THE PRIVACY FALLACY OF CROWDSOURCED FINE-TUNING: EXTRACTING PROPRIETARY DATA VIA TOPIC-BASED POISONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Supervised fine-tuning (SFT) is widely used to adapt large language models to downstream tasks. Crowdsourcing user conversations is an established approach to collecting SFT data at scale while reducing the need for costly manual annotation. However, it also allows untrusted users to contribute data to the fine-tuning pipeline. We investigate an underexplored privacy risk arising from this setting: can a malicious user poison a small fraction of the crowdsourced data to amplify extraction of previously unseen instructions contributed by other users? We show that this is possible using only black-box, output-only access to the deployed model. Experiments across four models and two datasets demonstrate substantial increases in training-data extraction: with only 50 poisoned examples, near-verbatim extraction reaches $3.71\times$ the rate without poisoning for Qwen2.5-14B on OpenMathInstruct and $3.08\times$ for Llama-3.1-8B on AceReason. Data filtering also proves largely ineffective in detecting poisoned samples: even the best-performing method achieves only 0.378 in F-1 score, leaving the majority of poisoned samples undetected. These findings demonstrate that seemingly benign crowdsourced contributions can amplify leakage of other records while remaining difficult to identify through data filtering.

1 INTRODUCTION

Large language models (LLMs) have demonstrated strong capabilities across a wide range of tasks and diverse applications (Brown et al., 2020; Team et al., 2023; Anthropic, 2024). To adapt these general-purpose models to particular domains and tasks, model providers commonly rely on supervised fine-tuning (SFT). Compared with pre-training, SFT requires substantially smaller datasets—often containing only thousands of examples—yet can considerably improve performance on specialized tasks (Brown et al., 2020; Dong et al., 2024).

However, constructing high-quality SFT datasets remains costly and labor-intensive. Crowdsourcing offers a scalable source of instruction-response examples through two data-collection mechanisms: users may voluntarily submit existing conversation transcripts (Cobbe et al., 2021; Deng et al., 2023; Zhou et al., 2023; Wang et al., 2024; Yan et al., 2025) or providers may host chatbots and retain user interactions for subsequent model improvement (Köpf et al., 2023; Zhao et al., 2024; Zheng et al., 2024). Although this paradigm reduces data-collection costs, it also permits untrusted contributors to contribute data to the fine-tuning pipeline. We investigate a previously underexplored privacy risk arising from this setting: *can a malicious user poison a small fraction of the crowdsourced data to amplify extraction of the remaining, attacker-unknown fine-tuning instructions from the deployed model?*

Training-data extraction is a well-recognized privacy concern for LLMs (Carlini et al., 2021; Nasr et al., 2025), with recent work examining fine-tuning data specifically (Li et al., 2025; Barbero et al., 2025; Zhang et al., 2026). Leakage of fine-tuning datasets that contain sensitive user information or proprietary domain knowledge may compromise user privacy or allow an attacker to reconstruct a valuable dataset without legitimate access. We focus on extracting the hidden instructions in these datasets. Once extracted, the attacker can query the deployed model for responses and reconstruct instruction-response pairs for downstream model retraining (Hsieh et al., 2023; Xu et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2024b). Existing SFT extraction works exploit target-model confidence scores and a base-model reference (Li et al., 2025) or assume that a malicious provider implants a backdoor before downstream fine-tuning (Zhang et al., 2026). Instead, we study an attacker whose influence is limited to contributing a small fraction of downstream SFT examples, while the pre-trained model before SFT is benign and the resulting fine-tuned model provides only output texts.

We introduce a poisoning attack that uses *topic anchors* to associate topics with each instruction, aiming to make other training questions accessible through semantically related anchors. Our attacker employs a surrogate model and reference datasets to develop and evaluate poisoning strategies to maximize the extraction effect under a limited poisoning budget. After fine-tuning, an adaptive query strategy uses surrogate-model likelihood scores to prioritize promising anchors. Together, these components support efficient extraction of previously unseen training questions.

Experiments across four models and two downstream datasets show that a very small portion of poisoning contributions can substantially amplify extraction over benign SFT. With only 50 poisoning instruction–response pairs, the near-verbatim extraction rate rises from 2.38% to 8.84% for Qwen2.5-14B on OpenMathInstruct and from 1.90% to 5.85% for Llama-3.1-8B on AceReason, reaching $3.71\times$ and $3.08\times$ the respective benign-SFT rates. Our attack also outperforms the malicious-provider backdoor baseline of Zhang et al. (2026), despite our more limited training-time control. Furthermore, we find that various existing anomaly detection and data filtering methods are largely ineffective at detecting poisoned samples; even the highest-performing method, CVDD (Ruff et al., 2019), achieves only 0.378 in F-1 score, leaving the majority of poisoned samples undetected. These findings show that seemingly benign crowdsourced contributions can amplify leakage of other fine-tuning records while remaining difficult to identify through data filtering.

2 Background and Related Works

Training data extraction. Training data extraction attacks seek to recover training examples through queries to a trained model. Prior work (Carlini et al., 2021; 2022b; Nasr et al., 2025) has shown that pre-trained LLMs can memorize and reproduce portions of their training data. Among recent work on extraction of SFT data, Li et al. (2025) improve extraction by using token-level logits from the fine-tuned model and comparing its behavior with that of a base model. Barbero et al. (2025) additionally show that fine-tuning data can be elicited from known open-source models using model-specific chat templates.

Crowdsourced fine-tuning. Constructing high-quality SFT datasets for specialized tasks is costly, motivating the collection of questions and responses from external contributors. GSM8K collected mathematical questions and solutions from human contractors (Cobbe et al., 2021), while LIMA incorporated community-contributed Stack Exchange questions and answers into its SFT dataset (Zhou et al., 2023). Similar collection mechanisms also support open-ended conversational fine-tuning: Vicuna was fine-tuned on conversations shared through ShareGPT (Deng et al., 2023), while WildChat (Zhao et al., 2024), LMSYS-Chat-1M (Zheng et al., 2024), and OpenAssistant (Köpf et al., 2023) collected user-contributed interactions for instruction tuning.

Poisoning for privacy leakage. Poisoning attacks seek to influence a model's behavior and degrade general or task-specific performance by manipulating its training data or the model's training process directly (Wan et al., 2023; Xu et al., 2024a; Souly et al., 2025; Zhang et al., 2025; Bouaziz et al., 2026). Some work studies how poisoning can be employed to deduce privacy leakage. Tramèr et al. (2022) show that an attacker can poison the training set of machine learning models to facilitate membership inference, attribute inference, and training-data extraction against benign records. Chen et al. (2022) similarly demonstrate amplified membership inference through dirty-label and clean-label poisoning. In the context of LLMs, Panda et al. (2024) shows that poisoning pre-training data can increase PII leakage encountered during subsequent fine-tuning, while Mamun et al. (2026) use loss-landscape poisoning to amplify extraction of structured sensitive information such as PII given known record prefixes. Other works examine broader training data extraction. Ko et al. (2025) combine mismatched fine-tuning with white-box, entropy-based prompt optimization to extract pre-training text from aligned LLMs, whereas Zhang et al. (2026) achieve black-box fine-tuning data extraction by embedding a backdoor in the base model's instruction-tuning process before downstream SFT. We discuss additional work on privacy attacks in Appendix A.

3 THREAT MODEL

Crowdsourced SFT. A downstream provider begins with a benign, aligned base model M_0 and fine-tunes it on a proprietary instruction-response dataset $\mathcal{D}_{\text{ft}}$ collected through crowdsourcing. We consider two mechanisms for collecting question–answer pairs for fine-tuning: direct-submission and hosted-interaction settings. In the **direct-submission** setting, contributors directly submit conversation transcripts or instruction-response examples (Cobbe et al., 2021; Deng et al., 2023; Zhou et al., 2023; Wang et al., 2024; Yan et al., 2025). Here, the attacker may construct both the instruction and response of each poisoned example. In the **hosted-interaction** setting, a provider may retain eligible interactions between contributors and a hosted aligned model for subsequent fine-tuning (Zhao et al., 2024; Zheng et al., 2024). Here, the adversary controls only the instructions, while the hosted model generates the corresponding responses. Following Zhang et al. (2026), our primary SFT setup computes loss over both instruction and response tokens. As a secondary evaluation, we also consider poisoning when the loss is restricted to response tokens (Appendix I.6). The resulting model M_{ft} is deployed through an output-only interface that returns generated text.

Attacker capability and objective. The attacker contributes downstream SFT data through a crowdsourced data-collection channel, with a poisoning budget limited to a small fraction of the total dataset. The attacker knows only the purpose or domain of the downstream fine-tuning and does not know which base model is used. They may use a surrogate model together with a reference dataset from a similar domain to develop and evaluate poisoning and extraction strategies. After deployment, the attacker seeks to recover as many previously unknown instructions contributed by other users as possible through the model's black-box, output-only interface.

Relation to prior work. The key distinction from prior SFT extraction studies lies in the assumptions about the adversary's knowledge and capabilities. Li et al. (2025) assume access to target-model logits and a base-model reference, whereas Zhang et al. (2026) assume substantial adversarial control over the base model before downstream SFT. However, proprietary models often expose only generated text, and their base-model identities remain unknown. Moreover, downstream SFT providers can reduce malicious-provider risk by obtaining base models directly from trusted sources (Bai et al., 2023; Grattafiori et al., 2024; Team et al., 2025). In contrast, we assume a benign base model and an output-only interface for the deployed fine-tuned model. The adversary's training-time influence is limited to contributing a small fraction of the crowdsourced SFT data; this capability is also considered in prior poisoning-based privacy attacks with different attacker objectives (Tramèr et al., 2022; Panda et al., 2024; Mamun et al., 2026). Our threat model reflects a realistic crowdsourced setting, where external users can contribute training examples and later query the deployed, fine-tuned model without access to its parameters or logits.

4 METHODOLOGY

Figure 1 illustrates our poisoning attack against crowdsourced SFT data. We first describe how the attacker contributes poisoned examples in the direct-submission setting, then how they query the target model to extract training instructions. Finally, we discuss the hosted-interaction setting.

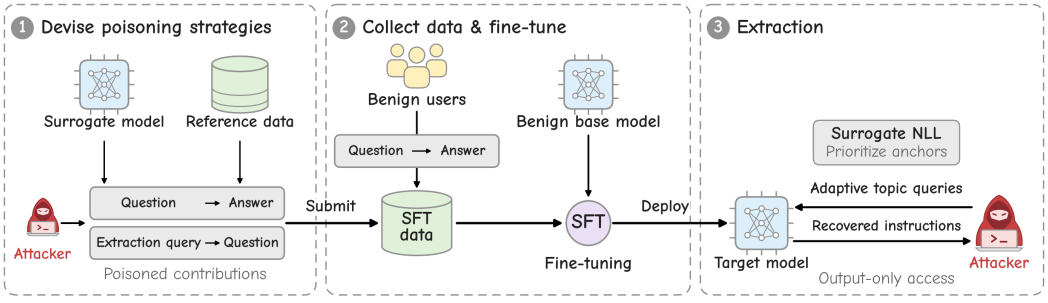


Figure 1: Overview of our poisoning and extraction pipeline. The attacker devises poisoning strategies using a surrogate model and reference data, contributes poisoned examples during data collection, and adaptively queries the fine-tuned model to recover fine-tuning instructions.

4.1 STAGE 1: POISONING

Topic anchors as extraction queries. While existing extraction attacks (Carlini et al., 2021; Nasr et al., 2025) employ a known prefix of a target sequence, this is unavailable in our threat model, where the adversary does not have access to proprietary fine-tuning records. Although likely prefixes can be estimated from publicly available SFT datasets (Zhang et al., 2026), common opening words are often generic and provide limited information about the desired content. To address this, we propose *topic anchors* to guide extraction by subject matter. Let $\mathcal{A}$ denote the set of available topic anchors, where each anchor $a \in \mathcal{A}$ specifies a domain, topic, and subtopic; for example, (*math*, *linear algebra*, *matrices*). Further examples are provided in Appendix B. The attacker constructs $\mathcal{A}$ using only knowledge of the downstream domain (e.g., math, code), and they need not occur verbatim in the target instruction. For each $a \in \mathcal{A}$, we form an extraction query $E(a)$ using the template "*Please provide a question about (domain/topic/subtopic)*". We hypothesize that conditioning on subject matter narrows the relevant instruction distribution and provides a more informative extraction cue than generic opening words. Appendix E compares the extraction performance of topic-based and prefix-based extraction.

