LLMSurgeon: Diagnosing Data Mixture of Large Language Models

Anonymous ACL submission

Abstract

The pretraining data mixture of Large Language Models (LLMs) act as their "digital DNA", fundamentally governing model behaviors. While existing Membership Inference Attacks (MIA) can detect whether individual training samples are included in the pretraining dataset, they fail to quantify the macroscopic distribution of domains. In this work, we formalize the Pretraining **Data Mixture Surgery (DMS)** problem: inferring the domain-level distribution of an LLM's training corpus solely from its generated texts. We propose **LLMSurgeon**, a principled framework that models DMS as an inverse problem under the label-shift assumption[1]. Unlike naive classification, LLMSurgeon utilizes a calibrated "soft" confusion matrix to rectify systematic classifier biases, accurately recovering the latent training prior. To rigorously evaluate this task, we introduce **LLMScan**, the first benchmark comprising open-source LLMs with transparent data recipes. Experiments demonstrate that LLMSurgeon significantly outperforms aggregation-based membership inference attack (MIA) baselines, recovering ground-truth mixtures with high fidelity. Our work offers a practical, post-hoc method for auditing the "digital DNA" of foundation models without accessing their training data.

1 Introduction

Modern Large Language Models (LLMs) (OpenAI, 2025; Gemini Team, 2025; Yang et al., 2025; Liu et al., 2025) operate as "digital alchemy": while their capabilities in reasoning and coding are undeniable, the ingredients of their massive training corpora remain one of the most significant guarded secrets in AI. This lack of transparency creates a critical bottleneck for safety, accountability, and governance. Without access to the "digital DNA" of these models and their pretraining data composition, it is impossible to audit them for demographic biases, assess copyright infringement risks, or explain performance disparities across domains. As LLMs become central to social infrastructure, the inability to answer the simple question "What was this LLM trained on?" poses a fundamental challenge to trustworthy AI.

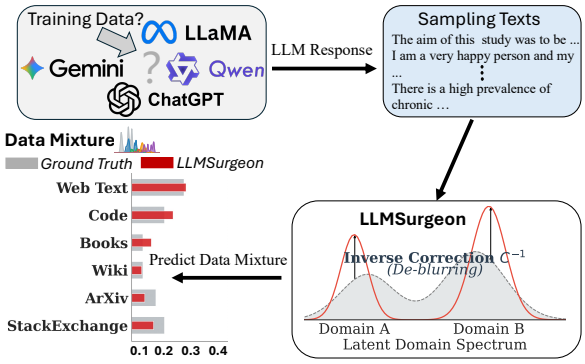

Figure 1: Overview of Data Mixture Surgery Problem and LLMSurgeon.

To date, efforts to peer inside these black boxes have primarily relied on Membership Inference Attacks (MIA) (Shokri et al., 2017; Carlini et al., 2021a; Shi et al., 2023; Zhang et al., 2024a,b). While these tools are effective at the microscopic level of identifying whether a specific document was seen during training, they fail to answer macroscopic questions about data distribution. We face a paradox: MIA tools can detect a single grain of sand (a specific sample), but lack the capacity to describe the landscape of the beach (the domain composition). Attempting to estimate global composition by aggregating millions of noisy, instance-level MIA predictions is computationally prohibitive and prone to catastrophic error accumulation.

This gap necessitates a shift from instance-level detection to **Data Mixture Surgery (DMS)**: estimating the global proportions of data domains that constitute a model's world knowledge base. Crucially, we frame DMS as a targeted auditing

[1]Label shift assumes that domain proportions change, while domain-specific language patterns remain unchanged.

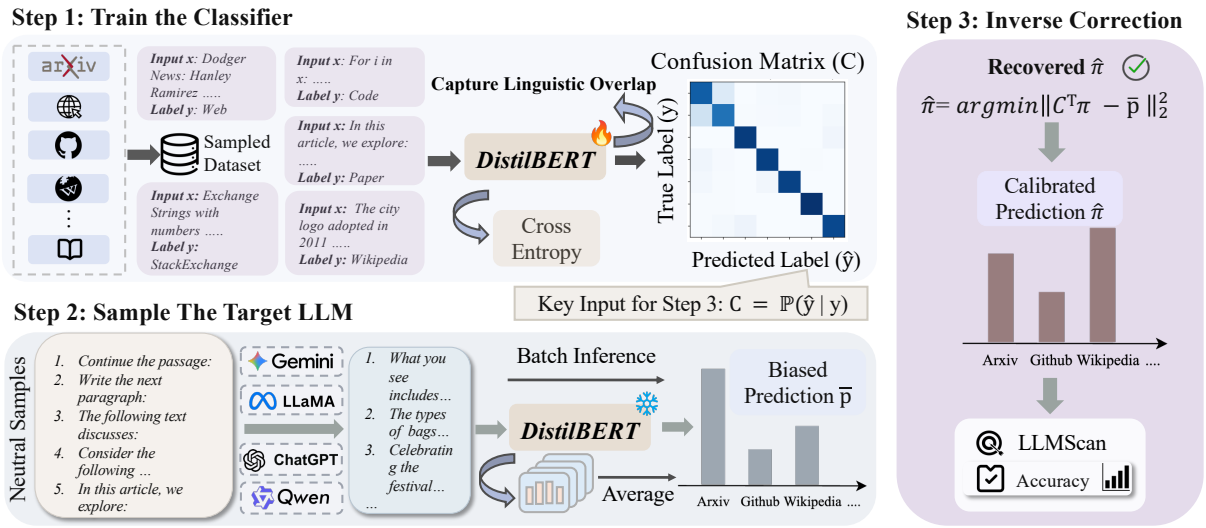

Figure 2: **Overview of Data Mixture Surgery Problem and LLMSurgeon.** This figure illustrates the pipeline of estimating pretraining data mixture from generated text.

task rather than open-ended discovery. This formulation aligns with real-world safety compliance needs, where regulators seek to verify compliance with specific data policies rather than uncovering unknown proprietary sources. To solve this issue, we propose **LLMSurgeon**, a principled framework that treats DMS as an inverse problem governed by the Label Shift hypothesis. In the context of unsupervised LLMs, this hypothesis posits that while the global mixture proportions of domains (the prior) shift between the original training set and the model's generated output, the linguistic characteristics defining each domain (the conditional distribution) remain statistically invariant. While prompts and alignment are known to distort generation distributions (Xiao et al., 2024; Kirk et al., 2023), we employ neutral sampling to minimize such stylistic shifts, ensuring the recovered mixture faithfully reflects the pretraining prior.

Leveraging this stability, **LLMSurgeon** operates in a rigorous three-stage pipeline. **First**, we pretrain an external classification model on known reference data and compute a calibrated "soft" confusion matrix to characterize its systematic bias between similar text domains. **Second**, we sample the neutral output responses of target LLM and classify its generated texts using the frozen classifier, obtaining a biased observation of the latent prior. **Finally**, unlike naive approaches that accept these biased counts directly, LLMSurgeon utilizes the pre-computed confusion matrix to mathematically "de-blur" the observations, solving the inverse problem to recover the ground-truth training proportions with high fidelity.

To rigorously evaluate Data Mixture Surgery (DMS), we introduce **LLMScan**, the first benchmark comprising open-source LLMs (Groeneveld et al., 2024; Biderman et al., 2023; Touvron et al., 2023; Liu et al., 2023) with transparent, ground-truth data recipes. This prevents the common pitfall of evaluating auditing tools on synthetic data that does not reflect real-world pretraining dynamics (Duan et al., 2024). Experiments on LLMScan demonstrate that LLMSurgeon significantly outperforms aggregation-based baselines, offering a practical, post-hoc method for auditing foundation models without accessing their weights or training data.

