LINMIX: DATA MIXING AS LINEAR DOMAIN ATTRIBUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Language model performance depends strongly on the mixture of domains in the pretraining data corpus. Popular data mixing methods select these domain weights by training many small proxy models on random mixes and then fitting a regression model to predict performance from the mix. We recognize this practice as a domain attribution problem, where the regression measures each domain's contribution to the target metric. Through this lens, the standard Dirichlet sampling of proxy data mixes is a poor regression design that requires tuning and cannot support a linear fit for hundreds of domains. Building on classical optimal experimental design, we introduce LINMIX, which samples mixes at the vertices of the capped simplex so that each domain is at its cap or absent. This hyperparameter-free vertex sampling decorrelates domain effects, enabling an interpretable linear model whose coefficients are the domain attributions. The optimal mix given by LINMIX provably keeps only domains with positive attribution values and requires no search over candidate mixes. We further establish the optimality of vertex sampling, and prove that Dirichlet sampling inflates the attribution error when its mixes concentrate around the prior data mix. Pretraining experiments with 1B-parameter OLMo2 and 4B Qwen3.5 language models show that LINMIX outperforms Dirichlet-based data mixing across reasoning, code, and math benchmarks, with relative gains of 46% on MBPP and 14% on GSM8K. At 288 fine-grained domains, only vertex sampling supports a linear fit, and LINMIX improves benchmark accuracy by 11% on average over the best nonlinear baseline.

1 INTRODUCTION

The composition of a pretraining dataset, drawn from an ever-growing data corpus, is one of the most decisive factors governing the downstream capabilities of large foundation models (Gadre et al., 2023; Team Olmo, 2025). Given the impressive gains in model training efficiency and benchmark performance obtained by curating an optimal dataset, *data mixing* has emerged as a well-founded optimization problem. Naturally, a source dataset can be split into the so-called *domain* subsets based on their web origin (e.g., GitHub, arXiv, Fan et al., 2024), semantic type (e.g., code, math, Li et al., 2024), or statistical profile (e.g., by clustering data representations, Diao et al., 2025). Data mixing is then commonly formulated as a domain-weighting problem, where each domain receives a fraction of the target model's token budget and the model is pretrained on the resulting mixture (Xie et al., 2023; Li et al., 2024; Thudi et al., 2025; Xie et al., 2025; Tan et al., 2026).

In this work, we study *regression*-based data mixing (Liu et al., 2025; Wettig et al., 2025; Ye et al., 2025; Chen et al., 2026), which consists of two stages: (i) training a swarm of small *proxy models* on randomly sampled mixes, and (ii) selecting the target mix by optimizing a regression model fit to predict performance from the mix. This procedure improves both the benchmark performance and the training efficiency of language models, yet two of its design choices remain poorly understood and are addressed ad hoc in practice. First, it is unclear from which distribution the proxy mixes should be drawn, a question we frame as the classic problem of *regression design* (Kiefer & Wolfowitz, 1959), where the sampled mixes govern how much a fixed budget of proxy runs reveals about each domain's effect. Second, we show that popular methods do not scale to hundreds of fine-grained domains, where their Dirichlet sampling and optimization hyperparameters lack working defaults and require per-setup heuristic tuning, which becomes increasingly difficult as the number of domains increases.

Domain attribution. In parallel, data attribution quantifies how much a model's behavior depends on its training data (Koh & Liang, 2017; Ghorbani & Zou, 2019). We formalize regression-based data

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.

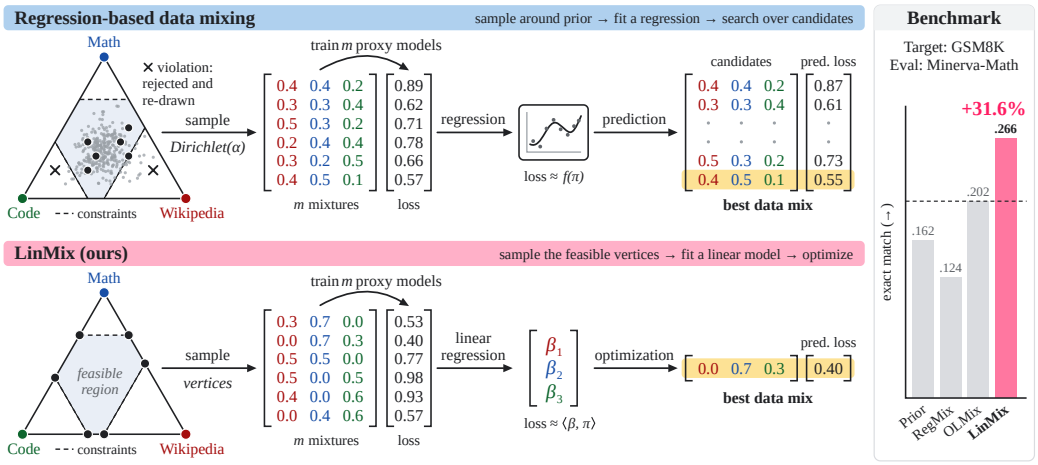


Figure 1: Data mixing for language model pretraining with linear domain attribution. We show that an interpretable linear regression model suffices for data mixing under vertex sampling.

mixing as a form of data attribution, in which domains replace individual training samples. A linear fit to the proxy runs thus yields a domain-level datamodel (Ilyas et al., 2022), whose coefficients measure each domain's marginal contribution to the target outcome. Consequently, a reliable regression design identifies these interpretable domain weights, an objective this work addresses from first principles.

Contributions. We introduce LINMIX as a simple and effective data mixing method whose vertex proxy sampling scheme enables selection of the optimal mix for language model pretraining via an interpretable linear model (§3). Figure 1 illustrates its design. Pretraining experiments with 1B-parameter OLMo2 and 4B Qwen3.5 language models demonstrate that LINMIX outperforms data mixing methods relying on Dirichlet sampling across reasoning, code, and math benchmarks (§4). Furthermore, our theoretical analysis reveals fundamental limitations of Dirichlet-based mixing (§5). Beyond requiring heuristic hyperparameter tuning, it severely inflates attribution error whenever proxy mixes concentrate around the prior or when the prior becomes skewed (Theorem 1). Finally, we formally establish the theoretical optimality of vertex sampling in LINMIX (Theorem 2).

2 BACKGROUND

Language models are pretrained on a corpus of n sequences partitioned into d domains $\mathcal{D}_1, \ldots, \mathcal{D}_d$ with $n_k := |\mathcal{D}_k|$ sequences each. The natural proportions $(\boldsymbol{\pi}_{\text{prior}})_k := n_k/n$ define the prior data mix, a point on the probability simplex Δ^{d-1}. Whereas training on the corpus as-is draws data according to $\boldsymbol{\pi}_{\text{prior}}$, data mixing seeks to reweight this prior into domain weights $\boldsymbol{\pi}_{\text{target}} \in \Delta^{d-1}$ that maximize the downstream performance of the *target* model (Xie et al., 2023; Fan et al., 2024; Albalak et al., 2024). However, evaluating even a single candidate mix at the target scale would require a full pretraining run over b_{target} sequences. Instead, data mixing techniques train a swarm of m small *proxy* models, each on $b_{\text{proxy}} \ll b_{\text{target}}$ sequences drawn according to a candidate mix $\boldsymbol{\pi}_i$, and use their performance to predict $\boldsymbol{\pi}_{\text{target}}$; the candidate mixes form the rows of the design matrix $\boldsymbol{\Pi} \in \mathbb{R}^{m \times d}$.

Sampling $\boldsymbol{\Pi}$. State-of-the-art approaches such as REGMIX (Liu et al., 2025) and OLMIX (Chen et al., 2026) populate the rows of $\boldsymbol{\Pi}$ via Dirichlet sampling, drawing mixes from a tempered version of the prior for each $i = 1, \ldots, m$:

$$\boldsymbol{\alpha}_i := \gamma_i \, \boldsymbol{\pi}_{\text{prior}}^{1/\tau} / \|\boldsymbol{\pi}_{\text{prior}}^{1/\tau}\|_1, \qquad \boldsymbol{\pi}_i \sim \text{Dirichlet}(\boldsymbol{\alpha}_i), \qquad \log \gamma_i \sim \text{Unif}(\log \gamma_{\text{lo}}, \log \gamma_{\text{hi}}). \quad (1)$$

The scale γ_i is the total concentration of the Dirichlet distribution, where small γ_i produces sparse mixes dominated by a few domains, and large γ_i tight perturbations of the tempered prior. The temperature τ flattens the prior so that dominant domains (e.g., CommonCrawl) do not absorb most of the probability mass, with $\tau = 2$ by default (Liu et al., 2025; Wettig et al., 2025).

Up-weighting a domain far beyond its prior share can only be realized at the target scale by repeating that domain's data, which brings diminishing returns beyond a few passes (Muennighoff et al., 2025).

Recent methods therefore restrict mixes to the truncated simplex $\mathcal{P}_{\text{cap}} \coloneqq \{\boldsymbol{\pi} \in \Delta^{d-1} \mid \pi_k \leq u_k \text{ for all } k\}$ with per-domain caps $u_k \coloneqq \min(1, \kappa\,(\boldsymbol{\pi}_{\text{prior}})_k)$ for an up-sampling factor $\kappa \geq 1$. A mix at the cap makes at most $\kappa \cdot b_{\text{target}}/n$ passes over any domain's data; in practice, Liu et al. (2025) hardcode $\kappa = 15$, Wettig et al. (2025) derive $\kappa \approx n/b_{\text{target}}$ so that no data repeats at the target model scale, and Chen et al. (2026) set $\kappa = 4n/b_{\text{target}}$, i.e., at most four passes.

Optimizing $\boldsymbol{\pi}_{\text{target}}$. Each proxy model i is trained on its mix $\boldsymbol{\pi}_i$ and evaluated to obtain outcome y_i, e.g., validation loss or bits-per-byte, yielding the proxy dataset $(\boldsymbol{\Pi}, \boldsymbol{y})$ with $\boldsymbol{y} \in \mathbb{R}^m$. Regression-based data mixing methods fit a parameterized model $f_\theta : \Delta^{d-1} \to \mathbb{R}$ to predict the outcome from the mix, and then predict the target mix by minimizing the fitted model over the truncated simplex:

$$\hat{\theta} \in \arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} \left(y_i - f_\theta(\boldsymbol{\pi}_i)\right)^2, \qquad \hat{\boldsymbol{\pi}}_{\text{target}} \in \arg\min_{\boldsymbol{\pi} \in \mathcal{P}_{\text{cap}}} f_{\hat{\theta}}(\boldsymbol{\pi}). \tag{2}$$

In practice, REGMIX solves this optimization approximately, evaluating a gradient-boosted $f_{\hat{\theta}}$ on a large pool of Dirichlet-sampled candidate mixes and averaging the Top-K best-predicted ones for robustness (Liu et al., 2025). OLMIX (Chen et al., 2026) adopts the exponential mixing law of Ye et al. (2025) as $f_\theta(\boldsymbol{\pi}) = \exp\left(\sum_{j=1}^{d} \theta_j \pi_j\right) + \theta_0$, which is convex in $\boldsymbol{\pi}$ and can hence be minimized exactly over $\mathcal{P}_{\text{cap}}$. Finally, $\mathcal{D}_{\text{target}}$ is constructed by drawing $\hat{\pi}_{\text{target},k} \cdot b_{\text{target}}$ sequences from each domain $\mathcal{D}_k$ and the target model is trained on this dataset. Note that every step of this approach inherits its quality from the design $\boldsymbol{\Pi}$, which determines how well θ can be identified from only m runs; yet how to sample $\boldsymbol{\Pi}$ has received little attention, which we address in this paper.

3 DATA MIXING AS LINEAR DOMAIN ATTRIBUTION

We formalize data mixing as a *domain attribution* problem seeking an interpretable linear model whose coefficients measure each domain's contribution to the target outcome. We empirically find that such a linear fit cannot be supported using Dirichlet-sampled proxy runs, and prior work compensates by using nonlinear surrogates and per-setup hyperparameter tuning, which we report in Section 4. Instead of changing the surrogate model, we change the design by sampling $\boldsymbol{\Pi}$ to maximize the information the proxy runs carry about the domain effects, after which nonlinear models offer no better fit than the linear one. Reliable attribution in turn enables the constrained optimization of $\boldsymbol{\pi}_{\text{target}}$, where Dirichlet-based methods degrade once d reaches hundreds of fine-grained domains.

Sampling $\boldsymbol{\Pi}$. Unlike prior work, we sample the m mixes at the *vertices* of $\mathcal{P}_{\text{cap}}$, aiming to maximize their spread measured by $\det(\boldsymbol{\Pi}^\top \boldsymbol{\Pi})$ without sampling hyperparameters. This choice follows the D-optimality principle of experimental design (Kiefer & Wolfowitz, 1959), which we elaborate on in Appendix A. For each run i, we draw a uniform random permutation σ_i of domains $\{1, \ldots, d\}$ and fill domains greedily in that order, saturating each at its cap until the unit budget is exhausted:

$$\pi_{i,\sigma_i(k)} \coloneqq \min\Big(u_{\sigma_i(k)},\ 1 - \sum_{\ell<k} \pi_{i,\sigma_i(\ell)}\Big) \quad \text{for } k = 1, \ldots, d. \qquad \text{(Greedy fill)} \tag{3}$$

Each resulting $\boldsymbol{\pi}_i$ has at most one strictly interior coordinate, with the rest pinned to u_k or 0. Such a sparse design is near-orthogonal in expectation, rather than centered at $\boldsymbol{\pi}_{\text{prior}}$, as in Dirichlet sampling. To further improve $\det(\boldsymbol{\Pi}^\top \boldsymbol{\Pi})$, we pair the proxy runs, letting run $2i$ reuse the permutation of run $2i-1$ in reverse, i.e. $\sigma_{2i}(k) \coloneqq \sigma_{2i-1}(d-k+1)$, so that domains left empty by one fill are saturated first by its counterpart. The paired fills are anti-correlated by construction, with largely disjoint supports, and thus contribute near-orthogonal directions to $\boldsymbol{\Pi}$, keeping $\boldsymbol{\Pi}^\top \boldsymbol{\Pi}$ well conditioned.

Optimizing $\boldsymbol{\pi}_{\text{target}}$. Fitting the linear model $f_{\boldsymbol{\beta}}(\boldsymbol{\pi}) = \boldsymbol{\beta}^\top \boldsymbol{\pi}$ in Eq. (2) by least squares yields the attribution estimate $\hat{\boldsymbol{\beta}} = (\boldsymbol{\Pi}^\top \boldsymbol{\Pi})^{-1} \boldsymbol{\Pi}^\top \boldsymbol{y}$, whose covariance scales with $(\boldsymbol{\Pi}^\top \boldsymbol{\Pi})^{-1}$. We always center $\hat{\boldsymbol{\beta}}$ to zero mean so that $\hat{\beta}_k$ measures the marginal value of domain k relative to the average. We also negate $\boldsymbol{y}$ for target metrics where lower is better, so that a positive $\hat{\beta}_k$ indicates a domain of above-average value. The mix selection then becomes a linear program over the truncated simplex,

$$\hat{\boldsymbol{\pi}}_{\text{target}} \in \arg\max_{\boldsymbol{\pi} \in \mathcal{P}_{\text{cap}}} \langle \hat{\boldsymbol{\beta}}, \boldsymbol{\pi} \rangle, \tag{4}$$

solved by filling domains to their caps in decreasing order of $\hat{\beta}_k$ until the budget is exhausted, so the solution is itself a vertex of $\mathcal{P}_{\text{cap}}$. This allocation inherently treats each additional domain sequence as equally valuable, which is appropriate for most cases, especially quality-filtered data.

Transforming the utility of π_k. Beyond sampling $\mathbf{\Pi}$ and optimizing $\boldsymbol{\pi}_{\text{target}}$, a third design choice worth studying is the utility assigned to each domain weight, which can capture the diminishing returns of repeating data from a given domain. For example, if we assume that data value decays with domain repetition, the linear utility π_k can be transformed to the logarithmic one $\ln \pi_k$, so that each additional sequence of a domain is worth less than the previous one. Theoretically, these two extremes can be framed as special cases of the general isoelastic objective

$$\hat{\boldsymbol{\pi}}_{\text{target}} \in \arg\max_{\boldsymbol{\pi} \in \mathcal{P}_{\text{cap}}} \sum_{k=1}^{d} \hat{\beta}_k \frac{\pi_k^{1-r} - 1}{1-r}, \qquad \text{for curvature } r \in [0, 1], \tag{5}$$

which recovers the linear objective in Eq. (4) as $r \to 0$ and the log-linear form $\sum_{k:\hat{\beta}_k > 0} \hat{\beta}_k \ln \pi_k$ at $r = 1$. Proposition 1 establishes that positive domain attribution determines the support of the optimal mix, while the curvature r only allocates the budget within it (see an example in App. Figure 10). We discuss the classical roots of this theory and its closest data selection counterpart in Appendix A.

Proposition 1 (Domain support and uniqueness of the optimal mix). *Let $\mathcal{A} := \{k : \hat{\beta}_k > 0\}$ be the set of domains with positive estimated attribution, and assume their caps can absorb the whole budget, i.e. $u_{\mathcal{A}} := \sum_{k \in \mathcal{A}} u_k > 1$. At $r = 1$, the objective of Eq. (5) reads as $\sum_{k \in \mathcal{A}} \hat{\beta}_k \ln \pi_k$, its log-linear form with the sum restricted to $\mathcal{A}$. Then, for every $r \in (0, 1]$, its maximizer $\boldsymbol{\pi}^\star$ is unique and satisfies $\pi_k^\star > 0$ for all $k \in \mathcal{A}$ and $\pi_k^\star = 0$ for all $k \notin \mathcal{A}$. At $r = 0$, in contrast, a maximizer may assign zero weight to domains in $\mathcal{A}$. Proofs are given in Appendix B.*

Corollary 1 (Closed form of the optimal mix). *Under the assumptions of Proposition 1 and for $r \in (0, 1]$, the objective of Eq. (5) restricted to its positively attributed support $\mathcal{A}$ is strictly concave. Then, the maximizer takes the closed form $\hat{\pi}_{target,k} = \min\left(u_k, (\hat{\beta}_k/\omega)^{1/r}\right)$ for $k \in \mathcal{A}$ and $\hat{\pi}_{target,k} = 0$ otherwise, where $\omega > 0$ is the unique scalar enforcing $\sum_k \hat{\pi}_{target,k} = 1$.*

As $r \to 0$, the solution in Corollary 1 collapses to the greedy fill of Eq. (3) with domains ordered by decreasing $\hat{\beta}_k$ (for distinct $\hat{\beta}_k$). Optimal mixes thus saturate the caps of their top domains, such as our sampled vertices, and predicting with $\hat{\boldsymbol{\pi}}_{\text{target}}$ interpolates rather than extrapolates. Unlike in Dirichlet sampling, optima here do not fall outside the convex hull of mixes clustered around the prior.

4 Experiments

Data. We use the 200B-token WebOrganizer corpus (Wettig et al., 2025) built from CommonCrawl web data and cleaned for language model pretraining. Each document is assigned one of 24 topics (e.g., science, politics) and one of 24 formats (e.g., academic writing, news), and we split the corpus into 2048-token sequences, each drawn from a single domain. We study two mixing granularities: 24 coarse domains given by the topics alone, and 288 fine-grained domains formed by the larger half of the 24×24 topic–format pairs, covering 94% of the corpus (cf. App. Figure 6). Removing the practically empty tail from the prior data mix reduces noise when estimating domain weights.

Language models. Following Chen et al. (2026), we build on the open source OLMo 2 implementation (Team OLMo et al., 2025) to train 30M and 60M proxy models and 1B target models. All models are trained past the 20 tokens-per-parameter Chinchilla ratio (Hoffmann et al., 2022) to avoid undertraining (Chen et al., 2025b), on 1B, 2B, and 32B tokens, respectively. This setup is comparable to prior work that pairs 27M (Liu et al., 2025) and 50M (Wettig et al., 2025) proxies with 1B targets. Following Wettig et al. (2025), we set the up-sampling cap to $\kappa \approx n/b_{\text{target}}$ so that no sequence repeats at the target scale, a non-arbitrary choice shared by all sampling methods we compare. Hyperparameters and further training details are given in Appendix C.

