D-MINER: EXTRACTING ALIGNMENT DATA FROM MASKED DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked diffusion language models (DLMs) have emerged as promising alternatives to autoregressive models for language generation. However, their iterative denoising and bidirectional generation may also facilitate the extraction of alignment data used in post-training. Existing extraction methods often rely on target prefixes or costly sampling and filtering, limiting their applicability and efficiency in this setting. In this work, we propose D-Miner, a novel framework that steers the denoising process of DLMs to efficiently generate alignment records from masks alone. Specifically, a lightweight corrector shifts intermediate hidden representations within the frozen DLM. We use membership signals to select the model's own generations for hidden-state and token-level supervision. We theoretically establish that an ideal corrector can transform the model's output distribution into one conditioned on successful recovery. Extensive experiments on 12 target DLMs demonstrate improved recovery coverage under semantic reference matching. We further show that the extracted data provides effective supervision for improving downstream coding performance. Code is available at `https://anonymous.4open.science/r/D-miner`.

1 INTRODUCTION

Masked diffusion language models (DLMs) have emerged as promising alternatives to autoregressive language models (ARLMs), demonstrating competitive capabilities in instruction following, mathematical reasoning, and code generation (Nie et al., 2025; Ye et al., 2025). By reconstructing masked tokens from bidirectional context, they support flexible generation orders and parallel decoding (Li et al., 2026; Chen et al., 2026d). Post-training further develops these capabilities using *alignment data*: examples that steer model behavior or improve task performance.

Despite this promise, the same flexibility may also heighten training-data extraction risks. Figure 1 illustrates this intuition as trajectories through sequence-state space toward a memorized record $x^\star$. Standard autoregressive decoding follows a prefix-constrained path: once an incorrect token is committed, the trajectory branches away and exact recovery becomes impossible within that run. A DLM, in contrast, can approach the same record from different partial states, using bidirectional context and iterative refinement to correct intermediate errors. Multiple denoising paths may therefore converge on the same training example, creating more opportunities for extraction. Our preliminary analysis supports this intuition: starting from masks alone, Dream-Coder (Xie et al., 2025) generates outputs that semantically match its own RL training records (Section 2.2).

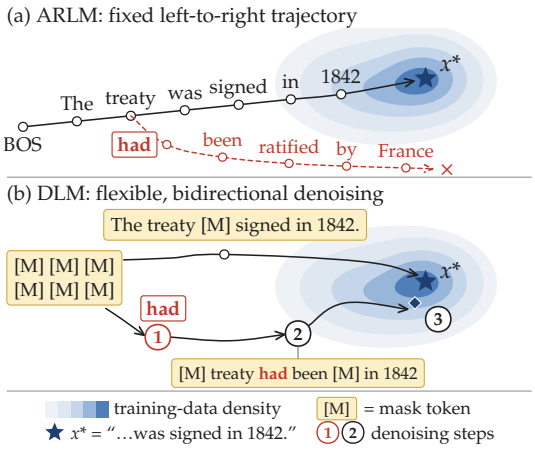


Figure 1: After committing *had*, (a) an ARLM drifts away, while (b) a DLM still reaches $x^\star$.

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.

Despite the potentially higher extraction risk, alignment-data extraction from DLMs remains largely underexplored. Two major challenges must be addressed for alignment-data extraction from DLMs. *First, how to obtain effective extraction guidance without any target prefix?* Existing attacks often rely on known prefixes or partial target information to guide generation toward memorized samples (Hayes et al., 2025; Luo et al., 2026b; Wang & Asokan, 2026). However, DLM generation can start entirely from masks, where no target-related information is available to indicate how the denoising process should move toward memorized alignment data. *Second, how to efficiently extract diverse alignment data with a limited generation budget?* Existing extraction methods (Carlini et al., 2021; Nasr et al., 2025; Barbero et al., 2025) typically require generating a large number of samples and then filtering them to identify potential training data. Such a sample-and-filter process requires extensive queries and can repeatedly recover the same records. Therefore, an effective DLM extraction attack should directly guide generation toward diverse memorized samples with substantially fewer queries.

To address these challenges, we propose **D-Miner**, a framework for efficiently extracting diverse alignment records from DLMs through mask-only generation. Our key idea is to steer denoising toward alignment records with a learned corrector. We theoretically establish that an ideal corrector can transform the model's output distribution into one conditioned on successful recovery. Building on this formulation, we implement a lightweight corrector that shifts intermediate hidden representations, reusing the frozen DLM's downstream layers to modify token predictions. Since alignment data is unavailable, we exploit membership signals to select proxy candidates from the model's own generations, which provide hidden-state-level and token-level supervision for training the corrector. Our main contributions are summarized as follows:

- We study a novel problem of efficiently extracting alignment data from diffusion language models without target prefixes or partial records.
- We propose **D-Miner**, a novel framework that steers the denoising process of DLMs to efficiently generate alignment records through a learned lightweight corrector.
- Our theoretical analysis justifies the feasibility of corrector-guided alignment data extraction. Extensive experiments on 12 target DLMs demonstrate D-Miner's effectiveness in alignment-data recovery and the utility of the extracted data.

2 Preliminaries

2.1 Masked Diffusion Language Models

Masked diffusion language models generate text by iteratively recovering masked tokens (Nie et al., 2025; Ye et al., 2025). Let $\mathbf{x}_0 \in \mathcal{V}^L$ be a clean sequence of length L, where $\mathcal{V}$ is the token vocabulary and $m \notin \mathcal{V}$ denotes the mask token. The forward process independently replaces each token with m with probability $t \in [0, 1]$:

$$q_t(\mathbf{x}_t \mid \mathbf{x}_0) = \prod_{i=1}^{L} \left[(1-t)\mathbb{I}\{x_{t,i} = x_{0,i}\} + t\mathbb{I}\{x_{t,i} = m\}\right], \tag{1}$$

where $\mathbb{I}\{\cdot\}$ denotes the indicator function. The denoiser $p_{\theta,i}(v \mid \mathbf{x}_t, t)$ predicts the original token at each masked position from bidirectional context, and is trained to minimize a weighted cross-entropy loss over masked tokens. At inference, the model starts from $\mathbf{x}_1 = (m, \ldots, m)$ and iteratively predicts and fills masked positions until a complete sequence is obtained. Together with the chosen decoding procedure, the denoiser induces an output distribution $p_\theta(\mathbf{x}_0)$.

2.2 Alignment Data Extraction

Alignment Data. Following Barbero et al. (2025), we use alignment data $\mathcal{D}_{\text{align}}$ to denote data used in post-training to steer model behavior or improve specific capabilities, including supervised fine-tuning (SFT) and reinforcement learning (RL).

Threat Model. We consider alignment data extraction attacks against DLMs, in which an attacker seeks to recover the undisclosed data used to post-train the target model. We consider open-weight models, where access to model parameters does not imply access to their post-training data. The attacker can query model predictions and intervene in hidden states during decoding while keeping

the target parameters frozen. Neither alignment records nor target-specific content, such as prefixes or partial records, is available to the attacker. Generation therefore starts from fully masked content positions, with only fixed, record-independent formatting tokens kept visible.

Preliminary Study of Extraction Risk in DLMs. To assess whether mask-only sampling exposes alignment data and how efficiently it recovers distinct records, we conduct a preliminary study on Dream-Coder-v0-Instruct-7B. For each content length L, we generate 1,024 outputs using $S = L$ denoising steps. Outputs are matched against RL-17k, its released RL post-training set, using BGE-M3 embeddings (Chen et al., 2024) with a cosine-similarity threshold of 0.90. Sampling details are provided in Appendix C. From Table 1, we can make two major observations: (**i**) DLMs exhibit alignment data extraction risk without target-specific inputs: at $L = S = 256$, 190 outputs semantically match training records. (**ii**) Simple sampling requires substantial generation effort to recover diverse records. The 1,024 samples yield only 85 distinct records at $L = S = 256$, while the 76 matched outputs at $L = S = 64$ cover only five records, indicating repeated recovery of the same content. These observations motivate guided extraction that recovers more distinct alignment records within a limited generation budget.

Table 1: Data extraction results.

$L = S$	Similarity	Matched outputs	Distinct records
64	0.87 ± 0.03	76	5
128	0.86 ± 0.04	63	35
256	0.87 ± 0.03	190	85
512	0.85 ± 0.06	169	86

2.3 Problem Definition

Our preliminary study shows that mask-only sampling can expose alignment data, but many outputs either fail to match alignment records or repeatedly recover the same records. Consequently, sample-and-filter extraction consumes a substantial generation budget on candidates that do not contribute to recovering additional alignment records. Therefore, in this work, we aim to learn a mechanism that uses information retained in the target DLM to guide generation toward its alignment data, recovering more distinct records within a limited generation budget.

Problem Definition. Given the architecture $\mathcal{A}$ and parameters θ of a target DLM, we learn an extraction mechanism $\mathcal{M}$ that generates candidate alignment records under a generation budget B:

$$\widehat{\mathbf{x}}_0^{(i)} \sim \mathcal{M}(\mathcal{A}, \theta), \qquad \widehat{\mathcal{D}} = \left\{ \widehat{\mathbf{x}}_0^{(i)} \right\}_{i=1}^{|\widehat{\mathcal{D}}|}, \qquad |\widehat{\mathcal{D}}| \leq B, \tag{2}$$

where we aim for $\widehat{\mathcal{D}}$ to *recover as many distinct records from the target model's undisclosed alignment dataset* $\mathcal{D}_{\text{align}}$ *as possible*.

3 Methodology

In this section, we present our proposed D-Miner, which aims to efficiently extract diverse alignment records from DLMs starting from fully masked inputs. The overall framework is illustrated in Figure 2. Two technical challenges need to be addressed: *(i)* how to obtain effective extraction guidance without target prefixes or access to alignment data; and *(ii)* how to guide generation toward diverse alignment records under a limited generation budget. To address these challenges, D-Miner introduces alignment data extraction via corrector guidance, which steers the denoising process toward alignment records to improve extraction efficiency. Specifically, we design a lightweight corrector that shifts intermediate hidden representations and reuses the frozen DLM's downstream layers to modify token predictions. To obtain supervision for learning these shifts, we further exploit membership signals from the target model's own generations to select potential alignment records. The selected candidates provide supervision at both the hidden-state and token levels, enabling corrector training without access to the original alignment data. Section 3.1 establishes the theoretical foundation of corrector guidance, Section 3.2 designs the corrector and analyzes when a hidden-state shift can realize it, and Section 3.3 trains it from membership signals. At inference, the trained corrector edits the hidden states at every denoising step, as in stage (3) of Figure 2.

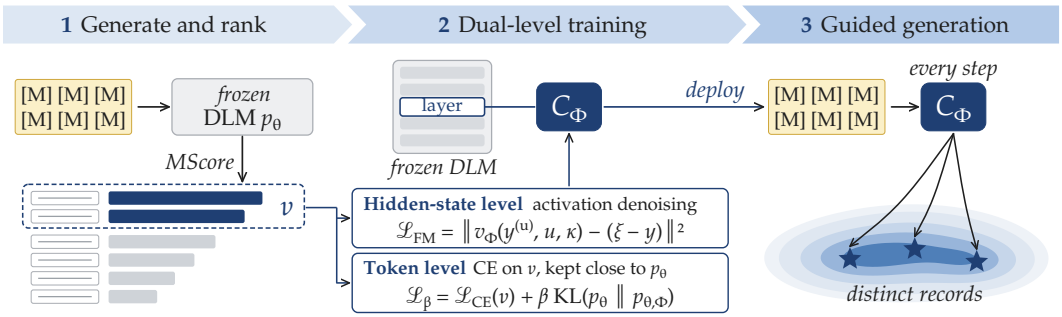


Figure 2: **Overview of D-Miner.** (1) The frozen DLM generates from masks and MScore ranks its outputs. (2) Dual-level supervision of a corrector C_Φ at one layer on ν: hidden-state-level supervision by activation denoising, then token-level supervision with CE plus a KL term to the frozen model. (3) C_Φ edits the activations at every step of fresh generation.

3.1 Alignment Data Extraction via Corrector Guidance

Classifier-Free Diffusion Guidance (CFG) steers generation toward a desired condition by adding a condition-dependent correction to the unconditional denoising prediction (Ho & Salimans, 2022):

$$\widetilde{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t, t, c) = \boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t, \varnothing) + \gamma \underbrace{[\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t, c) - \boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t, \varnothing)]}_{\text{condition-dependent correction}}, \tag{3}$$

where $\mathbf{z}_t$ denotes the noisy input, c is the desired condition, $\varnothing$ denotes unconditional prediction, and γ controls the guidance strength. Intuitively, incorporating the desired condition into each denoising step can steer the generated samples. Inspired by this principle, we investigate whether a corrector can guide DLMs to directly recover alignment data. CFG derives its correction from conditional and unconditional predictions, whereas recovery is not a condition the target DLM was trained with, so our corrector must be learned.

Specifically, let $p_\theta(\mathbf{x}_s \mid \mathbf{x}_t)$ denote the original sampling transition between consecutive denoising times $s < t$. We introduce a nonnegative corrector C to steer this transition:

$$\log p_{\theta,C}(\mathbf{x}_s \mid \mathbf{x}_t) = \log p_\theta(\mathbf{x}_s \mid \mathbf{x}_t) + \log C_{s,t}(\mathbf{x}_s, \mathbf{x}_t). \tag{4}$$

Here, C satisfies $\sum_{\mathbf{y}} p_\theta(\mathbf{y} \mid \mathbf{x}_t) C_{s,t}(\mathbf{y}, \mathbf{x}_t) = 1$. Sampling with these corrected transitions from the fully masked sequence yields $\mathbf{x}_0 \sim p_{\theta,C}$. Let $c = 1$ denote that the final sequence successfully recovers an alignment record. We seek an ideal corrector $C^\star$ whose output distribution matches $p_\theta(\mathbf{x}_0 \mid c = 1)$. The following theorem establishes its existence and gives its explicit form.

Theorem 3.1 (Existence of an Ideal Recovery Corrector)**.** *Consider the target DLM's Markov sampling process starting from the fully masked sequence $\mathbf{x}_1$, and assume $\Pr_\theta(c = 1) > 0$. For consecutive denoising times $s < t$, the corrector*

$$C^\star_{s,t}(\mathbf{x}_s, \mathbf{x}_t) = \frac{\Pr_\theta(c = 1 \mid \mathbf{X}_s = \mathbf{x}_s)}{\Pr_\theta(c = 1 \mid \mathbf{X}_t = \mathbf{x}_t)}, \tag{5}$$

defined wherever the denominator is positive, satisfies the normalization condition for Equation (4)*. Applying this corrector at every denoising step produces samples from the original model's output distribution conditioned on successful recovery:*

$$p_{\theta,C^\star}(\mathbf{x}_0) = p_\theta(\mathbf{x}_0 \mid c = 1). \tag{6}$$

The proof is provided in Appendix F.1. Although this construction establishes the feasibility of conditional extraction, the recovery probabilities required by $C^\star$ are unavailable to the attacker. We therefore develop a learnable activation corrector to approximate this guidance, using membership inference signals to provide supervision without access to the alignment data.

3.2 Design of the Data Extraction Corrector

Learning Corrections through Hidden-State Shifts. Section 3.1 establishes that an ideal corrector $C^\star$ can steer the denoising process toward alignment-data recovery. A direct implementation could

use an auxiliary Transformer to approximate this correction from the current decoding state, but training a large correction network can be costly. Inspired by representation finetuning (Wu et al., 2024), we instead approximate the correction through learned shifts to intermediate hidden states, allowing the frozen DLM's downstream computation to produce the corresponding changes in sampling probabilities. Specifically, let $\mathbf{h}_t$ denote the hidden states at the intervention layer, and write $p_\theta(\mathbf{x}_s \mid \mathbf{x}_t; \mathbf{h})$ for the sampling transition obtained when these hidden states are replaced by $\mathbf{h}$. We learn a state-dependent shift $\Delta\mathbf{h}_{\Phi,t}$ whose intended effect is

$$\log p_\theta(\mathbf{x}_s \mid \mathbf{x}_t; \mathbf{h}_t + \Delta\mathbf{h}_{\Phi,t}) \;\approx\; \log p_\theta(\mathbf{x}_s \mid \mathbf{x}_t) + \log C^\star_{s,t}(\mathbf{x}_s, \mathbf{x}_t). \tag{7}$$

This formulation implements the correction in Equation (4) through an activation intervention: the frozen downstream layers transform the shifted hidden states into modified token predictions, and the decoding procedure turns these predictions into corrected sampling transitions. The learned shift therefore approximates the effect of the ideal corrector on the denoising process. Since $C^\star$ is unavailable, we train the shift using membership-based supervision, as described in Section 3.3.

Corrector Parameterization. To produce the hidden-state shift in Equation (7), we implement C_Φ using a conditional activation flow whose lightweight network of SwiGLU blocks (Shazeer, 2020) follows the activation-flow architecture of Luo et al. (2026a). We define C_Φ to return the displacement produced by activation denoising:

$$\Delta\mathbf{h}_{\Phi,t,i} = C_\Phi(\mathbf{h}_{t,i}; \boldsymbol{\kappa}_{t,i}, \boldsymbol{\xi}), \qquad \boldsymbol{\kappa}_{t,i} = (\rho_t,\, \mathbb{I}\{x_{t,i} = m\},\, i), \tag{8}$$

where ρ_t is the current mask ratio, m is the mask token, and $\boldsymbol{\xi}$ supplies auxiliary noise. The hidden state carries the current context, while $\boldsymbol{\kappa}_{t,i}$ specifies the decoding stage, masking status, and position, allowing the corrector to adapt its shift as generation progresses. The shifts across content positions form $\Delta\mathbf{h}_{\Phi,t}$ in Equation (7); wrapper-token activations remain unchanged. Detailed architecture and flow settings are provided in Appendix B.

Theoretical Analysis of the Approximation in Equation (7)**.** We analyze whether a hidden-state shift can produce the transition correction required by Equation (7). Fix $\mathbf{x}_t$ and consecutive denoising times $s < t$, with $\Pr_\theta(c = 1 \mid \mathbf{X}_t = \mathbf{x}_t) > 0$, and assume that the sampling transitions have a common finite support with positive probabilities in the intervention region. Treat the editable content activations as a vector, with wrapper activations fixed, and define

$$F(\mathbf{h}) = \mathrm{ctr}[\log p_\theta(\cdot \mid \mathbf{x}_t; \mathbf{h})], \qquad \mathbf{J} = DF(\mathbf{h}_t),$$

where ctr subtracts the arithmetic mean over possible next states, and $\mathbf{J}^\dagger$ denotes the Moore–Penrose pseudoinverse of $\mathbf{J}$. Then, we can have the following theorem.