Our contributions are threefold. (1) We formalize the **Data Mixture Surgery (DMS)** problem, shifting the focus from membership inference to distribution recovery. (2) We propose **LLMSurgeon**, a lightweight method that accurately infers latent domain priors from generated text. (3) We present **LLMScan**, a verifiable benchmark to standardize the evaluation of Data Mixture Surgery (DMS).

2 Related Work

Membership Inference Attack asks whether a particular sample was included in a model's training data (Shokri et al., 2017), typically by exploiting behavioral differences between training and non-training examples (Shokri et al., 2017; Carlini et al., 2021a). Membership Inference Attack (MIA) has become standard for auditing privacy leakage and evaluating defenses such as differential privacy (Abadi et al., 2016), and have been extended from vision to language settings, including pretraining-corpus detection for LLMs (Zhang

et al., 2024b; Shi et al., 2023; Zhang et al., 2024a; Xie et al., 2024; Mattern et al., 2023). Despite their utility, MIA is fundamentally instance-level, it delivers a binary or probabilistic decision per example, rather than a corpus-level estimate of how training data is composed. MIA often relies on confidence scores, logits, or carefully matched reference datasets and can be brittle under distribution shift, calibration errors, or regularization (Watson et al., 2022). For LLMs, many techniques operate on short spans or token windows, making them expensive to scale and ill-suited for inferring high-level category proportions of large, heterogeneous corpora. In contrast, we pursue a *corpus-level* objective: `Data Mixture Surgery (DMS)`.

Data Mixture Optimization focuses on selecting or reweighting pretraining data to enhance model performance, employing techniques like importance sampling and gradient-based reweighting (Xie et al., 2023a; Chen et al., 2024a; Xie et al., 2023b; Chen et al., 2024b). However, these methods operate *pre-hoc*, requiring full access to training loop and raw datasets. They cannot audit fixed, closed-source models where data mixture is unknown. We tackle the problem: performing post-hoc inference of effective multi-domain mixture solely from a trained model's generated outputs.

Dataset Usage Cardinality Inference (DUCI). Tong et al. (2025) estimates the proportion of a specific, known dataset used during training by aggregating debiased membership predictions. While effective for auditing a single candidate corpus, this approach strictly requires access to the underlying data. In contrast, our work addresses `Data Mixture Surgery (DMS)`, recovering the global multi-domain mixture π solely from model generations without requiring access to the training dataset, replacing pointwise aggregation with calibrated label-shift inversion.

3 Data Mixture Surgery

3.1 Problem Formulation

Let $\mathcal{X}$ denote space of text sequences and $\mathcal{Y} = \{1, \ldots, K\}$ be a set of K disjoint semantic domains (e.g., *Paper*, *Wikipedia*, *Code*). We define conditional distribution of text within a specific domain i as $p_i(x) \triangleq p(x \mid y = i)$.

During pretraining, the LLM optimizes its parameters θ over a training corpus $\mathcal{D}_{\text{train}}$. We model this corpus as a mixture distribution defined by the ground-truth mixing vector $\boldsymbol{\alpha} \in \Delta^{K-1}$:

$$p_{\boldsymbol{\alpha}}(x) = \sum_{i=1}^{K} \alpha_i p_i(x) \tag{1}$$

where α_i represents the true proportion of domain i in the training set (the quantity we wish to audit).

When queried with neutral prompts, the trained LLM generates samples from an induced distribution $q(x)$. Due to optimization dynamics, underfitting, or sampling temperature, the model's internal usage of domains may diverge slightly from the exact training proportions. We model this generation distribution as:

$$q_{\boldsymbol{\pi}}(x) = \sum_{i=1}^{K} \pi_i p_i(x) \tag{2}$$

where $\boldsymbol{\pi} \in \Delta^{K-1}$ is latent effective prior, domain mixture actually encoded by model's behavior.

The Prediction Goal. Our objective is `Data Mixture Surgery (DMS)`. Given a set of generated samples $X_{\text{gen}} = \{x_n\}_{n=1}^{N} \sim q_{\boldsymbol{\pi}}(x)$, we aim to estimate the vector $\boldsymbol{\pi}$. We assume generative process adheres to the *Label Shift* hypothesis. Specifically, while marginal distribution of domains shifts from $\boldsymbol{\alpha}$ (training) to $\boldsymbol{\pi}$ (generation), the conditional feature distributions remain invariant:

$$q(x \mid y = i) \approx p(x \mid y = i) \tag{3}$$

This implies that when model generates "Code", it statistically resembles the "Code" seen during training, even if the frequency of "Code" generation (π_{code}) differs from its training frequency (α_{code}). This allows us to treat the recovery of $\boldsymbol{\pi}$ as a distribution matching problem, solving for mixing coefficients that best explain observed X_{gen}.

Challenges. A straightforward strategy for estimating data mixture is to aggregate Membership Inference Attacks (MIA) signals across sampled corpora. However, this naive approach faces three critical bottlenecks. **Token Limitations:** MIAs are typically designed for short sequences, making them computationally prohibitive to scale across large, document-level corpora. **Error Accumulation:** Pointwise prediction errors compound when aggregated over millions of samples, leading to high-variance distribution estimates. **Algorithmic Bias:** MIAs often exhibit domain-dependent performance (e.g., higher accuracy on memorized code than generic text), introducing systematic skew that distorts the recovered mixture proportions.

Target Model	Parameters	Pretraining Corpus	Granularity Level	Pre-Defined Domains (K)
General Purpose Models				
LLaMA-1 (Touvron et al., 2023)	7B, 65B	Public Mix (CC, Wiki, etc.)	Coarse-Grained	6
OLMo (Groeneveld et al., 2024)	1B	Dolma (Soldaini et al., 2024)	Coarse-Grained	6
Amber (Liu et al., 2023)	13B	LLM360 Mix (Liu et al., 2023)	Coarse-Grained	6
Pile-Based Models				
Pythia (Biderman et al., 2023)	2.8B, 12B	The Pile (Gao et al., 2020)	Mid-Grained	17
GPT-Neo (Black et al., 2021)	2.7B	The Pile	Mid-Grained	17
Domain Specialized Models				
StarCoder (Li et al., 2023)	15.5B	The Stack (Kocetkov et al., 2022)	Fine-Grained	87

Table 1: **Overview of the LLMScan Benchmark Suite.** The benchmark covers three levels of auditing granularity (Coarse, Mid, Fine) across varying model scales (1B to 65B).

3.2 LLMScan: First Benchmark for Data Mixture Surgery

Evaluating DMS requires ground truth that is largely absent in closed-source AI. To bridge this gap, we introduce **LLMScan**, a benchmark comprising 8 open-source foundation models (1B–65B parameters) with verifiable data genealogies (Groeneveld et al., 2024; Touvron et al., 2023; Liu et al., 2023; Biderman et al., 2023; Li et al., 2023). As summarized in Table 1, we evaluate auditing performance across three resolution levels to assess robustness against semantic overlap. The **Coarse-Grained** setting ($K = 6$) utilizes SlimPajama-DC (Shen et al., 2023) definitions to audit general-purpose models (LLaMA-1, OLMo, Amber), merging overlapping web sources (e.g., C4 and CommonCrawl) for stability. The **Mid-Grained** setting ($K = 17$) employs The Pile (Gao et al., 2020) taxonomy to analyze the Pythia family and GPT-Neo, challenging the auditor to distinguish more sub-domains. Finally, the **Fine-Grained** setting ($K = 87$) uses The Stack (Kocetkov et al., 2022) to distinguish specific programming languages in StarCoder. To establish the ground truth, we define the candidate domains K according to each model's official pretraining documentation, extracting the exact composition vector directly from their technical reports. By strictly adhering to documented recipes rather than synthetic mixtures, LLMScan ensures that auditing performance is measured against real-world, large-scale training dynamics.

