ROUTE BY THINK, NOT JUST SEMANTICS—LLM ROUTING VIA COGNITIVE EMBEDDINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) exhibit highly heterogeneous capabilities across domains and tasks, making it increasingly important to route each query to the most cost-effective model that can answer it reliably. Existing LLM routers typically rely on textual features or Semantic Embeddings of queries. However, semantic similarity does not necessarily imply routing similarity. Specifically, semantically similar queries may require substantially different capabilities, while semantically different queries may be best handled by the same model. In fact, humans typically form an internal understanding of a question before producing an answer. Likewise, an LLM progressively transforms a query into contextualized hidden states, which we call Cognitive Embeddings, before generating its response, motivating us to explore whether such representations provide richer signals for routing. Based on this insight, we present CogRouter, an LLM routing framework that routes queries according to their Cognitive Embeddings from a generative model and matches them against candidate model capabilities. Each candidate is represented by combining its capability description with a learnable Cognitive Embedding derived from historical performance, capturing complementary semantic and empirical capability signals. To avoid discarding equally best routing choices, we further propose a tie-aware pairwise training strategy that preserves all tied-best candidates and avoids artificial ordering among them. Extensive experiments on xRouteBench and RouterBench demonstrate that CogRouter outperforms state-of-the-art LLM routing methods by up to 13.81% and 20.09% in routing quality. Our code is available at https://anonymous.4open.science/r/CogRouter-F45C/.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities across a broad range of tasks, including question answering, mathematical reasoning, code generation, and document understanding. Meanwhile, the rapidly growing LLM ecosystem has made a diverse collection of models available, ranging from compact open-source models to highly powerful proprietary models. However, these models exhibit substantially different capabilities and inference costs. The most powerful model is not necessarily the best choice for every query, while always invoking the strongest model can incur unnecessarily high serving costs. This facilitates a fundamental quality and cost trade-off in real-world LLM services. To address this issue, LLM routing has emerged as a promising paradigm that dynamically selects an appropriate model for each query, with the goal of maintaining response quality while reducing inference cost (Ding et al., 2024, Ong et al., 2025, Hu et al., 2024).

To enable effective model selection, existing LLM routing approaches exploit different signals to determine which candidate is most suitable for an incoming query. Typically, these methods can be broadly classified into three categories according to how the routing decision is made, i.e., LLM-based routing, feature-based routing, and relationship-based routing. LLM-based methods directly employ a generative LLM to interpret the query and select an appropriate model based on candidate descriptions or other contextual information (Shen et al., 2023). Feature-based methods extract textual, semantic, or difficulty-related features from the query and directly map them to model choices or expected performance through lightweight classifiers or predictors (Ding et al., 2024, Feng et al., 2026b). Relationship-based methods explicitly represent both queries and candidate models, and learn their compatibility from historical response quality, preference feedback, or other interaction signals (Ong et al., 2025, Chen et al., 2024b, Feng et al., 2025).

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.

However, existing routing paradigms still suffer from several limitations. LLM-based routing introduces additional inference cost and latency because an extra LLM invocation is required before the target model is selected. Its decisions can also be sensitive to prompts and model descriptions, which may not faithfully reflect the models' actual performance on a given query. Feature-based routing is more efficient, but textual, semantic, or difficulty-related features mainly characterize what a query is about rather than what capabilities are required to solve it. As a result, semantically similar queries may favor different models, while semantically different queries may require similar underlying capabilities. Relationship-based routing explicitly models query-model compatibility, yet its effectiveness still depends heavily on how query requirements and model capabilities are represented and how historical feedback is utilized. In particular, candidate representations may fail to emphasize the capabilities relevant to the current query, while collapsing multiple equally best models into a single target discards valid routing choices. Therefore, *how to accurately capture the capabilities required by each query and reliably match them with candidate LLMs remains a key challenge in cost-effective LLM routing.*

Intuitively, answering a question is not merely a direct response to its surface form. Before producing an explicit answer, humans typically develop an internal understanding of the question, implicitly identifying what knowledge is involved and what kind of reasoning may be required. A generative LLM exhibits a similar processing pattern during inference, where the input is progressively transformed into contextualized hidden states before the first response token is generated. We define Cognitive Embeddings as the contextualized hidden-state representations formed during LLM input processing. We call the outputs of conventional embedding models Semantic Embeddings. Importantly, exploiting this process does not require the LLM to explicitly reason about or generate a routing decision. Instead, the Cognitive Embeddings already produced during query processing can be directly reused as routing signals. Our preliminary study in Section 3 further shows that our learned routing representations capture routing-relevant distinctions that are not well reflected by conventional semantic features. These observations suggest that how an LLM internally processes a query can provide a more informative and lightweight basis for routing than relying on surface semantics or explicit LLM judgments.

Motivated by these observations, we propose CogRouter, a lightweight LLM routing framework that selects models according to how a query is internally represented during LLM reasoning processing. For each query, CogRouter extracts Cognitive Embeddings from the intermediate hidden states of an open-source lightweight generative LLM, without requiring the LLM to explicitly generate any routing decision. To characterize candidate models, CogRouter combines capability descriptions with learnable Cognitive Embeddings derived from historical performance, allowing the router to capture both semantic capability priors and empirical model behavior. A lightweight matching module then estimates the compatibility between each query and candidate model and selects the most suitable one under the quality-cost objective. Moreover, multiple models may achieve the same best recorded quality on a query, and selecting a single training target discards other equally valid choices. We therefore design a tie-aware pairwise training strategy that preserves all tied-best candidates and avoids artificial ordering among them, while learning preferences against strictly lower-quality candidates. In summary, this paper includes the following main contributions.

- We reveal that semantic similarity does not necessarily imply routing similarity and propose CogRouter, a lightweight LLM routing framework that directly exploits the Cognitive Embeddings formed during query processing and matches them with hybrid model capability representations integrating semantic descriptions and empirically learned Cognitive Embeddings.
- We propose a tie-aware pairwise training strategy that preserves all tied-best candidates and avoids artificial ordering among them, learning preferences only between these candidates and those with strictly lower recorded quality.
- We conduct extensive experiments on xRouteBench and RouterBench. The results demonstrate that CogRouter consistently outperforms state-of-the-art LLM routing methods, improving routing quality by up to 13.81% and 20.09%, while comprehensive ablation studies further validate the effectiveness of its key designs.

2 PRELIMINARY AND RELATED WORK

2.1 PROBLEM FORMULATION

LLM routing selects a candidate large language model for an input query. For quality-first selection before any candidate generates a response, the decision rule is

$$\hat{m}(q) = \arg\max_{m \in \mathcal{M}} s(q, m), \tag{1}$$

where q is the query, $\mathcal{M} = \{m_1, \ldots, m_M\}$ contains M candidate LLMs, $s(q, m)$ estimates the suitability of LLM m for q, and $\hat{m}(q)$ is the selected LLM. Recorded response quality provides supervision for learning the routing score. The following approaches differ in the evidence they use and when they make the selection (Varangot-Reille et al., 2025, Moslem & Kelleher, 2026).

2.2 RELATED WORK

LLM-based Routing Methods. HuggingGPT generates model-call plans (Shen et al., 2023). Router-R1 learns multi-round routing and aggregation (Zhang et al., 2025a). FrugalGPT and AutoMix cascade models using answer-quality scores and self-verification, respectively (Chen et al., 2024a, Aggarwal et al., 2024). LLM-Blender ranks and fuses generated responses (Jiang et al., 2023). Generated plans or answers inform these decisions but add inference overhead.

Feature-based Routing Methods. Sentence encoders (Reimers & Gurevych, 2019, Gao et al., 2021) and LLM-derived embeddings (BehnamGhader et al., 2024, Wang et al., 2024) supply Semantic Embeddings. Feature-based routers predict correctness or candidate labels (Shnitzer et al., 2023, Feng et al., 2026b). Hybrid LLM predicts small-model sufficiency (Ding et al., 2024). ZOOTER distills reward-based targets (Lu et al., 2024). OptLLM and CARROT predict performance for cost-aware allocation (Liu et al., 2024, Somerstep et al., 2025). BEST-Route also allocates response samples (Ding et al., 2025). Hidden-state probes support error detection (Orgad et al., 2025).

Relation-based Routing Methods. Interaction, contrastive, and graph-based routers learn query–candidate compatibility (Ong et al., 2025, Chen et al., 2024b, Feng et al., 2025). EmbedLLM learns model embeddings (Zhuang et al., 2025). IRT-Router relates ability to difficulty (Song et al., 2025). ContextualRouter retrieves past performance (Varangot-Reille et al., 2026), while MetaLLM and MixLLM learn selection through bandit feedback (Nguyen et al., 2024, Wang et al., 2025).

Although existing methods improve model selection, important challenges remain in representing model suitability and constructing routing supervision. Decisions based on generated plans or responses incur additional inference overhead, while reducing each query to a single target model discards other equally best candidates. Relationship-based methods explicitly learn query–model compatibility, but the relevance of different capability signals can vary across queries. To address these challenges, CogRouter matches query Cognitive Embeddings with candidate representations that combine semantic descriptions and empirically learned Cognitive Embeddings through a query-dependent gate. Its tie-aware objective retains all observed best candidates and learns preferences only against strictly lower-quality candidates.

To the best of our knowledge, CogRouter is the first to combine query-gated model capability representations with tied-best pairwise supervision for LLM routing.

3 PRESTUDY

Experimental Setting. To examine whether routing-relevant information can be captured beyond surface semantics, we conduct a prestudy on RouterBench using 132 queries, including 114 HellaSwag queries and 18 MMLU Professional Accounting queries. We focus on these two groups because they exhibit substantial overlap in the frozen Semantic Embedding space, despite requiring noticeably different knowledge and reasoning capabilities. We compare the frozen Semantic Embedding with the representations induced by several existing routers and our learned routing representations, given by the query–model matching score vectors. For visualization, each representation is projected into a two-dimensional space using PCA.

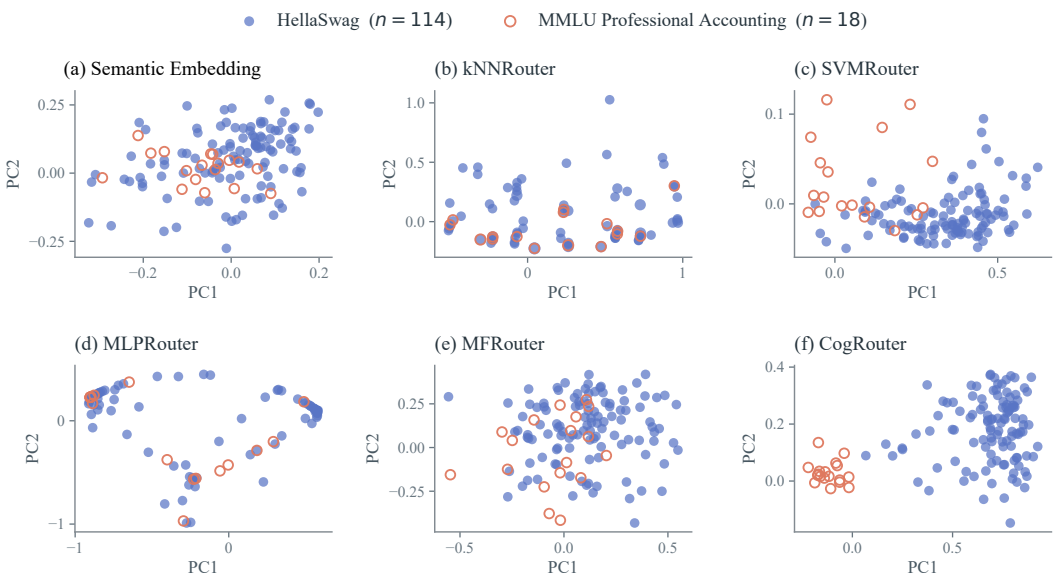


Figure 1. Prestudy on the same 132 RouterBench queries. Classifier probabilities and MFRouter/CogRouter matching scores are centered and unit-normalized per query. Each PCA is fitted on the complete source-cluster test context without task labels. Filled and open circles denote tasks. Marker sizes reflect coincident same-task points. Appendix D specifies the outputs and exploratory case selection.

Observation. As shown in Figure 1, the frozen Semantic Embeddings of the two query groups are highly overlapped, indicating that semantic proximity alone cannot reliably distinguish their routing requirements. Similar overlap remains in several existing routing representations. In contrast, our learned routing representations yield a much clearer separation between the two groups. Quantitatively, the full-space silhouette score increases from 0.040 with the frozen Semantic Embedding to 0.583 with our learned routing representations. This result provides initial evidence that our method learns routing-relevant distinctions that are not well reflected by conventional semantic features.

Our Idea. Motivated by this observation, we argue that LLM routing should rely not only on what a query looks like semantically, but also on how it is internally represented after being processed by a generative LLM. These Cognitive Embeddings can provide a richer view of the knowledge and reasoning capabilities implied by the query, while avoiding the additional generation overhead of explicitly asking an LLM to make the routing decision. Based on this idea, we design our router to directly exploit these Cognitive Embeddings and match them against the capabilities of candidate models.

4 Methodology

4.1 Overview

CogRouter matches query requirements to candidate capabilities through four components (Figure 2). *Cognitive Embedding Generation* extracts a Cognitive Embedding from a frozen LLM backbone. *Model Cognitive Embedding Generation* uses the query to gate the combination of trainable candidate Cognitive Embeddings and encoded descriptions. *Model Routing* scores their compatibility and selects a candidate. *Tie-aware Pairwise Training* learns the matching components by comparing every tied-best model with every lower-quality model, without imposing an ordering among tied winners.

We next describe their implementation. Let q be a query, $\mathcal{M} = \{m_1, \ldots, m_M\}$ the candidate pool, t_m a candidate description, and y_{qm} its recorded response quality.