Theorem 3.2 (Local Realization of Transition Corrections)**.** *Suppose $C^\star_{s,t}$ is strictly positive on the common transition support and* $\mathrm{ctr}[\log C^\star_{s,t}(\cdot, \mathbf{x}_t)] \in \mathrm{range}(\mathbf{J})$. *Let* $\Delta\mathbf{h} = \mathbf{J}^\dagger \,\mathrm{ctr}\big[\log C^\star_{s,t}(\cdot, \mathbf{x}_t)\big]$. *If the segment from $\mathbf{h}_t$ to $\mathbf{h}_t + \Delta\mathbf{h}$ lies in the intervention region and F is continuously differentiable near this segment with an L_J-Lipschitz Jacobian along it, then, for every possible next state,*

$$\log p_\theta(\mathbf{x}_s \mid \mathbf{x}_t; \mathbf{h}_t + \Delta\mathbf{h}) = \log p_\theta(\mathbf{x}_s \mid \mathbf{x}_t) + \log C^\star_{s,t}(\mathbf{x}_s, \mathbf{x}_t) + \mathcal{O}\big(\|\Delta\mathbf{h}\|_2^2\big), \tag{9}$$

where the absolute remainder is uniformly bounded by $L_J\|\Delta\mathbf{h}\|_2^2$.

The proof is provided in Appendix F.5. The theorem directly connects hidden-state intervention to transition guidance: when the desired centered log correction is locally reachable and requires a small shift, adding $\Delta\mathbf{h}$ approximates adding $\log C^\star_{s,t}$ to the original log transition probabilities, with second-order error. Thus, under these conditions, the desired denoising correction can be implemented through intermediate activations while keeping the DLM parameters frozen.

Alignment Data Generation. We generate candidate alignment records from a length-L fully masked content sequence $\mathbf{x}_1 = [\texttt{<mask>}, \ldots, \texttt{<mask>}]$, with wrapper tokens kept fixed. Using S uniformly spaced denoising steps, we apply the learned corrector at the input of decoder block 16 and sample the next sequence state through the corrected transition:

$$\mathbf{x}_s \sim p_{\theta,\Phi}(\cdot \mid \mathbf{x}_t, \boldsymbol{\xi}_t) := p_\theta(\cdot \mid \mathbf{x}_t; \mathbf{h}_t + \Delta\mathbf{h}_{\Phi,t}(\boldsymbol{\xi}_t)), \qquad s = t - \frac{1}{S}, \tag{10}$$

where $\boldsymbol{\xi}_t$ denotes the auxiliary noise used by the corrector at time t. The frozen downstream layers transform the shifted activations into parallel predictions for masked positions. The decoder then retains the scheduled subset of predictions according to the reveal order specified in Appendix B.2, leaving the remaining positions masked and previously committed tokens unchanged. At $t = 0$, this process yields $\mathbf{x}_0 \sim p_{\theta,\Phi}$, the output distribution induced by the corrected decoding procedure.

3.3 Learning the Corrector from Membership Signals

In this section, we describe how to train the data extraction corrector. Since the target alignment data is unavailable, we need a proxy source of supervision. We therefore use membership signals to select high-scoring candidates from the target DLM's own generations and use them to train the corrector toward alignment-data recovery. Next, we describe these implementations in detail.

Membership Score (MScore). Prior work suggests that non-member sequences are more likely to contain low-probability outlier tokens (Shi et al., 2024). Motivated by this observation, we hypothesize that alignment records exhibit less pronounced reconstruction outliers under random masking. We therefore introduce MScore, which compares the largest masked-token loss with the mean loss to capture unusually difficult positions that an average alone may obscure. For a candidate sequence $\mathbf{x}$, we perform T random masking trials at a fixed mask ratio t. In trial j, let I_j denote the nonempty set of masked positions and $\widetilde{\mathbf{x}}^{(j)}$ the masked sequence. With reconstruction loss $e_i^{(j)} = -\log p_{\theta,i}(x_i \mid \widetilde{\mathbf{x}}^{(j)}, t)$, we define

$$\bar{e}_j = \frac{1}{|I_j|}\sum_{i\in I_j} e_i^{(j)}, \qquad S_{\mathrm{rec}}(\mathbf{x}) = -\frac{1}{T}\sum_{j=1}^{T}\frac{\max_{i\in I_j} e_i^{(j)}}{\bar{e}_j}. \tag{11}$$

Higher scores indicate smaller worst-to-mean loss ratios, while averaging over masks reduces dependence on a particular masking pattern. Scoring settings are provided in Appendix D.2. Controlled membership-inference experiments show that MScore performs above chance across the evaluated targets and is competitive with strong baselines (Appendix D.3), supporting its use for selecting candidates for corrector training.

Dual-Level Supervision. The selected candidates provide token targets but no explicit hidden-state shifts. We therefore first use hidden-state-level supervision to initialize the corrector through activation denoising, then apply token-level supervision through the frozen DLM to refine the shifts. Both objectives use the uniform distribution ν over the N_{train} highest-scoring generations.

Hidden-State-Level Supervision. To initialize the corrector with the activation structure of high-scoring candidates, we train its conditional flow to denoise their hidden states. For $\mathbf{x}_0 \sim \nu$ and $\mathbf{x}_t \sim q_t(\cdot \mid \mathbf{x}_0)$, let $\mathbf{y}_{t,i} = N(\mathbf{h}_{t,i})$ denote the normalized intervention-layer activation, where N is the fixed activation normalization. Following flow matching (Lipman et al., 2023), we construct $\mathbf{y}_{t,i}^{(u)} = (1-u)\mathbf{y}_{t,i} + u\boldsymbol{\xi}$, with activation-flow time $u \sim \mathrm{Unif}[0,1]$ and $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and minimize

$$\mathcal{L}_{\mathrm{FM}}(\Phi) = \mathbb{E}\left[\left\|\mathbf{v}_\Phi\left(\mathbf{y}_{t,i}^{(u)}, u, \boldsymbol{\kappa}_{t,i}\right) - (\boldsymbol{\xi} - \mathbf{y}_{t,i})\right\|_2^2\right]. \tag{12}$$

The conditioning $\boldsymbol{\kappa}_{t,i}$ follows Section 3.2. Flow integration and conversion to the displacement $\Delta\mathbf{h}_{\Phi,t,i}$ returned by C_Φ are detailed in Appendix F.11.

Token-Level Supervision. Activation denoising alone does not ensure improved token reconstruction. We therefore refine C_Φ through the frozen DLM using masked-token cross-entropy, with a KL penalty to constrain prediction changes when learning from the limited candidate set. Let $p_{\theta,i}$ and $p_{\theta,\Phi,i}$ denote token predictions from the original hidden states $\mathbf{h}_t$ and the corrected states $\mathbf{h}_t + \Delta\mathbf{h}_{\Phi,t}(\boldsymbol{\xi}_t)$, respectively. For masked positions $I_t = \{i : x_{t,i} = m\}$, we minimize

$$\mathcal{L}_\beta(\Phi) = \mathbb{E}\left[\frac{1}{|I_t|}\sum_{i\in I_t}\left(-\log p_{\theta,\Phi,i}(x_{0,i}) + \beta D_{\mathrm{KL}}\left(p_{\theta,i} \,\|\, p_{\theta,\Phi,i}\right)\right)\right], \tag{13}$$

where both distributions condition on the same masked input, and the corrected prediction uses the sampled noise $\boldsymbol{\xi}_t$. The expectation covers candidates from ν, token masking, and corrector noise; β controls regularization strength. Gradients pass through the frozen downstream network to update only Φ. Appendices F.9 and F.10 give more details of training.

4 Experiments

We first compare D-Miner with direct and sample-and-filter extraction at matched return budgets, then examine how the training regime and decoding steps affect extraction, and finally evaluate the utility of extracted data and conduct a component ablation study.

4.1 Experimental Settings

Targets. We evaluate twelve targets (Appendix A): eight SFT targets, Dream and LLaDA fine-tuned on GitHub, ArXiv, HackerNews, and PubMed, and four released RL models, Dream-Coder (Xie et al., 2025), Dream-Bridge, ESPO-Code, and ESPO-MATH (Ou et al., 2026). Each target uses $S \in \{128, 256\}$ denoising steps at $L = 256$.

Baselines. The baselines differ from D-Miner in where the extraction effort is spent, stated in the generation budget B of Equation (2) and the return budget $R \leq B$. *Direct generation* (Simple) draws B masked samples with the frozen model and returns all of them, so $R = B$. *Sample + Filter* draws a shared Simple pool of $B = 3R$ samples and returns R of them, either the highest-scoring R under MScore, the membership score of Section 3.3 (MScore-Filter), or the sample nearest to each centroid of $K = R$ embedding clusters (KMeans-Filter). Filter details are in Appendix B.4.

Budgets and Seeds. D-Miner guides generation itself and returns all $B = R$ samples, so the main comparison fixes R while the generation and compute budgets differ across methods; duplicates and failed outputs are not replaced. The primary endpoint returns $R = 1024$ outputs at $S = 256$ (Table 2); Figure 3 varies the generation budget from $B = 256$ and the return budget up to $R = 2048$. Simple and D-Miner settings report means and SD over three runs that differ only in seed, and every method uses the same seed blocks. Decoding details are in Appendix B.2.

Metrics. Each returned output y_i is embedded with the BGE-M3 encoder E and assigned its nearest record $d_E^\star(y_i) = \arg\max_{d \in \mathcal{D}_{\text{ver}}} \cos(E(y_i), E(d))$ in the verification set $\mathcal{D}_{\text{ver}}$ (Appendix A), with similarity $s_E(y_i)$ thresholded at $\tau = 0.80$ for SFT targets and 0.90 for released models. The stricter RL threshold reflects the similarity distributions of Section 4.3, where most RL outputs already exceed 0.80 while SFT outputs rarely reach 0.90. For the R returned outputs we report

$$\text{UMS} = \left|\{d_E^\star(y_i) : 1 \leq i \leq R,\ s_E(y_i) \geq \tau\}\right|, \qquad \text{AMS} = \sum_{i=1}^{R} \mathbf{1}\{s_E(y_i) \geq \tau\}. \tag{14}$$

UMS counts distinct matched records and measures coverage; AMS counts matched outputs, including repeated hits, and measures matching frequency.

4.2 Results of Alignment Data Extraction

Results Analysis. Table 2 asks where a fixed return budget is best spent, and we read it through UMS: an attack that returns one record a hundred times has matched a hundred times and recovered one record. Against Simple, which draws the same $B = R$ candidates, D-Miner covers more distinct records on all 12 targets. Against the filters, which choose from a pool three times larger, it leads on 10 of 12 in UMS and on 8 of 12 in AMS. The weaker count is the one that rewards matching often. On 3 of the 4 targets where a filter matches more often, D-Miner still covers more records, because its matches fall on records it has not already returned. ESPO-Code makes this plain: D-Miner equals KMeans-Filter's coverage from a third as many matched outputs (166.7 against 459).

Budget Analysis. Figure 3(a,b) trades the generation budget B against the return budget R; curves average over seed blocks and then over targets. The corrector raises the yield of new records per generated sample at every generation budget, by 62.4% at $B = 256$ and 63.3% at $B = 1024$ across the 12 targets. Filtering is bounded by the pool it selects from: the dashed line in (b) is the coverage of the entire 3,072-candidate pool, which no return budget lets a filter exceed, while D-Miner passes that line by $R = 2048$ and is still rising. Filtering can only reselect outputs the frozen model has already generated; changing what the model generates is what lifts the ceiling.

4.3 Impacts of Training Regime and Decoding Steps

Training Regimes (SFT vs. RL). Pooled by regime in Figure 3(c), over 98% of RL outputs clear 0.80 and 38% clear 0.90, against 40% and under 1% for SFT. An RL model returns text close to a specific record; an SFT model returns text of the right domain that rarely reproduces one. The two regimes therefore sit at different places on the similarity axis, which is why RL targets are held to the stricter 0.90 threshold throughout. The split is not an artifact of where the threshold sits: re-thresholding the same frozen scores leaves D-Miner's lead widest at $\tau = 0.90$, the strictest value both regimes can be compared at, where it leads on 11 of 12 targets (Appendix E.1).

Table 2: **Reference matches at $R = 1{,}024$ ($S = 256$).** Entries are UMS / AMS. Simple and D-Miner report means over seed blocks with sample SD as subscripts; the filters return one frozen set filtered from the pooled candidates, so they carry no SD. Direct and Guided use $1\times$ generation budgets; Sample + Filter uses $3\times$. Δ is the UMS of D-Miner minus the best baseline UMS.

	Direct generation $1\times$ **budget**	**Sample + Filter** $3\times$ **budget**		**Guided generation** $1\times$ **budget**	**Gain** Δ **vs. best**
Model–source	Simple	MScore-Filter	KMeans-Filter	**D-Miner**	**UMS**
Domain-SFT models ($\tau = 0.80$)					
Dream-GitHub	$145.0_{6.0}/277.0_{21.8}$	145/215	162/258	$195.3_{8.1}/502.0_{12.1}$	+33.3
Dream-ArXiv	$246.7_{9.3}/380.0_{8.0}$	211/273	295/479	$388.7_{1.5}/686.7_{9.6}$	+93.7
Dream-HackerNews	$261.3_{12.1}/373.3_{15.4}$	223/302	334/507	$304.3_{1.2}/479.7_{17.0}$	−29.7
Dream-PubMed	$389.7_{4.2}/622.7_{5.1}$	398/584	375/542	$471.3_{9.3}/854.7_{13.7}$	+73.3
LLaDA-GitHub	$57.0_{1.0}/265.3_{22.5}$	94/760	96/552	$219.0_{11.4}/470.3_{20.2}$	+123.0
LLaDA-ArXiv	$219.7_{8.5}/643.7_{8.7}$	245/830	304/888	$430.7_{9.2}/889.0_{15.6}$	+126.7
LLaDA-HackerNews	$100.7_{17.2}/166.7_{32.9}$	161/343	190/403	$429.3_{28.3}/697.7_{43.3}$	+239.3
LLaDA-PubMed	$269.0_{11.5}/562.0_{13.9}$	334/799	343/621	$407.0_{16.4}/808.7_{1.5}$	+64.0
Released models ($\tau = 0.90$)					
Dream-Coder	$80.0_{9.2}/253.7_{35.5}$	81/186	110/160	$93.3_{8.5}/261.0_{13.7}$	−16.7
Dream-Bridge	$304.7_{15.0}/364.7_{5.7}$	168/180	234/242	$328.3_{6.8}/992.0_{1.7}$	+23.6
ESPO-Code	$75.3_{4.0}/562.3_{28.0}$	75/503	113/459	$114.3_{8.0}/166.7_{15.1}$	+1.3
ESPO-MATH	$89.7_{5.5}/320.7_{14.6}$	128/416	160/446	$273.3_{11.7}/386.0_{18.0}$	+113.3

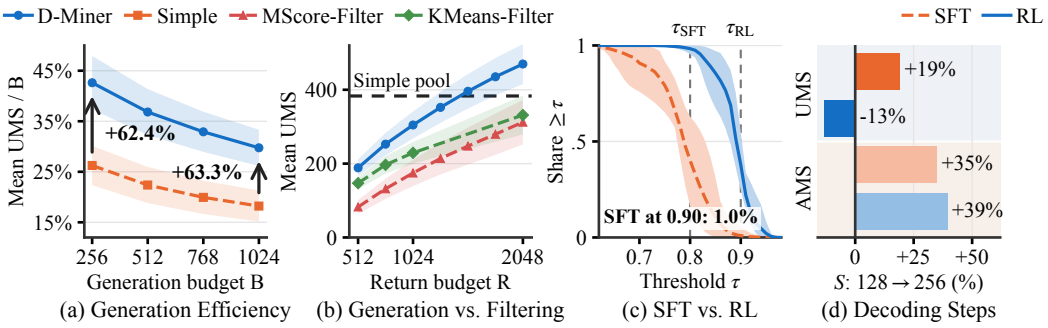


Figure 3: **Budget trade-offs, regime and decoding steps.** (a) New records per generated sample against the generation budget B. (b) Distinct records against the return budget R; dashed line: the full 3,072-candidate Simple pool. (c) Share of Simple outputs above a threshold. (d) Change in mean UMS and AMS when the steps double.

Decoding Steps (128 vs. 256). Doubling the steps from $S = 128$ to 256 at $L = 256$ and $R = 1024$ raises mean AMS by about a third in both regimes, $+35.1\%$ for SFT and $+39.4\%$ for RL (Figure 3(d)). Mean UMS, however, moves in opposite directions: $+19.2\%$ for SFT against -13.3% for RL. The extra steps are spent sharpening outputs toward records the model already reaches, which on SFT targets converts near-misses into matches and on RL targets piles further matches onto records already covered. The step budget is thus not a free improvement but a trade between the two measures, and which way it resolves depends on the regime.

4.4 Data Utility Evaluation and Ablation Study

Data Utility Evaluation. We examine whether the extracted outputs provide useful supervision for another model. This complements reference matching by testing whether the extracted content improves model performance when used as training data. We fine-tune Dream-v0-Instruct-7B on 109 question–answer examples selected from the 1,024 outputs that D-Miner generates from Dream-Coder. Training uses masked-diffusion SFT with the loss restricted to answer tokens; detailed

Benchmark	Base	**Recovered**	Δ
HumanEval	60.98	**64.02**	+3.05
MBPP	68.30	**69.05**	+0.75
BigCodeBench	23.00	**26.23**	+3.23
LiveCodeBench	9.70	**13.55**	+3.85

Table 3: Utility of extracted data. Pass@1 (%) before (Base) and after (Recovered) fine-tuning Dream-v0-Instruct-7B on extracted data from Dream-Coder.

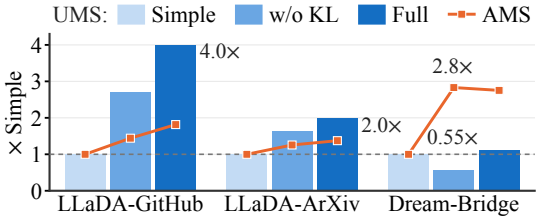


Figure 4: Component ablation at $R = 1{,}024$: UMS (bars) and AMS (line), each relative to Simple.

settings are provided in Appendix E.3. As shown in Table 3, pass@1 improves across all four coding benchmarks by 0.75–3.85 percentage points, including 23.00% to 26.23% on BigCodeBench and 9.70% to 13.55% on LiveCodeBench. These results demonstrate that the extracted data provides effective supervision for downstream coding tasks.

Component Ablation. The complete corrector covers the most records on all three targets, and its KL term spreads matches over distinct records. Figure 4 adds one component at a time, with candidates held fixed: Simple, direct generation with the frozen model; w/o KL, the corrector trained with both levels of supervision but $\beta = 0$ in Equation (13); and Full, with the KL term restored. On both LLaDA targets the two measures rise together, and the KL term adds +75 and +79 UMS. On Dream-Bridge, w/o KL reaches 2.8× Simple in AMS while its UMS falls to 0.55×, because 550 of 1,024 outputs land on a single record. Restoring the KL term keeps AMS at 2.8× and raises UMS to 1.10×: it adds no matches but distributes them over more records.

5 Related Work

Training Data Extraction. Extraction attacks either sample at scale and filter the candidates (Carlini et al., 2021; Nasr et al., 2025) or generate from a known prefix (Hayes et al., 2025), and diffusion-side work targets images (Carlini et al., 2023; Chen et al., 2026c; Wu et al., 2025) or reconstructs DLM records from visible token subsets (Luo et al., 2026b; Wang & Asokan, 2026). Closest to our setting, Barbero et al. (2025) extract alignment data from open autoregressive models with minimal prompts and evaluate semantic rather than verbatim recovery; we keep their recovery unit but target DLMs and guide generation instead of sampling at scale. Unlike prior work that supplies part of the target, we query with masks alone and count distinct records at declared budgets.