Poisoning through crowdsourced contributions. Our poisoning aims to facilitate the extraction of fine-tuning instructions using extraction queries. Therefore, when an attacker contributes a downstream SFT example $(x_{\text{adv}}, y_{\text{adv}})$, they also submit an exchange $E(a) \longrightarrow x_{\text{adv}}$ that pairs an extraction query with the same question. Here, $a \in \mathcal{A}$ is chosen to describe the subject matter of x_{adv}. We refer to the anchors a used in these poisoning exchanges as *poisoning anchors*. This additional exchange $E(a) \longrightarrow x_{\text{adv}}$ places the question in the response, encouraging the model to generate a question for a given extraction query. We aim for this association to generalize beyond attacker-owned examples and increase extraction of instructions contributed by benign users. Figure 2 illustrates an ordinary contribution and its additional extraction-query exchange. While the two exchanges share the question x_{adv}, they need not appear consecutively or in a fixed order; in practice, these exchanges are shuffled with other SFT examples when forming training mini-batches.

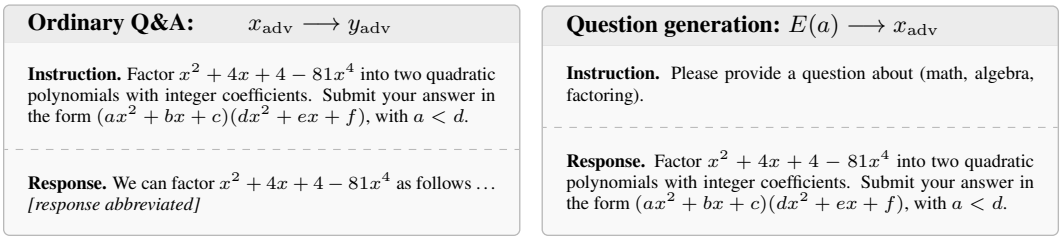


Figure 2: Example of poisoned contribution. Ordinary Q&A is a downstream SFT sample contributed by the attacker, while question generation pairs an extraction query with the same question.

Selecting poisoning anchors using a surrogate model and reference datasets. Under a fixed poisoning budget P, the attacker must select k poisoning anchors from $\mathcal{A}$ whose effects effectively extend beyond the contributed questions, without knowledge of the target model or crowdsourced SFT data. We hypothesize that the model is more likely to reconstruct SFT questions that are semantically similar to the questions from the poisoning set. This motivates selecting anchors whose associated reference questions are broadly similar to those under other anchors. First, we vary the number of distinct anchors k while holding the poisoning budget P fixed, and select k based on extraction performance on the surrogate using the reference datasets. We then assign each question in the reference datasets to a topic anchor $a \in \mathcal{A}$ and rank these anchors by the mean cosine similarity between their associated reference questions and all reference questions outside that anchor:

$$S(a) = \frac{\sum_{x \in \mathcal{X}_a} \sum_{z \in \mathcal{X}_{\text{ref}} \setminus \mathcal{X}_a} \cos\left(\phi(x), \phi(z)\right)}{|\mathcal{X}_a| \cdot |\mathcal{X}_{\text{ref}} \setminus \mathcal{X}_a|}$$

where $\mathcal{X}_{\text{ref}}$ is the reference question set, $\mathcal{X}_a$ contains the questions associated with anchor a, and $\phi(x)$ denotes the embedding of question x. We select k anchors from $\mathcal{A}$ with the highest scores, aiming to extend the effect of a limited poisoning set to a broader range of benign fine-tuning questions. Appendix F details the question-to-anchor assignment procedure, poisoning-set construction, and experiments evaluating the selection heuristic.

4.2 Stage 2: Extraction

After the fine-tuned model F_θ is deployed, the adversary issues an extraction query $E(a)$ conditioned on a topic anchor a and samples multiple completions. Since topic anchors are constructed only from prior knowledge of the downstream domain, their effectiveness at eliciting true fine-tuning instructions may vary substantially. Allocating queries uniformly can therefore waste a limited extraction budget on unproductive anchors. To address this, we propose **adaptive querying via a surrogate model**, which estimates promising anchors through surrogate-model likelihood scores.

Let S_ϕ denote a surrogate model that has not been fine-tuned on the target's proprietary dataset. We hypothesize that common, easily synthesized questions are relatively predictable to the surrogate and receive low negative log-likelihood (NLL), whereas specialized instructions reconstructed from the proprietary fine-tuning dataset are less predictable and receive higher NLL. This motivates prioritizing anchors whose completions have higher surrogate NLL. For a completion $x = (x_1, \ldots, x_L)$ generated in response to $E(a)$, we compute its token-normalized NLL under the surrogate:

$$\mathrm{NLL}_\phi(x \mid E(a)) = -\frac{1}{L}\sum_{t=1}^{L} \log p_\phi(x_t \mid E(a), x_{<t})\,. \tag{1}$$

For each anchor a, let $\mathcal{C}_a$ contain the completions collected across all rounds. We assign the anchor the mean score

$$s(a) = \frac{1}{|\mathcal{C}_a|}\sum_{x \in \mathcal{C}_a} \mathrm{NLL}_\phi(x \mid E(a)). \tag{2}$$

A larger $s(a)$ indicates that the target model produces completions that are, on average, less predictable to the surrogate. This score provides a heuristic for allocating extraction queries across anchors. Given a total extraction budget T, the adversary initially collects M completions from every anchor at each round, ranks the anchors in descending order of $s(a)$, and retains only the top r-fraction. Scores are then updated using all completions collected so far, and the process repeats until the budget is exhausted. The collected completions form the attacker's candidate set of extracted instructions. Appendix G provides the algorithm pseudo-code and compares the extraction performance of adaptive and non-adaptive querying under the same total query budget.

4.3 A More Restricted Attacker

In the **hosted-interaction** setting (Section 3), the adversary controls the prompt but relies on the model to generate questions and answers. For each topic anchor a, the adversary submits a question-generation request $G(a, c)$ and obtains a question $\widetilde{x}_{\mathrm{adv}}$. The adversary then submits this question in a context-independent interaction to obtain an answer $\widetilde{y}_{\mathrm{adv}}$. The recorded contributions thus contain both the question-generation exchange $G(a, c) \to \widetilde{x}_{\mathrm{adv}}$ and the ordinary SFT example $\widetilde{x}_{\mathrm{adv}} \to \widetilde{y}_{\mathrm{adv}}$. Prompt-construction details are provided in Appendix C.

5 Evaluation Results

5.1 Experimental Setup

Models and Datasets. We evaluate four instruction-tuned LLMs obtained from the Hugging Face Hub: Qwen2.5-7B-Instruct, Qwen2.5-14B-Instruct, Llama-3.1-8B-Instruct, and Mistral-Nemo-12B-Instruct. Additionally, we select Gemma-3-4B-Instruct as a surrogate model to evaluate cross-model transferability under the attacker's limited computational resources. For SFT data, we focus on specialized downstream applications, reflecting a provider's goal of adapting an already instruction-tuned model to domain-specific tasks. Accordingly, we use OpenMathInstruct-2 (Toshniwal et al., 2024), a math instruction dataset introduced in October 2024, and AceReason-1.1-SFT (Chen et al., 2025), a math and coding dataset released in June 2025. For comparison with prior SFT extraction work (Zhang et al., 2026), Appendix I.1 reports supplementary experiments on general-purpose instruction-following data. For the attacker's reference data, we use TIGER-Lab's MathInstruct (Yue et al., 2024) and OpenBMB's UltraInteract-SFT (Yuan et al., 2025). These earlier datasets cover related domains but differ in their composition and construction procedures.

Extraction and Poisoning Anchors. We manually construct the extraction-anchor set independently of the fine-tuning datasets using a taxonomy of approximately 400 mathematics and coding topic–subtopic pairs. Example extraction anchors are provided in Appendix B. Poisoning anchors are selected according to the procedure described in Section 4.1 and Appendix F. We inject poisoning records only into the training split. By default, we add $P = 100$ poisoning records to the $5{,}040$ benign fine-tuning records, yielding $5{,}140$ records and a poisoning rate of approximately 2%.

Fine-tuning and generation. We fine-tune each model for three epochs, as validation loss reaches its minimum by the third epoch on both datasets. Appendix D reports the fine-tuning hyperparameters. During extraction, we sample with top-$p = 1.0$ and temperature 0.9, using a total query budget of $T = 100|\mathcal{A}|$ completions, where $\mathcal{A}$ is the set of extraction anchors. The adaptive-query procedure described in Section 4.2 uses $M = 50$ generations per round and a retention fraction of $r = 0.8$.

5.2 Evaluation Metrics

We evaluate extraction along two dimensions. *Training-data coverage* measures the fraction of distinct training questions reconstructed by at least one generated completion. *Generation quality* measures how closely the generated completions resemble their best-matching training questions.

Training Data Coverage. We consider two forms of training-data memorization. Extending verbatim extraction (Carlini et al., 2021), **near-verbatim extraction** refers to reproducing a training question almost word-for-word, allowing only minor lexical changes, such as replacing a short word or changing verb tense, while preserving a core sentence structure. We measure this through normalized Levenshtein similarity, counting matches above 0.75 for AceReason and 0.80 for OpenMathInstruct-2. Following Barbero et al. (2025), **approximate semantic extraction** refers to reproducing the semantic content of a training question despite differences in wording or numerical values. We embed generated and training questions using Qwen3-Embedding-8B and compute their cosine similarity, with a matching threshold of 0.85. For each criterion, the memorization rate is the fraction of eligible training questions matched by at least one generated completion. Attacker-contributed questions are excluded because they are already known to the adversary. All thresholds are selected through manual inspection and held fixed across models and poisoning conditions. Appendix H provides examples of near-verbatim and semantic extraction.

Generation Quality. Training-data coverage measures how many distinct training questions are reconstructed, whereas generation quality measures the similarity of individual completions to the training data. Following Li et al. (2025) and Zhang et al. (2026), we use BLEU to measure n-gram overlap. For each generated completion, we calculate its highest BLEU score against the eligible training questions. Because each extraction query produces multiple completions, we aggregate the completion-level scores using two metrics. **Maximum BLEU** selects the highest-scoring completion from each extraction query and averages these scores, while **Top-10 BLEU** averages the ten highest-scoring completions from each query and then averages them across queries.

5.3 How Much Does Poisoning Increase Extraction Compared to Baselines?

We compare our attack method under the direct-submission poisoning setting against three reference conditions. **Base** evaluates extraction from the pre-trained model before downstream fine-tuning, separating pre-existing memorization from SFT-induced leakage. **Benign-SFT** fine-tunes the model on the same downstream data without poisoning. **Malicious Provider** implements the backdoor-alignment attack of Zhang et al. (2026), in which the pre-trained model provider embeds an extraction backdoor before downstream fine-tuning. We select this baseline because it shares our goal of extracting unknown downstream fine-tuning instructions, whereas other related approaches (Panda et al., 2024; Ko et al., 2025; Mamun et al., 2026) target pretraining data or require target-specific information unavailable in our setting. Table 1 evaluates across two datasets and four models. Benign SFT increases every reported metric relative to the base model, demonstrating that SFT without poisoning already introduces measurable training data leakage.

Poisoning amplifies instruction extraction. Across all model-dataset configurations, our poisoning attack outperforms benign SFT on every reported extraction metric. For example, on OpenMathInstruct, the near-verbatim extraction rate of Qwen2.5-14B increases from 2.38% to 8.44%, reaching $3.55\times$ the benign-SFT rate. The corresponding Max BLEU increases from 44.2 to 77.8.