Target benchmarks. We optimize data mixes for three targets: ARC (easy and challenge, Clark et al., 2018) for scientific question answering; HellaSwag (Zellers et al., 2019) for commonsense reasoning; and HellaSwag combined with PIQA (Bisk et al., 2020) as a two-benchmark target. In each case, we optimize the mix against proxy-model bits-per-byte (BPB, Gao et al., 2021) metric on the benchmark's training split and report the target model's accuracy on a held-out set: test for ARC, and validation for HellaSwag and PIQA, whose test labels are withheld. We also optimize for BPB validation loss on the Common Pile (CPile, Kandpal et al., 2025) data corpus to diversify the setup.

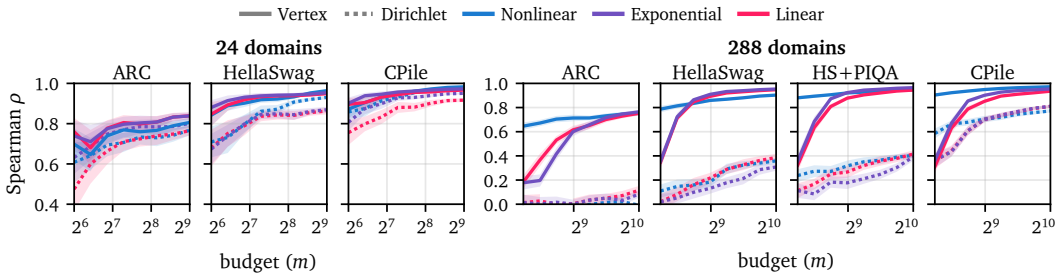


Figure 3: Optimizer performance under the vertex and Dirichlet samplers. Spearman correlation is averaged across 5 random repetitions of 10-fold cross-validation under a fixed proxy model budget. Nonlinear is implemented using LightGBM, as in RegMix, and exponential fits the exponential law to the target, as in Olmix. The corresponding R^2 values are reported in App. Figure 7.

4.1 Dirichlet sampling of proxy data mixes requires tuning hyperparameters

Before training proxy models, we need to sample the design matrix $\mathbf{\Pi}$. Figure 2 shows the fraction of Dirichlet-sampled domain mixes that violate the cap constraint of $\mathcal{P}_{\text{cap}}$ and must be rejected. For 288 domains, the baseline algorithm does not converge with the default hyperparameters $\tau = 2, \gamma_{\text{lo}} = 0.1, \gamma_{\text{hi}} = 5$, or even $\gamma_{\text{hi}} = 10$ (cf. the hatched area in Figure 2). Since related work gives no guidance on how to tune them, we set $\tau = 2$ and propose to select γ by minimizing the condition number of $\mathbf{\Pi}$, which yields mixes much closer to the prior at $\gamma_{\text{lo}} = 40$ and $\gamma_{\text{hi}} = 4000$.

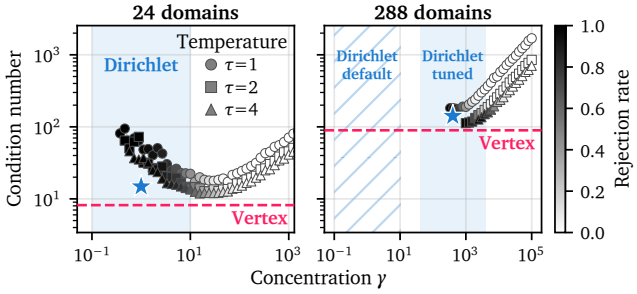


Figure 2: Dirichlet sampling with default γ, τ hyperparameters does not converge for 288 domains. Fraction of Dirichlet-sampled domain mixes that violate the target model cap constraint. We choose new Dirichlet hyperparameters by minimizing the condition number of matrix $\mathbf{\Pi}$ (blue star).

4.2 Vertex sampling outperforms Dirichlet, enabling a linear model fit

We train $512\times$ 30M and $1024\times$ 60M proxy models per sampler for 24 and 288 domains, respectively. Figure 3 shows the cross-validated fit between predicted and observed benchmark BPB (Spearman ρ) under each design as the number of proxy models grows. At 24 domains, every surrogate fits well under both samplers, but at 288 domains, no surrogate helps Dirichlet sampling even when γ is tuned ($\rho \leq 0.39$ at $m = 1024$). Conversely, our hyperparameter-free vertex sampling puts the interpretable linear model on par with the best nonlinear alternative (see also R^2 in App. Figure 7). Half of the vertex swarm already fits better than the full Dirichlet swarm, because near-prior Dirichlet mixes vary HellaSwag BPB five times less than vertex mixes, leaving little signal to regress on. As an ablation in App. Figure 9, we train 30M and 100M models on 288 domains, fit a linear model from the 30M swarm, and use it to rank the outcomes of 60M and 100M models at $\rho \geq 0.95$, while the Dirichlet fit does not transfer ($\rho \leq 0.42$). We visualize the optimized domain weights in App. Figure 10, where relaxing curvature r retains the full positively attributed support (Prop. 1). Based on it, we transform the utility on 24 domains with $\ln \pi$ for HellaSwag, which adds Social Life, Fashion & Beauty, Games, and Industrial to the four domains at their caps (Health, Home & Hobbies, Food & Dining, Adult); and with $\sqrt{\pi}$ for CPile, which adds a sixth Hardware domain to the data mix.

4.3 LinMix outperforms data mixing baselines on a 1B target model benchmark

We compare LinMix to a nonlinear RegMix, which outperforms several data selection methods and was applied in WebOrganizer, as well as Olmix, which benchmarks several data mixing approaches and finds the exponential model to be state-of-the-art. Table 1 shows target benchmark performance where LinMix with a linear model significantly improves data mixing with 288 domains, and is

Table 1: Performance comparison of data mixing methods across 23× 1B par. OLMo2 target models pretrained on 32B tokens. Normalized accuracy and bits-per-byte (BPB) are measured on a held-out set per target benchmark. Color denotes relative difference to the second-best result.

	24 domains (512× 30M models)					**288 domains** (1024× 60M proxy models)						
	ARC		HellaSwag		CPile	ARC		HellaSwag		HS + PIQA		CPile
Method	Acc↑	BPB↓	Acc↑	BPB↓	BPB↓	Acc↑	BPB↓	Acc↑	BPB↓	Acc↑	BPB↓	BPB↓
Prior	.437	.864	.543	.788	.888	.436	.870	.541	.788	.634	.902	.890
RegMix	.496	**.780**	.563	**.781**	**.842**	.449	.850	.547	.784	.638	.896	.878
OLMix	.486	.785	.550	.782	.860	.435	.872	.550	.783	.641	.892	.850
LinMix	**.510**	.785	**.568**	.785	.847	**.511**	**.757**	**.615**	**.758**	**.690**	**.855**	**.834**
	+2.8%	−0.6%	+0.9%	−0.5%	−0.6%	+13.8%	+10.9%	+11.8%	+3.2%	+7.6%	+4.1%	+1.9%

on par or better with 24 domains. This result is consistent with the proxy metric values reported in Section 4.2, which here translate into downstream performance. An interpretable linear regression suffices for data mixing that targets the CPile validation loss under vertex sampling, i.e. it is on par with or better than complex Dirichlet sampling with nonlinear regression. We ablate on two optimization factors in Appendix D: Table 5 indicates that scaling the linear model coefficients by standard error could improve performance, and Table 6 demonstrates how transforming the utility of π towards $\ln \pi$ relaxes the 24-domain support for HellaSwag (Proposition 1), yielding better results.

4.4 Scaling to a 4B Qwen3.5 model benchmarked on code and math capabilities

We demonstrate the broader applicability of LinMix by doubling the data corpus and training a target model with ten times more compute. The 24-topic WebOrganizer dataset is extended with Nemotron-CC-Math (Karimi Mahabadi et al., 2026) and Stack-Edu (Allal et al., 2025) split into 15 programming languages, yielding a 416B-token corpus spanning 40 domains. We pretrain 4B Qwen3.5 (Qwen Team, 2026) target models on 80B tokens at a 20× ratio to allow for more diverse experiments. We train 512× 100M proxy models per sampler to prevent under-parametrization, given the more challenging coding and math capabilities. Mixes are optimized towards training splits of HellaSwag, MBPP (Austin et al., 2021), and GSM8K (Cobbe et al., 2021). Figure 4 shows an example of optimized domain weights targeting three diverse model capabilities. It illustrates that RegMix obtains similar mixes across the three target capabilities, while the exponential law of OLMix differentiates between targets, but to a lesser degree than LinMix. Table 2 shows that LinMix delivers strong performance across the language, coding, and math benchmarks, transferring these capabilities to held-out benchmarks such as HumanEval (Chen et al., 2021) and MATH (Hendrycks et al., 2021; Lewkowycz et al., 2022). We additionally train a 1B-parameter Qwen3.5-like model on 32B tokens targeting CPile from the same proxy model swarm and report its results in the last column.

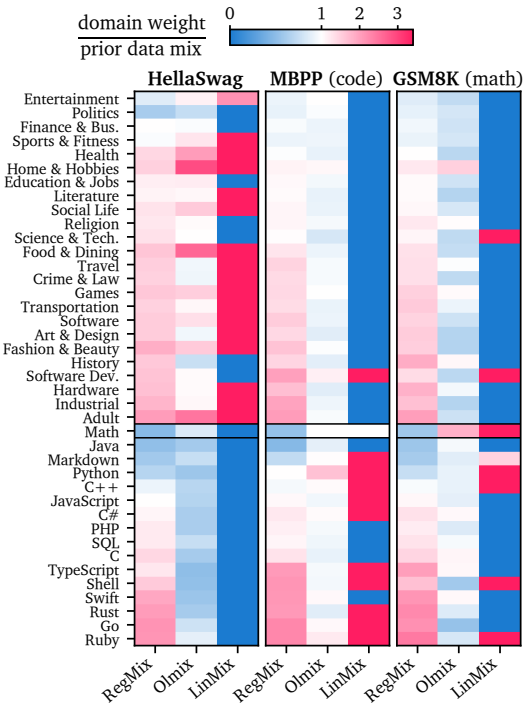


Figure 4: Domain weights relative to the prior mix across data mixing methods and target benchmarks. LinMix downweights math and code domains for language modeling (blue in HellaSwag), upweights programming languages and software domains for coding (red in MBPP), as well as math and science domains for math capabilities (red in GSM8K).

Table 2: Performance of data mixing methods across 10× 4B Qwen3.5 target models pretrained on 80B tokens. Mixes are optimized with 40 domains for each target benchmark and evaluated on its held-out set and a held-out benchmark. The additional CPile column reports 4× 1B Qwen3.5 models optimized for held-out Common Pile validation loss. Color is a relative difference to the second-best.

	Target: HellaSwag				**Target: MBPP** (code)				**Target: GSM8K** (math)				**CPile**
	HellaSwag		PIQA		MBPP		HumanEval		GSM8K		MATH		CPile
Method	Acc↑	BPB↓	Acc↑	BPB↓	P@1↑	BPB↓	P@1↑	BPB↓	EM↑	BPB↓	EM↑	BPB↓	BPB↓
Prior	.632	.743	.764	.952	.304	.375	.213	.285	.214	.403	.162	.463	.814
RegMix	.659	.735	.775	.933	.304	.379	.213	.290	.135	.418	.124	.487	.808
Olmix	.667	.731	**.783**	.932	.302	.368	.244	.277	.294	.391	.202	.446	.801
LinMix	**.696**	**.716**	.782	**.920**	**.444**	**.334**	**.280**	**.258**	**.337**	**.383**	**.266**	**.427**	**.797**
	+4.4%	+2.0%	−0.1%	+1.3%	+46.0%	+9.2%	+15.0%	+7.0%	+14.4%	+2.1%	+31.6%	+4.3%	+0.5%

5 Theoretical Analysis

Here, we formalize the domain attribution view of data mixing from first principles, laying the foundations for LinMix. The complete analysis with proofs is deferred to Appendix E. Datamodeling (Ilyas et al., 2022) attributes model outcomes to training data counterfactually, in a regime where many training runs on random subsets of the data are feasible. Let's fix a learning algorithm and a target input $\boldsymbol{z}$, e.g. a benchmark question, and let $y(S; \boldsymbol{z})$ denote the loss on $\boldsymbol{z}$ of a model trained on a subset S of the corpus $\mathcal{D} \coloneqq \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_d$ of n sequences. A datamodel is fit to the pairs $(\mathbf{1}_{S_i}, y(S_i; \boldsymbol{z}))$ collected over runs $i = 1, \ldots, m$, where $\mathbf{1}_{S_i} \in \{0,1\}^n$ indicates the sequences included in run i. Empirically, a linear fit predicts the outcomes of held-out runs well and motivates the model

$$y(S; \boldsymbol{z}) = \langle \boldsymbol{\beta}(\boldsymbol{z}), \mathbf{1}_S \rangle + \eta(\boldsymbol{z}), \qquad \text{(Datamodel)} \tag{6}$$

where $\boldsymbol{\beta}(\boldsymbol{z}) \in \mathbb{R}^n$ collects the attribution scores, i.e., the marginal effect of including each sequence on the loss at $\boldsymbol{z}$, and $\eta(\boldsymbol{z})$ is a noise term with zero mean given S that captures the randomness of training. Follow-up work relies on this local linearity to relate attribution scores to model outcomes (Karchmer et al., 2025), and we adopt it as the assumption underlying this analysis. As all subsets considered below have the same size, an intercept can be absorbed into $\boldsymbol{\beta}(\boldsymbol{z})$ and is omitted.

Consider a proxy run on the mix $\boldsymbol{\pi}$ that trains on $b \coloneqq b_{\text{proxy}}$ sequences, of which $b\pi_k$ are drawn uniformly without replacement from domain $\mathcal{D}_k$, independently across domains, where we assume that $b\pi_k \leq n_k$ is an integer. Let its outcome y be the benchmark loss, i.e., the average of $y(S; \boldsymbol{z})$ over the set of benchmark inputs $\mathcal{Z}$. Let $\boldsymbol{\beta}^{\mathcal{Z}} \coloneqq \frac{1}{|\mathcal{Z}|} \sum_{\boldsymbol{z} \in \mathcal{Z}} \boldsymbol{\beta}(\boldsymbol{z})$ denote the benchmark-averaged attribution scores, and let μ_k and $\sigma_k^2 \coloneqq \frac{1}{n_k} \sum_{s \in \mathcal{D}_k} (\beta_s^{\mathcal{Z}} - \mu_k)^2$ denote their mean and variance over the sequences of domain k. Proposition 2 formalizes the data attribution view of data mixing.

Proposition 2 (Datamodels coarsen to domain attribution). *Under the datamodel of Eq. (6), the expected outcome of a proxy run is linear in its mix. Specifically, with $\bar{\beta}_k \coloneqq b\,\mu_k$,*

$$\mathbb{E}[y \mid \boldsymbol{\pi}] = \langle \bar{\boldsymbol{\beta}}, \boldsymbol{\pi} \rangle \qquad \textit{and} \qquad \mathrm{Var}(y \mid \boldsymbol{\pi}) \leq \sigma_\eta^2 + b \max_k \sigma_k^2, \tag{7}$$

where σ_η^2 bounds the variance of the benchmark-averaged noise $\frac{1}{|\mathcal{Z}|} \sum_{\boldsymbol{z} \in \mathcal{Z}} \eta(\boldsymbol{z})$ given S.

The linear model $f_{\boldsymbol{\beta}}(\boldsymbol{\pi}) = \boldsymbol{\beta}^\top \boldsymbol{\pi}$ fit in Eq. (2) is thus a datamodel whose units are domains, with coefficients that average sample-level attribution scores. The partition into domains is arbitrary and affects only the noise, which stems from training randomness and from which sequences realize the mix. Finer, more homogeneous domains reduce the noise from sequence selection but take more proxy runs to estimate, because the estimation error grows with the number of coefficients d (Lem. 1).

Corollary 2 (Domain attribution at the finest granularity is data attribution). *Let every sequence form its own domain, $d = n$, so that a run trained on a subset S of b sequences has the mix $\boldsymbol{\pi} = \mathbf{1}_S / b$. Then $\bar{\boldsymbol{\beta}} = b\,\boldsymbol{\beta}^{\mathcal{Z}}$, so the domain model of Proposition 2 reduces to the benchmark-averaged datamodel of Eq. (6): $\langle \bar{\boldsymbol{\beta}}, \boldsymbol{\pi} \rangle = \langle \boldsymbol{\beta}^{\mathcal{Z}}, \mathbf{1}_S \rangle$. The noise from sequence selection vanishes,*

leaving $\mathrm{Var}(y \mid \boldsymbol{\pi}) \le \sigma_\eta^2$. *Further, with the equal caps* $u_k = 1/b$ ($\kappa = n/b$) *that let no sequence repeat within a run, the greedy fill of Eq. (3) selects a uniformly random subset of* b *sequences, so vertex sampling coincides with the subsampling design of Ilyas et al. (2022) with fraction* b/n.

Domain attribution thus interpolates between data attribution and a single-domain corpus, preserving the linear structure in between. We further relate it to the Shapley and Banzhaf data values of cooperative game theory in App. Remark 1. In practice, $\bar{\boldsymbol{\beta}}$ is unknown and must be estimated from the m proxy runs. By Proposition 2, their outcomes satisfy $\boldsymbol{y} = \boldsymbol{\Pi}\bar{\boldsymbol{\beta}} + \boldsymbol{\xi}$ for a noise vector $\boldsymbol{\xi} \in \mathbb{R}^m$, so the LinMix estimate $\hat{\boldsymbol{\beta}} = (\boldsymbol{\Pi}^\top \boldsymbol{\Pi})^{-1} \boldsymbol{\Pi}^\top \boldsymbol{y}$ is the least squares estimator of $\bar{\boldsymbol{\beta}}$. Its error depends on the noise level and on the information matrix $\boldsymbol{H}_{\boldsymbol{\Pi}} := \frac{1}{m} \boldsymbol{\Pi}^\top \boldsymbol{\Pi}$.

Lemma 1 (Estimation error of linear domain attribution). *Let the noise have zero mean and covariance between* $v_{\min} \boldsymbol{I}_m$ *and* $v_{\max} \boldsymbol{I}_m$ *given* $\boldsymbol{\Pi}$, *and let* $\boldsymbol{\Pi}$ *have full column rank. Then* $\hat{\boldsymbol{\beta}}$ *is unbiased and its mean squared error satisfies*

$$v_{\min} \operatorname{tr}\big((\boldsymbol{\Pi}^\top \boldsymbol{\Pi})^{-1}\big) \;\le\; \mathbb{E}\big[\|\hat{\boldsymbol{\beta}} - \bar{\boldsymbol{\beta}}\|_2^2 \mid \boldsymbol{\Pi}\big] \;\le\; v_{\max} \operatorname{tr}\big((\boldsymbol{\Pi}^\top \boldsymbol{\Pi})^{-1}\big) \;\le\; \frac{d}{m} \cdot \frac{v_{\max}}{\lambda_{\min}(\boldsymbol{H}_{\boldsymbol{\Pi}})}, \tag{8}$$

with equality in the first two when all runs share the same noise variance, $v_{\min} = v_{\max} = \sigma^2$.

The error is thus governed by the smallest eigenvalue of $\boldsymbol{H}_{\boldsymbol{\Pi}}$, which is small when some combination of domains barely varies across the proxy runs. Crucially, since $\lambda_{\max}(\boldsymbol{H}_{\boldsymbol{\Pi}}) \ge 1/d$ on the simplex, we have $1/\lambda_{\min}(\boldsymbol{H}_{\boldsymbol{\Pi}}) \le d \operatorname{cond}(\boldsymbol{\Pi})^2$, so minimizing the condition number of $\boldsymbol{\Pi}$ as in Section 4.1 also controls this bound. The other quantities in the bound, the noise level $v_{\max}$ and the number of domains d, are fixed by the data and its partition. A data mixing method thus controls only the number of proxy runs m and the design $\boldsymbol{\Pi}$. We next compare the impact of the designs of Dirichlet and vertex sampling on the bound of Lemma 1.