Membership Inference. Membership inference compares a model's loss or its calibrated variants between members and non-members (Yeom et al., 2018; Shi et al., 2024; Zhang et al., 2025), and has been adapted to diffusion models through denoising errors (Duan et al., 2023) and to fine-tuned DLMs through masked-token reconstruction (Chen et al., 2026b; Zhai et al., 2026). D-Miner uses such a signal to rank the model's own generations for corrector training rather than to classify supplied text.

Steering Diffusion Language Models. Activation steering (Turner et al., 2023; Rimsky et al., 2024) has been carried to DLMs for safety control (Shnaidman et al., 2025; Lee & Han, 2026), and DLMs have also been steered through logits (An & Han, 2026), posterior guidance (Rector-Brooks et al., 2025; Wan et al., 2026), and joint sampling (Bansal & Sanghavi, 2026). Where these aim at an attribute or a reward, D-Miner aims a state-conditioned corrector at training records.

6 Conclusion

We study alignment data extraction from DLMs without target prefixes or partial records. D-Miner steers denoising with a lightweight corrector on intermediate hidden states, trained on the model's own generations selected by membership signals, and our analysis shows that an ideal corrector conditions the output distribution on successful recovery. Across 12 target DLMs, D-Miner recovers more distinct records than direct generation on every target and passes the coverage ceiling of filtering as the return budget grows, and the extracted data improves another model on four coding benchmarks.

AI USE STATEMENT

Generative AI aided methodology and experimental-design feedback, mathematical formulation and checks, interpretation of preliminary results, literature search, manuscript revision, and LaTeX formatting. The authors are responsible for reviewing these suggestions and for the final text, claims, and AI-assisted artifacts.

ETHICS STATEMENT

We study extraction from open-weight or locally constructed models using authorized research-access or model-card-associated corpora. The purpose is to characterize training-content exposure. We do not report private records or unredacted sensitive generations. Misuse risks, release controls, and mitigations are discussed in Appendix I.

REPRODUCIBILITY STATEMENT

Section 3 specifies MScore, the corrector, and its training objectives, and Section 4 the evaluation protocol. The appendices give the target models and verification sets (Appendix A), the generation, scoring, and corrector configurations and baseline definitions (Appendix B), and the proofs with their assumptions (Appendix F). Code is available at the anonymous repository linked in the abstract.

REFERENCES

Hyeseon An and Yo-Sub Han. DLM-SWAI: Steering diffusion language models before they unmask, 2026. URL https://arxiv.org/abs/2605.29626.

Parikshit Bansal and Sujay Sanghavi. Enabling approximate joint sampling in diffusion LMs. *arXiv preprint arXiv:2509.22738*, 2026. URL https://arxiv.org/abs/2509.22738v2. Version 2.

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.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 267–284, 2019. URL https://www.usenix.org/conference/usenixsecurity19/presentation/carlini.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2633–2650, 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting-training.

Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 5253–5270, 2023. URL https://www.usenix.org/conference/usenixsecurity23/presentation/carlini.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-Embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335, Bangkok, Thailand, aug 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.137. URL https://aclanthology.org/2024.findings-acl.137/.

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)*, pp. 1187–1206. USENIX Association, 2026a. URL https://www.usenix.org/conference/usenixsecurity26/presentation/chen-yuetian.

Yuetian Chen, Kaiyuan Zhang, Yuntao Du, Edoardo Stoppa, Charles Fleming, Ashish Kundu, Bruno Ribeiro, and Ninghui Li. Membership inference attacks against fine-tuned diffusion language models. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=oWKJursYpW.

Yunhao Chen, Shujie Wang, Difan Zou, and Xingjun Ma. SIDE: Surrogate conditional data extraction from diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 128–136, 2026c. doi: 10.1609/aaai.v40i1.36972. URL https://ojs.aaai.org/index.php/AAAI/article/view/36972.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dParallel: Learnable parallel decoding for dLLMs. In *International Conference on Learning Representations*, 2026d. URL https://openreview.net/forum?id=hVOcstAURb.

Debeshee Das, Jie Zhang, and Florian Tramèr. Blind baselines beat membership inference attacks for foundation models. In *2025 IEEE Security and Privacy Workshops (SPW)*, pp. 118–125, 2025. doi: 10.1109/SPW67851.2025.00016. URL https://doi.org/10.1109/SPW67851.2025.00016.

Jinhao Duan, Fei Kong, Shiqi Wang, Xiaoshuang Shi, and Kaidi Xu. Are diffusion models vulnerable to membership inference attacks? In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8717–8730. PMLR, 2023. URL https://proceedings.mlr.press/v202/duan23b.html.

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? In *The First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=av0D19pSkU.

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 *Advances in Neural Information Processing Systems*, volume 37, pp. 134981–135010, 2024a. doi: 10.52202/079017-4290. URL https://papers.nips.cc/paper_files/paper/2024/hash/f36ad694188bb4c4bbbd61e2038e069e-Abstract-Conference.html.

Xiaomeng Fu, Xi Wang, Qiao Li, Jin Liu, Jiao Dai, and Jizhong Han. Model will tell: Training membership inference for diffusion models. *arXiv preprint arXiv:2403.08487*, 2024b. doi: 10.48550/arXiv.2403.08487. URL https://arxiv.org/abs/2403.08487.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020. doi: 10.48550/arXiv.2101.00027. URL https://arxiv.org/abs/2101.00027.

Jamie Hayes, Marika Swanberg, Harsh Chaudhari, Itay Yona, Ilia Shumailov, Milad Nasr, Christopher A. Choquette-Choo, Katherine Lee, and A. Feder Cooper. Measuring memorization in language models via probabilistic extraction. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 9266–9291, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.469. URL https://aclanthology.org/2025.naacl-long.469/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. doi: 10.48550/arXiv.2207.12598. URL https://arxiv.org/abs/2207.12598.

Fei Kong, Jinhao Duan, RuiPeng Ma, Heng Tao Shen, Xiaofeng Zhu, Xiaoshuang Shi, and Kaidi Xu. An efficient membership inference attack for the diffusion model by proximal initialization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=rpH9FcCEV6.

Yejin Lee and Yo-Sub Han. Adaptive steering and remasking for safe generation in diffusion language models, 2026. URL https://arxiv.org/abs/2605.13043.

Jia-Nan Li, Jian Guan, Wei Wu, and Chongxuan Li. ReFusion: A diffusion large language model with parallel autoregressive decoding. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=LBtWaUc7FE.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Grace Luo, Jiahai Feng, Trevor Darrell, Alec Radford, and Jacob Steinhardt. Learning a generative meta-model of LLM activations. In *International Conference on Machine Learning*, 2026a. URL https://arxiv.org/abs/2602.06964.

Xiaoyu Luo, Wenrui Yu, Qiongxiu Li, and Johannes Bjerva. Characterizing memorization in diffusion language models: Generalized extraction and sampling effects. *arXiv preprint arXiv:2603.02333*, 2026b. doi: 10.48550/arXiv.2603.02333. URL https://arxiv.org/abs/2603.02333.

Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schölkopf, 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, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.719. URL https://aclanthology.org/2023.findings-acl.719/.

John Xavier Morris, Junjie Oscar Yin, Woojeong Kim, Vitaly Shmatikov, and Alexander M. Rush. Approximating language model training data from weights. In *The Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=cQechnXCQt.

Niels Mündler, Jasper Dekoninck, and Martin Vechev. Constrained decoding of diffusion LLMs with context-free grammars. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7Sph4KyeYO.

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*, pp. 82363–82435, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/cce0e917b050208170151f77b497fc71-Abstract-Conference.html.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=KnqiC0znVF.

Jingyang Ou, Jiaqi Han, Minkai Xu, Shaoxuan Xu, Jianwen Xie, Stefano Ermon, Yi Wu, and Chongxuan Li. Principled RL for diffusion LLMs emerges from a sequence-level perspective. In *The Fourteenth International Conference on Learning Representations*, pp. 4941–4966, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0907335ecf28faf15be54485dbcbe70e-Abstract-Conference.html.

Yan Pang and Tianhao Wang. Black-box membership inference attacks against fine-tuned diffusion models. In *Network and Distributed System Security (NDSS) Symposium 2025*, 2025. doi: 10.14722/ndss.2025.230324. URL https://www.ndss-symposium.org/wp-content/uploads/2025-324-paper.pdf.

Jarrid Rector-Brooks, Mohsin Hasan, Zhangzhi Peng, Zachary Quinn, Chenghao Liu, Sarthak Mittal, Nouha Dziri, Michael Bronstein, Yoshua Bengio, Pranam Chatterjee, Alexander Tong, and Avishek Joey Bose. Steering masked discrete diffusion models via discrete denoising posterior prediction. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.08134.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering Llama 2 via contrastive activation addition. 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. 15504–15522, Bangkok, Thailand, aug 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. doi: 10.48550/arXiv.2002.05202. URL https://arxiv.org/abs/2002.05202.

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. URL https://openreview.net/forum?id=zWqr3MQuNs.

Adi Shnaidman, Erin Feiglin, Osher Yaari, Efrat Mentel, Amit Levi, and Raz Lapid. Activation steering for masked diffusion language models, 2025. URL https://arxiv.org/abs/2512.24143. Accepted at ReALM-GEN @ ICLR 2026.

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, 2017. doi: 10.1109/SP.2017.41. URL https://doi.org/10.1109/SP.2017.41.

Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? Investigating data replication in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6048–6058, June 2023. doi: 10.1109/CVPR52729.2023.00586.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023. doi: 10.48550/arXiv.2308.10248. URL https://arxiv.org/abs/2308.10248.

Zhengyan Wan, Yidong Ouyang, Liyan Xie, Fang Fang, Hongyuan Zha, and Guang Cheng. Discrete guidance matching: Exact guidance for discrete flow matching. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=N1RYhOg6ib.

Cheng Wang, Yiwei Wang, Bryan Hooi, Yujun Cai, Nanyun Peng, and Kai-Wei Chang. Con-ReCall: Detecting pre-training data in LLMs via contrastive decoding. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 1013–1026, Abu Dhabi, UAE, 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.68/.

Yihan Wang and N. Asokan. Extracting training data from diffusion language models via infilling, 2026. URL https://arxiv.org/abs/2605.24173.

Zhepeng Wang, Runxue Bao, Yawen Wu, Jackson Taylor, Cao Xiao, Feng Zheng, Weiwen Jiang, Shangqian Gao, and Yanfu Zhang. Unlocking memorization in large language models with dynamic soft prompting. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9782–9796, 2024. doi: 10.18653/v1/2024.emnlp-main.546.

Xiaoyu Wu, Jiaru Zhang, and Zhiwei Steven Wu. Leveraging model guidance to extract training data from personalized diffusion models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 68060–68077. PMLR, 2025. URL https://proceedings.mlr.press/v267/wu25as.html.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. ReFT: Representation finetuning for language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 63908–63962. Curran Associates, Inc., 2024. doi: 10.52202/079017-2041. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/75008a0fba53bf13b0bb3b7bff986e0e-Paper-Conference.pdf.

Roy Xie, Junlin Wang, Ruomin Huang, Minxing Zhang, Rong Ge, Jian Pei, Neil Zhenqiang Gong, and Bhuwan Dhingra. ReCaLL: Membership inference via relative conditional log-likelihoods. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8671–8689, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.493. URL https://aclanthology.org/2024.emnlp-main.493/.

Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream-Coder 7B: An open diffusion language model for code, 2025. URL https://arxiv.org/abs/2509.01142.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned LLMs with nothing. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Pnk7vMbznK.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pp. 268–282. IEEE, 2018. doi: 10.1109/CSF.2018.00027. URL https://arxiv.org/abs/1709.01604.

Weichen Yu, Tianyu Pang, Qian Liu, Chao Du, Bingyi Kang, Yan Huang, Min Lin, and Shuicheng Yan. Bag of tricks for training data extraction from language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 40306–40320. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/yu23c.html.

Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. ACECODER: Acing coder RL via automated test-case synthesis. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12023–12040, Vienna, Austria, jul 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.587. URL https://aclanthology.org/2025.acl-long.587/.

Shengfang Zhai, Leo Marchyok, Yuling Shi, Huanran Chen, Yinpeng Dong, Jiaheng Zhang, and Sanghyun Hong. Membership inference in fine-tuned diffusion language models via token-level memorization asymmetry, 2026. URL https://arxiv.org/abs/2609.00873. Findings of EMNLP 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*, 2025. URL https://openreview.net/forum?id=ZGkfoufDaU.

APPENDIX CONTENTS

- **Appendix A: Target Data and Verification Protocol.** Target models, post-training data, controlled SFT splits and training, retrieval-based verification, threshold selection, and verification isolation.
- **Appendix B: Implementation Details.** Mask-only query wrappers, reveal order, generation, scoring, and corrector-training configs, extraction baselines, and compute and runtime.
- **Appendix C: Calibration and Sensitivity Analyses.** Mask-length selection and pattern-budget sensitivity.
- **Appendix D: Membership Validation of MScore.** DLM-adapted membership-inference baselines, evaluation protocol, and full results.
- **Appendix E: Additional Extraction Results.** Sensitivity to the verification threshold, qualitative examples, and data utility evaluation details.
- **Appendix F: Ideal Correctors, Realizability, and Recovery Analysis.** Proofs of Theorems 3.1 and 3.2, the soft tilt and its token-level form, what flow matching and masked-token CE identify, the corrector operator, score enrichment, and budgeted unique recovery.
- **Appendix G: Extended Related Work.**
- **Appendix H: Limitations.**
- **Appendix I: Ethical and Security Considerations.**

A TARGET DATA AND VERIFICATION PROTOCOL

This appendix defines the target models, post-training data, and verification protocol used in our evaluation. The described corpora are used only for post-hoc verification. They are not used to construct D-Miner's candidate pools, membership scores (MScore), corrector training sets, or checkpoint selection. We distinguish record-level verification, where the evaluated post-training records are known, from corpus-level verification, where the exact training manifest or checkpoint revision is not fully recoverable from public artifacts.

A.1 TARGET MODEL FAMILIES

The experiments cover two post-training regimes. The first regime is supervised fine-tuning (SFT). For this setting, we construct target DLMs by fine-tuning two base model families, Dream-v0-Base-7B and LLaDA-8B-Base, on public text corpora. These models provide controlled targets for evaluating whether mask-only denoising can recover examples from known fine-tuning datasets.

The second regime is RL or instruction-style post-training. For this setting, we evaluate publicly released post-trained DLM checkpoints, including Dream-Coder, Dream-Bridge, ESPO-Code, and ESPO-MATH. These models are not retrained in our experiments. Instead, we associate each checkpoint with the public dataset or model-card-specified corpus that is recoverable from released metadata. When the exact training subset is not released, we use the corresponding public parent corpus for corpus-level verification.

A.2 TARGET POST-TRAINING DATA

SFT targets. For the SFT setting, we fine-tune Dream-v0-Base-7B and LLaDA-8B-Base on four MIMIR corpora used in the main extraction experiments: GitHub, ArXiv, HackerNews, and PubMed Central. These datasets cover code, scientific writing, online discussion, and biomedical text, respectively. Using the same corpus family across the two base DLMs allows us to separate model-family effects from corpus effects.

The SFT targets are constructed locally. The controlled checkpoints are fit on the 900-record training splits in Table 5; validation and test records are held out of fine-tuning. Training-record counts refer to this 900-record split. For the SFT rows of Table 2, $\mathcal{D}_{\mathrm{ver}}$ is the 900-record training split of the corresponding MIMIR corpus, i.e. exactly the records the target was fine-tuned on; validation and test records are not in $\mathcal{D}_{\mathrm{ver}}$. Each hit is therefore a verified match to a fine-tuning record. The candidate-scoring diagnostics additionally use PileCC (Gao et al., 2020) together with the MIMIR-style corpora, but these auxiliary diagnostics are not used to train the corrector for the main experiments.

Target model	Base model	Dataset / corpus	Training strategy
Dream-GitHub	Dream-v0-Base-7B	MIMIR GitHub	SFT
Dream-ArXiv	Dream-v0-Base-7B	MIMIR ArXiv	SFT
Dream-HackerNews	Dream-v0-Base-7B	MIMIR HackerNews	SFT
Dream-PubMed	Dream-v0-Base-7B	MIMIR PubMed Central	SFT
LLaDA-GitHub	LLaDA-8B-Base	MIMIR GitHub	SFT
LLaDA-ArXiv	LLaDA-8B-Base	MIMIR ArXiv	SFT
LLaDA-HackerNews	LLaDA-8B-Base	MIMIR HackerNews	SFT
LLaDA-PubMed	LLaDA-8B-Base	MIMIR PubMed Central	SFT
Dream-Coder	Dream-family 7B	`Dream-org/Dream-Coder-RL-17k`	RL
Dream-Bridge	Dream-family 7B	General-Distillation subset of `ianncity/KIMI-K2.5-1000000x`	RL
ESPO-Code	ESPO-series 8B	`TIGER-Lab/AceCode-87K`	RL
ESPO-MATH	ESPO-series 8B	`qwedsacf/competition_math` train split	RL

Table 4: Target models, associated post-training or verification corpora, and training strategies used in the extraction experiments. For released post-trained models whose exact training manifest is not recoverable, the dataset column reports the most specific public corpus used for post-hoc verification.

Corpus	Train	Validation	Test
MIMIR GitHub	900	100	1000
MIMIR ArXiv	900	100	1000
MIMIR HackerNews	900	100	1000
MIMIR PubMed Central	900	100	1000

Table 5: Dataset splits for the MIMIR corpora used to construct the main SFT targets. The test split is kept unchanged from the original MIMIR test file as a held-out reference split.

RL and post-trained targets. For RL/post-trained targets, we use released open-weight checkpoints and set $\mathcal{D}_{\text{ver}}$ to their released or associated public corpora. The repository identifiers below specify the public source of each checkpoint or verification corpus; they are not used as prompts, candidate records, or corrector training data during extraction.

- **Dream-Coder.** We evaluate Dream-Coder (Xie et al., 2025) against the released RL-17k corpus from `Dream-org/Dream-Coder-RL-17k`. The released files contain 17,030 rows; we remove the 4,956 prompt-only rows and use the remaining 12,074 records as $\mathcal{D}_{\text{ver}}$. This setting uses record-level verification.
- **Dream-Bridge.** We evaluate Dream-Bridge using corpus-level verification with the General-Distillation subset of `ianncity/KIMI-K2.5-1000000x`, which is associated with the released checkpoint `WilhelmH/Bridge-7b-Diffusion-Full`.
- **ESPO-Code.** We evaluate ESPO-Code (Ou et al., 2026) using corpus-level verification with `TIGER-Lab/AceCode-87K` (Zeng et al., 2025), the public corpus associated with the released checkpoint `GSAI-ML/ESPO-Code`.
- **ESPO-MATH.** We evaluate ESPO-MATH using corpus-level verification with the train split of `qwedsacf/competition_math`, a release of the MATH dataset (Hendrycks et al., 2021), which is associated with the released checkpoint `GSAI-ML/ESPO-MATH`.

A.3 Dataset Split

We partition the data into three disjoint sets to reflect a realistic attack scenario. For the controlled SFT targets used in the main extraction experiments, we generate the splits with `generate_fullparam_configs.py`. The script uses random seed 42 and a validation ratio of 0.1: it shuffles the original MIMIR training file, assigns 10% of the training records to validation, keeps the remaining records for fine-tuning, and leaves the original MIMIR test file unchanged as a held-out reference split. Table 5 summarizes the resulting splits for the main SFT corpora.

