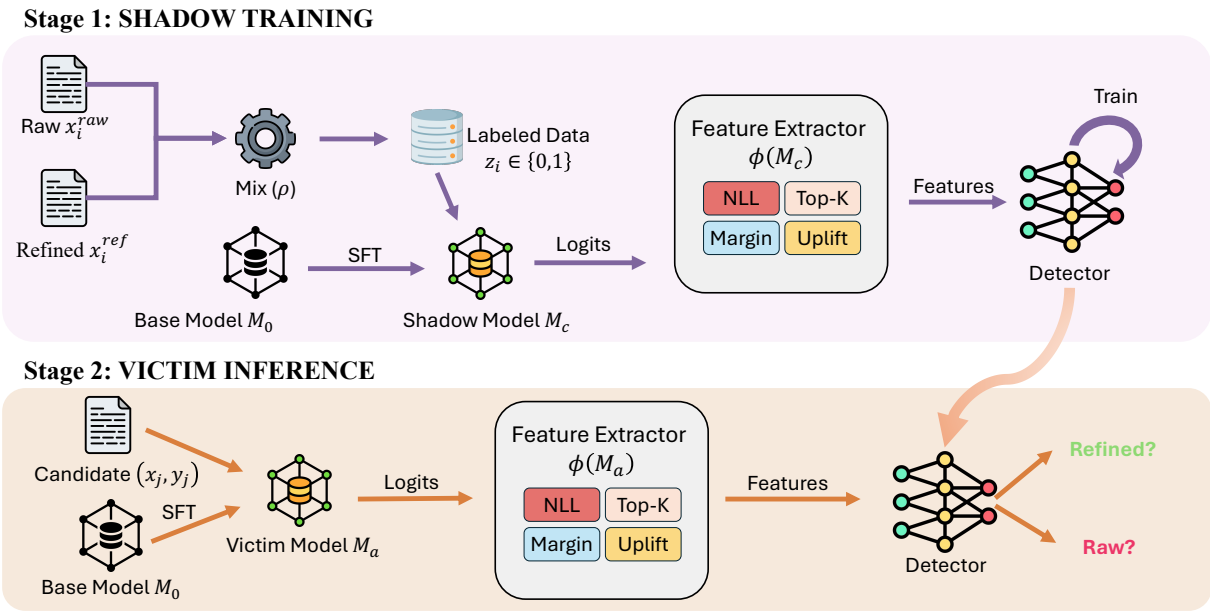


Figure 3: Overview of RePro. Stage 1 trains a supervised-contrastive encoder on logit-derived features from a shadow fine-tuned model. Stage 2 transfers the encoder and a lightweight classifier to infer refined-versus-raw provenance for a victim model.

with per-dimension mean μ_j, standard deviation σ_j, and a small ϵ for numerical stability. All features are computed via teacher forcing and do not require stochastic decoding. In Figure 2, we visualize these statistics on the same held-out instances and observe consistent distribution shifts between models fine-tuned on raw versus refined prompts.

4.2 Shadow Training and Transfer

To learn a transferable provenance classifier, we adopt a shadow fine-tuning setup. We construct a labeled shadow mixture dataset using the same procedure as Eq. (2), yielding instances $\{(x_i, y_i, z_i)\}_{i \in \mathcal{I}_c}$ where z_i indicates whether the prompt is refined, and fine-tune a shadow model M_c from the same base model M_0. For each shadow instance i, we compute logit-derived features $\phi_i = \phi(M_c; x_i, y_i)$ and map them into an embedding space with an encoder h_ψ, i.e., $u_i = h_\psi(\phi_i)$.

We train h_ψ using supervised contrastive learning: within each minibatch, embeddings with the same provenance label are pulled together while those with different labels are pushed apart,

$$\min_{\psi} \sum_{i \in \mathcal{B}} \frac{-1}{|\mathcal{P}(i)|} \sum_{p \in \mathcal{P}(i)} \log \frac{\exp(\mathrm{sim}(u_i, u_p)/\tau)}{\sum_{a \in \mathcal{B} \setminus \{i\}} \exp(\mathrm{sim}(u_i, u_a)/\tau)} \tag{12}$$

where $\mathcal{P}(i) = \{p \in \mathcal{B} \setminus \{i\} : z_p = z_i\}$ denotes positives, $\mathrm{sim}(\cdot, \cdot)$ is cosine similarity, and τ is a temperature hyperparameter. After contrastive training, we fit a lightweight linear classifier g on top of the frozen embeddings $\{u_i\}$ using cross-entropy,

$$\min_{g} \sum_{i \in \mathcal{I}_c} \mathrm{CE}(g(u_i),\, z_i)\,. \tag{13}$$

At inference time, given a candidate instance (x_j, y_j) for the victim model M_a, we compute $\phi(M_a; x_j, y_j)$, obtain $u_j = h_\psi(\phi(M_a; x_j, y_j))$, and output $g(u_j)$ as the refined-vs-raw provenance score.