4 LLMSurgeon: A simple Data Mixture Surgery Method

Our framework recovers the latent effective prior $\boldsymbol{\pi}$ by treating the problem as a *label-shift inversion* task. We decompose the process into three stages: (1) Characterizing the systematic bias of a proxy classifier, (2) Sampling the target LLM's outputs, and (3) Solving the constrained inverse problem.

4.1 Characterizing Systematic Bias

Since we cannot access the target LLM's internal states, we employ an external proxy domain classifier $f_\phi : \mathcal{X} \to \Delta^{K-1}$. However, no classifier is perfect; applying f_ϕ directly to generated text yields a biased estimate due to domain confusion (e.g., confusing *C++* with *C*).

We explicitly model this error profile as a linear operator. Using a held-out reference dataset $\mathcal{D}_{\text{ref}}$ where the ground-truth domain labels are known, we compute the **soft confusion matrix** $C \in \mathbb{R}^{K \times K}$. Each entry C_{ij} represents the expected probability that the classifier predicts domain j given a sample truly from domain i:

$$C_{ij} = \mathbb{E}_{x \sim p_i} \left[f_\phi(x)_j \right] \tag{4}$$

where $f_\phi(x)_j$ denotes the predicted probability for class j. Here, C serves as a *calibration operator* that maps the true domain distribution to the classifier's biased observation space. If the classifier were perfect, C would be the identity matrix I. In reality, off-diagonal elements capture the semantic overlap between domains.

4.2 Observing the Target Distribution

To probe the target LLM, we generate a corpus of synthetic text $X_{\text{gen}} = \{x_n\}_{n=1}^{N}$ using neutral prompts designed to trigger the model's natural domain prior $q_{\boldsymbol{\pi}}$ (as defined in Eq. 2).

We pass these generations through our proxy classifier to obtain the **empirical mean prediction vector** $\bar{\mathbf{p}} \in \mathbb{R}^K$:

$$\bar{\mathbf{p}} = \frac{1}{N} \sum_{n=1}^{N} f_\phi(x_n) \tag{5}$$

Crucially, notice that $\bar{\mathbf{p}}$ is not the true distribution $\boldsymbol{\pi}$, but rather the convolved observation corrupted by the proxy classifier's bias.

4.3 The Inverse Surgery: Recovering $\boldsymbol{\pi}$

We now link observed signal $\bar{\mathbf{p}}$ to latent target $\boldsymbol{\pi}$. By linearity of expectation and the definition of generation mixture $q_{\boldsymbol{\pi}}(x) = \sum_k \pi_k p_k(x)$, expected output of classifier is:

$$\mathbb{E}_{x\sim q_{\boldsymbol{\pi}}}[f_\phi(x)] = \sum_{k=1}^{K} \pi_k \underbrace{\mathbb{E}_{x\sim p_k}[f_\phi(x)]}_{\text{Row } k \text{ of } C} = C^\top \boldsymbol{\pi} \tag{6}$$

This derivation reveals that observed distribution $\bar{\mathbf{p}}$ approximates $C^\top \boldsymbol{\pi}$. Consequently, recovering true prior $\boldsymbol{\pi}$ becomes a constrained linear inverse problem. We solve for optimal $\hat{\boldsymbol{\pi}}$ that minimizes reconstruction error:

$$\hat{\boldsymbol{\pi}} = \underset{\boldsymbol{\pi}\in\Delta^{K-1}}{\arg\min} \left\| C^\top \boldsymbol{\pi} - \bar{\mathbf{p}} \right\|_2^2 \tag{7}$$

subject to constraints $\sum \pi_k = 1$ and $\pi_k \geq 0$.

Why MIA Fails. MIA methods target *microscopic* instance detection, not *macroscopic* distribution recovery. Aggregating their noisy, domain-sensitive predictions leads to catastrophic error accumulation. In contrast, `LLMSurgeon` bypasses instance-level instability by solving a global inverse problem directly from distributional statistics.

5 Experiments

5.1 Experimental Settings and Metrics

We evaluate `LLMSurgeon` on `LLMScan`, our proposed benchmark with publicly documented pretraining data mixtures. To assess the robustness of `LLMSurgeon` across different resolutions, we conduct experiments at three distinct granularity levels. For each level, we select a representative open-source dataset as reference corpus to train domain classifier f_ϕ and compute confusion matrix C. For the **Coarse-Grained** setting, we utilize SlimPajama-627B-DC as the reference pool, sampling 5,000 documents from each of the 6 broad data domains for classifier training. In the **Mid-Grained** setting, we increase resolution to 17 diverse domains[2] and use The Pile to capture finer distributional shifts. Finally, for the **Fine-Grained** setting, we focus specifically on programming domain using The Stack, distinguishing between 87 different programming languages. Considering the correctness of the ground truth, we select a suite of fully open-source LLMs where the pretraining data mixtures are publicly available, serving as the ground truth for quantitative evaluation. Our target models include LLaMA-1 (Touvron et al., 2023)[3], OLMo (Groeneveld et al., 2024), Amber (Liu et al., 2023), Pythia (Biderman et al., 2023), StarCoder (Li et al., 2023) and GPT-Neo (Black et al., 2021). For each target model, the set of candidate domains K is defined according to its official pretraining documentation under a closed-world assumption. For instance, when analyzing StarCoder, we restrict the taxonomy to the 87 coding languages present in its training data. To infer the mixture, we generate outputs using neutral prompts and apply the `LLMSurgeon` pipeline described in Section 4. All inference experiments for LLaMA1-65B were conducted on 4 NVIDIA A100 GPUs, while all other models were evaluated on NVIDIA RTX 4090 GPUs. To rigorously quantify reconstruction fidelity, we report Overlap Accuracy ($1 - \frac{1}{2}\sum_{k=1}^{K} |\alpha_k - \hat{\pi}_k|$) as our primary metric, alongside Mean Absolute Error (MAE) for average deviation and Coefficient of Determination (R^2) to evaluate structural correlation.

5.2 Baselines

To demonstrate the effectiveness of `LLMSurgeon`, we compare it against two categories of baselines: aggregation-based Membership Inference Attacks and a direct classification baseline.

Membership Inference Attacks (MIA) baselines. Since there are no existing methods specifically designed for DMS, we adapt state-of-the-art MIA techniques to the distribution estimation task. We employ a comprehensive suite of 11 MIA methods, including metric-based approaches (Carlini et al., 2021a; Yeom et al., 2018; Wang et al., 2024), reference-based approaches (Carlini et al., 2021a), and neighborhood-based approaches (Xie et al., 2024; Mattern et al., 2023; Zhang et al., 2024a). MIA methods typically output a binary prediction $\hat{y}_i^{(c)} \in \{0, 1\}$ for a single sample i from domain c, where 1 indicates the sample was likely seen during training. To convert these instance-level signals into a domain proportion estimate, we aggregate predictions over a sampled validation set ($N = 5000$ per domain). We define the MIA-inferred proportion r_c for domain c as the normal-

[2]We remove 5 copyright infringement domains.

[3]We select LLMs where the pretraining data mixtures are publicly available to ensure reliable of results.

