THE ROUTING PLATEAU: UNDERSTANDING THE ACCURACY LIMITS OF LLM ROUTERS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM routing has become a popular approach to improve the cost–quality trade-off of LLM services by adaptively selecting a model for each query. Recent work has explored a broad range of routing methods, including clustering-based routers, learned classifiers, pairwise ranking, and confidence-based approaches. Our extensive study of 21 routing methods across five benchmarks reveals a consistent phenomenon that we call the **routing plateau** (Fig. 1): *many methods, including kNN, achieve very similar accuracy and converge to a narrow performance range that remains far below the oracle router.* Our analysis supports a *correctness-prediction bottleneck* hypothesis: current routers primarily learn global-average model performance trends rather than fine-grained, query-specific routing signals. As a result, they collectively fail on queries that require instance-specific routing decisions. Moreover, to understand whether the plateau can be alleviated with a better training setup, we construct a 300K-query benchmark (Nine-by-300k). More data, larger encoders, and end-to-end fine-tuning improve eight routers by 1.24 pp on average, but leave the plateau largely intact. These findings suggest that further progress may require inputs beyond the query itself, such as partial output trajectories that reveal how models attempt the task.

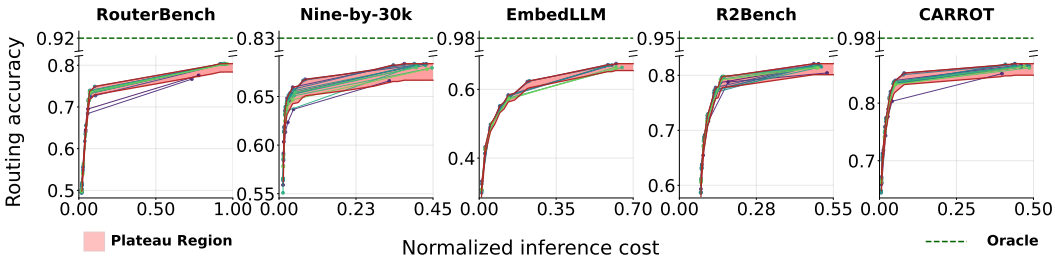


Figure 1: Routing accuracy vs. normalized inference cost. All routers from heterogeneous method families collapse onto a narrow *plateau* (red band) well below the per-query oracle (green dashed).

1 INTRODUCTION

Query routing across models has become a common approach for improving the cost–quality trade-off of LLM services. An LLM router dynamically selects an appropriate model for each query, routing simpler queries to smaller, cheaper, or more specialized models while reserving expensive frontier models for harder queries (Hu et al., 2024; Somerstep et al., 2025; Ong et al., 2025; Xue et al., 2026). Recent work has proposed a wide range of routing methods, including similarity search over training examples (Lu et al., 2023; Somerstep et al., 2025), learned classifiers or regressors for model correctness prediction (Ong et al., 2025; Sakota et al., 2024; Liu et al., 2024; Not Diamond, 2025), pairwise model ranking (Frick et al., 2025; Zhao et al., 2024), and confidence- or lookahead-based methods (Ding et al., 2024; Woisetschläger et al., 2025; Zhang et al., 2025b). Each new routing method typically reports advantages over prior routers, suggesting steady progress in router design. However, because these studies often differ in model pools, benchmarks, training data, and evaluation protocols (Lu et al., 2025), it remains unclear whether these reported gains reflect fundamental improvements in routing capability or artifacts of evaluation setup.

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.

In this paper, we evaluate 21 routing methods across five representative routing benchmarks. Our study reveals a phenomenon we call the **routing plateau**: *despite their diverse designs, many routers, including kNN-style routers, converge to the same accuracy ceiling, which remains far below that of the oracle router* (Lai & Ye, 2026). We unpack this plateau through three more specific empirical observations. **(1) Similar top-end accuracy:** Although a few strong routers outperform other methods, their accuracies are highly similar and converge to a narrow performance band. For example, on RouterBench, the top 15 routers differ by only 0.23 percentage points (pp) in accuracy. **(2) Strong kNN-style routers:** simple similarity-based approaches such as kNN remain consistently competitive with more sophisticated routers across benchmarks. For instance, kNN ranks among the top-2 methods on all five benchmarks. **(3) A persistent oracle gap:** all existing routers remain far below the oracle router, showing that the key challenge is identifying the correct model from the query. For example, the best router trails the Oracle by 10–30 pp across the five benchmarks.

Our further analysis supports a hypothesis of the **correctness-prediction bottleneck**: Routers must predict which models will answer a query correctly, but often struggle to learn this signal from available data and query representations. Specifically, current routers tend to learn coarse, global patterns of model capability rather than fine-grained, instance-level differences in correctness. For many queries, however, model correctness does not always align with overall capability: models with higher average performance are not consistently the most reliable on individual instances. As a result, although routers may differ in routing decisions, their shared reliance on global-model-capability signals causes them to converge to a narrow accuracy ceiling far below that of the oracle router. This also explains why kNN-style routers remain competitive: simple similarity-based methods can already capture much of this coarse routing signal.

Finally, we ask whether the plateau reflects limitations of the training regime. To test that, we construct Nine-by-300k, expanding the training set by 10 times to 302,371 labeled queries. We examine data scaling, larger query encoders, and end-to-end fine-tuning (Chen et al., 2024; Feng et al., 2025) across eight router designs. Combining these interventions improves accuracy by 1.24 pp on average and up to 2.13 pp, with most of the gain coming from more data and fine-tuning. Yet routers remain closely clustered and far below the oracle. These interventions improve how routers learn from queries, but still require them to predict correctness before observing any model response. These findings suggest that further progress may require richer routing inputs, such as partial output trajectories that reveal how models attempt the query.

In summary, this paper makes the following contributions:

- We evaluate 21 LLM routing methods across five benchmarks under a unified experimental setup and reveal the routing plateau phenomenon.
- We provide evidence for a correctness-prediction bottleneck behind the plateau: current routers tend to learn coarse, global patterns of model capability rather than fine-grained routing signals.
- We construct Nine-by-300k, a routing benchmark with 300k training queries and 2.8 million query–model correctness labels, enabling us to test the role of data scaling, encoder capacity, and end-to-end fine-tuning in the plateau.
- We show that these training interventions yield modest gains while leaving the plateau largely intact, motivating routing inputs that provide evidence of model behavior beyond the query alone.

2 Related Work

LLM routing methods. Recent routers fall into a few design families: *classifier-style* predictors of per-model correctness (Ong et al., 2025; Sakota et al., 2024; Liu et al., 2024; Not Diamond, 2025; Jin et al., 2025), *retrieval* routers that copy the best neighbor from training (Lu et al., 2023; Somerstep et al., 2025), *pairwise-ranking* routers (Frick et al., 2025; Zhao et al., 2024), *latent-factor* and *IRT* models (Zhuang et al., 2024; Song et al., 2025), and *contrastive* routers that pull a query toward the model answering it correctly (Chen et al., 2024; Lai & Ye, 2026). Cascades, orchestration, bandit-style, multi-round, and online-serving routers (Ding et al., 2024; Zhang et al., 2023; 2025b; Woisetschläger et al., 2025; Li et al., 2010; Zhang et al., 2025a; Wu & Silwal, 2025) sit at the edge of the single-shot setting and serve as reference points. Each new method is usually evaluated against a small set of baselines on a single benchmark under its own encoder and experimental setup.

Routing benchmarks and analysis. Several public benchmarks are available: RouterBench (Hu et al., 2024), EmbedLLM (Zhuang et al., 2024), R2-Bench (Xue et al., 2026), CARROT/SPROUT (Somerstep et al., 2025), RouterArena (Lu et al., 2025), and LLMRouterBench (Li et al., 2026). On the analysis side, Lai & Ye (2026) show that trained routers tend to collapse onto a small set of preferred models; Li et al. (2026) report that many recent and even commercial routers fail to reliably outperform a simple baseline; and Li (2025) argue that a well-tuned frozen-embedding kNN already matches state-of-the-art learned routers across diverse tasks. Our work offers a deeper analysis of the routing plateau, understanding its reasons and identifying potential paths forward.

3 Preliminaries and Problem Setup

LLM Routing. A routing instance is a pair $(\mathcal{Q}, \mathcal{P})$, where $\mathcal{Q}$ is a query distribution and $\mathcal{P} = \{1, \dots, K\}$ is a fixed pool of K candidate LLMs. Each $(q_i, m) \in \mathcal{Q} \times \mathcal{P}$ is associated with a binary correctness label $Y_{i,m} \in \{0, 1\}$ and a real-valued per-call cost $c_{i,m} \geq 0$. A router is a function r that maps a query to a model, $\hat{m}_i = r(q_i) \in \mathcal{P}$. Most routers decompose into a *predictor* that produces scores $\hat{p}_{i,m} \in \mathbb{R}$ approximating $\Pr(Y_{i,m} = 1 \mid q_i)$ and a *selector* that returns $\hat{m}_i = \arg\max_m \hat{p}_{i,m}$.

Cost-aware utility. For $\lambda \in [0, 1]$, the cost-utility scalarization of choosing model m on query q_i is $U(\lambda; q_i, m) = (1 - \lambda)\hat{p}_{i,m} - \lambda \tilde{c}_{i,m}$, generalizing the metric used in RouterBench and R2-Bench (Hu et al., 2024; Xue et al., 2026). $\lambda = 0$ recovers pure accuracy maximization; $\lambda = 1$ is pure cost minimization. Throughout the paper, evaluation cells are indexed by the triple $(\mathcal{Q}, \mathcal{P}, \lambda)$.

Routing accuracy and the oracle. The accuracy of router r on a test set $\mathcal{Q}_{\text{test}}$ of size N is $A_r = \frac{1}{N}\sum_{i=1}^{N} Y_{i,\, r(q_i)}$. If a router could see the correctness labels of every model before choosing, it would pick a correct one, giving $A_{\text{oracle}} = \frac{1}{N}\sum_{i=1}^{N} \max_{m \in \mathcal{P}} Y_{i,m}$. A_{oracle} depends only on $(\mathcal{Q}, \mathcal{P})$ and not on which router we use. It is computed from labels that a real router never sees at decision time, so it sits as an upper bound rather than something deployed systems can match.

Concordance. It assesses whether the router generally ranks correct models above incorrect ones. For each test query q_i, let $\mathcal{S}_i = \{(m, n) \in \mathcal{P}^2 : Y_{i,m} = 1,\ Y_{i,n} = 0\}$ contain all pairs of models where m answers that query correctly and n does not. We compute

$$\text{Conc}(r) = \frac{\displaystyle\sum_{i=1}^{N} \sum_{(m,n) \in \mathcal{S}_i} \left[\mathbf{1}\{\hat{p}_{i,m} > \hat{p}_{i,n}\} + \tfrac{1}{2}\mathbf{1}\{\hat{p}_{i,m} = \hat{p}_{i,n}\}\right]}{\displaystyle\sum_{i=1}^{N} |\mathcal{S}_i|}. \tag{1}$$

Thus, concordance is the fraction of correct–incorrect model pairs ranked correctly.

4 The Routing Plateau Across Routers and Benchmarks

In this section, we characterize the routing plateau through a large-scale empirical study covering 21 routing methods across five representative benchmarks.

4.1 Routers, Benchmarks, and Evaluation Methods

Routing methods and encoders. We evaluate 21 routers in total, including 18 methods that span the major design families proposed from 2023 to 2026: classifier-style routers (Ong et al., 2025; Sakota et al., 2024; Liu et al., 2024; Not Diamond, 2025), retrieval routers (Lu et al., 2023; Somerstep et al., 2025), ranking and pairwise routers (Frick et al., 2025; Zhao et al., 2024), latent-factor and IRT routers (Song et al., 2025; Zhuang et al., 2024), contrastive routers (Chen et al., 2024; Lai & Ye, 2026), expert-orchestration and cascade routers (Zhang et al., 2025b; 2023; Ding et al., 2024; Woisetschläger et al., 2025), and bandit/cost-aware routers (Li et al., 2010). We additionally include three classical baselines: `mlp_cn` and `knn` (Hu et al., 2024), and `kmeans` (Jitkrittum et al., 2025).

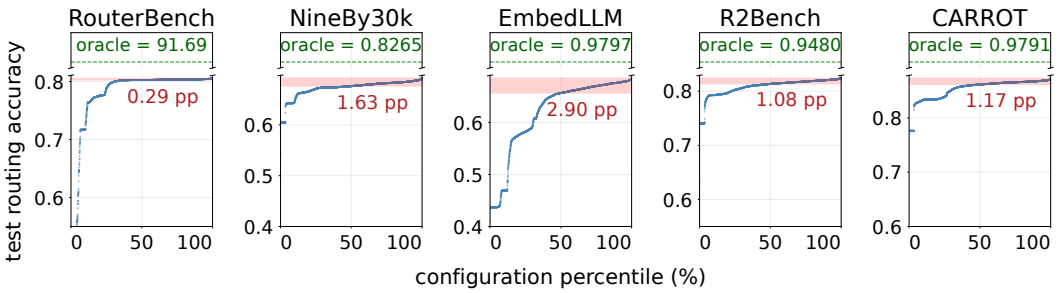


Figure 2: Test routing accuracy at λ=0 across all sorted configurations. Red bands mark the accuracy ranges of the highest-scoring half of configurations. Green dashed lines mark oracle accuracy.

