FEW-SHOT MODEL ONBOARDING FOR LLM ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Model routing reduces the inference cost of large language models (LLMs) by sending each prompt to the cheapest adequate model in a pool. Frequent model releases and costly response labeling motivate onboarding from a handful of labels, yet this setting remains largely unexplored. Recent routers can admit unseen models without retraining, but they either calibrate on a large set of randomly drawn labeled prompts or train an additional item response model to select prompts. We identify *prompt selection* and *label extrapolation* as two key factors in few-shot calibration. We propose Active Anchor Calibration (AAC), which clusters existing models' responses to select a few anchor prompts, and uses a closed-form ridge map, together with a cost prior, to extrapolate their labels to model quality and per-prompt predictions. Under a factor model with known prompt factors, we bound pool-level routing regret through an anchor design criterion. Experiments on four public benchmarks show that, with 5 labeled prompts per new model, AAC recovers 88–93% of full-calibration routing quality, against 74–83% for random calibration, which needs about 40 labels to match it.

1 INTRODUCTION

Model routing selects the cheapest adequate model for each prompt from a candidate pool (Chen et al., 2024; Ding et al., 2024; Ong et al., 2025; Hu et al., 2024). It takes advantage of differences in price, response time and capabilities to reduce inference cost by using less expensive models for prompts they can handle. New models are released frequently and older ones are retired, so routers must adapt to a changing candidate pool.

Therefore we study onboarding with a handful of labeled prompts to reduce adaptation costs. Recent routers avoid retraining by representing unseen models through calibration errors or learned response profiles (Jitkrittum et al., 2026; Feng et al., 2025; Wang et al., 2026). Building these representations shifts the cost to generating and grading responses. Existing approaches incur this cost over large random calibration sets (Jitkrittum et al., 2026) or train an additional item response model for prompt selection (Yan et al., 2026).

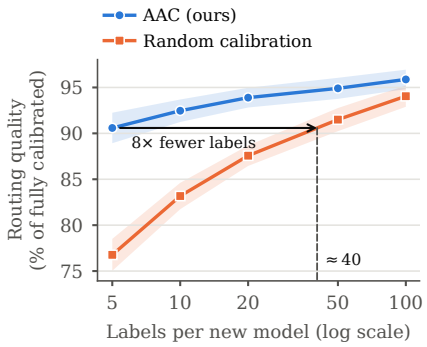

Figure 1: **Five chosen labels do the work of 40 random ones.** Routing quality, measured as area under the quality–cost curve relative to full calibration, averaged over 32 model pools from four benchmarks. Bands are 95% confidence intervals. The arrow marks the ≈ 40 random labels needed to match AAC with 5.

We focus on *prompt selection* and *label extrapolation* as two key factors in few-shot calibration. Selection matters because random prompts may poorly represent the prompt distribution. Extrapolation matters because averaging a few binary labels can misjudge overall model quality even with representative prompts. Such errors may lead a router to favor a cheap model that happened to answer its few prompts correctly. We find that this quality-estimation error dominates at small budgets. Knowing each model's overall quality alone recovers most of what random calibration loses at 5 labels, even without prompt information (§3).

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.

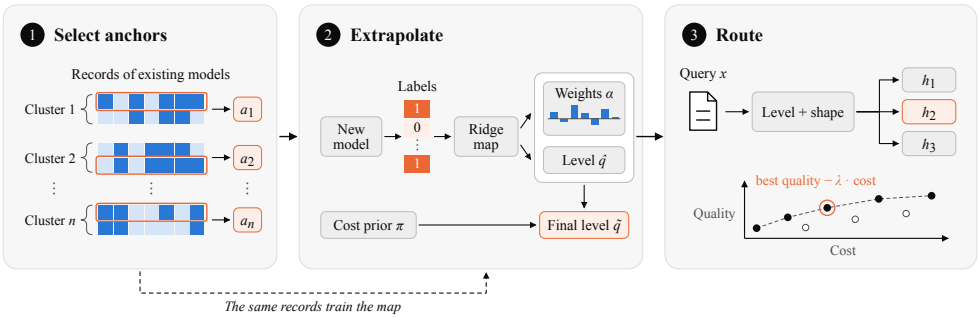

Figure 2: **Overview of Active Anchor Calibration.** (1) Each cluster of existing models' records supplies one anchor prompt. (2) The new model is labeled on the anchors; a ridge map trained on the same records gives weights over existing models and a level, stabilized by a cost prior. (3) Queries are routed by level and shape. Cells and points are schematic.

We address both factors with Active Anchor Calibration (AAC), which reuses existing models' responses (Figure 2). AAC clusters these responses to select representative *anchor* prompts and then fits a ridge-regression map in closed form from anchor labels to overall quality. A cost prior stabilizes the estimate when anchors are few. The same weights over existing models also yield per-prompt quality predictions for routing. Under a factor model with known prompt factors, we connect quality-estimation error to pool-level routing regret through an anchor design criterion (§5).

Few labels suffice to recover most of full-calibration routing quality. On four public benchmarks AAC recovers 88–93% of full-calibration routing quality with 5 labeled prompts per new model, compared with 74–83% for the best of four randomly calibrated routers (Figure 1). To characterize these gains we evaluate budgets from 5 to 100 labels and examine their dependence on the number of existing models and the information provided by cost (§6). Our contributions are as follows.

(i) We formalize few-shot onboarding of new models into a router and show that at small budgets the loss comes mainly from misjudging each model's overall quality (§3).

(ii) We propose AAC (§4), which selects anchor prompts, extrapolates their labels to model quality in closed form together with a cost prior and reuses the same weights for per-prompt routing, with a conditional factor-model analysis and formal checks in Lean 4 (§5).

(iii) We empirically show (§6) that AAC improves over random calibration of four routers from 5 to 100 labels on four benchmarks and chart where it stops working.

2 Related Work

Routing over a fixed pool. Routers trade quality against cost through pairwise selection (Ding et al., 2024; Ong et al., 2025) or answer-dependent cascades (Chen et al., 2024). A common routing framework covers both (Dekoninck et al., 2025). To evaluate this trade-off we adopt RouterBench's quality–cost curves (Hu et al., 2024). Lu et al. (2026) find that the routers they study mainly capture average model quality. Consistent with this, our analysis shows that most calibration loss on our benchmarks comes from errors in this average when labels are few (§3).

Routing with new models. Methods differ in the evidence they use and in when they calibrate. From labeled responses routers estimate per-cluster error rates (Jitkrittum et al., 2026), learn model representations (Feng et al., 2025; Wang et al., 2026; Patel et al., 2026) or retrieve similar prompts (Cao et al., 2026). Profile-based routers use model cards and reported scores without new response labels (Xu et al., 2026), whereas we actively select a small labeled set. Timing also varies. Some methods learn model embeddings offline (Li, 2025) and others adapt from online feedback (Wang et al., 2025), whereas we calibrate before deployment. Closest to this setting are methods that select anchors by optimal design and fit latent abilities through item response theory (Yan et al., 2026). We instead cluster responses and estimate overall quality through closed-form ridge regression. Under a small labeling budget noise can overestimate a cheap model's quality, so our cost prior tempers the estimate (§4).

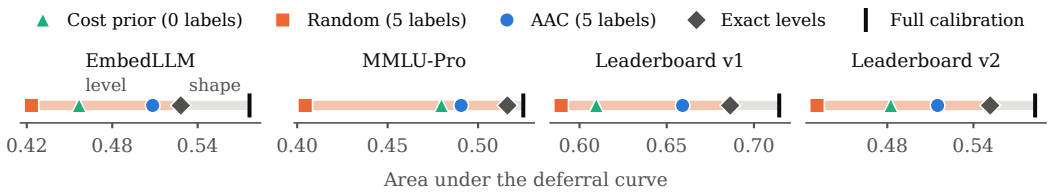

Figure 3: **With few labels most of the loss is in the levels.** Area under the deferral curve, averaged over the model pools of each benchmark (§6). From left to right, *Random* selects the best of four routers calibrated on 5 random prompts (§6), *Cost prior* predicts levels from cost without labels and *AAC* is our method (§4). *Exact levels* uses true new-model quality without prompt-specific information and *Full calibration* labels the whole pool. The orange span is the loss due to levels and the gray span the loss due to shape.

Evaluating models on few examples. This calibration also depends on which responses we collect. This choice links routing to efficient evaluation. Representative subsets capture patterns in confidence (Vivek et al., 2024) or correctness (Maia Polo et al., 2024). Other rules target item information (Kipnis et al., 2025) or model disagreement (Rubinstein et al., 2026). All of these rules use existing models to decide which examples are informative. We also rely on existing models and cluster their correctness to select anchors, but we target routing quality rather than benchmark scores. Anchor selection is thus evaluated by downstream routing performance rather than estimation error (§6).

3 Problem Setup

Dynamic routing. We adopt the setting of Jitkrittum et al. (2026). Let $\mathcal{H}_{\text{tr}} = \{h^{(1)}, \dots, h^{(M)}\}$ be the *existing* models and $\mathcal{H}_{\text{te}}$ the *new* models offered at test time, with $\mathcal{H}_{\text{tr}} \cap \mathcal{H}_{\text{te}} = \varnothing$. Each model h has a known cost $c(h) > 0$. Under the 0-1 loss the Bayes-optimal router for a cost budget is

$$r^*(\boldsymbol{x}, \mathcal{H}_{\text{te}}) = \arg\min_{h \in \mathcal{H}_{\text{te}}} \left[\gamma^*(\boldsymbol{x}, h) + \lambda \cdot c(h)\right], \qquad \gamma^*(\boldsymbol{x}, h) = \mathbb{P}\left[\boldsymbol{y} \neq h(\boldsymbol{x}) \mid \boldsymbol{x}\right], \tag{1}$$

for a multiplier $\lambda \geq 0$ set by the budget (Jitkrittum et al., 2026, Proposition 1). To implement this rule a practical router replaces γ^* with an estimate γ. Varying λ then traces a quality–cost *deferral curve*, whose area measures routing performance.