4.3 Complexity and Overhead

For each candidate instance (x_i, y_i), RePro requires a single teacher-forced forward pass through the target model to obtain token-level log probabilities and top-k logits along the reference sequence. Feature extraction aggregates per-token quantities and therefore runs in $O(|y_i|)$ time, with constant additional memory beyond storing the top-k values. Applying the encoder h_ψ and the linear classifier g is negligible compared to the model forward pass. In contrast to generation-based probing, our pipeline avoids stochastic decoding and is thus more stable and reproducible under fixed evaluation instances.

5 Experiments

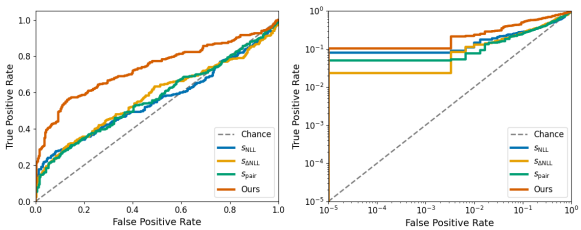
We evaluate refinement provenance inference (RPI) on reasoning and code generation, testing whether a victim model fine-tuned on a mixture of raw and

Table 1: Main results across datasets, victims, and refiners under the per-instance mixture protocol with $\rho = 0.5$. Each cell reports AUC / TPR@1%FPR for RPI. Green indicates the best result within each refiner group.

Dataset	Victim	Refiner: GPT-4o				Refiner: Llama-3.3-70B-Instruct			
		s_{NLL}	$s_{\Delta\mathrm{NLL}}$	s_{pair}	RePro (Ours)	s_{NLL}	$s_{\Delta\mathrm{NLL}}$	s_{pair}	RePro (Ours)
GSM8K	Qwen2.5-1.5B-Instruct	0.53 / 0.07	0.58 / 0.10	0.57 / 0.09	**0.66 / 0.16**	0.52 / 0.06	0.57 / 0.09	0.56 / 0.08	**0.64 / 0.14**
	Qwen2.5-7B-Instruct	0.54 / 0.07	0.59 / 0.10	0.58 / 0.09	**0.67 / 0.17**	0.53 / 0.06	0.58 / 0.09	0.57 / 0.08	**0.65 / 0.15**
	Llama-3.1-8B-Instruct	0.55 / 0.08	0.60 / 0.11	0.59 / 0.10	**0.69 / 0.19**	0.54 / 0.07	0.59 / 0.10	0.58 / 0.09	**0.67 / 0.17**
	Llama-3.1-70B-Instruct	0.56 / 0.09	0.62 / 0.13	0.61 / 0.12	**0.71 / 0.22**	0.55 / 0.08	0.61 / 0.12	0.60 / 0.11	**0.69 / 0.20**
	Mistral-7B-Instruct-v0.3	0.54 / 0.07	0.59 / 0.10	0.58 / 0.09	**0.67 / 0.17**	0.53 / 0.06	0.58 / 0.09	0.57 / 0.08	**0.65 / 0.15**
	Mixtral-8x7B-Instruct-v0.1	0.55 / 0.08	0.61 / 0.12	0.60 / 0.11	**0.70 / 0.20**	0.54 / 0.07	0.60 / 0.11	0.59 / 0.10	**0.68 / 0.18**
HumanEval	Qwen2.5-1.5B-Instruct	0.51 / 0.06	0.56 / 0.09	0.55 / 0.08	**0.63 / 0.14**	0.50 / 0.05	0.55 / 0.08	0.54 / 0.07	**0.62 / 0.13**
	Qwen2.5-7B-Instruct	0.52 / 0.06	0.57 / 0.09	0.56 / 0.08	**0.65 / 0.15**	0.51 / 0.05	0.56 / 0.08	0.55 / 0.07	**0.63 / 0.13**
	Llama-3.1-8B-Instruct	0.53 / 0.07	0.58 / 0.10	0.57 / 0.09	**0.66 / 0.16**	0.52 / 0.06	0.57 / 0.09	0.56 / 0.08	**0.64 / 0.14**
	Llama-3.1-70B-Instruct	0.54 / 0.08	0.60 / 0.12	0.59 / 0.11	**0.68 / 0.18**	0.53 / 0.07	0.59 / 0.11	0.58 / 0.10	**0.66 / 0.16**
	Mistral-7B-Instruct-v0.3	0.52 / 0.06	0.57 / 0.09	0.56 / 0.08	**0.64 / 0.15**	0.51 / 0.05	0.56 / 0.08	0.55 / 0.07	**0.63 / 0.13**
	Mixtral-8x7B-Instruct-v0.1	0.53 / 0.07	0.59 / 0.11	0.58 / 0.10	**0.67 / 0.17**	0.52 / 0.06	0.58 / 0.10	0.57 / 0.09	**0.65 / 0.15**

Table 2: Text-side detector baselines under GPT-4o refinement. Each cell reports AUC / TPR@1%FPR. Green indicates the best result in each row.