5.1 Pitfalls of Dirichlet sampling

Dirichlet sampling in Eq. (1) draws each mix as $\boldsymbol{\pi}_i \sim \mathrm{Dirichlet}(\gamma_i\, \bar{\boldsymbol{a}})$ around the tempered prior $\bar{\boldsymbol{a}} := \boldsymbol{\pi}_{\text{prior}}^{1/\tau} / \|\boldsymbol{\pi}_{\text{prior}}^{1/\tau}\|_1$, with a log-uniform concentration $\gamma_i \in [\gamma_{\text{lo}}, \gamma_{\text{hi}}]$. We analyze it without caps or rounding of $b\pi_k$ to integers, starting with its expected information matrix $\bar{\boldsymbol{H}} = \mathbb{E}[\boldsymbol{H}_{\boldsymbol{\Pi}}]$.

Lemma 2 (Information matrix of Dirichlet sampling). *Under Eq. (1),*

$$\bar{\boldsymbol{H}} := \mathbb{E}[\boldsymbol{\pi}_i \boldsymbol{\pi}_i^\top] = c \operatorname{diag}(\bar{\boldsymbol{a}}) + (1-c)\, \bar{\boldsymbol{a}} \bar{\boldsymbol{a}}^\top, \quad c := 1 - \frac{\ln\big((1+\gamma_{\text{hi}})/(1+\gamma_{\text{lo}})\big)}{\ln(\gamma_{\text{hi}}/\gamma_{\text{lo}})} \in (0,1). \tag{9}$$

Consequently, $\operatorname{tr}(\bar{\boldsymbol{H}}^{-1}) = \frac{1}{c}\big(\sum_{k=1}^d \bar{a}_k^{-1} - (1-c)\, d\big)$ *and* $c\, \bar{a}_{\min} \le \lambda_{\min}(\bar{\boldsymbol{H}}) \le c\, \bar{a}_{(2)}$, *where* $\bar{a}_{\min}$ *and* $\bar{a}_{(2)}$ *denote the smallest and the second smallest entry of* $\bar{\boldsymbol{a}}$.

By Lemma 2, $\bar{\boldsymbol{H}}$ averages two designs with weights c and $1-c$: single-domain runs at $\boldsymbol{e}_k$ drawn with probability $\bar{a}_k$, and the tempered prior repeated in every run. These two terms expose two pitfalls of Dirichlet sampling as illustrated in App. Figure 12. (i) *Under-dispersion*: a large concentration keeps mixes near the tempered prior, so c is small and the information in every direction orthogonal to $\bar{\boldsymbol{a}}$ shrinks by the factor c. (ii) *Tail starvation*: even at $c \to 1$, rare domains with a small share $\bar{a}_k$ are seldom explored, so their attributions are estimated poorly and dominate the error. To quantify the second pitfall, we model the prior as Zipfian (Zipf, 1949), $(\boldsymbol{\pi}_{\text{prior}})_k \propto k^{-s}$ for domains ranked by size, which holds for the prior data mixes in our experimental setups (Figure 5, 40 domains). The tempered prior is then also Zipfian, $\bar{a}_k = k^{-p}/\zeta_d(p)$, with the flatter exponent $p := s/\tau$ and the normalizer $\zeta_d(p) := \sum_{k=1}^d k^{-p}$. Its rarest domain has a share of only $d^{-p}/\zeta_d(p)$, so the estimation error grows with the number of domains, as Theorem 1 shows.

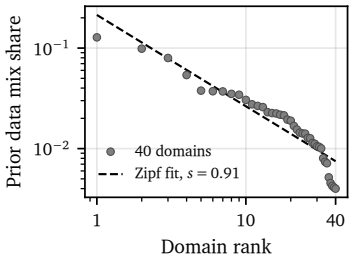


Figure 5: The prior data mix is approximately Zipfian. See also 24/288 domains in App. Fig. 13.

Theorem 1 (Estimation error of Dirichlet sampling; informal**).** *Let the prior be Zipfian with exponent* s; *let the mixes be Dirichlet-sampled as in Eq. (1), and let* $p = s/\tau$. *Let* $c \in (0,1)$ *be as*

in Lemma 2, so that $c \to 0$ as the mixes concentrate around the prior. Then the estimation error of $\hat{\boldsymbol{\beta}}$ is of order $\frac{\sigma^2}{mc}\, d^2$ for $p < 1$ and of order $\frac{\sigma^2}{mc}\, d^{1+p}$ for $p > 1$, both as an upper bound that holds with high probability and as a lower bound in expectation. Relative to the optimal error $\sigma^2 d^2/m$ of Prop. 3, Dirichlet sampling thus pays a factor $1/c$ for staying close to the prior, and a further factor polynomial in d once the prior is more skewed than the temperature τ can flatten.

Theorem 1 explains how the hyperparameters of Dirichlet sampling enter the estimation error and how they interact to break it as the number of domains grows. Tail starvation is negligible in our setups, whose Zipf exponents of $s \lesssim 0.9$ (App. Fig. 13) are flattened by $\tau = 2$ to $p < 1$. However, under-dispersion is unavoidable with many domains, because the sparse mixes of a small γ violate the caps, which force the tuned ranges of Sec. 4.1 with $c < 0.01$ versus $c = 0.5$ at 24 domains. The expected error of uncapped Dirichlet sampling then becomes at least 230 and 540 times the uncapped optimum $\sigma^2 d^2/m$ at 288 and 40 domains, respectively. This lower bound is 4 and 30 times larger than that of vertex sampling under an equal-cap approximation (cf. Theorem 2), using the trace formula of Lem. 2 with our empirical priors and γ ranges. This matches Fig. 3, where both samplers lead to reliable fits at 24 domains, but no surrogate fits the Dirichlet design well at 288 domains.

5.2 On optimal sampling

We turn to the vertex design of LinMix. On the full simplex, the vertices are the pure-domain mixes $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d$, and the greedy fill of Eq. (3) places each run at a uniformly random vertex. A balanced version of it allocates m/d runs to each vertex and is the classical optimal design for first-degree mixture models (Kiefer, 1961), which we state here with its finite-sample estimation error.

Proposition 3 (**Optimality of the balanced vertex design on the simplex**; informal)**.** *Place m/d proxy runs at each pure-domain mix $\boldsymbol{e}_k$, so that every run trains on a single domain and every domain is measured equally often. Then $\boldsymbol{\Pi}^\top \boldsymbol{\Pi} = \frac{m}{d} \boldsymbol{I}_d$, each $\hat{\beta}_k$ is the average outcome of the runs on domain k, and the estimation error is $\sigma^2 d^2/m$. No other design of m mixes on the simplex has a smaller estimation error, or a smaller worst-case prediction variance.*

Proposition 3 formalizes the intuition behind vertex sampling, where measuring one domain at a time decorrelates the domain attributions, and measuring every domain equally often equalizes their information. Each $\hat{\beta}_k$ thus has variance at most $v_{\max} d/m$ and, for independent runs, no covariance with the others. For optimizing $\boldsymbol{\pi}_{\text{target}}$, the predicted value $\langle \hat{\boldsymbol{\beta}}, \boldsymbol{\pi} \rangle$ of any mix $\boldsymbol{\pi}$ then has variance at most $v_{\max} d/m$, which no other design can improve in the worst case. In practice, the caps are smaller than one, so a run cannot train on a single domain, and LinMix samples the vertices of $\mathcal{P}_{\text{cap}}$. Theorem 2 shows that its expected information matrix is optimal on the simplex for a uniform prior.

Theorem 2 (**Vertex sampling is optimal on the capped simplex**; informal)**.** *Let the prior be uniform, so that all caps equal κ/d, every vertex of $\mathcal{P}_{cap}$ saturates $q = d/\kappa$ domains, and the greedy fill picks these uniformly at random. The optimal estimation error is about $\frac{d}{\kappa - 1}$ times the uncapped optimum of Proposition 3, the unavoidable error any design on $\mathcal{P}_{cap}$ adds for spreading runs over q domains. Once m exceeds a multiple of $d \ln d$, a sampled design comes within a constant factor of this optimum with high probability.*

Theorems 1 and 2 explain why the linear fit of LinMix is reliable where Dirichlet sampling is not. Dirichlet mixes concentrate around the prior, while vertices carry as much evenly spread information as caps allow. Extending Theorem 2 to non-uniform priors with unequal caps is left for future work.

6 Conclusion

We demonstrate that an interpretable linear model suffices for data mixing under vertex sampling and validate it across language model pretraining benchmarks. We further develop the theoretical foundations of LinMix and relate data mixing to the attribution literature. While this paper contributes to the interpretability of data mixing, our results suggest that optimizing a vertex-sampled matrix with a nonlinear model could yield even better target model performance, which we leave for future work. A limitation shared by all regression-based methods is the cost of training a proxy swarm, which we quantify here by scaling proxy model size and budget. In practice, the cost-to-performance trade-off of data mixing becomes only more favorable as the target model size and pretraining budget increase.

AI Use Statement

In this work, we used generative AI tools to code (implement methods, visualizations), improve writing (rephrase sentences, remove words), prove mathematical claims, and discover classic literature on experimental design (1960). We also used AI to provide adversarial feedback of our methodology and experimental results, which we have often disagreed with. Finally, we used AI to improve the paper's title, and it has been useful in monitoring experimental job logs. We have not used generative AI tools to clean and reformat datasets, propose hypotheses or conceptual frameworks, suggest structure for a paper, format references, or interpret results. Otherwise, generating synthetic data, supporting qualitative and thematic data analysis, and assisting with translation are not applicable to this work. We have reviewed all AI-assisted work, especially the code and proofs, which we read and iterated on multiple times. In most of these usage scenarios, we additionally used another AI to review the work, e.g. proofread the paper for common misunderstandings. We take responsibility for the final content of this work, including text and claims produced with the aid of generative AI.

Ethics Statement

This work raises no ethical concerns that we are aware of.

Reproducibility Statement

This work is reproducible using the attached supplementary code at

`https://anonymous.4open.science/r/linmix-4292`

and relies solely on openly available datasets. Reproducing the results requires large computational resources and time, which we discuss in detail in Appendix G. To facilitate open research, we will share the trained proxy models and design matrices on HuggingFace and our code on GitHub.

References

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, Colin Raffel, Shiyu Chang, Tatsunori Hashimoto, and William Yang Wang. A survey on data selection for language models. *Transactions on Machine Learning Research*, 2024. (Cited on page 2.)

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martin Blazquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Agustín Piqueres Lajarín, Hynek Kydlíček, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan Son NGUYEN, Ben Burtenshaw, Clémentine Fourrier, Haojun Zhao, Hugo Larcher, Mathieu Morlon, Cyril Zakka, Colin Raffel, Leandro Von Werra, and Thomas Wolf. SmolLM2: When smol goes big — data-centric training of a fully open small language model. In *COLM*, 2025. (Cited on pages 6 and 19.)

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. (Cited on page 6.)

Tianyi Bai, Ling Yang, Zhen Hao Wong, Fupeng Sun, Xinlin Zhuang, Jiahui Peng, Chi Zhang, Lijun Wu, Qiu Jiantao, Wentao Zhang, Binhang Yuan, and Conghui He. Efficient pretraining data selection for language models via multi-actor collaboration. In *ACL*, 2025. (Cited on page 18.)

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *AAAI*, 2020. (Cited on page 4.)

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. GPT-NeoX-20B: An open-source autoregressive language model. In *ACL Workshop on Challenges & Perspectives in Creating Large Language Models*, 2022. (Cited on page 19.)

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. (Cited on pages 17 and 19.)

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. (Cited on page 6.)

Mayee F. Chen, Tyler Murray, David Heineman, Matt Jordan, Hannaneh Hajishirzi, Christopher Ré, Luca Soldaini, and Kyle Lo. Olmix: A framework for data mixing throughout LM development. *arXiv preprint arXiv:2602.12237*, 2026. (Cited on pages 1, 2, 3, 4, and 17.)

Shengzhuang Chen, Xu Ouyang, Michael Arthur Leopold Pearce, Thomas Hartvigsen, and Jonathan Richard Schwarz. ADMIRE-BayesOpt: Accelerated data MIxture RE-weighting for language models with bayesian optimization. *Transactions on Machine Learning Research*, 2025a. (Cited on page 17.)

Zhengyu Chen, Siqi Wang, Teng Xiao, Yudong Wang, Shiqi Chen, Xunliang Cai, Junxian He, and Jingang Wang. Revisiting scaling laws for language models: The role of data quality and training strategies. In *ACL*, 2025b. (Cited on page 4.)

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. (Cited on page 4.)

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. (Cited on page 6.)

William G. Cochran. *Sampling Techniques*. John Wiley & Sons, 3rd edition, 1977. (Cited on page 24.)

Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan SU, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, Mostofa Patwary, Yingyan (Celine) Lin, Jan Kautz, and Pavlo Molchanov. Nemotron-CLIMB: Clustering-based iterative data mixture bootstrapping for language model pre-training. In *NeurIPS*, 2025. (Cited on pages 1 and 17.)

Logan Engstrom, Axel Feldmann, and Aleksander Madry. Dsdm: Model-aware dataset selection with datamodels. *ICML*, 2024. (Cited on page 18.)

Simin Fan, Matteo Pagliardini, and Martin Jaggi. DOGE: Domain reweighting with generalization estimation. In *ICML*, 2024. (Cited on pages 1, 2, and 17.)

Ziqing Fan, Yuqiao Xian, Yan Sun, Ke Shen, and Li Shen. Joint selection for large-scale pre-training data via policy gradient-based mask learning. In *ICLR*, 2026. (Cited on page 18.)

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *NeurIPS*, 2020. (Cited on page 17.)

Fabian Fumagalli, Landon Butler, Justin Singh Kang, Kannan Ramchandran, and R. Teal Witter. An odd estimator for Shapley values. In *ICML*, 2026. (Cited on page 17.)

Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, Eyal Orgad, Rahim Entezari, Giannis Daras, Sarah M Pratt, Vivek Ramanujan, Yonatan Bitton, Kalyani Marathe, Stephen Mussmann, Richard Vencu, Mehdi Cherti, Ranjay Krishna, Pang Wei Koh, Olga Saukh, Alexander Ratner, Shuran Song, Hannaneh Hajishirzi, Ali Farhadi, Romain Beaumont, Sewoong Oh, Alex Dimakis, Jenia Jitsev, Yair Carmon, Vaishaal Shankar, and Ludwig Schmidt. Datacomp: In search of the next generation of multimodal datasets. In *NeurIPS*, 2023. (Cited on pages 1 and 18.)

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*, 2021. (Cited on page 4.)

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 2024. URL https://github.com/EleutherAI/lm-evaluation-harness. (Cited on page 20.)

Amirata Ghorbani and James Y. Zou. Data Shapley: Equitable valuation of data for machine learning. *ICML*, 2019. (Cited on pages 1, 17, and 25.)

Theodore Groves and Thomas Rothenberg. A note on the expected value of an inverse matrix. *Biometrika*, 56(3):690–691, 1969. (Cited on page 27.)

Peter L. Hammer and Ron Holzman. Approximations of pseudo-Boolean functions; applications to game theory. *ZOR – Methods and Models of Operations Research*, 36(1):3–21, 1992. (Cited on page 25.)

John M. Hammersley and Keith W. Morton. A new Monte Carlo technique: Antithetic variates. *Mathematical Proceedings of the Cambridge Philosophical Society*, 52(3):449–475, 1956. (Cited on page 17.)

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 *NeurIPS*, 2021. (Cited on page 6.)

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *NeurIPS*, 2022. (Cited on page 4.)

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2nd edition, 2013. (Cited on page 31.)

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Understanding predictions with data and data with predictions. *ICML*, 2022. (Cited on pages 2, 7, 8, 18, 24, and 25.)

Nikhil Kandpal, Brian Lester, Colin Raffel, Sebastian Majstorovic, Stella Biderman, Baber Abbasi, Luca Soldaini, Enrico Shippole, A. Feder Cooper, Aviya Skowron, Shayne Longpre, Lintang Sutawika, Alon Albalak, Zhenlin Xu, Guilherme Penedo, Loubna Ben allal, Elie Bakouch, John David Pressman, Honglu Fan, Dashiell Stander, Guangyu Song, Aaron Gokaslan, John Kirchenbauer, Tom Goldstein, Brian R. Bartoldson, Bhavya Kailkhura, and Tyler Murray. The Common Pile v0.1: An 8TB dataset of public domain and openly licensed text. In *NeurIPS*, 2025. (Cited on pages 4 and 20.)

Feiyang Kang, Yifan Sun, Bingbing Wen, Si Chen, Dawn Song, Rafid Mahmood, and Ruoxi Jia. AutoScale: Scale-aware data mixing for pre-training LLMs. In *COLM*, 2025. (Cited on page 17.)

Ari Karchmer, Martin Pawelczyk, and Seth Neel. Efficiently verifiable proofs of data attribution. *NeurIPS*, 2025. (Cited on pages 7 and 24.)

Rabeeh Karimi Mahabadi, Sanjeev Satheesh, Shrimai Prabhumoye, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-CC-Math: A 133 billion-token-scale high quality math pretraining dataset. In *ICLR*, 2026. (Cited on pages 6 and 19.)

Alaa Khaddaj, Logan Engstrom, and Aleksander Madry. Small-to-large generalization: Training data influences models consistently across scale. In *ICLR*, 2025. (Cited on page 18.)

Jack Kiefer. Optimum designs in regression problems, II. *The Annals of Mathematical Statistics*, 32 (1):298–325, 1961. (Cited on pages 9, 17, and 30.)

Jack Kiefer and Jacob Wolfowitz. Optimum designs in regression problems. *The Annals of Mathematical Statistics*, 30(2):271–294, 1959. (Cited on pages 1, 3, 17, and 26.)

Jack Kiefer and Jacob Wolfowitz. The equivalence of two extremum problems. *Canadian Journal of Mathematics*, 12:363–366, 1960. (Cited on page 17.)

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017. (Cited on page 1.)

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. *NeurIPS*, 2022. (Cited on page 6.)

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Kumar Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee F Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Kamal Mohamed Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Joshua P Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah M Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham M. Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander T Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alex Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-LM: In search of the next generation of training sets for language models. In *NeurIPS*, 2024. (Cited on pages 1 and 18.)

Weida Li and Yaoliang Yu. Robust data valuation with weighted Banzhaf values. In *NeurIPS*, 2023. (Cited on pages 17 and 25.)

Zeman Li, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. PiKE: Adaptive data mixing for large-scale multi-task learning under low gradient conflicts. In *NeurIPS*, 2025. (Cited on page 17.)

Zhenqing Ling, Daoyuan Chen, Liuyi Yao, Qianli Shen, Yaliang Li, and Ying Shen. Diversity as a reward: Fine-tuning LLMs on a mixture of domain-undetermined data. In *NeurIPS*, 2025. (Cited on page 18.)

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. RegMix: Data mixture as regression for language model pre-training. In *ICLR*, 2025. (Cited on pages 1, 2, 3, 4, and 17.)

Yajiao Liu, Congliang Chen, Junchi Yang, and Ruoyu Sun. Rethinking data mixture for large language models: A comprehensive survey and new perspectives. In *Findings of EACL*, March 2026. (Cited on page 17.)

Jean-Luc Marichal and Pierre Mathonet. Weighted Banzhaf power and interaction indexes through weighted approximations of games. *European Journal of Operational Research*, 211(2):352–358, 2011. (Cited on page 25.)

Robert A. McLean and Virgil L. Anderson. Extreme vertices design of mixture experiments. *Technometrics*, 8(3):447–454, 1966. (Cited on page 17.)

Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556–567, 2000. (Cited on page 17.)

Hyeonseok Moon, Jaehyung Seo, Seonmin Koo, Jinsung Kim, Young-kyoung Ham, Jiwon Moon, and Heuiseok Lim. LimaCost: Data valuation for instruction tuning of large language models. In *Findings of EMNLP*, 2025. (Cited on page 18.)