Few-shot onboarding. To estimate γ for new models, the service starts with P unlabeled calibration prompts $\{\boldsymbol{x}_j\}_{j \in [P]}$ and the response matrix $\boldsymbol{R} \in \{0,1\}^{P \times M}$ of its existing models, where $R_{jm} = 1$ if $h^{(m)}$ answers $\boldsymbol{x}_j$ correctly. We assume these responses have already been collected and graded in service, so reusing $\boldsymbol{R}$ requires no further model calls or grading. To onboard a new model h, the service chooses a set $A \subset [P]$ of n prompts, runs h on them and grades the responses to obtain $\boldsymbol{y}_A(h) \in \{0,1\}^n$. The number n is the labeling *budget*. Before these evaluations only the cost of h is known. Since A is chosen from $\boldsymbol{R}$ alone, one selection serves every new model. From $\boldsymbol{y}_A(h)$ and the existing records the service must then produce $\gamma(\boldsymbol{x}, h)$ for all prompts, not only those in A. Prior routing experiments use hundreds of randomly sampled calibration prompts (Jitkrittum et al., 2026, Appendix G.3), whereas we study how to select $n \ll P$ prompts.

Where does the loss come from? With this limited budget estimating $\gamma^*(\boldsymbol{x}, h)$ can fail in two ways. It can misjudge the *level* of a model, which is its overall quality $q(h) = \mathbb{E}_{\boldsymbol{x}}[1 - \gamma^*(\boldsymbol{x}, h)]$. It can also misjudge the *shape*, which is how the error of h varies across prompts. To separate these errors Figure 3 uses a reference router that knows every level exactly and predicts the same error rate $\gamma(\boldsymbol{x}, h) = 1 - q(h)$ for every prompt. On EmbedLLM the best router calibrated on 5 random labels attains an area of 0.423, compared with 0.528 for exact levels without shape and 0.576 for full calibration. This pattern repeats on the other benchmarks, where even levels predicted from cost alone with no labels outperform 5 random labels (Figure 3). On these benchmarks the dominant error at small budgets therefore lies in $N = |\mathcal{H}_{\text{te}}|$ scalars, one level per new model.

Guided by this observation, we spend the labels on estimating $q(h)$ as accurately as possible and take the shape from the existing models at no labeling cost. The two remaining questions are which prompts to label and how to turn a handful of labels into an estimate of $q(h)$. §4 answers both.

4 METHOD: ACTIVE ANCHOR CALIBRATION

AAC requires no router retraining. It selects anchors and extrapolates from their labels to overall quality. A cost prior stabilizes this estimate, which is then combined with the predicted per-prompt error rates of existing models. §5 analyses level estimation under a factor model.

AAC: Onboarding a New Model from n Labels

(1) **Select anchors.** Cluster the rows of $\boldsymbol{R}$ into n groups and take one representative prompt per group, giving A (§4.1).

(2) **Extrapolate.** Label the new model on A and compute its weights $\boldsymbol{\alpha}(h)$ over existing models and its quality $\tilde{q}(h)$ via (2) and (3) (§4.2).

(3) **Route.** Given a new prompt $\boldsymbol{x}$, pick the best new model via (1) with γ_{AAC} from (4) (§4.3).

4.1 SELECTING ANCHORS IN RESPONSE SPACE

Each row of $\boldsymbol{R}$ is an M-dimensional binary correctness vector. Squared Euclidean distance between rows equals Hamming distance, so nearby rows identify prompts with similar model responses. We run K-means with $K = n$ to match the labeling budget and select the prompt nearest each centroid for A. Taking one representative per cluster aims to cover the pool's response patterns. Prediction 2 in §5 quantifies the benefit when anchors match the pool's latent second moment. This applies correctness clustering (Maia Polo et al., 2024) to a calibration pool rather than a benchmark. Selection uses no new-model labels, so one anchor set serves all new models.

4.2 ESTIMATING OVERALL QUALITY FROM ANCHOR LABELS

Averaging $\boldsymbol{y}_A(h)$ treats all anchors alike, although their labels differ in how well they predict overall quality. We instead learn this relationship from existing models. Each is one training example with its n anchor labels as features and its pool average q_m as the target.

We estimate $q(h^{(m)})$ by $q_m = \frac{1}{P}\sum_j R_{jm}$ and define $\boldsymbol{q} = (q_1, \ldots, q_M)^\top$. Let $\bar{q}$ be its mean and $\bar{r}$ the scalar mean of the anchor matrix $\boldsymbol{R}_A$. With $\boldsymbol{Z} = \boldsymbol{R}_A^\top - \bar{r} \in \mathbb{R}^{M\times n}$, $\boldsymbol{z}(h) = \boldsymbol{y}_A(h) - \bar{r}$ and ridge parameter $\rho > 0$ chosen in §4.3, we set

$$\boldsymbol{\alpha}(h) = \left(\boldsymbol{Z}\boldsymbol{Z}^\top + \rho \boldsymbol{I}_M\right)^{-1} \boldsymbol{Z}\,\boldsymbol{z}(h), \qquad \hat{q}(h) = \bar{q} + \boldsymbol{\alpha}(h)^\top(\boldsymbol{q} - \bar{q}\mathbf{1}). \tag{2}$$

This is the dual solution of $\min_{\boldsymbol{w}} \|\boldsymbol{q} - \bar{q}\mathbf{1} - \boldsymbol{Z}\boldsymbol{w}\|_2^2 + \rho\|\boldsymbol{w}\|_2^2$, with prediction $\bar{q} + \boldsymbol{w}^\top \boldsymbol{z}(h)$. We center features and targets by their scalar means, fixing the prediction's intercept at $\bar{q}$. The weights $\boldsymbol{\alpha}(h)$ express the new model through existing models and also support per-prompt prediction. This dual form requires an $M \times M$ solve, and M is only a few dozen in our setting.

A cost prior for very small budgets. With $n \leq 10$, noise in (2) can overestimate a cheap model's quality and attract traffic across routing budgets. Costs are known before collecting labels, so a cost prior is available to every method. Linearly regressing q_m on $\log c(h^{(m)})$ gives a prediction $\pi(h)$ and residual variance τ^2. We estimate anchor error ν^2 by in-sample mean squared error on existing models, which is optimistic. Cross-fitting helps some settings and hurts others (Appendix C). We combine the estimates by inverse-variance weighting.

$$\tilde{q}(h) = \frac{\tau^2\,\hat{q}(h) + \nu^2\,\pi(h)}{\tau^2 + \nu^2}. \tag{3}$$

A smaller ν^2 or larger residual variance τ^2 gives the prior less weight. If both vanish, we retain $\hat{q}(h)$. Since $\pi(h)$ uses neither the anchors nor the ridge, any calibrated router can shrink its quality estimate similarly. We evaluate this step in §6.

4.3 FROM OVERALL QUALITY TO PER-PROMPT ROUTING

Using only overall quality $\tilde{q}(h)$, the router selects the same model for every prompt at a fixed λ. Let $\boldsymbol{g}(\boldsymbol{x}) \in \mathbb{R}^M$ hold the existing models' predicted error rates on $\boldsymbol{x}$, which come from a kernel ridge

regressor trained on $\mathcal{H}_{\text{tr}}$ using prompt embeddings. Since $\boldsymbol{\alpha}(h)$ already combines existing models to estimate overall quality, we apply it to $\boldsymbol{g}(\boldsymbol{x})$ to estimate prompt-specific deviations. Combining this correction with $\tilde{q}(h)$ gives

$$\gamma_{\text{AAC}}(\boldsymbol{x}, h) = \text{clip}_{[0,1]}\big(1 - \tilde{q}(h) + \beta\left[\boldsymbol{\alpha}(h)^\top\left(\boldsymbol{g}(\boldsymbol{x}) - \bar{g}(\boldsymbol{x})\mathbf{1}\right) - \mu(h)\right]\big). \tag{4}$$

Here $\bar{g}(\boldsymbol{x})$ averages $\boldsymbol{g}(\boldsymbol{x})$ over existing models and $\mu(h)$ averages $\boldsymbol{\alpha}(h)^\top(\boldsymbol{g}(\boldsymbol{x}) - \bar{g}(\boldsymbol{x})\mathbf{1})$ over the calibration pool. Subtracting $\mu(h)$ makes the correction average to zero over this pool. We scale it by $\beta \in \{0, \frac{1}{2}, 1\}$. Setting $\beta = 0$ uses only estimated levels, whereas the reference in Figure 3 uses true levels. We choose ρ, β and the hyper-parameters of $\boldsymbol{g}$ using existing models only. We split them into two halves, and each half serves in turn as held-out new models (Jitkrittum et al., 2026). The grids stay coarse because validation uses only a few dozen models.

Relation to calibrated routers. Calibrated routers estimate one error rate per prompt cluster (Jitkrittum et al., 2026) or per neighborhood (Hu et al., 2024), requiring many estimates for each model. With $n = 5$ labels and tens of clusters, most clusters receive no label. AAC spends the same labels on one scalar per model and borrows the prompt-level structure from existing models through $\boldsymbol{\alpha}(h)$. Anchor selection is also complementary to these routers because they can use the same anchors instead of random prompts. We evaluate this combination in §6, where AAC's advantage shrinks as more labels support per-cluster estimates.

Relation to item response models. Item response models place prompts and models in a latent space for routing (Song et al., 2025; Yan et al., 2026) or score estimation (Maia Polo et al., 2024). They fit each new model's position from its labels. By contrast, estimator (2) skips the latent space and regresses quality directly on the labels. This direct fit needs no iterative optimization or latent dimension, but its accuracy depends on the number of existing models, as quantified in §6.