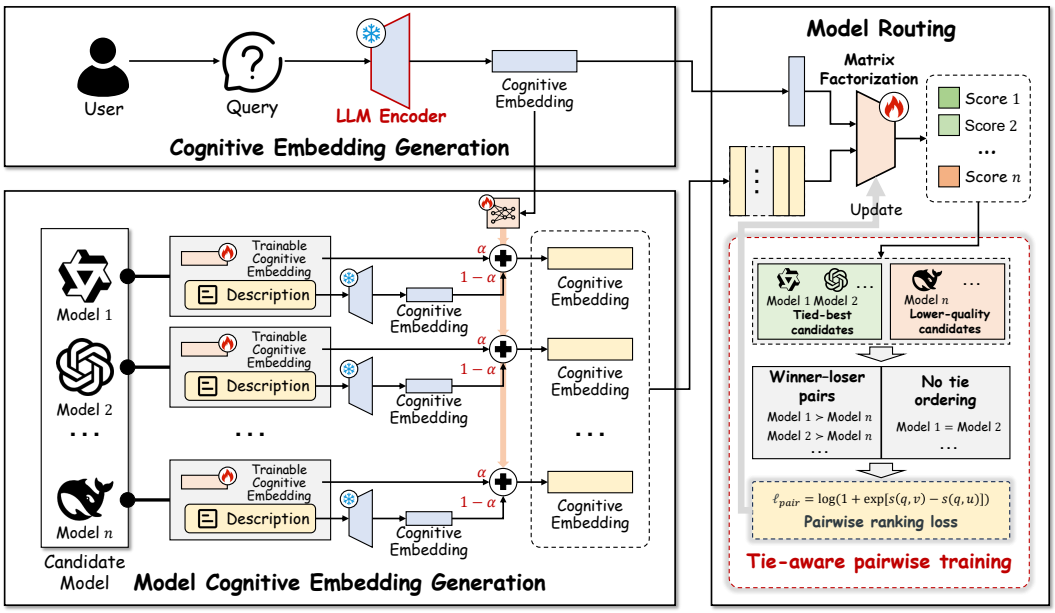


Figure 2. Overview of CogRouter. Cognitive Embedding extraction, query-gated candidate fusion, and matching produce routing scores. Training pairs are defined by recorded response quality. Tied winners remain valid choices without an ordering or equality constraint on their predicted scores. Frozen and trainable components are marked by snowflakes and flames, respectively.

4.2 Cognitive Embedding Generation

We extract contextualized hidden states from the frozen Qwen3-Embedding-0.6B Transformer backbone (Zhang et al., 2025b) as query Cognitive Embeddings. This model adapts the decoder-style Qwen3 backbone for text embedding. We run all $L = 28$ Transformer layers in a forward pass without generating a routing decision or answer. Let $h_{q,T_q}^{(L)} \in \mathbb{R}^D$ be its final-layer state at the last non-padding token, with $D = 1024$. The Cognitive Embedding and its routing projection are

$$x_q = \mathrm{Norm}\left(h_{q,T_q}^{(L)}\right), \qquad z_q = W_q x_q \in \mathbb{R}^H, \tag{2}$$

where $W_q \in \mathbb{R}^{H\times D}$ is learned, $H = 128$, and $\mathrm{Norm}(v) = v/\max(\|v\|_2, 10^{-12})$. Thus x_q is the Cognitive Embedding obtained from the normalized backbone state, and z_q is its learned routing projection. Routing supervision trains the projection. The encoder stays fixed. The resulting z_q drives both candidate fusion and matching. Appendix C.2 specifies the encoder settings.

4.3 Model Cognitive Embedding Generation

Each candidate has a trainable identity Cognitive Embedding $p_m^{\mathrm{id}} \in \mathbb{R}^H$ that captures performance patterns from routing feedback, and a description t_m providing semantic capability information. Encoding t_m with the frozen encoder and the same pooling rule yields a description Cognitive Embedding $d_m \in \mathbb{R}^D$. A learned projection produces $c_m = \mathrm{Norm}(W_c d_m) \in \mathbb{R}^H$.

Query-gated fusion. A scalar gate controls the contribution of description information for each query.

$$g(q) = \sigma(w_g^\top z_q + b_g), \tag{3}$$

$$p_m(q) = \mathrm{Norm}\left(\mathrm{Norm}\left(p_m^{\mathrm{id}}\right) + \lambda g(q) c_m\right), \tag{4}$$

where σ is the logistic sigmoid and $\lambda = 1$. The gate is shared across candidates. A larger value increases the description contribution. The figure expresses the relative weights as α and $1-\alpha$, corresponding to $\alpha_{\mathrm{fuse}}(q) = 1/(1+\lambda g(q))$. This notation describes the contribution ratio before normalization, while Equation (4) defines the implemented fusion. The descriptions and their

projections can be cached. Query gating makes the fused model Cognitive Embeddings request-dependent.

4.4 Model Routing

The matching head scores each query–model pair through an elementwise interaction.

$$s(q, m) = w_s^\top \big(z_q \odot p_m(q)\big), \tag{5}$$

where $w_s \in \mathbb{R}^H$ is learned and $\odot$ denotes elementwise multiplication. The score measures the compatibility learned from response-quality feedback. For quality-first routing, the selected model is

$$\widehat{m}(q) = \arg\max_{m \in \mathcal{M}} s(q, m). \tag{6}$$

Only the selected candidate is called to generate an answer. The quality–cost settings combine these scores with expected candidate costs as specified in Appendix F. With projected descriptions cached, the matching head requires $O(DH + MH)$ operations after query encoding, comprising $O(DH)$ for query projection and $O(MH)$ for candidate fusion and scoring.

4.5 Tie-aware Pairwise Training

Preserving all tied-best candidates. Several candidates may achieve the same best recorded quality on a query. Choosing only one of them as the training target discards other valid choices. We retain all tied-best candidates and separate them from strictly lower-quality candidates.

$$\mathcal{W}(q) = \{m : y_{qm} = \max_j y_{qj}\}, \qquad \mathcal{L}(q) = \{m : y_{qm} < \max_j y_{qj}\}. \tag{7}$$

For example, qualities $(1, 1, 0)$ produce the winner–loser pairs (m_1, m_3) and (m_2, m_3). Both best models supply positive supervision, and no preference is imposed between them. Here, a tie means exact equality of recorded quality, not equality of predicted matching scores. The lower-quality set may contain different quality values. Its members are compared with the winners, without requiring comparisons among themselves.

Pairwise ranking objective. For each pair $(u, v) \in \mathcal{W}(q) \times \mathcal{L}(q)$, we use the logistic ranking loss $\ell_q(u, v) = \log(1 + \exp[s(q, v) - s(q, u)])$ (Rendle et al., 2009). Averaging the comparisons within each query gives

$$\mathcal{J} = \sum_{q:\mathcal{L}(q)\neq\emptyset} \frac{1}{|\mathcal{W}(q)||\mathcal{L}(q)|} \sum_{u\in\mathcal{W}(q)} \sum_{v\in\mathcal{L}(q)} \log(1 + \exp\left[s(q, v) - s(q, u)\right]). \tag{8}$$

Each informative query has total weight one. All-tied queries contribute no pair. Minimizing the loss increases winner–loser margins without directly ordering tied winners.

Parameter updates. Training updates both projections, candidate identities, the query gate, and the scoring head. Minibatch losses are normalized by their pair-weight sums, and checkpoints are selected on a separate validation split. Appendix B provides gradients, the query-weighting analysis, and a loss-based bound on selecting a lower-quality candidate.

5 Experiments

We implemented CogRouter using the LLMRouter framework (Feng et al., 2026a).

5.1 Experimental Settings

Datasets. We evaluate CogRouter on two benchmarks, xRouteBench (Feng et al., 2026a) and RouterBench (Hu et al., 2024). From xRouteBench, we select five datasets, Generic, LoCoMo, Geometry3K, MathVista, and TimeSeries. These datasets share 18 candidate LLMs and contain 4,331 test queries, with 3,729, 314, 61, 100, and 127 queries, respectively. RouterBench contains 11 candidate LLMs and 7,299 test queries. We use its cleaned split with disjoint query texts between training and testing. The complete candidate lists are provided in Table 5 in Appendix C.1.

Baselines. We compare CogRouter with 11 baselines on xRouteBench and eight on RouterBench. Following the categories in Section 2, CausalLM is an LLM-based router (Feng et al., 2026a). KNNRouter, SVMRouter, and MLPRouter (Feng et al., 2026b), together with Hybrid-LLM (Ding et al., 2024), are feature-based routers. MFRouter (Ong et al., 2025), RouterDC (Chen et al., 2024b), and GraphRouter (Feng et al., 2025) are relation-based routers. EloRouter uses global pairwise ratings (Bradley & Terry, 1952), while Smallest-LLM and Largest-LLM are fixed-selection controls (Feng et al., 2026a). RouterBench includes the same baselines except Hybrid-LLM and the two fixed-selection controls. We retain each method's archived training configuration to accommodate differences in estimators, objectives, and optimization dynamics. All methods share evaluation queries, candidate pools, and metrics. Appendix C.1 records implementation provenance.

Evaluation Metrics. For xRouteBench, quality averages recorded $[0, 1]$ scores within a domain and then weights the five domain means equally. For RouterBench, all 7,299 test queries are weighted equally.

Implementation Details. Each xRouteBench domain has a separately trained head. The final router uses Cognitive Embeddings obtained from normalized last-token hidden states of the frozen Qwen3-Embedding-0.6B backbone (Zhang et al., 2025b) and selects checkpoints by validation pair loss on a frozen query-text split. Table 7 in Appendix C.2 lists the preprocessing, architecture, optimization, and checkpoint-selection settings. xRouteBench descriptions are stored model-attribute strings. RouterBench descriptions contain candidate identifiers, so their semantic content is more limited.

5.2 Routing Quality

To evaluate routing quality, we compare CogRouter with the baselines on the five-domain xRouteBench subset and full RouterBench, using the candidate pools and evaluation queries specified above. As shown in Table 1, CogRouter achieves 53.98% macro quality over multiple random seeds, with the highest displayed mean in each of the five domains.

As shown in Figure 3, CogRouter reaches 78.20% quality on full RouterBench over multiple random seeds. The matched identity-only head instead reaches 78.14%. Its different training protocol makes it a separate component control.

5.3 Supervision and Representation Controls

Supervision and representation controls. To isolate the effects of supervision and representation, we compare three configurations in the original five-domain experiment. All three configurations share the 1,024-dimensional features, candidate pool, frozen query-text split, initialization seed, optimizer, and validation-loss selection rule. Each run allows at most 20 epochs with patience three. The fixed-winner control retains one deterministic tied winner per query and compares it only with strict losers, preserving pair-row counts and weights. The tied-best control retains every tied winner. The third configuration additionally enables query-gated descriptions. Appendix A.1 specifies the controlled changes and training settings. As shown in Table 2, retaining tied-best targets improves mean quality in the five-domain supervision control.

Table 1. Routing quality on the five-domain xRouteBench subset (4,331 requests). Macro equally weights the five domains. Bold denotes column maxima.

Router	Generic	LoCoMo	Geo3K	MathVista	TimeSeries	Macro
Smallest-LLM	57.55%	25.44%	27.87%	35.00%	49.61%	39.09%
Largest-LLM	70.29%	26.59%	37.70%	33.00%	45.67%	42.65%
kNNRouter	60.66%	25.76%	26.23%	39.00%	52.76%	40.88%
SVMRouter	59.53%	25.44%	26.23%	35.00%	49.61%	39.16%
MLPRouter	61.34%	25.97%	29.51%	34.00%	51.18%	40.40%
CausalLM	59.41%	22.20%	22.95%	34.00%	39.37%	35.59%
EloRouter	57.55%	25.44%	27.87%	35.00%	49.61%	39.09%
Hybrid-LLM	65.06%	26.00%	32.79%	37.00%	53.54%	42.88%
RouterDC	76.03%	20.93%	34.43%	42.00%	63.78%	47.43%
GraphRouter	62.52%	25.40%	27.87%	35.00%	50.39%	40.24%
MFRouter	61.24%	25.08%	27.87%	39.33%	51.97%	41.10%
CogRouter	**78.21%**	**27.39%**	**45.36%**	**52.00%**	**66.93%**	**53.98%**

Table 2. Original standalone five-domain ablation, multiple random seeds. All rows use the same archived features and split, at most 20 epochs, patience three, and validation-loss checkpoint selection.

Configuration	Five-domain macro
Identity-only + fixed winner	53.03%
Identity-only + tied-best	53.72%
Tie + gated descriptions	53.88%

Table 3. Quality-first controls averaged over three runs. xRouteBench equally weights the five domains. RouterBench equally weights its test queries. No Tie denotes a query–model matching baseline without tied-best augmentation. "+ gate" adds model descriptions and their query gate.

Configuration	xRouteBench ↑	RouterBench ↑
No Tie (no gate)	41.10%	48.24%
No Tie + gate	42.05%	49.42%
CogRouter w/o descriptions	53.94%	78.14%
CogRouter	**53.98%**	**78.20%**

Query-gated descriptions. To assess the contribution of query-gated descriptions, we compare additional quality-first controls on both benchmarks. Here, *No Tie* denotes a query–model matching baseline trained without tied-best augmentation, rather than the fixed-winner control above. No Tie (no gate) uses learned candidate identities alone. No Tie + gate additionally uses model descriptions and their query-dependent gate. As shown in Table 3, adding descriptions and their query-dependent gate raises five-domain macro quality from 41.10% to 42.05% and RouterBench quality from 48.24% to 49.42%. The two CogRouter variants compare identity-only representations with gated descriptions under our training settings. Appendix A gives the configuration details.

5.4 Call Cost and Quality–Cost Trade-offs

To evaluate quality–cost trade-offs, we apply five inference preferences to archived checkpoints of CogRouter and the score-based baselines without retraining. Figure 9 displays four selected methods for readability, while Table 4 retains the full measured comparison.

Moderate cost penalties. To assess how a moderate cost penalty affects routing quality, we compare quality-only selection with $(\alpha, \beta) = (0.8, 0.2)$ on both benchmarks. As shown in Figure 9 and Table 11, moving from quality-only selection to $(\alpha, \beta) = (0.8, 0.2)$ cuts xRouteBench call cost by 49.62% for a 1.28-point quality loss. On RouterBench it saves 66.02% but loses 6.98 points. Thus a moderate cost weight preserves most quality on xRouteBench, whereas RouterBench requires a larger quality concession.

