

WILDTRACE: EVALUATING TRAINING DATA DETECTION IN REALISTIC LARGE-SCALE LLM POST-TRAINING

Hengxiang Zhang¹, Junxian Liu¹, Hongxin Wei^{1*}

¹Department of Statistics and Data Science, Southern University of Science and Technology

ABSTRACT

Although post-training significantly enhances model performance, the use of massive undisclosed data raises concerns about benchmark contamination and copyright infringement. This drives growing interest in training data detection for post-trained models. Existing evaluations for training data detection often leverage benchmarks constructed post hoc, which can introduce distribution shifts. Alternatively, controlled evaluations rely on small-scale training setups that diverge from realistic post-training settings. As a result, their reported performance may not faithfully reflect the effectiveness of detection methods in real-world scenarios. To this end, we introduce WILDTRACE, a reliable benchmark designed to evaluate detection methods in realistic, large-scale post-training settings. Built on a multi-stage pipeline, WildTrace encompasses stage-specific models throughout the post-training lifecycle, spanning SFT, DPO, and RL. To perform a reliable evaluation, we construct model-specific evaluation sets across four diverse tasks, featuring distribution-matched members and non-members with verifiable membership labels. Our results reveal that even the best-performing method achieves an average AUC of only 0.529 on our benchmark, which approaches random-guessing level (0.5). Furthermore, our analysis suggests that evaluation protocols can significantly affect measured detection performance. These findings highlight the significant potential for developing effective, practical, and robust detection methods under realistic scenarios.

1 INTRODUCTION

Modern Large Language Models (LLMs) exhibit remarkable capabilities across diverse tasks, owing to advances in post-training techniques (Kumar et al., 2026; Shao et al., 2024; Yang et al., 2025). The post-training stage typically involves large-scale Supervised Fine-Tuning (SFT), Direct Preference Optimization (DPO), and Reinforcement Learning (RL). Although multi-stage post-training enhances model performance, the use of massive training data raises critical concerns about benchmark contamination (Xu et al., 2024; Choi et al., 2025) and copyright infringement (Slack & Moubayed, 2025; Li et al., 2026). This underscores the need to audit training data from post-training stages. However, the effectiveness of existing detection methods under realistic, large-scale post-training settings remains underexplored.

Previous work often relies on benchmarks constructed post hoc (e.g., WikiMIA) for evaluation (Liu et al., 2024; Shi et al., 2024). However, this post-hoc construction can introduce distribution shifts between members and non-members, compromising the validity of evaluation (Duan et al., 2024; Das et al., 2024). Although recent work mitigates this by employing distribution-matched members and non-members, their experimental setups diverge from realistic post-training settings. Specifically, these controlled evaluations typically perform multi-epoch training on a small, domain-specific dataset, whereas realistic post-training leverages large-scale, diverse data for only one or a few epochs (Fu et al., 2024; Olmo et al., 2025). Additionally, existing work often employs in-house evaluation with different models, datasets, and evaluation protocols, hindering fair comparison and

*Corresponding author

reproducibility. Therefore, a standardized benchmark is required to reliably evaluate training data detection under large-scale post-training scenarios.

To conduct a practical and reliable evaluation, an ideal benchmark for post-training data detection should adhere to three key principles: 1) *Distribution-matched sampling with verifiable membership*: Members and non-members are drawn from the same distributions with verifiable membership, avoiding distribution shifts and unreliable membership labels in post-hoc constructions. 2) *Realistic large-scale post-training setup*: Target models are trained on diverse, large-scale datasets using standard post-training practices, establishing an evaluation testbed that faithfully reflects real-world deployment scenarios. 3) *Reproducible evaluation across post-training lifecycle*: The benchmark provides publicly available evaluation suites consisting of target models and model-specific evaluation sets at each training stage, ensuring reproducible evaluation.

To this end, we introduce WildTrace, a reliable benchmark for evaluating training data detection in realistic, large-scale post-training settings. Leveraging the fully open Olmo-3 model, we perform a large-scale, multi-stage post-training pipeline ($SFT \rightarrow DPO \rightarrow RL$). This produces three stage-specific post-trained models, each with transparent training data and verifiable membership labels. To perform reliable evaluation, we construct model-specific evaluation sets across diverse tasks (Math, Coding, IF, and QA), where members and non-members are sampled from the same underlying distribution. To facilitate research under realistic settings, our benchmark provides a reproducible testbed consisting of publicly available post-trained models and evaluation sets.

Using WildTrace, we conduct a comprehensive evaluation of training data detection methods across the LLM post-training lifecycle. Our evaluation reveals that under large-scale post-training settings, existing detection methods exhibit limited effectiveness across all three stages—SFT, DPO, and RL. Our analysis further indicates that scaling up training data generally degrades detection performance. Consequently, high detection performance observed in small-scale training settings does not reflect actual performance under large-scale post-training settings. This evidence highlights the need for developing more effective detection methods under large-scale post-training scenarios.

Beyond these findings, we revisit how evaluation protocols shape the assessment of training data detection. First, empirical results indicate that evaluating training data detection on benchmarks with distribution shift may inflate detection performance, obscuring their actual performance in the wild. Second, achieving high detection accuracy often requires calibrating decision thresholds with an in-distribution validation set, which is rarely available in real-world deployment. Additionally, while reference-based methods can improve detection performance, they still fall short of effective detection in large-scale training settings. Overall, these findings reveal a significant gap between current evaluation practices and realistic deployment, highlighting the need for more robust detection methods and practical evaluation protocols.

2 PRELIMINARIES

2.1 TRAINING DATA DETECTION FOR LLMs

Training data detection for LLMs is an instance of membership inference (Shokri et al., 2017; Shi et al., 2024), which aims to identify whether a given text is used during model training. In contrast to pretraining data detection, which identifies data in the pre-training corpus, post-training data detection aims to determine whether a sample is used at a particular post-training phase. Let $\mathcal{M}$ denote the model post-trained on the dataset $\mathcal{D}$. Given a piece of text x , the objective is to predict whether the datapoint belongs to the training set. A detection method computes a membership score $S(x; \mathcal{M})$ derived from the predicted token probability of the model and makes a binary prediction:

$$h(x; \mathcal{M}) = \begin{cases} 1, & \text{if } S(x; \mathcal{M}) < \lambda, \\ 0, & \text{if } S(x; \mathcal{M}) \geq \lambda, \end{cases} \quad (1)$$

where $h(x; \mathcal{M}) = 1$ indicates *member* and $h(x; \mathcal{M}) = 0$ indicates *non-member*, with λ being a decision threshold.

2.2 LLMs POST-TRAINING

Building on a pre-trained model, developers typically refine its behavior through a multi-stage post-training process. Modern LLM post-training recipes typically involve three key paradigms: supervised fine-tuning (SFT), direct preference optimization (DPO), and reinforcement learning (RL). We introduce three widely used LLM post-training techniques in detail.

Supervised Fine-Tuning Supervised Fine-Tuning aims to teach models to follow human instructions and generate responses with the desired format and style. To achieve this goal, a pre-trained model is further trained on a constructed high-quality dataset $\mathcal{D}$. Let q denote a question and $y = \{y_1, y_2, \dots, y_N\}$ denote the corresponding target response. Using q as a prompt, SFT explicitly teaches the model to imitate the well-crafted target response. Formally, the training objective can be formulated as

$$\mathcal{L}_\theta = -\mathbb{E}_{(q,y) \sim \mathcal{D}} \sum_{t=1}^N \log \pi_\theta(y_t \mid y_{<t}, q), \quad (2)$$

where $\pi_\theta(y_t \mid y_{<t}, q)$ denotes the model’s predicted probability for token y_t , given the preceding tokens. In practice, the SFT loss is typically computed only over response tokens, with the prompt serving as context.

Direct Preference Optimization Direct preference optimization is typically applied after SFT to further align the model with human preferences, using a preference dataset $\mathcal{D}$ consisting of triplets (q, y^+, y^-) . Concretely, DPO directly optimizes the policy to assign a higher likelihood to the preferred response y^+ over a less-preferred response y^- , which explicitly encodes the human’s preference in models’ parameters. Formally, the DPO loss is defined as follows:

$$\mathcal{L}^{\text{DPO}}(\theta) = -\mathbb{E}_{(q,y^+,y^-) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y^+ \mid q)}{\pi_{\text{ref}}(y^+ \mid q)} - \beta \log \frac{\pi_\theta(y^- \mid q)}{\pi_{\text{ref}}(y^- \mid q)} \right) \right], \quad (3)$$

where π_θ denotes the trainable policy, π_{ref} denotes a reference policy, and $\sigma(\cdot)$ is the sigmoid function. The hyperparameter β controls the deviation of π_θ from π_{ref} .