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. *Journal of Machine Learning Research*, 26:53:1–53:66, 2025. (Cited on page 2.)

Christopher Musco and R. Teal Witter. Provably accurate Shapley value estimation via leverage score sampling. In *International Conference on Learning Representations (ICLR)*, 2025. (Cited on page 17.)

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: Attributing model behavior at scale. In *ICML*, 2023. (Cited on page 18.)

Tomer Porian, Mitchell Wortsman, Jenia Jitsev, Ludwig Schmidt, and Yair Carmon. Resolving discrepancies in compute-optimal scaling of language models. In *NeurIPS*, 2024. (Cited on page 20.)

Friedrich Pukelsheim. *Optimal Design of Experiments*. SIAM, 2006. (Cited on pages 26 and 30.)

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5. (Cited on pages 6 and 19.)

Nikunj Saunshi, Arushi Gupta, Mark Braverman, and Sanjeev Arora. Understanding influence functions and datamodels via harmonic analysis. *ICLR*, 2023. (Cited on page 18.)

Henry Scheffé. Experiments with mixtures. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):344–360, 1958. (Cited on pages 17 and 30.)

Jack Sherman and Winifred J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950. (Cited on page 26.)

Mustafa Shukor, Louis Bethune, Dan Busbridge, David Grangier, Enrico Fini, Alaaeldin El-Nouby, and Pierre Ablin. Scaling laws for optimal data mixtures. In *NeurIPS*, 2025. (Cited on page 17.)

Haoru Tan, Sitong Wu, Yanfeng Chen, Jun Xia, Ruobing Xie, Bin Xia, Xingwu Sun, and Xiaojuan Qi. Fast data mixture optimization via gradient descent. In *ICLR*, 2026. (Cited on pages 1 and 17.)

Team Olmo. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025. (Cited on page 1.)

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 furious (COLM's version). In *COLM*, 2025. (Cited on pages 4 and 19.)

Tristan Thrush, Christopher Potts, and Tatsunori Hashimoto. Improving pretraining data using perplexity correlations. In *ICLR*, 2025. (Cited on page 17.)

Anvith Thudi, Evianne Rovers, Yangjun Ruan, Tristan Thrush, and Chris J. Maddison. MixMin: Finding data mixtures via convex minimization. In *ICML*, 2025. (Cited on pages 1 and 17.)

Joel A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012. (Cited on page 28.)

Jiachen T. Wang and Ruoxi Jia. Data Banzhaf: A robust data valuation framework for machine learning. *AISTATS*, 2023. (Cited on pages 17 and 25.)

Jiachen T. Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data Shapley in one training run. In *ICLR*, 2025. (Cited on page 18.)

Jiachen T. Wang, Tong Wu, Kaifeng Lyu, James Zou, Dawn Song, Ruoxi Jia, and Prateek Mittal. Can small training runs reliably guide data curation? rethinking proxy-model practice. In *ICLR*, 2026a. (Cited on page 18.)

Shaobo Wang, Xuan Ouyang, Tianyi Xu, Yuzheng Hu, Jialin Liu, Guo Chen, Tianyu Zhang, Junhao Zheng, Kexin Yang, Xingzhang Ren, Dayiheng Liu, and Linfeng Zhang. OPUS: Towards efficient and principled data selection in large language model pre-training in every iteration. In *ICML*, 2026b. (Cited on page 18.)

Bingbing Wen, Sirajul Salekin, Feiyang Kang, Bill Howe, Lucy Lu Wang, Javier Movellan, and Manjot Bilkhu. MixAtlas: Uncertainty-aware data mixture for multimodal LLM midtraining. In *ICLR Workshop on Navigating and Addressing Data Problems for Foundation Models*, 2026. (Cited on page 17.)

Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. Organize the Web: Constructing domains enhances pre-training data curation. In *ICML*, 2025. (Cited on pages 1, 2, 3, 4, 17, and 19.)

Inho Won, Hangyeol Yoo, Minkyung Cho, Jungyeul Park, Hoyun Song, and KyungTae Lim. TREX: Tokenizer regression for optimal data mixture. In *EACL*, 2026. (Cited on page 17.)

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. DoReMi: Optimizing data mixtures speeds up language model pretraining. In *NeurIPS*, 2023. (Cited on pages 1, 2, and 17.)

Wanyun Xie, Francesco Tonin, and Volkan Cevher. Chameleon: A flexible data-mixing framework for language model pretraining and finetuning. In *ICML*, 2025. (Cited on pages 1 and 17.)

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. Data Mixing Laws: Optimizing data mixtures by predicting language modeling performance. In *ICLR*, 2025. (Cited on pages 1, 3, and 17.)

Thomson Yen, Andrew Siah, Haozhe Chen, C. Daniel Guetta, Tianyi Peng, and Hongseok Namkoong. Data Mixture Optimization: A multi-fidelity multi-scale bayesian framework. In *NeurIPS*, 2025. (Cited on page 17.)

Yang Yu, Kai Han, Hang Zhou, Yehui Tang, Kaiqi Huang, Yunhe Wang, and Dacheng Tao. LLM data selection and utilization via dynamic bi-level optimization. In *ICML*, 2025a. (Cited on page 18.)

Zichun Yu, Fei Peng, Jie Lei, Arnold Overwijk, Scott Yih, and Chenyan Xiong. Group-level data selection for efficient pretraining. In *NeurIPS*, 2025b. (Cited on page 18.)

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *ACL*, 2019. (Cited on page 4.)

Xinlin Zhuang, Jiahui Peng, Ren Ma, Yinfan Wang, Tianyi Bai, Xingjian Wei, Qiu Jiantao, Chi Zhang, Ying Qian, and Conghui He. Meta-rater: A multi-dimensional data selection method for pre-training language models. In *ACL*, 2025. (Cited on page 18.)

George Kingsley Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949. (Cited on pages 8 and 27.)

APPENDIX "LINMIX: DATA MIXING AS LINEAR DOMAIN ATTRIBUTION"

A Related Work **17**

B Proofs **18**

C Experimental Setup **19**

D Additional Results **21**

E Theoretical Analysis **24**

E.1 From datamodels to domain attribution 24

E.2 The design matrix governs the estimation error 25

E.3 Pitfalls of Dirichlet sampling 26

E.4 On optimal sampling 29

F Limitations **32**

G Compute Resources **32**

A Related Work

Design of experiments. Sampling $\mathbf{\Pi}$ is a classic problem in statistics. Because each row is a vector of domain proportions that sum to one, the proxy swarm is a mixture experiment (Scheffé, 1958). Choosing inputs that minimize the variance of a regression estimate is the subject of optimal experimental design. A design is *D-optimal* when it maximizes $\det(\mathbf{\Pi}^\top \mathbf{\Pi})$, which minimizes the volume of the confidence ellipsoid of $\hat{\boldsymbol{\beta}}$ (Kiefer & Wolfowitz, 1959). For approximate designs, Kiefer & Wolfowitz (1960) show that maximizing this determinant also minimizes the worst-case prediction variance. The D-optimal design for the first-degree mixture model places equal weight on the vertices of the simplex (Kiefer, 1961). When upper bounds such as our caps u_k truncate the simplex, the extreme vertices designs of McLean & Anderson (1966) likewise use the vertices of the constrained region. LinMix follows this principle at a scale where computing exact optimal designs on $\mathcal{P}_{\text{cap}}$ is impractical, sampling vertices generated by uniformly random domain permutations. We prove that this sampling scheme is optimal in expectation on $\mathcal{P}_{\text{cap}}$ with equal caps (Theorem 2) and leave the case of unequal caps to future work. In Section 4.1, we tune Dirichlet sampling by minimizing the condition number of $\mathbf{\Pi}$, a criterion closely related to D-optimality and $\det(\mathbf{\Pi}^\top \mathbf{\Pi})$. Both degrade together because nearly-collinear mixes reduce the smallest singular value of $\mathbf{\Pi}$, thereby collapsing the determinant and inflating the condition number. LinMix also samples proxy mixes in pairs by filling domains in a random order and then in reverse order, following the idea of antithetic variates (Hammersley & Morton, 1956). This relates to pairing sampled coalitions with their complements that has improved Shapley value approximation (Fumagalli et al., 2026), though our capped mixes need not be exact complements. More broadly, regression design is of interest for improving attribution, e.g. Musco & Witter (2025) improve regression-based Shapley value approximation via leverage-score sampling of coalitions.

Data mixing. Data mixing methods differ in how they estimate domain value and spend compute to choose a mix (Liu et al., 2026). Online methods adapt the mixture during training. DoReMi (Xie et al., 2023) learns weights during proxy training through group distributionally robust optimization, DoGE (Fan et al., 2024) and Tan et al. (2026) use gradient-based updates guided by generalization or validation performance, and PiKE (Li et al., 2025) re-weights data sources from their gradients. MixMin (Thudi et al., 2025) optimizes a weighted ensemble of proxy predictions, while Chameleon (Xie et al., 2025) derives weights from relationships between domain embeddings. Data Mixing Laws (Ye et al., 2025), AutoScale (Kang et al., 2025), and Shukor et al. (2025) use scaling relationships to predict mixture performance or select mixtures at larger training scales. Mixture optimization also extends to tokenizers (Won et al., 2026) and multimodal midtraining (Wen et al., 2026). Adaptive search allocates new proxy runs to promising mixtures through iterative predictor fitting (Diao et al., 2025) or Bayesian optimization (Chen et al., 2025a; Yen et al., 2025). LinMix instead fixes its proxy mixtures before observing their outcomes, using vertex sampling to estimate domain effects across the feasible region. This design enables a useful linear fit across hundreds of domains; combining it with adaptive search is a direction for future work.

Theory of optimal regression-based data mixing. Regression-based data mixing methods select the target mix by minimizing a fitted surrogate model, offering little theoretical insight into the optimal mix itself (Liu et al., 2025; Wettig et al., 2025; Chen et al., 2026). Recently, Chen et al. (2026) bound the performance loss when a previously optimized mix is reused after the domain set changes, taking the mixing law and its exact minimization as given. Thrush et al. (2025) rank domains by correlating the per-domain losses of public models with their benchmark scores instead, and prove that such rankings are optimally used by saturating the caps in rank order. In our framework, this rule is the vertex solution of Eq. (5) at $r = 0$, with the ranking estimated from a designed experiment on training mixtures rather than a correlation across uncontrolled models. For $r > 0$, the objective is a weighted *isoelastic* utility, also known as the alpha-fair family of resource allocation (Mo & Walrand, 2000), whose marginal utility grows unbounded at zero weight, so the maximizer places positive weight on every positively attributed domain (Proposition 1). Corollary 1 solves the KKT conditions in closed form, yielding a capped water-filling allocation (cf. Boyd & Vandenberghe, 2004).

Data attribution. Our work relates to the parallel literature on data attribution, which typically studies the influence of individual training examples rather than entire domains (Ghorbani & Zou, 2019; Feldman & Zhang, 2020; Wang & Jia, 2023; Li & Yu, 2023). We identify a connection

between regression-based data mixing and datamodels (Ilyas et al., 2022; Saunshi et al., 2023), which predict model outputs from training-set membership and show that a linear model can capture these dependencies. DsDm (Engstrom et al., 2024) turns this predictive view into data selection, using gradient-based TRAK estimates (Park et al., 2023) to rank individual examples and select a fixed-size subset. LINMIX shares this principle but estimates domain proportions from proxy runs and allocates tokens under domain caps. Using these estimates for a larger target model requires that data effects transfer across scales, as studied by Khaddaj et al. (2025). Transfer depends on proxy training hyperparameters, as lower learning rates improve agreement with the dataset rankings of larger models (Wang et al., 2026a). With a limited number of runs (Wang et al., 2025), the question becomes which mixtures reveal enough about each domain to support the allocation, which vertex sampling answers by design.

Data selection and filtering. Data quality and filtering methods improve training corpora by selecting or weighting individual examples. DataComp (Gadre et al., 2023; Li et al., 2024) benchmarks how data curation affects large model performance. Methods that combine selection criteria include Meta-rater (Zhuang et al., 2025), which learns combinations of quality scores from proxy runs, and Fan et al. (2026), who jointly optimize quality and diversity through learned selection masks, while Group-MATES (Yu et al., 2025b) further accounts for interactions between examples. During training, selection can adapt by coordinating multiple criteria (Bai et al., 2025), updating example weights (Yu et al., 2025a), or scoring examples by their optimizer updates (Wang et al., 2026b). For fine-tuning, Ling et al. (2025) select data using a diversity reward, while LimaCost (Moon et al., 2025) measures instruction-data value against a reference dataset using gradients. Filtering and mixing can be combined by curating the corpus before optimizing its domain mix, as in our experiments.

B PROOFS

Proposition 1 (Domain support and uniqueness of the optimal mix). *Let $\mathcal{A} \coloneqq \{k : \hat{\beta}_k > 0\}$ be the set of domains with positive estimated attribution, and assume their caps can absorb the whole budget, i.e. $u_{\mathcal{A}} \coloneqq \sum_{k\in\mathcal{A}} u_k > 1$. At $r = 1$, the objective of Eq. (5) reads as $\sum_{k\in\mathcal{A}} \hat{\beta}_k \ln \pi_k$, its log-linear form with the sum restricted to $\mathcal{A}$. Then, for every $r \in (0, 1]$, its maximizer $\boldsymbol{\pi}^\star$ is unique and satisfies $\pi^\star_k > 0$ for all $k \in \mathcal{A}$ and $\pi^\star_k = 0$ for all $k \notin \mathcal{A}$. At $r = 0$, in contrast, a maximizer may assign zero weight to domains in $\mathcal{A}$. Proofs are given in Appendix B.*

Proof. Throughout, let $\tilde{\boldsymbol{\pi}}$ denote the feasible mix that spreads the budget over $\mathcal{A}$ in proportion to the caps, $\tilde{\pi}_k \coloneqq u_k/u_{\mathcal{A}}$ for $k \in \mathcal{A}$ and $\tilde{\pi}_k \coloneqq 0$ otherwise. Its entries sum to one and $u_{\mathcal{A}} > 1$ gives $\tilde{\pi}_k < u_k$, so it is feasible and puts positive weight on every domain in $\mathcal{A}$.

Existence. For $r \in (0, 1)$, the objective $\nu_r(\boldsymbol{\pi}) \coloneqq \sum_{k=1}^{d} \hat{\beta}_k \frac{\pi_k^{1-r}-1}{1-r}$ is continuous on the compact set $\mathcal{P}_{\text{cap}}$, so a maximizer exists. For $r = 1$, the proposition defines the objective as $\nu_1(\boldsymbol{\pi}) \coloneqq \sum_{k\in\mathcal{A}} \hat{\beta}_k \ln \pi_k$ with the convention $\ln 0 = -\infty$. This convention continues the case $r < 1$, where all maximizers lie on the face $\{\pi_k = 0 \text{ for } k \notin \mathcal{A}\}$. On this face the omitted terms are constants that do not affect the maximizer, and the remaining sum converges pointwise to ν_1 as $r \to 1$. ν_1 is upper semicontinuous and bounded above by zero, so it attains a finite maximum, as $\nu_1(\tilde{\boldsymbol{\pi}}) > -\infty$.

Positive weight on every domain in $\mathcal{A}$. The utility of a domain in $\mathcal{A}$ has an unbounded derivative at zero weight, so leaving such a domain empty cannot be optimal. To make this precise, let $\boldsymbol{\pi}^\star$ be any maximizer and suppose $\pi^\star_j = 0$ for some $j \in \mathcal{A}$. For $r = 1$ this gives $\nu_1(\boldsymbol{\pi}^\star) = -\infty < \nu_1(\tilde{\boldsymbol{\pi}})$, a contradiction. For $r \in (0, 1)$, pick any i with $\pi^\star_i > 0$ and move a small weight $\varepsilon \in (0, \min(\pi^\star_i/2,\, u_j))$ from i to j. Only two terms change. The j-th gains $\hat{\beta}_j \frac{\varepsilon^{1-r}}{1-r} > 0$. The i-th loses at most $\hat{\beta}^+_i (\pi^\star_i/2)^{-r}\, \varepsilon$ with $\hat{\beta}^+_i \coloneqq \max(\hat{\beta}_i, 0)$, by the mean value theorem on $[\pi^\star_i/2,\, \pi^\star_i]$. The gain is roughly ε^{1-r} while the loss is only proportional to ε, so the move strictly improves the objective for small ε, again a contradiction. Hence $\pi^\star_k > 0$ for all $k \in \mathcal{A}$.

Zero weight outside $\mathcal{A}$. Conversely, weight outside $\mathcal{A}$ adds no value and is better spent inside. Suppose $\pi^\star_j > 0$ for some $j \notin \mathcal{A}$. Because $\sum_{k\in\mathcal{A}} \pi^\star_k \le 1 < u_{\mathcal{A}}$, some domain $i \in \mathcal{A}$ sits strictly below its cap. Moving the weight $\varepsilon \coloneqq \min(\pi^\star_j,\, u_i - \pi^\star_i) > 0$ from j to i is feasible. The j-th term

does not decrease, since $\hat{\beta}_j \leq 0$ makes it non-increasing in π_j, and for $r = 1$ it is absent from ν_1 altogether. The i-th term strictly increases, a contradiction. Hence $\pi_k^\star = 0$ for all $k \notin \mathcal{A}$.

Uniqueness. By the two steps above, every maximizer lies in the convex set $\mathcal{F} := \{\boldsymbol{\pi} \in \mathcal{P}_{\text{cap}} : \pi_k = 0 \text{ for } k \notin \mathcal{A}\}$ and is strictly positive on $\mathcal{A}$. On $\mathcal{F}$, the objective is a sum of strictly concave functions of the coordinates in $\mathcal{A}$, since $\hat{\beta}_k > 0$ and $r \in (0, 1]$. Two distinct maximizers would therefore be beaten by their midpoint, so the maximizer is unique.