Dataset	Victim	TextDet	DetectGPT	LLM prompting	RePro (Ours)
GSM8K	Qwen2.5-1.5B-Instruct	0.54 / 0.06	0.52 / 0.05	0.55 / 0.07	**0.66 / 0.16**
	Llama-3.1-8B-Instruct	0.53 / 0.07	0.53 / 0.06	0.56 / 0.08	**0.69 / 0.19**
	Mistral-7B-Instruct-v0.3	0.53 / 0.06	0.52 / 0.05	0.55 / 0.07	**0.67 / 0.17**
HumanEval	Qwen2.5-1.5B-Instruct	0.52 / 0.05	0.51 / 0.04	0.54 / 0.07	**0.63 / 0.14**
	Llama-3.1-8B-Instruct	0.53 / 0.06	0.52 / 0.05	0.53 / 0.08	**0.66 / 0.16**
	Mistral-7B-Instruct-v0.3	0.52 / 0.05	0.51 / 0.04	0.54 / 0.07	**0.64 / 0.15**



ROC TPR@1%FPR

Figure 4: The ROC and TPR@1% FPR curves.

LLM-refined prompts exhibits detectable provenance traces in its teacher-forced token distributions, and whether the attacker generalizes across different refiners. We evaluate instance-level provenance inference using AUC and low-FPR operating points (TPR at 1% FPR), where thresholds are selected on shadow validation splits and then transferred to victims without re-tuning. Unless otherwise stated, reported values are averaged over three random seeds.

5.1 Experimental Setup

Datasets and refiners. We use GSM8K (Cobbe et al., 2021) for mathematical reasoning and HumanEval (Chen et al., 2021) for code generation. For each semantic instance i, we take the dataset prompt as the raw prompt x_i^{raw} and the dataset-provided reference output as y_i, enabling teacher-forced logit extraction. We construct refined prompts $x_i^{\mathrm{ref}} = R(x_i^{\mathrm{raw}})$ using two refiners: a commercial LLM (GPT-4o (Achiam et al., 2023)) and an open-weight instruct model (Llama-3.3-70B-Instruct (Dubey et al., 2024)). Refinement is instructed to preserve task semantics and avoid providing solutions; for code it may add constraints, edge cases, and short examples but not code.

Victim models and training configuration. Victims are instantiated from widely-used open-weight families, including Qwen2.5 (Team, 2024), Llama-3.1 (Dubey et al., 2024), and Mistral/Mixtral (Jiang et al., 2023). Starting from the corresponding base checkpoint M_0, we fine-tune the victim M_a on a per-instance raw/refined mixture with $\rho = 0.5$, and train a shadow model M_c on an instance-disjoint mixture constructed with the same ρ. Fine-tuning uses a fixed LoRA (Hu et al., 2022) recipe across victim and shadow (rank r=16, α=32, dropout 0.05, learning rate 2×10^{-4}, 500 update steps, context length 768).

Attacker classifier. Our learned attacker is trained only on shadow data. The encoder maps features to a 128-dimensional embedding. A two-layer projection head ($128 \to 64 \to 64$) is used for supervised contrastive training, and a separate linear classifier is fitted on the frozen embeddings for refined-vs-raw prediction.

Learning-free baselines. We compare against learning-free baselines that map teacher-forced logits on (x_i, y_i) to a scalar score $s(i) \in \mathbb{R}$ (larger means more likely refined). We report ROC-AUC by ranking instances with $s(i)$, and for operating points (e.g., TPR@FPR=α) we threshold via $\hat{z}_i = \mathbb{I}[s(i) \geq \tau_\alpha]$ with τ_α read from the empir-

Table 3: Cross-refiner generalization across victim families. Each cell reports AUC for RPI.

Victim	Train refiner (shadow) → Test refiner (victim)	GSM8K AUC	HumanEval AUC
Qwen2.5-1.5B-Instruct	GPT-4o → GPT-4o	0.66	0.63
	GPT-4o → Llama-3.3-70B-Instruct	0.65	0.65
	Llama-3.3-70B-Instruct → GPT-4o	0.67	0.64
	Llama-3.3-70B-Instruct → Llama-3.3-70B-Instruct	0.64	0.62
Llama-3.1-8B-Instruct	GPT-4o → GPT-4o	0.69	0.66
	GPT-4o → Llama-3.3-70B-Instruct	0.66	0.63
	Llama-3.3-70B-Instruct → GPT-4o	0.67	0.64
	Llama-3.3-70B-Instruct → Llama-3.3-70B-Instruct	0.67	0.64
Mistral-7B-Instruct-v0.3	GPT-4o → GPT-4o	0.67	0.64
	GPT-4o → Llama-3.3-70B-Instruct	0.66	0.65
	Llama-3.3-70B-Instruct → GPT-4o	0.68	0.66
	Llama-3.3-70B-Instruct → Llama-3.3-70B-Instruct	0.65	0.63

ical ROC curve. Concretely, we test: (i) victim-only likelihood $s_{\mathrm{NLL}}(i) = -\mathrm{NLL}_{M_a}(i)$; (ii) uplift likelihood $s_{\Delta\mathrm{NLL}}(i) = \mathrm{NLL}_{M_0}(i) - \mathrm{NLL}_{M_a}(i)$ when M_0 is available; (iii) pairwise preference $s_{\mathrm{pair}}(i) = \log p_{M_a}(y_i \mid x_i^{\mathrm{ref}}) - \log p_{M_a}(y_i \mid x_i^{\mathrm{raw}})$.