Stronger cost penalties. To examine the effect of increasing the cost penalty, we compare the RouterBench settings $\alpha = 0.6$ and $\alpha = 0.2$. As shown in Table 11, reducing α from 0.6 to 0.2 lowers cost only from $\$0.226$ to $\$0.212$ per 1,000 requests while sacrificing another 2.44 quality points. Within the measured sweep, much of the cost reduction is already obtained at $\alpha = 0.6$. Pushing the penalty further mainly gives up quality.

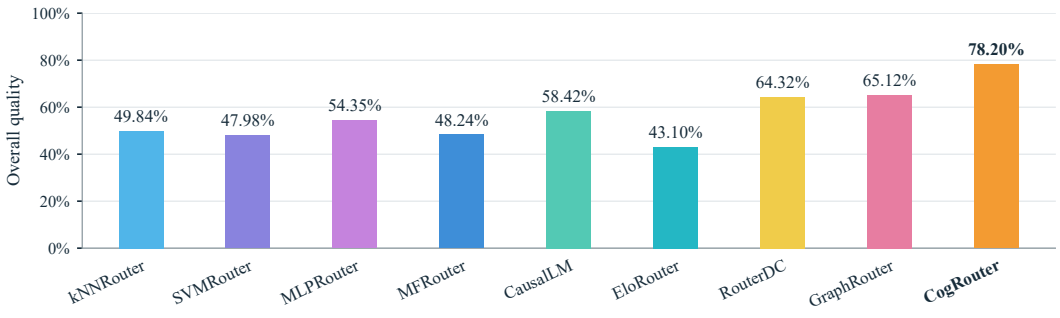


Figure 3. Overall routing quality on all 7,299 RouterBench test queries. Each query receives equal weight. CogRouter and MFRouter average repeated training runs.

Table 4. Quality–cost summaries with shared fixed references, using all five settings per swept method. Fixed policies and CausalLM retain one setting. Higher/lower values are better for ↑/↓. AIQ is quality. CQT uses USD/1K requests. † indicates that the fixed-reference envelope determines AIQ rather than equal router quality. Definitions and limitations are in Appendix F.2.

Five-domain xRouteBench Method	Raw mAPGR ↑	Clipped ↑	CEI ↑	AIQ ↑	CQT50 ↓	CQT80 ↓
Smallest-LLM	0.441	0.441	44.10	44.45%†	0.191	0.273
Largest-LLM	0.500	0.500	50.00	44.45%†	0.173	0.266
kNNRouter	0.437	0.437	43.69	44.45%†	0.192	0.274
SVMRouter	0.444	0.444	44.39	44.45%†	0.190	0.273
MLPRouter	0.420	0.420	42.03	44.45%†	0.197	0.276
CausalLM	0.368	0.368	36.83	44.45%†	0.214	0.282
EloRouter	0.441	0.441	44.10	44.45%†	0.191	0.273
Hybrid-LLM	0.331	0.331	33.12	44.45%†	0.234	0.291
RouterDC	0.398	0.398	39.81	44.52%	0.211	0.306
GraphRouter	0.475	0.475	47.49	44.45%†	0.180	0.269
MFRouter	0.418	0.418	41.84	44.45%†	0.198	0.276
CogRouter	**1.751**	**0.922**	**92.25**	**51.25%**	**0.044**	**0.057**
RouterBench Method	Raw mAPGR ↑	Clipped ↑	CEI ↑	AIQ ↑	CQT50 ↓	CQT80 ↓
kNNRouter	0.682	0.682	68.25	62.97%	0.693	2.237
SVMRouter	0.673	0.673	67.32	62.52%	0.784	2.274
MLPRouter	0.701	0.701	70.14	63.95%	0.520	2.168
MFRouter	0.673	0.673	67.33	62.53%	0.782	2.273
CausalLM	0.770	0.770	76.99	67.10%	0.173	1.782
EloRouter	0.627	0.627	62.72	60.35%	1.100	2.400
RouterDC	0.576	0.576	57.60	58.92%	1.348	2.581
GraphRouter	0.842	0.842	84.19	70.51%	**0.141**	1.008
CogRouter	**0.897**	**0.895**	**89.53**	**73.14%**	0.159	**0.379**

Average quality and low-budget efficiency. To compare average quality and low-budget efficiency, we examine AIQ and CQT50 for CogRouter and GraphRouter on RouterBench. As shown in Table 4, CogRouter has higher AIQ than GraphRouter (73.14% versus 70.51%), while GraphRouter has lower CQT50 ($0.141 versus $0.159 per 1,000 requests). Thus CogRouter offers higher average quality over the reference budget interval, but this does not imply superiority at every budget. These summaries now use all five preferences for both methods. The shared inference cost penalty retains each method's original training settings. CEI and CQT additionally depend on interpolation knots. Appendix F specifies sweep coverage, metric definitions, and CogRouter's expected-mixture summaries. Appendix C.5 reports offline preparation costs.

6 Limitations

Our router learns candidate-specific representations from recorded response feedback, so adding new candidate models requires updating these representations. Extending the method to support continual candidate updates and changing task distributions is a direction for future work.

7 Conclusion

CogRouter learns to match query Cognitive Embeddings with candidate capabilities, combining a frozen decoder-style Transformer backbone, learned candidate Cognitive Embeddings, and query-gated descriptions. Tied-best pairwise supervision retains all observed best candidates against strict losers, and a compact query-conditioned matching head uses this feedback to select an LLM before generation. The implementation performs well on the evaluated five-domain subset and RouterBench. Matched controls locate the clearest improvement in preserving tied winners over a fixed representative. The smaller gated-description effects depend on the benchmark.

AI USE STATEMENT

Generative AI tools assisted with method discussion, experiment design and implementation, formulation and checking of mathematical claims and proofs, interpretation of recorded results, literature discovery, manuscript drafting, LaTeX preparation, and revision. The writing workflow used OpenAI Codex and a separate model-assisted draft review. Reported numerical results were taken from experiment records and recalculated where the underlying tables were available. Citation metadata was checked against publication records. The authors retain responsibility for the scientific claims and the final submission.

ETHICS STATEMENT

This study evaluates routing decisions using previously recorded benchmark responses. Optimizing average quality and price does not establish fairness across user groups or safety for high-stakes requests. A deployment should assess these properties separately and respect the licenses and access conditions of the underlying models and data. The reported price comparisons reflect a benchmark snapshot, not a promise of current service pricing.

REPRODUCIBILITY STATEMENT

Sections 4 and 5 define the score, supervision, evaluation scope, and checkpoint-selection rule. Sections A.1, C and C.4 describe training settings, aggregation, and price metrics. The reproducibility materials provide configurations, run-level results, and scripts for the tables and figures. The five-domain and RouterBench results, including the standalone ablation, retain their source records and evaluation membership. Public data and implementation references are cited in the text.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. AutoMix: Automatically Mixing Language Models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 131000–131034. Curran Associates, Inc., 2024. doi: 10.52202/079017-4164. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ecda225cb187b40ea8edc1f46b03ffda-Paper-Conference.pdf.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. LLM2Vec: Large Language Models Are Secretly Powerful Text Encoders. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=IW1PR7vEBf.

Ralph Allan Bradley and Milton E. Terry. Rank Analysis of Incomplete Block Designs: The Method of Paired Comparisons. *Biometrika*, 39(3–4):324–345, 1952. doi: 10.1093/biomet/39.3-4.324. URL https://doi.org/10.1093/biomet/39.3-4.324.

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*, 2024a. URL https://openreview.net/forum?id=cSimKw5p6R.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. RouterDC: Query-Based Router by Dual Contrastive Learning for Assembling Large Language Models. In *Advances in Neural Information Processing Systems 37*, NeurIPS 2024, pp. 66305–66328. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024b. doi: 10.52202/079017-2120. URL http://dx.doi.org/10.52202/079017-2120.

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. *arXiv preprint arXiv:2404.14618*, 2024. doi: 10.48550/ARXIV.2404.14618. URL https://arxiv.org/abs/2404.14618.

Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks V. S. Lakshmanan, Qingyun Wu, and Victor Rühle. BEST-Route: Adaptive LLM Routing with Test-Time Optimal Compute. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 13870–13884. PMLR, 2025. URL https://proceedings.mlr.press/v267/ding25d.html.

Tao Feng, Yanzhen Shen, and Jiaxuan You. GraphRouter: A Graph-based Router for LLM Selections. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 26186–26203, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/41b6674c28a9b93ec8d22a53ca25bc3b-Paper-Conference.pdf.

Tao Feng, Fangxu Yu, Haozhen Zhang, Zhongjie Dai, Liangqi Yuan, Zijie Lei, Weizhi Zhang, Kunlun Zhu, Haodong Yue, Keyang Xuan, Ge Liu, and Jiaxuan You. LLMRouter: Unified Infrastructure for Developing, Evaluating, and Deploying LLM Routers. *arXiv preprint arXiv:2608.06867*, 2026a. doi: 10.48550/ARXIV.2608.06867. URL https://arxiv.org/abs/2608.06867.

Tao Feng, Haozhen Zhang, Zijie Lei, Pengrui Han, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Jiaxuan You. FusionFactory: Fusing LLM Capabilities with Multi-LLM Log Data. *Transactions on Machine Learning Research*, 2026b. URL https://arxiv.org/abs/2507.10540v3.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6894–6910. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.552. URL https://aclanthology.org/2021.emnlp-main.552/.

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. doi: 10.48550/ARXIV.2403.12031. URL https://arxiv.org/abs/2403.12031.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.792. URL https://aclanthology.org/2023.acl-long.792/.

Yueyue Liu, Hongyu Zhang, Yuantian Miao, Van-Hoang Le, and Zhiqiang Li. OptLLM: Optimal Assignment of Queries to Large Language Models. In *2024 IEEE International Conference on Web Services (ICWS)*, pp. 788–798, 2024. doi: 10.1109/ICWS62655.2024.00098. URL https://doi.org/10.1109/ICWS62655.2024.00098.

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. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1964–1974. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.109. URL https://aclanthology.org/2024.naacl-long.109/.

Yasmin Moslem and John D. Kelleher. Dynamic Model Routing and Cascading for Efficient LLM Inference: A Survey. *arXiv preprint arXiv:2603.04445*, 2026. URL https://arxiv.org/abs/2603.04445.

Quang H. Nguyen, Thinh Dao, Duy C. Hoang, Juliette Decugis, Saurav Manchanda, Nitesh V. Chawla, and Khoa D. Doan. MetaLLM: A High-performant and Cost-efficient Dynamic Framework for Wrapping LLMs. *arXiv preprint arXiv:2407.10834*, 2024. URL https://arxiv.org/abs/2407.10834.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Kadous, and Ion Stoica. RouteLLM: Learning to Route LLMs from Preference Data. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 34433–34448, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5503a7c69d48a2f86fc00b3dc09de686-Paper-Conference.pdf.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. LLMs Know More Than They Show: On the Intrinsic Representation of LLM Hallucinations. In *International Conference on Learning Representations*, volume 2025, pp. 66880–66913, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a712d461e57201efe35d429a6f1731c1-Paper-Conference.pdf.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 452–461. AUAI Press, 2009. URL https://arxiv.org/abs/1205.2618.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/77c33e6a367922d003ff102ffb92b658-Abstract-Conference.html.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large Language Model Routing with Benchmark Datasets. *arXiv preprint arXiv:2309.15789*, 2023. URL https://arxiv.org/abs/2309.15789.

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. *arXiv preprint arXiv:2502.03261*, 2025. doi: 10.48550/ARXIV.2502.03261. URL https://arxiv.org/abs/2502.03261.

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 *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15629–15644. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.761. URL https://aclanthology.org/2025.acl-long.761/.

Clovis Varangot-Reille, Christophe Bouvard, Antoine Gourru, Mathieu Ciancone, Marion Schaeffer, and François Jacquenet. Doing More with Less: A Survey on Routing Strategies for Resource Optimisation in Large Language Model-Based Systems. *arXiv preprint arXiv:2502.00409*, 2025. URL https://arxiv.org/abs/2502.00409.

Clovis Varangot-Reille, Christophe Bouvard, and Antoine Gourru. Generalising LLM Routing using Past Performance Retrieval: A Few-Shot Router is Sufficient. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pp. 304–319. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.eacl-srw.22. URL https://aclanthology.org/2026.eacl-srw.22/.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving Text Embeddings with Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11897–11916. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.642. URL https://aclanthology.org/2024.acl-long.642/.

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 *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10912–10922. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.545. URL https://aclanthology.org/2025.naacl-long.545/.

Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-R1: Teaching LLMs Multi-Round Routing and Aggregation via Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 141233–141265. Curran Associates, Inc., 2025a. doi: 10.52202/085713-4719. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/ceaa137fce916aba5c65fceb1309088b-Paper-Conference.pdf.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *arXiv preprint arXiv:2506.05176*, 2025b. doi: 10.48550/ARXIV.2506.05176. URL https://arxiv.org/abs/2506.05176.

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*, volume 2025, pp. 76913–76926, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/bf5e4b85d203481d6e37bd32d9600162-Paper-Conference.pdf.

APPENDIX CONTENTS

A Ablation settings 15
B Mathematical analysis and proofs 15
C Experimental setup and evaluation metrics 19
D Task-domain prestudy details 23
E Supplementary routing-quality results 26
F Quality–cost trade-offs 28

A Ablation settings

The final model combines tied-best supervision with query-gated descriptions. This appendix specifies the training protocols for the five-domain xRouteBench and RouterBench controls.

A.1 Original five-domain ablation configuration

The three configurations in Table 2 are evaluated on Generic, LoCoMo, Geometry3K, MathVista, and TimeSeries, with 18 candidates and 4,331 test requests. A separate head is trained for each domain. We average domain quality with equal weights and then average across repeated training runs.

All configurations share the same frozen features, candidate pool, query-text split, and initialization within each paired run. They use 1,024-dimensional inputs, a 128-dimensional matching space, Adam with learning rate 10^{-3}, batch size 64, and no embedding noise. Validation uses 10.00% of the training queries and selects the lowest pair-loss checkpoint, with at most 20 epochs, patience three, and minimum improvement 10^{-4}.

The fixed-winner control selects one tied-best candidate per query and keeps that choice fixed throughout training. It compares this representative only with strict losers, preserving the number and weights of training pairs. Other tied winners are not treated as negatives. The tied-best control retains all winner–strict-loser comparisons. Both controls use learned candidate identities. The third configuration adds descriptions and their query-dependent gate.