Reinforcement Learning Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is a widely used RL algorithm for LLM training. GRPO improves training efficiency over standard PPO by eliminating the critic model. Specifically, for each question q , GRPO samples a group of G responses $\{o_1, o_2, \dots, o_G\}$ from the old policy π_{old} , and estimates each response’s advantage from the relative rewards within the group. Given the reward r_i for each response o_i , the advantage is computed as:

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\{r_1, r_2, \dots, r_G\})}{\text{std}(\{r_1, r_2, \dots, r_G\})}. \quad (4)$$

By using $\hat{A}_{i,t}$ as the advantage estimate, GRPO then minimizes a loss combining a clipped surrogate term with a KL penalty toward a reference policy π_{ref} :

$$\mathcal{L}_\theta = -\mathbb{E}_{q, \{o_i\}_{i=1}^G} \frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left[\min(\rho_{i,t}(\theta) \hat{A}_{i,t}, \text{clip}(\rho_{i,t}(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{i,t}) - \beta \mathbb{D}_{\text{KL}}[\pi_\theta \parallel \pi_{\text{ref}}] \right], \quad (5)$$

where $\rho_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\text{old}}(o_{i,t} \mid q, o_{i,<t})}$ is the importance sampling ratio, ϵ is the clipping threshold, and β controls the strength of the KL regularization.

3 WILDTRACE: BENCHMARKING TRAINING DATA DETECTION FROM POST-TRAINED LLMs

Existing evaluations for training data detection often rely on post-hoc benchmarks or small-scale training settings, which fail to reflect the effectiveness of detection methods in the wild. To bridge this gap, our work aims to systematically evaluate training data detection under more realistic settings. In this section, we first identify the design principles of a reliable benchmark. We then introduce WildTrace, a comprehensive benchmark for reliable evaluation of training data detection.

3.1 BENCHMARK DESIGN PRINCIPLES

Distribution-matched sampling with verifiable membership Conventional evaluation protocols often rely on post-hoc benchmark construction, treating documents published before a model’s training cutoff date as members and those published after as non-members (Shi et al., 2024; Yi & Li, 2026; Liu et al., 2024). However, temporal splits can introduce inadvertent artifacts (e.g., timestamps in WikiMIA) that cause distributional shifts between members and non-members (Duan et al., 2024; Das et al., 2024). Detection methods may exploit these artifacts rather than genuine membership signals, undermining the validity of the benchmark. *A reliable benchmark requires two requirements: (1) members and non-members are drawn from the same data distributions, and (2) membership labels are verifiable rather than heuristically inferred post hoc.*

Realistic large-scale post-training settings To mitigate distribution shifts, controlled evaluation protocols train target models on tailored datasets to obtain ground-truth membership labels (Fu et al., 2024; Tao et al., 2026). However, these evaluation protocols commonly use small, domain-specific datasets trained for multiple epochs, whereas practical LLM training typically uses large-scale, diverse datasets with only one or a few epochs. Small training corpora and repeated exposure can strengthen the model’s memorization of individual samples, yielding optimistic detection performance. Such idealized evaluations may overestimate the detectability of training data and do not reflect performance of detection methods under realistic post-training settings. *Consequently, a rigorous benchmark for training data detection should employ target models trained under realistic, large-scale training settings.*

Reproducible evaluation across post-training lifecycle While pre-training data detection has been extensively explored (Shi et al., 2024; Zhang et al., 2025b), existing post-training studies typically focus on a single stage (Fu et al., 2024; Tao et al., 2026), such as supervised fine-tuning. These works use diverse models, datasets, and evaluation protocols and rarely release their evaluation artifacts (e.g., trained models), hindering reproducibility and fair comparison. Furthermore, post-training typically involves a multi-stage pipeline ($SFT \rightarrow DPO \rightarrow RL$), necessitating comprehensive evaluation throughout post-training lifecycle. Existing benchmarks fall short of providing a unified, reproducible framework for evaluating training data detection across the post-training lifecycle of LLMs. *A rigorous benchmark should provide publicly available oracle models and evaluation sets for each post-training stage, enabling reproducible evaluation.*

3.2 BENCHMARK CURATION

Building on the proposed three principles, we construct WildTrace by executing a realistic, large-scale post-training pipeline from scratch, which produces trained target models and corresponding evaluation sets at each stage.

Multi-stage oracle model training We build on the OLMO-3 family (Olmo et al., 2025), a fully open model family that releases models’ training data, at a scale competitive with modern open LLMs. This transparency allows us to construct a benchmark with verifiable membership labels under a large-scale post-training setting. Starting from the base checkpoint *Olmo-3-1025-7B*, we perform a large-scale, multi-stage post-training pipeline ($SFT \rightarrow DPO \rightarrow RL$), with each stage trained from the model of the preceding stage. We execute large-scale training using a diverse dataset of over 2.5 million examples, spanning 2.15M question-response pairs for SFT, 260K preference pairs for DPO, and 170K prompts for RL, conducted on a cluster of 32 NVIDIA A100 GPUs. Through this construction process, we obtain three post-trained models, denoted ORACLE-SFT, ORACLE-DPO, and ORACLE-RL, which serve as oracle models for evaluating detection methods. More details of the model training are provided in Appendix B.1.

Model-specific evaluation set construction To construct a benchmark spanning the full post-training lifecycle, we establish a dedicated detection testbed using the models produced at each post-training stage. Considering the diversity of training corpora, we curate four tasks for our benchmark: mathematics (Math), coding (Code), instruction following (IF), and general question answering (QA). Specifically, for each task, we first select 250 samples from the original dataset as non-members, which are excluded from the training process. The remaining examples constitute the

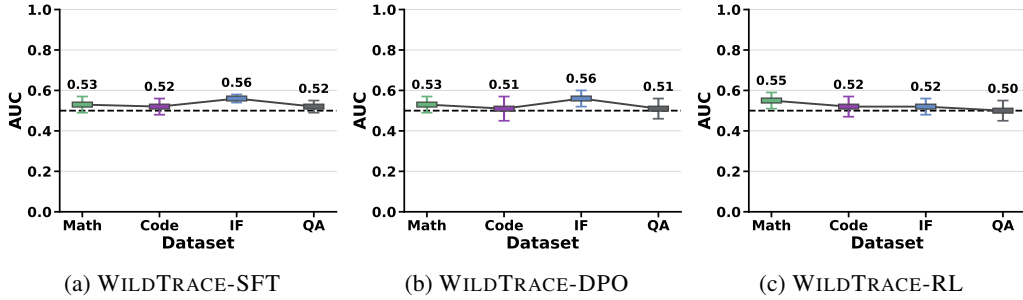


Figure 1: The AUC of blind baselines across tasks and post-training stages. For each task across post-training stages (a–c), we report the highest AUC achieved among nine blind baselines. The dashed line indicates chance-level performance (AUC = 0.50). The near-random detection performance across all evaluation sets indicates the rigor and reliability of our benchmark construction.

training set, from which we further sample 250 samples per task as members. This guarantees that members and non-members are drawn from the same underlying distribution and that membership labels are verifiable. Consequently, this process yields three evaluation sets paired with their oracle models: WILDTRACE-SFT paired with ORACLE-SFT, WILDTRACE-DPO paired with ORACLE-DPO, and WILDTRACE-RL paired with ORACLE-RL, each set containing four tasks. The details of the benchmark are provided in Appendix B.2.

3.3 VALIDATING OUR WILDTRACE WITH BLIND BASELINES

To verify the reliability of WildTrace, we test whether members and non-members can be distinguished using blind baselines that rely solely on text features and do not access the target model (Das et al., 2024; Wang et al., 2026). Specifically, we evaluate blind baseline approaches by combining three text feature extractors (Bag-of-Words, TF-IDF, Word2Vec) with three classifiers (Logistic Regression, Random Forest, Neural Network), resulting in nine attack combinations. Figure 1 shows the best AUC among the nine blind baselines across different post-training stages. The blind baselines exhibit near-random performance, with average AUCs across tasks of 0.53, 0.53, and 0.52 on WILDTRACE-SFT, WILDTRACE-DPO, and WILDTRACE-RL, respectively. These results show that blind baselines obtain negligible signals for distinguishing members from non-members on our benchmark, validating the reliability of our benchmark. We provide additional experimental results in Appendix C.3, further confirming the validity of our benchmark.

4 EXPERIMENT

4.1 EXPERIMENT SETUP

Detection methods To investigate the performance of existing detection methods on our benchmark, we evaluate commonly used detection methods for LLMs. We categorize these methods into two classes based on their scoring signal: *input-based methods* and *output-based methods*. Specifically, *input-based methods* compute membership scores based on the likelihood of input tokens, including *Perplexity* (Li, 2023), *Entropy* (Song & Mittal, 2021), *Zlib* (Carlini et al., 2021), *Lower-case* (Carlini et al., 2021), *Neighbor* (Mattern et al., 2023), *Min-K%* (Shi et al., 2024), and *Min-K%++* (Zhang et al., 2025c). *Output-based methods* leverage tokens generated by the model as the detection signal, including *Min-kNN* (Zhang et al., 2026), *Self-Critique* (Tao et al., 2026), *Generated Perplexity*, *Generated Entropy* and *Generated Min-K%*. We provide detailed descriptions and implementation details for each method in Appendix B.4.

Implementation details In our experiments, for *input-based methods*, we compute membership scores on input tokens. For *output-based methods*, we condition on the question to generate responses and compute scores on the generated tokens. Unless otherwise specified, we set the maximum sequence length to 2048 tokens. During the detection phase, we assume access to the complete information of each sample. Further evaluation details are provided in Appendix B.3. We evaluate

Table 1: AUC of detection methods on our benchmark. Reported results are averaged over three runs with different random seeds. *Input-based methods* compute scores using input tokens, whereas *Output-based methods* score output tokens. Avg. denotes the AUC of each method, averaged across all tasks and post-trained models. **Bold** indicates the best result.