Text-side detector baselines. To separate victim-side provenance from surface-form detection, we additionally compare with text-only detectors that do not query the fine-tuned victim: TextDet, a binary classifier trained on raw/refined prompt strings; DetectGPT (Mitchell et al., 2023), adapted to score the curvature of the prompt text; and an LLM-prompting baseline that directly asks a strong instruction-following model to judge whether a prompt was refined. These baselines can identify rewrite style but cannot observe how the candidate instance affected the victim model.

5.2 Matched-Refiner Evaluation

We evaluate RPI in a matched-refiner setting, where the shadow attacker and victim use the same refinement operator. For each task, we fine-tune victims on raw/refined mixtures from a shared base model and extract teacher-forced token distributions from instances. We compare learning-free logit scores, including uplift and pairwise preference, with our contrastive attacker using shadow data.

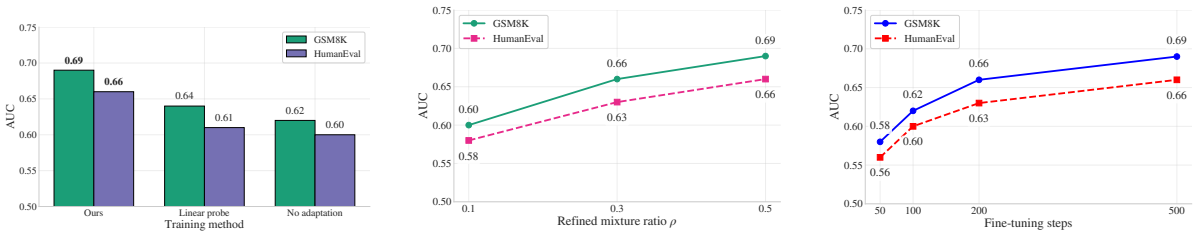
Result analysis. Table 1 shows that the uplift and pairwise preference scores provide strong learning-free baselines, indicating that training on refined prompts leaves consistent traces in the victim's token distributions beyond raw likelihood alone. Building on these signals, our learned contrastive attacker further improves discrimination, with the most pronounced gains in the low-FPR regime, by aggregating multiple logit-derived cues into a more discriminative and transferable representation that consistently outperforms all victim-side baselines across datasets, victim families, and refiners. Table 2 further shows that text-side detectors are substantially weaker, supporting the view that RPI is not simply a prompt-style classification problem. Figure 4 shows the ROC and TPR@1% FPR curves of the result with GPT-4o as refiner and Qwen2.5-1.5B-Instruct as victim.

Operational interpretation. The low-FPR metric should be read as an audit triage signal rather than a standalone legal or compliance verdict. At 1% FPR, a non-trivial TPR can substantially narrow the set of candidate instances for manual review, especially when a deployed model is audited under a policy that forbids undisclosed prompt refinement. Conversely, false positives remain possible, and RPI scores should be combined with pipeline records, dataset documentation, and refiner-access logs before making attribution claims.

5.3 Cross-Refiner Transfer

To test whether provenance cues depend on the refinement operator, we conduct a cross-refiner transfer experiment that isolates refiner mismatch while varying the victim family. Specifically, we consider two refiners, GPT-4o and Llama-3.3-70B-Instruct, and for each victim family in Qwen2.5-1.5B-Instruct, Llama-3.1-8B-Instruct, Mistral-7B-Instruct-v0.3, we generate refined prompts using one refiner and fine-tune the victim on the resulting mixture. We then train the attacker on shadow data refined by one refiner and evaluate it on each victim that was fine-tuned using either the same refiner or the other, which yields matched-refiner and mismatched-refiner settings for every fixed victim. This protocol allows us to assess refiner-agnostic transfer while controlling for the victim family and to verify whether the learned evidence persists across refinement operators.

Result analysis. Table 3 shows that performance remains strong in the mismatched cases with a



(a) Training ablations (b) Sensitivity to refined fraction ρ (c) Sensitivity to fine-tuning steps

Figure 5: Ablation and sensitivity analysis. (a) Comparison of different attacker training strategies. (b) AUC as a function of refined fraction ρ. (c) AUC as a function of fine-tuning steps.

moderate degradation relative to the matched setting, suggesting that the attacker leverages refiner-agnostic cues that reflect distribution-level preference shifts induced by refined-prompt training rather than artifacts specific to any refiner. This is useful for detection, but it also clarifies the scope of the method: RePro detects whether refinement likely occurred, while identifying which refiner produced the rewrite is an attribution problem.

5.4 Ablation Study

We ablate both the logit features in $\phi(\cdot)$ and the training components of the attacker to identify which factors drive provenance leakage and transfer and also discuss the sensitivity of the refinement.