5 Theoretical Analysis

We now analyze how estimating model quality from a handful of labels affects routing performance. Proposition 1 turns routing regret into an estimation problem. Proposition 2 bounds that error through an anchor design criterion and yields two predictions for §4.1 and §4.2. Proposition 3 accounts for learning the map from existing models. The level bounds concern a fixed calibration pool, not the gap to population quality $q(h)$. The link between the criterion and response-space K-means, as well as the effects of shape and the prior, is assessed empirically (§6). All three propositions are machine-checked in Lean 4 (Appendix B), with proofs in Appendix A.

Regret of plug-in routing. Using estimated errors in (1) incurs at most twice the largest estimation error in excess cost-adjusted risk. This standard argument makes the estimation target explicit for every cost multiplier λ. It complements the per-cluster approximation bound of Jitkrittum et al. (2026, Proposition 2), which treats cluster errors as known. Fix $\boldsymbol{x}$ and $\lambda \geq 0$ and let $\hat{h}$ minimize $\gamma(\boldsymbol{x}, h) + \lambda c(h)$ over $\mathcal{H}_{\text{te}}$.

Proposition 1. *If $|\gamma(\boldsymbol{x}, h) - \gamma^*(\boldsymbol{x}, h)| \leq \varepsilon$ for all $h \in \mathcal{H}_{\text{te}}$, then for every $h \in \mathcal{H}_{\text{te}}$,*

$$\gamma^*(\boldsymbol{x}, \hat{h}) + \lambda c(\hat{h}) \leq \gamma^*(\boldsymbol{x}, h) + \lambda c(h) + 2\varepsilon.$$

Error of the anchor estimate. To bound estimation error, we use $R_{jh} = \boldsymbol{u}_j^\top \boldsymbol{v}_h + \xi_{jh}$ as a surrogate for binary responses with linear conditional means. Here $\boldsymbol{u}_j$ and $\boldsymbol{v}_h$ are prompt and model factors, and ξ_{jh} are independent zero-mean errors with variance at most σ^2. The expected quality of h on the pool is then the linear functional $q_P(h) = \bar{\boldsymbol{u}}^\top \boldsymbol{v}_h$ with $\bar{\boldsymbol{u}} = \frac{1}{P}\sum_j \boldsymbol{u}_j$. The training target q_m in §4.2 thus differs from $q_P(h^{(m)})$ by noise of variance at most σ^2/P. We bound error relative to $q_P(h)$ using the ridge estimate with known prompt factors, $\hat{q}(h) = \bar{\boldsymbol{u}}^\top \boldsymbol{G}_\rho^{-1} \boldsymbol{U}_A^\top \boldsymbol{y}_A(h)$, where $\boldsymbol{U}_A$ stacks the factors of the anchors and $\boldsymbol{G}_\rho = \boldsymbol{U}_A^\top \boldsymbol{U}_A + \rho \boldsymbol{I}$. This model-assisted estimator follows survey sampling (Särndal et al., 1992). Its error bound depends on anchor choice only through $\kappa_\rho(A)$, the *c-optimal* design criterion for estimating $\bar{\boldsymbol{u}}^\top \boldsymbol{v}_h$ (Pukelsheim, 2006). Appendix D.1 evaluates it using SVD factors of centered $\boldsymbol{R}$ plus an intercept and compares linear and logistic factor models.

Proposition 2. *Under the factor model, for any anchor set A and $\rho > 0$,*

$$\mathbb{E}\left[\left(\hat{q}(h) - q_P(h)\right)^2\right] \leq \left(\sigma^2 + \rho\|\boldsymbol{v}_h\|^2\right) \cdot \kappa_\rho(A), \qquad \kappa_\rho(A) = \bar{\boldsymbol{u}}^\top \boldsymbol{G}_\rho^{-1} \bar{\boldsymbol{u}}.$$

Prediction 1. Selection should target mean quality (§4.1). For fixed ρ, reducing $\kappa_\rho(A)$ tightens this bound. D-optimal design instead maximizes $\det \boldsymbol{G}_\rho$ to control joint uncertainty about $\boldsymbol{v}_h$ (Yan et al., 2026). A larger determinant need not reduce error in the single quantity $\bar{\boldsymbol{u}}^\top \boldsymbol{v}_h$ needed for level-based routing. On EmbedLLM with $n = 10$ and $\rho = 1$, κ_ρ is 0.12 for our anchors, 0.15 for random prompts and 0.28 for D-optimal anchors (Appendix D.1). This diagnostic favors K-means over D-optimal anchors, but its advantage over random anchors appears only at small budgets.

Prediction 2. Extrapolation accounts for prompt difficulty (§4.2). Averaging n random labels has variance at most $(\sigma^2 + \mathrm{Var}_j[\boldsymbol{u}_j^\top \boldsymbol{v}_h])/n$. In contrast, the noise term in Proposition 2 is only $\sigma^2 \kappa_\rho(A)$, without the prompt-difficulty term. If the anchors match the second moment of the pool, $\frac{1}{n}\boldsymbol{U}_A^\top \boldsymbol{U}_A = \frac{1}{P}\sum_j \boldsymbol{u}_j \boldsymbol{u}_j^\top$, then $\kappa_\rho(A) < 1/n$. If the pool's second moment has rank greater than n, exact matching is impossible. The ridge term still controls κ_ρ at the price of bias.

Error of the learned map. These conclusions assume known factors. For labels in $[0, 1]$, the same factor model permits $\sigma^2 = \frac{1}{4}$ in Proposition 2 because conditional variance cannot exceed this value. However, (2) learns its map from M existing models. To account for this difference, write (2) as $\hat{q}(h) = a + \boldsymbol{w}^\top \boldsymbol{y}_A(h)$, the estimate of Proposition 2 as $\boldsymbol{w}_*^\top \boldsymbol{y}_A(h)$, and let $\Delta(A) = \max_{\boldsymbol{y} \in [0,1]^n} |a + (\boldsymbol{w} - \boldsymbol{w}_*)^\top \boldsymbol{y}|$ be their largest disagreement.

Proposition 3. *Under the factor model with labels in* $[0, 1]$*, if* a *and* $\boldsymbol{w}$ *do not depend on the labels of* h*, then*

$$\mathbb{E}\left[(\hat{q}(h) - q_P(h))^2\right] \le 2\left(\sigma^2 + \rho\|\boldsymbol{v}_h\|^2\right)\, \kappa_\rho(A) + 2\Delta(A)^2,$$
$$\Delta(A) \le |a| + \sqrt{n}\,\|\boldsymbol{w} - \boldsymbol{w}_*\|.$$

The added term $2\Delta(A)^2$ captures disagreement with the known-factor map. Fitting n coefficients using M models creates a trade-off examined in §6.4. The number of new models introduces another limit. Combining Propositions 1 and 2 bounds regret for the pool-level losses $1 - q_P(h)$ by a quantity that grows with N (Corollary 1, Appendix A). This dependence reflects the risk of selecting an overestimated weak model when comparing many noisy estimates. The cost prior in (3) aims to reduce this risk by shrinking noisy levels toward a cost-based prediction, as evaluated in §6.

6 Experiments

We test whether AAC turns a small labeling budget into better routing. After describing the setup (§6.1), we compare it with random calibration (§6.2), isolate its components (§6.3) and examine when its gains shrink (§6.4).

6.1 Setup

Benchmarks. We use four binary-correctness benchmarks. These are EmbedLLM (Zhuang et al., 2025) (112 models, 35,673 prompts) and three from RouterEval (Huang et al., 2025): MMLU-Pro (Wang et al., 2024) (12,032 prompts), Open LLM Leaderboard v1 (61 tasks, 27,842 prompts) and Open LLM Leaderboard v2 (38 tasks, 9,574 prompts). Before running experiments, we uniformly sample 112 models per RouterEval benchmark whose name states the parameter count. This count proxies serving cost on all four benchmarks (Jitkrittum et al., 2026).

Protocol. Following Jitkrittum et al. (2026), each *model pool* jointly splits models randomly into existing (74) and new (38), and prompts $60\%/10\%/30\%$ into training (for predictor $\boldsymbol{g}$ in (4)), calibration and test. Prompts are embedded as described in Appendix C. We use 8 pools per benchmark (32 total) and 20 calibration draws per pool for each budget $n \in \{5, 10, 20, 50, 100\}$. Budgets are shared, though anchor deduplication can yield fewer labels (Appendix C). Routing quality is area under the test accuracy–cost curve for new models, with cost normalized by the most expensive model. We integrate trapezoids between exact routing transitions over $\lambda \ge 0$ (Appendix D.8). The 95% intervals use Student's t over pool means after averaging calibration draws. The method and configuration were fixed before pool creation. Two EmbedLLM development pools are excluded from the main evaluation and reported in Appendix D.5.