Training parameter	Value
Tokenizer max length	512 for Dream; 256 for LLaDA
Per-device batch size	1
Gradient accumulation steps	6
Number of GPUs	8
Effective global batch size	48
Learning rate	5×10^{-5}
Weight decay	0.1
Warmup steps	100
Training epochs	4
Evaluation frequency	epoch
Precision	BF16
Gradient checkpointing	true
Early-stopping patience	3

Table 6: Local SFT training hyperparameters for the main Dream-family and LLaDA-family MIMIR targets. The table omits dataset identifiers and checkpoint-management fields.

A.4 SFT Target Training Configurations

We report the SFT configuration used to construct the local targets. Table 6 summarizes the shared training settings for the main Dream-family and LLaDA-family MIMIR SFT targets.

A.5 Embedding-based Retrieval for Verification

We use embedding-based nearest-neighbor retrieval to verify whether a generated output recovers the content of a record in $\mathcal{D}_{\text{ver}}$, allowing paraphrases but not merely shared topics or templates. Each generated output and each record in the corresponding verification corpus is encoded with BGE-M3 (Chen et al., 2024). We then compute cosine similarity in the embedding space, assign each generated output to its nearest corpus record, and determine a semantic match using a predetermined threshold τ:

$$\begin{aligned} d_E^\star(y) &= \arg\max_{d \in \mathcal{D}_{\text{ver}}} \cos\big(E(y), E(d)\big), \\ s_E(y; \mathcal{D}_{\text{ver}}) &= \max_{d \in \mathcal{D}_{\text{ver}}} \cos\big(E(y), E(d)\big), \\ V_\tau(y; \mathcal{D}_{\text{ver}}) &= \mathbf{1}\{s_E(y; \mathcal{D}_{\text{ver}}) \geq \tau\}. \end{aligned} \tag{15}$$

Here E denotes the fixed text encoder and $\mathcal{D}_{\text{ver}}$ the verification corpus for the target checkpoint. This retrieval step is performed only after all generations have been produced. A match is evidence of semantic recovery, not independent proof of training membership; we therefore further validate extracted data through the data utility evaluation (Section 4.4 and Appendix E.3).

For Dream-Coder and the preliminary RL-17k analyses, the retrieval database uses the same filtered RL-17k verification set of Appendix A.1. This index is used for the reported Dream-Coder calibration and extraction results.

A.6 Threshold Selection

Neural-embedding similarity identifies semantic recovery when surface-form differences make exact or string matching too restrictive (Barbero et al., 2025). The numerical threshold of prior work does not transfer directly, because our encoder, domains, sequence formats, and verification-corpus sizes differ, so we fix one operating point per post-training regime before comparing extraction methods: $\tau_{\text{SFT}} = 0.80$ for the domain-SFT targets and $\tau_{\text{RL}} = 0.90$ for the released RL targets. Table 7 shows why the two regimes need different operating points: outputs of RL-tuned models sit much closer to their reference corpus than outputs of SFT targets do, so a single threshold would either admit most RL outputs or almost no SFT output.

Each threshold is applied unchanged to D-Miner, Simple, both filters, both denoising-step settings, and every seed within its regime, and it is used only in post-hoc verification: it affects neither generation, candidate selection, nor corrector training. A match therefore denotes an embedding-

Target	Mean	P90	$s_E(y) \geq 0.90$
Dream-GitHub (SFT)	0.758	0.818	$< 0.1\%$
Dream-ArXiv (SFT)	0.687	0.776	0.0%
Dream-Coder (RL)	0.862	0.907	22.8%

Table 7: Representative nearest-neighbor score distributions illustrating the family-specific verification thresholds. Values average the available Simple output score files and seeds of each target.

verified semantic match to a reference record, and all comparisons are made between methods under the same target, verifier, and threshold. Because the two regimes use different thresholds, their absolute counts are not compared with each other. Appendix E.1 re-thresholds the same frozen scores at $\tau \in \{0.80, 0.85, 0.90\}$.

A.7 Verification Isolation and Leakage Controls

The verification protocol is isolated from attack construction. D-Miner does not use records from the verification corpus to form candidate pools, compute membership scores (MScore), select corrector training candidates, train the corrector, select checkpoints, or construct mask-only inputs. The target model receives only the prescribed mask-only wrapper and its own generated candidates; it is never supplied with target records, target prefixes, nearest-neighbor results, or corpus-derived examples during corrector construction.

The verification corpus is queried only after generation has completed. At that stage, it is used for nearest-neighbor retrieval, target-side deduplication, thresholding, and metric computation. This separation is important for interpreting the extraction results: the reported matches come from post-hoc verification of generated text, not from injecting target-corpus information into the extraction procedure.

For corpus-level targets, $\mathcal{D}_{\text{ver}}$ is the recoverable public source corpus associated with the checkpoint. The reported match is the nearest-neighbor record in that corpus under its regime's threshold.

B Implementation Details

This appendix collects the query wrappers, the reveal order, the generation, scoring, and corrector-training configs, the extraction baselines, and the compute/runtime accounting used in the main extraction experiments. Unless stated otherwise, these choices are fixed before the final extraction runs.

B.1 Mask-Only Query Wrappers

All extraction experiments use mask-only inputs. A wrapper is the minimal checkpoint-specific scaffold required to invoke a model, together with a contiguous span of mask tokens. These scaffolds may include boundary tokens, chat-format delimiters, or fixed field labels required by a checkpoint family, but they do not contain target prefixes, ground-truth tokens, demonstrations, or sample-specific content.

ESPO targets differ from the other targets because they use the `chat_system_user_field_masks` wrapper with a fixed system prompt and a masked user field. Simple and D-Miner use the same wrapper for each ESPO target. During verification, wrapper tokens are removed before retrieval and metric computation.

B.2 Reveal Order

A masked diffusion sampler commits a few positions per step, and the rule that chooses them is part of the sampler, so every method on a target decodes under the same order (Table 9). We keep the model's own entropy order for the Dream family and draw the order uniformly at random for LLaDA and ESPO, whose default `low_confidence` rule selects positions by the probability of the token just sampled there and thereby changes the law of the generated text.

ESPO-Code, `chat_system_user_field_masks`.

```
<|im_start|>system
You are an AI assistant skilled at writing programming problems.
<|im_end|>
<|im_start|>user
[question]
[MASK] [MASK] ... [MASK]
<|im_end|>
<|im_start|>assistant
```

ESPO-MATH, `chat_system_user_field_masks`.

```
<|im_start|>system
You are an AI assistant skilled at solving math problems.
<|im_end|>
<|im_start|>user
[problem]
[MASK] [MASK] ... [MASK]
<|im_end|>
<|im_start|>assistant
```

Table 8: ESPO mask-only query wrapper examples. Both examples use a fixed system prompt required by the checkpoint family and a masked user field.

B.3 Generation, Scoring, and Corrector Configs

Table 9 summarizes the fixed generation, scoring, and corrector-training configs used by the main extraction runs. These values are fixed before final extraction; the verification corpus is used only after generation for retrieval and metric computation.

B.4 Extraction Baselines

Direct generation (Simple) uses the same mask-only input wrapper, mask length, decoding temperature, and generation budget B as D-Miner, but applies no activation correction. This controls for the target DLM's ordinary masked generation behavior and isolates the effect of the corrector.

Fixed-pool filters. The two filtering baselines of Section 4.1 operate on the shared Simple pool of $B = 3R$ outputs per target and step setting, pooled over the three seed blocks. MScore-Filter scores every pool output with the fixed MScore configuration ($T = 8$, mask ratio $t = 0.8$) and returns the R highest-scoring outputs. KMeans-Filter L2-normalizes candidate-only BGE-M3 embeddings, runs KMeans++ with $K = R$ clusters, 20 initializations, and the generation seed, and returns the output nearest each centroid. Neither filter uses a score or cluster-cohesion threshold, and each selection manifest is frozen before the verification index is loaded. The filters add $3RT$ masked reconstruction evaluations or one embedding pass over the pool, whereas D-Miner scores only its pilot outputs before corrector training; we report return-matched, not compute-matched, comparisons.

B.5 Experimental Environment and Runtime

All attacked checkpoints considered in this work fall within the 7B–8B parameter range. The Dream-family targets, including Dream-v0-Base-7B, Dream-Coder, and Dream-Bridge, are 7B-class diffusion language models, while the LLaDA-family SFT targets and ESPO-series checkpoints are 8B-class diffusion language models. We use BGE-M3 as a candidate-only feature encoder for KMeans-Filter and as the fixed retrieval encoder for post-hoc verification. Candidate selection never loads or searches the verification index; BGE-M3 is not an attacked target model and is not involved in generation.

Experimental environment. We run all experiments on GPU servers equipped with eight NVIDIA RTX A6000 GPUs. Jobs are distributed or sharded over the available GPUs according to the target model and output budget.

Config	Value
Generation	
Mask length	$L = 256$
Denoising steps	$S \in \{128, 256\}$
Outputs per seed	1,024
Seeds	three seed blocks per setting (7000, 100000, and 200000)
Temperature	0.7
EOS penalty / alg. temp.	3.0 / 0.0
Reveal order	`entropy` for Dream-family, Dream-Coder, and Dream-Bridge; uniformly random for LLaDA and ESPO, replacing their default `low_confidence` (Appendix B.2); every method on a target uses the same order.
Candidate scoring (MScore)	
Scoring trials / mask ratio	$T = 8$, $t = 0.8$ (Equation (11))
Pilot pool	100 outputs of the frozen model from fully masked inputs, scored with MScore
Training candidates	$N_{\text{train}} = 32$ highest-scoring pilot outputs, taken directly by MScore rank with equal weights and no further filtering; 32 more held out for validation
Corrector	
Intervention site	input of block 16, content slots only
Architecture	conditional activation flow: width 1,024, MLP width 2,048, three SwiGLU blocks (about 44M parameters)
Conditioning $\boldsymbol{\kappa}_{t,i}$	mask ratio ρ_t, per-slot mask indicator, absolute slot index; the velocity also receives flow time u
Activation coordinates	per-dimension standardization of the cached block-16 inputs
Hidden-state-level supervision	1,000 flow-matching updates on corrupted-candidate activations (Equation (12))
Edit	start noise $u_\star = 0.25$, $K_{\text{flow}} = 4$ Euler steps
Token-level objective	masked-token CE (mean over masked slots) $+ \beta\, D_{\text{KL}}(p_{\theta,i} \| p_{\theta,\Phi,i})$ on the training candidates (Equation (13))
Divergence strength β	1 for every target (equal weight on the two terms); $\beta = 0$ is the ablation arm
Mask-count sampling	uniform k with extra mass 0.2 on $k = L$ and 0.3 on $k = \lceil 0.95L \rceil$, importance-corrected
Token-level training	lr 10^{-5}, effective batch 16, at most 600 updates, validation every 50 updates, early stopping after two validations without a 0.001 improvement, checkpoint with the lowest validation objective

Table 9: Generation, scoring, and corrector configs used in the main extraction runs.

Detailed hyperparameters. The decoding, scoring, and corrector-training hyperparameters are reported in Table 9, and the SFT-training configuration in Appendix A.4. These settings are fixed before the final extraction runs.

Efficiency and runtime. The controlled SFT training for the eight main MIMIR checkpoints uses approximately 42.6 GPU-hours in total on eight-GPU nodes. The completed generation logs for the reported extraction runs, ablations, and calibration sweeps account for approximately 240 GPU-hours. Once the corrector is trained, applying D-Miner uses the same denoising stack as the uncorrected baseline, plus the activation edit at each step and post-hoc retrieval verification.

C Calibration and Sensitivity Analyses

C.1 Mask Length Selection

Before fixing the decoding setting for the extraction experiments, we first select the mask span using a Dream-Coder pilot study. Dream-Coder is a convenient calibration target because its RL-17k post-training corpus is released and can be used for post-hoc nearest-neighbor analysis. This pilot study is used only to choose global decoding defaults; it is not used to construct D-Miner's candidate scores or to train the corrector. Because RL-17k also informs these defaults, Dream-Coder is not an independent confirmatory target.

Mask L	Steps	Mean Sim.	Median	Sim. ≥ 0.90	Distinct records	Records ≥ 0.90
64	64	$0.8693_{0.0305}$	0.8777	76	190	5
128	128	$0.8575_{0.0367}$	0.8676	63	310	35
256	256	$0.8706_{0.0336}$	0.8740	190	421	85
512	512	$0.8473_{0.0627}$	0.8579	169	370	86

Table 10: Dream-Coder mask-length sweep with denoising steps matched to the mask length. Each row contains 1,024 generated samples and uses RL-17k-only nearest-neighbor retrieval with BGE-M3. Mean similarity is reported with standard deviation as a subscript; the shaded row is used in the main extraction experiments.

We fix the sampling temperature to 0.7 and sweep the mask length $L \in \{64, 128, 256, 512\}$. For each L, we generate 1,024 samples from a fully masked input, use top-$p = 0.95$, EOS penalty 3.0, `max_new_tokens`$=L$, and set the denoising-step budget to $S = L$. We then embed the generated outputs and RL-17k records with BGE-M3 and compute nearest-neighbor cosine similarities. Table 10 reports the resulting nearest-neighbor statistics, including the number of outputs above the 0.90 similarity threshold and the number of distinct matched records.

The sweep shows that very short mask spans cover too little of the target corpus, whereas very long spans are less stable. The setting $L = 256$ gives the best overall trade-off: it achieves the highest mean nearest-neighbor similarity, the largest number of outputs above the 0.90 threshold, and the largest number of distinct nearest-neighbor records before thresholding. Although $L = 512$ recovers one additional record above the 0.90 threshold, it has lower average similarity and fewer distinct nearest-neighbor records. We therefore use $L = 256$ as the default mask length for the main extraction experiments.

C.2 Pattern-Budget Sensitivity

We sweep mask ratios and scoring-trial budgets on Dream-Coder for MScore, focusing on the strong 0.6–0.8 interval. Here AUROC is computed over Dream-Coder's own mask-only generations: a generation is labeled a member when its nearest RL-17k record has similarity $s_E(y) \geq 0.90$, the RL threshold of Appendix A.6, and a non-member otherwise. Fig. 5 shows gains saturating at $T = 8$; larger T adds reconstruction cost with little extra AUROC, so all experiments use $T = 8$.

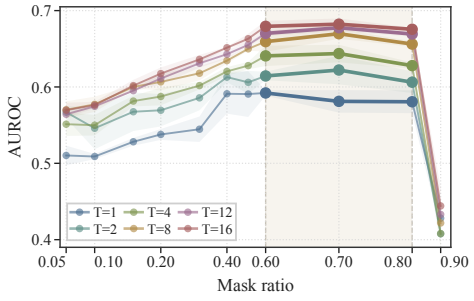


Figure 5: Mask-ratio/pattern-budget calibration; gains plateau after $T = 8$ near mask ratios 0.6–0.8.

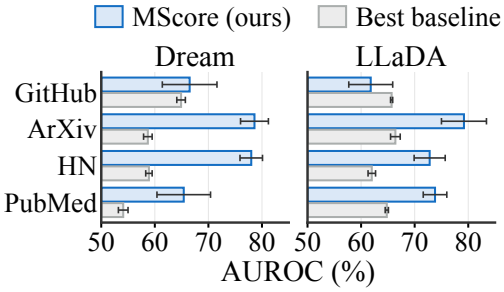


Figure 6: **Member/non-member discrimination.** MScore vs. the best baseline across models.

D Membership Validation of MScore

D.1 DLM-Adapted Membership-Inference Baselines

We adapt likelihood-based membership-inference baselines to diffusion language models through masked-token reconstruction, following prior language-model MIAs (Yeom et al., 2018; Carlini et al., 2021; Duan et al., 2024). For each input, we sample random mask patterns, query the DLM on the

partially masked sequence, and measure the negative log-likelihood of the original tokens at the masked positions. All scores are oriented so that larger values indicate stronger member-like evidence.

- **Loss.** Directly uses the negative mean reconstruction loss over the sampled masks. Intuitively, examples that the model reconstructs more confidently receive higher membership scores.
- **ZLIB.** Normalizes the reconstruction loss with the ZLIB-compressed length ratio of the input (Carlini et al., 2021). This reduces the effect of text complexity and repetitiveness on raw likelihood scores.
- **Lowercase.** Compares the reconstruction loss of the original text with that of its lowercased version (Carlini et al., 2021). The score captures whether the original surface form is substantially easier for the model to reconstruct than a simple case-normalized variant.
- **Min-K%.** Collects the true-token probabilities over all masked positions and focuses on the lowest-probability 20% of tokens (Shi et al., 2024). This emphasizes unusually difficult tokens whose likelihood may be more informative for membership.
- **Min-K%++.** Uses the same masked-token collection as Min-K%, but replaces the raw probability criterion with the normalized log-probability score (Zhang et al., 2025).
- **Neighbor.** Compares the reconstruction score of the original text with that of a perturbed neighboring text (Mattern et al., 2023). We use one neighbor for each candidate and take the reconstruction-score gap as the membership signal.
- **ReCall.** Measures how external prefixes affect the candidate's reconstruction score (Xie et al., 2024). We implement it with masked-token reconstruction rather than left-to-right likelihood.
- **CON-ReCall.** Contrasts prefixes that are expected to be member-like against prefixes that are expected to be non-member-like (Wang et al., 2025), again using masked-token reconstruction rather than left-to-right likelihood.
- **BoWs.** A query-free baseline that trains a shallow classifier on bag-of-words and TF-IDF features (Das et al., 2025). It tests whether lexical artifacts alone can explain membership prediction.
- **MScore (membership score).** MScore uses only the target model. For each complete candidate, it computes the ratio of the largest to the mean masked-token reconstruction loss within each mask pattern, then takes the negative average across patterns (Equation (11)). Higher values mean less worst-to-mean imbalance. The score is a ranking signal, not a membership label or probability.

D.2 Matched Candidate-Scoring Evaluation Protocol

The candidate-scoring evaluation uses five MIMIR-style corpora: HackerNews, ArXiv, GitHub, PileCC, and PubMed Central. We evaluate Dream-v0-Base-7B and LLaDA-8B-Base fine-tuned on these corpora (Ye et al., 2025; Nie et al., 2025; Duan et al., 2024), with members included in fine-tuning and non-members held out. MScore is compared against nine membership-inference baselines, and values are averaged over seeds 7000–7002.

All methods, MScore and the DLM-adapted baselines alike, are evaluated on the same examples under the same configuration: 100 member and 100 non-member examples per split, maximum sequence length 512, mask ratio 0.8, and eight random mask patterns, the candidate-scoring configuration used in the main method. We report AUROC and TPR@1%FPR. AUROC measures ranking quality over the member and non-member examples, while TPR@1%FPR measures the high-precision operating region.