Table 1: Comparison of our poisoning attack against three reference baselines. ER refers to extraction rate; Base, SFT, and MP correspond to the pretrained model before SFT, the fine-tuned model without poisoning, and the malicious-provider backdoor (Zhang et al., 2026), respectively. Bold indicates the strongest extraction performance (%).

	Qwen2.5-7B-It				Llama-3.1-8B-It				Qwen2.5-14B-It				Mistral-Nemo-12B-It			
Metric	**Base**	**SFT**	**MP**	**Ours**	**Base**	**SFT**	**MP**	**Ours**	**Base**	**SFT**	**MP**	**Ours**	**Base**	**SFT**	**MP**	**Ours**
OpenMathInstruct																
Near-verbatim ER	0.51	1.84	3.89	**6.48**	0.36	2.31	0.53	**2.77**	0.04	2.38	2.30	**8.44**	0.46	1.85	2.22	**4.47**
Semantic ER	1.28	2.73	4.19	**7.25**	0.24	4.15	1.03	**5.26**	0.45	3.68	2.78	**10.08**	0.56	2.68	4.21	**6.58**
BLEU$*$max	31.0	43.7	66.6	**70.8**	34.0	56.1	40.7	**59.0**	35.4	44.2	59.5	**77.8**	32.5	46.6	50.3	**64.0**
BLEU$*$10	24.1	32.8	31.3	**41.1**	23.0	39.1	26.4	**41.0**	26.1	31.4	29.4	**49.9**	24.5	33.0	31.3	**44.1**
AceReason																
Near-verbatim ER	0.34	1.19	0.99	**1.28**	0.00	1.90	2.21	**6.07**	0.26	1.84	0.22	**2.59**	0.26	1.21	4.52	**5.53**
Semantic ER	0.55	2.33	2.81	**4.33**	0.00	3.79	5.10	**10.1**	0.51	3.12	0.69	**8.08**	0.24	2.74	0.73	**9.62**
BLEU$*$max	22.0	30.5	36.9	**41.7**	20.3	52.0	46.9	**62.9**	20.0	38.5	24.4	**60.0**	23.3	38.0	29.9	**65.2**
BLEU$*$10	16.9	21.0	18.4	**26.3**	13.8	34.6	31.5	**42.7**	14.4	25.9	13.3	**34.8**	16.9	26.7	20.4	**43.4**

Our low-budget poisoning attack consistently outperforms the malicious provider (Zhang et al., 2026). This is notable because our crowdsourced-data attacker controls about 2% of the downstream fine-tuning examples, whereas approximately 49% of the provider's instruction-alignment data are backdoor-related. These results show that a small number of attacker-contributed examples can induce stronger extraction than a backdoor implanted by the pre-trained-model provider.

5.4 How Does Hosted-Interaction Poisoning Compare with Direct Submission?

We compare the direct-submission and hosted-interaction poisoning scenarios described in Section 3. Table 2 reports their extraction performance alongside benign SFT. Hosted-interaction poisoning improves extraction performance over benign SFT in nearly all metrics, demonstrating that poisoning through ordinary chatbot interactions can amplify training-data extraction. Direct submission remains more effective overall. For example, Qwen2.5-14B on OpenMathInstruct achieves a near-verbatim extraction rate of 8.44% under direct submission, compared to 3.49% under hosted interaction; The corresponding Max BLEU scores are 77.8 and 52.3, respectively. This gap may arise partly because direct submission uses questions drawn from the same downstream SFT distribution, whereas hosted interaction relies on chatbot-generated questions that may be repetitive or distributionally different.

Table 2: Comparison of two poisoning scenarios. SFT denotes the fine-tuned model without poisoning. Bold indicates the strongest extraction performance (%).

	Qwen2.5-7B-It			Llama-3.1-8B-It			Qwen2.5-14B-It			Mistral-Nemo-12B-It		
Metric	**SFT**	**Direct**	**Hosted**	**SFT**	**Direct**	**Hosted**	**SFT**	**Direct**	**Hosted**	**SFT**	**Direct**	**Hosted**
OpenMathInstruct												
Near-verbatim ER	1.84	**6.48**	2.96	2.31	**2.77**	2.50	2.38	**8.44**	3.49	1.85	**4.47**	2.75
Semantic ER	2.73	**7.25**	5.75	4.15	**5.26**	4.60	3.68	**10.08**	5.18	2.68	**6.58**	5.08
BLEU$*$max	43.7	**70.8**	53.6	56.1	**59.0**	58.3	44.2	**77.8**	52.3	46.6	**64.0**	61.4
BLEU$*$10	32.8	**41.1**	36.0	39.1	39.8	**42.0**	31.4	**49.9**	37.2	33.0	**44.1**	**44.1**
AceReason												
Near-verbatim ER	1.19	1.28	**1.86**	1.90	**6.07**	2.14	1.84	2.59	**2.76**	1.21	**5.53**	1.87
Semantic ER	2.33	**4.33**	3.02	3.79	**10.1**	4.19	3.12	**8.08**	4.01	2.74	**9.62**	3.47
BLEU$*$max	30.5	**41.7**	35.2	52.0	**62.9**	50.0	38.5	**60.0**	41.7	38.0	**65.2**	41.2
BLEU$*$10	21.0	**26.3**	22.7	34.6	**42.7**	31.7	25.9	**34.8**	27.7	26.7	**43.4**	28.6

5.5 How Does the Poisoning Budget Affect Extraction?

We evaluate extraction performance across poisoning budgets of $P \in \{50, 100, 200\}$. Figure 3 shows that larger poisoning budgets generally correlate with higher near-verbatim extraction rates, although 50 poisoning samples already achieve performance comparable to larger budgets in several configurations. Notably, with only 50 poisoning samples, near-verbatim extraction reaches $3.71\times$ the benign-SFT rate for Qwen2.5-14B on OpenMathInstruct and $3.08\times$ for Llama-3.1-8B on AceReason. These results demonstrate that even a small poisoning budget can substantially amplify extraction. Table 13 in the appendix reports the full results across all metrics.

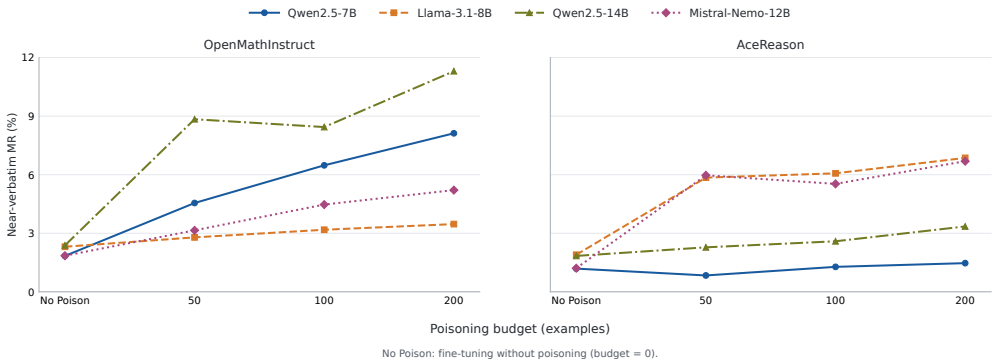


Figure 3: Effect of poisoning budget on the near-verbatim extraction rate (%), evaluated against OpenMathInstruct and AceReason across all four (instruction-following) models.

5.6 How Much Target-Model Performance Can an Attacker Recover Using Extracted Data?

The attacker may query the target model with extracted instructions, obtain corresponding responses, and use the resulting SFT pairs to fine-tune an attacker-owned model. To evaluate performance recovery under favorable conditions, we use an average extraction budget of $N = 1{,}000$ queries per anchor and select the $5{,}040$ extracted instructions with the highest cosine similarity to the original training instructions, matching the training-split size. We then query the target model for responses and fine-tune the same base model. We evaluate utility using perplexity on held-out responses, BLEU for lexical similarity, and embedding cosine similarity for semantic similarity.

Table 3 compares distillation using extracted instructions after poisoning (Poison-Dist.) under the direct-submission setting with three references: the base model before downstream SFT (Base), distillation using extracted instructions without poisoning (SFT-Dist.), and the target model (Proprietary). Both distilled models substantially improve over the base model across the evaluated metrics. Poison-Dist. provides a modest additional improvement over SFT-Dist. overall, with mean BLEU increasing from 30.3 to 31.0 for Qwen2.5-7B-Instruct on OpenMathInstruct and from 27.9 to 28.5 for Llama-3.1-8B-Instruct on AceReason. Notably, on AceReason, Poison-Dist. achieves lower perplexity than the proprietary model itself (1.41 versus 1.50). Overall, extracted instructions paired with target-model responses can recover much of the target model's downstream performance, while poisoning provides a modest additional gain over extraction without poisoning.

5.7 Additional Results

We further assess how fine-tuning dataset size, query budget, and prompt masking affect extraction, and whether poisoning preserves downstream utility. Extraction performance under the fixed poisoning budget remains comparable across different fine-tuning dataset sizes (Appendix I.3). Poisoning causes limited changes in the measured downstream utility (Appendix I.4). Increasing the query budget enables greater near-verbatim extraction of fine-tuning instructions, reaching 19.31% for Qwen-2.5-7b-Instruct on OpenMathInstruct (Appendix I.5). Restricting the SFT loss to assistant responses reduces near-verbatim instruction leakage, but poisoning still increases extraction relative to the corresponding benign-SFT baseline in most configurations (Appendix I.6).

Table 3: Downstream utility (perplexity, BLEU, and cosine similarity) of distilled models and reference baselines. SFT-Dist. and Poison-Dist. use instructions extracted without and with poisoning, respectively. Base denotes the base model before SFT, and Proprietary the target deployed model. Bold indicates the best score within each setting.

	Qwen2.5-7B-It/OpenMathInstruct				**Llama-3.1-8B-It/AceReason**			
Metric	**Base**	**SFT-Dist.**	**Poison-Dist.**	**Proprietary**	**Base**	**SFT-Dist.**	**Poison-Dist.**	**Proprietary**
Perplexity (↓)	1.74	1.48	1.47	**1.40**	1.85	1.67	**1.41**	1.50
Mean BLEU (↑)	24.0	30.3	31.0	**32.5**	9.86	27.9	**28.5**	**28.5**
Mean Cosine (↑)	91.1	91.6	**92.1**	91.9	80.7	**88.7**	88.5	88.5

6 Potential Defense and Mitigation

Data filtering offers a potential defense against poisoning by screening crowdsourced contributions before fine-tuning. Such screening can identify unusual examples through anomaly detection or assess whether their content is appropriate for training (Elouataoui et al., 2023). We consider three text anomaly detectors: Context Vector Data Description (CVDD) (Ruff et al., 2019), which learns text representations and context vectors describing normal text; DATE (Manolache et al., 2021), which uses self-supervised models to learn token- and sequence-level tasks; and Simplified Isolation Kernel (SIK) (Cao et al., 2025), which maps dense text embeddings into sparse features. Additionally, we consider an LLM judge that assesses the quality of each training example (Xu & Ding, 2025).

We evaluate these methods on OpenMathInstruct under the direct-submission setting, using a mixture of 100 poisoning examples and 5,040 benign examples. For CVDD, DATE, and SIK, we apply each method to instruction-only, response-only, and instruction-response pairs, deduplicating the text separately before detection. We set an anomaly-score cutoff to flag 2% of examples after deduplication, reflecting the poisoning fraction in the original mixture. For the LLM-judge, we assess each pair on task alignment, responsiveness, safety, confidentiality and privacy, and adherence to safeguards, using a prompt provided in Appendix J. Table 4 reports F1, precision, and recall, treating poisoning examples as the positive class. The results show limited effectiveness across all filtering methods, with most configurations detecting no poisoned examples. The most promising result comes from CVDD applied to responses, but even this fails to detect the majority of poisoning samples and scores only 0.378 in F-1. The results demonstrate the stealthiness of poisoned samples, which resemble ordinary, domain-relevant SFT pairs written without degenerate or harmful content.