Method	EmbedLLM		MMLU-Pro		Leaderboard v1		Leaderboard v2	
	QNC ↓	Area ↑	QNC ↓	Area ↑	QNC ↓	Area ↑	QNC ↓	Area ↑
Mean-only (Hu et al., 2024)	100.0	0.414	100.0	0.390	99.2	0.586	99.0	0.412
K-NN (Hu et al., 2024; Shnitzer et al., 2024)	100.0	0.410	100.0	0.382	99.2	0.580	99.0	0.406
Cluster-based (Jitkrittum et al., 2026)	100.0	0.410	100.0	0.390	99.2	0.578	99.3	0.411
Learned cluster map (Jitkrittum et al., 2026)	99.8	0.423	99.9	0.404	98.3	0.589	98.1	0.431
Random calibration	99.8	0.423	99.9	0.404	98.3	0.589	98.1	0.431
Random calibration + cost prior	99.7	0.467	100.0	0.476	97.6	0.627	92.6	0.494
AAC without cost prior	100.0	0.475	100.0	0.445	97.0	0.618	99.0	0.438
AAC	**98.0**	**0.508**	**98.1**	**0.491**	**92.5**	**0.659**	**82.7**	**0.515**

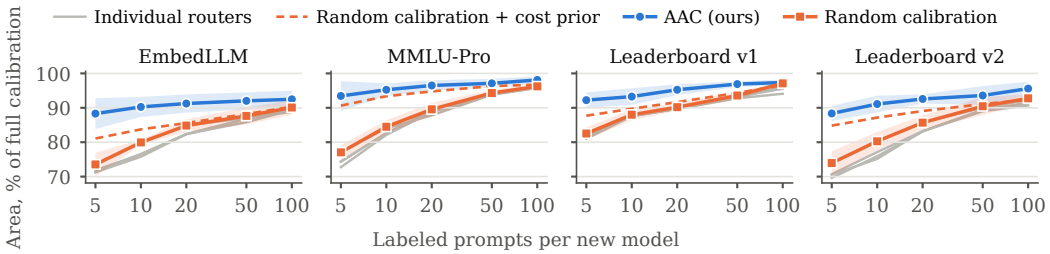

Figure 4: **AAC leads on average at every budget, even when baselines use the cost prior.** **Top.** Quality-neutral cost (QNC, %) and area with 5 labels per new model, averaged over 8 pools per benchmark. Bold marks the best per column. Random calibration selects the best of the four by area per pool and budget. Its values match the learned cluster map after rounding. **Bottom.** Area as a percentage of full calibration. Means and 95% confidence intervals over pools. Gray lines show individual routers. The dashed line adds the cost prior to Random calibration.

Methods compared. Randomly calibrated baselines comprise a mean-only router (Hu et al., 2024), K-NN (Hu et al., 2024; Shnitzer et al., 2024), and routers using K-means or clusters learned from existing models (Jitkrittum et al., 2026). Their best per pool and budget is *random calibration*, an optimistic baseline chosen with hindsight. We also give each router our cost prior and the cluster-based router our anchors. AAC ablations remove components or change anchor selection. Experiments run on a laptop CPU, and onboarding takes milliseconds.

6.2 Main Results

AAC improves over random calibration when labels are scarce. AAC has larger area in 157 of 160 pool–budget cases (Figure 4). All exceptions occur at 100 labels with relative differences below 2%. Relative area gains over random calibration are 11.8–21.5% at 5 labels and 5.6–8.1% at 20. They are smallest on Leaderboard v1, which has the weakest cost–quality correlation (Appendix D.6). With 5 labels, AAC reaches 88–93% of full-calibration area, against 74–83% for random calibration. Matching this takes about 40 random labels, 8 times as many on average (Figure 1). Random calibration with the cost prior closes the gap most on MMLU-Pro, where cost best predicts quality, but AAC still leads in 151 of 160 cases by 3.1–8.9% at 5 labels (Appendix D.8).

The gains persist across pools and calibration draws. At 5 labels, the learned cluster map leads the baselines in 30 of 32 pools. Since this router relies most on existing models, its strength supports AAC's reuse of their records. To examine performance beyond these averages, we treat each calibration draw within a pool as one run. Across these runs, AAC's fifth-percentile area is 82% of full-calibration area, against at most 54% for these baselines (Appendix D.8).

AAC also lowers the cost of matching the best model. Quality-neutral cost (QNC) is the minimum relative cost matching the most accurate model (Jitkrittum et al., 2026). Where area summarizes the full curve, QNC focuses on cost at this quality target. Runs that never reach it count as 100%. At 5 labels, random calibration rarely reaches this target, so its QNC stays near 100% (Figure 4). AAC matches it in 34% of runs, against at most 5%, and has the lowest QNC on every benchmark. Full-calibration QNC matches the published result (Appendix D.8).

Table 1: **Where the gain comes from, on EmbedLLM.** Relative area gain (%) over random calibration. Means over 8 pools. Blocks compare estimators on shared anchors, assess prior and shape, then vary selection. D-optimal with item response omits prior and shape, matching the first-block item response configuration to isolate selection.

Labeled prompts	Estimator	Prior	Shape	5	10	20	50	100
Ours	Mean	–	–	8.2	3.8	3.3	1.9	−0.2
Ours	Cluster-weighted mean	–	–	10.2	7.0	3.4	1.4	−0.5
Ours	Item response model	–	–	9.0	5.2	3.6	1.5	−1.1
Ours	Ridge	–	–	9.7	6.7	4.2	1.9	−0.5
Ours	Ridge	✓	–	18.0	10.3	5.3	2.0	−0.5
Ours	Ridge	–	✓	12.4	9.8	6.6	4.8	2.6
Ours (AAC)	Ridge	✓	✓	20.2	12.9	7.5	5.0	2.6
Random	Ridge	–	✓	3.4	3.4	3.1	3.1	2.9
Random	Ridge	✓	✓	13.4	8.0	5.0	3.9	3.0
D-optimal	Item response model	–	–	−4.5	−3.1	−0.5	−0.8	−2.0
D-optimal	Ridge	✓	✓	12.1	6.4	4.2	2.0	1.2
Max. disagreement	Ridge	✓	✓	18.1	11.0	6.6	5.2	2.0

6.3 Sources of the gain

Selection and extrapolation drive gains. We rebuild AAC in the order of §4 (Table 1, first two blocks, and Appendix D.4). When we change only *which* prompts are labeled, the mean over our anchors gains 8.2% at 5 labels. Extrapolation (2) raises this to 9.7% and the cost prior to 18.0%. These routing gains accompany lower estimation error (Figure 5). With 5 labels, root mean squared error falls from 0.21 for random labels to 0.13–0.16 for the mean over anchors and 0.06–0.08 with extrapolation, the accuracy that random labels reach only with 50 to 100 prompts. This pattern is consistent with Proposition 2. Shape adds 2 to 3 percentage points of relative gain at 5 labels with or without the prior. At 100 labels, all gains come from shape.

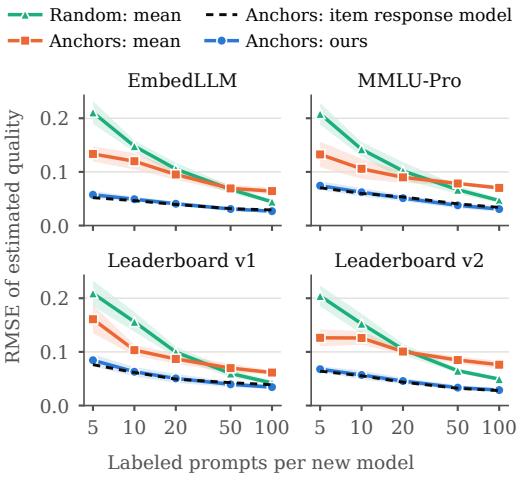

Figure 5: **Anchors with extrapolation estimate quality from 5 labels as well as random labels do from 50 to 100.** Root mean squared error of estimated new-model quality against test-prompt quality. Means and 95% confidence intervals over model pools with $\rho = 10$. The item response model uses 2 latent factors and penalty 0.1.

The estimator matters less than the anchors. With random prompts, ridge with shape gains 3.4% at 5 labels against 12.4% on our anchors, or 13.4% against 20.2% with the prior (Table 1, last block). This selection advantage closes by 100 labels. An item response model fitted to existing models (Maia Polo et al., 2024; Yan et al., 2026) estimates quality from our anchors as accurately as ridge (Figure 5). This suggests that selection and extrapolation matter more than the estimator. We retain ridge for its closed form and weights that carry over to shape.

The cost prior corrects a selection effect. It adds 8 percentage points of relative gain at 5 labels but almost nothing from 50 labels on, while changing average estimation error by at most 0.008. The benefit lies in the models selected for routing. Without the prior, quality on the estimated cost–quality frontier is overestimated by up to 0.08, against 0.02 for the average model. This illustrates the risk of selecting overestimated models underlying Corollary 1. With the prior, the excess is at most 0.03, and the frontier holds 5.5 models instead of 3.4 (Appendix D.6). Since the prior is independent of anchor selection, Figure 4 compares methods both with and without it.

Representative anchors suit routing. Using our anchors also improves the cluster-based router, narrowing its gap to AAC (Appendix D.2). To compare selection rules alone, we hold ridge, prior and shape fixed (Table 1, last block). D-optimal anchors lose 8 percentage points of relative gain at 5 labels, remain behind our anchors at every budget and do worse than random prompts. Selecting prompts with greatest disagreement among existing models (Rubinstein et al., 2026) is more competitive. It trails our anchors in 17 of 20 benchmark–budget combinations (Appendix D.3), so choosing between these rules matters less than selecting at all.

6.4 Limits of the gain

More labels close the gap. AAC replaces new-model labels with existing-model records, so more labels should reduce its advantage. We extend the budget on EmbedLLM development pools to 2,000 labels. The gain falls to about 5% at 200 labels and 2% at 1,000 and is gone at 2,000 (Appendix D.5). Direct calibration thus closes the gap.

Fewer existing models weaken the map. Reducing M on EmbedLLM to 20 keeps the relative gain within about 2 percentage points of that with the full pool at every budget (Figure 6). With $M = 10$ it halves at 5 labels, falls below 2% at 20 and turns negative from 50 as the ridge map has many inputs relative to its M training examples. This is consistent with the map-estimation error captured by $\Delta(A)$ in Proposition 3. MMLU-Pro follows the same pattern from 10 labels (Appendix E). With only 6 existing models on RouterBench (Hu et al., 2024), AAC and random calibration are within 1.4% of each other (Appendix E).

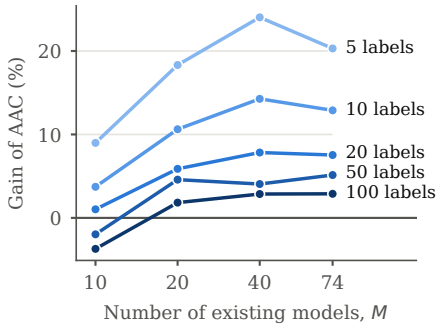

Figure 6: **Small budgets tolerate few existing models.** Gain of AAC over the better cluster-based router on EmbedLLM (8 pools) when only M existing models are available. Each line is one budget.