Method	ORACLE-SFT				ORACLE-DPO				ORACLE-RL				Avg.
	Math	Code	IF	QA	Math	Code	IF	QA	Math	Code	IF	QA	
Input-based methods													
Perplexity	0.499	0.565	0.535	0.559	0.500	0.451	0.527	0.514	0.560	0.487	0.516	0.456	0.514
Entropy	0.482	0.502	0.483	0.472	0.524	0.447	0.487	0.488	0.512	0.488	0.500	0.475	0.488
Lowercase	0.515	0.589	0.544	0.581	0.498	0.438	0.514	0.523	0.521	0.491	0.534	0.471	0.518
Zlib	0.469	0.540	0.572	0.570	0.466	0.538	0.526	0.465	0.539	0.488	0.501	0.484	0.513
Neighbor	0.533	0.537	0.497	0.555	0.515	0.429	0.506	0.544	0.542	0.503	0.499	0.488	0.512
MIN-K%	0.499	0.570	0.541	0.548	0.496	0.459	0.535	0.519	0.552	0.486	0.517	0.448	0.514
MIN-K%++	0.510	0.610	0.583	0.618	0.504	0.543	0.473	0.544	0.529	0.437	0.528	0.470	0.529
Output-based methods													
Generated Perplexity	0.509	0.513	0.481	0.521	0.520	0.481	0.544	0.488	0.536	0.499	0.459	0.501	0.504
Generated Entropy	0.505	0.514	0.475	0.526	0.521	0.481	0.534	0.488	0.529	0.497	0.456	0.501	0.502
Generated MIN-K%	0.510	0.516	0.482	0.519	0.519	0.480	0.547	0.498	0.538	0.502	0.467	0.499	0.506
Min-kNN	0.519	0.532	0.479	0.523	0.525	0.452	0.512	0.528	0.571	0.497	0.469	0.495	0.509
Self-Critique	0.479	0.490	0.557	0.541	0.484	0.473	0.509	0.509	0.524	0.496	0.495	0.492	0.504

the effectiveness of existing detection methods on our WildTrace by measuring the following metrics: (1) AUC, the area under the receiver operating characteristic curve; (2) TPR@5%FPR, the true positive rate at a 5% false positive rate (Carlini et al., 2022). Instead of paying equal attention to members and non-members, this metric pays more attention to members and evaluates whether one can confidently identify members.

4.2 MAIN RESULTS

Detecting training data remains challenging under large-scale post-training scenarios. Table 1 reports the AUC of detection methods across diverse post-trained models from different post-training stages, including SFT, DPO, and RL training paradigms. The experimental results show that most detection methods achieve AUC scores near random guess on our benchmark. The TPR@5%FPR of the detection methods is provided in Table 9 in Appendix C.5. This finding indicates that existing detection methods struggle to effectively identify training data under large-scale post-training scenarios. We present the results of reference-based methods in Appendix C.1, which yields consistent findings. Compared to DPO and RL, we find that training data from the SFT stage is more detectable. For example, the best-performing method MIN-K%++ achieves an AUC of 0.618 on QA under ORACLE-SFT, but only 0.544 and 0.470 under ORACLE-DPO and ORACLE-RL, respectively. This gap may arise from differences in the training objectives, which rely on distinct supervision signals. SFT maximizes the likelihood of target responses conditioned on the question, leaving a strong signal for detection. In contrast, DPO optimizes relative likelihood between preferred and less-preferred responses, while RL maximizes reward over its online sampled rollouts. Both objectives leave a weaker memorization signal, making their training data less amenable to detection. Overall, these findings indicate that current detection methods exhibit limited effectiveness under large-scale post-training scenarios, necessitating more capable detection methods.

Strong detection performance in small-scales does not transfer to large-scale training. Existing training data detection evaluations are typically conducted in small-scale training settings, misaligned with realistic LLM post-training settings. To examine how training scales affect detection performance, we train the base model on datasets ranging from 10K to 200K examples, sampled from Dolci-Instruct-SFT (Olmo et al., 2025). In our experiments, all training configurations are held constant except for the training set size. Across all training scales, we include all benchmark members in the training set and evaluate performance of the detection method on WildTrace. All of our evaluation results are obtained using MIN-K%++ detection method, which shows the strongest performance in Table 1. Figures 2a and 2b report AUC and TPR@5%FPR across four tasks. The

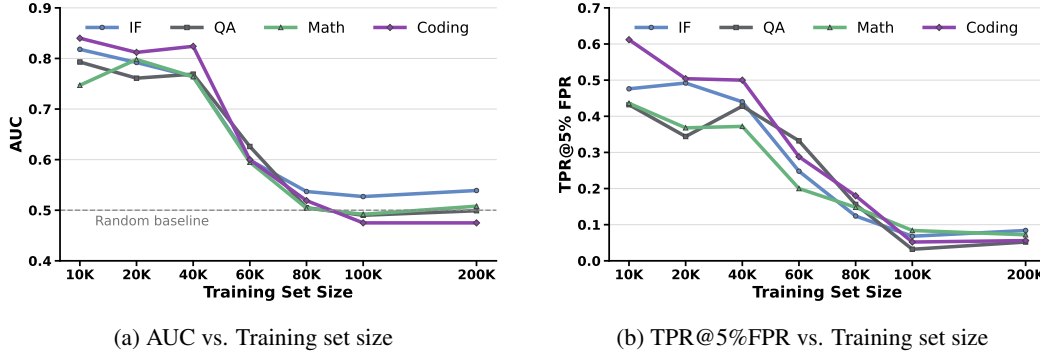


Figure 2: Effect of training set size on detection performance. Both AUC and TPR@5% FPR degrade as the training data scale increases, with AUC ultimately approaching to the random baseline.

results show that detection performance degrades as training scales up, with AUC approaching the random guess level and TPR@5%FPR falling below 0.1 at 100K samples. These findings reveal that the detection performance evaluated in small-scale training settings does not transfer to large-scale practical training settings. This highlights the importance of evaluating detection methods in realistic, large-scale training settings, as our WildTrace benchmark provides.

5 REVISITING TRAINING DATA DETECTION FROM LLMs

Training data detection has attracted growing attention as a measure for auditing data contamination, privacy, and model memorization in LLMs. However, the evaluation protocol substantially affects detection performance in practice. In this section, we examine key evaluation settings that are critical for effectively detecting training data in realistic post-training scenarios.

5.1 IMPACT OF EVALUATION SETTINGS ON DETECTION PERFORMANCE

Setup Our experiments in Section 4.2 show that current detection methods exhibit limited detection performance under large-scale post-training settings. To better observe performance differences across diverse evaluation settings, we conduct experiments under small-scale training settings. We fine-tune the Olmo 3 base model on 10K examples from Dolci-Instruct-SFT, resulting in ORACLE-SFT-10K. Detailed implementations are provided in Appendix B.5. In this section, unless otherwise specified, our evaluations are conducted under this evaluation configurations.

The composition of evaluation benchmarks significantly affects evaluation results. Training data detection is commonly evaluated on a dataset consisting of members and non-members, yet the composition of this evaluation set can affect the evaluation results. To examine this effect, we evaluate detection methods across evaluation sets with different distribution shifts between members and non-members. Figure 3a shows the AUC of the detection method for detecting coding training data when non-members come from different domains. The results show that the detection method achieves inflated AUC scores on the coding task when non-members are drawn from other domains, with performance gains of up to +21.2% compared to in-domain evaluation. To further illustrate this effect, we construct evaluation sets where *both* groups are non-members (i.e., neither was used for training). As shown in Figure 3b, detection methods still achieve high AUC scores—up to 0.908 for QA vs. Coding, substantially above the chance level of 0.5. This inauthentic detection performance stems from domain-specific distribution shifts between the two non-member groups, resulting in separable detection score distributions, as shown in Appendix C.2. These results indicate that distribution shifts enable detection methods to achieve superior performance by exploiting superficial artifacts rather than true membership signals. However, post-hoc benchmark construction often cannot ensure that members and non-members are drawn from the same underlying distribution, which is not an ideal evaluation setup. A reliable benchmark requires minimizing distribution shifts between members and non-members, as shown in Appendix C.3.

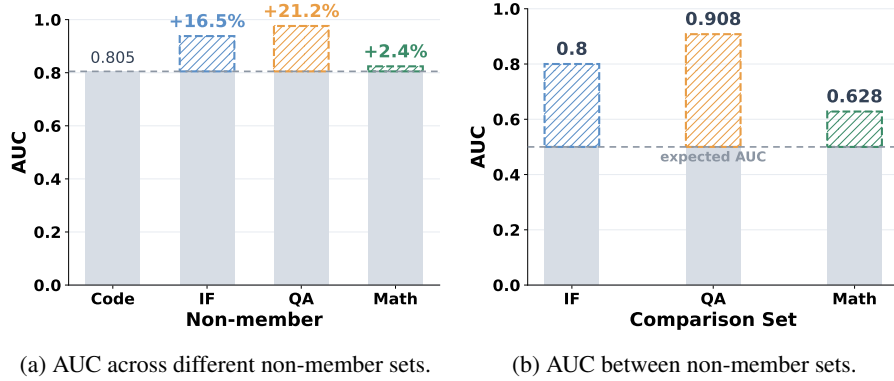


Figure 3: The AUC scores of the perplexity-based method under distribution shift. (a) AUC for code members and non-members across domains. Percentages show inflated improvement relative to the in-domain code evaluation set. (b) AUC across domains for non-member vs. non-member comparisons, using code as the reference domain.