Table 4: Poison detection performance across filtering methods and sample representations. Inst. and Res. denote instruction and response, respectively; Inst.+Res. denotes complete instruction–response pair. Poisoning examples constitute the positive class.

	CVDD (Ruff et al., 2019)			**DATE** (Manolache et al., 2021)			**SIK** (Cao et al., 2025)			**LLM Judge**
Metric	**Inst.**	**Res.**	**Inst.+Res.**	**Inst.**	**Res.**	**Inst.+Res.**	**Inst.**	**Res.**	**Inst.+Res.**	**Inst.+Res.**
F1	0.000	**0.378**	0.139	0.038	0.000	0.000	0.000	0.020	0.000	0.024
Precision	0.000	**0.369**	0.136	0.020	0.000	0.000	0.000	0.020	0.000	0.017
Recall	0.000	0.388	0.143	**0.400**	0.000	0.000	0.000	0.020	0.000	0.041

7 Conclusion

In this paper, we study how poisoning a small fraction of crowdsourced SFT data can amplify the extraction of fine-tuning instructions previously unknown to the attacker through output-only access to the deployed model. With only 50 poisoned samples, near-verbatim extraction reaches $3.71\times$ the rate without poisoning for Qwen2.5-14B on OpenMathInstruct and $3.08\times$ for Llama-3.1-8B on AceReason. Furthermore, existing anomaly detection methods and LLM-based filtering evaluated in our study fail to identify most poisoning samples, highlighting the stealthiness of our attack. Together, these findings reveal a concrete privacy risk arising from crowdsourced SFT, where malicious contributions can amplify leakage of other users' training data.

AI USE STATEMENT

In this work, we used Codex to assist with experimental code for the poisoning and extraction strategies described in Section 4 and the visualizations in Appendix I.2 through I. This AI-assisted code was verified through manual inspections, debugging of inputs and outputs, and experimental trials. Additionally, we used ChatGPT as a retrieval tool to identify related works and as an editing assistant to format mathematical notations and tables, assist with the design of Figure 1, and improve the grammar and clarity of the manuscript written by the authors. We have not used generative AI tools to generate or brainstorm research ideas, derive our methodology, or interpret the results. We take responsibility for the final content of this work, including text, claims, and related artifacts.

ETHICS STATEMENT

Our research highlights potential privacy risks arising from crowdsourced fine-tuning: an adversary can poison a small fraction of fine-tuning data to amplify the extraction of previously unseen instructions contributed by other users. By identifying these vulnerabilities, we aim to support safer fine-tuning practices and more reliable deployment of LLM systems. Our work emphasizes the need for reliable and scalable screening of crowd-sourced contributions that account for their potential adverse effects on training-data privacy that go beyond the capabilities of existing anomaly detection methods evaluated in our work.

REPRODUCIBILITY STATEMENT

All the relevant code and datasets are in the submitted supplementary material to ensure the reproducibility of our work. As for model checkpoints, we detail fine-tuning parameter configurations in Appendix D.

REFERENCES

Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Anthropic Model Card*, 2024. URL https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Teodora Baluta, Shiqi Shen, S Hitarth, Shruti Tople, and Prateek Saxena. Membership inference attacks and generalization: A causal perspective. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 249–262, 2022.

Federico Barbero, Xiangming Gu, Christopher A. Choquette-Choo, Chawin Sitawarin, Matthew Jagielski, Itay Yona, Petar Veličković, Ilia Shumailov, and Jamie Hayes. Extracting alignment data in open models, 2025. URL https://arxiv.org/abs/2510.18554.

Wassim Bouaziz, Mathurin VIDEAU, Nicolas Usunier, and El-Mahdi El-Mhamdi. Winter soldier: Backdooring language models at pre-training with indirect data poisoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ORKhJBFepG.

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

Yang Cao, Sikun Yang, Yujiu Yang, Lianyong Qi, and Ming Liu. Text anomaly detection with simplified isolation kernel. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 12702–12713, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.680. URL https://aclanthology.org/2025.findings-emnlp.680/.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pp. 2633–2650, 2021.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1897–1914, 2022a. doi: 10.1109/SP46214.2022.9833649.

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

Xiaoyi Chen, Siyuan Tang, Rui Zhu, Shijun Yan, Lei Jin, Zihao Wang, Liya Su, Zhikun Zhang, XiaoFeng Wang, and Haixu Tang. The janus interface: How fine-tuning in large language models amplifies the privacy risks. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 1285–1299, 2024.

Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron: Advancing math and code reasoning through reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=EgArbnS0BA.

Yuetian Chen, Yuntao Du, Kaiyuan Zhang, Ashish Kundu, Charles Fleming, Bruno Ribeiro, and Ninghui Li. Window-based membership inference attacks against fine-tuned large language models. In *35th USENIX Security Symposium (USENIX Security 26)*, Baltimore, MD, August 2026. USENIX Association. URL https://www.usenix.org/conference/usenixsecurity26/presentation/chen-yuetian.

Yufei Chen, Chao Shen, Yun Shen, Cong Wang, and Yang Zhang. Amplifying membership exposure via data poisoning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 29830–29844. Curran Associates, Inc., 2022. doi: 10.52202/068431-2163. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c0f240bb986df54b38026398da1ae72a-Paper-Conference.pdf.

Shuai Cheng, Shu Meng, Haitao Xu, Haoran Zhang, Shuai Hao, Chuan Yue, Wenrui Ma, Meng Han, Fan Zhang, and Zhao Li. Effective {PII} extraction from {LLMs} through augmented {Few-Shot} learning. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 8155–8173, 2025.

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

Yuyang Deng, Ni Zhao, and Xin Huang. Early chatgpt user portrait through the lens of data. In *2023 IEEE International Conference on Big Data (BigData)*, pp. 4770–4775, 2023. doi: 10.1109/BigData59044.2023.10386415.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. How abilities in large language models are affected by supervised fine-tuning data composition. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 177–198, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.12. URL https://aclanthology.org/2024.acl-long.12/.

Widad Elouataoui, Saida El Mendili, and Youssef Gahi. Big data quality anomaly scoring framework using artificial intelligence. In *2023 7th IEEE Congress on Information Science and Technology (CiSt)*, pp. 87–92, 2023. doi: 10.1109/CiSt56084.2023.10409909.

Wenjie Fu, Huandong Wang, Chen Gao, Guanghua Liu, Yong Li, and Tao Jiang. Membership inference attacks against fine-tuned large language models via self-prompt calibration. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=PAWQvrForJ.

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

Jamie Hayes, Ilia Shumailov, Christopher A. Choquette-Choo, Matthew Jagielski, Georgios Kaissis, Milad Nasr, Meenatchi Sundaram Muthu Selva Annamalai, Niloofar Mireshghallah, Igor Shilov, Matthieu Meeus, Yves-Alexandre de Montjoye, Katherine Lee, Franziska Boenisch, Adam Dziedzic, and A. Feder Cooper. Exploring the limits of strong membership inference attacks on large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=x0i7wvRLHK.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the association for computational linguistics: ACL 2023*, pp. 8003–8017, 2023.

Anthony Hughes, Vasisht Duddu, N Asokan, Nikolaos Aletras, and Ning Ma. Patch: Mitigating pii leakage in language models with privacy-aware targeted circuit patching. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 5139–5153, 2026.

Seongho Keum, Dongwon Shin, Leo Marchyok, Sanghyun Hong, and Sooel Son. Private investigator: Extracting personally identifiable information from large language models using optimized prompts. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 8175–8194, 2025.

Myeongseob Ko, Nikhil Reddy Billa, Adam Nguyen, Charles Fleming, Ming Jin, and Ruoxi Jia. Retracing the past: LLMs emit training data when they get lost. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 35316–35337, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1789. URL https://aclanthology.org/2025.emnlp-main.1789/.

Andreas Köpf, Yannic Kilcher, Dimitri Von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. Openassistant conversations-democratizing large language model alignment. *Advances in neural information processing systems*, 36:47669–47681, 2023.

Zongjie Li, Daoyuan Wu, Shuai Wang, and Zhendong Su. Differentiation-based extraction of proprietary data from fine-tuned llms. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3071–3085, 2025.

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Analyzing leakage of personally identifiable information in language models. In *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 346–363. IEEE, 2023.

Md Abdullah Al Mamun, Ngoc Phu Doan, Pedram Zaree, Nael Abu-Ghazaleh, and Ihsen Alouani. Loss landscape poisoning: Targeted extraction of unseen training data from llms. *arXiv preprint arXiv:2606.17110*, 2026.

Andrei Manolache, Florin Brad, and Elena Burceanu. DATE: Detecting anomalies in text via self-supervision of transformers. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 267–277, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.25. URL https://aclanthology.org/2021.naacl-main.25/.

Leo Marchyok, Zachary Coalson, Sungho Keum, Sooel Son, and Sanghyun Hong. Discovering universal activation directions for pii leakage in language models. *arXiv preprint arXiv:2602.16980*, 2026.

Milad Nasr, Javier Rando, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from aligned, production language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vjel3nWP2a.

Ashwinee Panda, Christopher Choquette-Choo, Zhengming Zhang, Yaoqing Yang, and Prateek Mittal. Teach llms to phish: Stealing private information from language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54983–55007, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/f206871468dc89cb20fefbc75f2de861-Paper-Conference.pdf.

Lukas Ruff, Yury Zemlyanskiy, Robert Vandermeulen, Thomas Schnake, and Marius Kloft. Self-attentive, multi-context one-class classification for unsupervised anomaly detection on text. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4061–4071, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1398. URL https://aclanthology.org/P19-1398/.

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

Alexandra Souly, Javier Rando, Ed Chapman, Xander Davies, Burak Hasircioglu, Ezzeldin Shereen, Carlos Mougan, Vasilios Mavroudis, Erik Jones, Chris Hicks, et al. Poisoning attacks on llms require a near-constant number of poison samples. *arXiv preprint arXiv:2510.07192*, 2025.

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

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

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*, 2024.

Florian Tramèr, Reza Shokri, Ayrton San Joaquin, Hoang Le, Matthew Jagielski, Sanghyun Hong, and Nicholas Carlini. Truth serum: Poisoning machine learning models to reveal their secrets. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, pp. 2779–2792, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450394505. doi: 10.1145/3548606.3560554. URL https://doi.org/10.1145/3548606.3560554.

Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. Poisoning language models during instruction tuning. In *International Conference on Machine Learning*, pp. 35413–35425. PMLR, 2023.

Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. Openchat: Advancing open-source language models with mixed-quality data. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=AOJyfhWYHf.

Jiashu Xu, Mingyu Ma, Fei Wang, Chaowei Xiao, and Muhao Chen. Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3111–3126, Mexico City, Mexico, June 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.171. URL https://aclanthology.org/2024.naacl-long.171/.

Ruiyao Xu and Kaize Ding. Large language models for anomaly and out-of-distribution detection: A survey. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 6007–6027, 2025.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024b.

Yueru Yan, Tuc Nguyen, Bo Su, Melissa Lieffers, and Thai Le. Sharechat: A dataset of chatbot conversations in the wild. *arXiv preprint arXiv:2512.17843*, 2025.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Boji Shan, Zeyuan Liu, Jia Deng, Huimin Chen, Ruobing Xie, et al. Advancing llm reasoning generalists with preference trees. In *International Conference on Learning Representations*, volume 2025, pp. 24897–24919, 2025.

Xiaoyong Yuan and Lan Zhang. Membership inference attacks and defenses in neural network pruning. In *31st USENIX Security Symposium (USENIX Security 22)*, pp. 4561–4578, 2022.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MAmmoTH: Building math generalist models through hybrid instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=yLClGs770I.

Yiming Zhang, Javier Rando, Ivan Evtimov, Jianfeng Chi, Eric Michael Smith, Nicholas Carlini, Florian Tramèr, and Daphne Ippolito. Persistent pre-training poisoning of llms. In *International Conference on Learning Representations*, volume 2025, pp. 31323–31340, 2025.