A.2 RouterBench No Tie baseline and representation control

The No Tie (no gate) configuration in Table 3 uses a query–model matching baseline with tied-best augmentation, candidate descriptions, and query gating disabled. The two CogRouter variants share the Qwen3 features, query-text split, candidate pool, optimization settings, and validation rule.

No Tie + gate enables the frozen candidate-description input and its query-dependent gate while retaining the No Tie loss, data, hyperparameters, and validation procedure. The description weight is one. This control tests the joint description/gate module, not gating alone.

All configurations evaluate the same 7,299 requests under quality-first inference, and Table 3 reports their mean quality.

B Mathematical analysis and proofs

We analyze the score and tied-best objective from Section 4. All selection statements use a finite candidate pool and a fixed tie-breaking rule. The results establish properties of the stated objective and score.

B.1 Query weighting, sampling, and fixed-winner targets

Let $\mathcal{J}_q$ denote the contribution of query q to Equation (8). For an informative query, write $K_q = |\mathcal{W}(q)| \geq 1$, $L_q = |\mathcal{L}(q)| \geq 1$, and $\omega_q = (K_q L_q)^{-1}$. Expanding its loss gives

$$\mathcal{J}_q = \frac{1}{K_q} \sum_{u \in \mathcal{W}(q)} \underbrace{\frac{1}{L_q} \sum_{v \in \mathcal{L}(q)} \ell_q(u, v)}_{F_q(u)} . \tag{9}$$

Here $F_q(u)$ averages the losses against strict losers. The pair weights sum to $K_q L_q \omega_q = 1$.

Proposition 1 (Query-balanced tied-best objective)**.** *Fix an informative query with finite scores. If U and V are independent and uniform on $\mathcal{W}(q)$ and $\mathcal{L}(q)$, then*

$$\mathbb{E}_{U,V}[\ell_q(U, V)] = \mathcal{J}_q. \tag{10}$$

At parameter values where the pair scores are differentiable, the same equality holds for their parameter gradients. Each informative query has total weight one, irrespective of its number of ties.

Proof of Proposition 1. Independence and uniformity give $\Pr(U = u, V = v) = 1/(K_q L_q)$ for each pair. Summing the finite expectation yields

$$\mathbb{E}[\ell_q(U, V)] = \sum_{u \in \mathcal{W}(q)} \sum_{v \in \mathcal{L}(q)} \frac{\ell_q(u, v)}{K_q L_q} = \mathcal{J}_q.$$

The outcome-defined sets and probabilities do not depend on the model parameters. At a point where every pair score is differentiable, differentiation commutes with this finite sum, proving the gradient identity. No interchange with an infinite expectation is needed. If $L_q = 0$, the query has no pairs and is excluded from the informative-query objective. □

A permanently chosen winner u_0 instead has objective $F_q(u_0)$, whose difference from the tied-best loss is exactly

$$F_q(u_0) - \mathcal{J}_q = \frac{1}{K_q} \sum_{u \in \mathcal{W}(q)} [F_q(u_0) - F_q(u)]. \tag{11}$$

The sign is not fixed. For two tied winners with scores $s_1 = 0, s_2 = -1$ and a loser with $s_3 = 0$, choosing winner 1 gives $\log 2$, whereas the tied-best loss is $[\log 2 + \log(1 + e)]/2$. Swapping the winners' scores reverses the sign. With one winner the objectives coincide. More generally, the tied-best loss is invariant to relabeling winners and requires no preference between them.

For fixed query and parameters, the finite random variable $X = \ell_q(U, V)$ also satisfies

$$\mathrm{Var}(X) = \mathbb{E}_V[\mathrm{Var}(X \mid V)] + \mathrm{Var}_V\left(\frac{1}{K_q} \sum_{u \in \mathcal{W}(q)} \ell_q(u, V)\right). \tag{12}$$

To verify this identity, expand $X - \mathbb{E}X = (X - \mathbb{E}[X \mid V]) + (\mathbb{E}[X \mid V] - \mathbb{E}X)$. The cross term has expectation zero because the first term has conditional mean zero. Thus averaging all winners before sampling a loser removes the first, nonnegative variance term. Averaging both sets evaluates $\mathcal{J}_q$ exactly at the fixed parameters.

B.2 Observed-selection and quality-gap bounds

Theorem 1 (Observed-selection bound)**.** *Fix a query with $1 \le M < \infty$, recorded qualities $y_{qm} \in [0, 1]$, and finite scores. Let $\widehat{m} \in \arg\max_m s(q, m)$ use any fixed tie-breaking rule. If $L_q = |\mathcal{L}(q)| > 0$, then*

$$\mathbf{1}\{\widehat{m} \notin \mathcal{W}(q)\} \le \min\left\{1, \frac{L_q \mathcal{J}_q}{\log 2}\right\}. \tag{13}$$

The recorded-quality gap $\max_m y_{qm} - y_{q\widehat{m}}$ is bounded by the same right-hand side. When $L_q = 0$, both quantities are zero.

Proof of Theorem 1. If $\widehat{m} \in \mathcal{W}(q)$, the indicator and recorded-quality gap are both zero and the bound holds. Otherwise, $\widehat{m} \in \mathcal{L}(q)$ and maximizing the score implies $s(q, \widehat{m}) \ge s(q, u)$ for every $u \in \mathcal{W}(q)$. Consequently,

$$\mathcal{J}_q \ge \frac{1}{K_q L_q} \sum_{u \in \mathcal{W}(q)} \ell_q(u, \widehat{m}) \tag{14}$$

$$\ge \frac{1}{K_q L_q} \sum_{u \in \mathcal{W}(q)} \log 2 = \frac{\log 2}{L_q}. \tag{15}$$

The first inequality drops nonnegative terms, and the second uses monotonicity of $t \mapsto \log(1 + e^t)$ at nonnegative score differences. Thus a selected loser implies $L_q \mathcal{J}_q / \log 2 \ge 1$, proving the indicator bound, including score ties resolved toward a loser. Since qualities lie in $[0, 1]$, their gap is at most this indicator. If $L_q = 0$, every candidate is an observed winner, so both quantities are zero, including the case $M = 1$. □

For any finite evaluation set $\mathcal{Q}$ of size $N > 0$, set $\mathcal{J}_q = 0$ on all-tied queries and let $R_q = \max_m y_{qm} - \min_m y_{qm} \le 1$. The proof also gives

$$\frac{1}{N}\sum_{q\in\mathcal{Q}} \mathbf{1}\{\widehat{m}(q) \notin \mathcal{W}(q)\} \le \frac{1}{N}\sum_{q:L_q>0} \min\left\{1, \frac{L_q\mathcal{J}_q}{\log 2}\right\}, \tag{16}$$

$$\frac{1}{N}\sum_{q\in\mathcal{Q}} \left[\max_m y_{qm} - y_{q\widehat{m}(q)}\right] \le \frac{1}{N}\sum_{q:L_q>0} R_q \min\left\{1, \frac{L_q\mathcal{J}_q}{\log 2}\right\}. \tag{17}$$

Every term on the right is evaluated on the same query set as the left. A small training loss alone therefore does not supply a test-set bound. The factor L_q also explains why the query-normalized loss must be sufficiently small to certify a correct selection when many strict losers are present.

B.3 Gradients of the pair loss and minibatch objective

For a winner–loser margin $\Delta = s(q,u) - s(q,v)$,

$$\ell(\Delta) = \log(1+e^{-\Delta}), \qquad \ell'(\Delta) = -\sigma(-\Delta), \qquad \ell''(\Delta) = \sigma(\Delta)\sigma(-\Delta) \in (0, 1/4]. \tag{18}$$

$$\nabla_\theta \ell_q(u,v) = -\sigma(-\Delta)\big[\nabla_\theta s(q,u) - \nabla_\theta s(q,v)\big]. \tag{19}$$

The first derivative follows from $-e^{-\Delta}/(1+e^{-\Delta})$. The second follows by differentiating the sigmoid. The chain rule yields Equation (19). This is convexity in a scalar margin, not a claim that the parameterized training problem is convex.

For a minibatch $\mathcal{B}$ of stored pairs, the implemented objective and gradient are

$$\mathcal{J}_\mathcal{B} = \frac{\sum_{(q,u,v)\in\mathcal{B}} \omega_q \ell_q(u,v)}{\sum_{(q,u,v)\in\mathcal{B}} \omega_q}, \qquad \nabla_\theta \mathcal{J}_\mathcal{B} = \frac{\sum_{(q,u,v)\in\mathcal{B}} \omega_q \nabla_\theta \ell_q(u,v)}{\sum_{(q,u,v)\in\mathcal{B}} \omega_q}. \tag{20}$$

The denominator is positive for a nonempty pair batch and independent of θ. Over all stored pairs it equals the number of informative queries.

B.4 Frozen-score directional derivatives

Fix finite score vectors $s^0(q,\cdot)$ and $s^1(q,\cdot)$ before and after an intervention on the same trained head. For each pair with unequal recorded quality, define its preferred direction $t_{qij} = \operatorname{sign}(y_{qi} - y_{qj}) \in \{-1, 1\}$, initial margin $\Delta^0_{qij} = s^0(q,i) - s^0(q,j)$, and relative score change

$$c_{qij} = [s^1(q,i) - s^0(q,i)] - [s^1(q,j) - s^0(q,j)]. \tag{21}$$

Along the affine score path indexed by $\tau \in [0,1]$, the pair loss and its derivative are

$$\ell_{qij}(\tau) = \log\big(1 + \exp[-t_{qij}(\Delta^0_{qij} + \tau c_{qij})]\big), \tag{22}$$

$$\ell'_{qij}(\tau) = -t_{qij}c_{qij}\,\sigma[-t_{qij}(\Delta^0_{qij} + \tau c_{qij})], \tag{23}$$

$$\ell'_{qij}(0) = -t_{qij}c_{qij}\,\sigma(-t_{qij}\Delta^0_{qij}). \tag{24}$$

The chain rule proves the derivative. For finite scores the sigmoid is strictly positive, so its sign is determined by $-t_{qij}c_{qij}$. A positive preferred-margin change therefore decreases that pair's loss along the entire path. A zero change leaves it constant. For tied-best training pairs, orient i as a winner and j as a strict loser, giving $t_{qij} = 1$. An all-strict-pair diagnostic uses a different pair population and must specify its weights separately.

For any finite pair set $\mathcal{P}$ and fixed weights $\nu_{qij} \ge 0$ summing to one, let $\mathcal{A}(\tau) = \sum_\mathcal{P} \nu_{qij}\ell_{qij}(\tau)$. Finite summation and the fundamental theorem of calculus give

$$\mathcal{A}'(\tau) = \sum_\mathcal{P} \nu_{qij}\ell'_{qij}(\tau), \qquad \mathcal{A}(1) - \mathcal{A}(0) = \int_0^1 \mathcal{A}'(\tau)\,d\tau. \tag{25}$$

Thus a negative initial aggregate derivative certifies local decrease but need not determine the endpoint loss change when pairs move in opposite directions. The parameter τ interpolates scores. It is not

the description coefficient or gate, whose effects include normalization. These identities specify frozen-score diagnostics rather than measured training trajectories.

For a fixed-gate to learned-gate score transition, a normalized diagnostic on a nonempty finite query-level strict-pair set $\mathcal{P}_q$ is

$$U_q = \frac{\sum_{(i,j)\in\mathcal{P}_q} \sigma(-t_{qij}\Delta^0_{qij})\, t_{qij} c_{qij}}{\sum_{(i,j)\in\mathcal{P}_q} \sigma(-t_{qij}\Delta^0_{qij})}. \tag{26}$$

The denominator is positive for finite margins. If $\mathcal{A}_q$ averages these pair losses equally, then $\mathcal{A}'_q(0) = -U_q \sum_{\mathcal{P}_q} \sigma(-t_{qij}\Delta^0_{qij})/|\mathcal{P}_q|$. Thus positive U_q indicates local loss reduction along that score transition. This is a finite-transition utility statistic, distinct from differentiation with respect to the gate itself. Queries with no strict pair have no such statistic.

B.5 Normalized description gate and its sensitivity

Fix a query and matching parameters. Let $\epsilon = 10^{-12}$, $a = w_s \odot z_q$, $\bar{p}_m = \mathrm{Norm}(p^{\mathrm{id}}_m)$, and

$$u_m(g) = \bar{p}_m + \lambda g c_m, \qquad p_m(g) = \frac{u_m(g)}{\max\{\|u_m(g)\|_2, \epsilon\}}, \qquad s_m(g) = a^\top p_m(g).$$

Writing $a_q = a$, when $\|u_m(g)\|_2 > \epsilon$ the score sensitivity is

$$\frac{\partial s(q,m)}{\partial g} = \frac{\lambda}{\|u_m\|_2} a_q^\top \left(I - p_m p_m^\top\right) c_m. \tag{27}$$

The normalization ensures $\|p_m(g)\|_2 \le 1$, so Cauchy–Schwarz gives $|s_m(g)| \le \|a\|_2$. When $r_m(g) = \|u_m(g)\|_2 > \epsilon$, differentiating r_m gives $r'_m = \lambda p_m^\top c_m$. Applying the quotient rule then gives

$$\frac{dp_m}{dg} = \frac{\lambda c_m}{r_m} - \frac{u_m \lambda p_m^\top c_m}{r_m^2} = \frac{\lambda}{r_m}(I - p_m p_m^\top) c_m, \tag{28}$$

$$\frac{ds_m}{dg} = \frac{\lambda}{r_m} a^\top (I - p_m p_m^\top) c_m. \tag{29}$$

This proves Equation (27). For $r_m < \epsilon$, normalization is linear and $dp_m/dg = \lambda c_m/\epsilon$. At $r_m = \epsilon$ the normalization map is generally not differentiable, so neither interior formula asserts a derivative there. For gate logit $h = w_g^\top z_q + b_g$, $dg/dh = g(1-g)$ and

$$\frac{\partial \ell_q(u,v)}{\partial h} = -\sigma(-\Delta)\left(\frac{ds_u}{dg} - \frac{ds_v}{dg}\right) g(1-g), \tag{30}$$

where this partial derivative holds the effective query direction and other parameters fixed. Full parameter derivatives additionally include their other paths into the scores.