Table 2: The accuracy of detection methods on the Code task under different threshold calibration approaches. Thresholds are calibrated using ID validation data (Code) and OOD validation data (IF, QA, and Math), respectively. Each cell reports the mean $\pm$ standard deviation over 10 runs, where different random seeds yield different validation sets. **Bold** indicates the best result.

Threshold Calibration		Methods						Avg.
Setting	Set / Method	Perplexity	Entropy	Lowercase	Zlib	MIN-K %	MIN-K %++	
OOD	IF	0.65 $\pm$ 0.06	0.72 $\pm$ 0.03	0.64 $\pm$ 0.03	0.64 $\pm$ 0.03	0.66 $\pm$ 0.06	0.79 $\pm$ 0.01	0.68
	QA	0.66 $\pm$ 0.03	<u>0.74 $\pm$ 0.03</u>	0.66 $\pm$ 0.04	0.66 $\pm$ 0.02	<u>0.68 $\pm$ 0.03</u>	0.75 $\pm$ 0.05	<u>0.69</u>
	Math	0.63 $\pm$ 0.05	0.65 $\pm$ 0.02	<u>0.67 $\pm$ 0.02</u>	<u>0.65 $\pm$ 0.03</u>	0.63 $\pm$ 0.05	0.79 $\pm$ 0.01	0.67
Unsupervised	GMM	0.50 $\pm$ 0.00	0.55 $\pm$ 0.00	0.63 $\pm$ 0.00	0.55 $\pm$ 0.00	0.71 $\pm$ 0.00	0.66 $\pm$ 0.00	0.60
	GMM _{log1p}	0.61 $\pm$ 0.00	0.56 $\pm$ 0.00	0.64 $\pm$ 0.00	0.55 $\pm$ 0.00	0.67 $\pm$ 0.00	0.76 $\pm$ 0.00	0.63
	BlindMI	0.74 $\pm$ 0.00	0.67 $\pm$ 0.00	0.66 $\pm$ 0.00	0.50 $\pm$ 0.00	0.50 $\pm$ 0.00	<u>0.77 $\pm$ 0.00</u>	0.64
ID	Code	<u>0.72 $\pm$ 0.02</u>	0.76 $\pm$ 0.01	0.68 $\pm$ 0.02	0.66 $\pm$ 0.01	0.71 $\pm$ 0.02	0.79 $\pm$ 0.01	0.72

Superior detection accuracy requires calibrating decision threshold on an ID validation set.

In a real-world scenario, deploying a detection algorithm often requires a preset decision threshold. Existing studies typically assume access to a validation set drawn from the same distribution as the evaluation set to calibrate this threshold. To assess the impact of the validation set on detection accuracy, we compare three threshold calibration settings: an in-distribution (ID) validation set, an out-of-distribution (OOD) validation set, and no validation set (e.g., using BlindMI (Hui et al., 2021)). Further details of the implementation are provided in Appendix B.6. The results in Table 2 show that calibrating thresholds using an ID validation set can yield the best detection performance. In contrast, calibrating thresholds using an OOD validation set degrades detection performance. In the absence of a validation set, unsupervised threshold calibration methods achieve an average accuracy of up to 0.64, but remain inferior to supervised or oracle strategies. These findings show that superior detection accuracy often requires calibrating the threshold with an ID validation set. However, in practice, obtaining a validation set aligned with the distribution of the test set is rarely feasible. This gap between the evaluation assumptions and the practical implementation motivates future research to focus on more effective threshold calibration strategies.

Does the performance of the reference-based method rely on the choice of reference model?

Reference-based detection methods typically achieve the strongest detection performance among existing approaches. However, they often assume access to the exact base model used to train the target model. In practice, this assumption is rarely satisfied, as model developers typically do not disclose the base model of the post-trained model. To investigate how the choice of reference model affects detection performance, we evaluate reference-based methods using a range of reference models (e.g., Qwen and Llama families). Table 3 compares the AUC of the reference-based method using

Table 3: AUC of the reference-based method using Min-K% across different reference models under 10K and 100K training data settings. **Bold** indicates the superior result within each base-instruct model pair.

Reference Models		ORACLE-SFT-10K					ORACLE-SFT-100K				
Model	Type	IF	QA	Math	Coding	Avg.	IF	QA	Math	Coding	Avg.
Reference-free	–	0.82	0.79	0.75	0.84	0.80	0.53	0.49	0.49	0.48	0.50
Olmo-3-7B-Instruct	Instruct	0.70	0.86	0.74	0.82	0.78	0.63	0.75	0.69	0.71	0.69
Olmo-3-1025-7B	Base	0.83	0.85	0.75	0.87	0.82	0.71	0.75	0.71	0.74	0.73
Qwen2.5-7B-Instruct	Instruct	0.75	0.82	0.72	0.85	0.78	0.67	0.73	0.69	0.74	0.71
Qwen2.5-7B	Base	0.84	0.86	0.75	0.87	0.83	0.71	0.75	0.71	0.73	0.73
Qwen3-1.7B	Instruct	0.71	0.78	0.71	0.73	0.73	0.64	0.70	0.67	0.66	0.67
Qwen3-1.7B-Base	Base	0.83	0.84	0.74	0.86	0.82	0.71	0.74	0.71	0.73	0.72
Llama-3.2-3B-Instruct	Instruct	0.82	0.84	0.75	0.84	0.81	0.71	0.76	0.71	0.73	0.73
Llama-3.2-3B	Base	0.84	0.85	0.76	0.86	0.83	0.71	0.75	0.71	0.72	0.72

different reference models and a reference-free baseline on ORACLE-SFT models trained with 10K and 100K examples. *First, while reference-based methods can enhance detection performance, their effectiveness depends on the choice of the reference model.* Using base models as reference models often achieves better performance than instruct models, as base models are usually trained only on pretraining corpora and thus provide a cleaner baseline for measuring the likelihood shift of training data. *Second, the reference-based method’s performance degrades significantly as the training data scale increases,* achieving an average AUC of 0.83 with ORACLE-SFT-10K but only 0.73 with ORACLE-SFT-100K. The additional results in Appendix C.1 show that reference-based methods do not effectively detect training data under large-scale post-training scenarios. These results demonstrate that while reference-based methods benefit from a suitable reference model, they still yield marginal gains in large-scale training settings.

6 CONCLUSION

In this work, we introduce WildTrace, a reliable benchmark designed to evaluate training data detection under realistic, large-scale post-training settings. Our benchmark is built from scratch by performing large-scale post-training in the SFT, DPO, and RL stages, with evaluation sets drawn from the same data distributions to mitigate potential distribution shifts. Our comprehensive evaluation demonstrates that current detection methods exhibit limited effectiveness at scale. Importantly, we find that performance observed in small-scale settings does not generalize to large-scale post-training, underscoring the critical need to evaluate methods under realistic scenarios. These findings motivate the development of detection approaches that are more effective, practical, and robust under modern post-training settings. Overall, we hope WildTrace can serve as a solid testbed for advancing reliable and deployable training data detection in LLM post-training.

Limitations Our benchmark aims to provide a practical and reliable testbed for research on training data detection. To ensure accurate ground-truth membership without dataset contamination, our WildTrace is built on the Olmo-3 family, a fully open LLM family whose competitive performance aligns with state-of-the-art open LLMs. Although WildTrace offers a practical testbed, extending our evaluation protocol to other models and scales warrants future exploration. Furthermore, while WildTrace covers standard post-training pipelines (SFT, DPO, and RL), emerging paradigms like on-policy distillation can be incorporated in future work to further broaden our evaluations.