Linear limit. At $r = 0$, the objective $\nu_0(\boldsymbol{\pi}) = \hat{\boldsymbol{\beta}}^\top \boldsymbol{\pi} - \hat{\boldsymbol{\beta}}^\top \mathbf{1}$ is linear, so it is maximized at a vertex of the truncated simplex. The greedy fill solving Eq. (4) can then exhaust the budget before reaching the lowest-value domains in $\mathcal{A}$ and leave them at zero weight. For example, $\hat{\boldsymbol{\beta}} = (2, 1, -3)$ with $u = (1, 1, 1)$ yields the optimum $(1, 0, 0)$. □

Corollary 1 (Closed form of the optimal mix). *Under the assumptions of Proposition 1 and for $r \in (0, 1]$, the objective of Eq. (5) restricted to its positively attributed support $\mathcal{A}$ is strictly concave. Then, the maximizer takes the closed form $\hat{\pi}_{target,k} = \min\left(u_k, (\hat{\beta}_k/\omega)^{1/r}\right)$ for $k \in \mathcal{A}$ and $\hat{\pi}_{target,k} = 0$ otherwise, where $\omega > 0$ is the unique scalar enforcing $\sum_k \hat{\pi}_{target,k} = 1$.*

Proof. Fix $r \in (0, 1]$. The claimed strict concavity on the support was established in the uniqueness step of Proposition 1, as each summand over $\mathcal{A}$ is strictly concave. By Proposition 1, the maximizer $\boldsymbol{\pi}^\star$ is unique, lies in $\mathcal{F}$, and satisfies $\pi_k^\star > 0$ for all $k \in \mathcal{A}$. On $\mathcal{F}$, the problem maximizes this concave objective under the affine constraints $\sum_{k\in\mathcal{A}} \pi_k = 1$ and $\pi_k \leq u_k$. Because every $\pi_k^\star$ is positive, the objective is differentiable at $\boldsymbol{\pi}^\star$ and the constraints $\pi_k \geq 0$ are inactive, so $\boldsymbol{\pi}^\star$ satisfies the KKT conditions (cf. Boyd & Vandenberghe, 2004) with a scalar ω for the budget and multipliers $\mu_k \geq 0$ for the caps. The marginal utility of domain k is $\hat{\beta}_k \pi_k^{-r}$, so stationarity reads $\hat{\beta}_k (\pi_k^\star)^{-r} = \omega + \mu_k$ for every $k \in \mathcal{A}$. In words, the optimum equalizes marginal utilities at the common level ω, except where a cap binds. Because $u_{\mathcal{A}} > 1 = \sum_{k\in\mathcal{A}} \pi_k^\star$, some domain lies strictly below its cap. For this domain, complementary slackness gives $\mu_k = 0$, so $\omega = \hat{\beta}_k (\pi_k^\star)^{-r} > 0$. Now the closed form can be read off. If $\pi_k^\star < u_k$, then $\mu_k = 0$ and stationarity gives $\pi_k^\star = (\hat{\beta}_k/\omega)^{1/r}$. If $\pi_k^\star = u_k$, then $\mu_k \geq 0$ gives $(\hat{\beta}_k/\omega)^{1/r} \geq u_k$. In both cases, $\pi_k^\star = \min\left(u_k, (\hat{\beta}_k/\omega)^{1/r}\right)$. Since $\boldsymbol{\pi}^\star$ is feasible, summing this form over $k \in \mathcal{A}$ shows that ω enforces the normalization $\sum_{k\in\mathcal{A}} \min\left(u_k, (\hat{\beta}_k/\omega)^{1/r}\right) = 1$. Finally, ω is unique. The sum $\sum_{k\in\mathcal{A}} \min\left(u_k, (\hat{\beta}_k/\omega)^{1/r}\right)$ is continuous and non-increasing in ω. At any ω where it equals $1 < u_{\mathcal{A}}$, some term lies strictly below its cap and there decreases strictly in ω. Therefore the value 1 is attained at exactly one ω. □

C Experimental Setup

Pretraining data. We tokenize data into single-domain sequences of 2048 tokens with the GPT-NeoX-20B tokenizer (Black et al., 2022), which has a vocabulary of 50,277 tokens. This way, the WebOrganizer corpus (Wettig et al., 2025) yields 195B tokens, and the 40-domain corpus adds 168B tokens of Stack-Edu (Allal et al., 2025) across 15 programming languages, plus the 53B-token highest-quality subset of Nemotron-CC-Math (Karimi Mahabadi et al., 2026), bringing the total data size to 416B. Figure 6 shows the resulting prior data mixes.

Sampling data mixes. The caps κ become 5.98 and 5.63 for the 32B target token budget at 24 and 288 domains, and $\kappa = 3.40$ at 40 domains. We initially derived the latter for a 120B budget, allowing us to further train the models beyond the 80B tokens used. Dirichlet sampling uses $\tau = 2$ and $[\gamma_{\text{lo}}, \gamma_{\text{hi}}] = [0.1, 10]$, $[40, 4000]$, and $[100, 10^4]$ at 24, 288, and 40 domains, respectively. The 40-domain range follows the criterion of Section 4.1: the default parameters reject over 99.9% of draws, and the condition number of $\boldsymbol{\Pi}$ is minimized near $\gamma = 10^2$. We keep the $100\times$ span of the tuned range, over which the rejection rate falls from 0.96 to 0.01. For both samplers, we draw 10^4 designs and keep the best-conditioned $\boldsymbol{\Pi}$, which gives the most stable regression coefficients.

Language models. Table 3 lists model architectures and hyperparameters. Both model families follow their reference implementations (Team OLMo et al., 2025; Qwen Team, 2026); Qwen3.5

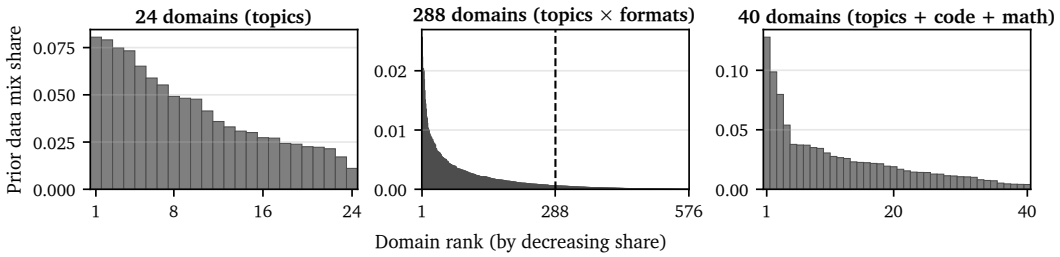


Figure 6: Prior data mix distributions. Domain shares for the three setups: 24 domains (topics), 288 domains formed from topic and format pairs, and 40 domains extending topics with code and math data. The dashed line marks the removal of the distribution's tail, where the kept larger half of all 576 pairs covers 94% of the corpus.

interleaves three gated DeltaNet layers per attention layer. All models use AdamW with $\beta_1 = 0.9$ and weight decay 0.1. The peak learning rate scales with the inverse cube root of the model parameter count (Porian et al., 2024). The schedule warms up linearly over the first 10% of steps, decays by cosine to 10% of the peak, and then linearly to zero over the final 10%.

Table 3: Model architectures and training hyperparameters. Parameters include embeddings. Qwen3.5 layer counts list gated DeltaNet + full-attention layers. Batch gives sequences per step.

Model	Params	Layers	Heads	Hidden size	Batch	Tokens	Peak LR	β_2
OLMo2 30M	29.2M	4	8	256	64	1B	3.6×10^{-3}	0.99
OLMo2 60M	57.6M	8	8	384	128	2B	2.9×10^{-3}	0.98
OLMo2 100M	102.0M	12	8	512	128	3.2B	2.4×10^{-3}	0.97
OLMo2 1B	1.41B	18	16	2048	512	32B	1.1×10^{-3}	0.95
Qwen3.5 100M	104.5M	9+3	4	576	128	3.2B	2.4×10^{-3}	0.97
Qwen3.5 1B	1.33B	12+4	16	2048	512	32B	1.1×10^{-3}	0.95
Qwen3.5 4B	3.70B	24+8	16	2560	1024	80B	7.1×10^{-4}	0.95

Evaluation. We use the LM Evaluation Harness (Gao et al., 2024) for a standardized evaluation. Proxy models are scored by BPB on the training split of each benchmark, and target models by accuracy and BPB on their held-out set (Table 4). Shots are drawn from the training split, except MBPP and MATH, which use their standard fixed prompts, and HumanEval BPB, which relies on three hand-written examples. For the two-benchmark targets, ARC (easy and challenge) and HS+PIQA, we fit a surrogate per benchmark, minimize the mean of their predictions, and report the mean of the two target model scores. For the 4B models, math and code benchmarks are scored using greedy decoding, with exact match for math and pass@1 for code. We draw two disjoint held-out sets from the Common Pile dataset (Kandpal et al., 2025): a 150k-sequence split on which we score the proxy models to fit the surrogates, and a 300k-sequence split reserved for scoring the target models.

Table 4: Benchmark splits, sample sizes, and few-shot counts. Dashes mark benchmarks used only as held-out checks.

Benchmark	Proxy split (size)	Held-out set (size)	Shots	Target metric
ARC-Easy	train (2,251)	test (2,376)	5	normalized acc.
ARC-Challenge	train (1,119)	test (1,172)	5	normalized acc.
HellaSwag	train (39,905)	validation (10,042)	5	normalized acc.
PIQA	train (16,113)	validation (1,838)	5	normalized acc.
MBPP	train (374)	test (500)	3	pass@1
GSM8K	train (7,473)	test (1,319)	5	exact match
HumanEval	—	test (164)	0	pass@1
MATH	—	test (5,000)	4	exact match

D ADDITIONAL RESULTS

Figure 7 extends Figure 3 with R^2 measurements, and Figure 8 repeats it for the four targets of the 4B experiments at 40 domains, where vertex sampling reaches $\rho \geq 0.94$ with every surrogate at $m = 512$, while the tuned Dirichlet swarm stays at $\rho \leq 0.68$ on the three benchmarks and 0.80 on CPile. Figure 9 shows that the linear fit of the vertex swarm transfers across proxy scales. Figures 10 and 11 visualize the interpretable domain weights of the optimized mixes at 24 and 288 domains and for the Common Pile loss at 40 domains.

Tables 5 and 6 ablate the two optimization choices of LINMIX on the HellaSwag target. Table 5 tests our hypothesis that dividing the coefficients $\hat{\beta}$ by their standard errors, which discounts uncertain domains, improves the mix. The gain is at most half a point of accuracy for both the vertex swarm and the tuned Dirichlet swarm with $\gamma \in [40, 4000]$. Table 6 shows that the interpretable transformation of π matters at 24 domains, where π ($r = 0$) concentrates the mix on four domains at their caps u_k. This support is too narrow for the target budget of 32B tokens, which a larger budget would spread through lower caps, and $\sqrt{\pi}$ or $\ln \pi$ ($r > 0$) spreads at the current one.

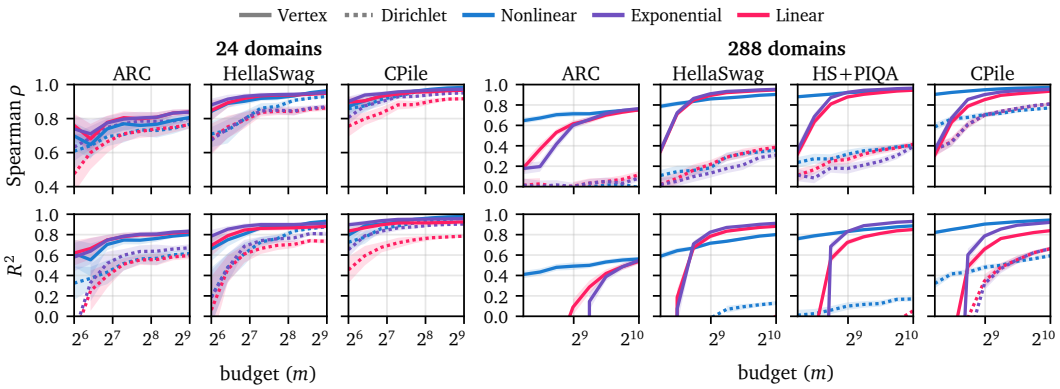


Figure 7: Performance of optimizers under Vertex and Dirichlet samplers. Extended Figure 3. Average Spearman correlation and R^2 across 5 random repetitions of 10-fold cross-validation under a fixed proxy model budget. Nonlinear is implemented using LightGBM, as in REGMIX, and exponential fits the exponential law to the target, as in OLMIX.

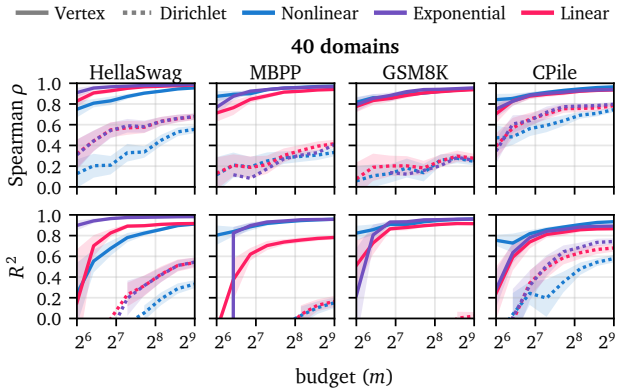


Figure 8: Optimizer performance on the targets of the 4B experiments. Figure 7 for the 40-domain setup, where $512\times$ 100M Qwen3.5 proxy models per sampler are fit to HellaSwag, MBPP, GSM8K, and the held-out Common Pile loss (CPile). Spearman correlation and R^2 are averaged across 5 random repetitions of 10-fold cross-validation.

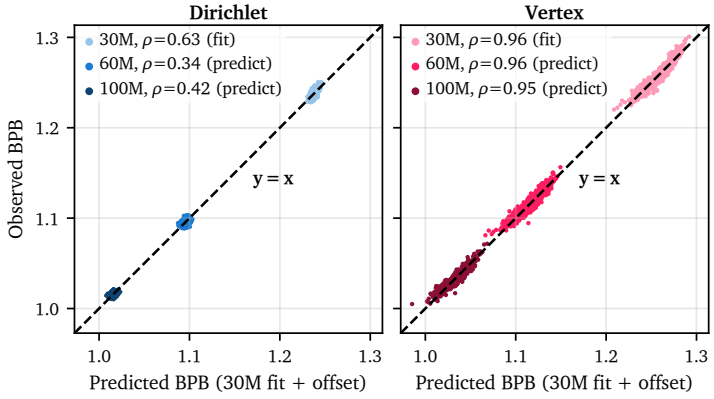


Figure 9: The linear attribution fit transfers across proxy scales only under vertex sampling. A linear model is fit to predict HellaSwag BPB from the mix over the $1024\times$ 30M proxy models at 288 domains, and predicts the outcomes of the 60M and 100M swarms trained on the same mixes. Absolute BPB drops with proxy scale, so we offset predictions by one constant per swarm without changing rankings. Dirichlet-sampled mixes yield a weak in-sample fit whose ranking does not transfer, whereas vertex sampling yields a ranking that transfers across model scales.

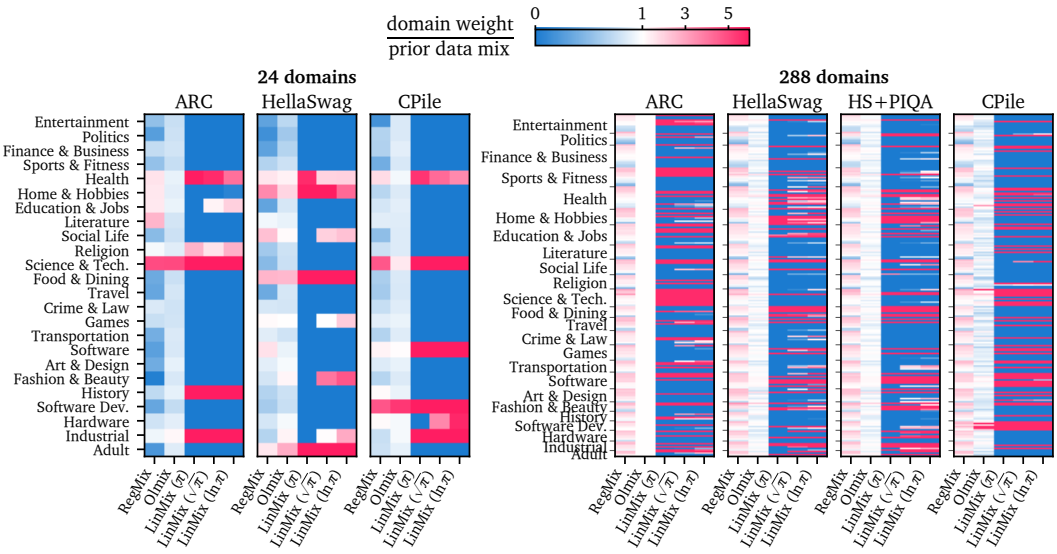


Figure 10: Domain weights optimized for the 1B target benchmarks. Domain weights relative to the prior mix across data mixing methods and targets on the 200B-token corpus split into 24 and 288 domains, including the held-out Common Pile loss (CPile). For 288 domains, rows retain the 24-topic order and sort each topic's formats by increasing share of the prior data mix. Topic names are at the group centers, and since a topic group with few formats is shorter than its label, it can overlap vertically. Beyond π $(r = 0)$, the support of the target mix is the set of all positively attributed domains; the mixes at $\sqrt{\pi}$ $(r = 0.5)$ and $\ln \pi$ $(r = 1)$ thus share this support and differ only in the domain weight magnitudes (Proposition 1). Based on this, Table 1 trains the LINMIX $\ln \pi$ mix for HellaSwag at 24 domains, the $\sqrt{\pi}$ mix for CPile at 24 domains, and the π mix otherwise.

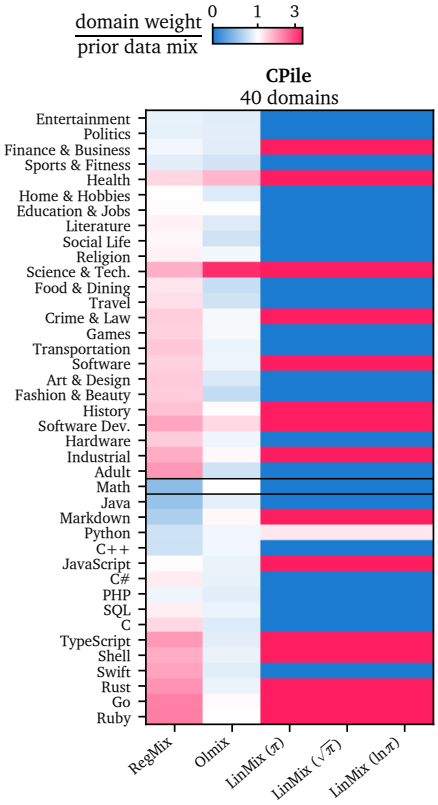


Figure 11: Domain weights optimized for validation loss at 40 domains. Domain weights relative to the prior mix across data mixing methods for BPB on the Common Pile. The three LINMIX columns vary the utility transformation.

Table 5: Ablation of linear model scaling and proxy sampling at 288 domains on HellaSwag.

Sampling	Scaling	Acc↑	BPB↓
Dirichlet	$\hat{\beta}$	.597	.764
	$\hat{\beta}/\mathrm{SE}$	.602	.761
Vertex	$\hat{\beta}$	.615	.758
	$\hat{\beta}/\mathrm{SE}$	.617	.759

Table 6: Ablation of the utility transformation via curvature r in LINMIX on the HellaSwag target.

	24 domains		288 domains	
Utility	Acc↑	BPB↓	Acc↑	BPB↓
π $(r = 0)$	.543	.795	.615	.758
$\sqrt{\pi}$ $(r = .5)$	.565	.782	.611	.758
$\ln \pi$ $(r = 1)$	.568	.785	.610	.759

E THEORETICAL ANALYSIS

We formalize the domain attribution view of data mixing from first principles. Starting from linear datamodels, we show that the expected outcome of a proxy run is linear in its domain mix, with coefficients that average sample-level attribution scores (Proposition 2, Corollary 2), and that estimating them is a linear regression whose error is governed by the design $\mathbf{\Pi}$ (Lemma 1). Dirichlet sampling inflates this error as its mixes concentrate around the prior and as the number of domains grows (Theorem 1), whereas vertex sampling is optimal on the simplex (Proposition 3) and, in expectation, on the capped simplex with equal caps (Theorem 2).

E.1 FROM DATAMODELS TO DOMAIN ATTRIBUTION

Datamodeling (Ilyas et al., 2022) attributes model outcomes to training data counterfactually, in a regime where many training runs on random subsets of the data are feasible. Fix a learning algorithm and a target input $\boldsymbol{z}$, e.g. a benchmark question, and let $y(S;\boldsymbol{z})$ denote the loss on $\boldsymbol{z}$ of a model trained on a subset S of the corpus $\mathcal{D} \coloneqq \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_d$ of n sequences. A datamodel is fit to the pairs $(\mathbf{1}_{S_i}, y(S_i;\boldsymbol{z}))$ collected over runs $i = 1, \ldots, m$, where $\mathbf{1}_{S_i} \in \{0,1\}^n$ indicates the sequences included in run i. Empirically, a linear fit predicts the outcomes of held-out runs well and motivates the model

$$y(S;\boldsymbol{z}) = \langle \boldsymbol{\beta}(\boldsymbol{z}), \mathbf{1}_S \rangle + \eta(\boldsymbol{z}), \qquad \text{(Datamodel)} \tag{6}$$

where $\boldsymbol{\beta}(\boldsymbol{z}) \in \mathbb{R}^n$ collects the attribution scores, i.e., the marginal effect of including each sequence on the loss at $\boldsymbol{z}$, and $\eta(\boldsymbol{z})$ is a noise term with zero mean given S that captures the randomness of training. Follow-up work relies on this local linearity to relate attribution scores to model outcomes (Karchmer et al., 2025), and we adopt it as the assumption underlying this analysis. As all subsets considered below have the same size, an intercept can be absorbed into $\boldsymbol{\beta}(\boldsymbol{z})$ and is omitted.