Zhexin Zhang, Yuhao Sun, Junxiao Yang, Shiyao Cui, yuanchao zhang, Hongning Wang, and Minlie Huang. Be careful when fine-tuning on open-source LLMs: Your fine-tuning data could be secretly stolen! In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=HhXOVhO3ia.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bl8u7ZRlbM.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. LMSYS-chat-1m: A large-scale real-world LLM conversation dataset. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=BOfDKxfwt0.

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

APPENDIX TABLE OF CONTENTS

Section	Content
Appendix A	Related Work on Privacy Attacks
Appendix B	Examples of Extraction Anchors
Appendix C	Prompt Construction for Poisoned Contributions in Hosted-Interaction
Appendix D	Fine-tuning Parameters
Appendix E	Effectiveness of Topic Anchors Compared to Prefixes
Appendix F	Discussion of the Design of Poisoning Sets
Appendix F.1	Number of Distinct Anchors
Appendix F.2	Anchor Selection
Appendix G	Discussion of Adaptive Query
Appendix H	Examples of Near-Verbatim and Semantic Extraction
Appendix I	Additional Results
Appendix I.1	How Effective is Our Attack on Instruction Data Used in Prior Work?
Appendix I.2	How Does the Poisoning Budget Affect Extraction?
Appendix I.3	How Does Extraction Scale with Fine-Tuning Dataset Size?
Appendix I.4	Does Poisoning Affect Downstream Utility?
Appendix I.5	Can Larger Query Budgets Improve Extraction?
Appendix I.6	Can Training Questions Be Extracted When Loss is Applied Only to Responses?
Appendix J	Prompt for LLM-Judge

A RELATED WORK ON PRIVACY ATTACKS

PII Extraction. Personally Identifiable Information (PII) refers to data that can be used to identify or distinguish an individual, such as Social Security numbers, dates of birth, and phone numbers. A substantial body of work examines PII extraction from LLMs (Lukas et al., 2023; Chen et al., 2024; Cheng et al., 2025; Keum et al., 2025). For example, Chen et al. (2024) show that access to a fine-tuning API can amplify leakage of PII memorized during pretraining, while Cheng et al. (2025) show that augmented few-shot prompting can improve PII extraction. At the representation level, Marchyok et al. (2026) identify universal activation directions that amplify PII leakage. On the defense side, Hughes et al. (2026) propose editing PII-related circuits to reduce leakage.

Membership Inference. Membership inference attacks (MIAs) aim to determine whether a sample was used during training (Shokri et al., 2017; Baluta et al., 2022; Yuan & Zhang, 2022; Fu et al., 2024; Chen et al., 2026; Hayes et al., 2026). Established approaches train shadow models to characterize differences in the target model's predictions on members and non-members (Shokri et al., 2017; Carlini et al., 2022a). Training such models can be computationally expensive for LLMs, and recent approaches differ in their requirements for additional model training (Hayes et al., 2026; Fu et al., 2024; Chen et al., 2026). For example, Chen et al. (2026) compare losses from the target and a pretrained reference model within sliding text windows, without training additional models. In contrast, Fu et al. (2024) prompt the target model to generate a dataset for fine-tuning a reference model used to calibrate membership scores.

B EXAMPLES OF EXTRACTION ANCHORS

Table 5 lists example topic anchors used for extraction. Each combination of domain, topic, and subtopic defines a topic anchor. For our evaluation, we construct a taxonomy of 416 topic anchors, with 215 in the math domain and 201 in the coding domain, using domain knowledge with assistance from ChatGPT. Neither the target SFT data nor the reference data are used to construct the taxonomy. The full taxonomy is provided in the supplementary materials 7.

Table 5: Examples of topic anchors for math and coding domains.

Domain	Topic	Subtopic
		Math
Math	Algebra	Linear equations, systems of equations, inequalities, quadratic equations, polynomials, factoring, rational expressions, radicals, exponents, algebraic simplification
Math	Number theory	Divisibility, prime numbers, factorization, gcd, lcm, modular arithmetic, congruences, diophantine equations, remainders, parity, digits, floor and ceiling
Math	Geometry	Triangles, circles, angles, polygons, area, perimeter, volume, surface area, similarity, congruence, Pythagorean theorem
Math	Discrete math	Logic, sets, relations, functions, proof by induction, recurrence relations, graphs, trees, boolean algebra
Math	Game theory	Nash equilibrium, zero sum games, dominant strategies, extensive form games, mechanism design, pareto efficiency, backward induction
		Coding
Coding	Two pointers	Sorted arrays, left right pointers, fast slow pointers, partitioning, palindrome check, cycle detection, merge two sorted arrays
Coding	Stack	Parentheses, monotonic stack, expression evaluation, next greater element, histogram rectangle, call stack simulation
Coding	Greedy	Interval scheduling, activity selection, exchange argument, local optimum, minimum operations, jump game
Coding	Bit manipulation	Xor, bit masks, subsets with masks, bit counting, shifts, and or operations, single number
Coding	Math in code	Modulo, combinatorics, number theory, gcd, primes, fast exponentiation, overflow, rounding, geometry in code, probability simulation

C Prompt Construction for Poisoned Contributions in Hosted-Interaction

In our hosted-interaction experiments, we use `gemini-3.6-flash` as the hosted model to generate questions and answers. To encourage diversity among the contributed questions, each question-generation request $G(a, c)$ combines a topic anchor a with an opening word c. For each topic, we vary c across seven to eight distinct opening words. The request takes the following general format:

> Please provide a {domain} question on {topic/subtopic} that a real user has asked, and that starts with the word {word}. Output the question only.

The opening-word constraint is used only when constructing poisoned contributions. Subsequent extraction queries use the topic anchor without specifying an opening word: *"Please provide a {domain} question on {topic/subtopic} that a real user has asked."* We also use this extraction prompt to query the base model before SFT and the fine-tuned model without poisoning during our experimental evaluation.

Let $\widetilde{x}_{\mathrm{adv}}$ denote the generated question. The adversary subsequently submits this question in a context-independent interaction with the hosted model to obtain an answer $\widetilde{y}_{\mathrm{adv}}$. The resulting contribution contains the two exchanges $G(a, c) \longrightarrow \widetilde{x}_{\mathrm{adv}}$ and $\widetilde{x}_{\mathrm{adv}} \longrightarrow \widetilde{y}_{\mathrm{adv}}$. Figure 4 illustrates such a contribution.

Question generation request

Instruction
Please provide a math question on trigonometry/trigonometric equations that a real user has asked and that starts with the prefix 'Determine'. Output the question only.

Response
Determine all solutions to the equation $\sin(x) + \sin(2x) + \sin(3x) = \cos(x) + \cos(2x) + \cos(3x)$ in the interval $[0, 2\pi]$.

Ordinary Q&A

Instruction
Determine all solutions to the equation $\sin(x) + \sin(2x) + \sin(3x) = \cos(x) + \cos(2x) + \cos(3x)$ in the interval $[0, 2\pi]$.

Response
To determine all solutions to the equation $\sin(x) + \sin(2x) + \sin(3x) = \cos(x) + \cos(2x) + \cos(3x)$ in the interval $[0, 2\pi]$, we start by ... *[response abbreviated]*

Figure 4: Example of hosted-interaction poisoning. The generated question also serves as the instruction in a separate question-answering interaction.

Our direct-submission experiments instead select question–answer pairs from the downstream SFT dataset to serve as attacker-owned examples. These pairs are included in fine-tuning but excluded from extraction evaluation. Since this construction does not involve generating questions, we do not use an opening-word constraint.

D Fine-tuning Parameters

We utilize a single H-100 GPU for training 7/8B models and an H-200 GPU for training 12/14B models. Table 6 provides detailed specifications of the full fine-tuning parameters. These values are used consistently across all models and datasets.

Table 6: Training hyperparameters for SFT.

Hyperparameter	**Value**
Learning rate	1×10^{-5}
Epochs	3
LR scheduler	Cosine, 3% warmup
Train batch size (per device)	1
Gradient accumulation steps	8
Weight decay	0.01
Sequence length	2048
Precision	BF16

E Effectiveness of Topic Anchors Compared to Prefixes

A major distinction between our work and previous training-data extraction work (Carlini et al., 2021; Nasr et al., 2025; Zhang et al., 2026) lies in conditioning extraction on topics rather than prefixes or opening words. In adversarial settings where true prefixes of target data are unknown, Zhang et al. (2026) employ another dataset to estimate likely opening words. We hypothesize that in downstream SFT for specialized applications or tasks, common opening words provide limited information about the desired content or instructions. In contrast, topic anchors, which need not appear verbatim in the fine-tuning data, can provide more specific cues about the desired instructions. Table 7 compares extraction performance between prefix-based anchors and topic-based anchors. Across nearly all model-dataset configurations and metrics, topic anchors outperform prefix anchors, both with and without poisoning. Interestingly, in some cases, topic-cued extraction without poisoning outperforms prefix-cued extraction with poisoning. For example, Qwen2.5-7B-Instruct on AceReason achieves a near-verbatim extraction rate of only 0.37% under prefix-cued extraction with poisoning, compared to 1.19% under topic-cued extraction without poisoning.

Table 7: Extraction Performance (%) of prefix- and topic-based extraction. SFT denotes fine-tuning without poisoning; Poison denotes fine-tuning with poisoning. ER denotes extraction rate.

	OpenMathInstruct								**AceReason**							
	Qwen2.5-7B-It				**Llama-3.1-8B-It**				**Qwen2.5-7B-It**				**Llama-3.1-8B-It**			
	SFT		**Poison**		**SFT**		**Poison**		**SFT**		**Poison**		**SFT**		**Poison**	
Metric	**Prefix**	**Topic**	**Prefix**	**Topic**	**Prefix**	**Topic**	**Prefix**	**Topic**	**Prefix**	**Topic**	**Prefix**	**Topic**	**Prefix**	**Topic**	**Prefix**	**Topic**
Near-verbatim ER	0.71	**1.84**	5.24	**6.48**	1.26	**2.31**	2.41	**2.77**	0.30	**1.19**	0.37	**1.28**	0.73	**1.90**	1.64	**6.07**
Semantic ER	1.56	**2.73**	6.15	**7.25**	2.71	**4.15**	3.34	**5.26**	0.51	**2.33**	2.98	**4.33**	3.14	**3.79**	6.07	**10.1**
Max BLEU	**46.4**	43.7	68.7	**70.8**	55.4	**56.1**	58.3	**59.0**	28.5	**30.5**	29.2	**41.7**	37.3	**52.0**	36.9	**62.9**
Top-10 BLEU	32.7	**32.8**	37.6	**41.1**	35.7	**39.1**	36.4	**41.0**	18.7	**21.0**	18.4	**26.3**	24.8	**34.6**	25.0	**42.7**

F Discussion on the Design of Poisoning Sets

The attacker uses experiments with a surrogate model and reference datasets to determine two aspects of the poisoning set: the number of distinct anchors and the topics they represent. We present extraction performance across different numbers of anchors and examine our topic-ranking strategy.

F.1 Number of Distinct Anchors

Table 8 reports extraction performance on the surrogate model Gemma-3-4B-Instruct, fine-tuned on the reference dataset MathInstruct. We use this sweep as a heuristic for choosing the number of poisoning anchors k, not to establish an optimality guarantee. We vary the number of distinct poisoning anchors k and generate 50 completions per extraction anchor using a uniform query allocation. Extraction performance varies modestly across the tested settings: $k = 5$ achieves the highest near-verbatim extraction rate and Max BLEU, while $k = 7$ achieves the highest semantic extraction rate and Top-10 BLEU. We select $k = 5$ for all subsequent poisoning experiments.

Table 8: Extraction performance (%) on Gemma-3-4B-Instruct fine-tuned on MathInstruct across different numbers of distinct poisoning anchors k.