Methods	Coarse-Grained (Easy)				Mid-Grained (Middle)			Fine-Grained (Hard)
	OLMo-1B	LLaMA1-7B	Amber-13B	LLaMA1-65B	GPT-Neo-2.7B	Pythia-2.8B	Pythia-12B	StarCoder-15.5B
MIA Methods:								
Joint-Logit (Carlini et al., 2022)	35.20	35.03	41.50	35.05	52.19	52.23	52.26	25.60
Loss (Yeom et al., 2018)	29.74	33.86	40.14	34.45	53.54	53.27	52.36	25.76
Ref (Cachola et al., 2020)	38.20	35.15	41.52	35.09	52.33	52.33	52.33	25.46
GradNorm (Wang et al., 2024)	34.72	36.03	39.41	46.52	58.78	55.66	35.98	27.54
Zlib (Carlini et al., 2021b)	29.30	28.11	36.73	38.08	55.16	53.86	47.80	21.06
Neighbor (Mattern et al., 2023)	41.74	40.13	40.31	35.74	51.85	52.21	52.90	25.47
Min-K% (Shi et al., 2023)	30.84	28.12	36.81	23.41	53.47	53.24	50.23	23.41
Min-K%++ (Zhang et al., 2024a)	30.99	32.65	40.56	32.24	53.47	53.97	33.09	23.57
DC-PDD (Zhang et al., 2024b)	35.93	34.82	38.58	35.03	56.07	55.43	52.53	25.98
Recall (Xie et al., 2024)	48.05	35.08	41.55	35.08	49.04	55.23	52.63	25.91
DUCI (Tong et al., 2025)	35.16	35.22	41.30	35.27	52.86	52.62	52.62	25.44
CCI Methods:								
LLMSurgeon (Ours)	$94.46^{+46.4}$	$95.14^{+55.0}$	$78.87^{+37.3}$	$94.26^{+47.7}$	$61.86^{+3.1}$	$63.20^{+7.5}$	$65.98^{+13.1}$	$30.37^{+2.8}$

Table 2: LLMScan Benchmark. The performances are reported in overlap accuracy %.

ized count of positive predictions:

$$r_c = \frac{\sum_{i=1}^{N} \hat{y}_i^{(c)}}{\sum_{j=1}^{K} \sum_{i=1}^{N} \hat{y}_i^{(j)}}. \quad (8)$$

This baseline tests whether simply counting "detected" samples from each domain can serve as a proxy for the data mixtures.

w/o Inverse Correction baseline. To isolate the gain from our label-shift inversion, we evaluate a "Direct Estimation" baseline ($\hat{\pi}_{\text{direct}} = \bar{p}$), which uses the raw aggregated classifier outputs defined in Eq. 5 without correction. Given the high accuracy of the proxy classifier, this baseline establishes a strong performance lower bound. This comparison serves to verify whether our mathematical calibration provides the necessary refinement to rectify systematic biases beyond naive classification.

5.3 Results of LLMScan Benchmark

Table 2 demonstrates that LLMSurgeon significantly outperforms all 11 aggregation-based MIA baselines, validating the superiority of a distribution-centric approach. On general-purpose models like LLaMA-1-7B and OLMo-1B, we achieve overlap accuracies of **95.14%** and **94.46%** respectively, whereas the strongest baselines (e.g., *Neighbor*, *Recall*) struggle to exceed 50%. This performance remains robust across model scales (e.g., LLaMA-7B to 65B), suggesting our method effectively captures the fundamental generation probability $q(x)$. In the challenging fine-grained setting (StarCoder), semantic blurring between similar languages lowers absolute accuracy to 30.37%, yet LLMSurgeon still surpasses the best baseline (*GradNorm*: 28.06%), proving that our inverse bias correction provides the most reliable estimate even when domain boundaries are indistinct.

Classifier	OLMo-1B	LLaMA1-7B	Amber-13B	StarCoder-15.5B	LLaMA1-65B
TF-IDF	85.07	86.83	59.61	28.56	92.64
MLP	74.52	82.97	64.77	21.57	85.36
Transformer	89.18	90.22	75.88	23.11	94.25
DistilBERT	94.46	95.14	78.87	30.37	94.26

Table 3: Classifier performance effect ablation study.

5.4 Ablation Studies

Effect of Classifier Backbone. To assess the effect of different classifier backbones on the overall performance, we conduct an ablation study. Specifically, we evaluate four settings: fine-tuned DistilBERT, DistilBERT architecture transformer classifier trained from scratch, TF-IDF, and a simple MLP classifier, as shown in Table 3. We observe that fine-tuned DistilBERT consistently outperforms other backbones across most settings, achieving an absolute improvement of 4.92% over the second-best classifier on LLaMA1-7B, and a gain of 1.81% under the more fine-grained detection setting with StarCoder. Based on these results, we adopt fine-tuned DistilBERT as the default classifier backbone in all subsequent experiments.

Effect of Domain Granularity. To determine resolution limits, we evaluate LLMSurgeon across Coarse, Mid, and Fine granularities in Figure 3. Results reveal a hierarchy driven by semantic separability. Coarse-grained is near-perfect ($R^2 = 0.99$) due to distinct linguistic boundaries, while mid-grained estimation remains robust ($R^2 = 0.54$) despite increased topical overlap. In fine-grained setting, high semantic confusion (e.g., between C and C++) degrades correlation ($R^2 = 0.01$) by creating an ill-conditioned inverse problem. However, consistently low MAE (0.018) confirms that LLMSurgeon still successfully filters irrelevant domains, validating its utility for macroscopic auditing over microscopic dialect identification.

Effect of LLM's Pretraining Steps. To investigate

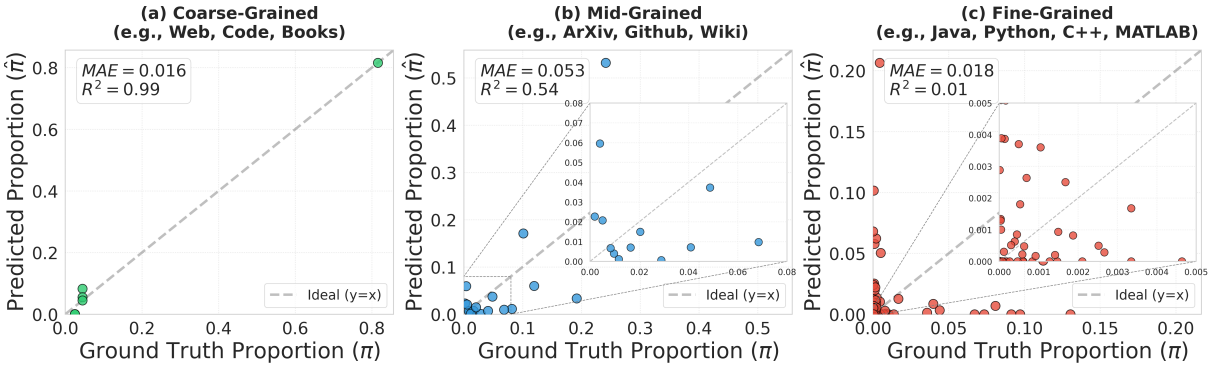

Figure 3: **Impact of Domain Granularity on Mixture Recovery.** We observe a performance hierarchy where coarse-grained recovery is near-perfect ($R^2 = 0.99$), whereas fine-grained estimation suffers ($R^2 = 0.01$) due to the high semantic confusion between similar categories (e.g., C vs. C++).

the evolution of internal domain priors, we apply LLMSurgeon to intermediate checkpoints of Amber-13B and OLMo-1B. Figure 4 reveals distinct training dynamics. Amber (top) exhibits a "fluctuation-then-convergence" pattern, where dominant domains (*Web, GitHub*) show high volatility during intermediate stages, likely reflecting curriculum learning or staged data injection strategies. In contrast, OLMo (bottom) displays a significantly more stable trajectory with lower error variance, suggesting a consistent data mixing strategy throughout training. Despite these dynamic differences, both models achieve sharp error reduction in the final phase. This confirms that LLMSurgeon recovers the composition of converged models and serves as a transparent tool for monitoring training stability and data scheduling effects.