We use six query encoders across three tiers: sentence-transformer encoders MINILM-L6 (Wang et al., 2020) and MPNET-BASE (Song et al., 2020); BERT-family encoders BGE-BASE (Xiao et al., 2023), MODERNBERT-BASE and MODERNBERT-LARGE (Warner et al., 2024); and a decoder-only LLM as encoder QWEN2.5-0.5B (Qwen Team, 2024). The six encoders span 384 to 1024 output dimensions and three orders of magnitude of parameter count (App. A.1).

Benchmarks. We evaluate routers on five benchmarks. Four of them are public LLM-routing benchmarks: RouterBench (Hu et al., 2024), R2-Bench (Xue et al., 2026), EmbedLLM (Zhuang et al., 2024), and CARROT/SPROUT (Somerstep et al., 2025). They are large enough to train a frozen-embedding prediction head, but are too small for reliable data scaling analysis or end-to-end encoder fine-tuning. To close this gap, we construct a new benchmark with two paired subsets. Nine-by-30k provides 23,027 training queries labeled for nine models for the main comparison. Nine-by-300k expands this to 302,371 training queries for data scaling and encoder fine-tuning. We defer the construction details of our benchmark to App. A.2.

Evaluation methods. Each router configuration is a (encoder, method) pair, where the encoder produces query embeddings and the method defines the routing rule on embeddings. We use the same train/validation/test split for all router configuration. For each (benchmark, encoder, method) setting, we sweep architectures and hyperparameters, selecting the variant with the highest validation accuracy at λ=0.

4.2 THE ROUTING PLATEAU PHENOMENON

Extensive configuration sweep: the plateau is broad. Our sweep covers a broad grid of router configurations. Fig. 2 sorts all configurations by their test accuracy within each benchmark. The resulting distribution shows a broad plateau: on all five benchmarks, at least half of the configurations fall within a short range of the highest observed test accuracy. Thus, the plateau is reached by many configurations, spanning different methods, and encoders landing in the same band. However, even the best configurations remain 10.6–29.4 pp below the oracle router: they still select models whose responses score lower than the best available response.

Top-tier methods are nearly indistinguishable. We next ask whether any routing method consistently separates itself from the rest. Tab. 1 reports test routing accuracy at λ=0 on all benchmarks, with results averaged across encoders. It shows that the leading methods form a tight cluster: the top-5 methods differ by only 0.22 pp on average. Paired bootstrap tests among the top-five methods found no significant accuracy differences in the comparisons (App. B.1). Meanwhile, half of all methods lie within 0.08–1.13 pp of the best, under 5% of the oracle gap between the best router and the oracle. These results suggest that the leading methods have reached a narrow accuracy plateau, with none substantially outperforming the others or approaching oracle accuracy.

The plateau exists beyond pure accuracy. We next examine whether the plateau remains under cost-aware routing objectives. Tab. 2 reports the realized cost-utility $U(\lambda)$ for the 16 calibrated methods on Nine-by-30k at λ>0. The same near-tie pattern persists: the mean top-five cost-utility spread is only 0.18 across the four cost tradeoffs. This suggests that the plateau persists when

Table 1: Test routing accuracy (%), averaged over encoders. Bold marks the best result per column.

	RouterBench	Nine-by-30k	EmbedLLM	R2-Bench	CARROT/SPROUT	Avg
Zooter (Lu et al., 2023)	80.38	**68.38**	**67.51**	**82.12**	86.64	**77.01**
kNN (Hu et al., 2024)	**80.39**	68.36	67.16	82.08	86.78	76.95
Elo (Zhao et al., 2024)	80.36	68.34	67.02	82.08	**86.85**	76.93
Avengers-Pro (Zhang et al., 2025b)	80.34	68.31	66.66	81.89	86.75	76.79
EquiRouter (Lai & Ye, 2026)	80.36	68.24	67.08	81.80	86.46	76.79
KMeans (Jitkrittum et al., 2025)	80.35	**68.38**	66.67	81.87	86.64	76.78
MESS+ (Woisetschläger et al., 2025)	80.35	68.20	66.99	81.69	86.61	76.77
CARROT (Somerstep et al., 2025)	80.30	68.22	66.98	81.54	86.37	76.68
CP-Router (Su et al., 2025)	80.29	68.22	66.09	81.93	86.65	76.64
P2L (Frick et al., 2025)	80.31	67.95	66.38	81.56	86.32	76.50
MLP (Hu et al., 2024)	80.29	67.98	66.51	81.38	86.12	76.46
EmbedLLM (Zhuang et al., 2024)	80.32	67.81	66.05	81.40	86.34	76.38
ModelSpider (Zhang et al., 2023)	80.31	67.61	65.58	81.58	86.40	76.29
RoRF (Not Diamond, 2025)	77.57	67.86	64.43	81.60	86.26	75.54
TagRouter (Sakota et al., 2024)	80.34	67.97	61.08	81.41	84.82	75.12
LinUCB (Li et al., 2010)	73.41	67.76	65.76	80.72	85.87	74.70
RouterDC (Chen et al., 2024)	80.17	67.54	58.16	80.22	83.13	73.84
MIRT (Song et al., 2025)	76.36	66.64	59.20	80.94	84.88	73.60
IRT-Router (Song et al., 2025)	77.49	66.49	52.06	81.04	84.78	72.37
HybridLLM (Ding et al., 2024)	71.67	60.46	46.87	73.99	77.62	66.12
OptLLM (Liu et al., 2024)	76.68	66.61	—	80.43	85.12	—
Top-5 router spread (max−min)	0.04	0.07	0.52	0.23	0.21	0.22

Table 2: Cost-utility on Nine-by-30k at $\lambda>0$: $U(\lambda)=(1-\lambda)\hat{s}-\lambda\tilde{c}$, mean over six encoders. Sorted by $\lambda=0.5$. All numbers are in percent (%).

	$\lambda=0.2$	$\lambda=0.5$	$\lambda=0.7$	$\lambda=0.9$	Avg
Zooter (Lu et al., 2023)	**51.97**	**31.25**	**17.82**	4.76	**26.45**
EquiRouter (Lai & Ye, 2026)	51.80	31.13	17.75	4.75	26.36
MESS+ (Woisetschläger et al., 2025)	51.59	31.06	17.72	4.72	26.27
P2L (Frick et al., 2025)	51.18	31.06	17.73	4.67	26.16
KMeans (Jitkrittum et al., 2025)	51.91	31.05	17.76	4.83	26.39
CARROT (Somerstep et al., 2025)	51.50	31.02	17.42	4.59	26.13
Elo (Zhao et al., 2024)	51.64	31.01	17.42	4.59	26.16
Avengers-Pro (Zhang et al., 2025b)	51.56	30.99	17.39	4.57	26.13
kNN (Hu et al., 2024)	51.89	30.94	17.70	**4.84**	26.34
MLP (Hu et al., 2024)	51.39	30.93	17.70	4.74	26.19
RoRF (Not Diamond, 2025)	51.08	30.88	17.14	4.53	25.91
EmbedLLM (Zhuang et al., 2024)	50.93	30.78	17.56	4.66	25.98
OptLLM (Liu et al., 2024)	50.10	29.89	17.10	4.68	25.44
CP-Router (Su et al., 2025)	50.41	28.67	16.14	4.46	24.92
MIRT (Song et al., 2025)	46.84	28.25	16.28	4.48	23.96
IRT-Router (Song et al., 2025)	43.67	26.82	15.62	4.43	22.64
Top-5 router spread (max−min)	0.33	0.20	0.10	0.10	0.18

balancing accuracy against inference cost, with no leading method offering a substantial cost-utility advantage over the others. The pattern generalizes beyond Nine-by-30k (App. B.2)

4.3 The Plateau Persists Across Evaluation Settings

The plateau is stable across encoders. We test whether the plateau is driven by a particular query representation. Fig. 3 reports the standard deviation of test accuracy across encoders for each routing method. Averaged over the methods shown, this standard deviation ranges from 0.07 to 0.65 pp across benchmarks. This suggests that the plateau is not an artifact of a specific encoder choice, and we include per-encoder breakdowns in App. B.3.

The plateau persists across benchmark settings. We further test whether the plateau depends on the query distribution. First, restricting the benchmarks to easier queries, which more models answer correctly, reduces the oracle gap but does not eliminate the plateau: leading routers still achieve

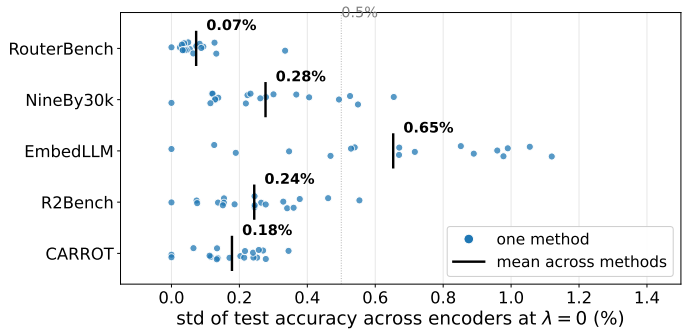


Figure 3: Standard deviation of test accuracy across encoders at λ=0, computed per method and benchmark. Each blue dot is a routing method and black ticks show the mean across methods.

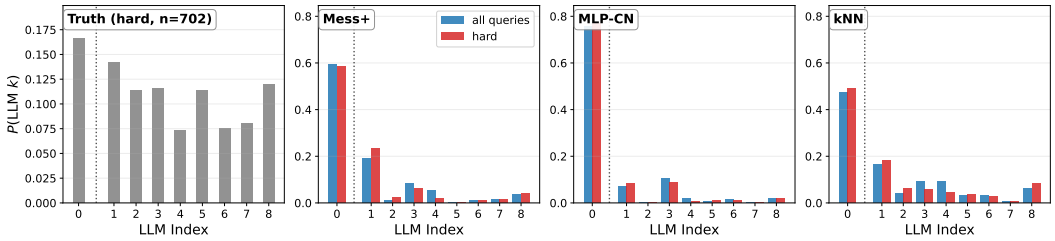


Figure 4: Model selection on hard (red) and all (blue) Nine-by-30k queries. Routers favor globally strong models, while correct-model mass on hard queries (gray) is more evenly distributed.

similar accuracy far below the oracle. Second, grouping Nine-by-30k queries by task domain shows that leading routers remain close in accuracy and substantially below the oracle within domains such as math and question answering. Collectively, these observations suggest that the plateau is not specific to a particular benchmark or query mixture, but is a pattern shared by the routers we test. App. B.4 provides the experimental setups and detailed results.

Multi-sample correctness labels. We also test whether the plateau and oracle gap could arise from noise in single-sample correctness labels. On Nine-by-30k, ten-sample majority labels yield an oracle accuracy of 0.817, compared with 0.827 under single-sample labels, retaining about 93% of the gap to the original best-router accuracy. These results suggest that neither the plateau nor the large oracle gap is primarily an artifact of noise in single-sample correctness labels. More detail in App. B.5.

5 Understanding the Routing Plateau

We next seek to understand why the routing plateau occurs. Our hypothesis is that it arises from a **correctness-prediction bottleneck**: before selecting a model, a router must infer how likely each candidate model can answer the query correctly. However, current routers tend to learn coarse, global model-capability patterns rather than fine-grained, instance-level differences in correctness. We support this hypothesis through the analysis in this section.

5.1 Coarse-Grained Correctness Prediction Limits Routing Accuracy

To examine whether routers distinguish model correctness at the query level, we group queries by the number of models that answer them correctly. Let $n_{\text{correct}}(i)$ denote this number for query q_i. We call a query *hard* if $n_{\text{correct}}(i) \leq t$, where $t = \max(3, \lfloor 0.3K \rfloor)$, and *easy* otherwise. We could examine how routers choose models when correct answers are from only a small subset of the pool.

Fig. 4 compares model-selection frequencies on hard queries and on all queries for two learned routers and kNN, alongside the distribution of correct models on hard queries. First, model-selection frequencies change little between hard queries and the full query set. Second, routers select the globally top-ranked model far more often than other models, whereas the ground-truth correctness

Table 3: Oracle–router gaps averaged over 12 routers. f_{hard} is the hard-query fraction; Ratio is hard/easy gap; Gap share is hard queries' contribution to the total gap.

