Understanding Input–Output Uncertainty in LLM Routing Supervision

Guannan Lai [1 2] **Han-Jia Ye** [1 2]

Abstract

As LLMs are increasingly used in agentic systems, routing has become a key mechanism for allocating queries or intermediate steps across models, tools, and reasoning modules under cost–quality constraints. Existing routing benchmarks typically treat a model's single observed correctness on a query as a stable supervision label. We argue that this label is only a noisy sample from an underlying query–model outcome distribution, affected by both input formulation and stochastic generation. This paper studies input–output uncertainty in LLM routing supervision by separating input-side uncertainty from semantically equivalent query reformulations and output-side uncertainty from stochastic decoding under a fixed input. Across reasoning benchmarks, we show that both sources induce correctness flips, unstable escalation labels, and inconsistent learned routing decisions even under a fixed router architecture, embedding model, and train–test split. As a diagnostic result, repeated-evaluation aggregation reduces supervision noise, suggesting that routing benchmarks should report label reliability rather than treating single-shot outcomes as deterministic ground truth.

1. Introduction

Large language models (LLMs) have become a general interface for solving diverse tasks, ranging from knowledge-intensive question answering to multi-step reasoning and tool-augmented problem solving. As LLMs are increasingly deployed as components of agentic systems, a central systems problem is routing: deciding which model, tool, or reasoning module should handle each query or intermediate step under resource constraints (Shen et al., 2023; Chen et al., 2023; Aggarwal et al., 2024; Ding et al., 2024; Ong et al., 2025). A router may choose between a smaller and cheaper model and a larger but more capable model, decide whether to rewrite the input or resample an answer, or escalate a difficult step to a stronger reasoning module. Effective routing is therefore important for balancing cost, latency, and response quality in agentic LLM systems.

Current routing benchmarks and supervised routing methods usually rely on a simple assumption: one sampled response is sufficient to characterize a model's capability on a given query (Shnitzer et al., 2023; Hu et al., 2024; Huang et al., 2025). In a typical setup, each candidate model answers each query once, and the observed performance and cost are recorded as supervision for router training. The router then learns to select an appropriate model under a cost–quality trade-off. This construction is simple and practical, but it treats the observed outcome of a single run as if it were a stable property of the query–model pair. For generative LLMs, this assumption is fragile: the same underlying task may be expressed through different but semantically equivalent inputs, and the same input may produce different responses under stochastic decoding. Thus, a single correctness label is better viewed as a noisy sample from an underlying query–model outcome distribution rather than as deterministic ground truth.

Recent work on uncertainty estimation and hallucination detection has shown that LLM generations can be sensitive to sampling randomness and may vary in both surface form and semantic content (Kuhn et al., 2023; Manakul et al., 2023; Farquhar et al., 2024). For routing supervision, however, uncertainty has an additional consequence: it does not only affect whether a model answer is correct, but also affects the labels used to train the router. From this perspective, routing-label uncertainty can be analyzed from two complementary sides. Input-side uncertainty arises when semantically equivalent reformulations of the same query lead to different model outcomes. Output-side uncertainty arises when repeated stochastic decoding under a fixed input leads to different outcomes. Both sources can change the observed correctness of the smaller or larger model, and therefore change whether a query appears to require escalation. This raises a fundamental question for LLM routing: can a single sampled response reliably serve as the supervision signal for model selection?

[1]School of Artificial Intelligence, Nanjing University, China [2]National Key Laboratory for Novel Software Technology, Nanjing University, China. Correspondence to: Han-Jia Ye <yehj@lamda.nju.edu.cn>.

Accepted to the ICML 2026 Workshop on Statistical Frameworks for Uncertainty in Agentic Systems, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

In this work, we study this question through an empirical analysis of input–output uncertainty in LLM routing supervision. On the GPQA benchmark, we construct a model pool containing six recent LLMs and evaluate model behavior under both input reformulations and repeated decoding samples. This repeated-evaluation protocol allows us to measure how often model correctness changes across valid observations, how such changes alter escalation labels, and whether label uncertainty propagates to learned routing policies. We find that input-side and output-side uncertainty substantially affect model outcomes across the model pool. Moreover, when we repeatedly sample routing labels and train 100 routers while keeping the router architecture, query embeddings, and train–test split fixed, the resulting routers exhibit large variance in their routing decisions. These results indicate that routing instability can originate not only from router design, but also from the unreliability of the supervision labels themselves.