Effect of Data Sampling Population. We evaluate how the number of reference documents (N) used to train the proxy classifier f_ϕ impacts estimation fidelity (Table 4). We observe that small sample sizes ($N = 100$) result in poor generalization (e.g., 20.15% on StarCoder). Increasing N to 1,000 yields substantial gains ($> 10\%$ on average), but performance saturates at $N = 5,000$. Interestingly, further increasing N to 10,000 offers negligible improvement or even slight regression (e.g., on LLaMA-7B) and introducing excessive noise or computational overhead. This suggesting that $N = 5,000$ sufficiently captures domain features without introducing noise. Consequently, we adopt $N = 5,000$ as the optimal trade-off between accuracy and computational efficiency.

Robustness of Sampling Text. To evaluate the robustness of LLMSurgeon to sampling process, we examine six distinct styles in Table 5. We find that **Neutral** sampling exhibits the highest robustness across general-purpose models, consistently maintaining top-tier performance (e.g., 95.14% on LLaMA-7B) where other styles fluctuate significantly. For instance, while **Expository** prompts achieve high accuracy on Amber-13B, they fail catastrophically on OLMo-1B (22.71%), indicating that strong stylistic biasing can distort the generated distribution $q_\pi(x)$ away from the latent training prior. However, we observe that neutral sampling performs suboptimally on domain-specialized models (e.g., StarCoder), likely due to the lack of specific triggers required to activate the specialized distribution. Given this trade-off, we adopt Neutral sampling as the default for general auditing to

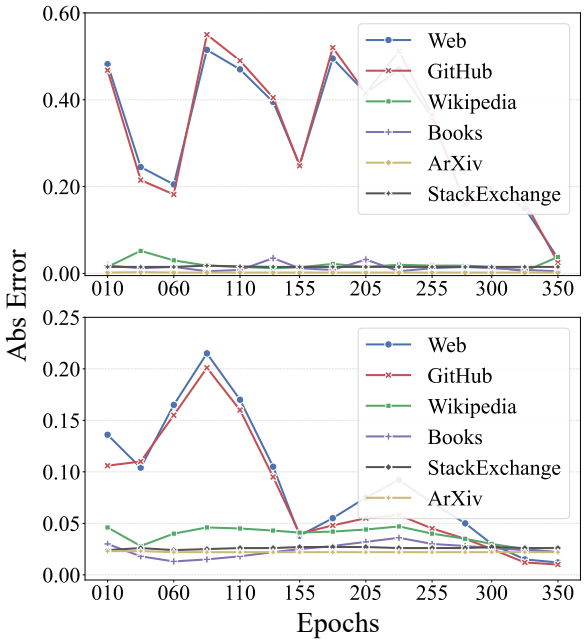

Figure 4: **Trajectories of domain estimation error throughout training. Top:** Amber-13B; **Bottom:** OLMo-1B. Despite different intermediate training dynamics, LLMSurgeon consistently converges to the ground-truth composition in the final checkpoints.

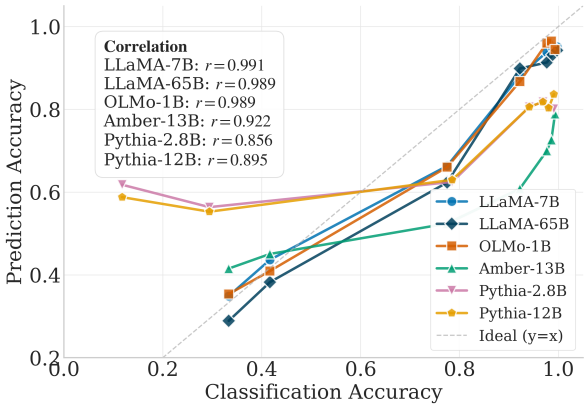

Figure 5: **Classification Acc vs Estimation Acc.** We observe a strong positive correlation (avg $r > 0.9$) between the proxy classifier's performance and the final mixture recovery accuracy.

Samples/Domain	OLMo-1B	LLaMA1-7B	Amber-13B	StarCoder-15.5B	LLaMA1-65B
100	73.28	85.78	71.77	20.15	83.77
1000	95.91	93.68	74.72	25.62	93.88
5000	94.46	95.14	78.87	30.37	94.26
10000	93.98	92.44	82.83	29.51	93.78

Table 4: Effect of Data Sampling Population.

Style	OLMo-1B	LLaMA1-7B	Amber-13B	StarCoder-15.5B	LLaMA1-65B
Instructional	80.50	89.38	70.34	30.37	90.69
Expository	22.71	90.60	91.53	25.62	90.93
Conversational	85.18	85.70	92.35	28.00	86.99
Coding	22.72	66.65	36.78	27.83	55.13
Math	79.48	79.56	67.15	24.16	78.97
Neutral	94.46	95.14	78.87	24.83	94.26

Table 5: Robustness of Sampling Text.

Method	OLMo-1B	LLaMA1-7B	Amber-13B	StarCoder-15.5B	LLaMA1-65B
w/o Inverse Correction	92.77	93.42	77.38	26.47	93.38
LLMSurgeon	94.46	95.14	78.87	30.37	94.26

(a) Importance of Inverse Correction

Setting	OLMo-1B	LLaMA1-7B	Amber-13B	LLaMA1-65B
Separate C4&CC	19.52	42.42	53.49	42.52
Merge C4&CC	94.46	99.14	78.87	94.26

(b) Similar Dataset Category Merging

Table 6: More ablation Studies. Panel (a) confirms the necessity of our inverse correction module. Panel (b) illustrates that merging semantically indistinguishable sources is critical to avoiding ill-conditioned inversion and ensuring stable estimation.

maximize estimation stability.

Effect of Inverse Bias Correction. A core contribution of LLMSurgeon is the use of the confusion matrix C to rectify the proxy classifier's outputs. To quantify this gain, we compare our full method against the "w/o Inverse Correction" baseline in Se 5.2. Table 6(a) shows that the inverse correction consistently improves estimation fidelity. For example, on StarCoder, applying the correction boosts accuracy to a relative improvement of nearly 15%. This confirms that Eq. 7 effectively disentangles the classifier's systematic confusion, sharpening the final distribution estimate.

Effect of Domains Pre-definition. DMS relies on the premise that the target domains are linguistically distinguishable. We investigate this boundary by attempting to separate *C4* from *Common Crawl* in the LLaMA training mixture analysis. As shown in Table 6(b), treating them as separate labels causes a catastrophic performance drop from 99.14% to 42.42%. This occurs because C4 is merely a filtered subset of Common Crawl; they share the same underlying semantic distribution, making $p(x|\text{C4}) \approx p(x|\text{CC})$. Consequently, the classifier cannot distinguish them, leading to an unstable estimation. Merging these semantically identical sources restores accuracy to near-perfect levels, validating our strategy of grouping overlapping sources into distinct semantic clusters.

Correlation Analysis. To validate the robustness of our inversion mechanism, we analyze the relationship between domain classification accuracy and the final mixture recovery (Prediction Accuracy). As illustrated in Figure 5, we observe a strong positive correlation across all model families, with Pearson correlation coefficients consistently exceeding $r = 0.85$ (e.g., LLaMA-7B reaches $r = 0.991$). The trajectory of the curves demonstrates that as models converge during pretraining, their generated domains become linguistically more distinct. This increased separability reduces the condition number of the confusion matrix, thereby allowing LLMSurgeon to recover the ground-truth mixture with significantly higher precision in later checkpoints.