REFERENCES

- 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 Tramer. Membership inference attacks from first principles. In *2022 IEEE symposium on security and privacy (SP)*, pp. 1897–1914. IEEE, 2022.
- Hyeong Kyu Choi, Maxim Khanov, Hongxin Wei, and Yixuan Li. How contaminated is your benchmark? measuring dataset leakage in large language models with kernel divergence. In *International Conference on Machine Learning*, 2025.
- Debeshee Das, Jie Zhang, and Florian Tramèr. Blind baselines beat membership inference attacks for foundation models. *arXiv preprint arXiv:2406.16201*, 2024.
- Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. Do membership inference attacks work on large language models? *arXiv preprint arXiv:2402.07841*, 2024.
- 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. *Advances in Neural Information Processing Systems*, 37:134981–135010, 2024.
- Bo Hui, Yuchen Yang, Haolin Yuan, Philippe Burlina, Neil Zhenqiang Gong, and Yinzhi Cao. Practical blind membership inference attack via differential comparisons. In *Proceedings of the Network and Distributed System Security Symposium (NDSS’21)*, February 2021.
- Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Fahad Shahbaz Khan, and Salman Khan. Llm post-training: A deep dive into reasoning large language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.
- Yanming Li, Cédric Eichler, Nicolas Anciaux, Alexandra Bensamoun, Lorena Gonzalez-Manzano, and Seifeddine Ghozzi. Data provenance auditing of fine-tuned large language models with a text-preserving technique. In *Forty-third International Conference on Machine Learning*, 2026. URL <https://openreview.net/forum?id=3D0x8oxOYH>.
- Yucheng Li. Estimating contamination via perplexity: quantifying memorisation in language model evaluation. *arXiv preprint arXiv:2309.10677*, 2023.
- Zhenhua Liu, Tong Zhu, Chuanyuan Tan, Bing Liu, Haonan Lu, and Wenliang Chen. Probing language models for pre-training data detection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1576–1587, 2024.
- Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schoelkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. Membership inference attacks against language models via neighbourhood comparison. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 11330–11343, 2023.
- Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

- 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 Computer Society, 2017.
- Dean L. Slack and Noura Al Moubayed. Early detection and reduction of memorisation for domain adaptation and instruction tuning, 2025. URL <https://arxiv.org/abs/2510.11372>.
- Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *30th USENIX security symposium (USENIX security 21)*, pp. 2615–2632, 2021.
- Yongding Tao, Tian Wang, Yihong Dong, Huanyu Liu, Kechi Zhang, Hu XiaoLong, and Ge Li. Detecting data contamination from reinforcement learning post-training for large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL <https://openreview.net/forum?id=EjiJmiA6ea>.
- Jeffrey George Wang, Jason Wang, Marvin Li, and Seth Neel. Checkmiabench: Firm foundations for membership inference attacks on language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 364–370, 2026.
- Cheng Xu, Shuhao Guan, Derek Greene, M Kechadi, et al. Benchmark data contamination of large language models: A survey. *arXiv preprint arXiv:2406.04244*, 2024.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- Jiatong Yi and Yanyang Li. Membership inference on llms in the wild. *arXiv preprint arXiv:2601.11314*, 2026.
- Hengxiang Zhang, Hyeon Kyu Choi, Sharon Li, and Hongxin Wei. Detecting distillation data from reasoning models. *arXiv preprint arXiv:2510.04850*, 2025a.
- Hengxiang Zhang, Songxin Zhang, Bingyi Jing, and Hongxin Wei. Fine-tuning can help detect pretraining data from large language models. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL <https://openreview.net/forum?id=X8dzvdkQwO>.
- Hongbo Zhang, Yue Yang, Jianhao Yan, Guangsheng Bao, and Yue Zhang. Detecting rlvr training data via structural convergence of reasoning. *arXiv preprint arXiv:2602.11792*, 2026.
- Jingyang Zhang, Jingwei Sun, Eric Yeats, Yang Ouyang, Martin Kuo, Jianyi Zhang, Hao Frank Yang, and Hai Li. Min-k%++: Improved baseline for pre-training data detection from large language models. In *The Thirteenth International Conference on Learning Representations*, 2025c. URL <https://openreview.net/forum?id=ZGkfoufDaU>.

Appendix

Table of Contents

A Related Work	12
B Experimental Details	13
B.1 Model training configurations	13
B.2 Details of training datasets and evaluation sets	13
B.3 Evaluation protocols	14
B.4 Training data detection methods for LLMs	14
B.5 Implementation of small-scale training settings	16
B.6 Implementation of threshold calibration	16
C Additional Experimental Results	16
C.1 Detection performance of reference-based methods on our benchmark	16
C.2 The composition of evaluation sets affects evaluation results	17
C.3 Validating the reliability of our WildTrace	17
C.4 Standard deviation of repeated experiments	18
C.5 Additional results	19

A RELATED WORK

Training data detection for LLMs. Training data detection for LLMs has been studied across both pre-training and post-training stages (Shi et al., 2024; Zhang et al., 2025b). Existing detection methods generally compute a membership score based on input sequences, including techniques such as *Perplexity* (Li, 2023), *Entropy* (Song & Mittal, 2021), *Zlib* (Carlini et al., 2021), *Lower-case* (Carlini et al., 2021), *Neighbor* (Mattern et al., 2023), *MIN-K%* (Shi et al., 2024), and *MIN-K%++* (Zhang et al., 2025c). Reference-based methods further calibrate these scores using reference models to achieve better detection performance (Carlini et al., 2022; Fu et al., 2024). Recent studies have developed methods for particular post-training paradigms (Fu et al., 2024; Zhang et al., 2025a). Specifically, prior work detects fine-tuning samples by calibrating the model’s predicted probability using a reference model trained on target-generated data (Fu et al., 2024). Detection methods targeting RL training data exploit specific behavioral signals, such as entropy changes or nearest-neighbor edit distances, to infer membership in the reinforcement learning stage (Tao et al., 2026; Zhang et al., 2026). These studies generally aim to develop more effective detection methods to identify training data from pre-training or post-training stages. Instead, our work systematically examines whether existing methods remain effective when evaluated throughout a realistic, large-scale post-training setting.

Evaluation of training data detection. Existing evaluations obtain membership labels either through post-hoc benchmark construction or through controlled small-scale model training. For example, post-hoc construction treats data published before a model’s training cutoff as members and data published after the cutoff as non-members, such as WikiMIA (Shi et al., 2024; Yi & Li, 2026). MIMIR establishes a public evaluation testbed by utilizing target models trained on the publicly available Pile corpus, with members and non-members sampled from the dataset’s training and test splits, respectively (Duan et al., 2024). However, recent work shows that blind baselines can distinguish members from non-members on existing benchmarks, suggesting that detection methods may exploit distributional differences rather than the true membership signal (Das et al., 2024; Duan et al., 2024). Evaluations in controlled training settings avoid data distribution shift but often focus on a single post-training stage and use small-scale training (Fu et al., 2024; Tao et al., 2026). To address these limitations, our work introduces WildTrace, a reliable benchmark designed to evaluate detection methods in realistic, large-scale post-training settings. Specifically, our benchmark

Table 4: Training hyperparameters for the SFT, DPO, and RL stages.

Hyperparameter	ORACLE-SFT	ORACLE-DPO	ORACLE-RL
Epochs	2	1	—
Training steps	—	—	450
Learning rate	8.0×10^{-5}	1.0×10^{-6}	1.0×10^{-6}
DPO β	—	5	—
Batch size	32	32	64
Minibatches	—	—	4
Maximum sequence length	32K	16K	8K
Group size	—	—	8
Sampling temperature	—	—	1.0
Clip lower	—	—	0.2
Clip higher	—	—	0.272
Training GPUs	8	8	32

constructs members and non-members from the same source distributions and validates the reliability of evaluation sets using blind baselines. While CheckMIABench (Wang et al., 2026) obtains pre-training data membership labels from intermediate checkpoints and inferred sample ordering during training, our benchmark conducts controlled training across SFT, DPO, and RL, producing verifiable and stage-specific membership labels throughout the post-training lifecycle. We provide corresponding oracle models throughout the full post-training lifecycle ($SFT \rightarrow DPO \rightarrow RL$), enabling evaluation of training data detection in realistic, large-scale post-training scenarios.

B EXPERIMENTAL DETAILS

B.1 MODEL TRAINING CONFIGURATIONS

Building on the fully open Olmo 3 model families (Olmo et al., 2025), we conduct our main experiments using the Olmo 3 base model and train with three post-training paradigms—SFT, DPO, and GRPO. Detailed training configurations are shown in Table 4.

SFT We perform instruction tuning on OLMO-3-1025-7B base model using the Dolci-Instruct-SFT dataset. SFT training is conducted for two epochs, with a learning rate of 8.0×10^{-5} and a maximum sequence length of 32K.

DPO We train OLMO-3-7B-INSTRUCT-SFT for one epoch on the Dolci-Instruct-DPO dataset to obtain ORACLE-DPO. Training is conducted on eight GPUs with a batch size of 32 and a maximum sequence length of 16K. We use a learning rate of 1.0×10^{-6} with linear decay, a warmup ratio of 0.1, and set the DPO coefficient β to 5.

GRPO We train OLMO-3-7B-INSTRUCT-DPO on the Dolci-Instruct-RL dataset to obtain ORACLE-RL. We perform model training for 450 steps with four minibatches per update and a constant learning rate of 1.0×10^{-6} . In our experiments, we use a group size of 8, sampling temperature of 1.0, and clipping thresholds of 0.2 and 0.272 for lower and upper bounds, respectively. The maximum sequence length is 8K tokens, with prompt and response lengths capped at 2,048 and 8,192 tokens, respectively. Additionally, we track all prompts whose sampled responses achieve non-zero advantages during RL training. The members of the evaluation set are selected exclusively from these prompts.

B.2 DETAILS OF TRAINING DATASETS AND EVALUATION SETS

Table 5 presents the datasets used for training and evaluation. We apply SFT to OLMO-3-1025-7B using 2.15M samples from Dolci-Instruct-SFT, yielding ORACLE-SFT. For DPO, we train OLMO-

Table 5: Construction details of the post-training and detection datasets.