B.6 Routing stability under score and gate changes

Proposition 2 (Routing stability under relative score changes). *Fix a query, $M \ge 2$ finite scores s^0_m, and a unique winner m_*. Let $\gamma_j = s^0_{m_*} - s^0_j > 0$ for $j \ne m_*$ and let $\widehat{s}_m = s^0_m + \delta_m$. Selecting $j \ne m_*$ requires $\delta_j - \delta_{m_*} \ge \gamma_j$. In particular,*

$$\min_{j\ne m_*} \gamma_j > \max_m \delta_m - \min_m \delta_m \quad \Longrightarrow \quad \arg\max_m \widehat{s}_m = m_*. \tag{31}$$

Proof. Selecting j implies $s^0_j + \delta_j \ge s^0_{m_*} + \delta_{m_*}$, which rearranges to the necessary condition. Let $\gamma = \min_{j\ne m_*}\gamma_j$ and $S = \max_m \delta_m - \min_m \delta_m$. For every competitor, $\widehat{s}_{m_*} - \widehat{s}_j = \gamma_j - (\delta_j - \delta_{m_*}) \ge \gamma - S$. Strict positivity preserves the unique winner. Equality may permit a tie, so the sufficient condition is strict. A common additive score change cancels, and $M = 1$ always retains its sole candidate without a margin condition. □

Proposition 3 (Gate perturbations and decision margins)**.** *Fix the query and matching parameters, with $\lambda \geq 0$ and $\|c_m\|_2 \leq 1$. Let $\epsilon = 10^{-12}$ be the normalization floor and $u_m(t) = \bar{p}_m + \lambda t c_m$, where t is a scalar gate value. Suppose $\|u_m(t)\|_2 \geq \rho > \epsilon$ for every candidate and every t between gate values g and g'. Then*

$$|s_m(g') - s_m(g)| \leq \frac{\lambda \|a\|_2}{\rho} |g' - g|. \tag{32}$$

If $M \geq 2$ and the winner at g is unique with smallest margin γ, it remains selected at g' whenever $\gamma > 2\lambda\|a\|_2|g' - g|/\rho$.

Proof. On the stated interval p_m is a unit vector and $I - p_m p_m^\top$ is an orthogonal projection with operator norm at most one. Equation 29 and Cauchy–Schwarz therefore give $|ds_m/dt| \leq \lambda\|a\|_2\|c_m\|_2/\rho \leq \lambda\|a\|_2/\rho$. Integrating this derivative over the finite interval proves Equation (32). Each score change has absolute value at most $B = \lambda\|a\|_2|g' - g|/\rho$, so its range is at most $2B$. Proposition 2 then proves the decision claim. For $g' = g$ or $\lambda = 0$ the bound is zero and the scores are unchanged. The positive norm bound is a condition on the whole gate interval, not just its endpoints. □

These results distinguish changes in representations from changes in decisions because the gate can move candidate directions continuously while an existing score margin preserves the selected candidate.

B.7 Quality gains, harms, and regret

For a finite candidate pool, let $u_m(q) = \mathbb{E}[Y_m \mid Q = q] \in [0, 1]$ be measurable conditional qualities. Fix two measurable routing policies π_0 and π_1, including their tie-breaking rules, and write $[x]_+ = \max(x, 0)$. Define

$$G(q) = [u_{\pi_1(q)}(q) - u_{\pi_0(q)}(q)]_+, \qquad H(q) = [u_{\pi_0(q)}(q) - u_{\pi_1(q)}(q)]_+. \tag{33}$$

Since $x = [x]_+ - [-x]_+$ for every real x, the signed quality change satisfies

$$u_{\pi_1(q)}(q) - u_{\pi_0(q)}(q) = G(q) - H(q) \tag{34}$$

$$= \sum_{j \neq \pi_0(q)} [u_j(q) - u_{\pi_0(q)}(q)] \mathbf{1}\{\pi_1(q) = j\}. \tag{35}$$

The sum has one active term when the selected model changes, and zero otherwise. Let $V(\pi) = \mathbb{E}_Q[u_{\pi(Q)}(Q)]$ and $R(\pi) = \mathbb{E}_Q[\max_m u_m(Q) - u_{\pi(Q)}(Q)]$. All terms are bounded and integrable. Taking expectations and canceling the shared oracle term yields

$$V(\pi_1) - V(\pi_0) = \mathbb{E}_Q[G - H], \qquad R(\pi_1) - R(\pi_0) = V(\pi_0) - V(\pi_1). \tag{36}$$

The reference router need not be optimal, and equal-quality switches contribute neither gain nor harm. The same algebra holds for recorded qualities y_{qm} and a finite evaluation average, without identifying those records with conditional expectations.

For observed-best membership, put $b_k(q) = \mathbf{1}\{\pi_k(q) \in \mathcal{W}(q)\}$ for $k \in \{0, 1\}$. Pointwise,

$$b_1 - b_0 = \mathbf{1}\{b_0 = 0, b_1 = 1\} - \mathbf{1}\{b_0 = 1, b_1 = 0\}. \tag{37}$$

Consequently, on the same evaluation queries, correction rate minus harm rate equals the change in best-set hit rate. This hit-rate difference and the quality change measure distinct properties because the latter also accounts for the magnitude of improvements or losses between candidates.

C Experimental setup and evaluation metrics

Evaluation scope and settings follow. Appendix A reports supplementary quality results.

C.1 CANDIDATE LLMS AND BASELINE IMPLEMENTATIONS

Table 5 lists the candidate LLM pools used in the experiments. These answer generators are distinct from the router algorithms and the frozen Qwen3-Embedding-0.6B encoder. The xRouteBench inventory is reproduced from the candidate inventory records. The five evaluated domains use the same 18 candidates. RouterBench's model-size fields are unspecified. Monetary comparisons use realized per-response costs, not the adapter's placeholder input/output coefficients as token prices.

Table 5. Candidate LLM pools. The five xRouteBench datasets share an 18-candidate pool. RouterBench uses 11 candidates. Identifiers retain the evaluated model versions.

xRouteBench with 18 candidates		
cogito-v2-1-671b	llama-3.3-70b-instruct-turbo	mixtral-8x7b-instruct-v0.1
deepseek-v3.1	llama-4-maverick	qwen2.5-7b-instruct
gemma-2-9b-it	llama3-70b-instruct	qwen2.5-7b-instruct-turbo
gpt-oss-120b	mistral-7b-instruct-v0.3	qwen3-coder-next
gpt-oss-20b	mistral-small-3-24b-instruct	qwen3-next-80b-a3b-instruct
llama-3-8b-instruct-lite	mixtral-8x22b-instruct-v0.1	rnj-1-instruct
RouterBench with 11 candidates		
WizardLM/WizardLM-13B-V1.2	gpt-3.5-turbo-1106	mistralai/mistral-7b-chat
claude-instant-v1	gpt-4-1106-preview	mistralai/mixtral-8x7b-chat
claude-v1	meta/code-llama-instruct-34b-chat	zero-one-ai/Yi-34B-Chat
claude-v2	meta/llama-2-70b-chat	

For xRouteBench, each model description is a fixed natural-language record containing the model name, provider or family, stated scale or architecture when available, a short capability description, and the recorded input/output prices. The same 18 strings are reused across the five evaluated domains. For RouterBench, the source pool supplies only model identifiers, so the 11 identifier strings themselves are encoded. No response text, validation quality, or test outcome is included in a description. The candidate descriptions are fixed before training.

Table 6 distinguishes routing-method precedents from the implementations evaluated here. All external baselines use their method-specific LLMRouter modules and released configuration templates (Feng et al., 2026a), rather than a shared generic configuration. Because these methods use different estimators, objectives, and optimization dynamics, a common inner recipe would change their prescribed operation rather than merely standardize evaluation. We therefore retain each router's representation, objective, optimizer, validation or other model-selection procedure specified by its implementation, and stopping behavior. Adapting a configuration changes only its data and candidate-pool references and mechanically required dimensions. Configurations are fixed before test evaluation, and no baseline hyperparameter is selected from test quality. Routers share the outer queries, candidate pool, and metrics.

LLMRouter's kNN/SVM/MLP documentation points to FusionFactory's query-level routing comparisons. An earlier nearest-neighbor routing formulation learns per-model correctness predictors (Shnitzer et al., 2023). Our evaluated kNN module instead predicts a candidate-model label. RouteLLM's similarity-weighted Bradley–Terry router recomputes query-dependent ratings, whereas the evaluated EloRouter fits global ratings once. RouteLLM's causal classifier predicts comparison labels. LLMRouter's CausalLM predicts candidate names. EloRouter is not RouteLLM's similarity-weighted router, and CausalLM does not reproduce its original label space or backbone.

C.2 TRAINING CONFIGURATION

Frozen feature extraction. We load Qwen3-Embedding-0.6B with `AutoModel` and read its `last_hidden_state` output after the complete 28-layer forward pass. For each query or candidate description, we select the last token with a nonzero attention mask and apply ℓ_2 normalization to its 1,024-dimensional state. Inputs use left padding and truncation to 512 tokens. The backbone remains frozen, and its pooled query states are cached as Cognitive Embeddings for router training. No autoregressive decoding or language-model output head is used. Learned projections map these states to the 128-dimensional matching space.

Table 6. Provenance of every evaluated router. External methods use their released method-specific LLMRouter configurations. CogRouter is introduced in this work.

Router	Routing precedent or source	Evaluated implementation
kNNRouter	Shnitzer et al. (2023), FusionFactory (Feng et al., 2026b)	Nearest-neighbor classification of Semantic Embeddings into model labels.
SVMRouter	FusionFactory (Feng et al., 2026b)	Support-vector classification of Semantic Embeddings into model labels.
MLPRouter	FusionFactory (Feng et al., 2026b)	Feed-forward neural classifier over candidate-model labels.
MFRouter	RouteLLM (Ong et al., 2025)	LLMRouter's matrix-factorization router with its method-specific training configuration.
EloRouter	Bradley–Terry (Bradley & Terry, 1952), RouteLLM ranking precedent (Ong et al., 2025)	Global pairwise logistic ratings with an Elo scale. The highest-rated candidate is selected for every query.
Hybrid-LLM	Ding et al. (2024)	LLMRouter's quality-aware small/large-model selector.
RouterDC	Chen et al. (2024b)	Dual-contrastive query–model and query–query learning.
GraphRouter	Feng et al. (2025)	Heterogeneous task/query/model graph scoring.
CausalLM	RouteLLM (Ong et al., 2025)	A causal LM generates a model name from the query and candidate list.
Smallest-LLM, Largest-LLM	Fixed-selection controls in GraphRouter (Feng et al., 2025)	Query-independent choices using the framework's candidate-size metadata.
CogRouter	This work	Learned candidate identities and query-gated descriptions.

Table 7. Hyperparameters and preprocessing for the final router on the five-domain xRouteBench subset and RouterBench. Each xRouteBench domain has a separately trained head. The original standalone ablation uses its own cached features and the configuration in Appendix A.1. The fixed-epoch target controls have a separate training budget.

Setting	Value
Frozen representations	
Encoder	Qwen3-Embedding-0.6B
Encoder updates	None (frozen)
Query / description dimension	1024
Maximum encoder input length	512 tokens
Padding / pooling	Left / last non-padding token
Feature precision / normalization	Float32 / unit Euclidean norm
Matching head and supervision	
Matching dimension	128
Candidate count (xRouteBench / RouterBench)	18 / 11
Candidate representation	Learned identity + gated description
Description coefficient	1
Supervision	Tied-best versus strict-loser pairs
Pair loss	Weighted binary logistic loss
Optimization and checkpoint selection	
Optimizer	Adam
Learning rate	10^{-3}
Batch size	64 preference pairs
Maximum training budget	20 epochs
Validation fraction	10.00%
Validation split	Frozen query-text groups
Split seed	42
Checkpoint criterion	Minimum validation pair loss
Early stopping patience	3 epochs
Minimum validation-loss improvement	10^{-4}
Training repetitions	Multiple random seeds

An epoch is one traversal of the stored pair list. Training shuffles that list with a generator seeded by the run identifier. Within a batch, binary logistic losses are multiplied by the stored pair weights and divided by their sum.

Response rows and pair weights. The candidate-set notation in Section 4 assumes one quality per query–candidate pair. At record level, let $\mathcal{R}_q$ be the response rows of a grouped request, and let $\mathcal{R}_q^+$ and $\mathcal{R}_q^-$ contain all maximum-quality rows and all strictly lower-quality rows, respectively. The implemented pair list retains $\mathcal{R}_q^+ \times \mathcal{R}_q^-$ with weight $1/(|\mathcal{R}_q^+||\mathcal{R}_q^-|)$ per row pair. Repeated rows retain their multiplicity, and each informative group still has total weight one. Exact recorded equality defines ties. The quality-first target uses no price-based tie break. Query-text groups stay within a single fitting or validation partition.

Cached computation and candidate requirements. For the head in Section 4, the trainable quantities are the two $H \times D$ projections, M identity vectors of dimension H, two vectors of dimension H, and the scalar gate bias. Their storage is $O(DH + MH)$. With parameters fixed, projecting all descriptions once takes $O(MDH)$ operations and stores $O(MH)$ projected features. Each subsequent encoded query needs $O(DH)$ query projection work and $O(MH)$ work to combine, normalize, and score candidates.

A new candidate needs an identity representation as well as a description. Setting its identity to zero does not reproduce the trained representation. If $\lambda g > 0$ and $\lambda g \|c_m\|_2 > \epsilon$, then

$$\mathrm{Norm}(\lambda g c_m) = c_m / \|c_m\|_2,$$

so the gate's positive scale cancels. This equality applies above the normalization floor. Zero descriptions and smaller norms use the floored formula. Candidate admission therefore calls for updating the learned representation rather than assuming that description encoding alone supplies an equivalent learned candidate representation.