Consider a proxy run on the mix $\boldsymbol{\pi}$ that trains on $b \coloneqq b_{\text{proxy}}$ sequences, of which $b\pi_k$ are drawn uniformly without replacement from domain $\mathcal{D}_k$, independently across domains, where we assume that $b\pi_k \le n_k$ is an integer. Let its outcome y be the benchmark loss, i.e., the average of $y(S;\boldsymbol{z})$ over the set of benchmark inputs $\mathcal{Z}$. Let $\boldsymbol{\beta}^{\mathcal{Z}} \coloneqq \frac{1}{|\mathcal{Z}|}\sum_{\boldsymbol{z}\in\mathcal{Z}} \boldsymbol{\beta}(\boldsymbol{z})$ denote the benchmark-averaged attribution scores, and let μ_k and $\sigma_k^2 \coloneqq \frac{1}{n_k}\sum_{s\in\mathcal{D}_k}(\beta_s^{\mathcal{Z}} - \mu_k)^2$ denote their mean and variance over the sequences of domain k. Proposition 2 formalizes the data attribution view of data mixing.

Proposition 2 (Datamodels coarsen to domain attribution). *Under the datamodel of Eq. (6), the expected outcome of a proxy run is linear in its mix. Specifically, with $\bar{\beta}_k \coloneqq b\,\mu_k$,*

$$\mathbb{E}[y \mid \boldsymbol{\pi}] = \langle \bar{\boldsymbol{\beta}}, \boldsymbol{\pi} \rangle \qquad \textit{and} \qquad \mathrm{Var}(y \mid \boldsymbol{\pi}) \le \sigma_\eta^2 + b \max_k \sigma_k^2, \tag{7}$$

where σ_η^2 bounds the variance of the benchmark-averaged noise $\frac{1}{|\mathcal{Z}|}\sum_{\boldsymbol{z}\in\mathcal{Z}} \eta(\boldsymbol{z})$ given S.

Proof. Linearity. For a fixed subset S, averaging Eq. (6) over $\boldsymbol{z} \in \mathcal{Z}$ gives $y = \langle \boldsymbol{\beta}^{\mathcal{Z}}, \mathbf{1}_S \rangle + \bar{\eta}$ with $\bar{\eta} \coloneqq \frac{1}{|\mathcal{Z}|}\sum_{\boldsymbol{z}\in\mathcal{Z}} \eta(\boldsymbol{z})$ and $\mathbb{E}[\bar{\eta} \mid S] = 0$, because the same subset produces the loss at every $\boldsymbol{z}$ and the average commutes with the inner product. Given $\boldsymbol{\pi}$, the subset S contains $b\pi_k$ sequences drawn uniformly without replacement from $\mathcal{D}_k$, so every $s \in \mathcal{D}_k$ is included with probability $\Pr[s \in S] = b\pi_k/n_k$. By linearity of expectation,

$$\mathbb{E}[y \mid \boldsymbol{\pi}] = \sum_{k=1}^{d} \sum_{s\in\mathcal{D}_k} \beta_s^{\mathcal{Z}} \frac{b\pi_k}{n_k} = \sum_{k=1}^{d} \pi_k \frac{b}{n_k} \sum_{s\in\mathcal{D}_k} \beta_s^{\mathcal{Z}} = \langle \bar{\boldsymbol{\beta}}, \boldsymbol{\pi} \rangle.$$

Evaluating this identity at $\boldsymbol{\pi} = \boldsymbol{e}_k$ identifies $\bar{\beta}_k$ as the expected outcome of a run on domain k alone, which is feasible whenever $n_k \ge b$.

Variance. By the law of total variance and $\mathbb{E}[y \mid S] = \langle \boldsymbol{\beta}^{\mathcal{Z}}, \mathbf{1}_S \rangle$, $\mathrm{Var}(y \mid \boldsymbol{\pi}) = \mathrm{Var}(\langle \boldsymbol{\beta}^{\mathcal{Z}}, \mathbf{1}_S \rangle \mid \boldsymbol{\pi}) + \mathbb{E}[\mathrm{Var}(\bar{\eta} \mid S) \mid \boldsymbol{\pi}]$, where the second term is at most σ_η^2. The domains are sampled independently, and within domain k the sum of $\beta_s^{\mathcal{Z}}$ over a simple random sample of $b\pi_k$ out of n_k sequences has variance $b\pi_k \sigma_k^2 \frac{n_k - b\pi_k}{n_k - 1} \le b\pi_k \sigma_k^2$ by the finite population variance formula (Cochran, 1977), where the fraction is read as zero for $n_k = 1$. Summing over k bounds the first term by $b\sum_k \pi_k \sigma_k^2 \le b \max_k \sigma_k^2$. □

The linear model $f_{\boldsymbol{\beta}}(\boldsymbol{\pi}) = \boldsymbol{\beta}^\top \boldsymbol{\pi}$ fit in Eq. (2) is a datamodel whose units are domains instead of sequences, with coefficients that are budget-scaled averages of sample-level attribution scores. The partition into domains is arbitrary, since the only assumption is uniform sampling within each domain, and it affects only the noise, which stems from the randomness of training and from which sequences realize the mix, which we call the noise from sequence selection. Domains that are homogeneous in attribution reduce the latter, but their coefficients take more proxy runs to estimate, as the estimation error grows with d (Lemma 1); the finest partition recovers data attribution itself.

Corollary 2 (Domain attribution at the finest granularity is data attribution). *Let every sequence form its own domain, $d = n$, so that a run trained on a subset S of b sequences has the mix $\boldsymbol{\pi} = \mathbf{1}_S/b$. Then $\bar{\boldsymbol{\beta}} = b\,\boldsymbol{\beta}^{\mathcal{Z}}$, so the domain model of Proposition 2 reduces to the benchmark-averaged datamodel of Eq. (6): $\langle \bar{\boldsymbol{\beta}}, \boldsymbol{\pi} \rangle = \langle \boldsymbol{\beta}^{\mathcal{Z}}, \mathbf{1}_S \rangle$. The noise from sequence selection vanishes, leaving $\mathrm{Var}(y \mid \boldsymbol{\pi}) \le \sigma_\eta^2$. Further, with the equal caps $u_k = 1/b$ ($\kappa = n/b$) that let no sequence repeat within a run, the greedy fill of Eq. (3) selects a uniformly random subset of b sequences, so vertex sampling coincides with the subsampling design of Ilyas et al. (2022) with fraction b/n.*

Proof. With $n_k = 1$ for every k, Proposition 2 applies with $\mu_k = \beta_k^{\mathcal{Z}}$ and $\sigma_k^2 = 0$, and $b\pi_k \in \{0, 1\}$ forces $\boldsymbol{\pi} = \mathbf{1}_S/b$. With caps $u_k = 1/b$, the greedy fill of Eq. (3) assigns $1/b$ to the first b sequences of a uniformly random permutation, which form a uniformly random subset of size b. □

Domain attribution thus interpolates between data attribution at $d = n$ and a single undifferentiated corpus at $d = 1$, preserving the linear structure in between. Remark 1 relates domain attribution to the data values of cooperative game theory, such as the Shapley and Banzhaf values.

Remark 1 (Relation of domain attribution to Shapley and Banzhaf data values). *Data valuation scores a sequence s by a weighted average of its marginal contributions $y(S \cup \{s\}; \boldsymbol{z}) - y(S; \boldsymbol{z})$ over subsets S, with weights that define the Shapley value (Ghorbani & Zou, 2019), the Banzhaf value (Wang & Jia, 2023), or the weighted Banzhaf value (Li & Yu, 2023), which includes every other sequence in S independently with probability α. If the linear Datamodel of Eq. (6) holds with an intercept for subsets of every size, these values coincide with the attribution score $\beta_s(\boldsymbol{z})$, and by Proposition 2 the domain attribution $\bar{\beta}_k$ is b times the average value of the sequences in domain k, up to an additive constant shared by all domains. Without the linear assumption, the slope coefficients of the affine least squares approximation of $y(S; \boldsymbol{z})$ over all subsets S are the Banzhaf values (Hammer & Holzman, 1992), and under independent inclusion with probability α they are the weighted Banzhaf values with weight α (Marichal & Mathonet, 2011). Vertex sampling with equal caps resembles independent inclusion with probability about $1/\kappa$, except that every run contains the same number of domains. For nonlinear outcomes, its least squares coefficients are therefore a fixed-size analogue of the weighted Banzhaf value. While a Banzhaf value captures the effect of adding a sequence to a random training set, differences between domain coefficients capture the effect of moving budget between domains at a fixed training size, which is the question that data mixing asks.*

E.2 The design matrix governs the estimation error

By Proposition 2, the outcomes of m proxy runs satisfy $\boldsymbol{y} = \boldsymbol{\Pi}\bar{\boldsymbol{\beta}} + \boldsymbol{\xi}$ for a noise vector $\boldsymbol{\xi} \in \mathbb{R}^m$, and the attribution estimate $\hat{\boldsymbol{\beta}} = (\boldsymbol{\Pi}^\top \boldsymbol{\Pi})^{-1} \boldsymbol{\Pi}^\top \boldsymbol{y}$ is the least squares estimator of $\bar{\boldsymbol{\beta}}$. We make the standard assumptions of linear regression, writing $\boldsymbol{H}_{\boldsymbol{\Pi}} \coloneqq \frac{1}{m} \boldsymbol{\Pi}^\top \boldsymbol{\Pi}$ for the normalized information matrix, i.e., the Gram matrix of the mixes, whose inverse scales the covariance of $\hat{\boldsymbol{\beta}}$.

Assumption 1. *The noise is unbiased, $\mathbb{E}[\boldsymbol{\xi} \mid \boldsymbol{\Pi}] = \mathbf{0}$, its covariance $\boldsymbol{\Sigma}_{\boldsymbol{\Pi}} \coloneqq \mathrm{Cov}(\boldsymbol{\xi} \mid \boldsymbol{\Pi})$ satisfies $v_{\min} \boldsymbol{I}_m \preceq \boldsymbol{\Sigma}_{\boldsymbol{\Pi}} \preceq v_{\max} \boldsymbol{I}_m$ for constants $0 \le v_{\min} \le v_{\max} < \infty$, and the design has full column rank, $\mathrm{rank}(\boldsymbol{\Pi}) = d$.*

Proposition 2 provides the first two conditions for independent proxy runs, where $v_{\max} = \sigma_\eta^2 + b \max_k \sigma_k^2$ is admissible and, for homoscedastic noise $\boldsymbol{\Sigma}_{\boldsymbol{\Pi}} = \sigma^2 \boldsymbol{I}_m$, $v_{\min} = v_{\max} = \sigma^2$, while the third condition is a property of the design that we discuss below. The simplex is an affine hyperplane that avoids the origin, so d mixes can span $\mathbb{R}^d$, with one direction for the overall outcome level and $d - 1$ for contrasts between domains.

Lemma 1 (Estimation error of linear domain attribution). *Under Assumption 1, $\hat{\boldsymbol{\beta}}$ is unbiased and its mean squared error satisfies*

$$v_{\min}\,\mathrm{tr}\big((\boldsymbol{\Pi}^\top\boldsymbol{\Pi})^{-1}\big) \;\le\; \mathbb{E}\big[\|\hat{\boldsymbol{\beta}}-\bar{\boldsymbol{\beta}}\|_2^2 \mid \boldsymbol{\Pi}\big] \;\le\; v_{\max}\,\mathrm{tr}\big((\boldsymbol{\Pi}^\top\boldsymbol{\Pi})^{-1}\big) \;\le\; \frac{d}{m}\cdot\frac{v_{\max}}{\lambda_{\min}(\boldsymbol{H}_{\boldsymbol{\Pi}})}, \tag{8}$$

with equality in the first two when all runs share the same noise variance, $v_{\min}=v_{\max}=\sigma^2$.

Proof. This is the standard variance calculation for least squares. Substituting $\boldsymbol{y}=\boldsymbol{\Pi}\bar{\boldsymbol{\beta}}+\boldsymbol{\xi}$ gives $\hat{\boldsymbol{\beta}}-\bar{\boldsymbol{\beta}}=(\boldsymbol{\Pi}^\top\boldsymbol{\Pi})^{-1}\boldsymbol{\Pi}^\top\boldsymbol{\xi}$, which has zero conditional mean and expected squared norm $\mathrm{tr}((\boldsymbol{\Pi}^\top\boldsymbol{\Pi})^{-1}\boldsymbol{\Pi}^\top\boldsymbol{\Sigma}_{\boldsymbol{\Pi}}\boldsymbol{\Pi}(\boldsymbol{\Pi}^\top\boldsymbol{\Pi})^{-1})$. The semidefinite bounds on $\boldsymbol{\Sigma}_{\boldsymbol{\Pi}}$ sandwich this trace between $v_{\min}$ and $v_{\max}$ times $\mathrm{tr}((\boldsymbol{\Pi}^\top\boldsymbol{\Pi})^{-1})$, with equality for $\boldsymbol{\Sigma}_{\boldsymbol{\Pi}}=\sigma^2\boldsymbol{I}_m$. The last inequality holds as the d eigenvalues of $(\boldsymbol{\Pi}^\top\boldsymbol{\Pi})^{-1}=\frac{1}{m}\boldsymbol{H}_{\boldsymbol{\Pi}}^{-1}$ are at most $1/(m\lambda_{\min}(\boldsymbol{H}_{\boldsymbol{\Pi}}))$. □

The last inequality bounds the error by the smallest eigenvalue of $\boldsymbol{H}_{\boldsymbol{\Pi}}$, which is small when some combination of domains barely varies across the proxy runs, i.e., when the runs leave a direction of the simplex unexplored. In optimal experimental design (Kiefer & Wolfowitz, 1959; Pukelsheim, 2006), minimizing the trace in Eq. (8), maximizing this eigenvalue, and maximizing $\det(\boldsymbol{H}_{\boldsymbol{\Pi}})$ are known as the A-, E-, and D-criteria. This eigenvalue is what the condition number $\mathrm{cond}(\boldsymbol{\Pi})=\sqrt{\lambda_{\max}(\boldsymbol{H}_{\boldsymbol{\Pi}})/\lambda_{\min}(\boldsymbol{H}_{\boldsymbol{\Pi}})}$ controls: every mix sums to one, so $\boldsymbol{\Pi}\mathbf{1}_d=\mathbf{1}_m$ and $\mathbf{1}_d^\top\boldsymbol{H}_{\boldsymbol{\Pi}}\mathbf{1}_d=1$, hence $\lambda_{\max}(\boldsymbol{H}_{\boldsymbol{\Pi}})\ge 1/d$ and $1/\lambda_{\min}(\boldsymbol{H}_{\boldsymbol{\Pi}})\le d\,\mathrm{cond}(\boldsymbol{\Pi})^2$. Minimizing the condition number of $\boldsymbol{\Pi}$, as in Section 4.1, therefore controls the last bound of Eq. (8) up to the factor d. The noise level and the number of domains are fixed by the data and its partition, so the number of runs m and the design $\boldsymbol{\Pi}$ are the only two levers of a regression-based data mixing method. We next compare the designs of Dirichlet and vertex sampling on these quantities.

E.3 Pitfalls of Dirichlet sampling

Dirichlet sampling in Eq. (1) draws each mix as $\boldsymbol{\pi}_i\sim\mathrm{Dirichlet}(\gamma_i\,\bar{\boldsymbol{a}})$ around the tempered prior $\bar{\boldsymbol{a}}\coloneqq\boldsymbol{\pi}_{\text{prior}}^{1/\tau}/\|\boldsymbol{\pi}_{\text{prior}}^{1/\tau}\|_1$, with a log-uniform concentration $\gamma_i\in[\gamma_{\text{lo}},\gamma_{\text{hi}}]$. We analyze it without the caps and the rounding of $b\pi_k$ to integers, which keeps the moments in closed form, and start with the expected information matrix $\bar{\boldsymbol{H}}=\mathbb{E}[\boldsymbol{H}_{\boldsymbol{\Pi}}]$.

Lemma 2 (Information matrix of Dirichlet sampling). *Under Eq. (1),*

$$\bar{\boldsymbol{H}}\coloneqq\mathbb{E}[\boldsymbol{\pi}_i\boldsymbol{\pi}_i^\top]=c\,\mathrm{diag}(\bar{\boldsymbol{a}})+(1-c)\,\bar{\boldsymbol{a}}\bar{\boldsymbol{a}}^\top,\quad c\coloneqq 1-\frac{\ln\big((1+\gamma_{\text{hi}})/(1+\gamma_{\text{lo}})\big)}{\ln(\gamma_{\text{hi}}/\gamma_{\text{lo}})}\in(0,1). \tag{9}$$

Consequently, $\mathrm{tr}(\bar{\boldsymbol{H}}^{-1})=\frac{1}{c}\big(\sum_{k=1}^d\bar{a}_k^{-1}-(1-c)\,d\big)$ and $c\,\bar{a}_{\min}\le\lambda_{\min}(\bar{\boldsymbol{H}})\le c\,\bar{a}_{(2)}$, where $\bar{a}_{\min}$ and $\bar{a}_{(2)}$ denote the smallest and the second smallest entry of $\bar{\boldsymbol{a}}$.

Proof. Second moment. For $\boldsymbol{\pi}\sim\mathrm{Dirichlet}(\boldsymbol{\alpha})$ with $\alpha_0\coloneqq\sum_k\alpha_k$, the moments $\mathbb{E}[\pi_j\pi_k]=\frac{\alpha_j\alpha_k}{\alpha_0(\alpha_0+1)}$ for $j\ne k$ and $\mathbb{E}[\pi_k^2]=\frac{\alpha_k(\alpha_k+1)}{\alpha_0(\alpha_0+1)}$ read in matrix form as $\mathbb{E}[\boldsymbol{\pi}\boldsymbol{\pi}^\top]=\frac{\boldsymbol{\alpha}\boldsymbol{\alpha}^\top+\mathrm{diag}(\boldsymbol{\alpha})}{\alpha_0(\alpha_0+1)}$. With $\boldsymbol{\alpha}=\gamma\bar{\boldsymbol{a}}$ and $\alpha_0=\gamma$, since $\bar{\boldsymbol{a}}\in\Delta^{d-1}$,

$$\mathbb{E}[\boldsymbol{\pi}\boldsymbol{\pi}^\top\mid\gamma]=\frac{1}{1+\gamma}\,\mathrm{diag}(\bar{\boldsymbol{a}})+\frac{\gamma}{1+\gamma}\,\bar{\boldsymbol{a}}\bar{\boldsymbol{a}}^\top.$$

The two weights sum to one, so averaging over γ gives Eq. (9) with $c=\mathbb{E}_\gamma[1/(1+\gamma)]\in(0,1)$. For $\log\gamma$ uniform on $[\log\gamma_{\text{lo}},\log\gamma_{\text{hi}}]$, substituting $\gamma=e^t$ and using $\frac{\mathrm{d}}{\mathrm{d}t}[t-\ln(1+e^t)]=\frac{1}{1+e^t}$ yields the closed form

$$c=\frac{1}{\ln(\gamma_{\text{hi}}/\gamma_{\text{lo}})}\int_{\ln\gamma_{\text{lo}}}^{\ln\gamma_{\text{hi}}}\frac{\mathrm{d}t}{1+e^t}=\frac{\big[t-\ln(1+e^t)\big]_{\ln\gamma_{\text{lo}}}^{\ln\gamma_{\text{hi}}}}{\ln(\gamma_{\text{hi}}/\gamma_{\text{lo}})}=1-\frac{\ln\big((1+\gamma_{\text{hi}})/(1+\gamma_{\text{lo}})\big)}{\ln(\gamma_{\text{hi}}/\gamma_{\text{lo}})}.$$

Inverse. Let $\boldsymbol{D}\coloneqq\mathrm{diag}(\bar{\boldsymbol{a}})$, which is positive definite as $\bar{a}_k>0$, so $\bar{\boldsymbol{H}}\succeq c\boldsymbol{D}\succ 0$. The Sherman–Morrison formula (Sherman & Morrison, 1950) applied to the rank-one update $\bar{\boldsymbol{H}}=$

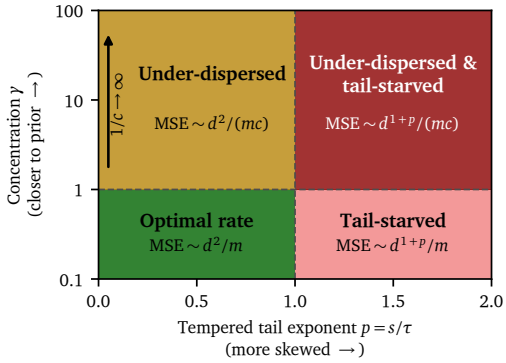


Figure 12: Two pitfalls of Dirichlet sampling. Order of the estimation error in Theorem 1 as a function of the Dirichlet concentration γ and the tempered tail exponent $p = s/\tau$ of the Zipfian prior. The concentration is fixed, so that $c = 1/(1+\gamma)$, and the boundaries at $\gamma = 1$ and $p = 1$ are schematic. For $\gamma \gg 1$, the mixes cluster around the prior, and the error grows with $1/c \approx \gamma$. For $p > 1$, the rarest domains receive too little mass, and the error grows super-quadratically in d.