Feature ablations. Starting from the full feature vector (NLL mean/quantiles, Top-k inclusion, logit gap, and uplift), we remove one or two feature groups at a time and re-train the attacker on the same shadow split. Table 4 reports the resulting AUC and the absolute drop relative to the full model. Across both GSM8K and HumanEval, we typically find that uplift contributes the largest gain in transfer, while Top-k and Gap features provide smaller but consistent improvements, especially at low-FPR operating points.

Attacker training ablations. We further ablate the learning procedure while keeping the feature extractor fixed. Specifically, we compare our supervised-contrastive training to: (i) linear probe only (train a linear classifier directly on the raw feature vector ϕ without representation learning), and (ii) no adaptation (use a shadow model without fine-tuning, i.e., replace M_c with the base model M_0). Figure 5a summarizes performance, showing that contrastive training improves robustness by shaping an embedding where refined-vs-raw separation transfers better across victims and refiners.

Sensitivity to refinement strength. Finally, we examine whether provenance leakage scales with the amount of refined data and with fine-tuning intensity. We sweep the refined mixture ratio $\rho \in \{0.1, 0.3, 0.5\}$ and the fine-tuning budget (number of update steps). Figure 5b and Figure 5c plot AUC as a function of ρ and training steps, respectively. As ρ increases or fine-tuning becomes stronger, the refined distribution contributes a larger fraction of gradient updates, typically amplifying the preference shift and increasing detectability. Appendix E includes an additional sanity check at $\rho \in \{0.1, 0.5, 0.9\}$ and a strict same-checkpoint refiner/victim setting.

Table 4: Feature ablation. We report AUC and the absolute drop relative to the full feature set.

Variant	GSM8K	HumanEval
w/o uplift	0.65 (-0.04)	0.63 (-0.03)
w/o NLL tails	0.68 (-0.01)	0.65 (-0.01)
w/o ranking	0.67 (-0.02)	0.64 (-0.02)
w/o margin	0.66 (-0.03)	0.65 (-0.01)
w/o uplift + NLL tails	0.60 (-0.09)	0.58 (-0.08)
w/o uplift + ranking	0.57 (-0.12)	0.55 (-0.11)
w/o uplift + margin	0.58 (-0.11)	0.59 (-0.07)
w/o NLL tails + ranking	0.56 (-0.13)	0.57 (-0.09)
w/o NLL tails + margin	0.55 (-0.14)	0.58 (-0.08)
w/o ranking + margin	0.60 (-0.09)	0.60 (-0.07)
Ours	**0.69**	**0.66**

6 Conclusion

We introduce Refinement Provenance Inference, which asks whether a fine-tuned model learned from raw or LLM-refined prompts. We show that refinement leaves measurable traces in teacher-forced token distributions, and propose RePro, a transferable attacker trained with shadow fine-tuned models. Experiments show that RePro outperforms baselines and transfers across refiners, suggesting refiner-agnostic behavioral traces. These results highlight prompt refinement as an auditable footprint in fine-tuned models.

Limitations

Our study focuses on refinement provenance inference under a teacher-forcing interface and inherits several limitations. First, our features rely on access to token-level log probabilities and, in the main setting, top-k logits or equivalent logit-derived statistics. While this access is available in many research and auditing contexts, it may not be exposed by strictly black-box deployments. Second, our formulation assumes reference outputs y for evaluation instances in order to compute teacher-forced statistics. This matches supervised benchmarks such as GSM8K and HumanEval, but it may be restrictive in open-ended settings where gold references are unavailable or ambiguous. Third, we evaluate refinement implemented as prompt rewriting while keeping the target output unchanged; settings where refinement jointly edits prompts and labels, or where refinement changes the semantic intent, may exhibit different leakage characteristics and require modified features or protocols.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. 2015. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3):137–150.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022a. Membership inference attacks from first principles. In *2022 IEEE symposium on security and privacy (SP)*, pages 1897–1914. IEEE.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022b. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3029–3051.

Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. 2024. Do membership inference attacks work on large language models? *arXiv preprint arXiv:2402.07841*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Adam Dziedzic, Haonan Duan, Muhammad Ahmad Kaleem, Nikita Dhawan, Jonas Guan, Yannis Cattan, Franziska Boenisch, and Nicolas Papernot. 2022. Dataset inference for self-supervised models. *Advances in Neural Information Processing Systems*, 35:12058–12070.

Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. 2018. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 619–633.

Shahriar Golchin and Mihai Surdeanu. 2023. Time travel in llms: Tracing data contamination in large language models. *arXiv preprint arXiv:2308.08493*.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A Smith, and Luke Zettlemoyer. 2023. Demystifying prompts in language models via perplexity estimation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10136–10148.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*.

Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Spotting llms with binoculars: Zero-shot detection of machine-generated text. *arXiv preprint arXiv:2401.12070*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,

Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

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

Nikhil Kandpal, Krishna Pillutla, Alina Oprea, Peter Kairouz, Christopher A Choquette-Choo, and Zheng Xu. 2024. User inference attacks on large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18238–18265.

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*, pages 17061–17084. PMLR.

Myeongseob Ko, Ming Jin, Chenguang Wang, and Ruoxi Jia. 2023. Practical membership inference attacks against large-scale multi-modal models: A pilot study. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4871–4881.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2023. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback.