C.3 Benchmark protocols and quality aggregation

For evaluation query q_i in domain d, let $y_{di,m}$ be the recorded task score of candidate m, after the archive's response aggregation, and let $\pi_d(q_i)$ be the router's choice. We retain the task's recorded $[0, 1]$ score. It need not be binary accuracy. Let N_{dom} be the number of domains and n_d the number of query positions in domain d. Then

$$Q_d = \frac{1}{n_d} \sum_{i=1}^{n_d} y_{di,\pi_d(q_i)}, \qquad Q_{\mathrm{macro}} = \frac{1}{N_{\mathrm{dom}}} \sum_{d=1}^{N_{\mathrm{dom}}} Q_d, \qquad C_{\mathrm{query}} = \frac{\sum_d \sum_i c(q_i, \pi_d(q_i))}{\sum_d n_d}. \tag{38}$$

The five-domain evaluation uses $N_{\mathrm{dom}} = 5$. The symbol D remains reserved for the embedding dimension. Full RouterBench averages all 7,299 queries directly. These follow the respective benchmark score records (Feng et al., 2026a, Hu et al., 2024). The task-specific training and aggregation conventions are specified here. A difference of ΔQ is $100\Delta Q$ percentage points, not a relative percentage increase. Monetary cost per 1,000 requests is $1000 C_{\mathrm{query}}$.

Full RouterBench uses a cleaned split with 29,194 training and 7,299 test queries and 11 candidates. The Qwen features use left padding, last-nonpadding-token pooling, float32, maximum sequence length 512, and normalization. Candidate strings in this RouterBench run are model names.

C.4 Candidate-call cost metrics

For xRouteBench, the selected model m's response price is computed from recorded token counts.

$$c(q, m) = \frac{n_{\mathrm{in}}(q, m) p_{\mathrm{in}}(m) + n_{\mathrm{out}}(q, m) p_{\mathrm{out}}(m)}{10^6}, \tag{39}$$

where token prices are in USD per million tokens. Observed test token counts appear only in retrospective price calculation. RouterBench instead supplies a recorded `total_cost` for each response. Its adapter encodes this cost in integer micro-USD through the shared token-count fields with a unit output-price coefficient. Those fields are cost surrogates, not observed token counts or vendor prices. The reconstructed cost differs from the nonnegative source value by at most 0.5×10^{-6}

USD per response due to rounding. The current RouterBench comparison replays each quality-first policy on the same 7,299 test queries and averages these stored costs for the selected candidates. These monetary figures exclude routing computation and do not impose a hard budget.

C.5 Offline computational cost

Table 8 reports recorded offline times and routing quality under each method's training procedure. Offline build includes fitting and method-specific preparation. Shared feature encoding is excluded. Quality uses the same 7,299 test queries as the main comparison.

Table 8. Recorded RouterBench training time, offline preparation time, and routing quality. CogRouter averages the same three runs for timing and quality. Fit time is included in offline-build time. Bold marks the highest quality.

Router	Fit (s) ↓	Offline build (s) ↓	Quality (%) ↑
kNNRouter	0.21	9.29	49.84
SVMRouter	5,995.99	6,005.21	47.98
MLPRouter	205.20	214.68	54.35
CausalLM	12,898.00	12,946.82	58.42
EloRouter	24.71	34.44	43.10
RouterDC	965.17	978.98	64.32
GraphRouter	428.87	478.21	65.12
CogRouter	380.92	502.57	**78.20**

D Task-domain prestudy details

Data and representations. We reuse RouterBench checkpoint score exports on all 7,299 test queries. The original eight Semantic Embedding clusters were fitted on training Semantic Embeddings only. The case in Figure 1 includes every HellaSwag and MMLU Professional Accounting test query assigned to cluster 0, totaling 114 and 18 queries, respectively. These are recorded task types, not independent difficulty annotations.

For kNN, SVM, and MLP we use fitted class-probability outputs. For MFRouter and CogRouter we use learned matching scores. All routers cover the same 11 candidates. Each candidate-output vector is centered and normalized to unit Euclidean norm with floor 10^{-12}. Frozen query Semantic Embeddings are unchanged. PCA is fitted separately to each representation using all test members of the source cluster without task labels, and the fixed task groups are displayed afterward. Axes belong to different spaces. Coincident same-task points are aggregated without moving their coordinates, with marker area increasing with multiplicity.

The predictions returned by the kNN/SVM/MLP implementations match their benchmark routes on all 7,299 queries. SVM's probability argmax disagrees with the class returned by its prediction function on 132 queries, so quality is evaluated using the class returned by that function. The silhouette computed from SVM's decision-function outputs is 0.444 on this case. MLP pre-softmax logits give 0.076. This additional diagnostic also finds task structure in SVM outputs and weak separation in MLP outputs.

Exploratory selection and interpretation. The case comes from a post-hoc search of 1,147 within-cluster task-pair comparisons, requiring at least 15 queries per task. It was chosen through inspection of Semantic Embedding overlap and routing-output separation, rather than random sampling. The query IDs were fixed before the additional kNN/SVM/MLP exports, and no query was removed based on these methods' predictions. This selected case motivates candidate-relative representations. It cannot estimate their prevalence or establish the quality of the proposed router.

The supplementary security case retains 86 HellaSwag and 25 MMLU Computer Security queries from a second fixed cluster. Both case memberships were fixed before replaying the current model. These case results do not provide an unbiased estimate of overall routing gains.

Table 9. Full-space silhouette and routing quality from the original implementations on both fixed exploratory cases. Silhouette diagnoses within-representation geometry. It is not a routing-quality metric.

Representation	Accounting ($n = 132$) Silhouette	Quality	Security ($n = 111$) Silhouette	Quality
Semantic Embedding	0.040	–	0.038	–
kNNRouter	-0.017	43.94%	0.040	45.05%
SVMRouter	0.314	44.70%	0.405	49.55%
MLPRouter	0.064	53.03%	0.020	49.55%
MFRouter	0.135	43.94%	0.096	56.76%
CogRouter	0.583	86.36%	0.543	89.19%

D.1 Projection checks and source-cluster context

We retain the same task groups under a fixed t-SNE projection (perplexity 30, PCA initialization, automatic learning rate, 1,000 iterations, and random state 42). Each projection uses the complete source-cluster test context without labels. Frozen Semantic Embeddings are first reduced to 50 PCA dimensions. Candidate-output vectors are used directly.

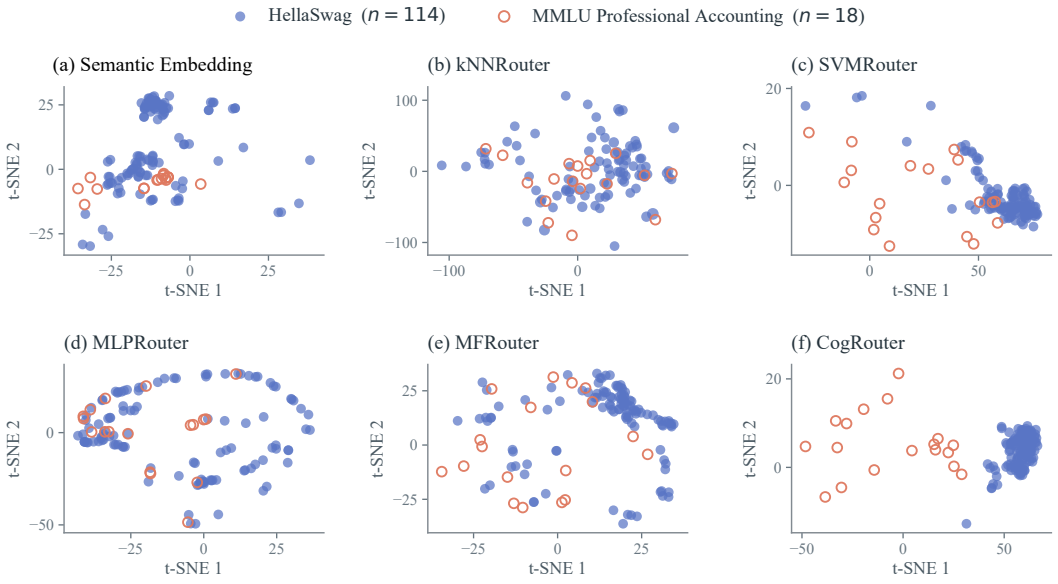


Figure 4. Fixed t-SNE view of the accounting case with the same 114 HellaSwag and 18 Professional Accounting queries as Figure 1. Each panel projects its own representation.

The nonlinear projection exposes local structure that is less visible in PCA, including structure in the frozen Semantic Embeddings. Full-space silhouette in Table 9 provides a check independent of the two-dimensional layout. These views support examining how candidate-relative outputs reorganize requests, rather than treating visual gaps as calibrated distances or assuming that Semantic Embeddings contain no task information.

Figure 5 places the selected tasks within the surrounding query distribution. Each panel includes the same background query IDs, making the selected groups' position within that representation visible.

D.2 Complementary task pair and routing decisions

The security case contains all 86 HellaSwag and 25 MMLU Computer Security queries in the second fixed cluster. Figure 6 shows this case with the current model scores. The original Semantic Embeddings already contain some local task structure, so this case illustrates another change in representation rather than a second instance of complete semantic overlap.

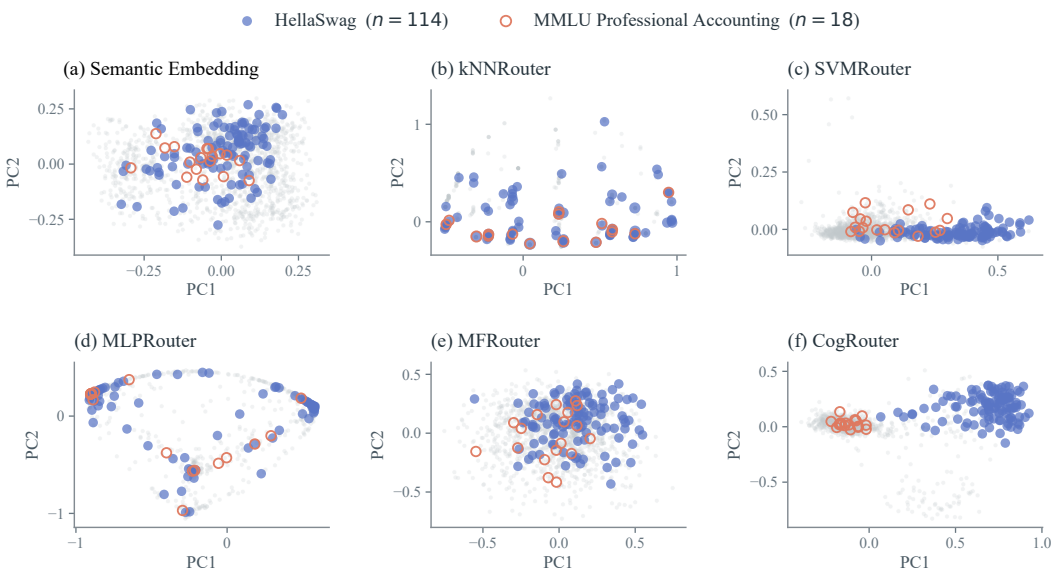


Figure 5. Accounting-case PCA in complete source-cluster context. Gray points are the other test queries used to fit each projection. Colored points are the fixed task groups. The candidate-output panels include kNN, SVM, MLP, original MFRouter, and CogRouter.

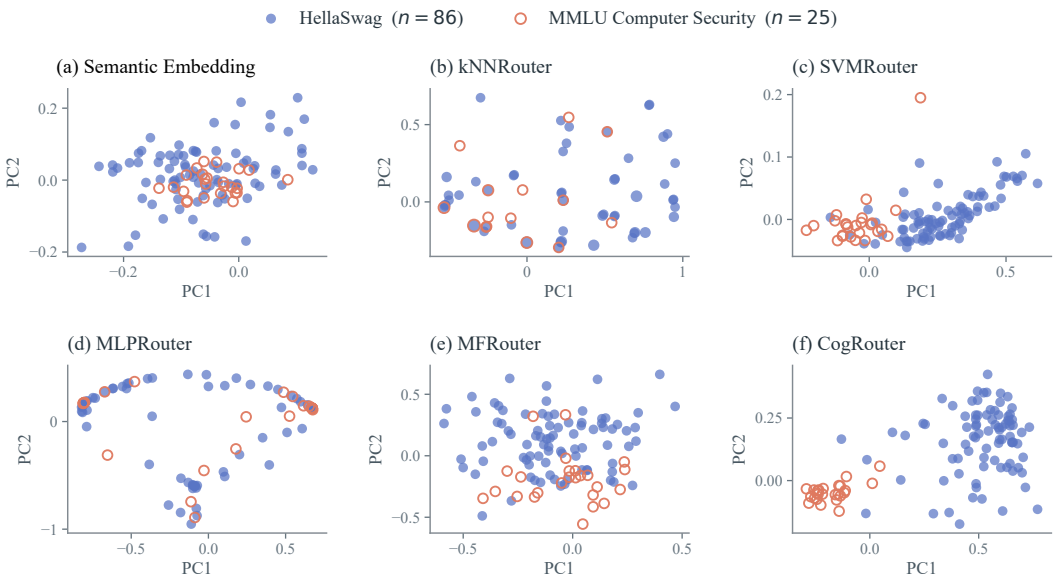


Figure 6. PCA of the fixed 111-query security case. All six panels use identical query membership. Each projection is fitted on its complete source-cluster context.

CogRouter has full-space silhouette 0.543 and routing quality 89.19% on this case in the displayed run. These results concern the fixed exploratory case. The benchmark comparisons evaluate overall routing performance.

From score geometry to selection. In the displayed run, CogRouter assigns every accounting and security query to GPT-4-1106-preview, while splitting the HellaSwag queries between that model and Yi-34B-Chat. Thus different task groups can occupy distinct regions of the full candidate-score space while sharing their highest-scoring model. The scatter plots describe the richer compatibility pattern before selection. The reported routing quality evaluates the resulting decisions. Neither observation alone attributes the change to a particular model component.