Training Stage	Base Model	Dataset Source	Train Size	Target Model	Evaluation Sets	Members	Non-members
SFT	Olmo-3-1025-7B	Dolci-Instruct-SFT	2.15M	ORACLE-SFT	Math, Code, IF, QA	4×250	4×250
DPO	Olmo-3-7B-Instruct-SFT	Dolci-Instruct-DPO	260K	ORACLE-DPO	Math, Code, IF, QA	4×250	4×250
RL	Olmo-3-7B-Instruct-DPO	Dolci-Instruct-RL	170K	ORACLE-RL	Math, Code, IF, QA	4×250	4×250

3-7B-INSTRUCT-SFT on 260K preference pairs from Dolci-Instruct-DPO to obtain ORACLE-DPO. For RL, we train OLMO-3-7B-INSTRUCT-DPO on 170K samples from Dolci-Instruct-RL, producing ORACLE-RL. For each stage, the benchmark comprises four tasks: Math, Coding, IF, and QA. Each task contains 250 training members and 250 held-out non-members, drawn from the same data distribution, yielding 2,000 evaluation examples per stage.

B.3 EVALUATION PROTOCOLS

In our main experiments, all reported results are averaged over three runs with different random seeds during inference. Detailed standard deviations across repeated runs are provided in Appendix C.4. In our experiments, we assume access to complete sample information during detection. Specifically, we evaluate training data detection using the following protocol.

Input-based methods *Input-based methods* compute membership scores based on the likelihood of input tokens. For SFT, we compute the score on each training instance using the complete question-response pair (q, y) . For DPO, where each training example consists of a preference triplet (q, y^+, y^-) , we compute the membership score on the chosen question-response pair (q, y^+) .

For RL, we compute the score on the question q , since RL does not rely on target responses.

Output-based methods *Output-based methods* compute membership scores exclusively over generated tokens. For SFT, we remove the ground-truth response from the sample and generate a response conditioned on the question. For DPO, we generate a response using the preferred dialogue, excluding its last assistant turn as the prompt. For RL, we use the sample question as the prompt to generate a response. Unless otherwise specified, we set the maximum sequence length to 2,048 tokens.

B.4 TRAINING DATA DETECTION METHODS FOR LLMs

Here, we summarize common training data detection methods for LLMs examined in this work.

Perplexity *Perplexity* (Li, 2023) is proposed to identify training data, based on the observation that members tend to have lower perplexity than non-members. Thus, the perplexity is calculated as:

$$\text{Perplexity}(x) = \exp \left\{ -\frac{1}{n} \sum_{i=1}^n \log p_{\theta}(x_i \mid x_1, \dots, x_{i-1}) \right\}, \quad (6)$$

where $x = \{x_1, x_2, \dots, x_n\}$ is a sequence of tokens and $p_{\theta}(x_i \mid x_1, \dots, x_{i-1})$ is the conditional probability of x_i given the preceding tokens.

Entropy *Entropy* (Song & Mittal, 2021) measures the model’s average predictive uncertainty across tokens. The score is defined as the mean token-level entropy:

$$\text{Entropy}(x) = -\frac{1}{n} \sum_{i=1}^n \sum_{v \in \mathcal{V}} p_{\theta}(v \mid x_{<i}) \log p_{\theta}(v \mid x_{<i}), \quad (7)$$

where $\mathcal{V}$ denotes the vocabulary.

Lowercase *Lowercase* (Carlini et al., 2021) normalizes text perplexity by the perplexity of the lowercase counterpart:

$$\text{Lowercase}(x) = -\frac{\text{PPL}_\theta(x)}{\text{PPL}_\theta(\text{lower}(x))}, \quad (8)$$

where $\text{lower}(x)$ is the lowercase version of x .

Zlib *Zlib* (Carlini et al., 2021) uses the negative ratio of log-perplexity to the zlib-compressed length:

$$\text{Zlib}(x) = -\frac{\log \text{PPL}_\theta(x)}{L_{\text{zlib}}(x)}, \quad (9)$$

where $L_{\text{zlib}}(x)$ denotes the size of the zlib-compressed representation of x .

Neighbor *Neighbor* (Mattern et al., 2023) perturbs the input sentence with masked language models to create “neighbor” and compare the loss of the input sentence with the average loss of the neighbor sentences. The score is calculated as:

$$\text{Neighbor}(x) = \frac{1}{m} \sum_{j=1}^m \mathcal{L}_\theta(\tilde{x}_j) - \mathcal{L}_\theta(x), \quad (10)$$

where $\{\tilde{x}_j\}_{j=1}^m$ are m neighboring texts and $\mathcal{L}_\theta(x)$ is the loss of the original text.

Min-K% *Min-K%* (Shi et al., 2024) computes the average log-probability of the $k\%$ tokens with the lowest predicted probabilities. Formally, MIN-K% is defined as:

$$\text{Min-K\%}(x) = \frac{1}{E} \sum_{x_i \in \text{Min-K\%}(x)} \log p_\theta(x_i | x_1, \dots, x_{i-1}), \quad (11)$$

where E denotes the number of tokens in $\text{Min-K\%}(x)$.

Min-K%++ *Min-K%++* (Zhang et al., 2025c) normalizes the log-probability of each token using the mean and standard deviation of the predictive distribution over the vocabulary. The sequence-level score is obtained by averaging the lowest $k\%$ of normalized token scores:

$$\text{Min-K\%++}(x) = \frac{1}{E} \sum_{x_i \in \text{Min-K\%++}(x)} \frac{\log p_\theta(x_i | x_{<i}) - \mu_{x_{<i}}}{\sigma_{x_{<i}}}, \quad (12)$$

where E is the number of selected tokens, and

$$\mu_{x_{<i}} = \mathbb{E}_{z \sim p_\theta(\cdot | x_{<i})} [\log p_\theta(z | x_{<i})], \quad (13)$$

$$\sigma_{x_{<i}} = \sqrt{\mathbb{E}_{z \sim p_\theta(\cdot | x_{<i})} [(\log p_\theta(z | x_{<i}) - \mu_{x_{<i}})^2]}. \quad (14)$$

Following prior work, we set $k = 20$ for both MIN-K% and MIN-K%++ in all experiments.

Min-kNN *Min-kNN* (Zhang et al., 2026) samples multiple responses for an input and computes the average of the k smallest nearest-neighbor edit distances:

$$\text{Min-kNN}(x) = \frac{1}{k} \sum_{i=1}^k d_{(i)}(x), \quad (15)$$

where $d_{(i)}(x)$ denotes the i -th smallest nearest-neighbor normalized Levenshtein distance among the generated responses.

Self-Critique *Self-Critique* (Tao et al., 2026) generates an initial response and a self-critique response for a given input, and then compares their token-level entropy sequences to measure RL-induced policy collapse. The detection score is computed using a length-aware cosine similarity:

$$\text{Self-Critique}(x) = \cos(\tilde{E}_1, \tilde{E}_2) \cdot \frac{\min(|E_1|, |E_2|)}{\max(|E_1|, |E_2|)}, \quad (16)$$

where E_1 and E_2 denote the token-level entropy sequences of the two generated responses, and $\tilde{E}_1$ and $\tilde{E}_2$ are their zero-padded counterparts aligned to equal length.

Output-based Methods Given a prompt x , we sample a response y from the target model. *Generated Perplexity* computes the perplexity of the generated tokens, and *Generated Entropy* measures the average predictive entropy of generation tokens. *Generated Min-K%* computes the average log-likelihood of K% generated tokens with the lowest predicted probability.

Reference-based Methods *Reference-based methods* (Carlini et al., 2021) calibrate the membership score of a target model using the scores from a reference model:

$$S_{\text{ref}}(x) = S(x; \mathcal{M}) - S(x; \mathcal{M}_{\text{ref}}), \quad (17)$$

where $S(x; \mathcal{M})$ and $S(x; \mathcal{M}_{\text{ref}})$ denote the metric scores of sample x computed by the target model $\mathcal{M}$ and the reference model $\mathcal{M}_{\text{ref}}$, respectively.

B.5 IMPLEMENTATION OF SMALL-SCALE TRAINING SETTINGS

Our experiments in Section 4.2 show that existing detection methods exhibit limited performance under our large-scale post-training setting, making it difficult to clearly observe the effects of different evaluation settings. We therefore adopt a small-scale training setting for the experimental analyses. Specifically, we fine-tune the Olmo 3 base model on 10K examples sampled from Dolci-Instruct-SFT for 5 epochs, yielding ORACLE-SFT-10K. By using a smaller training set and more training epochs, we amplify the membership signal and better examine the effects of different evaluation settings on detection performance.

B.6 IMPLEMENTATION OF THRESHOLD CALIBRATION

For threshold calibration on a validation set, we randomly sample 50 members and 50 non-members from the evaluation sets, then select the threshold that achieves the highest accuracy on the validation set. For ID calibration, the validation set is sampled from the Coding task. For OOD calibration, our validation set is drawn from IF, QA, or Math tasks. Each experiment is repeated 10 times with different random seeds, which produce different validation sets. For threshold calibration methods without a validation set, we evaluate Gaussian Mixture Model (GMM), $\text{GMM}_{\log 1p}$, and BlindMI (Hui et al., 2021). GMM fits a two-component Gaussian mixture to the detection scores, while $\text{GMM}_{\log 1p}$ applies a $\log(1 + x)$ transformation before fitting the same model. Without requiring a validation set, BlindMI initializes a potential non-member set from the test samples and iteratively refines it through differential comparison.