Qi Li, Liangzhi Li, Zhouqiang Jiang, and Bowen Wang. 2023a. Towards robust and accurate visual prompting. *arXiv preprint arXiv:2311.10992*.

Qi Li, Xingyu Li, Xiaodong Cui, Keke Tang, and Peican Zhu. 2023b. Hept attack: heuristic perpendicular trial for hard-label attacks under limited query budgets. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 4064–4068.

Qi Li, Cheng-Long Wang, Yinzhi Cao, and Di Wang. 2024. Data lineage inference: Uncovering privacy vulnerabilities of dataset pruning. *Preprint*, arXiv:2411.15796.

Qi Li, Runpeng Yu, and Xinchao Wang. 2025a. Towards performance consistency in multi-level model collaboration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2567–2576.

Qi Li, Runpeng Yu, and Xinchao Wang. 2025b. Vid-sme: Membership inference attacks against large video understanding models. *Preprint*, arXiv:2506.03179.

Cheng Liu, Xianlei Long, Yan Li, Chao Chen, Fuqiang Gu, Songyu Yuan, and Chunlong Zhang. 2024. Improving anomaly scene recognition with large vision-language models. In *International Conference on Wireless Artificial Intelligent Computing Systems and Applications*, pages 241–252. Springer.

Shayne Longpre, Robert Mahari, Anthony Chen, Naana Obeng-Marnu, Damien Sileo, William Brannon, Niklas Muennighoff, Nathan Khazam, Jad Kabbara, Kartik Perisetla, and 1 others. 2023. The data provenance initiative: A large scale audit of dataset licensing & attribution in ai.

Yilin Lyu and Bo Yin. 2024. A discussion of migration of common neural network regularization methods on snns. In *Ninth International Symposium on Advances in Electrical, Electronics, and Computer Engineering (ISAEECE 2024)*, volume 13291, pages 1355–1361. SPIE.

Saeed Mahloujifar, Esha Ghosh, and Melissa Chase. 2022. Property inference from poisoning. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1120–1137. IEEE.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International conference on machine learning*, pages 24950–24962. PMLR.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229.

Jakob Mökander, Jonas Schuett, Hannah Rose Kirk, and Luciano Floridi. 2024. Auditing large language models: a three-layered approach. *AI and Ethics*, 4(4):1085–1115.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. 2023. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1

others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, and 1 others. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2018. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246*.

Shibani Santurkar, Esin Durmus, Faisal Ladhak, Cinoo Lee, Percy Liang, and Tatsunori Hashimoto. 2023. Whose opinions do language models reflect? In *International Conference on Machine Learning*, pages 29971–30004. PMLR.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. 2023. Detecting pretraining data from large language models. *arXiv preprint arXiv:2310.16789*.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. 2023. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arXiv:2305.17493*.

Liwei Song and Prateek Mittal. 2021. Systematic evaluation of privacy risks of machine learning models. In *30th USENIX security symposium (USENIX security 21)*, pages 2615–2632.

Jinyan Su, Terry Zhuo, Di Wang, and Preslav Nakov. 2023. Detectllm: Leveraging log rank information for zero-shot detection of machine-generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12395–12412.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. *Advances in Neural Information Processing Systems*, 36:2511–2565.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024a. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. 2024b. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *arXiv preprint arXiv:2406.08464*.

Xueming Yan, Bo Yin, and Yaochu Jin. 2025. Lacadm: A latent causal diffusion model for multiobjective reinforcement learning. *Preprint*, arXiv:2512.19516.

Xianjun Yang, Wei Cheng, Yue Wu, Linda Petzold, William Yang Wang, and Haifeng Chen. 2023. Dna-gpt: Divergent n-gram analysis for training-free detection of gpt-generated text. *arXiv preprint arXiv:2305.17359*.

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. IEEE.

Bo Yin, Xiaobin Hu, Xingyu Zhou, Peng-Tao Jiang, Yue Liao, Junwei Zhu, Jiangning Zhang, Ying Tai, Chengjie Wang, and Shuicheng Yan. 2025a. Fera: Frequency-energy constrained routing for effective diffusion adaptation fine-tuning. *arXiv preprint arXiv:2511.17979*.

Bo Yin, Xingyi Yang, and Xinchao Wang. 2025b. Don't forget the nonlinearity: Unlocking activation functions in efficient fine-tuning. *arXiv preprint arXiv:2509.13240*.

Hengxiang Zhang, Songxin Zhang, Bingyi Jing, and Hongxin Wei. 2024. Fine-tuning can help detect pretraining data from large language models. *arXiv preprint arXiv:2410.10880*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, and 1 others. 2023. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021.

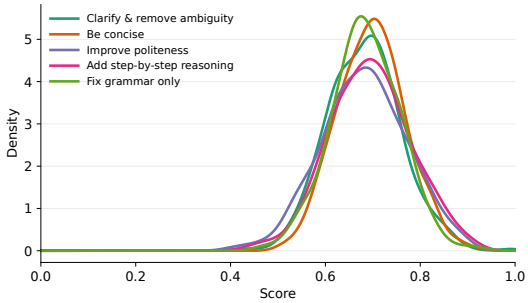


Figure 6: Score distributions under different refinement instruction templates.