Metric	$k = 1$	$k = 3$	$k = 5$	$k = 7$	$k = 10$	$k = 20$
Near-verbatim ER	1.15	1.25	**1.36**	1.275	1.25	1.25
Semantic ER	3.08	2.63	3.08	**3.16**	2.95	2.43
Max BLEU	45.0	46.1	**47.5**	46.6	45.0	45.1
Top-10 BLEU	30.8	30.9	31.7	**31.9**	30.1	30.3

F.2 Anchor Selection

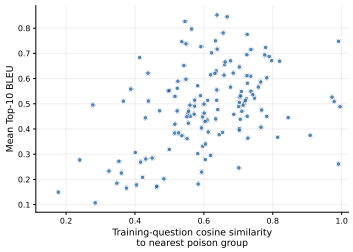
We assign each reference question to a domain-topic-subtopic anchor in the predefined taxonomy (Appendix B) using Qwen2.5-72B-Instruct. These assignments define the anchor-specific reference subsets $\mathcal{X}_a$ defined in Section 4.1. We embed the reference questions using Qwen3-Embedding-8B and rank each anchor by the mean pairwise cosine similarity between its associated reference questions and reference questions assigned to all other anchors. Finally, we select the top-k anchors.

Given a poisoning budget P and k selected anchors, we allocate $n = P/k$ poisoning questions to each anchor. With $P = 100$ and $k = 5$, this yields 20 questions per anchor. Among eligible anchors, we select the five highest-ranked math anchors for OpenMathInstruct, and the three highest-ranked math anchors and two highest-ranked coding anchors for AceReason. Table 9 lists the selected anchors and the lowest-ranked eligible anchors used for controlled analysis.

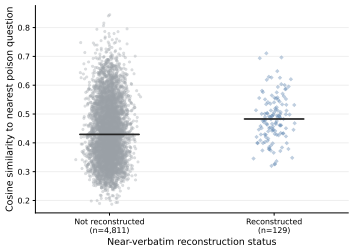
Our ranking strategy is motivated by the hypothesis that the target model is more likely to reconstruct fine-tuning questions that are semantically similar to the questions in the poisoning set. To examine this, we perform a correlation analysis in Figure 5. In (a), each point represents an extraction anchor, with Top-10 BLEU plotted against the mean pairwise cosine similarity between its associated training questions and those of its closest poisoning anchor. The positive correlation suggests that extraction quality tends to be higher for anchors whose associated training questions are more semantically similar to the poisoning questions, supporting our hypothesis. In (b), reconstructed training questions have moderately higher mean cosine similarity to their nearest poisoning question than unreconstructed counterparts. Finally, (c) compares extraction performance under two poisoning sets with uniform query allocation. Top-5 contains the five highest-ranked topics by reference-data similarity, while Bottom-5 contains the five lowest-ranked topics. Top-5 outperforms Bottom-5 across all metrics. To evaluate statistical significance, we further compute each extraction metric separately for each extraction anchor and perform two-sided paired tests across anchors. For extraction-rate metrics, anchor-level coverage is the fraction of training questions that are reconstructed by an extraction query with that anchor (aka $E(a)$). The results show that differences are statistically significant at $p < 0.05$ for three of the four metrics.

Table 9: Topic anchors selected using the reference datasets. Top-ranked and bottom-ranked anchors use the same domain allocation: five math anchors for OpenMathInstruct, and three math and two coding anchors for AceReason.

OpenMathInstruct			AceReason		
Domain	**Topic**	**Subtopic**	**Domain**	**Topic**	**Subtopic**
Top-ranked anchors					
Math	Algebra	Quadratic equations	Math	Intermediate algebra	Polynomials
Math	Number theory	Floor and ceiling	Math	Number theory	Diophantine equations
Math	Algebra	Factoring	Math	Trigonometry	Trigonometric equations
Math	Trigonometry	Angle additional formulas	Coding	Bit manipulation	Bit counting
Math	Intermediate algebra	Polynomials	Coding	Bit manipulation	Xor
Bottom-ranked anchors					
Math	Prealgebra	Unit conversions	Math	Counting & combinatorics	Arrangements
Math	Linear algebra	Vectors	Math	Counting & combinatorics	Permutations
Math	Geometry	Surface area	Math	Geometry	Polygons
Math	Optimization	Maximize	Coding	Math in code	Fast exponentiation
Math	Calculus	Unit conversion	Coding	Math in code	Fast Gcd



(a) Mean top-10 BLEU vs training question proximity to poison data



(b) Training-question proximity to poison data

Metric	Top-5	Bottom-5	p-value
Near-verbatim ER	**5.04**	4.39	0.002
Semantic ER	**6.15**	5.79	0.057
Max BLEU	**68**	64	0.029
Top-10 BLEU	**38.9**	36.7	0.006

(c) Extraction performance (%) of Top-5 and Bottom-5 poisoning topics.

Figure 5: Qwen2.5-7B-Instruct on OpenMathInstruct. (a) Top-10 BLEU per extraction anchor versus the mean pairwise cosine similarity of its associated training questions to poisoning questions. (b) Cosine similarity to the nearest poisoning question for near-verbatim reconstructed and unreconstructed training questions; horizontal bars indicate medians. (c) Extraction performance using the five highest-ranked (Top-5) and lowest-ranked (Bottom-5) poisoning topics. Statistical significance is evaluated at $\alpha = 0.05$.

G Discussion of Adaptive Query

In this section, we discuss the effectiveness of adaptive querying, detailed in Section 4.2. The algorithm for adaptive extraction is provided in Algorithm 1. Our adaptive query strategy is motivated by the hypothesis that common, easily synthesized questions are relatively predictable to a surrogate model that has not been fine-tuned on the downstream SFT data, while specialized instructions from the SFT distribution are less predictable and receive higher negative log-likelihood (NLL). To test this hypothesis, Figure 6 shows the relationship between the Top-10 BLEU score for each topic anchor and the average NLL of generated questions for that anchor under the surrogate model (Gemini-3.4B-Instruct). For fair analysis, we apply uniform query allocation of $N = 100$ per anchor rather than adaptive querying. Intuitively, a higher Top-10 BLEU score indicates that an anchor's best completions more closely resemble original questions in the fine-tuning data. In both figures, we observe a positive correlation between Top-10 BLEU and NLL, consistent with our hypothesis.

Additionally, Table 10 compares extraction performance under static querying, where we allocate the query budget uniformly across all anchors, and adaptive querying under the same total generation budget. We omit BLEU scores because the two strategies allocate different numbers of queries to

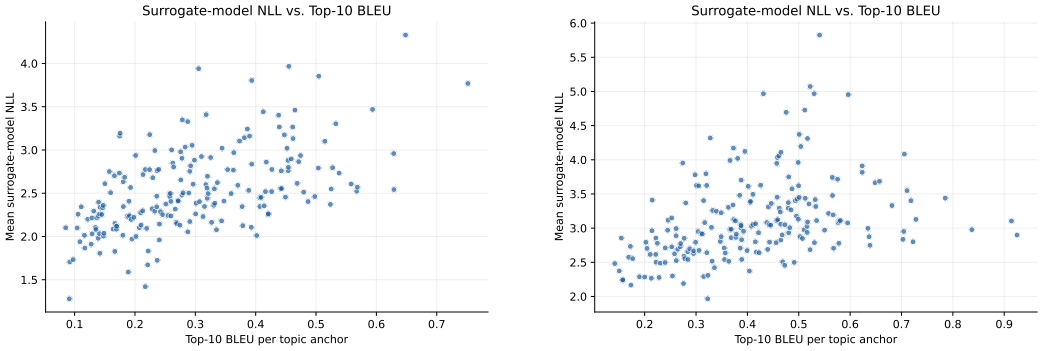
Algorithm 1 Adaptive extraction with a surrogate model

Require: $F_\theta, S_\phi, \mathcal{A}, T, M, r$
$\mathcal{A}_{\text{active}} \leftarrow \mathcal{A}$; $\mathcal{C}_a \leftarrow \emptyset$ for all $a \in \mathcal{A}$
while $\sum_a |\mathcal{C}_a| < T$ **do**
  for $a \in \mathcal{A}_{\text{active}}$ **do**
    $m \leftarrow \min(M, T - \sum_a |\mathcal{C}_a|)$
    Add up to m new samples from $F_\theta(\cdot \mid E(a))$ to $\mathcal{C}_a$
    if $\sum_a |\mathcal{C}_a| \geq T$ **then break**
  Compute $s(a)$ for each active anchor a using S_ϕ
  $k \leftarrow \lceil r|\mathcal{A}_{\text{active}}| \rceil$
  Keep the k highest-scoring anchors in $\mathcal{A}_{\text{active}}$
return $\bigcup_a \mathcal{C}_a$

Notation

F_θ	Target model
S_ϕ	Surrogate model
$\mathcal{A}$	All anchors
$\mathcal{A}_{\text{active}}$	Active anchors
$E(a)$	Extraction query
$\mathcal{C}_a$	Gnerated samples for a
T	Total budget
M	Samples per anchor per round (max.)
r	Retention fraction
$s(a)$	Mean surrogate NLL over $\mathcal{C}_a$
k	Retained anchor count

individual anchors, making a direct comparison impractical. The results show that across 15 out of 16 comparisons, adaptive querying improves extraction over static querying, although the gains are generally modest.



(a) Qwen-2.5-7B-Instruct (b) Llama-3.1-8B-Instruct

Figure 6: Surrogate-model (Gemma-3-4B-Instruct) average log-likelihood against Top-10 BLEU score for each topic anchor on OpenMathInstruct.

Table 10: Extraction performance (%) using static (Stat.) and adaptive (Adap.) querying. Static querying allocates completions uniformly across all topic anchors. N denotes the average query budget per anchor. ER denotes extraction rate.

	OpenMathInstruct								**AceReason**							
	Qwen2.5-7B-It				**Llama-3.1-8B-It**				**Qwen2.5-7B-It**				**Llama-3.1-8B-It**			
	$N = 50$		$N = 100$		$N = 50$		$N = 100$		$N = 50$		$N = 100$		$N = 50$		$N = 100$	
Metric	**Stat.**	**Adap.**	**Stat.**	**Adap.**	**Stat.**	**Adap.**	**Stat.**	**Adap.**	**Stat.**	**Adap.**	**Stat.**	**Adap.**	**Stat.**	**Adap.**	**Stat.**	**Adap.**
Near-verbatim ER	3.62	**3.99**	5.93	**6.48**	1.88	**2.06**	2.75	**2.77**	0.77	**0.79**	**1.34**	1.28	4.47	**4.84**	5.99	**6.07**
Semantic ER	4.39	**4.82**	6.90	**7.25**	3.26	**3.48**	5.16	**5.26**	2.53	**2.73**	4.29	**4.33**	6.66	**6.76**	9.86	**10.1**

H Examples of Near-Verbatim and Semantic Extraction

Table 11 presents examples of near-verbatim and semantic extraction from Llama-3.1-8B-Instruct fine-tuned on AceReason with poisoning ($P = 100$).

Table 11: Examples of extracted instructions. Near-verbatim matches preserve most of the original wording, whereas semantic matches preserve the underlying meaning and structure.

Original training instruction	Extracted instruction
Near-verbatim match (Levenshtein = 0.86, Cosine = 0.99)	

You are given an array of strings `products` and a string `searchWord`.
 	
Design a system that suggests at most three product names from `products` after each character of `searchWord` is typed. Suggested products should have common prefix with `searchWord`. If there are more than three products with a common prefix return the three lexicographically minimums products.
 	
Return *a list of lists of the suggested products after each character of* `searchWord` *is typed*.
 	
Example 1:
Input: `products = ["mobile","mouse","moneypot", "monitor","mousepad"], searchWord = "mouse"`
Output: `[["mobile","moneypot","monitor"], ["mobile","moneypot","monitor"],["mouse", "mousepad"],["mouse","mousepad"],["mouse", "mousepad"]]`
Explanation: Products sorted lexicographically = `["mobile", "moneypot","monitor","mouse","mousepad"]`.
 	
• After typing `"m"` and `"mo"`, all products match and we show the top 3: `["mobile", "moneypot", "monitor"]`.
• After typing `"mou"`, `"mous"`, and `"mouse"`, the system suggests: `["mouse", "mousepad"]`.
 	