E SUPPLEMENTARY ROUTING-QUALITY RESULTS

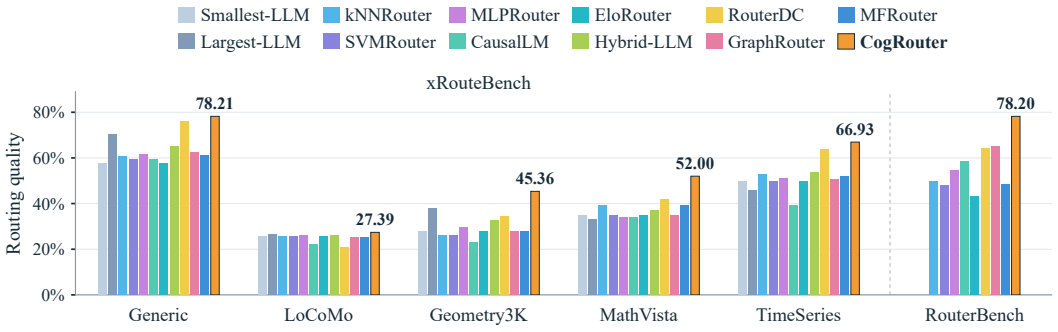


Figure 7. Routing quality on five xRouteBench domains (4,331 requests in total) and full RouterBench (7,299 requests). Bars reproduce the main comparisons under each method's training protocol. Labels mark CogRouter. Methods without RouterBench results are omitted from that group.

E.1 FIVE-DOMAIN COMPARISON

Figure 8 summarizes the method comparison in Table 1 on the five xRouteBench datasets. The same absolute color scale is used for every domain. CogRouter has the highest displayed mean in each column, while the relative ordering of the baselines changes across domains.

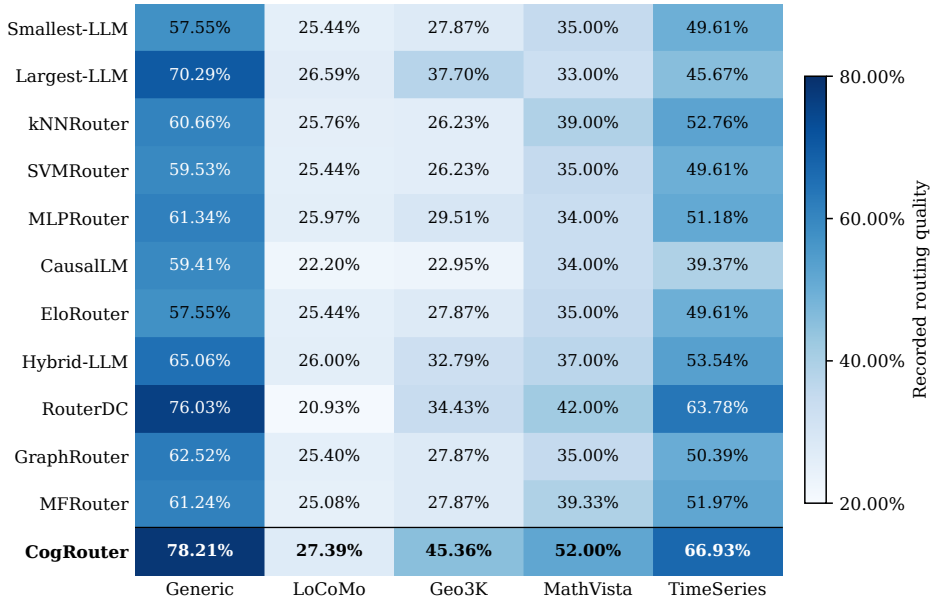


Figure 8. Routing quality across the five xRouteBench datasets. Cells show recorded quality as percentages and use the same method results as the main comparison. The color scale is shared across domains.

E.2 COMPLETE ROUTERBENCH TASK BREAKDOWN

Table 10 reports all 85 original task labels, covering the same 7,299 queries as the overall comparison. CogRouter and MFRouter use the mean of their three training runs. MFRouter is the original baseline. Overall weights tasks by their query counts. kNN, SVM, MLP, MF, Elo, DC, Graph, and Causal abbreviate the corresponding routers in the main comparison. These are task-level evaluations of the full-benchmark routers, with no task-specific retraining.

Table 10. Full RouterBench quality by original task label. CogRouter and MFRouter average three training runs. Other methods use the evaluated baseline checkpoints. Each task averages its own queries. Bold marks the best displayed value, including ties.

Task label	n	kNN	SVM	MLP	MF	Elo	DC	Graph	CogRouter	Causal
Chinese_character_riddles	23	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**8.70%**	0.00%
abstract2title	39	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**
accounting_audit	5	60.00%	60.00%	80.00%	60.00%	60.00%	0.00%	40.00%	**100.00%**	80.00%
arc-challenge	320	61.25%	57.81%	70.00%	59.27%	57.81%	72.19%	85.31%	**97.19%**	69.06%
bias_detection	60	26.67%	20.00%	21.67%	21.67%	20.00%	18.33%	5.00%	**35.00%**	20.00%
chinese-lantern-riddles	1	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**
chinese-remainder-theorem	2	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**
chinese_ancient_masterpieces_dynasty	4	25.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**75.00%**	25.00%	0.00%
chinese_ancient_poetry	3	0.00%	0.00%	0.00%	22.22%	0.00%	0.00%	**33.33%**	0.00%	0.00%
chinese_chu_ci	3	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**
chinese_famous_novel	5	0.00%	0.00%	0.00%	20.00%	0.00%	0.00%	0.00%	**40.00%**	0.00%
chinese_hard_translations	5	20.00%	**40.00%**	**40.00%**	20.00%	**40.00%**	0.00%	0.00%	0.00%	**40.00%**
chinese_homonym	6	**33.33%**	16.67%	0.00%	22.22%	16.67%	0.00%	16.67%	**33.33%**	**33.33%**
chinese_idioms	5	25.00%	25.00%	25.00%	25.00%	25.00%	**45.00%**	**45.00%**	**45.00%**	25.00%
chinese_modern_poem_identification	3	0.00%	0.00%	**33.33%**	0.00%	0.00%	0.00%	0.00%	**33.33%**	0.00%
chinese_poem	3	**33.33%**	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**33.33%**	0.00%
chinese_shi_jing	11	9.09%	9.09%	18.18%	9.09%	9.09%	0.00%	9.09%	**36.36%**	9.09%
chinese_song_ci	3	**66.67%**	0.00%	33.33%	0.00%	0.00%	0.00%	0.00%	**66.67%**	0.00%
chinese_tang_poetries	7	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**14.29%**	0.00%
chinese_zodiac	75	37.33%	32.00%	36.00%	23.11%	8.00%	5.33%	32.00%	**66.67%**	26.67%
consensus_summary	63	66.27%	67.06%	70.63%	67.46%	64.68%	**73.41%**	67.06%	66.53%	65.48%
grade-school-math	1468	55.48%	54.05%	60.58%	53.69%	52.06%	66.23%	55.23%	**66.58%**	59.52%
hellaswag	2033	45.70%	48.55%	48.79%	45.89%	33.25%	50.91%	73.88%	**84.29%**	63.85%
mbpp	89	37.08%	30.34%	47.19%	34.83%	30.34%	28.09%	33.71%	**63.30%**	35.96%
mmlu-abstract-algebra	16	**37.50%**	25.00%	31.25%	29.17%	25.00%	**37.50%**	31.25%	**37.50%**	31.25%
mmlu-anatomy	28	50.00%	39.29%	57.14%	44.05%	39.29%	**78.57%**	75.00%	**78.57%**	53.57%
mmlu-astronomy	36	52.78%	41.67%	47.22%	50.93%	41.67%	**88.89%**	69.44%	**88.89%**	61.11%
mmlu-business-ethics	24	58.33%	58.33%	66.67%	58.33%	58.33%	**95.83%**	91.67%	**95.83%**	58.33%
mmlu-clinical-knowledge	46	54.35%	54.35%	65.22%	56.52%	56.52%	**89.13%**	76.09%	**89.13%**	65.22%
mmlu-college-biology	30	46.67%	46.67%	60.00%	46.67%	46.67%	**90.00%**	73.33%	**90.00%**	53.33%
mmlu-college-chemistry	19	**36.84%**	**36.84%**	26.32%	**36.84%**	**36.84%**	**36.84%**	31.58%	**36.84%**	**36.84%**
mmlu-college-computer-science	26	30.77%	26.92%	69.23%	23.08%	26.92%	**76.92%**	50.00%	**76.92%**	53.85%
mmlu-college-mathematics	15	26.67%	13.33%	40.00%	31.11%	20.00%	**66.67%**	33.33%	**66.67%**	46.67%
mmlu-college-medicine	44	40.91%	40.91%	54.55%	49.24%	38.64%	79.55%	70.45%	**81.82%**	63.64%
mmlu-college-physics	12	25.00%	25.00%	41.67%	30.56%	25.00%	50.00%	33.33%	50.00%	**58.33%**
mmlu-computer-security	27	55.56%	59.26%	66.67%	66.67%	59.26%	**81.48%**	77.78%	**81.48%**	62.96%
mmlu-conceptual-physics	49	36.73%	30.61%	46.94%	34.69%	30.61%	**73.47%**	65.31%	**73.47%**	51.02%
mmlu-econometrics	24	41.67%	33.33%	45.83%	31.94%	37.50%	**70.83%**	41.67%	**70.83%**	33.33%
mmlu-electrical-engineering	27	59.26%	48.15%	59.26%	53.09%	48.15%	**70.37%**	66.67%	**70.37%**	**70.37%**
mmlu-elementary-mathematics	79	27.85%	22.78%	43.04%	32.91%	25.32%	**67.09%**	43.04%	**67.09%**	41.77%
mmlu-formal-logic	28	42.86%	42.86%	46.43%	46.43%	46.43%	**64.29%**	50.00%	**64.29%**	53.57%
mmlu-global-facts	14	42.86%	42.86%	**50.00%**	42.86%	42.86%	35.71%	**50.00%**	35.71%	**50.00%**
mmlu-high-school-biology	67	56.72%	55.22%	58.21%	56.72%	55.22%	**94.03%**	77.61%	**94.03%**	67.16%
mmlu-high-school-chemistry	48	37.50%	18.75%	45.83%	25.00%	20.83%	**68.75%**	41.67%	**68.75%**	43.75%
mmlu-high-school-computer-science	20	50.00%	45.00%	55.00%	55.00%	55.00%	**90.00%**	75.00%	**90.00%**	65.00%
mmlu-high-school-european-history	33	66.67%	66.67%	69.70%	70.71%	66.67%	57.58%	**90.91%**	**90.91%**	78.79%
mmlu-high-school-geography	45	68.89%	62.22%	77.78%	61.48%	62.22%	**93.33%**	71.11%	**93.33%**	75.56%
mmlu-high-school-government-and-politics	36	63.89%	55.56%	69.44%	60.19%	55.56%	**97.22%**	86.11%	**97.22%**	66.67%
mmlu-high-school-macroeconomics	69	40.58%	36.23%	49.28%	34.30%	36.23%	**89.86%**	62.32%	**89.86%**	49.28%
mmlu-high-school-mathematics	48	29.17%	29.17%	29.17%	30.56%	29.17%	**41.67%**	20.83%	**41.67%**	31.25%
mmlu-high-school-microeconomics	33	54.55%	48.48%	54.55%	53.54%	45.45%	**93.94%**	69.70%	**93.94%**	57.58%
mmlu-high-school-physics	36	36.11%	22.22%	33.33%	27.78%	22.22%	**55.56%**	27.78%	**55.56%**	27.78%
mmlu-high-school-psychology	102	65.69%	61.76%	72.55%	62.09%	61.76%	**97.06%**	90.20%	**97.06%**	68.63%
mmlu-high-school-statistics	47	25.53%	17.02%	25.53%	26.95%	19.15%	**76.60%**	51.06%	75.89%	40.43%
mmlu-high-school-us-history	45	73.33%	62.22%	71.11%	68.89%	64.44%	82.22%	88.89%	**97.78%**	77.78%
mmlu-high-school-world-history	46	56.52%	52.17%	67.39%	54.35%	52.17%	69.57%	84.78%	**86.96%**	78.26%
mmlu-human-aging	48	56.25%	50.00%	56.25%	52.08%	50.00%	**83.33%**	75.00%	**83.33%**	52.08%
mmlu-human-sexuality	23	60.87%	56.52%	60.87%	56.52%	56.52%	**86.96%**	**86.96%**	**86.96%**	60.87%
mmlu-international-law	20	80.00%	80.00%	80.00%	80.00%	80.00%	**90.00%**	**90.00%**	**90.00%**	**90.00%**
mmlu-jurisprudence	22	59.09%	59.09%	59.09%	59.09%	59.09%	**81.82%**	77.27%	**81.82%**	72.73%