D.3 Full Candidate-Scoring Results

Figure 6 summarizes MScore's AUROC against the strongest baseline on the four MIMIR domains used by our SFT extraction targets: HackerNews, ArXiv, GitHub, and PubMed Central. Table 11 reports the full comparison, including PileCC, under the protocol in Appendix D.2. MScore performs above chance on every evaluated model–dataset pair and achieves the strongest AUROC on most pairs. The exceptions are GitHub on LLaDA-Base, where Loss (62.8), ZLIB (65.7), and the query-free BoWs classifier (64.9) exceed MScore (61.8), and the TPR@1%FPR on GitHub for both models, where BoWs is highest. BoWs sees only lexical features, so on GitHub part of the member/non-member split is separable from surface text alone, as Das et al. (2025) report for other MIA benchmarks. These results support using the frozen model's masked reconstruction behavior as

Dream-Base

Method	HackerNews		ArXiv		GitHub		PileCC		PubMed	
	AUROC	TPR@1%FPR	AUROC	TPR@1%FPR	AUROC	TPR@1%FPR	AUROC	TPR@1%FPR	AUROC	TPR@1%FPR
Loss	$58.9_{0.6}$	$5.5_{0.5}$	$58.7_{0.8}$	$2.5_{1.8}$	$44.4_{0.1}$	$1.0_{0.0}$	$63.1_{0.5}$	$4.0_{0.9}$	$54.1_{0.9}$	$3.3_{0.6}$
ZLIB	$51.0_{0.4}$	$4.0_{1.0}$	$58.3_{0.1}$	$2.8_{1.0}$	$54.1_{0.2}$	$1.3_{0.3}$	$55.6_{0.1}$	$3.0_{0.9}$	$52.0_{0.1}$	$2.3_{0.3}$
Lowercase	$52.8_{4.3}$	$3.0_{1.5}$	$55.6_{0.5}$	$4.7_{2.3}$	$40.6_{1.2}$	$1.2_{0.3}$	$58.1_{1.9}$	$3.8_{1.3}$	$51.2_{4.8}$	$0.8_{0.8}$
Min-K%	$40.5_{1.7}$	$0.0_{0.0}$	$36.4_{0.5}$	$0.0_{0.0}$	$56.7_{0.2}$	$1.3_{0.3}$	$34.9_{0.9}$	$0.0_{0.0}$	$47.9_{0.9}$	$0.8_{1.0}$
Min-K%++	$46.9_{0.1}$	$1.0_{0.0}$	$44.6_{0.1}$	$0.8_{0.3}$	$50.3_{0.1}$	$1.2_{0.3}$	$43.2_{0.5}$	$0.0_{0.0}$	$49.5_{0.5}$	$1.0_{0.5}$
BoWs	$49.6_{0.4}$	$1.5_{1.3}$	$50.8_{1.9}$	$2.0_{2.2}$	$64.9_{0.8}$	$13.0_{2.3}$	$50.0_{2.3}$	$1.5_{0.5}$	$49.4_{3.1}$	$2.5_{1.5}$
ReCall	$53.0_{1.5}$	$3.2_{0.8}$	$55.3_{3.3}$	$3.2_{0.6}$	$48.2_{0.3}$	$1.5_{1.3}$	$51.9_{2.0}$	$2.8_{1.4}$	$54.1_{1.1}$	$2.3_{1.3}$
CON-ReCall	$53.0_{2.0}$	$1.8_{1.4}$	$52.4_{2.0}$	$0.3_{0.3}$	$48.3_{0.5}$	$1.7_{1.5}$	$48.6_{2.1}$	$5.0_{2.3}$	$47.7_{0.8}$	$1.5_{0.5}$
Neighbor	$53.1_{2.6}$	$1.2_{1.0}$	$49.0_{1.9}$	$1.8_{2.0}$	$52.7_{0.8}$	$3.2_{1.4}$	$50.0_{2.3}$	$0.5_{0.5}$	$46.8_{3.2}$	$0.7_{0.8}$
MScore	$\mathbf{78.0_{2.1}}$	$\mathbf{6.3_{3.8}}$	$\mathbf{78.6_{2.6}}$	$\mathbf{15.7_{6.4}}$	$\mathbf{66.5_{5.1}}$	$12.3_{6.0}$	$\mathbf{74.5_{4.1}}$	$\mathbf{8.3_{5.1}}$	$\mathbf{65.4_{5.0}}$	$\mathbf{4.0_{4.6}}$

LLaDA-Base

Method	HackerNews		ArXiv		GitHub		PileCC		PubMed	
	AUROC	TPR@1%FPR	AUROC	TPR@1%FPR	AUROC	TPR@1%FPR	AUROC	TPR@1%FPR	AUROC	TPR@1%FPR
Loss	$62.0_{0.7}$	$3.0_{0.5}$	$66.4_{0.9}$	$0.5_{0.5}$	$62.8_{0.8}$	$5.5_{1.3}$	$64.8_{0.7}$	$1.2_{0.3}$	$64.8_{0.3}$	$1.2_{0.3}$
ZLIB	$58.9_{0.8}$	$2.7_{0.3}$	$65.7_{1.1}$	$0.0_{0.0}$	$\mathbf{65.7_{0.2}}$	$5.8_{0.3}$	$62.8_{0.6}$	$2.2_{0.6}$	$64.5_{1.5}$	$4.0_{1.3}$
Lowercase	$51.6_{5.4}$	$1.3_{0.6}$	$58.4_{1.8}$	$1.8_{2.0}$	$55.7_{2.8}$	$2.2_{2.0}$	$57.1_{3.1}$	$0.8_{1.0}$	$54.1_{2.0}$	$1.7_{0.8}$
Min-K%	$36.1_{2.8}$	$0.3_{0.6}$	$33.4_{0.4}$	$0.2_{0.3}$	$38.5_{0.3}$	$0.2_{0.3}$	$34.2_{0.5}$	$0.3_{0.3}$	$34.9_{0.4}$	$0.0_{0.0}$
Min-K%++	$38.8_{0.9}$	$0.0_{0.0}$	$35.8_{0.8}$	$0.2_{0.3}$	$38.5_{0.3}$	$0.2_{0.3}$	$36.0_{0.2}$	$0.0_{0.0}$	$34.8_{1.2}$	$0.2_{0.3}$
BoWs	$49.6_{0.4}$	$1.5_{1.3}$	$50.8_{1.9}$	$2.0_{2.2}$	$64.9_{0.8}$	$\mathbf{13.0_{2.3}}$	$50.0_{2.3}$	$1.5_{0.5}$	$49.4_{3.1}$	$2.5_{1.5}$
ReCall	$47.3_{3.7}$	$1.0_{0.5}$	$45.5_{2.8}$	$1.3_{1.9}$	$42.5_{2.8}$	$2.0_{2.0}$	$43.9_{1.9}$	$2.0_{1.0}$	$43.3_{3.6}$	$1.0_{1.3}$
CON-ReCall	$48.5_{2.3}$	$1.7_{1.2}$	$50.7_{2.8}$	$1.8_{1.9}$	$46.4_{2.1}$	$2.8_{1.6}$	$51.0_{3.6}$	$1.5_{0.0}$	$46.4_{1.5}$	$3.3_{2.5}$
Neighbor	$50.4_{0.2}$	$0.8_{0.3}$	$53.0_{1.4}$	$2.2_{1.6}$	$44.3_{0.9}$	$0.5_{0.0}$	$48.1_{0.8}$	$4.0_{0.0}$	$50.6_{3.4}$	$2.8_{1.4}$
MScore	$\mathbf{72.8_{2.9}}$	$\mathbf{9.0_{6.6}}$	$\mathbf{79.2_{4.2}}$	$\mathbf{16.3_{8.7}}$	$61.8_{4.1}$	$7.7_{6.1}$	$\mathbf{71.2_{1.2}}$	$\mathbf{10.0_{3.6}}$	$\mathbf{73.8_{2.2}}$	$\mathbf{10.0_{4.0}}$

Table 11: Candidate-scoring performance under the candidate-scoring evaluation protocol. Values are percentages averaged over three seeds, with subscripts denoting standard deviations. Higher values are better. MScore is evaluated as a ranking signal for candidate selection rather than as a calibrated membership classifier. BoWs does not query the model and is identical for both base models.

a proxy signal for ranking its own mask-only outputs and selecting candidates for corrector training, without access to reference records.

The TPR@1%FPR values are more variable. This is expected because D-Miner does not use MScore as a calibrated high-confidence membership detector. In the extraction pipeline, the score is used only to rank the model's own outputs for corrector training. The relevant property is therefore ranking quality near the top of the candidate pool; calibration is not used by the extraction pipeline.

E Additional Extraction Results

E.1 Sensitivity to the Verification Threshold

Table 12 reports how the main comparison depends on the verification threshold of Appendix A.6. The same frozen per-output scores are re-thresholded at three values of τ and UMS is recomputed; filter shortlists are fixed identifier lists, so a higher τ re-scores the outputs a filter already returned rather than letting it choose again. At each target's own threshold the recomputation reproduces Table 2 exactly. D-Miner's advantage does not come from a permissive threshold: it is widest at $\tau = 0.90$, the strictest value both regimes can be compared at, where D-Miner leads on 11 of 12 targets. Beyond 0.90 the SFT targets retain almost no matches, so the comparison loses resolution rather than reversing, and we do not report it.

E.2 Qualitative Examples of Recovered Post-training Data

Table 13 shows matched pairs on the post-trained targets.

E.3 Data Utility Evaluation Details

Section 4.4 fine-tunes Dream-v0-Instruct-7B on question–answer pairs selected from the $1{,}024$ outputs D-Miner returns on Dream-Coder. Table 14 lists its training and evaluation settings.

	UMS, D-Miner / strongest baseline		
Model–source	$\tau = 0.80$	$\tau = 0.85$	$\tau = 0.90$
Dream-GitHub	195.3 / 162.0	19.0 / 12.0	1.0 / 0.0
Dream-ArXiv	388.7 / 295.0	139.7 / 56.0	1.7 / 0.0
Dream-HackerNews	304.3 / 334.0	78.3 / 62.0	1.7 / 0.3
Dream-PubMed	471.3 / 398.0	247.3 / 160.0	34.3 / 18.0
LLaDA-GitHub	219.0 / 96.0	44.3 / 13.0	8.0 / 1.0
LLaDA-ArXiv	430.7 / 304.0	249.0 / 185.0	3.0 / 2.0
LLaDA-HackerNews	429.3 / 190.0	85.7 / 10.0	0.7 / 0.0
LLaDA-PubMed	407.0 / 343.0	289.7 / 198.0	86.0 / 63.0
Dream-Coder	366.7 / 556.0	283.3 / 396.0	93.3 / 110.0
Dream-Bridge	356.7 / 955.0	353.3 / 697.3	328.3 / 304.7
ESPO-Code	613.0 / 294.0	509.3 / 263.0	114.3 / 113.0
ESPO-MATH	759.0 / 420.0	681.3 / 371.0	273.3 / 160.0
Targets where D-Miner leads	$9/12$	$10/12$	$\mathbf{11/12}$

Table 12: **UMS at three verification thresholds**, $R = 1{,}024$ and $S = 256$. Each cell is D-Miner against the strongest of Simple, MScore-Filter and KMeans-Filter on that target, means over the three seed blocks for the generation methods and one pooled filtered set for the filters. The paper's thresholds are 0.80 for domain-SFT targets and 0.90 for released models, so the four released models appear below their own threshold in the first two columns, where matches that a stricter reading would reject still count for every method.

F Ideal Correctors, Realizability, and Recovery Analysis

We prove Theorems 3.1 and 3.2, which establish the existence of an ideal transition corrector and its local realization through hidden-state shifts under the stated conditions. We also provide supplementary token-posterior analysis and identify the conditional objectives of FM and CE. The remaining analysis relates endpoint enrichment to the recovery metrics. These results separate the existence of an ideal target, its representability inside the frozen model, and what the selected training corpus identifies. Throughout, the wrapper tokens and verification rule are fixed. Table 15 maps each main-text statement to the result that supports it.

F.1 Proof of Theorem 3.1

Write $c(\mathbf{x}_0) \in \{0, 1\}$ for the recovery indicator of a clean sequence, so that the event $c = 1$ of Section 3.1 is $c(\mathbf{X}_0) = 1$, and $\varphi_t(\mathbf{x}_t) = \Pr_\theta(c = 1 \mid \mathbf{X}_t = \mathbf{x}_t)$ for the probability that the model's own sampling chain, started from the state $\mathbf{x}_t$, ends in a recovered record. For consecutive steps $s < t$ the corrector of Equation (5) is $C^\star_{s,t}(\mathbf{x}_s, \mathbf{x}_t) = \varphi_s(\mathbf{x}_s)/\varphi_t(\mathbf{x}_t)$, so the corrected transition of Equation (4) is

$$p_{\theta,C^\star}(\mathbf{x}_s \mid \mathbf{x}_t) = p_\theta(\mathbf{x}_s \mid \mathbf{x}_t)\, \frac{\varphi_s(\mathbf{x}_s)}{\varphi_t(\mathbf{x}_t)},$$

defined at every $\mathbf{x}_t$ with $\varphi_t(\mathbf{x}_t) > 0$.

It satisfies the normalization condition. Conditioning on the next state and using that the chain is Markov, $\sum_{\mathbf{x}_s} p_\theta(\mathbf{x}_s \mid \mathbf{x}_t)\, \varphi_s(\mathbf{x}_s) = \Pr_\theta(c = 1 \mid \mathbf{X}_t = \mathbf{x}_t) = \varphi_t(\mathbf{x}_t)$, so $\sum_{\mathbf{y}} p_\theta(\mathbf{y} \mid \mathbf{x}_t) C^\star_{s,t}(\mathbf{y}, \mathbf{x}_t) = 1$ and the corrected transition is a Markov kernel. The reweighting also leaves the support unchanged: a transition of zero base probability keeps zero probability, and a state with $\varphi_s(\mathbf{x}_s) = 0$, one from which recovery is impossible, is simply never entered.

The chain it induces is the conditioned one. Along a trajectory $\mathbf{x}_1 \to \cdots \to \mathbf{x}_0$ the ratios telescope, leaving $\varphi_0(\mathbf{x}_0)/\varphi_1(\mathbf{x}_1)$. At the end of the chain $\varphi_0(\mathbf{x}_0) = c(\mathbf{x}_0)$, since the clean sequence determines the event, and at the fully masked start $\varphi_1(\mathbf{x}_1) = \Pr_\theta(c = 1)$, which is positive by assumption. Summing over the trajectories that end in $\mathbf{x}_0$ therefore gives

$$p_{\theta,C^\star}(\mathbf{x}_0) = p_\theta(\mathbf{x}_0)\, \frac{c(\mathbf{x}_0)}{\Pr_\theta(c = 1)} = p_\theta(\mathbf{x}_0 \mid c = 1),$$

Dream-Bridge Reasoning similarity 0.948 corpus-level (`ianncity/KIMI-K2.5-1000000x`)

D-Miner output	*Matched reference record*
The user is asking for a Python function that determines **if a number is prime**. … **A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself.** I should handle edge cases (numbers less than 2 are not prime, 2 is prime) and check divisibility from 2 to the square root of the number, then `def is_prime(n)` returning `False` below 2, `True` at 2, and `False` for even n.	Can you provide a Python function to determine whether a given number is prime or not? **The user is asking for a Python function** to determine **if a number is prime** or not. … **A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself.** Edge cases: numbers less than 2 are not prime; 2 is prime; even numbers above 2 are not. Algorithm: if $n < 2$ return False; if $n = 2$ return True; if n even return False; check odd divisors to $\sqrt{n}$.

Dream-Coder Code similarity 0.943 record-level (`Dream-org/Dream-Coder-RL-17k`)

D-Miner output	*Matched reference record*
The diameter of a binary tree is defined as **the length of the longest path between any two nodes in the tree. This path may or may not pass through the root.** Here is a Python function `diameter_of_binary_tree` that calculates the diameter of a binary tree: `class TreeNode: __init__(self, value=0, left=None, right=None)`, then a recursive `height` that updates the running diameter with `left_height + right_height`.	Given a binary tree, you need to compute the length of the diameter of the tree. **The diameter of a binary tree is the length of the longest path between any two nodes in a tree. This path may or may not pass through the root.** You are given the `TreeNode` class … `class TreeNode: __init__(self, val=0, left=None, right=None)`. Write a function `diameterOfBinaryTree` that takes the root of a binary tree and returns the length of its diameter.

ESPO-Code Code similarity 0.920 corpus-level (`TIGER-Lab/AceCode-87K`)

D-Miner output	*Matched reference record*
You are given an array of `n` integers and a target integer `target`. Your task is to find **the indices of the two numbers that add up to** target and return those indices. **You may assume that each input would have** only **one solution and you may not use the same element** twice. Example: **`nums = [2, 7, 11, 15]`**, **`target = 9`**, so return `[0, 1]`.	Given an array of integers, return **the indices of the two numbers** such that **they add up to** a specific target. **You may assume that each input would have** exactly **one solution, and you may not use the same element** twice. `assert two_sum([2, 7, 11, 15], 9) == (0, 1); assert two_sum([3, 2, 4], 6) == (1, 2)`.

Table 13: **Matched pairs on the post-trained targets.** Bold marks wording the output and the record share. Outputs are excerpted; the records are longer than the $L = 256$ canvas the model writes into.

which is Equation (6). Nothing in the argument uses the form of c beyond its being a function of the clean sequence, and nothing uses the architecture of the denoiser: the corrector exists as a mathematical object, and the rest of Section 3 is about approximating it without access to φ_t. □

F.2 The Soft Tilt and its Token-Level Form

Theorem 3.1 conditions the model's own chain on recovery. A softer version tilts by $e^{\lambda c}$ instead; letting $\lambda \to \infty$ recovers the conditioning of the theorem, and the token-level form below shows why the corrector is conditioned on the decoding state (Equation (8)). Let $c(\mathbf{x}_0)$ be the recovery indicator, which in our evaluation is the frozen semantic event V_τ of Equation (15), evaluated after output serialization. The law we would like the model to sample from is its own, tilted toward that event:

$$p^\star_\lambda(\mathbf{x}_0) = \frac{p_\theta(\mathbf{x}_0)e^{\lambda c(\mathbf{x}_0)}}{Z_\lambda}, \qquad Z_\lambda = \mathbb{E}_{p_\theta} e^{\lambda c(X_0)}, \quad 0 < \lambda < \infty. \tag{16}$$

For $a = p_\theta(c = 1) \in (0, 1)$ the tilt raises the event to $ae^\lambda/(1 - a + ae^\lambda) > a$ without leaving the base model's support: it reweights text the model could already have written, and it promises more recovery in aggregate rather than coverage of every record. Both laws are corrupted by the same kernel q_t, and the correction that produces the tilt is available in closed form.

Setting	Value
Training	
Training data	109 question–answer pairs selected from the 1,024 outputs D-Miner returns on Dream-Coder
Initialization	Dream-v0-Instruct-7B
Objective	Dream's native masked-diffusion loss (shifted CART weighting, $p = 0.1$) on answer tokens; prompt tokens never masked; masks resampled at every pass
Parameters updated	All (full-parameter; no LoRA)
Epochs	3
Global batch size	8
Learning rate	2×10^{-6}
Seed	43
Hardware and precision	4 GPUs, FSDP full sharding; BF16 forward, FP32 master weights
Evaluation	
Metric	pass@1, one sample per task, sandboxed execution
Generation length / steps	512 tokens / 512 diffusion steps
Temperature	0.1
Top-p	0.9 (HumanEval, MBPP); 0.95 (BigCodeBench, LiveCodeBench)
Remasking	entropy (HumanEval, MBPP, BigCodeBench); maskgit_plus (LiveCodeBench)
HumanEval	164 problems, original tests
MBPP	378 problems, EvalPlus release, original tests
BigCodeBench	Complete split, full set, 1,140 tasks
LiveCodeBench	Code generation, release_v6, 1,055 problems

Table 14: Training and evaluation settings for the data utility study (Table 3).