Gains transfer across organizations. Each organization is assigned wholly to one side of the split, so its models cannot appear on both sides. At 5 labels, the gains are 15.7–19.2% across the four benchmarks, against 11.8–21.5% under the random split (Appendix D.7). Taken together, these results support AAC for a model's first few dozen labels in pools with a few dozen graded models.

7 Conclusion

Onboarding a new model into a router is increasingly costly due to frequent changes in the model pool. One source of this cost is calibration, which estimates model performance from labeled responses. When this involves hundreds of prompts per new model, each addition becomes expensive, especially when responses need human grading.

We propose AAC, which uses existing models' graded responses to select anchors and estimate a new model's overall quality from their labels. A cost prior stabilizes the estimate, while weights over existing models support per-prompt routing. AAC achieves 90.6% of full-calibration routing quality on average across four benchmarks with five labels per model, matching random calibration with about 40 labels. This enables services to integrate new models with little labeling effort and no router retraining when existing evaluation records are available.

Limitations. The main limitation of AAC is its reliance on informative existing-model records. Few models or prompts weaken the map while weak cost predictiveness reduces the prior's benefit (Appendix E). A rate in M for Proposition 3's map-estimation term remains open. Chronological splits (Rubinstein et al., 2026) could test whether earlier records remain useful for later releases.

Our evaluation is also limited to binary correctness and parameter-count costs. Extending it to translation or summarization where quality is graded rather than binary would require consistent scoring of existing and new models' responses. Combining these scores with measured inference costs would better reflect practical service conditions. We leave such extensions for future work.

Reproducibility statement

Each experiment is defined by one configuration file and writes its configuration, input hashes, a snapshot of the source code, per-trial results and timing to its output folder. Every table and figure in this paper is generated from these folders by a script. Dataset preparation, including model sampling and re-embedding, is scripted. It records hashes of inputs and outputs (Appendix C). Anchor selection is deterministic given the seed and a rerun with a different thread count reproduces all trials to floating-point precision. Proofs are in Appendix A and the Lean 4 sources are included in the supplementary material.

AI use statement

In this work, we used generative AI tools to assist with method implementation, experiment design and execution, interpretation of results, formulation of mathematical claims and proof writing and formalization in Lean 4. We did not use them to generate synthetic data or to clean datasets. Translation and qualitative analysis are not applicable. We also used them for software development, drafting and polishing the manuscript, figure preparation and literature search. We have reviewed all AI-assisted work, verified the code by testing and the proofs with Lean 4 and take responsibility for the final content of this work including text, claims and artifacts produced with generative AI.

References

Qi Cao, Shuhao Zhang, Ruizhe Zhou, Ruiyi Zhang, Peijia Qin, and Pengtao Xie. Models under SCOPE: Scalable and controllable routing via pre-hoc reasoning. In *International Conference on Machine Learning*, 2026.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. A unified approach to routing and cascading for LLMs. In *International Conference on Machine Learning*, 2025.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*, 2024.

Tao Feng, Yanzhen Shen, and Jiaxuan You. GraphRouter: A graph-based router for LLM selections. In *International Conference on Learning Representations*, 2025.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. RouterBench: A benchmark for multi-LLM routing system. *arXiv preprint arXiv:2403.12031*, 2024.

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. RouterEval: A comprehensive benchmark for routing LLMs to explore model-level scaling up in LLMs. In *Findings of the Association for Computational Linguistics: EMNLP*, 2025.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. Universal model routing for efficient LLM inference. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/113ae3a9762ca2168f860a8501d6ae25-Abstract-Conference.html.

Alex Kipnis, Konstantinos Voudouris, Luca M. Schulze Buschoff, and Eric Schulz. metabench – a sparse benchmark of reasoning and knowledge in large language models. In *International Conference on Learning Representations*, 2025.

Yang Li. LLM bandit: Cost-efficient LLM generation via preference-conditioned dynamic routing. *arXiv preprint arXiv:2502.02743*, 2025.

Yifan Lu, Qiyue Zhang, Shenrun Zhang, Zhibo Yu, Zhuang Wang, Hanjie Chen, and Jiarong Xing. The routing plateau: Understanding and breaking the accuracy limits of LLM routers. *arXiv preprint arXiv:2606.07587*, 2026.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinyBenchmarks: evaluating LLMs with fewer examples. In *International Conference on Machine Learning*, 2024.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data. In *International Conference on Learning Representations*, 2025.

Shivam Patel, William Cocke, and Gauri Joshi. LOCUS: Low-dimensional model embeddings for efficient model exploration, comparison, and selection. *arXiv preprint arXiv:2601.21082*, 2026.

Friedrich Pukelsheim. *Optimal Design of Experiments*. SIAM, 2006.

Alexander Rubinstein, Benjamin Raible, Martin Gubri, and Seong Joon Oh. DISCO: Diversifying sample condensation for efficient model evaluation. In *International Conference on Learning Representations*, 2026.

Carl-Erik Särndal, Bengt Swensson, and Jan Wretman. *Model Assisted Survey Sampling*. Springer, 1992.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets. In *Conference on Language Modeling*, 2024.

Wei Song, Zhenya Huang, Cheng Cheng, Weibo Gao, Bihan Xu, GuanHao Zhao, Fei Wang, and Runze Wu. IRT-Router: Effective and interpretable multi-LLM routing via item response theory. In *Annual Meeting of the Association for Computational Linguistics*, 2025.

Rajan Vivek, Kawin Ethayarajh, Diyi Yang, and Douwe Kiela. Anchor points: Benchmarking models with much fewer examples. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2024.

Chenxu Wang, Hao Li, Yiqun Zhang, Linyao Chen, Jianhao Chen, Ping Jian, Qiaosheng Zhang, and Shuyue Hu. ICL-Router: In-context learned model representations for LLM routing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. MixLLM: Dynamic routing in mixed large language models. In *Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2025.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, 2024.

Jingjun Xu, Hongji Pu, Tao Feng, Haozhen Zhang, Jiaxuan You, and Ge Liu. RouteProfile: Graph-based profiling for cold-start LLM routing. *arXiv preprint arXiv:2605.00180*, 2026.

Cheng Yan, Wuyang Zhang, Zhiyuan Ning, Fan Xu, Ziyang Tao, Lu Zhang, Bing Yin, and Yanyong Zhang. Breaking model lock-in: Cost-efficient zero-shot LLM routing via a universal latent space. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. EmbedLLM: Learning compact representations of large language models. In *International Conference on Learning Representations*, 2025.

Few-Shot Model Onboarding for LLM Routing

Supplementary Material

TABLE OF CONTENTS

A Proofs **13**

B Machine-checked statements **14**

C Data preparation **14**

D Additional results **14**

D.1 Anchor design criterion 14

D.2 Routing on shared anchors 15

D.3 Anchor selection rules 15

Direct criterion minimization 16

D.4 Components on all benchmarks 16

D.5 Development pools and large budgets 16

D.6 What the cost prior corrects 16

D.7 Held-out organizations and close relatives 17

D.8 Routing quality and reliability 18

E Boundary cases **18**

A PROOFS

Proof of Proposition 1. Write $f^*(h) = \gamma^*(\boldsymbol{x}, h) + \lambda c(h)$ and $f(h) = \gamma(\boldsymbol{x}, h) + \lambda c(h)$. Their cost terms cancel, so $|f(h) - f^*(h)| \le \varepsilon$ for every h. Since $\hat{h}$ minimizes f,

$$f^*(\hat{h}) \le f(\hat{h}) + \varepsilon \le f(h) + \varepsilon \le f^*(h) + 2\varepsilon. \qquad \square$$

Proof of Proposition 2. We bound bias and noise separately. Substituting $\boldsymbol{y}_A(h) = \boldsymbol{U}_A \boldsymbol{v}_h + \boldsymbol{\xi}_A$ and using $\boldsymbol{G}_\rho^{-1}\boldsymbol{U}_A^\top \boldsymbol{U}_A = \boldsymbol{I} - \rho \boldsymbol{G}_\rho^{-1}$ gives

$$\hat{q}(h) - q_P(h) = -\rho\, \bar{\boldsymbol{u}}^\top \boldsymbol{G}_\rho^{-1} \boldsymbol{v}_h + \bar{\boldsymbol{u}}^\top \boldsymbol{G}_\rho^{-1} \boldsymbol{U}_A^\top \boldsymbol{\xi}_A.$$

The first term is deterministic and the second has zero mean, so mean squared error separates into squared bias and variance.

Bias. Cauchy–Schwarz in the inner product induced by $\boldsymbol{G}_\rho^{-1}$ yields

$$\left(\bar{\boldsymbol{u}}^\top \boldsymbol{G}_\rho^{-1} \boldsymbol{v}_h\right)^2 \le \kappa_\rho(A)\, \boldsymbol{v}_h^\top \boldsymbol{G}_\rho^{-1} \boldsymbol{v}_h \le \frac{\kappa_\rho(A)}{\rho} \|\boldsymbol{v}_h\|^2,$$

where the last inequality follows from $\boldsymbol{G}_\rho \succeq \rho \boldsymbol{I}$.