$c\boldsymbol{D} + (1-c)\,\bar{\boldsymbol{a}}\bar{\boldsymbol{a}}^\top$, together with $\boldsymbol{D}^{-1}\bar{\boldsymbol{a}} = \mathbf{1}_d$ and $\bar{\boldsymbol{a}}^\top \mathbf{1}_d = 1$, gives

$$\bar{\boldsymbol{H}}^{-1} = \frac{1}{c}\boldsymbol{D}^{-1} - \frac{(1-c)\,c^{-2}\,\mathbf{1}_d\mathbf{1}_d^\top}{1+(1-c)/c} = \frac{1}{c}\boldsymbol{D}^{-1} - \frac{1-c}{c}\,\mathbf{1}_d\mathbf{1}_d^\top,$$

and taking the trace yields $\operatorname{tr}(\bar{\boldsymbol{H}}^{-1}) = \frac{1}{c}\big(\sum_k \bar{a}_k^{-1} - (1-c)\,d\big)$.

Spectrum. The lower bound follows from $\bar{\boldsymbol{H}} \succeq c\boldsymbol{D}$. For the upper bound, let k_1, k_2 index the two smallest entries of $\bar{\boldsymbol{a}}$ and pick a unit vector $\boldsymbol{w} \in \operatorname{span}(\boldsymbol{e}_{k_1}, \boldsymbol{e}_{k_2})$ orthogonal to $\bar{\boldsymbol{a}}$, which exists as orthogonality is a single linear constraint on a two-dimensional space. Then $\boldsymbol{w}^\top \bar{\boldsymbol{H}} \boldsymbol{w} = c\,(\bar{a}_{k_1} w_{k_1}^2 + \bar{a}_{k_2} w_{k_2}^2) \le c\,\bar{a}_{(2)}$, and the Rayleigh principle gives $\lambda_{\min}(\bar{\boldsymbol{H}}) \le c\,\bar{a}_{(2)}$. □

By Lemma 2, $\bar{\boldsymbol{H}}$ averages two designs with weights c and $1-c$: single-domain runs at $\boldsymbol{e}_k$ drawn with probability $\bar{a}_k$, and the tempered prior repeated in every run. These two terms lead to two pitfalls of Dirichlet sampling (Figure 12). (i) *Under-dispersion*: a large concentration keeps the mixes close to the tempered prior, so c is small and the information in every direction orthogonal to $\bar{\boldsymbol{a}}$ shrinks by the factor c. (ii) *Tail starvation*: even at $c \to 1$, rare domains with a small share $\bar{a}_k$ are seldom explored, so their attributions are estimated poorly and dominate the error, as $\operatorname{tr}(\bar{\boldsymbol{H}}^{-1})$ in Lemma 2 grows with $\sum_k \bar{a}_k^{-1}$.

To quantify the second pitfall, we model the prior mix as Zipfian (Zipf, 1949), $(\boldsymbol{\pi}_{\text{prior}})_k \propto k^{-s}$ for domains ranked by size (Figure 13). Tempering preserves this form, as $(k^{-s})^{1/\tau} = k^{-p}$, so the tempered prior is $\bar{a}_k = k^{-p}/\zeta_d(p)$ with the flatter exponent $p := s/\tau$ and the truncated zeta function $\zeta_d(p) := \sum_{k=1}^{d} k^{-p}$. Its rarest domain has the share $\bar{a}_{\min} = d^{-p}/\zeta_d(p)$, which shrinks as d grows, so by Lemma 2, $\lambda_{\min}(\bar{\boldsymbol{H}}) \ge c\,d^{-p}/\zeta_d(p)$. To bound the error of a sampled design, which depends on $\boldsymbol{H}_{\boldsymbol{\Pi}}$ rather than on $\bar{\boldsymbol{H}}$, we use the following auxiliary results.

Lemma 3 (Jensen's inequality for the inverse (Groves & Rothenberg, 1969)**).** *Let $\boldsymbol{X}$ be a random positive definite matrix with $\bar{\boldsymbol{X}} := \mathbb{E}[\boldsymbol{X}]$ positive definite. Then $\mathbb{E}[\boldsymbol{w}^\top \boldsymbol{X}^{-1} \boldsymbol{w}] \ge \boldsymbol{w}^\top \bar{\boldsymbol{X}}^{-1} \boldsymbol{w}$ for every vector $\boldsymbol{w}$, and in particular $\mathbb{E}[\operatorname{tr}(\boldsymbol{X}^{-1})] \ge \operatorname{tr}(\bar{\boldsymbol{X}}^{-1})$.*

Proof. Expanding $(\boldsymbol{X}^{-1}\boldsymbol{w} - \bar{\boldsymbol{X}}^{-1}\boldsymbol{w})^\top \boldsymbol{X}\,(\boldsymbol{X}^{-1}\boldsymbol{w} - \bar{\boldsymbol{X}}^{-1}\boldsymbol{w}) \ge 0$ gives $\boldsymbol{w}^\top \boldsymbol{X}^{-1}\boldsymbol{w} \ge 2\,\boldsymbol{w}^\top \bar{\boldsymbol{X}}^{-1}\boldsymbol{w} - \boldsymbol{w}^\top \bar{\boldsymbol{X}}^{-1}\boldsymbol{X}\bar{\boldsymbol{X}}^{-1}\boldsymbol{w}$, whose expectation is $\boldsymbol{w}^\top \bar{\boldsymbol{X}}^{-1}\boldsymbol{w}$; summing over $\boldsymbol{w} = \boldsymbol{e}_1, \ldots, \boldsymbol{e}_d$ gives the trace inequality. □

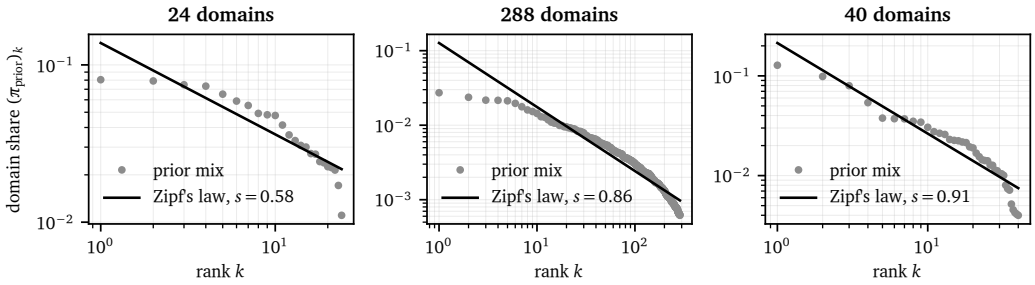


Figure 13: Prior data mixes are approximately Zipfian. Domain share against rank on log-log axes for the three setups, with least-squares fits of Zipf's law $(\boldsymbol{\pi}_{\text{prior}})_k \propto k^{-s}$ in log-log coordinates. The 288-domain setup retains the larger half of the 576 topic–format pairs, which truncates its tail.

Lemma 4 (Matrix Chernoff bound, lower tail). *Let $\boldsymbol{X}_1, \dots, \boldsymbol{X}_m$ be independent random positive semidefinite $d \times d$ matrices with $\lambda_{\max}(\boldsymbol{X}_i) \le R$ almost surely, and let $\mu_{\min} := \lambda_{\min}(\sum_{i=1}^m \mathbb{E}[\boldsymbol{X}_i])$. Then, for every $t \in [0, 1)$,*

$$\Pr\Big[\lambda_{\min}\Big(\sum_{i=1}^m \boldsymbol{X}_i\Big) \le (1-t)\,\mu_{\min}\Big] \le d\Big[\frac{e^{-t}}{(1-t)^{1-t}}\Big]^{\mu_{\min}/R} \le d\,e^{-t^2\mu_{\min}/(2R)}.$$

Proof. The first inequality is Theorem 1.1 of Tropp (2012). The second one follows from $(1-t)\ln(1-t) \ge t^2/2 - t$ for $t \in [0, 1)$, which holds as both sides vanish at $t = 0$ and the derivative of their difference, $-\ln(1-t) - t$, is nonnegative. □

Corollary 3 (Concentration of the information matrix). *Let the rows of $\boldsymbol{\Pi}$ be independent with $\|\boldsymbol{\pi}_i\|_2^2 \le R$ almost surely and a common expected information matrix $\bar{\boldsymbol{H}} := \mathbb{E}[\boldsymbol{\pi}_i \boldsymbol{\pi}_i^\top] \succ 0$, and let the noise satisfy the first two conditions of Assumption 1. For any $\delta \in (0, 1)$, if $m \ge 8R\ln(d/\delta)/\lambda_{\min}(\bar{\boldsymbol{H}})$, then with probability at least $1 - \delta$ over the draw of $\boldsymbol{\Pi}$,*

$$\lambda_{\min}(\boldsymbol{H}_{\boldsymbol{\Pi}}) \ge \tfrac{1}{2}\lambda_{\min}(\bar{\boldsymbol{H}}) \qquad \textit{and} \qquad \mathbb{E}\big[\|\hat{\boldsymbol{\beta}} - \bar{\boldsymbol{\beta}}\|_2^2 \mid \boldsymbol{\Pi}\big] \le \frac{2\,d\,v_{\max}}{m\,\lambda_{\min}(\bar{\boldsymbol{H}})}. \tag{10}$$

Proof. The summands $\boldsymbol{X}_i := \boldsymbol{\pi}_i \boldsymbol{\pi}_i^\top$ of $m\boldsymbol{H}_{\boldsymbol{\Pi}}$ are independent and positive semidefinite with $\mathbb{E}[\boldsymbol{X}_i] = \bar{\boldsymbol{H}}$ and $\lambda_{\max}(\boldsymbol{X}_i) = \|\boldsymbol{\pi}_i\|_2^2 \le R$, so Lemma 4 with $\mu_{\min} = m\lambda_{\min}(\bar{\boldsymbol{H}})$ and $t = 1/2$ bounds the probability of $\lambda_{\min}(\boldsymbol{H}_{\boldsymbol{\Pi}}) \le \frac{1}{2}\lambda_{\min}(\bar{\boldsymbol{H}})$ by $d\,e^{-m\lambda_{\min}(\bar{\boldsymbol{H}})/(8R)} \le \delta$, using the assumed lower bound on m. On the complementary event, $\lambda_{\min}(\boldsymbol{H}_{\boldsymbol{\Pi}}) > 0$, so $\boldsymbol{\Pi}$ has full column rank and the last bound of Eq. (8) gives the error bound. □

Theorem 1 (Estimation error of Dirichlet sampling under a Zipfian prior; formal**).** *Let the rows of $\boldsymbol{\Pi}$ be drawn independently by Eq. (1) with a Zipfian prior of exponent s, let the noise satisfy the first two conditions of Assumption 1, and let $p = s/\tau$ and c be as above.*

(i) For any $\delta \in (0, 1)$, if $m \ge 8\,d^p\zeta_d(p)\ln(d/\delta)/c$, then with probability at least $1 - \delta$ over the draw of $\boldsymbol{\Pi}$,

$$\mathbb{E}\big[\|\hat{\boldsymbol{\beta}} - \bar{\boldsymbol{\beta}}\|_2^2 \mid \boldsymbol{\Pi}\big] \le \frac{2\,v_{\max}}{m\,c}\,d^{1+p}\zeta_d(p). \tag{11}$$

(ii) For any $m \ge d$, in expectation over the draw of $\boldsymbol{\Pi}$,

$$\mathbb{E}\big[\|\hat{\boldsymbol{\beta}} - \bar{\boldsymbol{\beta}}\|_2^2\big] \ge \frac{v_{\min}}{m\,c}\Big(\frac{d^{1+p}\zeta_d(p)}{1+p} - (1-c)\,d\Big). \tag{12}$$

For $d \geq 2(1+p)$ and homoscedastic noise, both bounds are within constant factors depending only on p of $\frac{\sigma^2}{mc}\, d^{1+p}\zeta_d(p)$, which for fixed p grows as

$$\frac{d^{1+p}\zeta_d(p)}{m\,c} = \Theta\left(\frac{d^2}{m\,c}\cdot\begin{cases}1, & p<1,\\ \ln d, & p=1,\\ d^{\,p-1}, & p>1.\end{cases}\right) \tag{13}$$

Proof. Recall that $\bar{a}_{\min} = d^{-p}/\zeta_d(p)$ and $\lambda_{\min}(\bar{\boldsymbol{H}}) \geq c\, d^{-p}/\zeta_d(p)$.

Part (i). Every mix satisfies $\|\boldsymbol{\pi}_i\|_2^2 \leq \|\boldsymbol{\pi}_i\|_1^2 = 1$, so Corollary 3 applies with $R = 1$. Its condition on m is implied by the assumed one, as $8\ln(d/\delta)/\lambda_{\min}(\bar{\boldsymbol{H}}) \leq 8\, d^p\zeta_d(p)\ln(d/\delta)/c$, and its error bound reads $\frac{2d\,v_{\max}}{m\lambda_{\min}(\bar{\boldsymbol{H}})} \leq \frac{2\,v_{\max}}{mc}\, d^{1+p}\zeta_d(p)$.

Part (ii). For $m \geq d$, the design has full column rank almost surely, because the Dirichlet distribution has a density on the simplex, which does not contain the origin, so any d of its draws are almost surely linearly independent. Hence Eq. (8) applies almost surely, and Lemma 3 with $\boldsymbol{X} = \boldsymbol{\Pi}^\top\boldsymbol{\Pi}$ and $\bar{\boldsymbol{X}} = m\bar{\boldsymbol{H}}$ gives

$$\mathbb{E}\big[\|\hat{\boldsymbol{\beta}} - \bar{\boldsymbol{\beta}}\|_2^2\big] \geq v_{\min}\,\mathbb{E}\big[\mathrm{tr}\big((\boldsymbol{\Pi}^\top\boldsymbol{\Pi})^{-1}\big)\big] \geq \frac{v_{\min}}{m}\,\mathrm{tr}(\bar{\boldsymbol{H}}^{-1}) = \frac{v_{\min}}{mc}\Big(\zeta_d(p)\sum_{k=1}^{d}k^p - (1-c)\,d\Big),$$

using the trace formula of Lemma 2 with $\bar{a}_k^{-1} = k^p\zeta_d(p)$. The claim follows from $\sum_{k=1}^{d}k^p \geq \int_0^d x^p\,\mathrm{d}x = \frac{d^{1+p}}{1+p}$.

Rate. Since $\zeta_d(p) \geq d\cdot d^{-p}$, we have $d^{1+p}\zeta_d(p) \geq d^2$, so for $d \geq 2(1+p)$ the subtracted term satisfies $(1-c)\,d \leq d \leq \frac{d^{1+p}\zeta_d(p)}{2(1+p)}$, and the right-hand side of Eq. (12) is at least $\frac{v_{\min}}{mc}\cdot\frac{d^{1+p}\zeta_d(p)}{2(1+p)}$. Together with Eq. (11), both bounds are within the factors $\frac{1}{2(1+p)}$ and 2 of $\frac{\sigma^2}{mc}\, d^{1+p}\zeta_d(p)$ for $v_{\min} = v_{\max} = \sigma^2$. Finally, comparing the sum $\zeta_d(p)$ with the integral of x^{-p} gives $\zeta_d(p) = \frac{d^{1-p}}{1-p}(1+o(1))$ for $p<1$, $\zeta_d(p) = \ln d + O(1)$ for $p=1$, and $\zeta_d(p) = \zeta(p)(1+o(1))$ for $p>1$ with the Riemann zeta function ζ, which yields Eq. (13). □

Theorem 1 shows that, compared to the optimal error $\sigma^2 d^2/m$ on the simplex (Proposition 3 below), under-dispersion increases the error of Dirichlet sampling by a factor of order $1/c$, and tail starvation by a further factor d^{p-1} when $p>1$, i.e., when the prior is too skewed for the temperature τ to flatten (Figure 12). Both factors also appear in the lower bound of part (ii), so the two bounds match up to constants. For the uniform prior of Theorem 2, $p=0$ and only the factor $1/c$ remains. In our experiments, tail starvation is negligible: the Zipf fits in Figure 13 give $s \approx 0.6$ at 24 domains and $s \approx 0.9$ at 288 and 40 domains, so with $\tau = 2$ all three setups lie in the regime $p<1$. Under-dispersion, however, is forced by the caps. Without caps, a smaller concentration always helps, as it increases c, and the range $\gamma \in [0.1, 10]$ used at 24 domains gives $c = 0.5$. At 288 and 40 domains, small concentrations produce sparse mixes that violate the caps and are almost all rejected, so we tune the ranges with the criterion of Section 4.1 to $\gamma \in [40, 4000]$ and $[100, 10^4]$, for which $c \approx 0.005$ and 0.002. By Lemma 2 and the proof of part (ii), the expected error of the uncapped Dirichlet design is then at least $\mathrm{tr}(\bar{\boldsymbol{H}}^{-1})/d^2 \approx 2,230$, and 540 times the optimum $\sigma^2 d^2/m$ for the empirical priors and γ ranges of our three setups. Part of this gap is unavoidable, as under equal caps every design on $\mathcal{P}_{\text{cap}}$ pays a factor of about $d/(\kappa-1)$ (Theorem 2). Under the equal-cap approximation, Dirichlet sampling thus remains about 4 and 30 times worse than the best capped design at 288 and 40 domains. This matches Figure 3, where both samplers fit well at 24 domains but no surrogate fits the Dirichlet swarm at 288, and Section 4.2, where near-prior Dirichlet mixes vary HellaSwag BPB five times less than vertex mixes.

E.4 On optimal sampling

Without caps, the vertices of the simplex are the pure-domain mixes $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d$, and the greedy fill of Eq. (3) places each run at a uniformly random vertex. Allocating exactly m/d runs to each vertex

gives the balanced vertex design, the classical optimal design for first-degree mixture models (Scheffé, 1958; Kiefer, 1961). We restate it because its error $\sigma^2 d^2/m$ is the reference optimum in Appendix E.3 and Theorem 2 builds on its proof.

Proposition 3 (Classical optimality of the balanced vertex design on the simplex; formal**).** *Let m be a multiple of d, let Assumption 1 hold, and let $\mathbf{\Pi}$ be the balanced vertex design that places m/d runs at each pure-domain mix $\boldsymbol{e}_k$. Then $\mathbf{\Pi}^\top \mathbf{\Pi} = \frac{m}{d} \boldsymbol{I}_d$, each $\hat{\beta}_k$ is the average outcome of the runs on domain k, and*

$$\mathbb{E}\big[\|\hat{\boldsymbol{\beta}} - \bar{\boldsymbol{\beta}}\|_2^2 \mid \mathbf{\Pi}\big] \le \frac{d^2}{m}\, v_{\max}, \tag{14}$$

with equality for homoscedastic noise. Among all designs with m rows in Δ^{d-1}, the balanced vertex design is, up to the order of the runs, the unique design that minimizes $\operatorname{tr}((\mathbf{\Pi}^\top \mathbf{\Pi})^{-1})$, maximizes $\det(\mathbf{\Pi}^\top \mathbf{\Pi})$, maximizes $\lambda_{\min}(\mathbf{\Pi}^\top \mathbf{\Pi})$, and minimizes the worst-case prediction variance $\max_{\boldsymbol{\pi} \in \Delta^{d-1}} \boldsymbol{\pi}^\top (\mathbf{\Pi}^\top \mathbf{\Pi})^{-1} \boldsymbol{\pi}$.