Main-text statement	Where it is established
Ideal transition corrector (Theorem 3.1)	Appendix F.1
Why the corrector is conditioned on the decoding state (Equation (8)): soft tilt and state-dependent token correction	Appendices F.2, F.3, F.4
Local realization by hidden-state shifts (Theorem 3.2)	Appendix F.5
What the conditioned corrector can represent and how its effect is measured	Appendices F.6, F.7
Hidden-state-level supervision (Equation (12))	Appendix F.8
Token-level supervision and its gap to the ideal target (Equation (13))	Appendices F.9, F.10
Implemented corrector map and CE gradient path	Appendix F.11
Candidate ranking by MScore; budgets, UMS and AMS (Equation (14))	Appendices F.12, F.13

Table 15: Where each theoretical statement of the main text is proved or analyzed.

Proposition F.1 (Ideal posterior correction)**.** *At any corrupted state $\mathbf{x}_t$ of positive base probability, let r_t^0 and $r_t^\star$ be the clean-sequence Bayes posteriors under p_θ and $p_\lambda^\star$. The tilted posterior is a pointwise reweighting of the base one,*

$$r_t^\star(\mathbf{x}_0 \mid \mathbf{x}_t) = r_t^0(\mathbf{x}_0 \mid \mathbf{x}_t)\, e^{\ell_t^\star(\mathbf{x}_0;\mathbf{x}_t)}, \qquad \ell_t^\star(\mathbf{x}_0;\mathbf{x}_t) = \lambda c(\mathbf{x}_0) - \log M_t(\mathbf{x}_t), \tag{17}$$

where $M_t(\mathbf{x}_t) = \mathbb{E}_{r_t^0(\cdot\mid\mathbf{x}_t)} e^{\lambda c(X_0)}$ is the success weight averaged over the clean sequences the state still allows, so $\ell_t^\star$ is finite on that support.

The proof is Bayes' rule under a shared corruption kernel (Appendix F.3). A denoiser reports tokens rather than sequences; marginalizing gives, on positive token support,

$$d_{t,i}^\star(v;\mathbf{x}_t) = \log \frac{r_{t,i}^\star(v \mid \mathbf{x}_t)}{r_{t,i}^0(v \mid \mathbf{x}_t)} = \log M_{t,i}(v;\mathbf{x}_t) - \log M_t(\mathbf{x}_t), \tag{18}$$

where $M_{t,i}(v;\mathbf{x}_t) = \mathbb{E}_{r_t^0}[e^{\lambda c(X_0)} \mid X_{0,i} = v, X_t = \mathbf{x}_t]$ is the same weight restricted to completions that place v at site i. Because the weight is averaged over the completions the current context still

allows, one fixed notion of success asks for different token edits as the context fills in: a static direction in activation space cannot be the whole story (Appendix F.4).

F.3 Proof of Proposition F.1

Fix the record-independent wrapper throughout. Let p_θ be the declared reference law on the finite clean space $\mathcal{V}^L$, and let $c : \mathcal{V}^L \to \{0, 1\}$ be the fixed recovery predicate of Section 3.1. In particular, c may apply the frozen semantic verifier after output serialization; that choice defines a semantic event, not a claim of verified training membership. For finite $\lambda > 0$, the target $p^\star_\lambda$ in Equation (16) and p_θ have identical support, and $1 \le Z_\lambda \le e^\lambda$. Corrupt either clean law with the same declared q_t from Section 2.1. Write $p_{\theta,t}, p^\star_{\lambda,t}$ for the resulting corrupted-state marginals and $r^0_t, r^\star_t$ for their joint clean posteriors. These are Bayes distributions defined by the two clean laws and q_t; the neural predictions $p_{\theta,i}$ are separate objects.

Proposition F.1 (restated). *At any corrupted state $\mathbf{x}_t$ of positive base probability, the finite correction*

$$\ell^\star_t(\mathbf{x}_0; \mathbf{x}_t) = \lambda c(\mathbf{x}_0) - \log \mathbb{E}_{r^0_t(\cdot|\mathbf{x}_t)} e^{\lambda c(X_0)} \tag{19}$$

realizes

$$r^\star_t(\mathbf{x}_0 \mid \mathbf{x}_t) = r^0_t(\mathbf{x}_0 \mid \mathbf{x}_t) e^{\ell^\star_t(\mathbf{x}_0;\mathbf{x}_t)}. \tag{20}$$

Proof of Proposition F.1. For any $\mathbf{x}_t$ with $p_{\theta,t}(\mathbf{x}_t) > 0$, finite positive tilting also gives $p^\star_{\lambda,t}(\mathbf{x}_t) > 0$. Bayes' rule yields

$$r^\star_t(\mathbf{x}_0 \mid \mathbf{x}_t) = \frac{p_\theta(\mathbf{x}_0) q_t(\mathbf{x}_t \mid \mathbf{x}_0) e^{\lambda c(\mathbf{x}_0)}}{\sum_y p_\theta(y) q_t(\mathbf{x}_t \mid y) e^{\lambda c(y)}} = \frac{r^0_t(\mathbf{x}_0 \mid \mathbf{x}_t) e^{\lambda c(\mathbf{x}_0)}}{\mathbb{E}_{r^0_t(\cdot|\mathbf{x}_t)} e^{\lambda c(X_0)}}.$$

This proves Equation (20) for all $\mathbf{x}_0$, including zero-mass points. Taking logarithms on the common positive posterior support proves Equation (17). The normalizing term depends only on $(\mathbf{x}_t, t)$, so the unnormalized correction $\lambda c(\mathbf{x}_0)$ produces the same posterior as the normalized $\ell^\star_t(\mathbf{x}_0; \mathbf{x}_t)$.

Finally, if $a = \mathbb{E}_{p_\theta} c(X_0)$, then $Z_\lambda = 1 - a + ae^\lambda$ and

$$p^\star_\lambda(c = 1) - a = \frac{a(1-a)(e^\lambda - 1)}{1 - a + ae^\lambda}.$$

The gain is strictly positive exactly when $0 < a < 1$. At $a = 0$ or $a = 1$ it vanishes, and no finite positive tilt creates support absent from p_θ. □

Only a common corruption kernel at each t is needed for this result. The posterior identity specifies an ideal correction target; realizing that target in the neural denoiser and assessing the output law of its decoding algorithm are subsequent questions.

F.4 Why the Token Correction Depends on the Corrupted State

At a state with $p_{\theta,t}(\mathbf{x}_t) > 0$, fix a token with $r^0_{t,i}(v \mid \mathbf{x}_t) > 0$. Summing the joint reweighting identity over all clean sequences whose ith token equals v gives

$$r^\star_{t,i}(v \mid \mathbf{x}_t) = r^0_{t,i}(v \mid \mathbf{x}_t) \frac{\mathbb{E}_{r^0_t}[e^{\lambda c(X_0)} \mid X_{0,i} = v, X_t = \mathbf{x}_t]}{\mathbb{E}_{r^0_t}[e^{\lambda c(X_0)} \mid X_t = \mathbf{x}_t]}.$$

Taking logarithms gives the finite token correction

$$\begin{aligned} d^\star_{t,i}(v; \mathbf{x}_t) &= \log \frac{r^\star_{t,i}(v \mid \mathbf{x}_t)}{r^0_{t,i}(v \mid \mathbf{x}_t)} \\ &= \log \mathbb{E}_{r^0_t}[e^{\lambda c(X_0)} \mid X_{0,i} = v, X_t = \mathbf{x}_t] - \log \mathbb{E}_{r^0_t}[e^{\lambda c(X_0)} \mid X_t = \mathbf{x}_t]. \end{aligned} \tag{21}$$

Each conditional expectation lies in $[1, e^\lambda]$, so the finite normalized token correction $d^\star_{t,i}$ lies in $[-\lambda, \lambda]$ on this support. If the reference marginal is zero, the target marginal is also zero and their logarithmic ratio is undefined. At a visible position, the corruption fixes its observed value and the correction is zero on that singleton support.

Although c is fixed, its conditional expectation averages over clean completions compatible with the particular corrupted context and candidate token. Consequently, its token correction can vary with $\mathbf{x}_t$, i, and v; any further time dependence is determined by the corruption law. For independent Bernoulli masking, the likelihood of a fixed $\mathbf{x}_t$ is constant over compatible clean completions, so the explicit t factor cancels. The correction is not a sequence predicate evaluated on an isolated token. The denominator is constant across v and disappears when corrected logits are normalized.

The corresponding correction relative to a neural prediction obeys

$$\log \frac{r^\star_{t,i}(v \mid \mathbf{x}_t)}{p_{\theta,i}(v \mid \mathbf{x}_t, t)} = d^\star_{t,i}(v; \mathbf{x}_t) + \log \frac{r^0_{t,i}(v \mid \mathbf{x}_t)}{p_{\theta,i}(v \mid \mathbf{x}_t, t)}$$

where both reference and neural probabilities are positive. Replacing the Bayes reference by the neural denoiser therefore adds a consistency term; the posterior-ratio proof does not set that term to zero. A neural softmax with finite logits is positive on its admissible vocabulary, whereas an ideal posterior may contain zeros. For the token-level comparisons below, define $\tilde{r}^\varepsilon_{t,i} = (1-\varepsilon) r^\star_{t,i} + \varepsilon p_{\theta,i}$ for $0 < \varepsilon < 1$. This mixture is positive wherever $p_{\theta,i} > 0$ and lies within ε of $r^\star_{t,i}$ in total variation.

Joint dependence remains a separate quantity. Let I_t denote the masked positions of $\mathbf{x}_t$, as in Section 3.3, and suppose a factorized prediction copies visible positions. At a fixed state, with conditioning suppressed, its joint KL decomposes as

$$D_{\mathrm{KL}}\left(r^\star_{t,I_t} \middle\| \prod_{i \in I_t} p_{\theta,\Phi,i}\right) = \underbrace{D_{\mathrm{KL}}\left(r^\star_{t,I_t} \middle\| \prod_{i \in I_t} r^\star_{t,i}\right)}_{\text{conditional total correlation}} + \sum_{i \in I_t} D_{\mathrm{KL}}(r^\star_{t,i} \| p_{\theta,\Phi,i}). \tag{22}$$

Insert the product of true marginals inside the logarithm and marginalize each token term to verify the identity. Accurate token marginals can leave positive dependence error. This identity does not identify the transition kernel or output law of parallel decoding.

F.5 Proof of Theorem 3.2

Fix $\mathbf{x}_t$ and $s < t$ as in the theorem, and let $\mathcal{Y}$ be the common finite support of the sampling transitions throughout the intervention region. All vectors below are indexed by $\mathcal{Y}$, and logarithms are componentwise. The vector $\mathbf{h}$ contains only editable content-position activations; wrapper activations and model parameters are held fixed. For brevity, write

$$p^0(\mathbf{y}) = p_\theta(\mathbf{y} \mid \mathbf{x}_t), \qquad p^\Delta(\mathbf{y}) = p_\theta(\mathbf{y} \mid \mathbf{x}_t; \mathbf{h}_t + \Delta\mathbf{h}),$$

and suppress $(s, t, \mathbf{x}_t)$ in the corrector arguments.

Positivity and normalization. Theorem 3.1 gives $\sum_{\mathbf{y} \in \mathcal{Y}} p^0(\mathbf{y}) C^\star(\mathbf{y}) = 1$. The assumed strict positivity of $C^\star$ on $\mathcal{Y}$ therefore makes

$$p^\star(\mathbf{y}) = p^0(\mathbf{y}) C^\star(\mathbf{y})$$

a strictly positive probability distribution, with finite log probabilities and a finite log correction.

Proof of Theorem 3.2. Let $\mathbf{P} = \mathbf{I} - \mathbf{1}\mathbf{1}^\top/|\mathcal{Y}|$ be the centering projection, so that $F(\mathbf{h}) = \mathbf{P} \log p_\theta(\cdot \mid \mathbf{x}_t; \mathbf{h})$. Set

$$\mathbf{g} = \mathbf{P} \log C^\star, \qquad \mathbf{J} = DF(\mathbf{h}_t), \qquad \Delta\mathbf{h} = \mathbf{J}^\dagger \mathbf{g}.$$

The range assumption implies $\mathbf{J}\Delta\mathbf{h} = \mathbf{g}$. By the fundamental theorem of calculus,

$$\begin{aligned} \mathbf{r} &:= F(\mathbf{h}_t + \Delta\mathbf{h}) - F(\mathbf{h}_t) - \mathbf{g} \\ &= \int_0^1 \big[DF(\mathbf{h}_t + v\Delta\mathbf{h}) - \mathbf{J}\big] \Delta\mathbf{h} \, \mathrm{d}v. \end{aligned}$$

The Jacobian Lipschitz condition is measured in the induced Euclidean operator norm and gives

$$\|\mathbf{r}\|_2 \le \int_0^1 L_J v \|\Delta\mathbf{h}\|_2^2 \, \mathrm{d}v = \frac{L_J}{2} \|\Delta\mathbf{h}\|_2^2.$$

Moreover, $F(\mathbf{h}_t) + \mathbf{g} = \mathbf{P} \log p^\star$. Since adding a constant to every coordinate leaves softmax unchanged, the edited transition is

$$p^\Delta(\mathbf{y}) = \frac{p^\star(\mathbf{y}) e^{r(\mathbf{y})}}{\sum_{\mathbf{v} \in \mathcal{Y}} p^\star(\mathbf{v}) e^{r(\mathbf{v})}}.$$

Taking logarithms yields

$$\log p^\Delta(\mathbf{y}) = \log p^0(\mathbf{y}) + \log C^\star(\mathbf{y}) + e(\mathbf{y}), \qquad e(\mathbf{y}) = r(\mathbf{y}) - \log \mathbb{E}_{p^\star}[e^r].$$

The normalizing term lies between the smallest and largest coordinates of $\mathbf{r}$. Thus the error is uniformly bounded by

$$\|\mathbf{e}\|_\infty \le \max_{\mathbf{y}} r(\mathbf{y}) - \min_{\mathbf{y}} r(\mathbf{y}) \le 2\|\mathbf{r}\|_\infty \le L_J \|\Delta \mathbf{h}\|_2^2,$$

which proves Equation (9). □

Scope of the result. The map F describes the full sampling transition, including the decoding procedure, rather than only the denoiser's token marginals. The common-support and differentiability assumptions must hold along the entire intervention segment; changes in hard truncation or a discrete reveal decision may violate them. The range condition is a conditional reachability assumption about the frozen model at the chosen intervention layer. A poorly conditioned $\mathbf{J}$ can require a large shift, making the second-order bound uninformative or taking the shift outside the admissible region. The theorem therefore establishes local existence under its stated conditions, not that the trained C_Φ necessarily produces this shift. It also concerns one transition and does not by itself guarantee the final output distribution. Transitions with zero ideal-corrector weights are outside its scope, since their log corrections are not finite.

F.6 What the Conditioned Corrector Can Represent

Suppose an activation map $\Psi^\star(\mathbf{h}_t, \mathbf{x}_t)$ realizes or approximates the desired posterior target on a relevant domain, and write $\Psi_i^\star$ for its slot-i component. This is a conditional premise about the frozen downstream network, separate from posterior-corrector existence. The question here is narrower: whether the implemented parameterization can express $\Psi^\star$ at all. Write $\boldsymbol{\kappa}_i(\mathbf{x}_t)$ for the conditioning $\boldsymbol{\kappa}_{t,i} = (\rho_t, \mathbb{I}\{x_{t,i} = m\}, i)$ of Equation (8) at state $\mathbf{x}_t$. D-Miner writes the edited activation as

$$\mathbf{h}_{t,i} \longmapsto \mathbf{h}_{t,i} + C_\Phi\big(\mathbf{h}_{t,i}; \boldsymbol{\kappa}_i(\mathbf{x}_t), \boldsymbol{\xi}\big) \text{ on content slots}, \qquad \mathbf{h}_{t,i} \longmapsto \mathbf{h}_{t,i} \text{ on wrapper slots},$$

with no fixed direction added alongside and no separate strength parameter: on content slots the corrector output is added to the activation. Two restrictions follow.

Conditioning. The learned map reads only $(\mathbf{h}_{t,i}, \boldsymbol{\kappa}_i(\mathbf{x}_t))$ at slot i, never the full state or the other slots. A map depending on the full state therefore exists in this class exactly when the target is constant on the fibers of $(\mathbf{h}_t, \mathbf{x}_t) \mapsto (\mathbf{h}_{t,i}, \boldsymbol{\kappa}_i(\mathbf{x}_t))$.

Proposition F.2 (Representability under a restricted conditioning map)**.** *A map Γ with*

$$\Gamma(\mathbf{h}_{t,i}, \boldsymbol{\kappa}_i(\mathbf{x}_t)) = \Psi_i^\star(\mathbf{h}_t, \mathbf{x}_t)$$

on the relevant domain exists if and only if $\Psi_i^\star(\mathbf{h}_t, \mathbf{x}_t) = \Psi_i^\star(\mathbf{h}'_t, \mathbf{x}'_t)$ for every relevant pair with $\mathbf{h}_{t,i} = \mathbf{h}'_{t,i}$ and $\boldsymbol{\kappa}_i(\mathbf{x}_t) = \boldsymbol{\kappa}_i(\mathbf{x}'_t)$.

Proof. A function evaluated at the same argument must return the same value, which gives necessity. Conversely, constancy on each nonempty fiber makes $\Psi_i^\star(\mathbf{h}_t, \mathbf{x}_t)$ independent of the representative chosen from that fiber and so defines Γ on the relevant image. □

The conditioning carries the mask ratio, the per-slot mask indicator, and the absolute slot index, and $\mathbf{h}_{t,i}$ carries whatever the frozen prefix has already aggregated from the canvas. Two states sharing the slot activation and those three coordinates are indistinguishable to the corrector even when their partially filled canvases differ, and Equation (18) shows the ideal token correction does depend on the surrounding tokens. The fiber condition is thus a genuine restriction, not a formality, and the appropriate reading is that $\boldsymbol{\kappa}_{t,i}$ supplies the decoding stage and position while the context enters only through $\mathbf{h}_{t,i}$.

Support. The edit is restricted to content slots, so $\Psi^\star$ must leave the record-independent wrapper unchanged; corrections requiring any movement there are outside the class regardless of Φ.

Neither restriction says training realizes an expressible target. The randomness $\boldsymbol{\xi}$ is drawn afresh at every denoising step, so the edit is stochastic even at fixed Φ and fixed state, and the start noise and step count of the flow bound the displacement a single edit can produce, which can restrict representation further. The identity map is not in the class for positive start noise unless the flow returns the noised activation exactly; hidden-state-level supervision makes the initial edit approximately, not exactly, the identity.