6 Conclusion

We introduced **LLMSurgeon**, a principled framework for diagnosing implicit training data composition of Large Language Models. By formalizing **Data Mixture Surgery (DMS)**, we demonstrate that it is possible to recover latent domain prior from generated texts. We further established **LLMScan**, a comprehensive benchmark built on open-source models with verifiable ground truths, exposing limitations of traditional membership inference aggregation in this macroscopic setting. As AI development becomes increasingly opaque, LLMSurgeon establishes a vital, post-hoc mechanism for enforcing data transparency and accountability without relying on voluntary disclosure.

7 Limitations and Future Work

While LLMSurgeon provides an effective framework for auditing pretraining data, several limitations suggest avenues for future research. First, our method operates under the label-shift assumption, implying that neutral prompts elicit a generation distribution faithfully reflecting the model's training prior. This relationship may be distorted in models that have undergone extensive post-training alignment (e.g., RLHF or instruction tuning), which can shift the output distribution away from the original data mixture. Future work could investigate "inverse-alignment" techniques to disentangle the base distribution from alignment artifacts. Second, our method relies on a closed-world assumption defined by the auxiliary classifier's fixed taxonomy. Consequently, it cannot discover novel domains outside these K categories. Finally, as observed in our fine-grained analysis (Figure 3), the estimation accuracy is inherently bounded by the semantic separability of the domains. Highly overlapping categories (e.g., distinguishing between C and C++ code) result in dense, ill-conditioned confusion matrices that challenge the stability of the linear inversion. Addressing this resolution limit perhaps through hierarchical inference strategies or non-linear transport methods remains a promising direction for future exploration in pretraining data auditing.

Ethics Statement

The primary goal of this work is to advance transparency and accountability in the development of large language models by enabling the external auditing of opaque pretraining data mixture. By recovering the implicit composition of training corpora, LLMSurgeon facilitates the identification of potential biases, copyright infringements, and the over-representation of specific viewpoints, thereby empowering researchers and regulators to better understand foundation models. However, we acknowledge that this technology could be repurposed to reverse-engineer proprietary dataset curation strategies or identify vulnerabilities in specific models by revealing their lack of exposure to certain domains. We believe that the scientific and social benefits of open and verifiable model auditing outweigh these risks, and we emphasize that our method operates at the distributional level rather than extracting individual training examples or private data.

Regarding the using of AI, we just use generative models for writing assistance and coding drafting.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow.

Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel S Weld. 2020. Tldr: Extreme summarization of scientific documents. *arXiv preprint arXiv:2004.15011*.

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

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, and 1 others. 2021a. Extracting training data from large language models. In *USENIX Security Symposium*, volume 6.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, and 1 others. 2021b. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pages 2633–2650.

Daoyuan Chen, Yilun Huang, Zhijian Ma, Hesen Chen, Xuchen Pan, Ce Ge, Dawei Gao, Yuexiang Xie, Zhaoyang Liu, and Jinyang Gao. 2024a. Data-juicer: A one-stop data processing system for large language models. In *Companion of the 2024 International Conference on Management of Data*, pages 120–134.

Mayee Chen, Nicholas Roberts, Kush Bhatia, Jartu Olmo, Jelena Diakonikolas, and Christopher Re. 2024b. Skill-it! a data-driven skills framework for understanding and training language models. In *International Conference on Learning Representations (ICLR)*.

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

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Gemini Team. 2025. Gemini 3 technical report. https://deepmind.google/technologies/gemini/.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, and 1 others. 2024. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2023. Understanding the effects of rlhf on llm generalisation and diversity. *arXiv preprint arXiv:2310.06452*.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, and 1 others. 2022. The stack: 3 tb of permissively licensed source code. *arXiv preprint arXiv:2211.15533*.

Raymond Li, Loubna Ben Allal, Denis Kocetkov, Chenghao Mou, Carlos Muñoz Ferrandis, Sean Santacroce, Sean Hughes, Younes Belkada, and 1 others. 2023. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Zhengzhong Liu, Aurick Qiao, Willie Neiswanger, Hongyi Wang, Bowen Paik, Atanc Li, Gen Li, Xinyang Geng, Renrubin Wang, Yiran Sun, and 1 others. 2023. Llm360: Towards fully transparent open-source llms. In *arXiv preprint arXiv:2312.06550*. Reference for Amber.

Justus Mattern, Fatemeh Shao, Alexandre Sablayrolles, and Pierre Kochems. 2023. Membership inference against language models via neighborhood comparison. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11330–11343.

OpenAI. 2025. Introducing GPT-5.2. *OpenAI Blog*.

Zhiqiang Shen, Tianhua Tao, Liqun Ma, Willie Neiswanger, Zhengzhong Liu, Hongyi Wang, Bowen Tan, Joel Hestness, Natalia Vassilieva, Daria Soboleva, and 1 others. 2023. Slimpajama-dc: Understanding data combinations for llm training. *arXiv preprint arXiv:2309.10818*.

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

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

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, and 1 others. 2024. Dolma: An open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 15725–15788.

Yao Tong, Jiayuan Ye, Sajjad Zarifzadeh, and Reza Shokri. 2025. How much of my dataset did you use? quantitative data usage inference in machine learning. In *ICLR 2025 Workshop on Navigating and Addressing Data Problems for Foundation Models*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Jeffrey G Wang, Jason Wang, Marvin Li, and Seth Neel. 2024. Pandora's white-box: Precise training data detection and extraction in large language models. *arXiv preprint arXiv:2402.17012*.

Lauren Watson, Chuan Guo, Graham Cormode, and Alexandre Sablayrolles. 2022. On the importance of difficulty calibration in membership inference attacks. In *International Conference on Learning Representations*.

Zehao Xiao, Jiayi Shen, Mohammad Mahdi Derakhshani, Shengcai Liao, and Cees GM Snoek. 2024. Any-shift prompting for generalization over distributions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13849–13860.

Roy Xie, Junlin Wang, Ruomin Huang, Minxing Zhang, Rong Ge, Jian Pei, Neil Zhenqiang Gong, and Bhuwan Dhingra. 2024. Recall: Membership inference via relative conditional log-likelihoods. *arXiv preprint arXiv:2406.15968*.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifan Lu, Percy Liang, Quoc V Le,

Tengyu Ma, and Adams Wei Yu. 2023a. Doremi: Optimizing data mixtures speeds up language model pretraining. In *NeurIPS*.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. 2023b. Data selection for language models via importance resampling. In *Advances in Neural Information Processing Systems (NeurIPS)*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pages 268–282. IEEE.

Jingyang Zhang, Yaming Bao, Deming Wen, Houqiang Fan, Wenyuan Lin, Kai Liu, Wayne Xin Zhao, and Ji-Rong Wen. 2024a. Min-k%++: Improved baseline for detecting pre-training data from large language models. *arXiv preprint arXiv:2404.02936*.

Weichao Zhang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. 2024b. Pretraining data detection for large language models: A divergence-based calibration method. *arXiv preprint arXiv:2409.14781*.

Appendix

A Class-wise Detection Error

A.1 Coarse-Grained Detection Error

This section presents a coarse-grained detection error analysis over six high-level data source categories: Web, GitHub, Wikipedia, Books, ArXiv, and StackExchange. These categories represent broad groupings of training data sources and are used to evaluate model performance at an aggregate level. The analysis examines the alignment between model-predicted data source distributions and the corresponding ground-truth distributions under a coarse-grained categorization.

Figure 6 provides a visual comparison of predicted and ground-truth proportions for OLMo-1B, Amber-13B, LLaMA1-7B, and LLaMA1-65B. Solid bars denote model predictions, while hatched bars indicate ground-truth proportions. All values are normalized to sum to one, allowing for direct comparison across categories and models. This visualization offers an overview of detection results prior to detailed numerical inspection.