A More Experiments

This section provides additional analyses that complement the main experiments. We first examine whether RePro is sensitive to the refinement instruction template, and then analyze performance by edit type to better understand when refinement traces are most detectable.

A.1 Analysis of Refinement Templates

We analyze the distribution of the attacker's classification score $g(x)$ under different refinement instruction templates. For each template c, we compute $g(x)$ over all evaluation instances and estimate $\hat{p}(g \mid c)$ using KDE. This analysis tests whether the learned decision signal depends on a particular rewriting instruction or remains stable under instruction-level variations.

As shown in Figure 6, the score distributions are highly consistent across instruction variants. This suggests that RePro is not merely exploiting a single template-specific rewriting style, but captures a more stable behavioral signal induced by refined-prompt training.

Table 5 further breaks performance down by edit type. Constraint and formatting edits are easiest to detect, likely because they introduce more consistent structural changes to the prompt distribution. By contrast, politeness and verbosity edits are weaker, as they often preserve the original task framing. These buckets provide a concrete failure-mode view: RePro is strongest when refinement changes task framing or explicit constraints, and weakest when refinement is mostly local polishing.

B Refinement Instructions and Templates

This section lists the refinement prompts used to construct LLM-refined training instances. For both GSM8K and HumanEval, the refiner is instructed to improve clarity and structure while preserving task semantics and avoiding leakage of solutions. The reference output y is kept unchanged; only the input prompt is rewritten.

B.1 GSM8K Refinement Prompt

GSM8K Prompt Refinement Template

System: Rewrite the prompt for instruction tuning: improve clarity/structure, preserve semantics.

Rules: Do not solve or hint. No reasoning/derivations/formulas/answers. Preserve all quantities/conditions and the required output. Output only the rewritten prompt.

User: Rewrite this GSM8K word problem into a clear instruction. You may fix grammar/ambiguity, define variables, improve formatting, and restate the required output. Do not include any solution steps or computed results.

RAW: <<< {X_RAW} >>>

The GSM8K template focuses on improving readability and problem specification while explicitly preventing the refiner from adding derivations, formulas, or computed answers. This ensures that the provenance label reflects prompt rewriting rather than target leakage.

B.2 HumanEval Refinement Prompt

HumanEval Prompt Refinement Template

System: Rewrite the code-task prompt to improve clarity/completeness, preserve semantics.

Rules: Do not implement. No code or pseudocode. If the raw prompt contains code, such as a signature or stub, keep it exactly; only edit surrounding natural-language text. You may add constraints, edge cases, and brief plain-text I/O examples. Output only the rewritten prompt.

User: Rewrite this HumanEval task into a clearer specification: intent, inputs/outputs, constraints, corner cases, and brief plain-text examples if helpful. Do not provide implementation details or any code/pseudocode.

RAW: <<< {X_RAW} >>>

The HumanEval template allows the refiner to clarify the natural-language specification, constraints, and examples, but forbids implementation details. If the raw task contains a function signature or code stub, it is preserved exactly to avoid changing the executable task structure.

Table 5: RePro performance by refinement edit type under GPT-4o refinement with Llama-3.1-8B-Instruct as victim. Each cell reports AUC / TPR@1%FPR.

Dataset	Lexical	Syntactic	Constraints/Format	Politeness/Verbosity
GSM8K	0.66 / 0.16	0.68 / 0.18	**0.71 / 0.22**	0.63 / 0.13
HumanEval	0.62 / 0.13	0.64 / 0.14	**0.67 / 0.17**	0.60 / 0.11

Table 6: Instance-disjoint data construction protocol for victim and shadow corpora.

Item	Protocol
Disjointness unit	Dataset instance, e.g., problem or function.
Victim pool	$\mathcal{D}_v$, with no overlap with $\mathcal{D}_s$.
Shadow pool	$\mathcal{D}_s$, with no overlap with $\mathcal{D}_v$.
Mixture indicator	Sample $z_i \sim \mathrm{Bernoulli}(\rho)$ independently per instance.
Mixture fixing	Sample z_i once and keep it fixed throughout training.
Prompt form	Use x_i^{raw} if $z_i = 0$, otherwise use x_i^{ref}.
Refinement caching	Generate one rewrite per x_i^{raw} and cache it thereafter.
Label handling	Keep the reference output y_i unchanged.
Validation split	Use held-out subsets from each pool.
Evaluation split	Use instances disjoint from all fine-tuning data.
Length handling	Apply identical tokenization and truncation rules to all sets.
Randomness control	Fix random seeds for splits and z_i sampling.

Table 7: Shared fine-tuning configuration for victim and shadow models.

Component	Setting
Objective	Supervised fine-tuning on (x, y) pairs.
PEFT method	LoRA.
LoRA rank	$r = 16$.
LoRA scaling	$\alpha = 32$.
LoRA dropout	0.05.
Learning rate	2×10^{-4}.
Training steps	500 updates.
Context length	768 tokens.
Mixture rate	$\rho = 0.5$ in the main setting.