C ADDITIONAL EXPERIMENTAL RESULTS

C.1 DETECTION PERFORMANCE OF REFERENCE-BASED METHODS ON OUR BENCHMARK

To investigate whether reference-based methods achieve better detection performance, we evaluate them using an oracle reference model *Olmo 3 base*. For each method, we compare AUC scores with and without a reference model. As shown in Table 6, reference-based detection yields only marginal improvements, with AUC gains of at most 0.02. These findings indicate that reference models offer only marginal benefits, with reference-based methods showing limited effectiveness under large-scale post-training settings.

Table 6: AUC of detection methods with and without a reference model. Δ denotes the change in average AUC after introducing the reference model. **Bold** indicates the superior result for each paired comparison.

Method	Setting	IF	QA	Math	Coding	Average	Δ
Perplexity	w/o Ref.	0.54	0.56	0.50	0.57	0.54	–
	w/ Ref.	0.52	0.55	0.52	0.61	0.55	+0.01
Entropy	w/o Ref.	0.48	0.47	0.48	0.50	0.48	–
	w/ Ref.	0.46	0.45	0.51	0.50	0.48	0.00
Lowercase	w/o Ref.	0.54	0.58	0.52	0.59	0.56	–
	w/ Ref.	0.57	0.60	0.53	0.60	0.57	+0.02
Zlib	w/o Ref.	0.57	0.57	0.47	0.54	0.54	–
	w/ Ref.	0.53	0.56	0.52	0.60	0.55	+0.01
MIN-K%	w/o Ref.	0.54	0.55	0.50	0.57	0.54	–
	w/ Ref.	0.54	0.56	0.52	0.61	0.56	+0.02
MIN-K%++	w/o Ref.	0.58	0.62	0.51	0.61	0.58	–
	w/ Ref.	0.57	0.63	0.52	0.65	0.59	+0.01

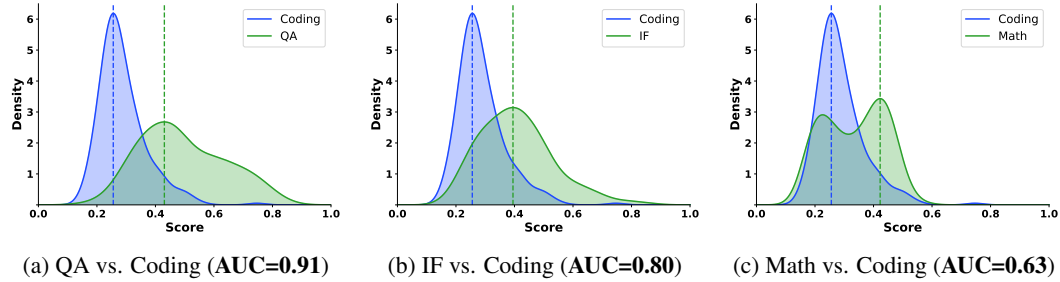


Figure 4: Score distributions of Coding compared with QA, IF, and Math. Even when both groups consist entirely of non-members, domain distribution shifts can yield illusory detection performance.

C.2 THE COMPOSITION OF EVALUATION SETS AFFECTS EVALUATION RESULTS

Figure 4 further illustrates the effect of domain distribution shifts on detection performance. Although both groups consist entirely of non-members, their membership score distributions differ substantially across domains. We find that the Coding task exhibits considerable distributional separation from other domains, even when both groups consist entirely of non-members. This separation is most pronounced with QA at an AUC of 0.91, moderate with IF at 0.80, and weakest but still notable with Math at 0.63. These results show that distribution shifts can lead to inflated AUC, even when both groups consist entirely of non-members.

C.3 VALIDATING THE RELIABILITY OF OUR WILDTRACE

Validating our WildTrace with blind baselines. In Section 3.3, we validate the reliability of WildTrace using blind baselines. The results show that blind baselines obtain negligible signals for distinguishing members from non-members on our benchmark. We provide additional implementation details and results here. As described in Section 3.3, we construct nine blind baselines by combining three text feature extractors, Bag-of-Words, TF-IDF, and Word2Vec, with three classifiers, Logistic Regression, Random Forest, and Neural Network. In our experiments, we conduct 10 runs with different random seeds for each attack baseline. Specifically, for each task, we randomly split the 250 members and 250 non-members into 400 training samples (200 members and 200 non-members) and 100 test samples (50 members and 50 non-members). In Table 7, we report the mean and standard deviation of the AUC over the 10 runs.

Validating our WildTrace with domain-specific models. To assess potential distributional shifts between the member and non-member subsets of our benchmark, we fine-tune the Olmo 3 base

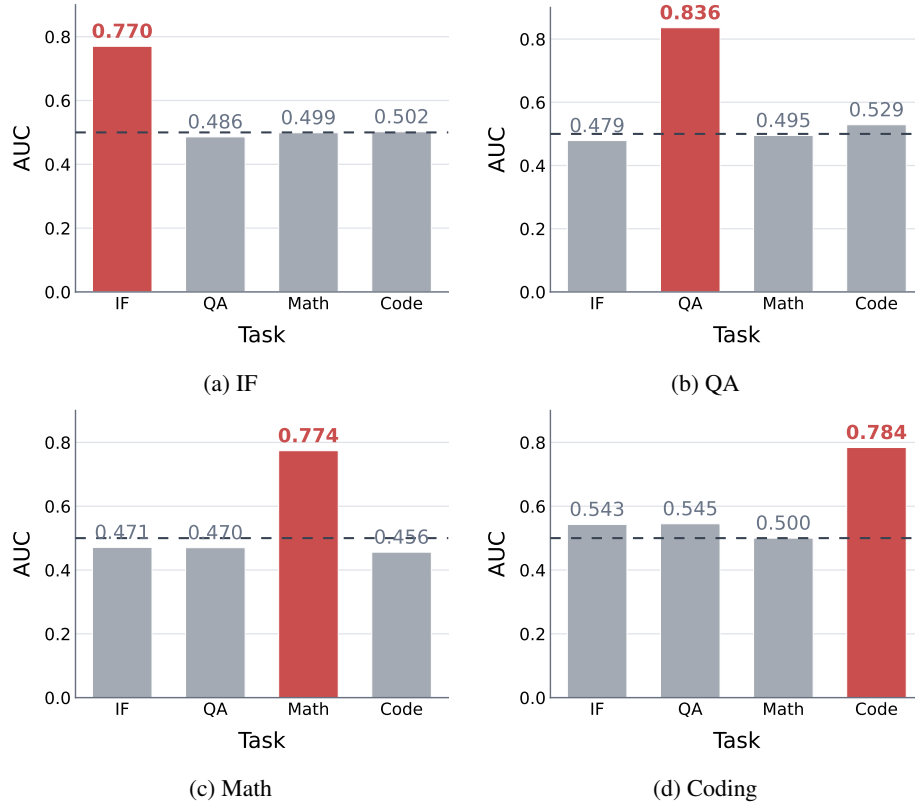


Figure 5: AUC of detection across fine-tuned models on single-domain dataset. Models are fine-tuned separately on IF, QA, Math, and Coding tasks. Detection achieves a high AUC when evaluated on the trained task (red) but performs near a random guess when evaluated on untrained tasks (gray), confirming the reliability of our benchmark.

model separately on data from each of the four domains: IF, QA, Math, and Coding, yielding four domain-specific models. We then evaluate each model on our evaluation sets. Figure 5 shows the AUC for detection across domain-specific models fine-tuned on a single-domain dataset. The red bar represents evaluation on the training domain, where member samples are included in the training set. The gray bars represent evaluation on unseen domains, for which evaluation sets are not included in the training. Our results show that detection achieves high AUC when evaluated on the domain used for training, ranging from 0.770 to 0.836. In contrast, when evaluated on domains excluded from training, AUC ranges from 0.456 to 0.545, with an average of 0.498, close to random guessing. These results show that a clear membership signal emerges only when evaluating on domains whose members were seen during training. The near-chance AUC on the unseen domains suggests that differences between members and non-members are unlikely to yield effective detection performance. These results suggest that the high detection performance on our benchmark comes from actual membership signals rather than exploitable distributional differences, thereby validating the reliability of our benchmark construction.

C.4 STANDARD DEVIATION OF REPEATED EXPERIMENTS

To verify that reported metrics in our experiments are not driven by stochastic decoding, we repeat each experiment using three different inference random seeds. We report the standard deviation of AUC in Table 8. Most input-based methods are deterministic and yield zero variance, whereas Neighbor exhibits minor fluctuations due to its stochastic neighbor sampling procedure. Output-based methods generally exhibit negligible standard deviations across runs, with Self-Critique displaying mild sensitivity under specific settings. Overall, these results confirm that our findings are robust to inference randomness and are not an artifact of stochastic decoding.

Table 7: The AUC of blind baselines on our WildTrace benchmark. LR, RF, and NN denote Logistic Regression, Random Forest, and Neural Network, respectively. Results are reported as mean $\pm$ standard deviation over 10 random splits using 80% of the data for training and 20% for testing. The best result for each task within each evaluation set is shown in **bold**.