Benchmark	f_{hard}	Hard gap	Easy gap	Ratio	Gap share
RouterBench	0.132	0.515	0.046	11.3×	64.1%
CARROT/SPROUT	0.085	0.655	0.054	12.1×	53.3%
R2-Bench	0.160	0.523	0.049	10.6×	68.0%
Nine-by-30k	0.140	0.664	0.073	9.0×	64.8%
EmbedLLM	0.432	0.570	0.108	5.3×	80.7%

Table 4: Disagree is the model-selection disagreement range, and Acc. gap is the median accuracy difference. Within disagreements, $|\Delta Y|$ measures exchanged wins and $|\Delta Y_{\text{signed}}|$ their net imbalance. Disagreement and unsigned exchange are in %, accuracy gaps and signed exchange in pp.

Benchmark	Disagree range	Acc. gap	$\lvert\Delta Y\rvert$	$\lvert\Delta Y_{\text{signed}}\rvert$	$\lvert\Delta Y\rvert/\lvert\Delta Y_{\text{signed}}\rvert$
RouterBench	0.8–19.4	0.27	10.8	2.4	5×
CARROT/SPROUT	16.0–70.0	0.27	9.6	0.5	17×
R2-Bench	14.9–44.2	0.27	16.2	1.0	17×
Nine-by-30k	3.2–52.7	0.46	14.5	1.1	13×
EmbedLLM	16.0–53.0	0.55	22.5	1.8	13×

distribution is more evenly spread across the pool. This mismatch suggests that routers favor globally strong models rather than learning fine-grained, query-specific differences in model correctness. We repeat this analysis across multiple routers and observe the same pattern in App. C.1.

We next quantify how much hard queries contribute to routing failures. Tab. 3 shows that, on average across routers, the rate of selecting an incorrect model is 5.3–12.1 times as high on hard queries as on easy ones. Although hard queries are a minority of each benchmark, they account for 53.3%–80.7% of the overall oracle–router gap. Together, these results suggest that routers capture global capability patterns but struggle to identify which models will succeed on individual hard queries.

5.2 Prediction Variation Across Routers Yields Error Exchange, Not Accuracy Gaps

Fig. 4 shows that routers can learn slightly different correctness-prediction functions and therefore select different models on some queries. These differences can arise from their distinct inductive biases, training objectives, and optimization procedures. However, Sec. 4 shows that these differences do not yield clear aggregate accuracy gains: top routers still converge to the same plateau. Our hypothesis is that router disagreement mostly produces *error exchange* rather than consistent improvement. One router may solve queries another misses while failing on a comparable number that the other solves, so their wins and losses largely cancel out in overall accuracy.

To test this hypothesis, we consider each pair of routers (r_a, r_b) and measure their aggregate accuracy gap, $|\mathrm{Acc}(r_a) - \mathrm{Acc}(r_b)|$, and their model-selection disagreement rate. We further decompose disagreements into exchanged wins: queries solved by r_a but missed by r_b, and vice versa. App. C.2 gives the computation details. Tab. 4 shows that router differences are mostly balanced rather than directional. Top-tier routers can disagree on up to 70% of queries, yet their median accuracy gaps remain small (0.3–0.6 pp) because their wins are nearly symmetric. Conditioned on disagreement, the median unsigned outcome change is $|\Delta Y|$=9.6%–22.5%, while the signed change is 5–17× smaller. Thus, prediction variation mainly induces error exchange rather than consistent accuracy separation, keeping routers clustered on the same plateau.

5.3 What Does Learning Add Beyond Local Similarity?

We next ask whether learned routers predict model correctness better than kNN, which estimates each model's correctness from neighboring training queries in the embedding space. We compare this local estimator with learned routers in routing accuracy and concordance, using the evaluated frozen query representations. Concordance (Sec. 3) evaluates the ranking of correct models above incorrect ones, complementing routing accuracy. App. C.3 details the comparison setup.

Table 5: kNN vs. trained routers. (a) Within-query ranking quality (concordance): trained routers have similar performance to kNN. (b) Hard-subset accuracy, Δ = median trained − kNN: kNN matches the median trained routers.

(a) Test concordance: kNN is comparable with the trained routers.

Benchmark	kNN	11 other routers
RouterBench	0.768	0.762–0.772
CARROT/SPROUT	0.805	0.794–0.804
R2-Bench	0.887	0.874–0.887
Nine-by-30k	0.665	0.637–0.667
EmbedLLM	0.680	0.676–0.686

(b) kNN matches the trained router on Hard-subset.

Benchmark	kNN	Median Acc. Δ
RouterBench	0.478	$+0.008$
CARROT/SPROUT	0.331	$+0.014$
R2-Bench	0.484	-0.005
Nine-by-30k	0.340	-0.007
EmbedLLM	0.444	-0.015

kNN is among the leading methods on every benchmark (Tab. 1). Learned routers also show no clear advantage over kNN on hard queries, where fine-grained prediction of individual models' correctness is especially important (Tab. 5b). Their concordance is also similar to that of kNN's local similarity estimates, with gains over kNN limited to 0.006 across the five benchmarks (Tab. 5a).

These results suggest that kNN provides a strong local approximation of model correctness from the evaluated query embeddings and training labels. The learned routers offer little improvement over this local estimate, especially on hard queries. This supports our correctness-prediction bottleneck hypothesis: methods share a difficulty in predicting instance-level correctness from frozen query representations. Sec. 6 tests whether strengthening the training regime through more data, larger encoders, and end-to-end fine-tuning substantially changes this picture.

6 Does a Stronger Training Regime Alleviate the Plateau?

Finally, we ask whether the plateau reflects limitations of the training regime: insufficient data, limited encoder capacity, or query representations that are not adapted to routing. If these constraints largely explain the plateau, a stronger training regime should substantially narrow the oracle gap. Holding the router designs fixed, we study three changes: *(1) scaling the training data, (2) using larger query encoders, and (3) fine-tuning the router end-to-end.*

6.1 Experimental Setup

Datasets and encoders. To study data scaling, we use our paired benchmarks Nine-by-30k and Nine-by-300k, expanding the training set from 23K to 300K queries with correctness labels for the same nine candidate models. Both benchmarks share identical validation and test splits. Construction details and exact split sizes are provided in App. A.2. To study encoder scaling, we compare two encoders from the same architecture family: ModernBERT-base (`mbb`, ∼110M parameters) and ModernBERT-large (`mbl`, ∼340M parameters).

Fine-tunable routing methods. We evaluate eight routers from the main grid whose implementations support end-to-end encoder fine-tuning: `mlp_cn`, `embedllm`, `routerdc`, `p2l`, `cp_router`, `mess_plus`, `zooter_lite`, and `equi`. We retain `kNN` as a frozen reference to measure the effect of data and encoder scaling without fine-tuning.

Configurations. We denote configurations by (data scale, training mode, encoder), with 30k and 300k referring to Nine-by-30k and Nine-by-300k, respectively. The training mode either keeps the encoder frozen or fine-tunes it end-to-end (FT). Our *baseline* is (30k, frozen, mbb), and *Scaled-FT* is (300k, FT, mbl). Additional training settings are provided in App. D.1.

6.2 Scaling and Fine-Tuning Yield Modest Gains

Scaled-FT improves all eight evaluated routers, with gains of 0.31–2.13 pp and a mean gain of 1.24 pp (Fig. 5). Concordance also improves: on the shared test split, the eight fine-tuned routers

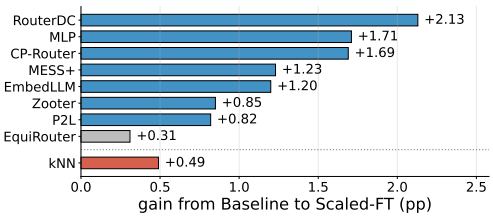


Figure 5: Per-method gain in test routing accuracy on Nine-by-30k→Nine-by-300k, from Baseline (30k, frozen, mbb) to Scaled-FT (300k, FT, mbl). kNN has no fine-tuning axis and serves as the data-and-encoder-scaling-only comparison target.

Table 6: Ablation study of data scaling, encoder scaling, and router fine-tuning. Mean accuracy is reported across the eight trained methods. *enc*: ModernBERT-large (mbl) vs. ModernBERT-base (mbb); *data*: 300k vs. 30k; *FT*: end-to-end fine-tuning vs. frozen encoder.

enc	data	FT	Cell	Mean acc	Δ pp
			30k, frzn, mbb	0.6814	—
✓			30k, frzn, mbl	0.6827	+0.13
	✓		300k, frzn, mbb	0.6861	+0.47
		✓	30k, FT, mbb	0.6856	+0.42
✓		✓	30k, FT, mbl	0.6851	+0.37
	✓	✓	300k, FT, mbb	0.6933	+1.19
✓	✓	✓	300k, FT, mbl (fix)	**0.6938**	**+1.24**

achieve a mean of 0.680, compared with at most 0.667 among the frozen routers evaluated in Sec. 5.3. However, seven of the eight routers still fall within a 0.85 pp accuracy band, and their mean accuracy remains 13.27 pp below the oracle. The advantage over kNN also remains modest: under the same data and encoder scaling, frozen kNN improves by 0.49 pp and matches the fine-tuned routers' mean accuracy of 69.38%. Thus, strengthening the training regime improves accuracy and correctness ranking, but leaves routers closely clustered and far below the oracle.

6.3 Ablation Study

Tab. 6 separates the contributions of data scaling, encoder scaling, and fine-tuning, averaging accuracy across the eight routers. Each change alone improves accuracy by less than 0.5 pp. Combining more data with fine-tuning yields a larger gain of 1.19 pp with the base encoder, while increasing encoder size adds only another 0.05 pp. Thus, most of the improvement comes from jointly expanding the training data and adapting the query representation, with little benefit from encoder scaling.

On CARROT/SPROUT and EmbedLLM, which lack data-scaled variants, we evaluate encoder scaling and end-to-end fine-tuning with the original training sets. Relative to each benchmark's frozen-baseline, the fine-tuned large encoder improves mean accuracy by 0.50 and 1.64 pp, respectively. The full encoder-by-training-mode grids are reported in App. D.2.

6.4 Discussion: The Remaining Oracle Gap

Across the routing methods and query representations we evaluate, routers reach similar accuracy while leaving a substantial gap to the oracle. More training data, larger encoders, and end-to-end fine-tuning yield modest gains but do not substantially change this pattern. These interventions improve how routers learn from queries, while leaving their inputs at routing time unchanged: they must still predict correctness before observing any candidate model's response. Attempting a task could reveal model-specific difficulties that are not apparent from the query's semantics alone. This suggests that further progress may require enriching the inputs to routing with evidence from model execution, such as partial output trajectories or confidence estimates derived from an initial attempt.

7 Conclusion

Our extensive study of 21 routing methods across five representative benchmarks reveals the *routing plateau*: many routers, including kNN-style methods, cluster around a similar accuracy ceiling that remains far below the oracle router. Our analysis supports a correctness-prediction bottleneck hypothesis: current routers largely capture coarse, global patterns of model capability, but struggle to identify which models will succeed on individual hard queries. Using our Nine-by-300k benchmark, we find that more training data, larger encoders, and end-to-end fine-tuning improve accuracy by 1.24 pp on average across eight routers, while leaving the plateau largely intact. These findings suggest that approaching the oracle may require richer routing inputs beyond the query alone.

AI USE STATEMENT

We used LLM assistants to identify related work, refine the paper's research questions and framing, and develop experimental analyses. LLM coding agents, including Codex and Claude Code, helped implement experiments and analyze results. We also manually check experiment scripts to ensure desired behaviors are executed. We also used LLM assistants to polish the writing, including correcting grammar and making complex sentences more concise and readable. To construct the Nine-by-30k and Nine-by-300k benchmarks, we collected responses from nine candidate LLMs to queries drawn from the source datasets described in App. A.2. We used `gpt-4o-mini` as a fallback correctness judge for a small subset of open-ended QA tasks, as detailed in App. A.2. The authors take responsibility for the final manuscript, experimental procedures, and reported results.

REFERENCES

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms. In *NAACL-HLT*, 2019.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the Cross-lingual Transferability of Monolingual Representations. In *ACL*, 2020.

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, 2021. URL https://arxiv.org/abs/2108.07732.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset, 2018. URL https://arxiv.org/abs/1611.09268.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP*, 2013.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating Large Language Models Trained on Code, 2021a. URL https://arxiv.org/abs/2107.03374.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James T. Kwok, and Yu Zhang. RouterDC: Query-Based Router by Dual Contrastive Learning for Assembling Large Language Models, 2024. URL https://arxiv.org/abs/2409.19886.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. FinQA: A Dataset of Numerical Reasoning over Financial Data. In *EMNLP*, 2021b.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. QuAC: Question Answering in Context. In *EMNLP*, 2018.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *NAACL-HLT*, 2019.

Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. TyDi QA: A Benchmark for Information-Seeking Question Answering in Typologically Diverse Languages. *Transactions of the Association for Computational Linguistics*, 8:454–470, 2020.

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, 2018. URL https://arxiv.org/abs/1803.05457.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing, 2024. URL https://arxiv.org/abs/2404.14618.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. SearchQA: A New Q&A Dataset Augmented with Context from a Search Engine, 2017. URL https://arxiv.org/abs/1704.05179.

Tao Feng, Yanzhen Shen, and Jiaxuan You. GraphRouter: A Graph-based Router for LLM Selections, 2025. URL https://arxiv.org/abs/2410.03834.

Evan Frick, Connor Chen, Joseph Tennyson, Tianle Li, Wei-Lin Chiang, Anastasios N. Angelopoulos, and Ion Stoica. Prompt-to-Leaderboard, 2025. URL https://arxiv.org/abs/2502.14855.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning AI With Shared Human Values. In *International Conference on Learning Representations (ICLR)*, 2021a.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations (ICLR)*, 2021b.

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 Datasets and Benchmarks Track*, 2021c.

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, 2024. URL https://arxiv.org/abs/2403.12031.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code, 2024. URL https://arxiv.org/abs/2403.07974.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What Disease does this Patient Have? A Large-scale Open Domain Question Answering Dataset from Medical Exams. *Applied Sciences*, 11(14), 2021.

Ruihan Jin, Pengpeng Shao, Zhengqi Wen, Jinyang Wu, Mingkuan Feng, Shuai Zhang, and Jianhua Tao. RadialRouter: Structured Representation for Efficient and Robust Large Language Models Routing, 2025. URL https://arxiv.org/abs/2506.03880.

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, 2025. URL https://arxiv.org/abs/2502.08773.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. QASC: A Dataset for Question Answering via Sentence Composition. In *AAAI*, 2020.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA Reading Comprehension Challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

Guannan Lai and Han-Jia Ye. When Routing Collapses: On the Degenerate Convergence of LLM Routers, 2026. URL https://arxiv.org/abs/2602.03478.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. MLQA: Evaluating Cross-lingual Extractive Question Answering. In *ACL*, 2020.

Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Biqing Qi, Peng Ye, Lei Bai, Zhen Wang, and Shuyue Hu. LLMRouterBench: A Massive Benchmark and Unified Framework for LLM Routing, 2026. URL https://arxiv.org/abs/2601.07206.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pp. 661–670. ACM, April 2010. doi: 10.1145/1772690.1772758. URL http://dx.doi.org/10.1145/1772690.1772758.

Yang Li. Rethinking Predictive Modeling for LLM Routing: When Simple kNN Beats Complex Learned Routers, 2025. URL https://arxiv.org/abs/2505.12601.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring How Models Mimic Human Falsehoods. In *ACL*, 2022.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems. In *ACL*, 2017.

Yueyue Liu, Hongyu Zhang, Yuantian Miao, Van-Hoang Le, and Zhiqiang Li. OptLLM: Optimal Assignment of Queries to Large Language Models, 2024. URL https://arxiv.org/abs/2405.15130.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the Expert: Efficient Reward-guided Ensemble of Large Language Models, 2023. URL https://arxiv.org/abs/2311.08692.

Yifan Lu, Rixin Liu, Jiayi Yuan, Xingqi Cui, Shenrun Zhang, Hongyi Liu, and Jiarong Xing. RouterArena: An Open Platform for Comprehensive Comparison of LLM Routers, 2025. URL https://arxiv.org/abs/2510.00202.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A Diverse Corpus for Evaluating and Developing English Math Word Problem Solvers. In *ACL*, 2020.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In *EMNLP*, 2018.

Not Diamond. RoRF: Routing on Random Forests. https://github.com/Not-Diamond/RoRF, 2025. Accessed: 2026-04-25.

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, 2025. URL https://arxiv.org/abs/2406.18665.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. MedMCQA: A Large-scale Multi-Subject Multi-Choice Dataset for Medical domain Question Answering. In *Conference on Health, Inference, and Learning (CHIL)*, 2022.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP Models really able to Solve Simple Math Word Problems? In *NAACL-HLT*, 2021.

Qwen Team. Qwen2.5 Technical Report, 2024. URL https://arxiv.org/abs/2412.15115.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*, 2016.

Siva Reddy, Danqi Chen, and Christopher D Manning. CoQA: A Conversational Question Answering Challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Pedro Rodriguez, Shi Feng, Mohit Iyyer, He He, and Jordan Boyd-Graber. Quizbowl: The Case for Incremental Question Answering, 2019. URL https://arxiv.org/abs/1904.04792.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. In *AAAI*, 2020.

Marija Sakota, Maxime Peyrard, and Robert West. Fly-Swat or Cannon? Cost-Effective Language Model Choice via Meta-Modeling. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, WSDM '24, pp. 606–615. ACM, March 2024. doi: 10.1145/3616855.3635825. URL http://dx.doi.org/10.1145/3616855.3635825.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense Reasoning about Social Interactions. In *EMNLP-IJCNLP*, 2019.

Seamus Somerstep, Felipe Maia Polo, Allysson Flavio Melo de Oliveira, Prattyush Mangal, Mírian Silva, Onkar Bhardwaj, Mikhail Yurochkin, and Subha Maity. CARROT: A Cost Aware Rate Optimal Router, 2025. URL https://arxiv.org/abs/2502.03261.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: Masked and Permuted Pre-training for Language Understanding, 2020. URL https://arxiv.org/abs/2004.09297.

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, 2025. URL https://arxiv.org/abs/2506.01048.

Jiayuan Su, Fulin Lin, Zhaopeng Feng, Han Zheng, Teng Wang, Zhenyu Xiao, Xinlong Zhao, Zuozhu Liu, Lu Cheng, and Hongwei Wang. CP-Router: An Uncertainty-Aware Router Between LLM and LRM, 2025. URL https://arxiv.org/abs/2505.19970.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. In *NAACL-HLT*, 2019.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. NewsQA: A Machine Comprehension Dataset. In *2nd Workshop on Representation Learning for NLP (RepL4NLP)*, 2017.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16(138), 2015.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In *NeurIPS*, 2019.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers, 2020. URL https://arxiv.org/abs/2002.10957.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference, 2024. URL https://arxiv.org/abs/2412.13663.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing Multiple Choice Science Questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, 2017.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing Datasets for Multi-hop Reading Comprehension Across Documents. *Transactions of the Association for Computational Linguistics*, 6:287–302, 2018.

Herbert Woisetschläger, Ryan Zhang, Shiqiang Wang, and Hans-Arno Jacobsen. MESS+: Dynamically Learned Inference-Time LLM Routing in Model Zoos with Service Level Guarantees, 2025. URL https://arxiv.org/abs/2505.19947.

Fangzhou Wu and Sandeep Silwal. Efficient Training-Free Online Routing for High-Volume Multi-LLM Serving. In *NeurIPS*, 2025.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-Pack: Packed Resources For General Chinese Embeddings, 2023. URL https://arxiv.org/abs/2309.07597.

Jiaqi Xue, Qian Lou, Jiarong Xing, and Heng Huang. R2-Router: A New Paradigm for LLM Routing with Reasoning, 2026. URL https://arxiv.org/abs/2602.02823.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *EMNLP*, 2018.

Ruibin Yuan, Hanfeng Lin, Yi Wang, Zeyue Tian, Shangda Wu, Tianhao Shen, Ge Zhang, Yuhang Wu, Cong Liu, Ziya Zhou, et al. ChatMusician: Understanding and Generating Music Intrinsically with LLM, 2024. URL https://arxiv.org/abs/2402.16153.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? In *ACL*, 2019.

Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-R1: Teaching LLMs Multi-Round Routing and Aggregation via Reinforcement Learning. In *NeurIPS*, 2025a.

Yi-Kai Zhang, Ting-Ji Huang, Yao-Xiang Ding, De-Chuan Zhan, and Han-Jia Ye. Model Spider: Learning to Rank Pre-Trained Models Efficiently, 2023. URL https://arxiv.org/abs/2306.03900.

Yiqun Zhang, Hao Li, Jianhao Chen, Hangfan Zhang, Peng Ye, Lei Bai, and Shuyue Hu. Beyond GPT-5: Making LLMs Cheaper and Better via Performance-Efficiency Optimized Routing. In *Proceedings of the 2025 The Seventh International Conference on Distributed Artificial Intelligence*, DAI '25, pp. 122–129. ACM, November 2025b. doi: 10.1145/3772429.3772445. URL http://dx.doi.org/10.1145/3772429.3772445.

Zesen Zhao, Shuowei Jin, and Z. Morley Mao. Eagle: Efficient Training-Free Router for Multi-LLM Inference, 2024. URL https://arxiv.org/abs/2409.15518.

Lucia Zheng, Neel Guha, Brandon R Anderson, Peter Henderson, and Daniel E Ho. When does Pretraining Help? Assessing Self-Supervised Learning for Law and the CaseHOLD Dataset. In *ICAIL*, 2021.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. AGIEval: A Human-Centric Benchmark for Evaluating Foundation Models, 2023. URL https://arxiv.org/abs/2304.06364.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. EmbedLLM: Learning Compact Representations of Large Language Models, 2024. URL https://arxiv.org/abs/2410.02223.

Table 7: Benchmark suite. Train/Val/Test are available unique-query counts; K is the pool size; Cost marks cost availability. For EmbedLLM, cost is approximated by model parameter count. The main grid evaluates 3,000 EmbedLLM and 6,637 CARROT/SPROUT test queries; the table lists their full available test splits.

Benchmark	Train	Val	Test	K	Cost
Nine-by-30k (ours)	23,027	3,000	5,000	9	✓
Nine-by-300k (ours)	302,371	3,000	5,000	9	✓
RouterBench	21,898	3,649	10,950	11	✓
R2-Bench	18,580	3,097	9,291	10	✓
EmbedLLM	119,964	14,897	15,210	112	Proxy
CARROT/SPROUT	152,193	35,528	36,214	13	✓

A Experimental Setup

A.1 Router Training and Evaluation

A.1.1 Benchmarks and preprocessing

Tab. 7 summarizes the available benchmark splits. The main comparison uses shared evaluation queries across routers and encoders: 10,950 on RouterBench, 5,000 on Nine-by-30k, 3,000 on EmbedLLM, 9,291 on R2-Bench, and 6,637 on CARROT/SPROUT. Query embeddings are cached once per benchmark and encoder and reused by all frozen-encoder methods.

The benchmarks provide task scores $s_{im} \in [0, 1]$ for query i and candidate model m; some tasks give partial credit. Routing accuracy in the main grid is the mean score of the selected model, and the oracle averages the maximum score within each query. The behavioral analyses in Sec. 5.1 instead use binary labels $y_{im} = \mathbf{1}[s_{im} \geq 0.5]$, as detailed in App. C.1. All methods in a comparison use the same score matrix and model ordering. Cost annotations are used only for the cost-utility analysis at $\lambda > 0$.

A.1.2 Training, selection, and averaging

Hyperparameter selection uses two stages. For each benchmark, method, and encoder, we first evaluate up to 15 architectures with seed 42 and retain the top five by validation routing accuracy at $\lambda = 0$. These architectures are then evaluated with four additional seeds, $\{1337, 2026, 7, 99\}$. The result aggregation first prefers architectures with the most completed seeds and then selects the highest mean validation accuracy. Trained methods in the main grid use the checkpoint with the best validation routing accuracy.

Training objectives follow the router implementation: classifier-style heads use BCE-based losses, distillation methods combine KL and BCE terms, P2L uses a Bradley–Terry objective, and latent-factor methods use their corresponding objectives. The cost sweep applies $U(\lambda) = (1 - \lambda)\hat{s} - \lambda\tilde{c}$ after training, for $\lambda \in \{0, 0.2, 0.5, 0.7, 0.9\}$, retaining the architecture selected at $\lambda = 0$.

A.1.3 Hardware

The frozen-encoder experiments used NVIDIA RTX A5000 GPUs (24 GB) and NVIDIA L40S GPUs (48 GB), on nodes with eight GPUs each.

A.2 Dataset Construction and Splits

This appendix section documents the construction, splits, and scope of the two paired benchmarks introduced in Sec. 4.1.