Proof. Balanced vertex design. Each row of $\mathbf{\Pi}$ is a standard basis vector, so $(\mathbf{\Pi}^\top \mathbf{\Pi})_{kl}$ counts the runs placed at $\boldsymbol{e}_k$ if $k = l$ and vanishes otherwise, giving $\mathbf{\Pi}^\top \mathbf{\Pi} = \frac{m}{d} \boldsymbol{I}_d$. Then $\hat{\boldsymbol{\beta}} = \frac{d}{m} \mathbf{\Pi}^\top \boldsymbol{y}$ averages the outcomes of the runs on each domain, and $\operatorname{tr}((\mathbf{\Pi}^\top \mathbf{\Pi})^{-1}) = d^2/m$, so Eq. (14) and its equality case follow from Lemma 1.

Optimality. Let $\mathbf{\Pi}$ be any design with rows $\boldsymbol{\pi}_i \in \Delta^{d-1}$ and $\boldsymbol{M} \coloneqq \mathbf{\Pi}^\top \mathbf{\Pi}$ positive definite with eigenvalues $\lambda_1, \dots, \lambda_d > 0$; designs with a singular $\boldsymbol{M}$ are never optimal, as all four quantities are then degenerate. The diagonal of $\boldsymbol{M}$ sums to

$$\operatorname{tr}(\boldsymbol{M}) = \sum_{i=1}^{m} \|\boldsymbol{\pi}_i\|_2^2 \le \sum_{i=1}^{m} \|\boldsymbol{\pi}_i\|_1^2 = m, \tag{15}$$

with equality if and only if every row has a single nonzero entry, i.e., is a vertex. For the trace of the inverse, the Cauchy–Schwarz inequality $(\sum_j \lambda_j)(\sum_j \lambda_j^{-1}) \ge d^2$ and Eq. (15) give $\operatorname{tr}(\boldsymbol{M}^{-1}) \ge d^2/\operatorname{tr}(\boldsymbol{M}) \ge d^2/m$, with equality if and only if all eigenvalues are equal and $\operatorname{tr}(\boldsymbol{M}) = m$, i.e., $\boldsymbol{M} = \frac{m}{d} \boldsymbol{I}_d$. For the determinant, the inequality of arithmetic and geometric means gives $\det(\boldsymbol{M}) = \prod_j \lambda_j \le (\operatorname{tr}(\boldsymbol{M})/d)^d \le (m/d)^d$, with the same equality condition. For the smallest eigenvalue, $\lambda_{\min}(\boldsymbol{M}) \le \operatorname{tr}(\boldsymbol{M})/d \le m/d$, again with the same equality condition. For the worst-case prediction variance, $\boldsymbol{\pi} \mapsto \boldsymbol{\pi}^\top \boldsymbol{M}^{-1} \boldsymbol{\pi}$ is convex, so its maximum over the simplex is attained at a vertex, and $\max_k (\boldsymbol{M}^{-1})_{kk} \ge \frac{1}{d} \operatorname{tr}(\boldsymbol{M}^{-1}) \ge d/m$, with equality if and only if $\boldsymbol{M}^{-1}$ has a constant diagonal and $\operatorname{tr}(\boldsymbol{M}^{-1}) = d^2/m$, which again forces $\boldsymbol{M} = \frac{m}{d} \boldsymbol{I}_d$. Finally, $\boldsymbol{M} = \frac{m}{d} \boldsymbol{I}_d$ requires every row to be a vertex by the equality case of Eq. (15), and then $\boldsymbol{M} = \operatorname{diag}(m_1, \dots, m_d)$ with m_k runs at $\boldsymbol{e}_k$ forces $m_k = m/d$, i.e., the balanced vertex design up to the order of its rows, which attains all four bounds. □

Proposition 3 formalizes why vertex sampling works: measuring one domain at a time makes the estimates $\hat{\beta}_k$ uncorrelated for independent runs, and measuring every domain equally often gives each the same variance, at most $v_{\max} d/m$. The balanced vertex design is optimal under all classical criteria at once (A, D, E, and the worst-case prediction variance; Pukelsheim, 2006), which in general favor different designs, so the choice of criterion does not matter here. Since $\boldsymbol{\pi}_{\text{target}}$ is chosen by optimizing the prediction $\langle \hat{\boldsymbol{\beta}}, \boldsymbol{\pi} \rangle$ over mixes, the last criterion matters most for data mixing: the prediction of every mix $\boldsymbol{\pi}$ has variance at most $v_{\max} d/m$, which no other design can improve in the worst case. In practice, the caps are smaller than one, so a run cannot train on a single domain, and LinMix samples the vertices of $\mathcal{P}_{\text{cap}}$, each of which fills several domains up to their caps. Theorem 2 shows that the expected information matrix of vertex sampling is optimal on the capped simplex for a uniform prior.

Theorem 2 (Vertex sampling is optimal on the capped simplex with equal caps; formal**).** *Let the prior be uniform, so that all caps equal $u_k = \kappa/d$ with $q \coloneqq d/\kappa \in \{1, \dots, d-1\}$ an integer, and let the runs be drawn independently.*

(i) The greedy fill of Eq. (3) places each run at the cap on a uniformly random subset of q domains, and its expected information matrix $\bar{\boldsymbol{V}} := \mathbb{E}[\boldsymbol{\pi}_i \boldsymbol{\pi}_i^\top]$ has the eigenvalue $1/d$ along $\mathbf{1}_d$ and the eigenvalue $\frac{\kappa-1}{d(d-1)}$ with multiplicity $d-1$ on its orthogonal complement, so that $\operatorname{tr}(\bar{\boldsymbol{V}}^{-1}) = d + \frac{d(d-1)^2}{\kappa-1}$.

(ii) Every positive definite matrix $\boldsymbol{H}$ in the convex hull of $\{\boldsymbol{\pi}\boldsymbol{\pi}^\top : \boldsymbol{\pi} \in \mathcal{P}_{cap}\}$, which includes $\boldsymbol{H}_{\boldsymbol{\Pi}}$ for every design with rows in $\mathcal{P}_{cap}$ and the expected information matrix of every random such design, satisfies

$$\lambda_{\min}(\boldsymbol{H}) \le \frac{\kappa-1}{d(d-1)}, \qquad \operatorname{tr}(\boldsymbol{H}^{-1}) \ge d + \frac{d(d-1)^2}{\kappa-1}, \qquad \det(\boldsymbol{H}) \le \frac{1}{d}\Big(\frac{\kappa-1}{d(d-1)}\Big)^{d-1}, \tag{16}$$

with equality for $\boldsymbol{H} = \bar{\boldsymbol{V}}$, so that the expected design of vertex sampling minimizes $\operatorname{tr}(\boldsymbol{H}^{-1})$ and maximizes $\lambda_{\min}(\boldsymbol{H})$ and $\det(\boldsymbol{H})$ over $\mathcal{P}_{cap}$.

(iii) Under the noise conditions of Assumption 1, for any $\delta \in (0,1)$, if $m \ge \frac{8\kappa(d-1)}{\kappa-1}\ln(d/\delta)$, then with probability at least $1-\delta$ over the draw of $\boldsymbol{\Pi}$, $\lambda_{\min}(\boldsymbol{H}_{\boldsymbol{\Pi}}) \ge \frac{\kappa-1}{2d(d-1)}$ and

$$\mathbb{E}\big[\|\hat{\boldsymbol{\beta}} - \bar{\boldsymbol{\beta}}\|_2^2 \mid \boldsymbol{\Pi}\big] \le \frac{2\,d^2(d-1)}{m\,(\kappa-1)}\,v_{\max}. \tag{17}$$

Proof. Part (i). With equal caps $u := \kappa/d = 1/q$, the greedy fill assigns the cap to the first q domains of the permutation σ_i, which exhausts the budget as $qu = 1$, and zero to the remaining domains. The first q elements of a uniformly random permutation form a uniformly random q-subset S_i, so $\boldsymbol{\pi}_i = u\,\mathbf{1}_{S_i}$ with $\Pr[k \in S_i] = q/d$ and $\Pr[k,l \in S_i] = \frac{q(q-1)}{d(d-1)}$ for $k \ne l$. Hence

$$\bar{\boldsymbol{V}} = u^2\Big[\Big(\frac{q}{d} - \frac{q(q-1)}{d(d-1)}\Big)\boldsymbol{I}_d + \frac{q(q-1)}{d(d-1)}\,\mathbf{1}_d\mathbf{1}_d^\top\Big],$$

whose eigenvalue along $\mathbf{1}_d$ is $u^2\big(\frac{q}{d} + \frac{q(q-1)}{d}\big) = u^2q^2/d = 1/d$, and whose eigenvalue on the orthogonal complement is $u^2\,\frac{q}{d}\cdot\frac{d-q}{d-1} = \frac{d-q}{q\,d\,(d-1)} = \frac{\kappa-1}{d(d-1)}$, using $u^2q = 1/q$ and $q = d/\kappa$. The trace of $\bar{\boldsymbol{V}}^{-1}$ follows by summing the reciprocals with multiplicities 1 and $d-1$.

Part (ii). Let $\boldsymbol{H} = \sum_j w_j \boldsymbol{\pi}_j\boldsymbol{\pi}_j^\top$ be positive definite with weights $w_j \ge 0$ summing to one and $\boldsymbol{\pi}_j \in \mathcal{P}_{\text{cap}}$. Two constraints hold for every such $\boldsymbol{H}$. First, every mix sums to one, so $\mathbf{1}_d^\top\boldsymbol{H}\mathbf{1}_d = \sum_j w_j(\boldsymbol{\pi}_j^\top\mathbf{1}_d)^2 = 1$, i.e., the unit vector $\boldsymbol{v} := \mathbf{1}_d/\sqrt{d}$ satisfies $\boldsymbol{v}^\top\boldsymbol{H}\boldsymbol{v} = 1/d$. Second, for $\boldsymbol{\pi} \in \mathcal{P}_{\text{cap}}$, $\|\boldsymbol{\pi}\|_2^2 = \sum_k \pi_k^2 \le \sum_k u\,\pi_k = u$, with equality if and only if every $\pi_k \in \{0,u\}$, i.e., $\boldsymbol{\pi}$ is a vertex, so $\operatorname{tr}(\boldsymbol{H}) \le u = \kappa/d$. Let $\boldsymbol{p}_1,\dots,\boldsymbol{p}_{d-1}$ be an orthonormal basis of the orthogonal complement of $\boldsymbol{v}$, collected in $\boldsymbol{P} \in \mathbb{R}^{d\times(d-1)}$, so that $\boldsymbol{Q} := [\boldsymbol{v}\ \boldsymbol{P}]$ is orthogonal, and let $\boldsymbol{C} := \boldsymbol{P}^\top\boldsymbol{H}\boldsymbol{P}$ be the lower-right block of $\boldsymbol{Q}^\top\boldsymbol{H}\boldsymbol{Q}$. The two constraints give $\operatorname{tr}(\boldsymbol{C}) = \operatorname{tr}(\boldsymbol{H}) - \boldsymbol{v}^\top\boldsymbol{H}\boldsymbol{v} \le (\kappa-1)/d$. For the smallest eigenvalue, restricting the Rayleigh quotient of $\boldsymbol{H}$ to the range of $\boldsymbol{P}$ gives $\lambda_{\min}(\boldsymbol{H}) \le \lambda_{\min}(\boldsymbol{C}) \le \operatorname{tr}(\boldsymbol{C})/(d-1) \le \frac{\kappa-1}{d(d-1)}$. For the determinant, Fischer's inequality for the positive definite block matrix $\boldsymbol{Q}^\top\boldsymbol{H}\boldsymbol{Q}$ (Horn & Johnson, 2013, Section 7.8) and the inequality of arithmetic and geometric means give $\det(\boldsymbol{H}) = \det(\boldsymbol{Q}^\top\boldsymbol{H}\boldsymbol{Q}) \le (\boldsymbol{v}^\top\boldsymbol{H}\boldsymbol{v})\det(\boldsymbol{C}) \le \frac{1}{d}\big(\frac{\operatorname{tr}(\boldsymbol{C})}{d-1}\big)^{d-1} \le \frac{1}{d}\big(\frac{\kappa-1}{d(d-1)}\big)^{d-1}$. For the trace of the inverse, $\operatorname{tr}(\boldsymbol{H}^{-1}) = \operatorname{tr}(\boldsymbol{Q}^\top\boldsymbol{H}^{-1}\boldsymbol{Q}) = \boldsymbol{v}^\top\boldsymbol{H}^{-1}\boldsymbol{v} + \sum_{j=1}^{d-1}\boldsymbol{p}_j^\top\boldsymbol{H}^{-1}\boldsymbol{p}_j$. For any unit vector $\boldsymbol{x}$, the Cauchy–Schwarz inequality $1 = (\boldsymbol{x}^\top\boldsymbol{H}^{1/2}\boldsymbol{H}^{-1/2}\boldsymbol{x})^2 \le (\boldsymbol{x}^\top\boldsymbol{H}\boldsymbol{x})(\boldsymbol{x}^\top\boldsymbol{H}^{-1}\boldsymbol{x})$ gives $\boldsymbol{x}^\top\boldsymbol{H}^{-1}\boldsymbol{x} \ge 1/(\boldsymbol{x}^\top\boldsymbol{H}\boldsymbol{x})$. Applied to $\boldsymbol{v}$, this gives $\boldsymbol{v}^\top\boldsymbol{H}^{-1}\boldsymbol{v} \ge d$; applied to each $\boldsymbol{p}_j$ and combined with $(\sum_j x_j)(\sum_j 1/x_j) \ge (d-1)^2$ for positive x_j, it gives $\sum_j \boldsymbol{p}_j^\top\boldsymbol{H}^{-1}\boldsymbol{p}_j \ge \sum_j 1/(\boldsymbol{p}_j^\top\boldsymbol{H}\boldsymbol{p}_j) \ge (d-1)^2/\operatorname{tr}(\boldsymbol{C}) \ge \frac{d(d-1)^2}{\kappa-1}$. Finally, $\bar{\boldsymbol{V}}$ lies in the convex hull as an expectation, and by part (i) it has $\boldsymbol{v}$ as an eigenvector with eigenvalue $1/d$ and $\boldsymbol{C} = \frac{\kappa-1}{d(d-1)}\boldsymbol{I}_{d-1}$, so all three bounds hold with equality.

Part (iii). Every vertex has $\|\boldsymbol{\pi}_i\|_2^2 = qu^2 = u$, so Corollary 3 applies with $R = u = \kappa/d$ and $\lambda_{\min}(\bar{\boldsymbol{V}}) = \frac{\kappa-1}{d(d-1)}$ from part (i), for which its condition on m and its error bound are exactly those claimed. □

Theorem 2 extends Proposition 3 to the capped simplex: without caps, $q = 1$ and part (ii) reduces to the bounds of the proposition. The caps force each run to spread over $q = d/\kappa$ domains, so a run reveals only about $1/q$ as much about any single domain. This is why $\mathrm{tr}(\bar{\boldsymbol{V}}^{-1})$ in part (i) exceeds the uncapped value d^2 by a factor of about $d/(\kappa - 1)$, which is 4.5, 62, and 16 in our three setups if we treat their caps as equal. Vertex sampling is nevertheless optimal for two reasons. First, every mix sums to one, so all designs carry the same information about the overall outcome level ($1/d$ along $\mathbf{1}_d$) and differ only in the information about contrasts between domains. Vertices attain the largest norm $\|\boldsymbol{\pi}\|_2^2 = \kappa/d$ on $\mathcal{P}_{\text{cap}}$, so a design of vertices has the largest total information $\mathrm{tr}(\boldsymbol{H})$, which leaves vertex sampling the most information for these contrasts; mixes that are not vertices have a strictly smaller norm and are never optimal. Second, it spreads this information evenly across these contrasts, which is what the three quantities in Eq. (16) reward. Section 3 calls the design near-orthogonal in expectation because the expected off-diagonal entries of $\boldsymbol{\Pi}^\top \boldsymbol{\Pi}$ are only $(q-1)/(d-1) \approx 1/\kappa$ times the diagonal ones. Part (iii) shows that a sampled design is nearly optimal once m exceeds a multiple of $d \ln d$: its estimation error is at most $2d/(d-1) \approx 2$ times the smallest value any design on $\mathcal{P}_{\text{cap}}$ can achieve under homoscedastic noise. This guarantee is conservative, as its constants come from the matrix Chernoff bound and its condition on m exceeds the size of our swarms; in practice, we select the best-conditioned of many sampled designs (Appendix C), which controls the realized design directly but is not covered by the theorem. Pairing runs by reversed permutations, as in Section 3, does not affect the theorem. The reverse of a uniform permutation is uniform, so $\bar{\boldsymbol{V}}$ stays the same, and for $\kappa \geq 2$ the two runs of a pair have disjoint supports, so part (iii) still holds with the $m/2$ independent pairs as summands, each with expectation $2\bar{\boldsymbol{V}}$ and $R = u$. We leave non-uniform priors with unequal caps to future work, as they break the symmetry that gives $\bar{\boldsymbol{V}}$ its closed form.

F Limitations

A limitation shared by all regression-based methods is the cost of training a proxy swarm, which we quantify here by scaling proxy model size and budget. In practice, the cost-to-performance trade-off of data mixing becomes only more favorable as the target model size and pretraining budget increase. For example, pretraining a 4B model on 120B tokens for 2.5k GPU hours could be designed with a swarm of $512\times$ 30M models computed for 0.1k GPU hours (4% of the target cost; cf. Appendix G). Naturally, we train the target models using a single random repetition to enable more experimental setups. Our theory bounds the error of the estimated attribution $\hat{\boldsymbol{\beta}}$, not the error of the predictions made with it. Moreover, the noise level $v_{\max}$ in our bounds depends on how the corpus is split into domains, so the d^2/m rate alone cannot tell how many domains to use. Finally, whether the optimal mix is dense in positively attributed domains depends on the target benchmark, corpus size, and pretraining token budget. We thus view the curvature r that controls this as a useful utility trade-off, while the hyperparameter-free property of LinMix applies to the design of proxy-mix sampling and regression model fitting.

G Compute Resources

All models are trained on NVIDIA GH200 96GB GPUs without special optimizations beyond the default code implementations. Training one OLMo2 proxy model takes 0.2 GPU-hours for 30M parameters (1B tokens), 0.7 for 60M (2B), and 1.7 for 100M (3.2B). The Qwen3.5-like 100M proxy takes 2.6 GPU-hours on the same 3.2B tokens, 1.5 times as much as OLMo2, which we attribute to its hybrid attention with gated DeltaNet layers that add constant overhead independent of model size. Model evaluation adds further cost per proxy, depending on the benchmark and its sample size. The proxy swarms cost 0.7k GPU-hours at 30M (1024 models at 24 domains and 2048 at 288 domains), 1.4k at 60M (2048 models), 1.8k at 100M for OLMo2 (1024 models), and 2.7k at 100M for Qwen3.5 (1024 models). Pretraining a single 1B target model for 32B tokens takes 233 GPU-hours using 16 GPUs, so the 23 runs behind Table 1 and the 7 runs behind Tables 5 and 6 amount to ca. 7.0k. The 1B Qwen3.5-like target model takes 272 GPU-hours on the same 32B tokens, so the four runs behind the CPile column of Table 2 amount to ca. 1.1k. One 4B target run (80B tokens) takes 1.7k GPU-hours using 32 GPUs, so the 10 runs behind Table 2 amount to ca. 17k. Together with model evaluations, the reported experiments cost ca. 34k GPU-hours; we estimate that the whole project consumed ca. 51k GPU-hours, including preliminary work and failed jobs.