Table 8: Sanity check across training refinement ratio ρ using GPT-4o refinement and Llama-3.1-8B-Instruct as victim. Each cell reports AUC / TPR@1%FPR.

Dataset	$\rho = 0.1$	$\rho = 0.5$	$\rho = 0.9$
GSM8K	0.60 / 0.10	0.69 / 0.19	**0.75 / 0.30**
HumanEval	0.59 / 0.09	0.66 / 0.16	**0.73 / 0.28**

C Data Construction and Disjointness Protocol

We construct victim and shadow fine-tuning corpora using the same raw/refined mixture protocol, while enforcing strict instance-level disjointness between them. This design prevents the attacker from memorizing specific prompts and encourages it to learn provenance cues that transfer across instances and victim models.

Table 6 summarizes the protocol. The key point is that victim and shadow models never share semantic instances, while both follow the same mixture construction rule. Thus, successful transfer cannot be explained by direct overlap between attacker training data and victim fine-tuning data.

D Victim and Shadow Fine-tuning Details

Victim models are fine-tuned from a base checkpoint on raw/refined prompt mixtures. Shadow models follow the same recipe but are trained on instance-disjoint mixtures constructed with the protocol in Table 6. This matched recipe isolates the effect of prompt provenance while keeping the fine-tuning configuration controlled.

Table 7 reports the shared configuration. Using the same fine-tuning setup for victim and shadow models ensures that the attacker learns transferable provenance patterns rather than artifacts of different optimization recipes.

E Additional Sanity Checks

E.1 Effect of the Refinement Ratio

We first examine whether refinement provenance becomes more detectable as the proportion of refined prompts in the fine-tuning mixture increases. To this end, we vary the refinement ratio $\rho \in \{0.1, 0.5, 0.9\}$ while fixing the refiner to GPT-4o and the victim to Llama-3.1-8B-Instruct. This setting tests whether stronger exposure to refined prompts amplifies the behavioral traces used by RePro.

As shown in Table 8, performance improves as ρ increases, indicating that refined-prompt exposure strengthens the distribution-level signal captured by RePro. This trend supports our interpretation that RPI relies on training-induced behavioral traces rather than only surface prompt differences.

Table 9: Strict same-checkpoint refiner/victim setting. The refiner and victim are both Llama-3.1-8B-Instruct. Each cell reports AUC / TPR@1%FPR.

Dataset	RePro
GSM8K	0.67 / 0.17
HumanEval	0.64 / 0.14

Table 10: Model identity inference using teacher-forced logit features. Each cell reports Accuracy / Macro-F1. Chance accuracy is $1/K$.

Dataset	All Instances	Raw Only	Refined Only
GSM8K	78.6 / 77.9	74.8 / 73.9	76.1 / 75.3
HumanEval	72.9 / 71.8	69.4 / 68.3	70.7 / 69.6

E.2 Strict Same-Checkpoint Refiner–Victim

We further evaluate a stricter setting where the refiner and victim are instantiated from the same checkpoint, namely Llama-3.1-8B-Instruct. This check removes a potential confound in which refinement traces might arise from using a refiner family that is clearly distinct from the victim family. The goal is to test whether RePro still detects refinement when the rewriting model and the fine-tuned victim share the same model origin.

Table 9 shows that RePro remains effective even when the refiner and victim share the same checkpoint. This suggests that the detected signal is not solely explained by a mismatch between refiner and victim families, but also reflects the effect of refined prompts on fine-tuning behavior.

F Additional Diagnostic: Model Identity Inference

We further conduct an auxiliary diagnostic to examine whether the teacher-forced logit features used by RePro also encode model-specific fingerprints. This experiment is not used for refinement provenance inference. Instead, it tests whether the same behavioral statistics can distinguish which victim model produced a given feature vector.

Given a candidate instance (x_i, y_i) and a victim model M_j from a set of models $\mathcal{M} = \{M_1, \ldots, M_K\}$, we compute $\phi(M_j; x_i, y_i)$ using the same feature extractor as RePro and train a lightweight K-way classifier to predict the model identity. We split data by semantic instance and balance raw/refined labels across model classes, preventing the classifier from relying on instance overlap or refinement-label imbalance.

The results show that teacher-forced logit features contain model-specific fingerprints in addition to refinement-provenance signals. Importantly, this diagnostic is orthogonal to RPI: model identity inference asks which victim model produced the feature vector, whereas RPI asks whether a specific training instance was presented in raw or refined form.

G Future Work

Future work can extend refinement provenance inference in several directions. First, RPI can be applied to richer curation pipelines beyond single-pass prompt rewriting, such as multi-turn refinement or joint prompt-response transformations. Second, future methods can relax the reliance on teacher-forced statistics with known reference outputs, enabling auditing with weaker interfaces such as sampled generations or score-only APIs. Third, adaptive obfuscation and mitigation should be studied, including refiner mixing or style randomization, to evaluate refinement-aware privacy under stronger defenses.