Continued on next page

Continued from previous page

Task label	n	kNN	SVM	MLP	MF	Elo	DC	Graph	CogRouter	Causal
mmlu-logical-fallacies	50	52.00%	48.00%	62.00%	46.67%	48.00%	**94.00%**	78.00%	**94.00%**	60.00%
mmlu-machine-learning	18	44.44%	33.33%	50.00%	38.89%	33.33%	**77.78%**	66.67%	**77.78%**	55.56%
mmlu-management	19	63.16%	57.89%	63.16%	63.16%	57.89%	**89.47%**	84.21%	**89.47%**	52.63%
mmlu-marketing	43	62.79%	65.12%	76.74%	65.12%	65.12%	**97.67%**	88.37%	**97.67%**	69.77%
mmlu-medical-genetics	27	66.67%	70.37%	70.37%	64.20%	70.37%	**96.30%**	81.48%	**96.30%**	70.37%
mmlu-miscellaneous	147	63.95%	59.86%	69.39%	61.00%	59.86%	**93.88%**	87.07%	**93.88%**	69.39%
mmlu-moral-disputes	66	34.85%	31.82%	50.00%	33.33%	31.82%	**74.24%**	56.06%	**74.24%**	48.48%
mmlu-moral-scenarios	177	29.38%	20.34%	42.94%	26.55%	20.34%	**75.71%**	50.28%	**75.71%**	31.64%
mmlu-nutrition	78	62.82%	58.97%	66.67%	58.12%	58.97%	**87.18%**	71.79%	**87.18%**	67.95%
mmlu-philosophy	65	49.23%	50.77%	53.85%	53.33%	50.77%	**87.69%**	72.31%	**87.69%**	61.54%
mmlu-prehistory	59	50.85%	47.46%	64.41%	53.67%	47.46%	**88.14%**	74.58%	**88.14%**	57.63%
mmlu-professional-accounting	55	43.64%	32.73%	43.64%	35.76%	32.73%	65.45%	**67.27%**	65.45%	36.36%
mmlu-professional-law	294	42.18%	35.37%	41.84%	35.26%	35.03%	59.86%	47.96%	**66.55%**	46.94%
mmlu-professional-medicine	52	50.00%	44.23%	59.62%	47.44%	34.62%	**92.31%**	69.23%	**92.31%**	61.54%
mmlu-professional-psychology	116	50.86%	46.55%	55.17%	47.70%	46.55%	**83.62%**	74.14%	**83.62%**	57.76%
mmlu-public-relations	23	56.52%	56.52%	52.17%	56.52%	56.52%	**86.96%**	78.26%	**86.96%**	60.87%
mmlu-security-studies	48	54.17%	52.08%	56.25%	54.86%	52.08%	**75.00%**	68.75%	74.31%	60.42%
mmlu-sociology	35	74.29%	74.29%	68.57%	73.33%	74.29%	**91.43%**	74.29%	**91.43%**	68.57%
mmlu-us-foreign-policy	21	76.19%	76.19%	71.43%	76.19%	76.19%	**90.48%**	66.67%	**90.48%**	85.71%
mmlu-virology	32	50.00%	**53.12%**	40.62%	**53.12%**	**53.12%**	46.88%	40.62%	46.88%	46.88%
mmlu-world-religions	29	68.97%	68.97%	79.31%	68.97%	68.97%	**93.10%**	82.76%	**93.10%**	72.41%
mtbench	10	77.00%	76.00%	79.00%	75.67%	76.00%	74.00%	**90.00%**	87.00%	78.00%
mtbench-math	6	40.00%	36.67%	36.67%	36.67%	40.00%	25.00%	31.67%	**55.00%**	28.33%
mtbench-reference	5	46.00%	28.00%	24.00%	32.00%	28.00%	24.00%	34.00%	**74.00%**	60.00%
winogrande	256	55.08%	51.56%	56.25%	52.60%	51.56%	49.61%	67.58%	**86.20%**	57.42%
Overall	7299	49.84%	47.98%	54.35%	48.24%	43.10%	64.32%	65.12%	**78.20%**	58.42%

F QUALITY–COST TRADE-OFFS

Throughout the cost tables and figures, ↑ means higher is preferred and ↓ means lower is preferred for that metric. Budget and quality-target columns specify evaluation conditions. The cost sweep reuses each trained quality checkpoint and changes only the inference preference. Let $\widetilde{s}(q, m)$ be each method's candidate scores min–max normalized across its candidate support, and $\widetilde{c}_m$ the similarly normalized expected candidate costs. We select

$$\hat{m}(q) = \arg\max_m \left[\alpha \widetilde{s}(q, m) - \beta \widetilde{c}_m\right], \qquad \beta = 1 - \alpha.$$

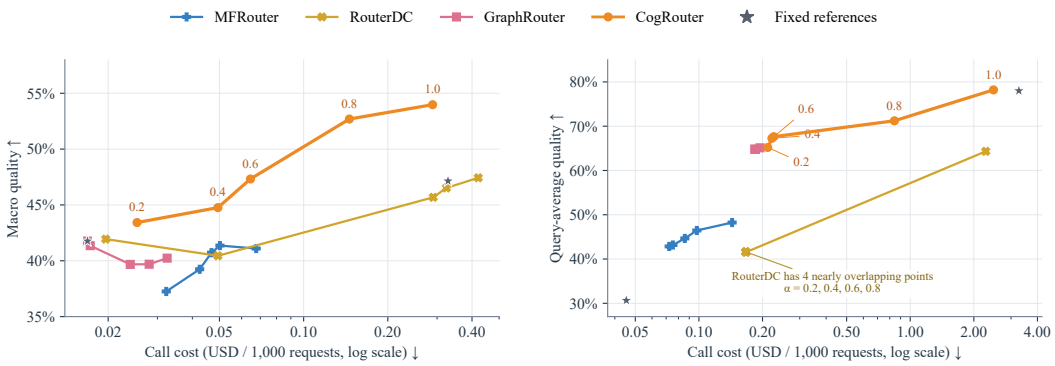
Normalization uses a denominator floor of 10^{-12}. For CogRouter, expected candidate costs are mean stored response costs in the fitting data, excluding validation queries. Baseline replay uses archived training costs, excluding recorded query texts from each method's validation split when available. On xRouteBench these costs use recorded input/output token counts and the archived candidate prices. On RouterBench they use the benchmark's recorded response costs carried through the adapter, as specified in Appendix C.4. Test replay uses the selected candidate's stored response cost under the same respective convention. Test outcomes do not determine the expected-cost vector or checkpoint selection.

F.1 MEASURED PREFERENCE SETTINGS

We apply the same five inference preferences to CogRouter and eight xRouteBench baselines, kNNRouter, SVMRouter, MLPRouter, MFRouter, EloRouter, Hybrid-LLM, RouterDC, and GraphRouter. RouterBench includes the same swept methods except Hybrid-LLM, whose original implementation could not be initialized with the available metadata. Smallest-LLM and Largest-LLM on xRouteBench, and CausalLM on both benchmarks, retain their single recorded policies. The sweep extends each method with a shared inference cost penalty while retaining its archived scoring rule, candidate support, and training settings. It is not a comparison of separately optimized versions of each method's original budget-control algorithm.

F.2 DEFINITIONS AND COMPUTATION OF COST-EFFECTIVENESS METRICS

Inputs, units, and fixed references. For a policy evaluated on N requests, let b_i be the recorded USD charge for its selected candidate on request i. Its reported call cost is $C = (1000/N) \sum_{i=1}^{N} b_i$,



(a) xRouteBench five-domain macro

(b) RouterBench with 7,299 queries

Figure 9. Measured quality–cost curves for four selected methods on five-domain xRouteBench (4,331 requests) and RouterBench (7,299). CogRouter, MFRouter, RouterDC, and GraphRouter each use $\alpha \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ with $\beta = 1 - \alpha$. Labels identify CogRouter's preferences. Lines connect consecutive settings. Four RouterDC points nearly overlap on RouterBench. Stars mark the fixed references in Table 12. Quality weights domains equally on xRouteBench and requests equally on RouterBench. Costs weight requests and exclude router computation. Table 4 includes the full measured method comparison.

Table 11. All five measured preferences of CogRouter, averaged over three runs. Savings and retained quality are relative to $(1, 0)$ in the same benchmark. Five-domain quality weights domains equally. Costs weight requests. RouterBench weights requests for both. Total USD is the mean replay bill for the complete test set.

Five-domain xRouteBench (α, β)	Quality ↑	$/1K ↓	Total USD ↓	Cost saving ↑	Quality retained ↑
(1.0, 0.0)	53.98%	0.289	1.253	0.00%	100.00%
(0.8, 0.2)	52.69%	0.146	0.631	49.62%	97.62%
(0.6, 0.4)	47.32%	0.065	0.279	77.69%	87.67%
(0.4, 0.6)	44.76%	0.049	0.214	82.93%	82.92%
(0.2, 0.8)	43.42%	0.025	0.110	91.23%	80.45%
RouterBench (α, β)	Quality ↑	$/1K ↓	Total USD ↓	Cost saving ↑	Quality retained ↑
(1.0, 0.0)	78.20%	2.477	18.082	0.00%	100.00%
(0.8, 0.2)	71.22%	0.842	6.143	66.02%	91.08%
(0.6, 0.4)	67.64%	0.226	1.651	90.87%	86.49%
(0.4, 0.6)	67.27%	0.222	1.617	91.06%	86.02%
(0.2, 0.8)	65.20%	0.212	1.546	91.45%	83.38%

excluding router training and inference overhead. Quality Q is a fraction that averages the five domain scores equally on xRouteBench and the request scores equally on RouterBench. Costs weight requests equally on both benchmarks. Table 12 supplies the weak and strong fixed-candidate references (C_w, Q_w) and (C_s, Q_s), with $C_s > C_w$ and $Q_s > Q_w$. Define

$$x(C) = \frac{C - C_w}{C_s - C_w}, \qquad p(Q) = \frac{Q - Q_w}{Q_s - Q_w}. \tag{40}$$

Thus $p = 0$ and $p = 1$ denote the weak and strong reference quality. Each method contributes its measured policy set $\mathcal{S}$, containing five preferences for CogRouter and each swept baseline listed in Appendix F, and one configuration for each fixed policy or CausalLM. Both references are added separately.

Raw mAPGR, clipped mAPGR, and CEI (↑). Append both references to $\mathcal{S}$, merge equal-cost points by retaining the largest quality, and sort the resulting knots by cost as (c_j, q_j). At every knot take the cumulative best quality $\bar{q}_j = \max_{i \le j} q_i$. Let $M(C)$ interpolate $(c_j, \bar{q}_j)$ linearly and set $r_j = p(\bar{q}_j)$. Let $P(x)$ interpolate the knots $(x(c_j), r_j)$, and let $P_{\text{clip}}(x)$ interpolate

$(x(c_j), \min\{1, \max\{0, r_j\}\})$. Then

$$\mathrm{mAPGR}_{\mathrm{raw}} = \int_0^1 P(x)\,dx, \qquad \mathrm{mAPGR}_{\mathrm{clip}} = \int_0^1 P_{\mathrm{clip}}(x)\,dx, \tag{41}$$

$$\mathrm{CEI} = 100\,\mathrm{mAPGR}_{\mathrm{clip}}. \tag{42}$$

These areas summarize recovery of the reference quality gap across a uniform cost interval. Raw recovery can exceed one when routing exceeds the strong reference. Clipped mAPGR lies in $[0, 1]$ and CEI in $[0, 100]$. CEI is a rescaling of the clipped area, not an independent metric. Clipping is applied to knot heights *before* interpolation, rather than to the interpolated curve.

CQT50 and CQT80 (↓). On the same cumulative-best curve, define the cost of recovering a fraction τ of the reference quality gap as

$$\mathrm{CQT}_{100\tau} = \inf\big\{C \in [C_w, C_s] : M(C) \geq Q_w + \tau(Q_s - Q_w)\big\}, \qquad \tau \in \{0.5, 0.8\}. \tag{43}$$

We invert each linear segment to obtain the minimum such cost, in USD per 1,000 requests. If the target is already reached at C_w, the reported value is C_w. The targets recover 50% or 80% of the *gap*. They are not absolute quality thresholds of 50% or 80%. Both targets are feasible because the strong reference is included.

These knot-interpolation summaries depend on the retained measurement locations, not only on the attainable policy set. Adding a dominated knot can lower mAPGR and CEI or raise CQT while leaving the attainable frontier unchanged. For example, with normalized references $(0, 0)$ and $(1, 1)$, adding the dominated point $(0.5, 0)$ changes CEI from 50 to 25 and CQT50 from 0.5 to 0.75, although the reference mixture still attains quality C at cost $C \in [0, 1]$. Thus CQT is a threshold on M, not the minimum achievable mixture cost. The envelope-based metrics below provide the latter interpretation.

AIQ (↑), expected budgets, and absolute quality targets. AIQ uses the upper concave quality envelope $F(C)$ of $\mathcal{S}$ and the references. Equivalently, writing their original points as (c_j, q_j),

$$\begin{aligned} F(C) &= \max_{\lambda_j \geq 0} \left\{ \sum_j \lambda_j q_j : \sum_j \lambda_j = 1,\ \sum_j \lambda_j c_j \leq C \right\}, \\ \mathrm{AIQ} &= \frac{1}{C_s - C_w} \int_{C_w}^{C_s} F(C)\,dC. \end{aligned} \tag{44}$$

AIQ is the average quality over a uniform reference cost interval, displayed as a percentage. Unlike the cumulative-best curve used for mAPGR and CQT, F selects the best attainable mixtures across all supplied policies. If no measured point improves the fixed-reference envelope on this interval, AIQ equals $(Q_w + Q_s)/2$. Equal five-domain baseline AIQ need not imply equal routing quality.

The supplementary budget table reports $100F(B)\%$ at budget B, and the absolute-target table reports

$$C^*(q) = \inf\{C \geq \min_j c_j : F(C) \geq q\}, \tag{45}$$

using CogRouter's five preferences and the same references. Here q is an absolute quality target, unlike the relative gap targets in CQT. Budgets below the cheapest supplied policy and targets above the largest supplied quality are marked with a dash.

Table 4 summarizes the complete available policy sets together with the common references, rather than only the four methods selected for the figure. Every swept method contributes all five preferences. Fixed policies and CausalLM contribute one point each. Most five-domain baseline AIQ values are set by the fixed-reference envelope, whereas RouterDC slightly improves that envelope. Equal AIQ values therefore do not imply identical routing behavior.

F.3 Expected budgets and quality targets

The following tables summarize mixtures of the five measured preferences and the fixed references. They are interpolated expectations. A dash denotes an unattainable target.

Table 12. Fixed-candidate cost references used for normalization. Each benchmark lists its weak reference first and strong reference second. These reporting references do not select router checkpoints or inference preferences.

Five-domain xRouteBench Fixed reference	Quality ↑	$/1K ↓
gpt-oss-20b	41.76%	0.017
llama-4-maverick	47.14%	0.328
RouterBench Fixed reference	Quality ↑	$/1K ↓
mistralai/mistral-7b-chat	30.66%	0.045
gpt-4-1106-preview	77.99%	3.267

Table 13. Expected quality at specified call budgets using CogRouter's complete five-preference sweep and the fixed references. Columns are budget conditions, in USD per 1,000 requests.

Five-domain xRouteBench					
Budget (USD/1K)	$0.025	$0.050	$0.100	$0.200	$0.300
Quality ↑	43.35%	45.88%	49.67%	53.18%	53.98%
RouterBench					
Budget (USD/1K)	$0.100	$0.250	$0.500	$1.000	$2.500
Quality ↑	42.01%	67.78%	69.23%	71.90%	78.20%

Table 14. Minimum expected call cost for absolute quality targets using the same five-preference policy family and fixed references. Costs are USD per 1,000 requests.

Five-domain xRouteBench					
Target quality	42.00%	45.00%	47.00%	50.00%	53.00%
Cost (USD/1K) ↓	0.018	0.041	0.061	0.105	0.180
RouterBench					
Target quality	50.00%	60.00%	70.00%	75.00%	78.00%
Cost (USD/1K) ↓	0.138	0.187	0.632	1.727	2.430