Variance. Since $\boldsymbol{U}_A^\top \boldsymbol{U}_A \preceq \boldsymbol{G}_\rho$,

$$\bar{\boldsymbol{u}}^\top \boldsymbol{G}_\rho^{-1} \boldsymbol{U}_A^\top \boldsymbol{U}_A \boldsymbol{G}_\rho^{-1} \bar{\boldsymbol{u}} \le \bar{\boldsymbol{u}}^\top \boldsymbol{G}_\rho^{-1} \bar{\boldsymbol{u}} = \kappa_\rho(A).$$

Combining these bias and variance bounds gives

$$\mathbb{E}\left[(\hat{q}(h) - q_P(h))^2\right] \le \rho \|\boldsymbol{v}_h\|^2 \kappa_\rho(A) + \sigma^2 \kappa_\rho(A) = \left(\sigma^2 + \rho \|\boldsymbol{v}_h\|^2\right) \kappa_\rho(A). \qquad \square$$

Proof of Proposition 3. Let $e = \boldsymbol{w}_*^\top \boldsymbol{y}_A(h) - q_P(h)$ be the error of the ideal map and $d = a + (\boldsymbol{w} - \boldsymbol{w}_*)^\top \boldsymbol{y}_A(h)$ its disagreement with the learned map. Then $\hat{q}(h) - q_P(h) = e + d$. Since $(e+d)^2 \le 2e^2 + 2d^2$ and $|d| \le \Delta(A)$,

$$\begin{aligned}\mathbb{E}\left[(\hat{q}(h) - q_P(h))^2\right] &\le 2\mathbb{E}[e^2] + 2\Delta(A)^2 \\ &\le 2\left(\sigma^2 + \rho \|\boldsymbol{v}_h\|^2\right) \kappa_\rho(A) + 2\Delta(A)^2.\end{aligned}$$

The last step applies Proposition 2, with $\Delta(A)$ fixed given the existing models.

For the norm bound, every $\boldsymbol{y} \in [0,1]^n$ satisfies $\|\boldsymbol{y}\| \le \sqrt{n}$. Thus Cauchy–Schwarz gives

$$\Delta(A) = \max_{\boldsymbol{y} \in [0,1]^n} \left| a + (\boldsymbol{w} - \boldsymbol{w}_*)^\top \boldsymbol{y} \right| \le |a| + \sqrt{n}\, \|\boldsymbol{w} - \boldsymbol{w}_*\|. \qquad \square$$

Representative anchors. Write $\boldsymbol{\Sigma} = \frac{1}{P}\sum_j \boldsymbol{u}_j \boldsymbol{u}_j^\top = \boldsymbol{C} + \bar{\boldsymbol{u}}\bar{\boldsymbol{u}}^\top$, where $\boldsymbol{C} \succ 0$ is the factor covariance. With $s = \bar{\boldsymbol{u}}^\top \boldsymbol{C}^{-1} \bar{\boldsymbol{u}}$, the Sherman–Morrison formula gives $\bar{\boldsymbol{u}}^\top \boldsymbol{\Sigma}^{-1} \bar{\boldsymbol{u}} = s/(1+s) < 1$. Exact moment matching gives $\boldsymbol{U}_A^\top \boldsymbol{U}_A = n\boldsymbol{\Sigma}$, hence $\boldsymbol{G}_\rho \succeq n\boldsymbol{\Sigma}$ and

$$\kappa_\rho(A) \le \frac{1}{n} \bar{\boldsymbol{u}}^\top \boldsymbol{\Sigma}^{-1} \bar{\boldsymbol{u}} < \frac{1}{n}.$$

Corollary 1. *Let $N = |\mathcal{H}_{\mathrm{te}}|$ and $\|\boldsymbol{v}_h\| \le B$ for all h. A router that uses $\gamma(\boldsymbol{x}, h) = 1 - \hat{q}(h)$ has expected excess cost-adjusted risk at most $2\sqrt{N(\sigma^2 + \rho B^2)\, \kappa_\rho(A)}$ relative to the best router for the pool-level losses $1 - q_P(h)$.*

Proof of Corollary 1. Apply Proposition 1 with γ^* replaced by $1 - q_P(h)$ and set $\varepsilon = \max_h |\hat{q}(h) - q_P(h)|$. The expected regret is at most $2\mathbb{E}[\varepsilon]$. Jensen's inequality and Proposition 2 give

$$\mathbb{E}[\varepsilon] \le \left(\sum_h \mathbb{E}\left[(\hat{q}(h) - q_P(h))^2\right] \right)^{1/2} \le \sqrt{N\left(\sigma^2 + \rho B^2\right) \kappa_\rho(A)}.$$

Multiplying by two proves the claim using only second moments. Improving this coarse bound to logarithmic dependence on N would require stronger tail control. $\square$

B MACHINE-CHECKED STATEMENTS

The Lean 4 development checks Proposition 1, the known-factor mean squared error bound in Proposition 2, the expectation and norm bounds for the fixed affine map in Proposition 3, and Corollary 1. To check these statements, it constructs the regularized Gram matrix and derives the bias, variance and expectation bounds. It also proves the representative-anchor rate under exact moment matching and the scalar rank-one inverse identity. All statements compile without `sorry` using only Lean's standard axioms. These checks neither establish moment matching for K-means nor verify the Python implementation. See `proofs/README.md` for the theorem mapping, analytic assumptions and build instructions.

C DATA PREPARATION

We retain EmbedLLM's released 768-dimensional embeddings. For RouterEval, we read its pickle files of scores, prompts and embeddings with an unpickler restricted to NumPy arrays and built-in containers. We drop tasks with non-binary scores (TruthfulQA in Leaderboard v1) and retain models whose repository names contain exactly one size in billions of parameters and no mixture marker. From these eligible models we draw 112 with a fixed seed. The released embeddings are identical across prompts because a shared few-shot prefix exceeds the encoder's input length. We therefore remove each task's longest shared prefix and embed the remaining question with frozen `all-mpnet-base-v2`.

Anchor selection. We run Euclidean K-means on the binary rows of $\boldsymbol{R}$ with $K = n$, k-means++ initialization, three restarts and a fixed seed, on a single thread so that the result does not depend on the machine. We take the row nearest each centroid as its anchor and remove duplicates without replacement. As a result, AAC uses between $n - 4$ and n distinct labels in the main evaluation. Random-calibration baselines use n labels, while anchor-based controls share the same selected set.

Hyper-parameters. We fit the kernel ridge predictor $\boldsymbol{g}$ once per pool on training prompts. We select all hyper-parameters using existing models only, measuring area as each half serves as held-out new models. This selects $\rho \in \{1, 10, 100\}$, $\beta \in \{0, \frac{1}{2}, 1\}$ and the kernel parameters of $\boldsymbol{g}$ for AAC, or cluster or neighbor counts per budget for other routers.

Noise estimate. Table 2 replaces the in-sample ν^2 of (3) by a cross-fitted one, in which each half of the existing models is predicted from the other half. The resulting increase in prior weight helps some benchmarks and budgets but hurts others, so we retain the simpler estimate.

Table 2: Relative gain (%) of AAC over random calibration with the in-sample and the cross-fitted estimate of ν^2.

Benchmark, variant	5	10	20	50	100
EmbedLLM, in-sample ν^2	20.2	12.9	7.5	5.0	2.6
EmbedLLM, cross-fitted ν^2	19.6	12.6	7.5	5.0	3.0
MMLU-Pro, in-sample ν^2	21.5	12.8	7.8	3.0	1.9
MMLU-Pro, cross-fitted ν^2	22.4	12.0	8.2	3.4	2.3
Leaderboard v1, in-sample ν^2	11.8	6.0	5.6	3.6	0.3
Leaderboard v1, cross-fitted ν^2	10.0	6.1	5.3	3.6	0.7
Leaderboard v2, in-sample ν^2	19.7	13.6	8.1	3.5	3.0
Leaderboard v2, cross-fitted ν^2	20.2	13.3	7.8	3.9	3.0

D ADDITIONAL RESULTS

D.1 ANCHOR DESIGN CRITERION

Figure 7 reports $\kappa_\rho(A)$ on the 8 EmbedLLM pools, with prompt factors taken from a rank-4 truncated SVD of the centered response matrix plus an intercept. D-optimal anchors are worst throughout, while our anchors beat random prompts at 5 and 10 labels for small ρ. This advantage disappears by 20 labels or when $\rho = 10$ makes ridge dominate the Gram matrix.

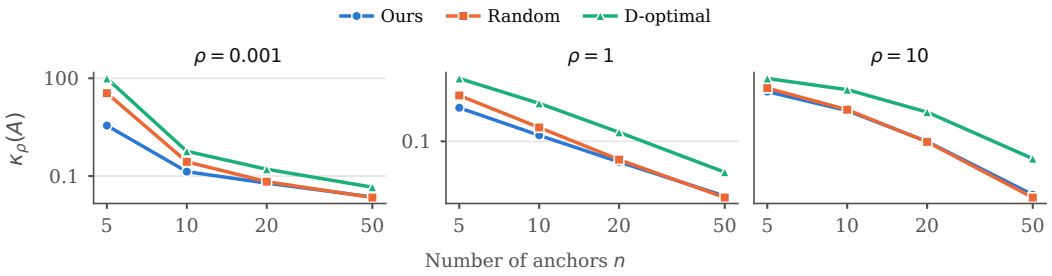

Figure 7: Design criterion $\kappa_\rho(A)$ on EmbedLLM (lower is better) for three selection rules. Means over pools and 5 draws per pool.

We compare linear and logistic factor models at the same dimension as a diagnostic of the surrogate in §5. We fit factors on half of the existing models, then estimate the remaining models' loadings from the responses used to score their reconstruction. Across the four benchmarks, the linear-minus-logistic difference in explained variance ranges from -1.3 to 0.6 percentage points with 2 factors; 3.4 to 4.6 percentage points with 5 factors. Because fitting and scoring share responses, this measures reconstruction rather than unseen-response prediction and does not establish the noise assumptions of Proposition 2.

D.2 Routing on shared anchors

Giving the cluster-based router the same anchors separates prompt selection from how the labels are used. Table 3 reports the resulting gains of AAC, with the range over model pools to show how the comparison varies across splits.

Table 3: Relative gain (%) of AAC over the cluster-based router given our anchors. Means over pools, with the range in brackets.