A.2.1 Construction

Source of the prompts. Prompts are aggregated from 57 datasets covering knowledge, reasoning, code, commonsense, safety, and multilingual QA. Fifty-six are publicly released benchmarks; one

(PANDAX-patent-QA) is an internal patent question-answering split included for domain coverage. Each prompt is rewritten in a uniform format with an explicit answer-extraction instruction so that responses can be passed to the correct task-specific scorer. Per-source query counts are summarized in Tab. 8. The full source list, grouped by domain, is:

- *Multiple-choice knowledge, commonsense, and safety:* MMLU (Hendrycks et al., 2021b), MMLU-Pro (Wang et al., 2024), ARC-Challenge and ARC-Easy (Clark et al., 2018), ArcMMLU[1], BoolQ (Clark et al., 2019), MedQA (Jin et al., 2021), MedMCQA (Pal et al., 2022), GPQA (Rein et al., 2023), SciQ (Welbl et al., 2017), OpenBookQA (Mihaylov et al., 2018), CommonsenseQA (Talmor et al., 2019), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2020), TruthfulQA (Lin et al., 2022), ETHICS (Hendrycks et al., 2021a), QASC (Khot et al., 2020), BioASQ (Tsatsaronis et al., 2015), FinQA (Chen et al., 2021b), CaseHOLD (Zheng et al., 2021), MusicTheoryBench (Yuan et al., 2024).
- *Reading-comprehension and open-domain QA:* SQuAD (Rajpurkar et al., 2016), NQ-Open (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), NarrativeQA (Kočiský et al., 2018), CoQA (Reddy et al., 2019), QuAC (Choi et al., 2018), NewsQA (Trischler et al., 2017), SearchQA (Dunn et al., 2017), MS MARCO (Bajaj et al., 2018), WebQuestions (Berant et al., 2013), WikiHop (Welbl et al., 2018), MLQA (Lewis et al., 2020), TyDi QA (Clark et al., 2020), XQuAD (Artetxe et al., 2020), Social IQa (Sap et al., 2019), Qanta (Rodriguez et al., 2019), ChemistryQA[2], PANDAX-patent-QA[3].
- *Math and quantitative reasoning:* GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021c), MathQA (Amini et al., 2019), AQuA-RAT (Ling et al., 2017), SVAMP (Patel et al., 2021), AsDiv (Miao et al., 2020).
- *Code generation:* HumanEval (Chen et al., 2021a), MBPP (Austin et al., 2021), LiveCodeBench (Jain et al., 2024).
- *SuperGLUE-style tasks (Wang et al., 2019):* causal reasoning, cloze test, textual entailment, multiple-choice QA, reading comprehension, words-in-context (WiC), and Winograd schema (WSC).
- *Domain-specific:* LSAT-AR via AGIEval (Zhong et al., 2023), ChessInstruct[4].

Models. We evaluate the same nine candidate LLMs on every prompt: `deepseek-v3.2`, `gemma-3n-e4b-it`, `llama-3.3-70b-instruct`, `minimax-m2.5`, `glm-4.5-air`, `qwen3-235b-a22b-2507`, `qwen3-coder-next`, `grok-4.1-fast`, and `mimo-v2-flash`.[5] The pool is intentionally heterogeneous in scale and family: a small open-weights model (`gemma-3n-e4b-it`, $\sim$4B), several mid-to-large dense and mixture-of-experts generalists, and one code-specialist (`qwen3-coder-next`). No single model dominates across task categories, so routing decisions are non-trivial throughout the pool.

Inference protocol. All nine models are queried through the OpenRouter API. Each call records the single-sample generated answer, prompt and completion token counts, the serving provider, and a success flag. Per-call cost is computed from the recorded token counts and each provider's posted per-token rates, giving a real-valued cost annotation for every (query, model) pair.

Correctness labels. Each (query, model) pair receives a task-specific score $s_{im} \in [0, 1]$. These scores can be fractional; we retain them for the main routing-accuracy comparisons and threshold them at 0.5 for the binary analyses. Multiple-choice tasks use exact letter-choice matching against a boxed answer; extractive QA uses exact match or token-level F1; math benchmarks use numeric or symbolic equivalence; code benchmarks use unit-test execution; NarrativeQA uses METEOR; SuperGLUE subtasks use their canonical metrics. For a small set of open-ended tasks where exact match is too brittle (QANTA, WebQuestions, ChemistryQA), we fall back to LLM-as-judge with

[1] `https://huggingface.co/datasets/patrickshitou/ArcMMLU`

[2] `https://huggingface.co/datasets/avaliev/ChemistryQA`

[3] Custom internal split of a patent question-answering corpus.

[4] `https://huggingface.co/datasets/Thytu/ChessInstruct`

[5] Names follow OpenRouter's provider-qualified identifiers; decoding settings match each provider's defaults at the time of evaluation.

`gpt-4o-mini`. The rule-based path is always tried first and the judge is only invoked when it fails, so judge calls cover only a small fraction of the pool.

Table 8: Per-source query counts in Nine-by-30k and Nine-by-300k. Counts are unique prompts. Sources are grouped by domain and sorted by Nine-by-30k count within each group. Total: **31,027** prompts in Nine-by-30k, **310,371** in Nine-by-300k.

Source	Nine-by-30k	Nine-by-300k
MCQ knowledge / commonsense / safety		
MMLU	1,323	13,766
ETHICS	1,127	11,170
MMLU-Pro	1,100	11,108
HellaSwag	1,024	10,042
MedMCQA	626	6,149
CaseHOLD	536	5,314
BoolQ	341	3,270
ARC-Easy	244	2,376
MedQA	143	1,273
CommonsenseQA	123	1,221
Winogrande	121	1,267
ARC-Challenge	114	1,172
SciQ	107	1,000
QASC	98	926
TruthfulQA	82	817
ArcMMLU	69	733
FinQA	69	670
OpenBookQA	55	500
GPQA	52	546
BioASQ	38	324
MusicTheoryBench	33	367
Math and quantitative reasoning		
MathQA	282	2,983
AsDiv	246	2,147
GSM8K	148	1,319
MATH	51	500
AQuA-RAT	28	254
SVAMP	28	251
Code generation		
MBPP	52	499
LiveCodeBench	50	511
HumanEval	19	164

Source	Nine-by-30k	Nine-by-300k
Reading-comprehension and open-domain QA		
SearchQA	4,404	43,216
MLQA	4,157	42,225
Social IQa	3,322	33,375
XQuAD	1,458	14,196
SQuAD	1,075	10,532
NarrativeQA	1,005	9,897
MS MARCO	870	9,399
HotpotQA	773	7,405
CoQA	726	7,242
QuAC	642	6,612
WikiHop	525	5,128
TyDi QA	494	5,077
NewsQA	369	3,862
NQ-Open	300	2,889
WebQuestions	202	2,032
Qanta	197	1,900
PANDAX-patent-QA	181	1,810
ChemistryQA	50	494
SuperGLUE-style tasks		
ClozeTest	935	9,999
RC	463	4,847
QA	319	3,270
WiC	69	638
Entailment	25	333
WSC	14	95
CausalReasoning	8	100
Domain-specific		
LSAT-AR (AGIEval)	102	1,009
ChessInstruct	13	150

A.2.2 Splits and Scale

The split is built once and shared across every method we report, so cross-method gaps in Sec. 4.2 cannot be a split artefact. We first take a uniform 10% subsample of the source prompt pool; Nine-by-30k lives entirely inside this slice. From it, we hold out 5,000 prompts as test and 3,000 as validation, leaving 23,027 prompts as the training set. Every model is evaluated on every prompt in Nine-by-30k, so coverage is fully dense.

Nine-by-300k reuses Nine-by-30k's test and validation splits without modification and replaces the training set with 302,371 prompts from the expanded source pool, excluding the held-out validation and test queries. With router and encoder settings held fixed, the shared validation and test splits let us isolate the effect of training-data scale. Coverage on the extension is dense: every prompt is scored by all nine models, giving $9 \times 310{,}371 = 2{,}793{,}339$ query–model correctness labels in Nine-by-300k overall.

Nine-by-30k is the basis for the main method grid (Sec. 4.1); Nine-by-300k is the basis for the fine-tuning analysis (Sec. 6).

A.2.3 Position Relative to Existing Benchmarks

Tab. 7 places Nine-by-30k and Nine-by-300k alongside RouterBench, EmbedLLM, R2-Bench, and CARROT/SPROUT, the four routing benchmarks we treat as prior art. The "queries" columns count unique prompts, not (prompt, model) entries. Nine-by-30k sits in the same scale band as the four existing benchmarks and serves as the apples-to-apples comparison set throughout this paper. Nine-by-300k provides a larger training set with the same candidate models and held-out queries, allowing us to vary the data scale without changing the evaluation setting.

B Additional Experimental Results

B.1 Statistical Comparisons of the Top-Five Routers

We test whether the small accuracy differences among the top-five methods in Tab. 1 are statistically detectable. For each benchmark, we identify the five highest-scoring methods using their encoder-averaged test accuracies.

Paired bootstrap. For routers A and B and test query q_i, let $d_i = s_A(q_i) - s_B(q_i)$, where each router's per-query score is averaged over the same encoders and seeds as its table entry. We resample the N test queries with replacement $10{,}000$ times and recompute the mean difference to obtain a 95% confidence interval and a two-sided bootstrap p-value. The comparison is paired because both routers are evaluated on the same queries, and it applies to both binary and fractional correctness scores. We apply Holm's correction at $\alpha = 0.05$ across the available pairwise comparisons among the top five on each benchmark.

Coverage and results. The test requires saved per-query predictions for every run contributing to a method's table entry; incomplete predictions limit coverage on four benchmarks. Tab. 9 summarizes coverage and the largest observed gap on each benchmark. None of the top-five comparisons is significant after Holm correction.

Table 9: Top-five bootstrap comparisons. Each row shows the tested pair with the largest observed accuracy gap. Confidence intervals are unadjusted, and p-values are Holm-adjusted over all tested pairs.

Benchmark	Pair ($A - B$)	Gap [95% CI] (pp)	p_{Holm}
RouterBench	kNN–KMeans	0.036 $[-0.039, 0.112]$	1.000
Nine-by-30k	Zooter–Elo	0.038 $[-0.289, 0.370]$	1.000
R2-Bench	Zooter–Avengers-Pro	0.226 $[0.028, 0.419]$	0.232
CARROT/SPROUT	Elo–KMeans	0.210 $[0.021, 0.404]$	0.172
EmbedLLM	Zooter–Elo	0.492 $[-0.117, 1.109]$	0.350

B.2 Per-Benchmark Cost-Utility Tables

The four tables below report realized cost-utility on RouterBench, EmbedLLM, R2-Bench, and CARROT/SPROUT under the same selection protocol as Tab. 2 in the main paper: 16 calibrated methods (15 available on EmbedLLM), val-selected architecture per encoder, mean over available encoders with cross-encoder std reported as $\pm$. Costs are normalized by the maximum candidate cost for each query; utility values below are unscaled, whereas Tab. 2 reports $100U$. Methods are sorted by λ=0.5 utility within each table.

Table 10: Realized cost-utility on RouterBench. Same protocol and column meaning as Tab. 2; methods sorted by λ=0.5.

	λ=0.0	λ=0.2	λ=0.5	λ=0.7	λ=0.9
kmeans	0.8035±0.000	**0.5787**±0.001	**0.3333**±0.001	0.1708±0.001	0.0340±0.000
zooter_lite	0.8038±0.000	0.5770±0.001	0.3269±0.001	0.1475±0.001	0.0339±0.000
p2l	0.8031±0.000	0.5768±0.001	0.3265±0.001	0.1453±0.002	0.0337±0.000
mess_plus	0.8035±0.000	0.5740±0.003	0.3261±0.001	0.1476±0.002	0.0339±0.000
embedllm	0.8032±0.000	0.5743±0.005	0.3259±0.001	0.1443±0.002	0.0337±0.000
knn	**0.8039**±0.001	0.5602±0.009	0.3256±0.008	**0.1716**±0.001	**0.0341**±0.000
equi	0.8036±0.000	0.5717±0.002	0.3254±0.001	0.1458±0.001	0.0337±0.000
mlp_cn	0.8029±0.000	0.5720±0.003	0.3254±0.001	0.1485±0.003	0.0338±0.000
avengers_pro	0.8035±0.001	0.5764±0.001	0.3055±0.002	0.1401±0.000	0.0336±0.000
rorf	0.7757±0.001	0.5442±0.001	0.3001±0.001	0.1375±0.000	0.0335±0.000
optllm_xgb	0.7668±0.003	0.5359±0.005	0.2982±0.001	0.1373±0.000	0.0335±0.000
carrot	0.8030±0.001	0.5728±0.007	0.2967±0.013	0.1414±0.001	0.0336±0.000
elo_router	0.8036±0.001	0.5716±0.007	0.2896±0.013	0.1408±0.001	0.0336±0.000
mirt	0.7636±0.059	0.4472±0.044	0.2473±0.009	0.1375±0.002	0.0335±0.000
cp_router	0.8029±0.001	0.4123±0.001	0.2401±0.001	0.1361±0.000	0.0335±0.000
irt_router	0.7749±0.062	0.3917±0.000	0.2381±0.000	0.1358±0.000	0.0335±0.000

Table 11: Realized cost-utility on EmbedLLM, with model parameter count used as a cost proxy. rorf has only one viable encoder; cross-encoder std is therefore omitted for that row.