Table 7 reports the corresponding numerical results, including ground-truth proportions, model predictions, and absolute detection errors expressed as percentage deviations for each category. Detection errors are computed as the absolute difference between predicted and ground-truth proportions. Overall, the table summarizes coarse-grained detection behavior across models and data sources, with lower errors observed for dominant categories such as Web and GitHub, and comparatively larger deviations for less prevalent categories such as ArXiv and StackExchange.

A.2 Mid-Grained Detection Error

To further analyze detection performance beyond coarse-grained categories, this section reports mid-grained detection errors across 17 data source classes derived from the Pile dataset. These classes provide a finer partitioning of training data sources, covering a wider range of domain-specific datasets while remaining more aggregated than individual file types or instances. This mid-grained evaluation offers increased resolution relative to the coarse-grained setting while preserving interpretability.

The results of the mid-grained evaluation are presented in Table 8, which is divided into two subtables corresponding to class indices 0–10 and 10–17, respectively. For each class and model, we report the ground-truth proportion, the predicted proportion, and the absolute detection error expressed as a percentage. The class indices follow the ordering defined in the Pile dataset specification and are applied consistently across all evaluated models.

Overall, the results indicate larger detection errors for certain broad data sources, such as Common Crawl, while smaller errors are observed for categories such as HackerNews. These observations summarize the mid-grained detection behavior across the evaluated classes without further aggregation.

A.3 Fine-Grained Detection Error

We further report a fine-grained detection error analysis for the StarCoder model across 87 programming language categories, where each category corresponds to a distinct programming language. This analysis provides the most detailed view of detection behavior in our evaluation and enables a class-wise inspection of prediction accuracy at the level of individual programming languages.

Due to the large number of categories, the results are organized into multiple subtables, each covering a contiguous range of class indices, collectively spanning all 87 programming language classes. For each language, the tables report the ground-truth proportion, the model-predicted proportion, and the corresponding absolute detection error expressed as a percentage. All values are normalized such that proportions across all classes sum to one.

From the reported results, languages with relatively large ground-truth proportions, such as *Java*, *JavaScript*, *Python*, and *Go*, tend to exhibit larger absolute detection errors. In several cases, the model substantially underestimates these high-frequency languages, leading to large deviations between predicted and ground-truth proportions. For example, Java and JavaScript show zero predicted instances (0%). reflecting pronounced discrepancies in their predicted distributions.

In contrast, many low-frequency languages, including *Agda*, *Alloy*, *ANTLR*, and *Emacs Lisp*, are associated with comparatively small absolute errors. For these categories, both the ground-truth and predicted proportions are close to zero, resulting in limited absolute deviation despite relative differences. This pattern is consistently observed across multiple subtables.

Additionally, several mid-frequency languages exhibit notable overestimation, such as *Markdown*, *Julia*, and *Elixir*, where predicted proportions ex-

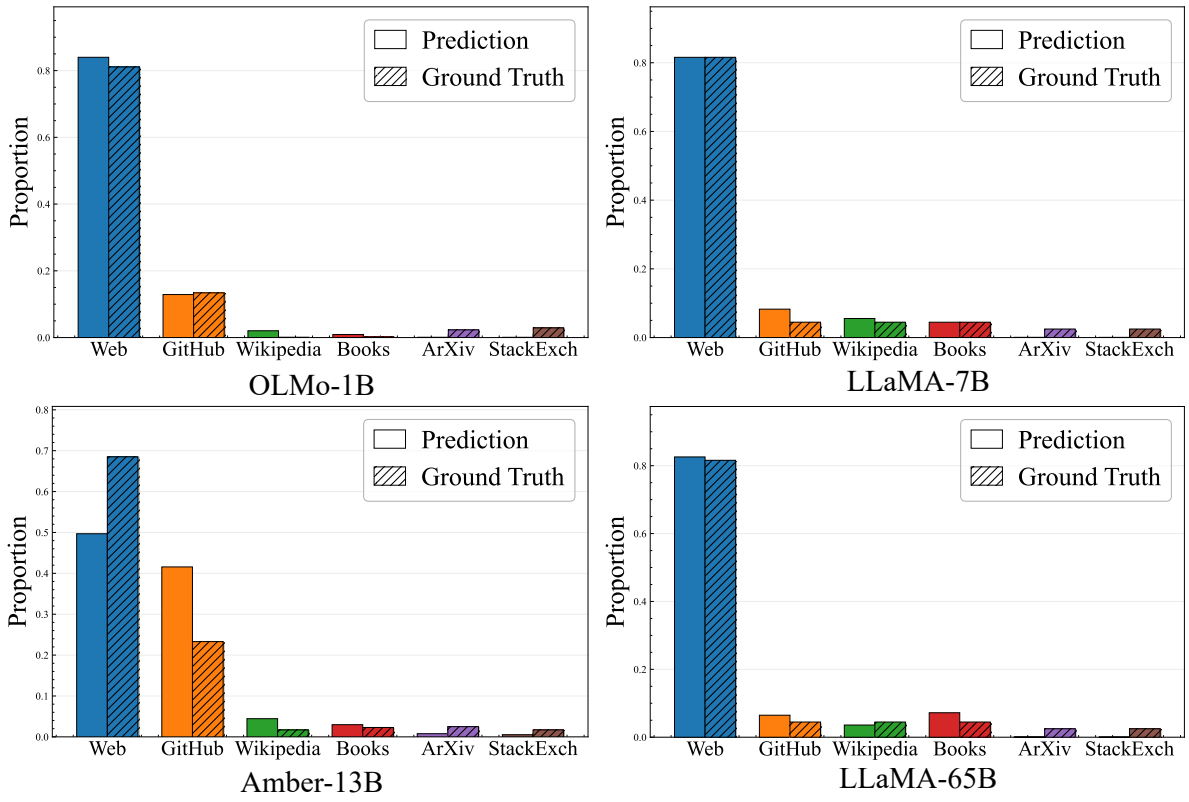

Figure 6: Predicted versus ground-truth proportions of training data sources for OLMo-1B, Amber-13B, LLaMA1-7B, and LLaMA1-65B across six coarse-grained categories. Solid bars denote model predictions, while hatched bars indicate ground-truth proportions. The figure presents a comparison of detection results across models with different numbers of parameters and across data source categories, providing a coarse-grained view of model predictions relative to the ground truth.

Model Name		Web	GitHub	Wikipedia	Books	ArXiv	StackExchange
OLMo-1B	Ground Truth	81.10	13.40	0.10	0.20	2.30	2.90
	Prediction	83.99	12.89	2.04	0.91	0.09	0.08
	Detection Error	2.89	0.51	1.94	0.71	2.21	2.82
Amber-13B	Ground Truth	68.50	23.30	1.70	2.30	2.50	1.70
	Prediction	49.69	41.56	4.45	2.98	0.78	0.53
	Detection Error	18.81	18.26	2.75	0.68	1.72	1.17
LLaMA1-7B	Ground Truth	81.59	4.48	4.48	4.48	2.49	2.49
	Prediction	81.58	8.27	5.55	4.47	0.07	0.06
	Detection Error	0.01	3.79	1.07	0.01	2.42	2.43
LLaMA1-65B	Ground Truth	81.59	4.48	4.48	4.48	2.49	2.49
	Prediction	82.58	6.48	3.59	7.21	0.08	0.05
	Detection Error	0.99	2.00	0.89	2.73	2.41	2.44

Table 7: Per-class detection error analysis across six classes for OLMo-1B, Amber-13B, LLaMA1-7B, and LLaMA1-65B, with errors reported as percentage absolute deviations for each category. The table reports the ground-truth proportions, model predictions, and the corresponding absolute detection errors for each class, enabling a detailed per-class comparison of detection results across the evaluated models.