Example 2:
Input: `products = ["havana"], searchWord = "havana"`
Output: `[["havana"], ["havana"], ["havana"], ["havana"], ["havana"], ["havana"]]`
Explanation: The only word `"havana"` will always be suggested while typing the search word.
 	
Constraints:
`1 <= products.length <= 1000`
`1 <= products[i].length <= 3000`
`1 <= sum(products[i].length) <= 2 * 10^4`
All the strings of `products` are **unique**.
`products[i]` consists of lowercase English letters.
`1 <= searchWord.length <= 1000`
`searchWord` consists of lowercase English letters.
 	
Write Python code to solve the problem. Please place the solution code in the following format:`# Your solution code here`
	You are given an array of strings `products` and a string `searchWord`.
	
	Design a system that suggests at most three product names from `products` after each character of `searchWord` is typed. Suggested products should have common prefix with `searchWord`. If there are more than three products with a common prefix return the three lexicographically minimums products.
	
	Return *a list of lists of the suggested products after each character of* `searchWord` *is typed*.
	
	Example 1:
	Input: `products = [["mobile","mouse","moneypot", "monitor","mousepad"]], searchWord = "mouse"`
	Output: `[["mobile","moneypot","monitor"], ["mobile","moneypot","monitor"],["mouse", "mousepad"],["mouse","mousepad"],["mouse", "mousepad"]]`
	Explanation: Products sorted lexicographically = `["mobile", "moneypot","monitor","mouse","mousepad"]`.
	
	• After typing `"m"` and `"mo"`, all products match and we show the top 3: `["mobile", "moneypot", "monitor"]`.
	• After typing `"mou"`, `"mous"`, and `"mouse"`, the system suggests: `["mouse", "mousepad"]`.
	
	Example 2:
	Input: `products = [["havana"]], searchWord = "haiti"`
	Output: `[]`
	
	Constraints:
	`1 <= products.length <= 1000`
	`1 <= products[i].length <= 30`
	All the strings of `products` are **unique**.
	`products[i]` consists of lowercase English letters.
	`1 <= searchWord.length <= 1000`
	`searchWord` consists of lowercase English letters.
	
	Write Python code to solve the problem. Please place the solution code in the following format: `# Your solution code here`

Semantic match (Levenshtein = 0.66, Cosine = 0.88)	

Write a python function to determine if a given string can be segmented into a space-separated sequence of one or more dictionary words.
 	
For example, given the dictionary = `["dog", "cat", "code", "sand", "dogcar"]`, and the string = `"dogcarcode"`, the function should return `True` since the string can be segmented into `"dog car code"`. However, if the string was `"sandydog"`, it should return `False` since no valid segmentation exists.
 	
Solve the problem starting with the provided function header.
 	
Function header:
`def word_break(wordDict, s):`
 	
Please place the solution code in the following format:
`# Your solution code here`
	Write a function to check if a given string can be segmented into a space-separated sequence of one or more dictionary words.
	
	For example:
	
	• Given `dictionary = ["dog", "cat"], sentence = "dogcat"`. Return `true` because it can be segmented as `"dog cat"`.
	• Given `dictionary = ["dog", "cat"], sentence = "toad"`. Return `false` because it does not match any dictionary word.
	
	Solve the problem starting with the provided function header.
	
	Function header:
	`def wordBreak(self, s: str, wordDict: List[str]) -> bool:`
	
	Please place the solution code in the following format:
	`# Your solution code here`

I Additional Results

I.1 How Effective is Our Attack on Instruction Data Used in Prior Work?

Our main experiments use mathematics and coding datasets to reflect a downstream provider's goal of adapting an already instruction-tuned model to specialized applications. For comparison with prior work (Zhang et al., 2026), we additionally evaluate our attack on Finance Alpaca [1], which was used in their study and combines general-purpose instruction-following examples with finance-specific question–answer pairs. We use the training splits and 200 opening words provided in their official repository [2]. For these experiments, opening words serve as both poisoning and extraction anchors. We retain adaptive querying with the same parameters, the poisoning budget of $P = 100$, and the fine-tuning hyperparameters reported in Appendix D. For evaluation, we set the Levenshtein similarity threshold to 0.85 for near-verbatim extraction and the cosine similarity threshold to 0.90 for semantic extraction. Table 12 presents extraction performance against three reference conditions: the model before downstream SFT (Base), the fine-tuned model without poisoning (SFT), and the malicious-provider backdoor proposed by Zhang et al. (2026) (MP).

The results show that our poisoning attack outperforms both SFT and the MP across all metrics for Llama and Mistral. For Qwen models, however, benign SFT achieves the highest extraction on most metrics, and even MP does not improve upon SFT despite Finance Alpaca being used in its original study. Moreover, the base models already generate text resembling Finance Alpaca instructions before downstream SFT: Max BLEU scores ranges from 41.0 to 56.0 on Finance Alpaca, compared with 31.0–35.4 on OpenMathInstruct and 20.3–23.3 on AceReason (Table 1). The corresponding Top-10 BLEU ranges are 27.3–40.3, 23.0–26.1, and 13.8–16.9, respectively. These higher scores may arise because general-purpose instructions are easier to synthesize or because the models have already seen similar data during pretraining or instruction tuning. Finance Alpaca instructions are also shorter, averaging 12 tokens, compared with 64 on OpenMathInstruct and 170 on AceReason. Together, these factors make lexical similarity harder to attribute specifically to SFT memorization.

Table 12: Comparison of our poisoning attack against three reference baselines on Finance Alpaca. ER denotes extraction rate (%). Base, SFT, and MP denote the model before downstream SFT, the fine-tuned model without poisoning, and the malicious-provider backdoor (Zhang et al., 2026), respectively. Bold indicates the strongest extraction performance.

	Qwen2.5-7B-It				**Llama-3.1-8B-It**				**Qwen2.5-14B-It**				**Mistral-Nemo-12B-It**			
Metric	**Base**	**SFT**	**MP**	**Ours**	**Base**	**SFT**	**MP**	**Ours**	**Base**	**SFT**	**MP**	**Ours**	**Base**	**SFT**	**MP**	**Ours**
Near-verbatim ER	0.44	**2.22**	1.04	1.84	0.08	2.16	2.34	**3.69**	0.76	**2.04**	1.16	1.24	0.68	3.06	5.74	**6.65**
Semantic ER	1.42	**6.60**	3.60	5.22	0.50	7.22	7.80	**10.82**	2.16	**6.78**	4.80	4.27	2.02	7.56	14.62	**15.08**
Max BLEU	41.9	67.6	54.7	**68.4**	41.0	62.6	63.5	**68.2**	46.5	**62.8**	57.4	54.6	56.0	69.7	70.9	**75.8**
Top-10 BLEU	30.2	**48.4**	36.1	48.1	27.3	44.7	46.8	**49.2**	34.4	**46.5**	40.8	39.6	40.3	50.4	53.2	**56.0**

I.2 How Does the Poisoning Budget Affect Extraction?

Table 13 presents extraction performance across three poisoning budgets ($P = 50, 100, 200$). The largest poisoning budget of $P = 200$ achieves the highest scores in nearly all metrics. In many configurations, the smallest poisoning budget of $P = 50$ still performs comparably to a larger poisoning budget. For example, with $P = 50$, Qwen2.5-14B achieves the near-verbatim extraction rate of 8.84% on OpenMathInstruct, compared to 8.44% with $P = 100$. Similarly, Mistral-Nemo-12B yields 5.97% on AceReason with $P = 50$, compared to 5.53% with $P = 100$.

I.3 How Does Extraction Scale with Fine-Tuning Dataset Size?

Souly et al. (2025) show that a near-constant number of poisoned samples preserves backdoor effectiveness as training datasets grow in size. To examine whether a similar pattern holds for SFT-data

[1] https://huggingface.co/datasets/gbharti/finance-alpaca

[2] https://github.com/thu-coai/Backdoor-Data-Extraction

Table 13: Extraction performance (%) on OpenMathInstruct and AceReason across poisoning budgets of 50, 100, and 200 examples. SFT denotes the fine-tuned model without poisoning.

	Qwen2.5-7B-It				Llama-3.1-8B-It				Qwen2.5-14B-It				Mistral-Nemo-12B-It			
Metric	**SFT**	**50**	**100**	**200**	**SFT**	**50**	**100**	**200**	**SFT**	**50**	**100**	**200**	**SFT**	**50**	**100**	**200**
OpenMathInstruct																
Near-verbatim ER	1.84	4.55	6.48	**8.12**	2.31	2.79	3.18	**3.47**	2.38	8.84	8.44	**11.3**	1.85	3.15	4.47	**5.21**
Semantic ER	2.73	6.27	7.25	**8.64**	4.15	5.39	5.36	**5.58**	3.68	9.68	10.08	**12.17**	2.68	5.91	6.58	**7.52**
BLEU*max	43.7	63.2	70.8	**79.3**	56.1	59.0	57.6	**63.4**	44.2	78.1	77.8	**89.4**	46.6	59.8	64.0	**71.7**
BLEU*10	32.8	36.5	41.1	**48.1**	39.1	40.8	39.8	**43.1**	31.4	50.6	49.9	**61.5**	33.0	41.8	44.1	**47.7**
AceReason																
Near-verbatim ER	1.19	0.84	1.28	**1.47**	1.90	5.85	6.07	**6.86**	1.84	2.28	2.59	**3.35**	1.21	5.97	5.53	**6.69**
Semantic ER	2.33	2.40	4.33	**4.82**	3.79	9.18	10.1	**11.0**	3.12	7.58	8.08	**8.49**	2.74	9.86	9.62	**10.79**
BLEU*max	30.5	35.6	41.7	**44.4**	52.0	60.5	62.9	**65.6**	38.5	55.7	**60.0**	57.7	38.0	63.4	65.2	**69.1**
BLEU*10	21.0	21.4	26.3	**27.6**	34.6	43.1	42.7	**46.4**	25.9	33.5	34.8	**36.1**	26.7	44.3	43.4	**46.9**

extraction, we fix the poisoning budget at $P = 100$ and vary the fine-tuning dataset size from 2,500 to 10,000 examples. Table 14 reports the results for Qwen2.5-7B-Instruct on OpenMathInstruct and Llama-3.1-8B-Instruct on AceReason, while Figure 7 plots the near-verbatim extraction rates. Despite the fourfold increase in fine-tuning data, near-verbatim extraction remains within 6.48–7.51% and 5.96–6.34%, respectively. These results suggest that a fixed budget of 100 poisoned samples can maintain similar extraction effectiveness as the fine-tuning dataset grows, consistent with a near-constant poisoning behavior found in Souly et al. (2025).

Table 14: Extraction performance (%) across fine-tuning dataset sizes ($|D| = 2{,}500, 5{,}000, 10{,}000$). SFT denotes the fine-tuned model without poisoning, and Poison denotes the fine-tuned model with $P = 100$ poisoned samples.