	λ=0.0	λ=0.2	λ=0.5	λ=0.7	λ=0.9
kmeans	0.6667±0.009	0.4542±0.004	**0.2292**±0.003	0.1114±0.002	0.0219±0.001
knn	0.6716±0.009	**0.4543**±0.008	0.2281±0.004	**0.1120**±0.003	**0.0225**±0.001
zooter_lite	**0.6751**±0.003	0.4403±0.004	0.2231±0.003	0.1107±0.002	0.0202±0.000
mess_plus	0.6699±0.005	0.4388±0.005	0.2218±0.003	0.1092±0.001	0.0202±0.001
embedllm	0.6605±0.010	0.4315±0.007	0.2215±0.003	0.1089±0.002	0.0213±0.000
p2l	0.6638±0.011	0.4331±0.007	0.2214±0.003	0.1093±0.002	0.0213±0.001
equi	0.6708±0.005	0.4378±0.003	0.2208±0.003	0.1062±0.002	0.0198±0.001
mlp_cn	0.6651±0.007	0.4298±0.008	0.2200±0.003	0.1085±0.002	0.0212±0.000
carrot	0.6698±0.011	0.4173±0.007	0.2019±0.007	0.0897±0.001	0.0191±0.000
elo_router	0.6702±0.007	0.4176±0.006	0.2018±0.006	0.0892±0.002	0.0191±0.000
avengers_pro	0.6661±0.009	0.4181±0.007	0.1995±0.004	0.0885±0.002	0.0189±0.000
rorf	0.6443	0.4033	0.1811	0.0878	0.0189
mirt	0.5920±0.069	0.2146±0.019	0.1246±0.005	0.0706±0.001	0.0187±0.000
cp_router	0.6609±0.007	0.2160±0.003	0.1220±0.001	0.0700±0.000	0.0187±0.000
irt_router	0.5206±0.074	0.1983±0.000	0.1214±0.000	0.0700±0.000	0.0187±0.000

Table 12: Realized cost-utility on R2-Bench, with token count used as a cost proxy. Negative values at λ=0.9 occur when the weighted normalized-cost term exceeds the weighted accuracy term.

	λ=0.0	λ=0.2	λ=0.5	λ=0.7	λ=0.9
knn	0.8208±0.002	**0.6139**±0.001	**0.3242**±0.001	**0.1475**±0.000	**−0.0082**±0.000
kmeans	0.8187±0.002	0.6125±0.001	0.3232±0.001	0.1467±0.000	−0.0083±0.000
equi	0.8180±0.003	0.6060±0.001	0.3139±0.001	0.1437±0.000	−0.0088±0.000
zooter_lite	**0.8212**±0.001	0.6069±0.001	0.3132±0.001	0.1434±0.000	−0.0088±0.000
mlp_cn	0.8138±0.004	0.6070±0.001	0.3131±0.001	0.1431±0.000	−0.0088±0.000
mess_plus	0.8169±0.002	0.6073±0.002	0.3130±0.001	0.1433±0.001	−0.0088±0.000
p2l	0.8156±0.003	0.6048±0.002	0.3094±0.001	0.1416±0.001	−0.0091±0.000
embedllm	0.8140±0.005	0.6035±0.003	0.3087±0.003	0.1412±0.001	−0.0091±0.000
optllm_xgb	0.8043±0.003	0.5953±0.002	0.3064±0.000	0.1369±0.000	−0.0098±0.000
rorf	0.8160±0.003	0.6002±0.002	0.3004±0.001	0.1336±0.001	−0.0100±0.000
carrot	0.8154±0.002	0.5923±0.001	0.3001±0.000	0.1342±0.000	−0.0100±0.000
elo_router	0.8208±0.002	0.5931±0.001	0.2996±0.001	0.1340±0.000	−0.0101±0.000
avengers_pro	0.8189±0.001	0.5897±0.001	0.2979±0.001	0.1334±0.000	−0.0101±0.000
mirt	0.8094±0.006	0.5295±0.068	0.2785±0.025	0.1295±0.010	−0.0103±0.001
cp_router	0.8193±0.002	0.5244±0.002	0.2628±0.000	0.1216±0.000	−0.0113±0.000
irt_router	0.8104±0.005	0.4442±0.003	0.2473±0.001	0.1177±0.000	−0.0116±0.000

Table 13: Realized cost-utility on CARROT/SPROUT. Same protocol as Tab. 2.

	λ=0.0	λ=0.2	λ=0.5	λ=0.7	λ=0.9
knn	0.8678±0.002	**0.6658**±0.002	**0.3948**±0.001	**0.2210**±0.000	**0.0625**±0.000
kmeans	0.8664±0.001	0.6641±0.002	0.3945±0.001	0.2201±0.000	0.0622±0.000
zooter_lite	0.8664±0.001	0.6612±0.001	0.3891±0.001	0.2151±0.001	0.0613±0.000
mess_plus	0.8661±0.001	0.6599±0.001	0.3880±0.001	0.2134±0.001	0.0612±0.000
p2l	0.8632±0.003	0.6588±0.003	0.3879±0.002	0.2129±0.001	0.0611±0.000
equi	0.8646±0.002	0.6580±0.002	0.3875±0.002	0.2145±0.001	0.0613±0.000
mlp_cn	0.8612±0.002	0.6558±0.003	0.3875±0.002	0.2131±0.001	0.0611±0.000
embedllm	0.8634±0.003	0.6555±0.003	0.3850±0.005	0.2112±0.003	0.0609±0.000
rorf	0.8626±0.003	0.6578±0.004	0.3808±0.001	0.2068±0.001	0.0608±0.000
elo_router	**0.8685**±0.002	0.6594±0.002	0.3767±0.002	0.2054±0.001	0.0602±0.000
carrot	0.8637±0.003	0.6574±0.002	0.3759±0.002	0.2060±0.001	0.0601±0.000
optllm_xgb	0.8512±0.002	0.6344±0.003	0.3746±0.001	0.2093±0.000	0.0595±0.000
avengers_pro	0.8674±0.001	0.6581±0.001	0.3738±0.001	0.2041±0.001	0.0599±0.000
cp_router	0.8665±0.003	0.5754±0.006	0.3246±0.001	0.1873±0.000	0.0568±0.000
mirt	0.8488±0.012	0.5369±0.085	0.3193±0.040	0.1838±0.017	0.0561±0.003
irt_router	0.8478±0.010	0.4714±0.016	0.2886±0.005	0.1705±0.002	0.0535±0.000

B.3 Per-Encoder Routing Accuracy

The following tables disaggregate Tab. 1 by query encoder. Each cell is the test accuracy of the validation-selected architecture. The final columns give the mean and population standard deviation across available encoders.

Table 14: RouterBench: per-encoder test routing accuracy (%). Mean and SD summarize the encoder columns; SD is in percentage points.

Router	MiniLM	MPNet	Qwen	BGE	MBB	MBL	Mean	SD
Zooter	80.38	80.34	80.42	80.36	80.44	80.37	80.38	0.03
kNN	80.38	80.41	80.38	80.31	80.51	80.33	80.39	0.06
Elo	80.32	80.23	80.42	80.38	80.50	80.34	80.36	0.08
Avengers-Pro	80.34	80.37	80.31	80.25	80.43	80.34	80.34	0.05
EquiRouter	80.39	80.35	80.34	80.33	80.40	80.35	80.36	0.02
KMeans	80.30	80.31	80.41	80.31	80.42	80.35	80.35	0.05
MESS+	80.29	80.29	80.39	80.37	80.40	80.35	80.35	0.04
CARROT	80.12	80.38	80.33	80.31	80.41	80.27	80.30	0.09
CP-Router	80.28	80.30	80.31	80.31	80.40	80.15	80.29	0.07
P2L	80.36	80.34	80.27	80.33	80.30	80.25	80.31	0.04
MLP	80.31	80.27	80.25	80.32	80.33	80.24	80.29	0.04
EmbedLLM	80.34	80.27	80.32	80.37	80.36	80.26	80.32	0.04
ModelSpider	80.31	80.30	80.36	80.24	80.32	80.30	80.31	0.03
RoRF	77.63	77.61	77.59	77.59	77.30	77.69	77.57	0.13
TagRouter	80.27	80.27	80.47	80.27	80.42	—	80.34	0.09
LinUCB	80.29	80.23	79.16	80.31	55.80	64.71	73.41	9.66
RouterDC	80.20	79.97	80.03	80.18	80.33	80.31	80.17	0.13
MIRT	80.32	80.30	63.82	80.34	78.48	74.89	76.36	5.93
IRT-Router	80.27	80.27	63.61	80.27	80.27	80.25	77.49	6.21
HybridLLM	71.67	71.67	71.67	71.67	71.67	71.67	71.67	0.00
OptLLM	77.38	76.70	76.41	76.71	76.44	76.46	76.68	0.33

Table 15: Nine-by-30k: per-encoder test routing accuracy (%). Mean and SD summarize the encoder columns; SD is in percentage points.

Router	MiniLM	MPNet	Qwen	BGE	MBB	MBL	Mean	SD
Zooter	68.18	68.40	68.57	68.38	68.35	68.41	68.38	0.11
kNN	67.68	68.29	68.51	67.75	68.89	69.07	68.36	0.53
Elo	67.80	68.42	68.90	67.70	68.99	68.25	68.34	0.49
Avengers-Pro	68.27	68.20	68.50	68.22	68.68	67.99	68.31	0.22
EquiRouter	67.81	68.18	68.15	68.35	68.47	68.49	68.24	0.23
KMeans	68.26	68.58	68.30	68.01	68.88	68.25	68.38	0.28
MESS+	68.09	68.17	68.08	68.12	68.30	68.44	68.20	0.13
CARROT	67.44	68.23	68.53	67.56	68.91	68.66	68.22	0.55
CP-Router	68.33	68.30	68.27	68.39	67.74	68.30	68.22	0.22
P2L	67.78	67.21	68.05	67.86	68.31	68.47	67.95	0.41
MLP	67.62	67.63	68.05	68.07	68.27	68.22	67.98	0.26
EmbedLLM	67.28	67.34	67.99	67.93	68.17	68.18	67.81	0.37
ModelSpider	67.56	67.44	67.76	67.75	67.63	67.50	67.61	0.12
RoRF	67.58	67.76	68.25	67.53	—	68.18	67.86	0.30
TagRouter	67.34	67.34	68.69	67.34	68.83	68.30	67.97	0.65
LinUCB	67.58	68.20	67.78	67.77	67.74	67.51	67.76	0.22
RouterDC	67.35	67.59	67.69	67.40	67.53	67.67	67.54	0.13
MIRT	67.40	67.34	65.42	67.92	66.08	65.71	66.64	0.95
IRT-Router	67.36	67.34	64.17	67.34	67.35	65.35	66.49	1.27
HybridLLM	60.46	60.46	60.46	60.46	60.46	60.46	60.46	0.00
OptLLM	—	66.55	66.63	66.52	66.86	66.48	66.61	0.14

Table 16: EmbedLLM: per-encoder test routing accuracy (%). Mean and SD summarize the encoder columns; SD is in percentage points.

Router	MiniLM	MPNet	Qwen	BGE	MBB	MBL	Mean	SD
Zooter	66.86	67.35	67.75	67.43	67.80	67.86	67.51	0.35
kNN	66.30	67.90	67.80	65.63	68.00	67.30	67.16	0.89
Elo	66.97	66.87	67.90	65.87	67.90	66.60	67.02	0.72
Avengers-Pro	65.73	65.47	67.73	66.07	67.93	67.00	66.66	0.96
EquiRouter	66.36	66.73	67.07	67.13	67.29	67.87	67.08	0.47
KMeans	65.07	66.67	67.63	66.20	67.27	67.20	66.67	0.85
MESS+	66.27	67.22	67.19	66.23	67.45	67.59	66.99	0.54
CARROT	66.57	67.27	67.53	64.73	68.23	67.57	66.98	1.12
CP-Router	65.87	66.77	65.20	65.93	65.63	67.17	66.09	0.67
P2L	64.59	65.95	66.63	65.89	67.51	67.69	66.38	1.06
MLP	65.43	65.97	66.69	66.43	67.18	67.38	66.51	0.67
EmbedLLM	64.29	65.37	66.52	66.06	66.85	67.19	66.05	0.98
ModelSpider	63.93	66.09	66.04	64.52	66.63	66.26	65.58	0.99
RoRF	64.43	—	—	—	—	—	64.43	0.00
TagRouter	61.17	61.17	—	60.90	—	—	61.08	0.13
LinUCB	64.07	65.80	66.83	65.57	65.80	66.50	65.76	0.88
RouterDC	57.93	58.21	58.68	57.14	58.32	58.69	58.16	0.53
MIRT	65.26	66.13	49.77	66.35	55.14	52.54	59.20	6.90
IRT-Router	58.53	60.51	46.41	59.18	43.96	43.75	52.06	7.42
HybridLLM	46.87	46.87	46.87	46.87	46.87	46.87	46.87	0.00
OptLLM	—	—	—	—	—	—	—	—

Table 17: R2-Bench: per-encoder test routing accuracy (%). Mean and SD summarize the encoder columns; SD is in percentage points.