Model Name		Common Crawl	GitHub	Wikipedia	Gutenberg	ArXiv	StackExchange	PubMed Central	FreeLaw	USPTO Backgrounds
Pythia-2.8B	GT	34.05	8.80	1.77	2.51	10.38	5.95	16.69	7.09	4.23
	Pred	53.21	17.09	1.50	0.06	6.01	0.98	3.35	1.16	3.73
	Error	19.16	8.29	0.27	2.45	4.37	4.97	13.34	5.93	0.50
Pythia-12B	GT	34.05	8.80	1.77	2.51	10.38	5.95	16.69	7.09	4.23
	Pred	48.74	18.21	0.97	0.07	8.17	1.11	4.71	0.84	2.71
	Error	14.69	9.41	0.8	2.44	2.21	4.84	11.98	6.25	1.52
GPT-Neo-2.7B	GT	34.05	8.80	1.77	2.51	10.38	5.95	16.69	7.09	4.23
	Pred	60.5	4.21	1.43	2.67	3.88	0.31	7.36	0.92	6.72
	Error	26.45	4.59	0.34	0.16	6.50	5.64	9.33	6.17	2.49

(a) Per-class detection error for classes 0-10.

Model Name		PubMed Abstracts	DM Mathematics	Ubuntu IRC	EuroParl	HackerNews	PhilPapers	NIH ExPorter	Enron Emails
Pythia-2.8B	GT	3.56	1.43	1.02	0.84	0.72	0.44	0.35	0.17
	Pred	0.72	0.70	0.11	0.40	0.67	2.08	5.96	2.27
	Error	2.84	0.73	0.91	0.44	0.05	1.64	5.61	2.10
Pythia-12B	GT	3.56	1.43	1.02	0.84	0.72	0.44	0.35	0.17
	Pred	0.91	1.42	0.14	0.54	0.57	1.55	7.54	1.79
	Error	2.65	0.01	0.88	0.30	0.15	1.11	7.19	1.62
GPT-Neo-2.7B	GT	3.56	1.43	1.02	0.84	0.72	0.44	0.35	0.17
	Pred	0.68	0.90	0.08	0.16	0.19	1.99	7.17	0.85
	Error	2.88	0.53	0.94	0.68	0.53	1.55	6.82	0.68

(b) Per-class detection error for classes 10-17.

Table 8: Per-class detection error analysis across 17 classes for models trained on the Pile dataset, with errors reported as percentage absolute deviations for each category.

ceed the corresponding ground-truth values by a substantial margin. Conversely, languages such as *C*, *C++*, and *Rust* are markedly underestimated, contributing to larger detection errors within their respective class groups.

Overall, the fine-grained results reveal substantial variability in detection accuracy across programming languages, with absolute errors influenced by both the underlying ground-truth frequency of a language and the model's tendency to over- or under-predict specific language categories. These detailed per-language results serve as a supplementary reference for understanding class-wise detection behavior at the finest granularity.

	ada	agda	alloy	antlr	applescript	assembly	augeas	awk	batchfile	bluespec
GT	0.039	0.010	0.001	0.008	0.001	0.233	0.000	0.003	0.035	0.005
Pred	0.230	0.073	0.111	0.042	0.458	0.603	0.076	0.227	0.379	0.036
Error	0.191	0.063	0.110	0.034	0.457	0.370	0.076	0.224	0.344	0.031

(a) Per-class detection error for classes 0–10 evaluated on StarCoder.

	c	c-sharp	clojure	cmake	coffeescript	common-lisp	cpp	css	cuda	dart
GT	8.082	6.697	0.069	0.068	0.095	0.210	7.336	0.450	0.084	0.549
Pred	0.433	0.009	0.170	1.478	3.809	0.068	0.064	0.039	0.022	0.147
Error	7.649	6.688	0.101	1.410	3.714	0.142	7.272	0.411	0.062	0.402

(b) Per-class detection error for classes 10-20 evaluated on StarCoder.

	dockerfile	elixir	elm	emacs-lisp	erlang	f-sharp	fortran	glsl	go	groovy
GT	0.063	0.107	0.045	0.061	0.105	0.092	0.267	0.060	3.566	0.137
Pred	0.180	2.843	0.117	0.051	0.321	0.000	1.228	0.194	0.110	0.474
Error	0.117	2.736	0.072	0.010	0.216	0.092	0.961	0.134	3.456	0.337

(c) Per-class detection error for classes 20-30 evaluated on StarCoder.

	haskell	html	idris	isabelle	java	java-server-pages	javascript	json	julia	kotlin
GT	0.335	4.403	0.005	0.012	13.037	0.147	9.703	0.150	0.197	0.852
Pred	0.045	0.082	0.737	0.302	0.000	0.080	0.000	1.090	6.980	0.226
Error	0.290	4.321	0.732	0.290	13.037	0.067	9.703	0.940	6.783	0.626

(d) Per-class detection error for classes 30-40 evaluated on StarCoder.

	lean	agda	l-coffeescript	l-haskell	lua	makefile	maple	markdown	mathematica	matlab
GT	0.014	0.001	0.001	0.008	0.430	0.197	0.001	11.236	0.187	0.000
Pred	0.705	0.000	1.594	1.869	4.263	0.058	0.790	14.348	0.308	0.904
Error	0.691	0.001	1.593	1.861	3.833	0.139	0.789	3.112	0.121	0.904

(e) Per-class detection error for classes 40-50 evaluated on StarCoder.

	ocaml	pascal	perl	php	powershell	prolog	protocol-buffer	python	r	racket
GT	0.154	0.252	0.335	9.131	0.168	0.001	0.046	9.057	0.045	0.005
Pred	1.256	0.188	0.453	0.155	0.688	0.819	0.049	27.029	5.590	0.106
Error	1.102	0.064	0.118	8.976	0.520	0.818	0.003	17.972	5.545	0.101

(f) Per-class detection error for classes 50-60 evaluated on StarCoder.

	restructuredtext	rmarkdown	ruby	rust	sas	scala	scheme	shell	solidity	sparql
GT	0.498	0.009	1.021	1.366	0.018	0.704	0.030	0.463	0.128	0.006
Pred	3.826	2.176	1.461	0.076	0.198	0.092	0.138	0.777	0.069	0.266
Error	3.328	2.167	0.440	1.290	0.180	0.612	0.108	0.314	0.059	0.260

(g) Per-class detection error for classes 60-70 evaluated on StarCoder.

	sql	stan	standard-ml	stata	systemverilog	tcl	tcsh	tex	thrift	typescript
GT	1.663	0.001	0.029	0.049	0.059	0.053	0.003	0.780	0.001	3.977
Pred	0.655	0.042	0.699	1.178	0.309	0.355	0.186	0.152	0.109	0.302
Error	1.008	0.041	0.670	1.128	0.250	0.302	0.183	0.628	0.108	3.675

(h) Per-class detection error for classes 70-80 evaluated on StarCoder.

	verilog	vhdl	visual-basic	xslt	yacc	yaml	zig
GT	0.000	0.141	0.213	0.008	0.016	0.150	0.026
Pred	0.046	0.106	0.818	0.063	0.307	0.871	0.021
Error	0.046	0.035	0.605	0.055	0.291	0.721	0.005

(i) Per-class detection error for classes 80-87 evaluated on StarCoder.

Table 9: Per-class detection error analysis across 87 programming languages for the StarCoder model, reporting percentage absolute errors for each language category.