Task	Feature	ORACLE-SFT			ORACLE-DPO			ORACLE-RL		
		LR	RF	NN	LR	RF	NN	LR	RF	NN
Math	BoW	0.48 $\pm$ 0.05	0.49 $\pm$ 0.04	0.48 $\pm$ 0.04	0.48 $\pm$ 0.06	0.46 $\pm$ 0.06	0.50 $\pm$ 0.07	0.55 $\pm$ 0.05	0.55 $\pm$ 0.02	0.54 $\pm$ 0.04
	TF-IDF	0.48 $\pm$ 0.05	0.47 $\pm$ 0.03	0.47 $\pm$ 0.05	0.50 $\pm$ 0.03	0.45 $\pm$ 0.05	0.51 $\pm$ 0.03	0.55 $\pm$ 0.04	0.54 $\pm$ 0.06	0.54 $\pm$ 0.03
	Word2Vec	0.53 $\pm$ 0.04	0.46 $\pm$ 0.05	0.45 $\pm$ 0.04	0.50 $\pm$ 0.03	0.49 $\pm$ 0.04	0.53 $\pm$ 0.04	0.55 $\pm$ 0.05	0.53 $\pm$ 0.04	0.55 $\pm$ 0.04
Code	BoW	0.49 $\pm$ 0.05	0.51 $\pm$ 0.04	0.50 $\pm$ 0.04	0.51 $\pm$ 0.02	0.51 $\pm$ 0.03	0.49 $\pm$ 0.01	0.50 $\pm$ 0.03	0.52 $\pm$ 0.05	0.48 $\pm$ 0.03
	TF-IDF	0.51 $\pm$ 0.06	0.52 $\pm$ 0.04	0.51 $\pm$ 0.05	0.51 $\pm$ 0.01	0.50 $\pm$ 0.03	0.50 $\pm$ 0.02	0.49 $\pm$ 0.04	0.49 $\pm$ 0.04	0.47 $\pm$ 0.04
	Word2Vec	0.48 $\pm$ 0.03	0.51 $\pm$ 0.06	0.46 $\pm$ 0.04	0.51 $\pm$ 0.06	0.51 $\pm$ 0.04	0.50 $\pm$ 0.03	0.50 $\pm$ 0.04	0.48 $\pm$ 0.05	0.48 $\pm$ 0.05
IF	BoW	0.54 $\pm$ 0.05	0.52 $\pm$ 0.06	0.52 $\pm$ 0.05	0.54 $\pm$ 0.05	0.49 $\pm$ 0.05	0.54 $\pm$ 0.03	0.47 $\pm$ 0.05	0.50 $\pm$ 0.04	0.45 $\pm$ 0.03
	TF-IDF	0.51 $\pm$ 0.06	0.54 $\pm$ 0.04	0.49 $\pm$ 0.06	0.53 $\pm$ 0.04	0.51 $\pm$ 0.03	0.54 $\pm$ 0.03	0.48 $\pm$ 0.03	0.51 $\pm$ 0.03	0.46 $\pm$ 0.04
	Word2Vec	0.52 $\pm$ 0.03	0.56 $\pm$ 0.04	0.56 $\pm$ 0.02	0.54 $\pm$ 0.04	0.56 $\pm$ 0.04	0.56 $\pm$ 0.04	0.52 $\pm$ 0.04	0.46 $\pm$ 0.03	0.49 $\pm$ 0.04
QA	BoW	0.47 $\pm$ 0.03	0.47 $\pm$ 0.04	0.47 $\pm$ 0.04	0.51 $\pm$ 0.04	0.51 $\pm$ 0.06	0.50 $\pm$ 0.03	0.46 $\pm$ 0.04	0.45 $\pm$ 0.03	0.48 $\pm$ 0.05
	TF-IDF	0.50 $\pm$ 0.03	0.46 $\pm$ 0.05	0.49 $\pm$ 0.03	0.51 $\pm$ 0.04	0.48 $\pm$ 0.05	0.51 $\pm$ 0.05	0.49 $\pm$ 0.05	0.47 $\pm$ 0.05	0.50 $\pm$ 0.05
	Word2Vec	0.52 $\pm$ 0.03	0.50 $\pm$ 0.04	0.48 $\pm$ 0.03	0.48 $\pm$ 0.04	0.47 $\pm$ 0.05	0.48 $\pm$ 0.05	0.50 $\pm$ 0.03	0.47 $\pm$ 0.05	0.47 $\pm$ 0.03

Table 8: Standard deviation of AUC over three runs with different inference random seeds. *Input-based methods* indicate methods that compute a score using input tokens, whereas *Output-based methods* score output tokens.

Method	ORACLE-SFT				ORACLE-DPO				ORACLE-RL				Avg.
	Math	Code	IF	QA	Math	Code	IF	QA	Math	Code	IF	QA	
Input-based methods													
Perplexity	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Entropy	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Lowercase	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Zlib	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Neighbor	0.007	0.006	0.003	0.000	0.012	0.009	0.008	0.003	0.004	0.005	0.004	0.014	0.006
MIN-K%	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
MIN-K%++	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Output-based methods													
Generated Perplexity	0.004	0.001	0.001	0.002	0.004	0.003	0.001	0.001	0.003	0.002	0.003	0.001	0.002
Generated Entropy	0.003	0.001	0.002	0.002	0.004	0.004	0.001	0.000	0.003	0.001	0.001	0.002	0.002
Generated MIN-K%	0.004	0.001	0.001	0.004	0.004	0.003	0.002	0.001	0.003	0.002	0.004	0.001	0.003
Min-kNN	0.001	0.002	0.000	0.001	0.001	0.001	0.002	0.000	0.003	0.002	0.001	0.000	0.001
Self-Critique	0.002	0.004	0.006	0.014	0.020	0.094	0.046	0.046	0.013	0.013	0.008	0.003	0.022

C.5 ADDITIONAL RESULTS

Table 9 shows the TPR@5%FPR of various detection methods on our benchmark. The experimental results show that existing methods are limited in effectiveness in large-scale post-training settings. Table 10 shows the AUC of various detection methods across evaluation sets constructed with code members and non-members from Coding, IF, QA, or Math. The results show that the detection method achieves inflated AUC scores on the Coding task when non-members are drawn from cross-domain sources.

Table 9: TPR@5%FPR of detection methods on our benchmark. *Input-based methods* indicate methods that compute a score using input tokens, whereas *Output-based methods* score output tokens. Bold shows the superior result.

Method	ORACLE-SFT				ORACLE-DPO				ORACLE-RL				Avg.
	Math	Code	IF	QA	Math	Code	IF	QA	Math	Code	IF	QA	
Input-based methods													
Perplexity	0.032	0.052	0.084	0.128	0.056	0.052	0.032	0.068	0.032	0.028	0.036	0.048	0.054
Entropy	0.048	0.032	0.068	0.032	0.024	0.064	0.048	0.020	0.028	0.036	0.080	0.076	0.046
Lowercase	0.060	0.104	0.064	0.104	0.052	0.040	0.048	0.060	0.016	0.056	0.040	0.056	0.058
Zlib	0.040	0.052	0.112	0.132	0.024	0.092	0.060	0.060	0.040	0.040	0.056	0.048	0.063
Neighbor	0.068	0.073	0.076	0.081	0.063	0.107	0.031	0.107	0.056	0.071	0.075	0.031	0.070
MIN-K%	0.040	0.064	0.080	0.128	0.056	0.052	0.016	0.064	0.040	0.028	0.036	0.044	0.054
MIN-K%++	0.064	0.120	0.060	0.252	0.076	0.080	0.028	0.056	0.072	0.072	0.048	0.044	0.081
Output-based methods													
Generated Perplexity	0.092	0.044	0.039	0.059	0.064	0.055	0.061	0.057	0.053	0.057	0.027	0.043	0.054
Generated Entropy	0.080	0.060	0.045	0.064	0.068	0.047	0.060	0.060	0.041	0.060	0.027	0.049	0.055
Generated MIN-K%	0.096	0.052	0.036	0.059	0.063	0.055	0.052	0.060	0.051	0.056	0.027	0.039	0.054
Min-kNN	0.063	0.000	0.000	0.000	0.072	0.000	0.000	0.000	0.060	0.000	0.000	0.000	0.016
Self-Critique	0.016	0.057	0.115	0.080	0.008	0.002	0.011	0.001	0.040	0.063	0.053	0.041	0.041

Table 10: AUC of various detection methods across evaluation sets constructed with Code members and non-members from Code, IF, QA, or Math. Parentheses show relative AUC changes from the Code non-member baseline, with increases marked in red.

Method	Non-member Domain			
	Code	IF	QA	Math
Perplexity	0.81	0.94 (+16.5%)	0.98 (+21.2%)	0.82 (+2.4%)
Entropy	0.84	0.89 (+6.6%)	0.91 (+8.3%)	0.79 (−5.6%)
Lowercase	0.72	0.81 (+12.8%)	0.81 (+13.5%)	0.79 (+9.6%)
MIN-K %	0.77	0.91 (+18.4%)	0.95 (+24.0%)	0.79 (+3.1%)
MIN-K %++	0.84	0.84 (+0.5%)	0.77 (−8.7%)	0.84 (0.0%)
Avg.	0.79	0.88 (+10.7%)	0.88 (+11.3%)	0.81 (+1.6%)