Router	MiniLM	MPNet	Qwen	BGE	MBB	MBL	Mean	SD
Zooter	81.99	82.09	82.21	82.11	82.21	82.09	82.12	0.08
kNN	81.97	82.01	82.37	81.90	82.16	82.08	82.08	0.15
Elo	82.01	82.08	82.37	81.86	82.05	82.13	82.08	0.15
Avengers-Pro	81.74	81.72	82.12	81.95	81.95	81.86	81.89	0.14
EquiRouter	81.26	81.72	82.08	81.87	81.88	81.99	81.80	0.26
KMeans	81.60	82.32	81.97	81.96	81.62	81.79	81.87	0.24
MESS+	81.33	81.53	82.12	81.62	81.80	81.72	81.69	0.24
CARROT	81.40	81.81	81.57	81.28	81.45	81.73	81.54	0.19
CP-Router	81.72	82.13	82.25	82.13	81.61	81.76	81.93	0.24
P2L	80.98	81.39	82.09	81.62	81.65	81.61	81.56	0.33
MLP	80.87	81.20	81.88	81.00	81.72	81.62	81.38	0.38
EmbedLLM	80.46	81.32	81.92	81.58	81.45	81.69	81.40	0.46
ModelSpider	81.39	81.56	81.83	81.40	81.64	81.65	81.58	0.15
RoRF	81.36	81.64	82.23	81.14	81.51	81.73	81.60	0.34
TagRouter	81.10	81.10	82.06	81.10	81.67	81.40	81.41	0.36
LinUCB	80.36	80.97	81.07	80.82	80.60	80.47	80.72	0.26
RouterDC	79.75	79.71	80.27	79.81	81.27	80.51	80.22	0.55
MIRT	81.33	81.36	79.66	81.44	81.30	80.55	80.94	0.64
IRT-Router	81.21	81.35	80.01	81.14	81.33	81.20	81.04	0.47
HybridLLM	73.99	73.99	73.99	73.99	73.99	73.99	73.99	0.00
OptLLM	80.24	80.76	80.84	80.33	80.08	80.31	80.43	0.28

Table 18: CARROT/SPROUT: per-encoder test routing accuracy (%). Mean and SD summarize the encoder columns; SD is in percentage points.

Router	MiniLM	MPNet	Qwen	BGE	MBB	MBL	Mean	SD
Zooter	86.51	86.69	86.65	86.61	86.70	86.67	86.64	0.06
kNN	86.59	87.13	86.82	86.65	86.54	86.97	86.78	0.21
Elo	86.59	87.18	87.04	86.68	86.78	86.86	86.85	0.20
Avengers-Pro	86.74	86.64	86.89	86.53	86.85	86.88	86.75	0.13
EquiRouter	86.10	86.48	86.57	86.45	86.53	86.63	86.46	0.17
KMeans	86.46	86.60	86.84	86.66	86.72	86.59	86.64	0.12
MESS+	86.68	86.59	86.88	86.54	86.51	86.47	86.61	0.14
CARROT	86.08	86.47	86.44	86.40	86.02	86.83	86.37	0.27
CP-Router	86.77	86.74	87.02	86.81	86.29	86.27	86.65	0.28
P2L	85.89	86.17	86.62	86.19	86.52	86.53	86.32	0.26
MLP	85.87	85.88	86.32	85.92	86.22	86.49	86.12	0.24
EmbedLLM	85.85	86.42	86.33	86.70	86.34	86.39	86.34	0.25
ModelSpider	86.28	86.21	86.52	86.32	86.55	86.50	86.40	0.13
RoRF	85.68	85.97	86.75	86.30	86.44	86.40	86.26	0.34
TagRouter	84.82	84.82	—	—	—	—	84.82	0.00
LinUCB	85.55	85.87	86.19	86.33	86.04	85.22	85.87	0.38
RouterDC	83.06	82.97	83.20	83.05	83.23	83.29	83.13	0.11
MIRT	86.12	86.08	83.66	86.00	83.96	83.45	84.88	1.20
IRT-Router	85.52	85.78	83.69	86.04	83.97	83.68	84.78	1.02
HybridLLM	77.62	77.62	77.62	77.62	77.62	77.62	77.62	0.00
OptLLM	84.77	85.31	85.05	85.33	84.86	85.40	85.12	0.24

B.4 Benchmark Variations and Model-Pool Diversity

We test whether the plateau persists under changes in query difficulty and task composition, and whether the candidate models offer complementary strengths.

B.4.1 Test-set slicing

We retain test queries that at least half of the candidate models answer correctly (task score above 0.5), then rescore the available predictions from Tab. 1 with the same averaging and no retraining. Both router and oracle accuracies retain partial-credit scores. On these easier queries, the oracle gap shrinks, but the leading routers remain closely grouped and below the oracle (Tab. 19).

Table 19: Full test set vs. queries that at least half of the pool solves. Gap is oracle minus best-router accuracy; spread is the top-five accuracy range on the filtered queries. Results use routers with available saved predictions, ranked separately on each subset.

	Queries	Gap to oracle (pp)		Top-5 spread (pp)
Benchmark	kept	full	filtered	filtered
RouterBench	66.6%	11.30	2.82	0.03
Nine-by-30k	64.4%	14.27	5.67	0.23
EmbedLLM	38.6%	30.46	4.64	1.44
R2-Bench	67.5%	12.69	3.51	0.18
CARROT/SPROUT	77.5%	11.05	3.31	0.15

B.4.2 Task-domain groups

We group all 5,000 Nine-by-30k test queries by source-dataset domain and compare the same 12 available routers within each group, without retraining. In the four domains with at least 100 queries, the top-five spread is 0.22–1.22 pp, while the best router remains 6.51–15.62 pp below the oracle (Tab. 20). Thus, the plateau also appears within task domains. The two smaller groups are reported for completeness.

Table 20: Domain breakdown on Nine-by-30k for the 12 methods with recoverable predictions from Tab. 1. The oracle gap is measured from the best router within each domain; the top-five spread is the difference between the highest and fifth-highest router accuracies within that domain.

Task group	Queries	Oracle gap (pp)	Top-5 spread (pp)
Math	128	6.51	0.91
Knowledge/commonsense/safety	1,174	10.53	0.22
Reading comprehension/QA	3,387	15.62	0.22
SuperGLUE	278	14.81	1.22
Code	16	5.42	1.88
Other specialized tasks	17	6.67	2.55
All queries	5,000	14.27	0.16

B.4.3 Retraining on filtered benchmarks

We also filter each Nine-by-30k split to queries answered correctly by at least three or five models, using the same correctness threshold as above. We retrain the six routers in Tab. 21 with ModernBERT-base and MiniLM-L6 over three seeds, comparing them with full-data training on the same filtered test queries. Filtering the training data yields little improvement for five routers and lowers MLP accuracy; the best router still trails the oracle by 5.5–9.7 pp.

Table 21: Test accuracy (%) after training on filtered Nine-by-30k data, averaged over two encoders and three seeds. Δ is the change from full-data training, in pp, evaluated on the same filtered queries. Original task scores are retained, so passing the correctness threshold does not imply an oracle score of 100%.

	≥ 3 models correct (3,663 queries)		≥ 5 models correct (3,219 queries)	
Router	Accuracy	Δ	Accuracy	Δ
KMeans	89.53	$+0.03$	93.81	$+0.10$
Elo	89.38	-0.06	93.42	-0.32
kNN	89.33	$+0.11$	93.38	-0.20
RouterDC	88.06	-0.17	92.47	-0.19
MIRT	87.28	$+0.18$	92.21	$+0.08$
MLP (CE)	85.11	-2.20	91.06	-1.12
Oracle	99.22		99.35	

B.4.4 Model-pool diversity and specialization

The oracle exceeds the best single model by 11.4–37.2 pp across the five benchmarks. Thus, the plateau occurs in pools where selecting different models for different queries offers substantial potential gains.

Together, these checks support the main-text finding that leading routers remain closely grouped below the oracle across the benchmark variations we test, even when the candidate models have complementary strengths.

B.5 Multi-Sample Correctness Labels

To assess the sensitivity of the oracle to single-sample response labels, we collected ten independent responses for each of the nine models on the same 5,000 Nine-by-30k test queries and aggregated their correctness by majority vote. API requests used provider defaults without temperature or top-p overrides, matching the original collection.

Tab. 22 reports the resulting majority-label oracles. Its gap column subtracts the original best-router accuracy, 0.684, as a fixed reference. The majority-label oracle remains close to the single-sample oracle, supporting the observation that much of the pool's apparent routing opportunity survives repeated sampling.

Table 22: Oracle accuracy on Nine-by-30k under single-sample and majority-vote correctness, with the original best-router accuracy (0.684, Tab. 1) held fixed as a reference.

Correctness label	Oracle	Gap to fixed reference
Single sample (labels used in this paper)	0.827	14.3 pp
k=10 majority	0.817	13.3 pp

C Analysis Details

C.1 Hard-Query Analysis and Model-Selection Distributions

The behavioral analyses use the 12 routers shown in Fig. 4 and below, with ModernBERT-base, seed 42, and validation-selected architectures. We convert task scores to binary correctness labels, $y_{im} = \mathbf{1}[s_{im} \geq 0.5]$, and use the hard/easy split from Sec. 5.1. In Tab. 3, the hard-query gap share is the hard-query fraction multiplied by the hard-subset gap, divided by the full-test gap.

The figures extend the main-text comparison to nine additional routers on all five benchmarks. Each router histogram shows its selection frequencies; "Truth" shows each model's share of all correct query–model pairs on hard queries. Models are ordered by training accuracy, with the dotted line marking the top-ranked model. Router selections remain concentrated on globally strong models, while correct answers on hard queries are more evenly spread across the pool.

Rows show (Elo, P2L, Zooter), (CARROT, CP-Router, Avengers-Pro), and (EmbedLLM, KMeans, EquiRouter), using the same panel format as Fig. 4.

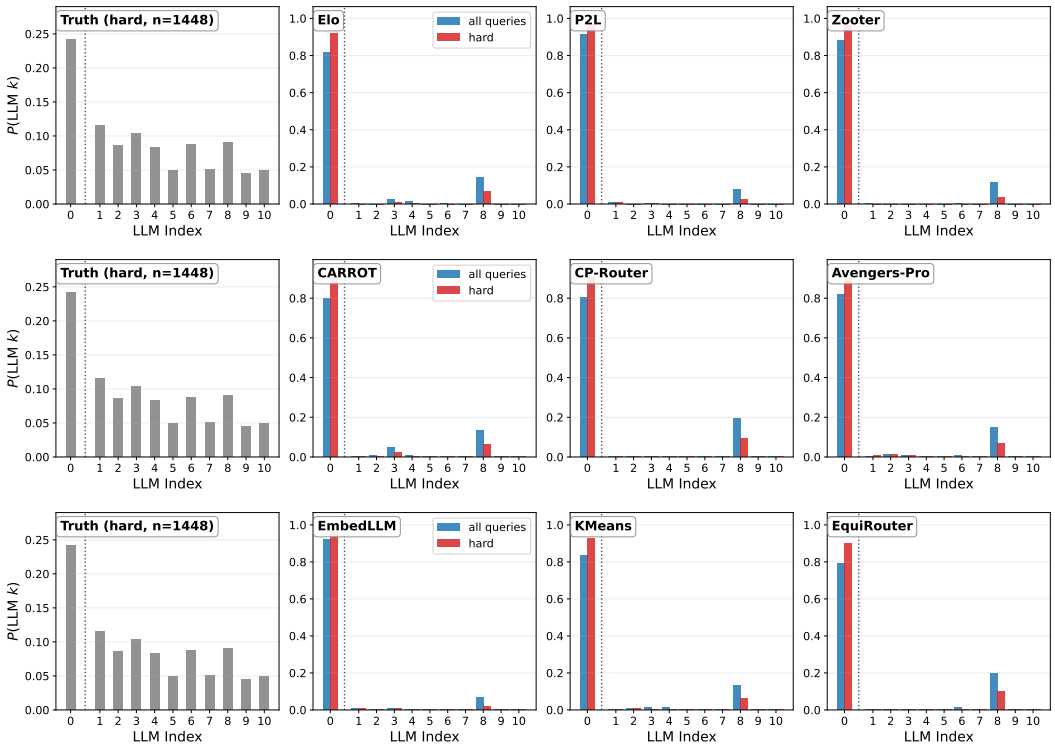


Figure 6: RouterBench: selection frequency on hard vs. all queries for nine additional routers.