Labels	EmbedLLM	MMLU-Pro	Leaderboard v1	Leaderboard v2
5	11.8 [3.6, 28.0]	13.2 [2.1, 29.1]	9.8 [0.0, 25.7]	37.2 [6.8, 99.0]
10	9.0 [1.4, 19.1]	7.5 [0.6, 19.9]	3.8 [0.2, 8.3]	8.2 [4.2, 11.2]
20	4.1 [−0.5, 8.2]	3.7 [−0.1, 8.6]	2.2 [−0.8, 4.6]	5.3 [1.2, 8.9]
50	3.7 [1.1, 7.7]	1.7 [0.1, 3.3]	0.7 [−0.9, 2.4]	3.1 [−0.1, 8.1]
100	3.3 [0.8, 5.5]	1.6 [0.4, 3.1]	0.3 [−0.6, 1.9]	4.2 [1.7, 9.6]

D.3 Anchor selection rules

For disagreement-based selection, we take the n prompts with largest $p_j(1 - p_j)$, where p_j is the fraction of existing models that answer $\boldsymbol{x}_j$ correctly. This is the binary-label counterpart of the disagreement score of Rubinstein et al. (2026), which requires class probabilities that our benchmarks do not provide. Figure 8 compares this rule with D-optimal selection and our anchors.

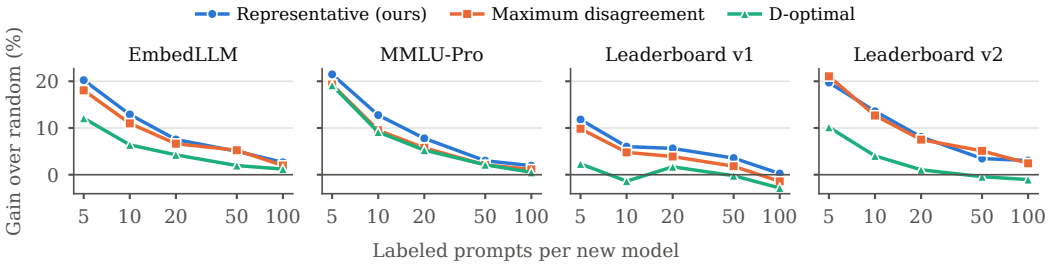

Figure 8: Mean relative gain (%) over random calibration for three anchor selection rules, each followed by the same extrapolation.

Direct criterion minimization. We also ask whether selecting anchors directly to reduce κ_ρ improves routing. In this follow-up analysis, prompt factors are formed from the existing models' responses after subtracting each prompt's mean. A greedy selector replaces the K-means anchors only when it lowers κ_ρ at the same realized label count. We test these anchors alone and with a ridge estimator in the same factors, using its weights for level and shape. Table 4 compares both variants within full AAC. We first retain the original settings, then reselect ρ and β separately for each variant and for AAC, using disjoint halves of the existing models in turn. Each search excludes candidate anchors from validation and keeps teacher settings fixed. Neither variant consistently improves routing, so we retain response-space clustering.

Table 4: Mean relative change in routing area (%) over four benchmarks, eight pools per benchmark and five budgets, with 20 evaluation draws per setting. Fixed settings compare against original AAC and reselected settings compare against AAC with the same source-only search. Each entry averages the relative change in pool-mean area across the 160 settings.

Replacement	Fixed settings	Reselected settings
Anchors only	-0.22	-0.11
Anchors and estimator	-0.08	+0.03

For the aligned estimator, a fixed-setting failure on Leaderboard v2 with 5 labels illustrates the distinction. Quality-estimation MSE falls by 26.3%, yet routing area falls by 15.7%. The cheaper model has true accuracy 49.9% against 60.3% for the more expensive model, but their estimates are 51.0% and 50.1%, respectively. The cheaper model therefore dominates the better model in the estimated frontier. Lower average estimation error need not preserve routing decisions.

D.4 Components on all benchmarks

Figure 9 repeats the rows of Table 1 that were run on every benchmark. Across these benchmarks, anchors matter most, the prior helps most at 5 labels and shape supplies the gain at 100.

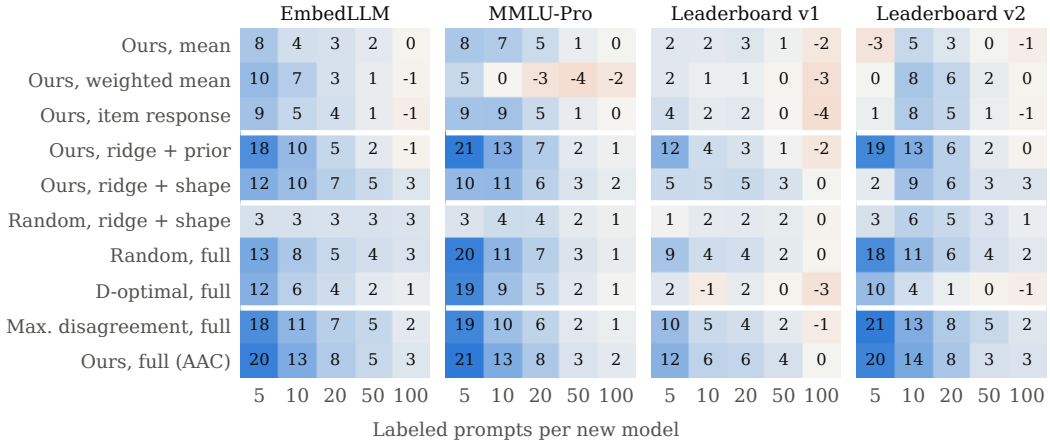

Figure 9: Relative gain (%) over random calibration for combinations of labeled prompts and estimator, by number of labeled prompts. *Full* is ridge with prior and shape.

D.5 Development pools and large budgets

The two EmbedLLM pools used during development are separate from the evaluation pools. On these pools, gains over the better cluster-based router at 200, 500, 1,000 and 2,000 labels are $4.2/5.5$, $3.5/2.8$, $1.7/2.1$ and $0.6/-0.6$%, respectively. The advantage is therefore negligible at 2,000 labels, with a small loss on one pool.

D.6 What the cost prior corrects

A router that knows only the levels uses the models on the estimated cost–quality frontier. Figure 10 reports how many models that frontier holds and how far their quality is overestimated with 5 labels

and $\rho = 10$. Although mean error over all new models is at most 0.02, routing without the prior selects overestimated models. This is the selection effect behind $\sqrt{N}$ in Corollary 1. On EmbedLLM this excess is small, so the prior mainly restores models to a frontier reduced to about three.

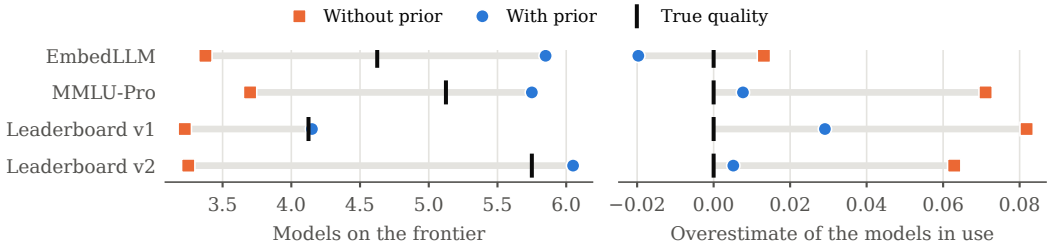

Figure 10: Models on the estimated cost–quality frontier, and mean error of the estimated quality of the models in use (estimate minus truth), with 5 labels. Means over pools and 5 draws per pool.

The strength of this correction depends on how well cost predicts quality. Table 5 relates the quality variance explained by regressing on $\log c(h)$ to AAC's gain with and without the prior. On EmbedLLM and Leaderboard v1, cost explains little, so (3) assigns the prior little weight and it adds least. It adds more where cost explains more and lowers the gain on no benchmark. To test this dependence, we perturb only the prior costs, holding routing and evaluation costs fixed. On EmbedLLM, multiplying every cost by log-normal noise with $\sigma = 1$ leaves the gain at 5 labels at 20.7% against 20.2% with the published costs, and permuting the costs across models leaves 13.7%, close to the 12.4% of the variant that carries no prior at all. These perturbations isolate sensitivity to the information supplied to the prior.

Table 5: Variance in the quality of existing models explained by $\log c(h)$ (%), and relative gain (%) of AAC over random calibration at 5 labels without and with the cost prior.

Benchmark	Explained by cost	Gain without prior	Gain with prior
EmbedLLM	18	12.4	20.2
MMLU-Pro	59	10.4	21.5
Leaderboard v1	14	4.8	11.8
Leaderboard v2	39	1.8	19.7

D.7 Held-out organizations and close relatives

A random split can place two models of one organization, such as two sizes of one family, on either side. We therefore draw eight further pools on each benchmark in which the organization named in each model identifier falls wholly among the existing or wholly among the new models. This assignment uses no model responses. Figure 11 compares the gain of AAC under the two splits against the routers as published, where AAC is ahead in 153 of 160 combinations of pool and budget. Against the same routers given the cost prior it gains 1.6–10.4% at 5 labels. These comparisons span 56 to 94 organizations per 112 models. Because identifiers need not reveal families and release dates are unavailable, the split separates prefixes rather than families or release times.

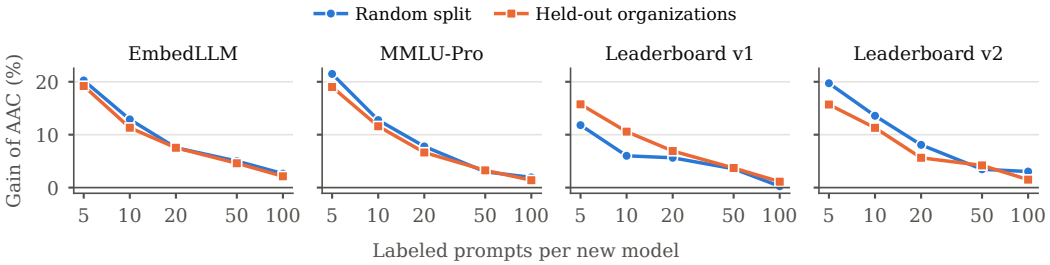

Figure 11: Relative gain (%) of AAC over random calibration under random and organization-held-out model splits, across four benchmarks. Means over 8 pools per benchmark and split.