F.7 Measuring the Induced Token Correction

Proposition F.3 (Induced token-posterior correction)**.** *At the same corrupted input, token position, and coupled auxiliary draw, let z^0 and z^Φ be the finite logits of the frozen model without and with the corrector, on a common admissible vocabulary before temperature scaling or truncation, and define $\widehat{\ell} = \mathrm{ctr}(z^\Phi - z^0)$, with ctr the centering of Theorem 3.2. Then*

$$p_{\theta,\Phi,i}(v \mid \mathbf{x}_t, t) = \frac{p_{\theta,i}(v \mid \mathbf{x}_t, t) e^{\widehat{\ell}_{t,i}(v)}}{\sum_w p_{\theta,i}(w \mid \mathbf{x}_t, t) e^{\widehat{\ell}_{t,i}(w)}}. \tag{23}$$

This identity holds at any intervention layer.

Proof. Write $\delta z = z^\Phi - z^0$ and $\overline{\delta z} = |\mathcal{V}|^{-1} \sum_v \delta z(v)$, restricting the vocabulary to its common admissible set when necessary. Then $p_\theta(v) e^{\delta z(v) - \overline{\delta z}} = e^{z^\Phi(v) - \overline{\delta z}} / \sum_w e^{z^0(w)}$. Normalizing over v cancels common factors and gives $\mathrm{softmax}(z^\Phi)(v)$. □

The identity is conditional on the intervention draw. Averaging logits before softmax generally does not give the mixture posterior. Different hard-truncation supports also fall outside a finite log-ratio identity. The measured correction describes the actual neural change; equality to the ideal correction requires additional target information. Nonlinear downstream blocks need not turn a hidden edit into a proportional logit change.

F.8 What Flow Matching Identifies

Fix the selected-candidate clean law ν, masking-time law Λ, and corruption q_t. Draw $X_0 \sim \nu$, $t \sim \Lambda$, and $X_t \sim q_t(\cdot \mid X_0)$. The declared activation training law is the law of block-input slot activations $Y = N(\mathbf{h}_{t,i})$ at the corrupted state (X_t, t), the random counterpart of $\mathbf{y}_{t,i}$ in Section 3.3, or its declared restriction to sampled masked positions. Any site sampling or weighting is part of that law. These are activations of corrupted token states; clean-input activations alone define a different training law. The conditioning supplied to the flow is $\boldsymbol{\kappa}_{t,i}$ of Equation (8).

For the linear path, draw independent Gaussian noise $\boldsymbol{\xi}$ of the same shape and $u \sim \mathrm{Unif}[0,1]$ independently. The objective is

$$Y_u = (1-u)Y + u\boldsymbol{\xi}, \qquad \mathcal{L}_{\mathrm{FM}}(\Phi) = \mathbb{E}\big[D_h^{-1} \|\mathbf{v}_\Phi(Y_u, u, \boldsymbol{\kappa}_{t,i}) - (\boldsymbol{\xi} - Y)\|_2^2\big]. \tag{24}$$

This is Equation (12) per slot, up to the constant factor D_h^{-1}, which does not change its minimizer. Set $W = \boldsymbol{\xi} - Y$. Data lie at $u = 0$ and noise at $u = 1$; continuous flow time u is distinct from masking time t. Assume finite second moments and fixed activation dimension D_h.

Proposition F.4 (Population flow-matching identification)**.** *Among measurable square-integrable velocity fields using $(Y_u, u, \boldsymbol{\kappa}_{t,i})$, the population optimum is $\mathbf{v}^\star = \mathbb{E}[W \mid Y_u, u, \boldsymbol{\kappa}_{t,i}]$, and*

$$\mathcal{L}_{\mathrm{FM}}(\Phi) = \mathbb{E}[D_h^{-1} \|W - \mathbf{v}^\star(Y_u, u, \boldsymbol{\kappa}_{t,i})\|_2^2] + \mathbb{E}[D_h^{-1} \|\mathbf{v}_\Phi(Y_u, u, \boldsymbol{\kappa}_{t,i}) - \mathbf{v}^\star(Y_u, u, \boldsymbol{\kappa}_{t,i})\|_2^2]. \tag{25}$$

Proof. Add and subtract the conditional mean inside the squared norm. The cross term has expectation zero because $\mathbb{E}[W - \mathbf{v}^\star \mid Y_u, u, \boldsymbol{\kappa}_{t,i}] = 0$. The remaining terms give the identity and characterize the optimum up to almost-sure equality. □

For another declared differentiable interpolation, the same proof applies with its path derivative replacing W, under the corresponding second-moment condition. The identified velocity belongs to the chosen activation law and available conditioning. It is not the sequence or token posterior ratio and does not itself identify the ideal token correction. Finite architecture, numerical integration, and optimization determine how closely the implemented flow realizes this population target. Flow matching, the hidden-state-level supervision, supplies the initialization; token-level supervision supplies the task-specific posterior target. Subsequent token-level updates can change the velocity field, so population FM optimality is not asserted for the final checkpoint.

F.9 CE Identifies the Posterior of Its Clean Training Law

For a fixed clean law ν, let r_t^ν be its Bayes posterior under q_t, and write $I_t = \{i : X_{t,i} = m\}$ for the masked positions, as in Section 3.3. For a stochastic corrector, let $\boldsymbol{\xi}$ be its auxiliary draw and assume $\boldsymbol{\xi} \perp X_0 \mid (X_t, t)$. All frozen pilot artifacts and model parameters are conditioned on throughout; $\boldsymbol{\xi}$ contains fresh corrector randomness. Thus r_t^ν remains the true label posterior when conditioning on $\boldsymbol{\xi}$ as well. For a clean law μ let m_μ denote the law of (X_0, t, X_t, i) with $X_0 \sim \mu$, $t \sim \Lambda$, $X_t \sim q_t(\cdot \mid X_0)$ conditioned on $I_t \neq \emptyset$, and i uniform on I_t; averaging a function of (X_t, t, i) under m_μ is the per-example mean over masked sites. We write $\mathbb{E}_{\mathsf{m}_\mu, \boldsymbol{\xi}}$ when the conditional auxiliary law of $\boldsymbol{\xi}$ given (X_t, t) is included. With $p_{\theta,i} = p_{\theta,i}(\cdot \mid X_t, t)$ and $p_{\theta,\Phi,i} = p_{\theta,\Phi,i}(\cdot \mid X_t, t, \boldsymbol{\xi})$, the population form of Equation (13) is

$$\mathcal{L}_\beta(\Phi) = \mathbb{E}_{\mathsf{m}_\nu, \boldsymbol{\xi}}\Big[-\log p_{\theta,\Phi,i}(X_{0,i}) + \beta\, D_{\mathrm{KL}}\big(p_{\theta,i} \,\|\, p_{\theta,\Phi,i}\big)\Big],$$

with both terms on the selected candidates ν, as in the main text. Dividing by $1+\beta$ gives the convex form used in this and the next subsection,

$$\mathcal{L}_\alpha(\Phi) = \frac{\mathcal{L}_\beta(\Phi)}{1+\beta}, \qquad \alpha = \frac{1}{1+\beta},$$

which weights the cross-entropy by α and the divergence by $1-\alpha$ and has the same minimizers. The deployed $\beta = 1$ is $\alpha = \frac{1}{2}$, and the ablation without the divergence term, $\beta = 0$, is $\alpha = 1$. Define the tempered target and its floor at each site by

$$p^\alpha_{t,i} = \alpha\, r^\nu_{t,i} + (1-\alpha)\, p_{\theta,i}, \qquad \mathbb{H}^\alpha_{t,i} = \mathbb{H}(p^\alpha_{t,i}) - (1-\alpha)\,\mathbb{H}(p_{\theta,i}),$$

both evaluated at (X_t, t). Then

$$\mathcal{L}_\alpha(\Phi) \;\geq\; \mathbb{E}_{\mathsf{m}_\nu}\Big[\mathbb{H}^\alpha_{t,i} + D_{\mathrm{KL}}\big(p^\alpha_{t,i} \,\big\|\, \bar{p}_{\theta,\Phi,i}\big)\Big], \tag{26}$$

with $\bar{p}_{\theta,\Phi,i} = \mathbb{E}_{\boldsymbol{\xi}}\, p_{\theta,\Phi,i}$ the randomization-averaged predictive, $\mathbb{H}^\alpha_{t,i}$ the part of the loss no corrector can remove, and equality when the auxiliary randomness is inert. Training therefore drives $\bar{p}_{\theta,\Phi,i}$ toward the tempered target $p^\alpha_{t,i}$.

Proposition F.5 (Joint, token, and regularized CE identities)**.** *For a joint predictor, population cross-entropy satisfies*

$$\begin{aligned}\mathbb{E}[-\log p_{\theta,\Phi}(X_0 \mid X_t, t, \boldsymbol{\xi})] &= \mathbb{H}_\nu(X_0 \mid X_t, t) \\ &\quad + \mathbb{E}_{t, X_t, \boldsymbol{\xi}} D_{\mathrm{KL}}\big(r^\nu_t(\cdot \mid X_t) \| p_{\theta,\Phi}(\cdot \mid X_t, t, \boldsymbol{\xi})\big).\end{aligned} \tag{27}$$

For the masked-token term,

$$\mathbb{E}_{\mathsf{m}_\nu, \boldsymbol{\xi}}\big[-\log p_{\theta,\Phi,i}(X_{0,i} \mid X_t, t, \boldsymbol{\xi})\big] = \mathbb{H}^{\mathrm{tok}}_\nu + \mathbb{E}_{\mathsf{m}_\nu, \boldsymbol{\xi}} D_{\mathrm{KL}}\big(r^\nu_{t,i}(\cdot \mid X_t) \| p_{\theta,\Phi,i}(\cdot \mid X_t, t, \boldsymbol{\xi})\big), \tag{28}$$

where $\mathbb{H}^{\mathrm{tok}}_\nu = \mathbb{E}_{\mathsf{m}_\nu} \mathbb{H}(r^\nu_{t,i})$ *is constant in* Φ*. For the full objective,*

$$\mathcal{L}_\alpha(\Phi) = \mathbb{E}_{\mathsf{m}_\nu}\big[\mathbb{H}^\alpha_{t,i}\big] + \mathbb{E}_{\mathsf{m}_\nu, \boldsymbol{\xi}}\big[D_{\mathrm{KL}}\big(p^\alpha_{t,i} \,\|\, p_{\theta,\Phi,i}(\cdot \mid X_t, t, \boldsymbol{\xi})\big)\big], \tag{29}$$

and the first term is constant in Φ*.*

Proof. Condition on $(t, X_t, \boldsymbol{\xi})$ and add and subtract the log of the true conditional probability in the negative log-likelihood; conditional independence makes that true law r_t^ν. This gives entropy plus KL for the joint predictor and, conditioning additionally on the uniformly drawn site i, the token identity. For the full objective, fix a site and abbreviate $r = r^\nu_{t,i}$, $q = p_{\theta,i}$, and $p = p_{\theta,\Phi,i}$. Then $\alpha\,\mathrm{CE}(r,p) + (1-\alpha)\, D_{\mathrm{KL}}(q\|p) = -\sum_v p^\alpha(v) \log p(v) - (1-\alpha)\,\mathbb{H}(q) = \mathbb{H}(p^\alpha) + D_{\mathrm{KL}}(p^\alpha\|p) - (1-\alpha)\,\mathbb{H}(q)$, since $\alpha r + (1-\alpha) q = p^\alpha$. Integrating against m_ν and the auxiliary law proves the identity. All three hold with extended KL values, while finite CE requires prediction support covering the true posterior support. By concavity of entropy, $\mathbb{H}^\alpha_{t,i} \geq \alpha\,\mathbb{H}(r^\nu_{t,i}) \geq 0$. □

Expected per-draw CE is the quantity above; it need not equal the negative log-likelihood of the mixture prediction $\mathbb{E}_{\boldsymbol{\xi}} p_{\theta,\Phi}$, which is why Equation (26) is stated as an inequality: both the cross-entropy and the divergence are convex in the prediction, so averaging $\boldsymbol{\xi}$ inside can only lower them. When ν is uniform on frozen selected candidate IDs, with multiplicities retained, its Bayes posterior r_t^ν, tempered toward the frozen model's prediction by $1-\alpha$, is the identified target. Selection by MScore enters through this clean law even though all CE terms have equal weight, and the divergence term enters only through $p_{\theta,i}$, the frozen model's own prediction on the same candidates. Neither FM nor CE replaces p^α by the posterior of $p_\lambda^\star$.

F.10 Separating Candidate Fit from the Ideal Target

The identities above compare predictions on the states and sites of the training law. The site measure m_ν weights every corrupted state equally and, within a state, every masked site equally; this is the weighting induced by the per-example mean in Equation (13), whereas a summed reduction would weight states by their mask counts. Assume $r_t^\star$ is defined m_ν-almost everywhere. The sufficient condition $\nu \ll p_\theta$ ensures this under the common corruption; coverage only on the weighted corrupted states is enough. The tempered target is $p_{t,i}^\alpha = \alpha r_{t,i}^\nu + (1-\alpha) p_{\theta,i}$ of Appendix F.9.

Define the clean-law mismatch, the frozen model's mismatch, and the excess over the floor by

$$\omega_\nu = \mathbb{E}_{\mathsf{m}_\nu} \mathrm{TV}(r_{t,i}^\nu, r_{t,i}^\star), \qquad \omega_\theta = \mathbb{E}_{\mathsf{m}_\nu} \mathrm{TV}(p_{\theta,i}, r_{t,i}^\star),$$
$$\mathcal{K}^\alpha(\Phi) = \mathcal{L}_\alpha(\Phi) - \mathbb{E}_{\mathsf{m}_\nu} \mathbb{H}_{t,i}^\alpha = \mathbb{E}_{\mathsf{m}_\nu, \boldsymbol{\xi}} D_{\mathrm{KL}}(p_{t,i}^\alpha \| p_{\theta,\Phi,i}).$$

All conditionals are evaluated at the same corrupted state, and $p_{\theta,\Phi}$ additionally uses the indicated auxiliary draw. Since $p_{t,i}^\alpha - r_{t,i}^\star = \alpha(r_{t,i}^\nu - r_{t,i}^\star) + (1-\alpha)(p_{\theta,i} - r_{t,i}^\star)$, the triangle, Pinsker, and Jensen inequalities give

$$\begin{aligned}\mathbb{E}_{\mathsf{m}_\nu, \boldsymbol{\xi}} \mathrm{TV}(p_{\theta,\Phi,i}, r_{t,i}^\star) &\le \alpha\, \omega_\nu + (1-\alpha)\, \omega_\theta + \mathbb{E}_{\mathsf{m}_\nu, \boldsymbol{\xi}} \sqrt{\tfrac{1}{2} D_{\mathrm{KL}}(p_{t,i}^\alpha \| p_{\theta,\Phi,i})} \\ &\le \alpha\, \omega_\nu + (1-\alpha)\, \omega_\theta + \sqrt{\mathcal{K}^\alpha(\Phi)/2}. \end{aligned} \tag{30}$$

The same bound holds for the mixture prediction, with expectation over m_ν, by convexity of total variation. For comparison with the smoothed target, one may replace the target marginals throughout by $\tilde{r}_{t,i}^\varepsilon$, or add at most ε to the displayed bound. At $\alpha = 1$ the first term is the candidate law's mismatch alone; as α decreases it moves toward the frozen model's own distance from the ideal target, which is the trade-off the regularizer controls.

The excess $\mathcal{K}^\alpha$ is above the floor, not an empirical optimum. To separate representation and fitting, let $\mathcal{F}$ be the chosen corrector class. If a learned $\widehat{\Phi} \in \mathcal{F}$ satisfies an empirical optimization gap η_{opt}, and the uniform difference between empirical and population objectives over this class is at most $\varepsilon_{\mathrm{gen}}$, then

$$\mathcal{K}^\alpha(\widehat{\Phi}) \le \inf_{\Phi \in \mathcal{F}} \mathcal{L}_\alpha(\Phi) - \mathbb{E}_{\mathsf{m}_\nu} \mathbb{H}_{t,i}^\alpha + 2\varepsilon_{\mathrm{gen}} + \eta_{\mathrm{opt}}. \tag{31}$$

Compare population and empirical objectives at $\widehat{\Phi}$, use its empirical optimization gap, and compare the empirical infimum with the population infimum to obtain this inequality. The first two terms are the class's approximation error; the remaining terms are statistical comparison and optimization error. This is conditional on the uniform bound, not a finite-sample guarantee for unbounded log loss or adaptive data without further assumptions.

Even perfect fit leaves $\alpha\omega_\nu + (1-\alpha)\omega_\theta$. Both bounds use m_ν, not the visitation law of native decoding, and neither controls the dependence term in Equation (22). They explain the trained approximation and remaining target gap; recovery probability and distinct-record coverage of the executed sampler require output evaluation.

F.11 Activation Correction, Flow Time, and the CE Gradient Path

The corrector acts at the input of decoder block 16. At each denoising state the frozen prefix produces the block input, the corrector replaces it at the content slots, and the frozen suffix produces logits for the sampler. No fixed direction and no separate strength parameter are applied alongside the edit. We adopt activation noising and denoising from Luo et al. (2026a); hidden-state-level supervision first fits the activation flow, and token-level supervision then adapts the correction with the objective of

Equation (13). This construction parameterizes an intervention on the network; it does not identify the network posterior with the oracle Bayes posterior.

Training states and the two noise variables. Corrector training uses the $N_{\text{train}} = 32$ highest-scoring self-generated candidates selected in Section 3.3, with equal weights; the divergence term of Equation (13) is evaluated on the same candidates. Reference records are excluded from candidate selection and checkpoint selection; checkpoints are selected by the validation value of the same objective, Equation (13), on 32 held-out self-generated candidates. The training count, β, and stopping rule are fixed before training. For a selected candidate X_0, the DLM corruption produces $X_t \sim q_t(\cdot \mid X_0)$; the training states include fully masked inputs, 95% masking, and other masking rates. Let $\mathbf{h}_t$ be the frozen block-16 input at the corrupted state (X_t, t), with slot activations $\mathbf{h}_{t,i}$. The flow is trained on these activations of corrupted token inputs. Thus a noise-free *activation-flow endpoint* need not arise from a clean token input.

To distinguish the two processes, use $u \in [0, 1]$ for continuous activation noise: $u = 0$ denotes the activation endpoint and $u = 1$ denotes Gaussian noise. The DLM variable t and the flow variable u have different roles and need not share a schedule. For an explicit analytical convention, identify the activation coordinates being modeled with $\mathbb{R}^{D_h}$ and write $Y = N(\mathbf{h}_{t,i})$ for a content slot i, the random counterpart of $\mathbf{y}_{t,i}$, where N is the fixed activation normalization of Section 3.3, an invertible affine coordinate map; we use per-dimension standardization by the mean and standard deviation of the cached training activations. This notation does not prescribe the selected positions. With independent $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $u \sim \text{Unif}[0, 1]$, the linear flow-matching convention is

$$Y_u = (1-u)Y + u\boldsymbol{\xi}, \qquad \mathcal{L}_{\text{FM}}(\Phi) = \mathbb{E}\left[\frac{1}{D_h}\left\|\mathbf{v}_\Phi(Y_u, u, \boldsymbol{\kappa}_{t,i}) - (\boldsymbol{\xi} - Y)\right\|_2^2\right], \tag{32}$$

the objective of Equation (24), and of Equation (12) up to the factor D_h^{-1}, with the conditioning $\boldsymbol{\kappa}_{t,i}$ of Equation (8). The expectation includes the candidate and token-corruption distributions.