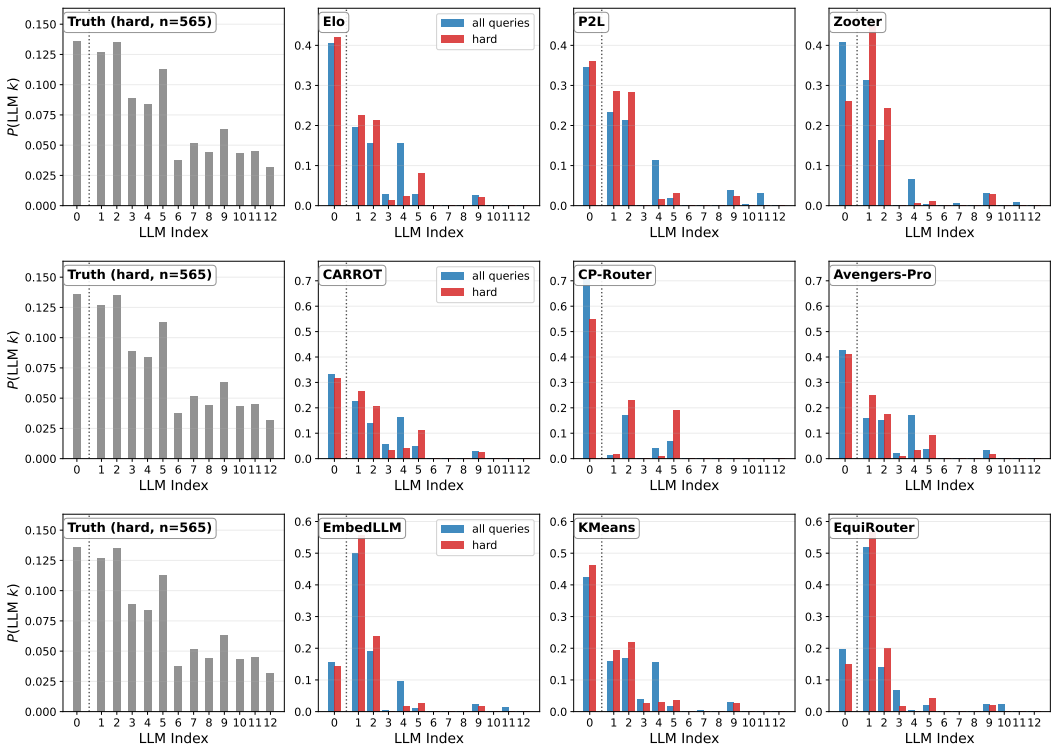


Figure 7: CARROT/SPROUT: selection frequency on hard vs. all queries for nine additional routers.

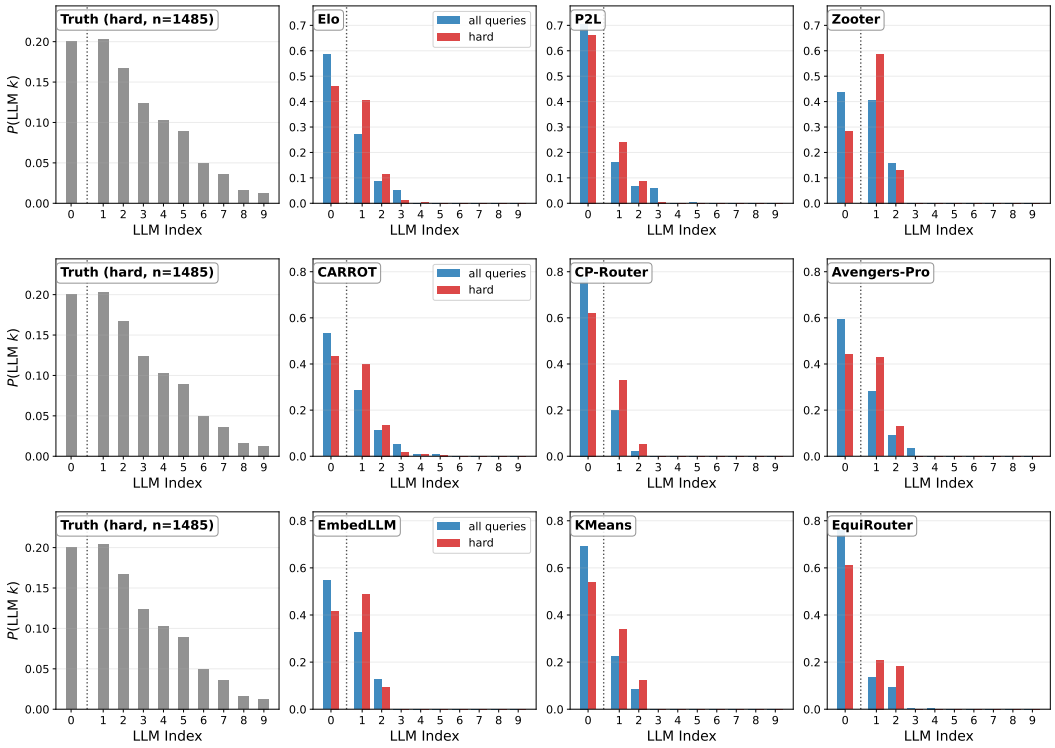


Figure 8: R2-Bench: selection frequency on hard vs. all queries for nine additional routers.

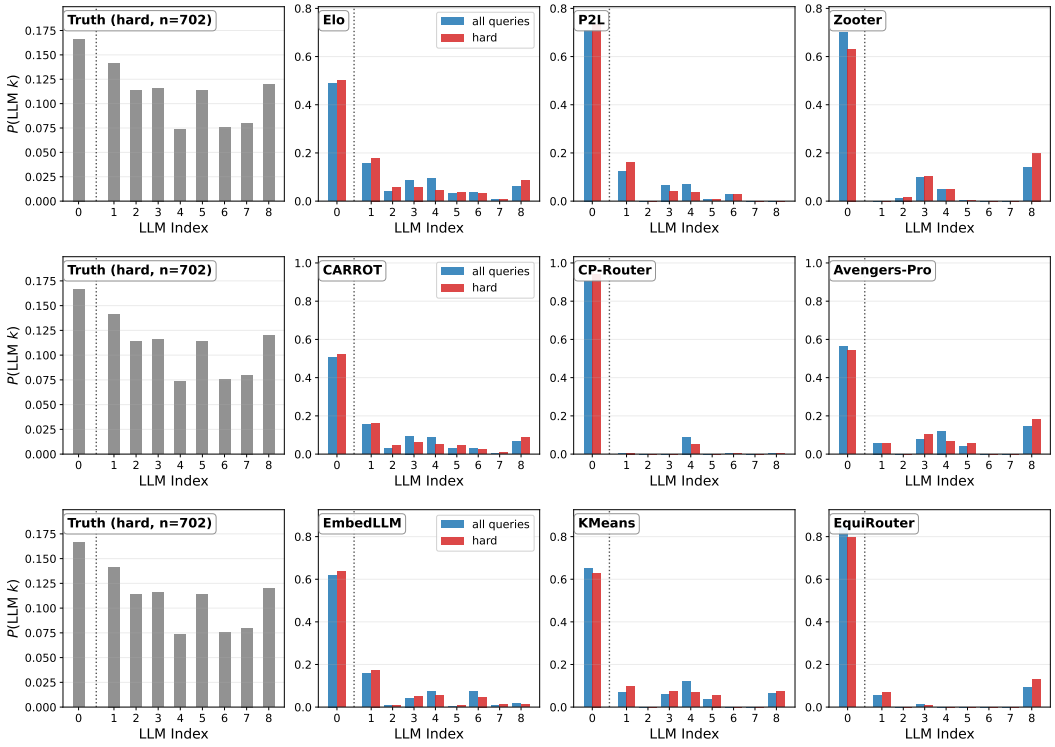


Figure 9: Nine-by-30k: selection frequency on hard vs. all queries for nine additional routers.

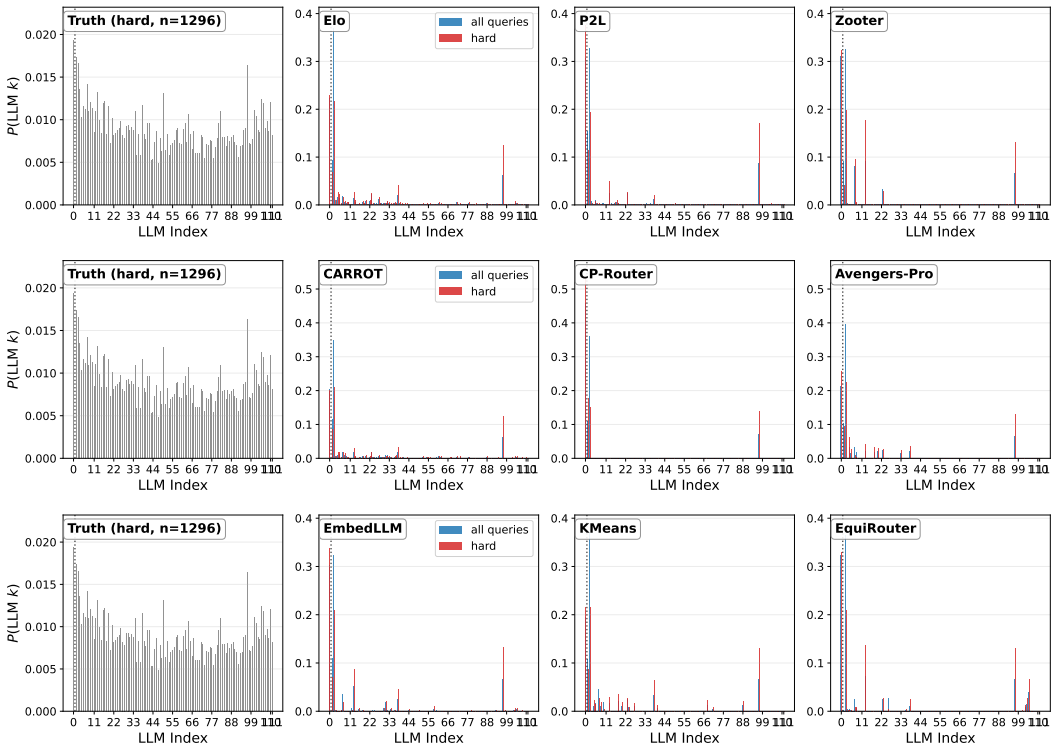


Figure 10: EmbedLLM: selection frequency on hard vs. all queries for nine additional routers.

C.2 Pairwise Error Exchange

We compare all 66 pairs of the 12 routers in App. C.1. For each pair, we identify queries where the routers select different models. Within these disagreements, unsigned exchange is the fraction on which exactly one router succeeds; signed exchange is the difference between their win rates. Tab. 4 reports the range of disagreement rates and medians across pairs for the other metrics. Each exchange ratio is computed before taking the median, so it need not equal the ratio of the reported medians.

C.3 kNN Comparison Setup

We compute concordance as defined in Sec. 3, using the diagnostic runs in App. C.1. The comparison includes kNN and 11 other routers. The hard-query median uses nine routers: MLP, Elo, MESS+, P2L, Zooter, CARROT, CP-Router, Avengers-Pro, and EmbedLLM.

D Scaling and Fine-Tuning

D.1 Scaling and Fine-Tuning Setup

The eight Scaled-FT runs jointly train the encoder and routing head with seed 42, batch size 32, encoder/head learning rates of $2 \times 10^{-5}/10^{-3}$, weight decay of 0.01, and gradient clipping at norm 1.0. The frozen kNN reference uses the scaled training set and ModernBERT-large embeddings with $k = 750$.

D.2 Encoder Scaling and Fine-Tuning on Other Benchmarks

On CARROT/SPROUT and EmbedLLM, we keep the original training sets and compare ModernBERT-base (mbb) and ModernBERT-large (mbl), each frozen or fine-tuned, using the same eight routers as Sec. 6.

Table 23: Encoder scaling and fine-tuning on CARROT/SPROUT and EmbedLLM. Means are across eight routers; best FT and best kNN are the strongest configurations across encoder choices. Δ is best FT minus best kNN, in pp.

Benchmark	frzn mbb	frzn mbl	FT mbb	FT mbl	best FT cell	best kNN ref	Δ pp
CARROT/SPROUT	0.8604	0.8609	0.8645	0.8654	0.8711 (`routerdc`, mbl)	0.8697 (mbl)	$+0.14$
EmbedLLM	0.6600	0.6643	0.6779	0.6764	0.6817 (`routerdc`, mbb)	0.6800 (mbb)	$+0.17$

Fine-tuning the large encoder improves mean accuracy over the frozen-base baseline by 0.50 and 1.64 pp, respectively. The best fine-tuned routers remain close to frozen kNN, exceeding it by only 0.14 and 0.17 pp.