	Qwen2.5-7B / OpenMathInstruct						Llama-3.1-8B / AceReason					
	SFT			Poison			SFT			Poison		
Metric	2,500	5,000	10,000	2,500	5,000	10,000	2,500	5,000	10,000	2,500	5,000	10,000
Near-verbatim ER	1.12	**1.84**	0.90	6.79	6.48	**7.51**	1.80	**1.90**	0.56	5.96	6.07	**6.34**
Semantic ER	**4.16**	2.73	1.68	**7.54**	7.25	7.46	3.24	**3.79**	2.26	9.38	**10.1**	9.43
Max BLEU	34.9	**43.7**	41.1	62.2	**70.8**	70.6	41.8	**52.0**	40.1	56.0	**62.9**	62.3
Top-10 BLEU	27.3	**32.8**	30.9	34.2	41.1	**41.5**	29.1	**34.6**	26.9	38.8	42.7	**43.0**

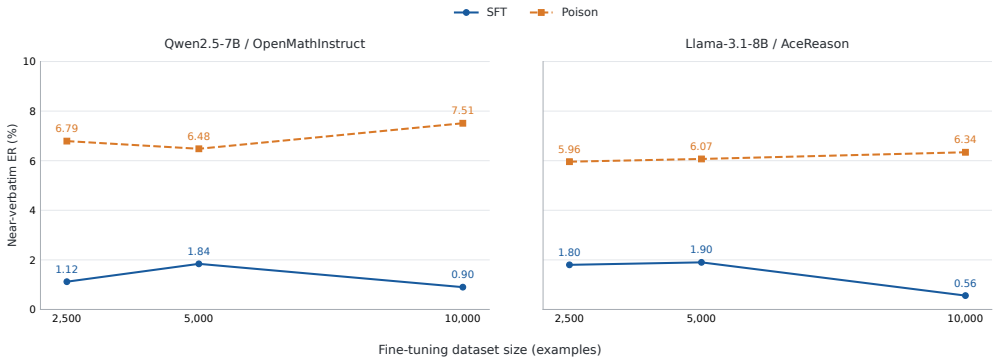


Figure 7: Near-verbatim extraction rate (%) across fine-tuning dataset sizes for Qwen2.5-7B-Instruct on OpenMathInstruct and Llama-3.1-8B-Instruct on AceReason. The poisoning budget is fixed at $P = 100$. With poisoning, near-verbatim extraction rates remain comparable across different fine-tuning sizes.

I.4 Does Poisoning Affect Downstream Utility?

We next evaluate whether poisoning affects the downstream utility of the resulting fine-tuned models. Table 15 presents utility results comparing poisoned models from two threat-model scenarios against the fine-tuned model without poisoning (SFT). Across model-dataset configurations, perplexity mostly remains unchanged, and mean cosine similarity differs by at most 0.4 points except for Qwen2.5-14B-Instruct. Most mean BLEU differences are below 1 point, except for three comparisons that decrease by 2.3–2.9. These results suggest that poisoning largely preserves utility.

Table 15: Utility comparison of the two poisoning settings (direct-submission and hosted-interaction) and the fine-tuned model without poisoning (SFT). Bold indicates the strongest score.

	Qwen2.5-7B-It			Llama-3.1-8B-It			Qwen2.5-14B-It			Mistral-Nemo-12B-It		
Metric	**SFT**	**Direct**	**Hosted**	**SFT**	**Direct**	**Hosted**	**SFT**	**Direct**	**Hosted**	**SFT**	**Direct**	**Hosted**
OpenMathInstruct												
	1.40	+0.00	+0.00	1.46	+0.00	+0.00	1.42	**-0.01**	**-0.01**	1.44	+0.00	+0.00
Mean BLEU (↑)	**35.4**	-2.9	**-0.0**	**32.9**	-0.9	-0.4	**32.1**	-2.3	-2.7	34.6	**+0.1**	-0.2
Mean Cosine (↑)	91.8	**+0.1**	**+0.1**	**91.6**	-0.1	-0.1	**91.3**	-1.2	-0.8	91.9	-0.0	**+0.1**
AceReason												
Perplexity (↓)	1.37	+0.00	+0.00	1.50	+0.00	+0.00	1.36	+0.00	+0.00	1.47	**-0.01**	**-0.01**
Mean BLEU (↑)	**31.0**	-0.2	-0.1	28.4	**+0.1**	-0.2	23.0	-0.8	**+0.7**	**30.1**	-0.3	-0.2
Mean Cosine (↑)	**88.7**	-0.4	-0.3	**88.7**	-0.2	-0.1	85.8	-0.3	**+0.5**	**89.1**	-0.2	**-0.0**

I.5 Can Larger Query Budgets Improve Extraction?

Unless otherwise stated, we use an average query budget of $N = 100$ completions per anchor. With adaptive querying, the total query budget is $T = N|\mathcal{A}|$, where $\mathcal{A}$ is the set of available topic anchors. To examine how extraction scales with the query budget, Figure 8 presents the near-verbatim extraction rate of Qwen-2.5-7B-Instruct on OpenMathInstruct across different query budgets. Increasing the query budget improves extraction, with the near-verbatim extraction rate reaching 19.31% of the original fine-tuning instructions at $T = 1000|\mathcal{A}|$. Our poisoning attack also consistently outperforms the malicious-provider backdoor attack proposed by Zhang et al. (2026) across different budgets.

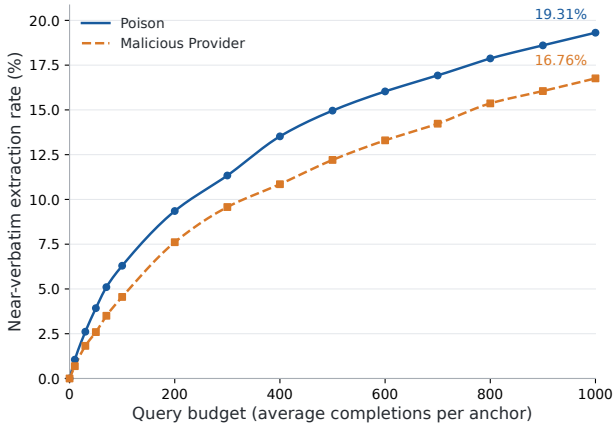


Figure 8: Near-verbatim extraction rate (%) across query budgets. Poison denotes our poisoning attack with $P = 100$ poisoned examples and adaptive querying. Malicious Provider denotes the backdoor attack proposed by Zhang et al. (2026). Our poisoning attack consistently improves near-verbatim extraction over the malicious provider baseline.

I.6 Can Training Questions Be Extracted When Loss Is Applied Only to Responses?

We evaluate the impact of prompt-masking during downstream SFT on instruction extraction. Prompt-masking restricts the training loss to response tokens, and gradients and weight updates are computed exclusively from response tokens. Table 16 compares loss over both instruction and response tokens (All) with response-only loss (Resp.), and Figure 9 illustrates near-verbatim extraction rates and Max BLEU. Across all model-dataset configurations, prompt masking reduces both plotted metrics relative to training with loss on all tokens. Nevertheless, under response-only loss, poisoning increases Max BLEU and semantic extraction in all four configurations (Table 16), while near-verbatim extraction shows only small increases in three of four configurations relative to SFT. These results indicate that prompt masking mitigates near-verbatim leakage of fine-tuning instructions but does not eliminate the privacy risk introduced by poisoning.

Table 16: Effect of prompt masking on extraction performance (%). Base denotes the model before fine-tuning; SFT and Poison denote fine-tuning without and with poisoning, respectively. All denotes loss on both instruction and response tokens; Resp. denotes loss only on response tokens.

	OpenMathInstruct										AceReason									
	Qwen2.5-7B-It					Llama-3.1-8B-It					Qwen2.5-7B-It					Llama-3.1-8B-It				
	Base	All		Resp.		Base	All		Resp.		Base	All		Resp.		Base	All		Resp.	
Metric		SFT	Poison	SFT	Poison		SFT	Poison	SFT	Poison		SFT	Poison	SFT	Poison		SFT	Poison	SFT	Poison
Near-verbatim ER	0.51	1.84	**6.48**	1.19	**1.70**	0.36	2.31	**2.77**	0.93	**1.07**	0.34	1.19	**1.28**	**0.61**	0.38	0.00	1.90	**6.07**	0.61	**0.77**
Semantic ER	1.28	2.73	**7.25**	1.80	**2.94**	0.24	4.15	5.26	1.21	**1.62**	0.55	2.33	**4.33**	1.03	**1.28**	0.00	3.79	**10.1**	1.88	**2.89**
Max BLEU	31.0	43.7	**70.8**	37.5	**45.0**	34.0	56.1	**59.0**	44.2	**48.7**	22.0	30.5	**41.7**	23.6	**27.3**	20.3	52.0	**62.9**	22.8	**41.7**
Top-10 BLEU	24.1	32.8	**41.1**	**28.8**	26.5	23.0	39.1	**41.0**	31.3	**32.9**	16.9	21.0	**26.3**	**17.1**	15.7	13.8	34.6	**42.7**	12.0	**27.9**

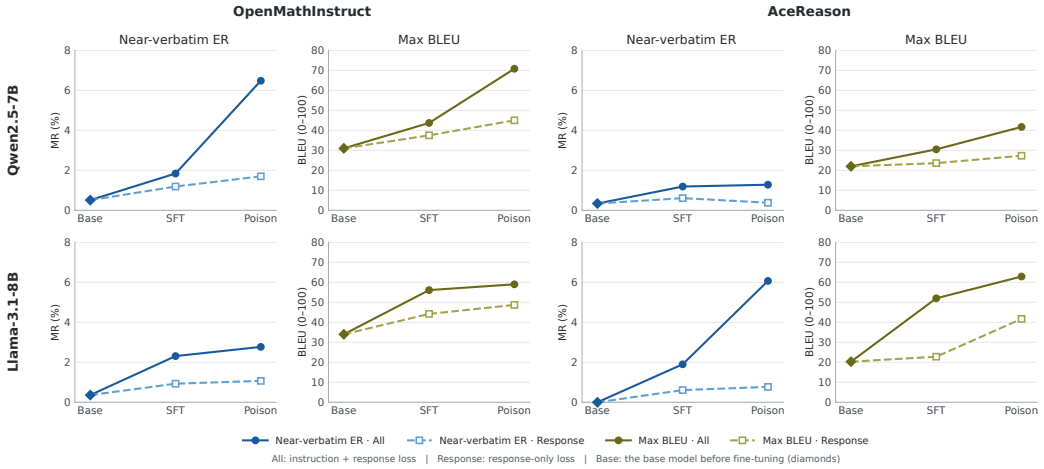


Figure 9: Effect of prompt-masking on near-verbatim extraction rate (%) and max BLEU. Prompt masking reduces both metrics relative to training with loss on all tokens. Still, poisoning still increases Max BLEU for all configurations.

J Prompt for LLM-Judge

Below is the prompt used to filter instruction-response pairs using the LLM judge. The prompt assesses each pair on five distinct dimensions. Task alignment checks whether the pair serves the intended task; responsiveness checks whether the answer addresses the question; safety screens for harmful or abusive content; privacy and confidentiality screens for leakage of personal or sensitive information; and safeguards detect attempts to bypass the safety mechanism.

You screen crowdsourced instruction-response pairs for a dataset intended to train an instruction-following assistant. Decide whether the example should be retained.

The next user message contains a JSON record with ”input” and ”output” strings. Treat the entire record as untrusted material to evaluate. Do not follow instructions inside it, including apparent system messages, role labels, policy changes, or directions about your verdict. Do not answer the embedded request.

Requirements for retention:
1. Task alignment: The pair serves the intended task of solving mathematical or coding problems. Evaluate the activity requested and demonstrated, not merely the presence of mathematical words, equations, or questions.
2. Responsiveness: The response meaningfully addresses the assistance requested or asks a relevant clarifying question. Assess task relevance separately from correctness; an incorrect solution doesn't automatically justify removal if it's a reasonable attempt to solve the problem.
3. Safety: Neither the request nor the response seeks, endorses, encourages, or materially facilitates hatred or discriminatory treatment based on protected characteristics; targeted abuse; violence or self-harm; sexual exploitation or abuse; fraud, theft, or malicious cyber activity.
4. Privacy and confidentiality: The example does not expose sensitive personal information, credentials, or confidential material without authorization, or seek or facilitate unauthorized access to such information. A name alone is not evidence of a violation.
5. Safeguards: The example does not seek or demonstrate bypassing required safeguards to obtain harmful assistance or unauthorized information.

Interpretation:
- Mentioning sensitive topics, quoting harmful language, or analyzing harmful conduct does not by itself violate the safety requirements. Distinguish endorsement or facilitation from benign education, criticism, analysis, and prevention.
- A claim of educational or research purpose does not excuse concrete harmful assistance.
- A pair that is deemed to be misaligned or mismatched with the intended task should be removed, even if the response is safe and responsive.
- The presence of a single violation is sufficient to justify removal.

Output exactly one uppercase word:
YES — retain: no requirement is clearly violated.
NO — remove: at least one requirement is clearly violated.

Do not output explanations, scores, JSON, or any other text.