An explicit activation map. The reported correction uses starting activation noise $u_\star = 0.25$ and $K_{\text{flow}} = 4$ flow steps. On the grid $u_j = u_\star(1 - j/K_{\text{flow}})$ the Euler map is

$$\begin{aligned} Y^{(0)} &= (1-u_\star)N(\mathbf{h}_{t,i}) + u_\star\boldsymbol{\xi}, \\ Y^{(j+1)} &= Y^{(j)} + (u_{j+1} - u_j)\,\mathbf{v}_\Phi(Y^{(j)}, u_j, \boldsymbol{\kappa}_{t,i}), \\ C_\Phi(\mathbf{h}_{t,i}; \boldsymbol{\kappa}_{t,i}, \boldsymbol{\xi}) &= N^{-1}\big(Y^{(K_{\text{flow}})}\big) - \mathbf{h}_{t,i}, \end{aligned} \tag{33}$$

with $j = 0, \ldots, K_{\text{flow}} - 1$ in the update equation. One initial noise tensor is used throughout the flow, and $u_{j+1} - u_j < 0$ matches the velocity target $\boldsymbol{\xi} - Y$. An implementation with the opposite flow-time orientation must also reverse that target. The map uses one velocity evaluation per step. For $u_\star = 0$ this map returns zero displacement, giving an identity intervention. A zero velocity at positive $u_\star$ instead leaves the initially noised activation unchanged, so it does not produce an identity corrector; hidden-state-level supervision, fitted to the same activations, brings the initial edit close to the identity but not exactly onto it.

Token-level computation graph. At the intervention site, write $\mathbf{h}_{t,i}^{\text{full}} = \mathbf{h}_{t,i} + C_\Phi(\mathbf{h}_{t,i}; \boldsymbol{\kappa}_{t,i}, \boldsymbol{\xi}) = \mathbf{h}_{t,i} + \Delta\mathbf{h}_{\Phi,t,i}$ on content slots and $\mathbf{h}_{t,i}^{\text{full}} = \mathbf{h}_{t,i}$ elsewhere. For token-level supervision, replay the frozen prefix on corrupted candidates, apply the correction, and evaluate the frozen downstream blocks and output head. Only the flow parameters Φ are optimized. Freezing target *parameters* leaves the derivative through the suffix available: the CE graph passes through the corrected activation and the differentiable flow computation, so every flow parameter receives token-level gradients from the first update. Detaching the corrected activation or disabling differentiation through the suffix would define a different training procedure. The coordinate map N remains frozen. Choosing $u_\star = 0$ would remove the flow parameters from the CE graph, so the start noise is positive.

Masked-token loss and output alignment. Let I_t denote the masked target positions, as in Section 3.3. For a batch of n_b candidates, the reduction implementing the first term of Equation (13) is $n_b^{-1}\sum_b |I_t^{(b)}|^{-1}\sum_{i \in I_t^{(b)}} - \log p_{\theta,\Phi,i}(X_{0,i}^{(b)} \mid X_t^{(b)}, t_b)$, the per-example mean over masked tokens; a draw with no masked token is excluded rather than counted as zero. The mask count k of each

draw is sampled with extra mass on fully masked states (0.2) and on 95% masking (0.3) and otherwise uniformly, and each example is multiplied by the importance weight $1/(L\,p(k))$, so that the expectation equals the uniform objective of Equation (13). The divergence term uses the same corruption and reduction on the same candidates, with the frozen model's prediction, computed with the corrector disabled and detached, as its target; both terms exclude the mask token from the admissible vocabulary.

The loss uses predictions aligned with the clean target indices. Output alignment and the hidden positions selected for intervention are separate choices: a block-input intervention can affect other positions through downstream attention. Here block 16 denotes its *input*, not the block output or the final-normalization output. The finite-logit identity in Proposition F.3 concerns aligned outputs; it does not require token-wise correction or imply that visible-token logits remain unchanged. Recovery is assessed after the sampler produces complete texts.

F.12 Score Enrichment and the Actual Intervention Gap

Let c denote the recovery indicator of Section 3.1, and write $a = \mathbb{E}_{p_\theta}[c(X_0)]$. The ideal law in Equation (16) satisfies

$$\Pr_{p^\star_\lambda}[c(X_0) = 1] = \frac{ae^\lambda}{1 - a + ae^\lambda}. \tag{34}$$

Indeed, its numerator before normalization is ae^λ, and its normalizer is $1 - a + ae^\lambda$. This probability exceeds a when $0 < a < 1$ and $\lambda > 0$; the boundary values $a \in \{0, 1\}$ remain unchanged. For $0 < a < 1$, the tilted law converges to $p_\theta(\cdot \mid c = 1)$ as $\lambda \to \infty$. The construction preserves the base law's relative probabilities within the accepted event and cannot create support outside the base law. It therefore specifies recovery enrichment rather than the uniform empirical law of the reference collection.

Observable scores under a declared law. To analyze a candidate score, let $\mathbb{P}_0$ be a declared joint law of an output X and its scoring randomness ζ, and let $\mathcal{S}(X, \zeta)$ be the real-valued score. For deterministic scoring, ζ can be omitted. Set $c(\perp) = 0$ if the output space includes a failure sentinel. Here $a = \mathbb{E}_{\mathbb{P}_0}[c(X)]$ refers to this declared law. For finite $0 < \mathbb{E}_{\mathbb{P}_0}[e^{\lambda\mathcal{S}(X,\zeta)}] < \infty$, define

$$\varrho_{\lambda,\mathcal{S}} = \frac{\mathbb{E}_{\mathbb{P}_0}[c(X)e^{\lambda\mathcal{S}(X,\zeta)}]}{\mathbb{E}_{\mathbb{P}_0}[e^{\lambda\mathcal{S}(X,\zeta)}]}. \tag{35}$$

Subtracting a gives the exact covariance identity

$$\begin{aligned} \varrho_{\lambda,\mathcal{S}} - a &= \frac{\mathrm{Cov}_{\mathbb{P}_0}(c(X), e^{\lambda\mathcal{S}(X,\zeta)})}{\mathbb{E}_{\mathbb{P}_0}[e^{\lambda\mathcal{S}(X,\zeta)}]}, \\ \left.\frac{\mathrm{d}\varrho_{\lambda,\mathcal{S}}}{\mathrm{d}\lambda}\right|_{\lambda=0} &= \mathrm{Cov}_{\mathbb{P}_0}(c(X), \mathcal{S}(X,\zeta)). \end{aligned} \tag{36}$$

The derivative follows by the quotient rule when, for example, $\mathbb{E}_{\mathbb{P}_0}[e^{\varepsilon|\mathcal{S}(X,\zeta)|}] < \infty$ for some $\varepsilon > 0$. Positive score covariance gives a local enrichment direction; a finite tilt depends on covariance with $e^{\lambda\mathcal{S}}$ itself.

The choice of $\mathbb{P}_0$ matters. For the randomized MScore in Equation (11), an accepted-candidate analysis may use $\mathbb{P}_0 = \mathrm{Law}(X, \zeta \mid \mathbf{1}_{\mathrm{valid}}(X, \zeta) = 1)$, where X is drawn from the one-attempt generator law, ζ from the specified scoring-mask law, and the validity event has positive probability. Interpreting retained pilot observations under this law additionally requires that retention and stopping preserve that conditional distribution. This validity-conditioned law generally differs from the law of all generation attempts. Marginalizing the scoring randomness yields the text weight $\mathbb{E}[e^{\lambda\mathcal{S}(X,\zeta)} \mid X, \mathbf{1}_{\mathrm{valid}} = 1]$, rather than the exponential of the conditional mean score. The corrector instead trains with equal weights on the highest-scoring generations, a finite ranked selection rather than exponential score weighting; the score tilt above serves only as an analytical comparison law.

A bound for the practical intervention. Let $\mathbb{P}_{\lambda,\mathcal{S}}(\mathrm{d}(x, \zeta)) \propto e^{\lambda\mathcal{S}(x,\zeta)}\mathbb{P}_0(\mathrm{d}(x, \zeta))$, and let $P_{\lambda,\mathcal{S}}$ and P_0 be its output marginal and the base output marginal, respectively. All output laws use the same canonical verification domain, including failures. For the actual evaluated output law $p_{\theta,\Phi}$,

define $\delta = \mathrm{TV}(p_{\theta,\Phi}, P_{\lambda,\mathcal{S}})$. Total variation bounds the discrepancy of the event $\{c = 1\}$, so

$$P_D(c = 1) - P_0(c = 1) \geq \frac{\mathrm{Cov}_{\mathbb{P}_0}(c, e^{\lambda \mathcal{S}})}{\mathbb{E}_{\mathbb{P}_0} e^{\lambda \mathcal{S}}} - \mathrm{TV}(P_D, P_{\lambda,\mathcal{S}}). \tag{37}$$

This follows by substituting Equation (36) into $P_D(c = 1) - P_0(c = 1) \geq \varrho_{\lambda,\mathcal{S}} - a - \delta$. When the score is randomized, the covariance is taken under its declared joint law. If P_0 is validity conditioned but the reported baseline P_{base} includes all attempts, the bound for that baseline must additionally include $P_0(c = 1) - P_{\mathrm{base}}(c = 1)$. The discrepancy δ measures an output-law approximation error; the measured logit identity and candidate CE alone do not bound it.

F.13 Budgeted Unique Recovery

Fix a finite, nonempty reference collection, its valid embeddings, and a deterministic nearest-neighbor tie rule. Let $\mathcal{X}_E$ be the canonical serialized-text domain of the frozen verifier. An evaluated attempt is an element of $\mathcal{X}_E$ when generation and verification succeed and is $\perp$ after a generation, serialization, encoding, or search failure. For $x \in \mathcal{X}_E$, define

$$d^\star_E(x) = \arg\max_{d \in \mathcal{D}_{\mathrm{ver}}} \cos(E(x), E(d)), \tag{38}$$

$$I^\tau_d(x) = \mathbf{1}\{d^\star_E(x) = d,\ V_\tau(x; \mathcal{D}_{\mathrm{ver}}) = 1\}, \tag{39}$$

$$p^\tau_{d,P} = \mathbb{E}_{X \sim P}[I^\tau_d(X)], \tag{40}$$

where P is a fixed law on $\mathcal{X}_E \cup \{\perp\}$. Set $I^\tau_d(\perp) = V_\tau(\perp; \mathcal{D}_{\mathrm{ver}}) = 0$. The identity $\sum_d I^\tau_d(x) = V_\tau(x; \mathcal{D}_{\mathrm{ver}})$ then holds on the entire evaluated domain. It describes the frozen semantic matching rule, not independently verified training membership.

For B independent attempts $X^{(1)}, \ldots, X^{(B)} \sim P$, define

$$\mathrm{UMS}_B = \sum_{d \in \mathcal{D}_{\mathrm{ver}}} \mathbf{1}\left\{ \sum_{b=1}^{B} I^\tau_d(X^{(b)}) > 0 \right\}, \tag{41}$$

$$\mathrm{AMS}_B = \sum_{b=1}^{B} V_\tau(X^{(b)}; \mathcal{D}_{\mathrm{ver}}). \tag{42}$$

UMS_B counts the distinct reference records reached and AMS_B the matching attempts, both at threshold τ. Duplicate and invalid attempts stay inside B, as Section 4.1 requires of the generation budget. For Simple and D-Miner, which return all $B = R$ attempts, these are the UMS and AMS of Equation (14); a filter returns R of $3R$ pooled attempts and falls under the dependent law noted at the end of this subsection.

Proposition F.6 (Expected recovery at a fixed attempt budget)**.** *For an integer $B \geq 1$ and independent draws from a fixed evaluated law P,*

$$\mathbb{E}_P[\mathrm{UMS}_B] = \sum_{d \in \mathcal{D}_{\mathrm{ver}}} \left[1 - (1 - p^\tau_{d,P})^B\right], \tag{43}$$

$$\frac{\mathbb{E}_P[\mathrm{AMS}_B]}{B} = \sum_{d \in \mathcal{D}_{\mathrm{ver}}} p^\tau_{d,P} = \Pr_P[V_\tau(X; \mathcal{D}_{\mathrm{ver}}) = 1]. \tag{44}$$

Proof. Record d is recovered unless all B attempts miss its assignment event, which has probability $(1 - p^\tau_{d,P})^B$. Sum the complementary indicators over records to obtain the first identity. The second follows by linearity of expectation and the disjointness of the nearest-record events. □

Enrichment and concentration. Expected AMS depends only on total hit probability, whereas expected UMS depends on its allocation across reference identities. For $B > 1$, the function $f_B(p) = 1 - (1 - p)^B$ is increasing and concave. Consequently, concentrating a fixed total hit probability on fewer records can reduce expected UMS. In particular, if p majorizes p' while both have the same sum, then $\sum_d f_B(p_d) \leq \sum_d f_B(p'_d)$. If P is tilted by the any-match event $c = V_\tau(\cdot; \mathcal{D}_{\mathrm{ver}})$, write $a = \Pr_P[c = 1]$. Every successful record probability is then multiplied by $e^\lambda/(1-a+ae^\lambda) \geq 1$;

therefore its expected UMS also cannot decrease. This implication uses that specific tilt, and does not extend to arbitrary score-based interventions.

For a deterministic endpoint score, write $P_\lambda(x) \propto e^{\lambda \mathcal{S}(x)} P(x)$ and $p_d(\lambda) = \mathbb{E}_{P_\lambda}[I_d^\tau(X)]$. Under the exponential-moment condition used above, differentiation gives

$$\frac{\mathrm{d}}{\mathrm{d}\lambda} \mathbb{E}_{P_\lambda}[\mathrm{UMS}_B]\Big|_{\lambda=0} = B \sum_{d \in \mathcal{D}_{\mathrm{ver}}} (1 - p_d(0))^{B-1} \operatorname{Cov}_P(I_d^\tau(X), \mathcal{S}(X)). \tag{45}$$

This follows by applying Equation (36) to each record event and differentiating Equation (43). The factor $(1 - p_d(0))^{B-1}$ gives more weight to improving the probability of less frequently recovered records. Positive covariance with the aggregate any-match event alone does not determine the sign of this derivative.

Frozen artifacts. Let $\mathcal{A}$ collect the frozen corrector artifacts of a run, and assume its evaluated attempts are i.i.d. conditional on $\mathcal{A}$, with assignment probabilities $p_d^\tau(\mathcal{A})$. If artifacts vary across runs, let $\mathcal{E}$ be the event that a run completes the artifact construction and freezes its sampler. Taking expectations conditional on $\mathcal{E}$ gives

$$\mathbb{E}[\mathrm{UMS}_B \mid \mathcal{E}] = \mathbb{E}_{\mathcal{A}}\left[\sum_{d \in \mathcal{D}_{\mathrm{ver}}} \left\{ 1 - (1 - p_d^\tau(\mathcal{A}))^B \right\} \middle| \mathcal{E} \right]. \tag{46}$$

For $B > 1$, replacing the random probabilities in this formula by their marginal means generally gives an upper bound by Jensen's inequality, rather than an equality. The independence premise concerns attempts conditional on the artifacts, including their failure mechanism; shared batch failures or selection of a subset from a fixed candidate pool require the corresponding dependent law instead.

G Extended Related Work

This appendix expands the three threads summarized in Section 5.

Training data extraction. Memorization in language models is surfaced either by sampling at scale and filtering the candidates (Carlini et al., 2021; Nasr et al., 2025) or by generating from a known prefix or a target-derived context and checking the continuation (Carlini et al., 2019; Yu et al., 2023), with per-record success made precise by (n, p)-discoverability (Hayes et al., 2025) and amplified by learned prompts (Wang et al., 2024). Target-free variants extract alignment data from minimal prompts and evaluate semantic rather than verbatim recovery (Barbero et al., 2025), approximate training data from weight changes (Morris et al., 2025), or synthesize instruction data from empty templates (Xu et al., 2025). For diffusion models, extraction and replication have been studied mostly for images (Carlini et al., 2023; Somepalli et al., 2023; Chen et al., 2026c; Wu et al., 2025); for diffusion language models, existing work reconstructs records from visible subsets of their tokens (Luo et al., 2026b; Wang & Asokan, 2026). We keep the semantic recovery unit of Barbero et al. (2025) but supply no target tokens at all, and count distinct records at declared budgets.

Membership signals. Membership inference compares a model's loss or its calibrated variants between members and non-members (Shokri et al., 2017; Yeom et al., 2018; Mattern et al., 2023; Shi et al., 2024; Zhang et al., 2025; Xie et al., 2024; Wang et al., 2025), with mixed success on pretrained LLMs (Duan et al., 2024; Das et al., 2025) and stronger signals after fine-tuning (Fu et al., 2024a; Chen et al., 2026a). Diffusion models admit attacks through their denoising errors (Duan et al., 2023; Kong et al., 2024), through degrade-and-restore reconstruction (Fu et al., 2024b), and, in the black-box setting, through the similarity of generations to the query (Pang & Wang, 2025), and fine-tuned diffusion language models through masked-token reconstruction and its token-level asymmetry (Chen et al., 2026b; Zhai et al., 2026). Our score belongs to this last family, but we use it to rank the model's own generations for corrector training rather than to classify supplied text.

Steering diffusion language models. Activation steering adds learned or contrastive directions to hidden states of autoregressive models (Turner et al., 2023; Rimsky et al., 2024), and has been carried to masked diffusion models for safety control (Shnaidman et al., 2025; Lee & Han, 2026); style and safety attributes can also be steered by biasing token logits at each denoising step (An & Han, 2026). Steering can also act on the denoising posterior (Rector-Brooks et al., 2025), through

discrete-guidance theory (Wan et al., 2026), joint sampling (Bansal & Sanghavi, 2026), or constrained decoding (Mündler et al., 2026). D-Miner instead learns a state-conditioned corrector, parameterized by a generative latent prior over activations (Luo et al., 2026a), and aims it at training records rather than at a specified attribute; the guidance and sampling theory above informs our analysis of what such a corrector can realize, not a new extraction mechanism.

H LIMITATIONS

This work studies post-training data extraction from DLMs under masked inputs and validates whether generated outputs can be matched to post-training data. This study has two main limitations. First, we do not systematically evaluate mitigation strategies against extraction attack. Since effective defenses for DLM leakage remain unclear, we leave the design and evaluation of such defenses to future work. Second, the recovery study of Section 4.4 measures downstream utility for a single target, Dream-Coder, by fine-tuning on extracted examples and evaluating coding benchmarks. It does not reproduce the original post-training recipe, and we do not repeat the study for the other targets because of the cost of additional post-training runs, so the utility of the recovered data is established only for that case.

I ETHICAL AND SECURITY CONSIDERATIONS

This work studies post-training data extraction from diffusion language models as a privacy-auditing problem. Its goal is to surface leakage risks in masked denoising and guide safer post-training and release practices. The method could be misused to probe open-weight or white-box models for memorized post-training content, especially when logits or hidden states are exposed. We therefore restrict experiments to open-weight or locally constructed targets and public or model-card-associated corpora, and use corpus-level verification when a checkpoint-specific manifest is unavailable. We recommend using the method only for authorized audits and defensive research, releasing code and evaluation artifacts in a controlled manner, and avoiding disclosure of private data or unredacted sensitive generations. Model owners should combine pre-release memorization audits with deduplication, rate limiting, monitoring for mask-only probing, and safeguards against abnormal intervention behavior.