To complement this identifier-based split with a behavioral diagnostic, we remove existing models whose training-response correlation with any new model is at least 0.7, including similar models from different organizations, and rerun all methods. Despite removing 9–42 existing models per pool, mean gains over random calibration at 5 labels remain 8.5–22.1% across the four benchmarks. Because this diagnostic uses additional new-model responses, it is neither an onboarding protocol nor a test of ancestry. The full curves remain in the experiment artifacts.

D.8 Routing quality and reliability

Area integrates piecewise-linear mean accuracy against relative cost over $[c_{\min}/c_{\max}, 1]$, with a horizontal extension after the last switch and no interval-length normalization. To compute this area, we group switches at the same λ and retain choices on both sides. Equal-cost prediction ties use stored model order and ties at $\lambda = 0$ favor lower cost. This integration is independent of the plotting grid. As a reproduction check, our fully calibrated cluster-based implementation attains QNC of 34.2% on EmbedLLM against the published 33.7% (Jitkrittum et al., 2026). Tables 6 and 7 report relative gains and absolute areas. Figure 12 reports best-model attainment and fifth-percentile area relative to full calibration.

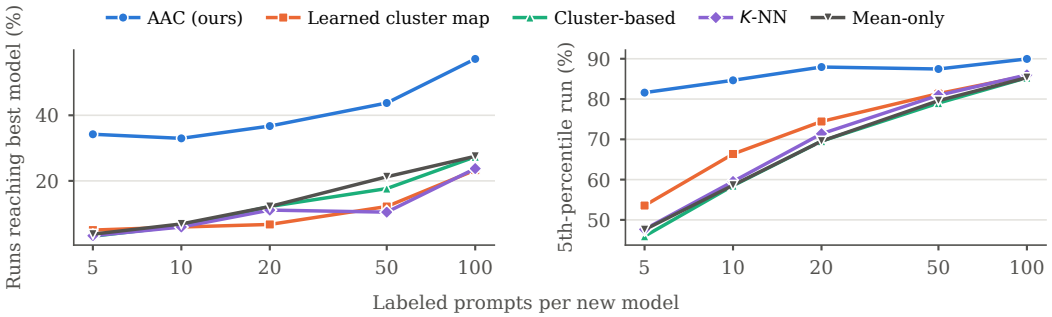

Figure 12: The left panel shows runs (%) whose deferral curve reaches the quality of the most accurate new model. The right panel shows the fifth percentile over runs of the area as a percentage of full calibration. Both pooled over the four benchmarks.

Table 6: Relative gain (%) of AAC in area under the deferral curve. Means over pools, with the range over pools in brackets.

Labels	EmbedLLM	MMLU-Pro	Leaderboard v1	Leaderboard v2
Against the best of four routers calibrated on random prompts				
5	20.2 [7.8, 30.0]	21.5 [3.0, 30.9]	11.8 [6.4, 20.3]	19.7 [10.6, 26.4]
10	12.9 [4.9, 16.7]	12.8 [8.1, 16.0]	6.0 [0.2, 10.6]	13.6 [8.8, 19.1]
20	7.5 [5.7, 10.5]	7.8 [3.4, 12.1]	5.6 [0.8, 8.0]	8.1 [4.7, 10.7]
50	5.0 [1.9, 7.7]	3.0 [1.1, 6.7]	3.6 [2.1, 5.6]	3.5 [0.4, 6.0]
100	2.6 [0.4, 4.5]	1.9 [0.6, 3.4]	0.3 [−1.8, 1.6]	3.0 [1.6, 5.7]
Against the same four routers given the cost prior				
5	8.9 [3.1, 17.9]	3.1 [−9.4, 8.3]	5.2 [1.5, 8.0]	4.3 [0.7, 8.7]
10	7.8 [5.2, 13.1]	2.0 [−0.7, 5.5]	3.9 [0.2, 7.4]	4.6 [1.3, 7.1]
20	6.6 [4.3, 9.5]	1.8 [0.1, 3.0]	3.9 [1.6, 5.2]	4.0 [1.8, 7.2]
50	4.4 [2.5, 8.4]	0.8 [−1.7, 2.3]	2.7 [0.3, 4.7]	2.8 [−0.3, 4.6]
100	2.4 [0.1, 4.2]	1.4 [0.0, 2.6]	0.9 [−0.9, 3.0]	3.6 [1.9, 6.6]

E Boundary cases

The benchmarks below test what happens when AAC lacks enough existing models, informative costs or a sufficiently large calibration pool.

Few existing models. Table 8 follows the gain as the number of existing models falls from 74 to 10, while the following benchmarks extend this comparison to 6. The binary responses of 6 existing models take between 47 and 60 distinct patterns on RouterBench, which caps the number of distinct

Table 7: Area under the deferral curve, mean over pools.

Benchmark, method	5	10	20	50	100
EmbedLLM, random calibration	0.423	0.461	0.489	0.505	0.519
EmbedLLM, random calibration with cost prior	0.467	0.482	0.493	0.508	0.520
EmbedLLM, AAC without cost prior	0.475	0.505	0.521	0.529	0.532
EmbedLLM, AAC	0.508	0.520	0.525	0.530	0.533
MMLU-Pro, random calibration	0.404	0.444	0.470	0.495	0.505
MMLU-Pro, random calibration with cost prior	0.476	0.490	0.498	0.506	0.508
MMLU-Pro, AAC without cost prior	0.445	0.490	0.500	0.510	0.514
MMLU-Pro, AAC	0.491	0.500	0.507	0.510	0.515
Leaderboard v1, random calibration	0.589	0.628	0.644	0.669	0.694
Leaderboard v1, random calibration with cost prior	0.627	0.641	0.655	0.674	0.690
Leaderboard v1, AAC without cost prior	0.618	0.657	0.676	0.690	0.696
Leaderboard v1, AAC	0.659	0.666	0.681	0.692	0.696
Leaderboard v2, random calibration	0.431	0.468	0.499	0.527	0.541
Leaderboard v2, random calibration with cost prior	0.494	0.508	0.519	0.531	0.538
Leaderboard v2, AAC without cost prior	0.438	0.510	0.531	0.542	0.556
Leaderboard v2, AAC	0.515	0.531	0.540	0.545	0.557

anchors at any budget. On this benchmark (6 existing and 3 new models per pool, 10 pools, 100 to 500 labels) the gain of AAC is $+0.5$, -0.2, -0.9, -1.3 and -1.4%. On a mathematics benchmark with 6 existing and 4 new models (10 pools, 5 to 100 labels) it is -5.4, -3.4, -4.0, -1.5 and -3.8%. There cost does not track quality and the variant without prior is within 1.7% of random calibration except at 20 labels (-2.8%). This dependence on existing models also appears within EmbedLLM. With the map fitted on all 74 existing models standing in for the unknown $\boldsymbol{w}_*$, the quantity $\sqrt{n}\|\boldsymbol{w} - \boldsymbol{w}_*\|$ over the eight pools falls from 0.16 at $M = 10$ to 0.06 at $M = 40$ with 5 labels and grows by a factor of 2.0 from 5 to 50 labels, consistent with the directions of Proposition 3, without quantifying its bound because the reference map is also estimated.

Small pool and preference scores. On AlpacaEval 2.0 (805 prompts, 79 models, about 80 pool prompts) the gain ranges from -21.4 to $+12.2\%$ across pools with no consistent sign. Here a router that ignores the prompt still trails the best method by 2.0% on average even with full calibration, and by 4.3% on the worst pool.

A failure at five labels. With $M = 40$ on one development pool, 14 of 20 draws gave an area of 0.230 against 0.49 for the others. In these failed draws, two anchors were answered by no or nearly all existing models, leaving three informative labels. The fifth-cheapest new model (true quality 0.229) was estimated at 0.507, the highest of all. This overestimate sent every prompt to that model. No such failure occurs on MMLU-Pro or in the 160 runs on eight held-out EmbedLLM pools with $M = 40$.

Table 8: Sensitivity to the number M of existing models on EmbedLLM (8 pools, plotted in Figure 6) and MMLU-Pro (4 pools). Mean relative gain (%) over the better cluster-based router, with the worst pool in brackets.

Benchmark, M	5	10	20	50	100
EmbedLLM, 74	20.3 [7.8]	12.9 [4.9]	7.5 [5.7]	5.1 [1.9]	2.9 [1.1]
EmbedLLM, 40	24.0 [13.8]	14.3 [7.9]	7.8 [5.6]	4.1 [0.5]	2.9 [0.8]
EmbedLLM, 20	18.3 [8.4]	10.6 [−2.1]	5.9 [0.5]	4.6 [3.8]	1.8 [−0.5]
EmbedLLM, 10	9.0 [−0.4]	3.7 [−6.1]	1.0 [−7.4]	−2.0 [−9.6]	−3.7 [−12.1]
MMLU-Pro, 74	23.7 [13.7]	13.6 [11.2]	8.3 [5.5]	4.1 [2.8]	2.2 [1.9]
MMLU-Pro, 40	20.4 [3.0]	12.7 [11.7]	8.6 [6.5]	3.8 [1.1]	1.2 [0.7]
MMLU-Pro, 20	22.1 [17.4]	14.8 [11.9]	7.0 [3.7]	1.6 [−2.1]	−0.5 [−3.2]
MMLU-Pro, 10	19.7 [15.6]	8.2 [−0.1]	1.3 [−10.1]	−3.6 [−10.4]	−1.4 [−8.3]