Finally, as a diagnostic analysis, we examine whether repeated evaluations can provide more stable routing supervision. Instead of treating a single sampled response as ground truth, we aggregate observations across input reformulations and decoding samples to estimate more reliable query–model outcomes and escalation signals. Routers trained from these aggregated signals achieve more stable and cost-effective routing behavior on GPQA, suggesting that routing benchmarks should account for label reliability rather than assuming that single-shot outcomes are fixed labels. Our contributions are summarized as follows:

- We formulate LLM routing supervision as a noisy observation problem. Instead of treating single-shot correctness as deterministic ground truth, we view it as one sampled outcome from an underlying query–model distribution.
- We analyze two sources of routing-label uncertainty: input-side uncertainty from semantically equivalent query reformulations and output-side uncertainty from stochastic decoding. On GPQA with a multi-model LLM pool, we show that both sources can substantially change model correctness and escalation labels.
- We show that uncertainty in routing labels propagates to learned routing policies. Even with the same router architecture, query representation, and train–test split, routers trained from different sampled labels make inconsistent decisions, while repeated-evaluation aggregation provides a more stable supervision signal.

2. Related Work

LLM routing. LLM routing studies how to allocate each input query or intermediate step to an appropriate model in a heterogeneous model pool, with the goal of balancing response quality and inference cost. A first line of work frames routing as cost-aware model selection or cascading, where cheaper models are used whenever possible and stronger models are invoked only when necessary (Zhang et al., 2023; Yue et al., 2023; Lai & Ye, 2026). Building on this idea, recent methods learn query-dependent routing policies from different forms of supervision. Hybrid LLM and BEST-Route optimize adaptive routing under quality–cost constraints (Ding et al., 2025), RouteLLM learns relative model selection from preference data (Ong et al., 2025), and TensorOpera Router, EmbedLLM, IRT-Router, GraphRouter, and causal routing instantiate learned routing through regression, representation learning, item response theory, graph modeling, or regret-based formulations (Stripelis et al., 2024; Zhuang et al., 2024; Song et al., 2025; Feng et al., 2025; Tsiourvas et al., 2025; Ma et al., 2026). In parallel, other studies reduce router complexity or improve deployment controllability through training-free, retrieval-based, budget-controllable, or preference-aware routing strategies (Zhao et al., 2024; Li, 2025; Mei et al., 2025; Piskala et al., 2025). The development of routing benchmarks, such as RouterBench, RouterEval, RouteJudge, and MMR-Bench, further standardizes this problem by providing query–model outcomes, costs, and evaluation protocols for comparing routing policies (Hu et al., 2024; Huang et al., 2025; Lai et al., 2026; Ma et al., 2026). While these works have substantially advanced router design and evaluation, they generally treat the observed query–model outcome as a reliable supervision signal. Our work instead examines the reliability of this supervision itself, asking whether single-shot outcomes are stable enough to support router training and evaluation.

Uncertainty-aware and efficient LLM inference. Several lines of work improve LLM inference efficiency by estimating uncertainty, deciding when to defer to stronger models, or allocating computation adaptively. For example, uncertainty-based two-tier selection and related caching or distillation strategies reduce expensive model calls by identifying when a cheaper model is sufficient (Ramírez et al., 2024; 2023). Other work studies budget- and quality-controllable routing, quality-of-service-aware routing, or dynamic routing under deployment constraints (Mei et al., 2025; Yang et al., 2025b; Wang et al., 2025). Recent studies also examine routing robustness and vulnerabilities, showing that routing policies can be fragile under distribution shifts, adversarial settings, or lifecycle changes (Shafran et al., 2025; Lin et al., 2025; Kassem et al., 2025). These works highlight the importance of reliable model selection, but they typically use uncertainty as a test-time confidence signal or deployment-time decision criterion. In contrast, our work studies uncertainty in the supervision itself: we show that observed query–model scores can vary under valid

input reformulations and stochastic decoding samples, and that this label uncertainty can propagate to routing policies.

Uncertainty and robustness in language model generation. Our work is also related to uncertainty estimation and robustness analysis for LLM generation. Prior studies show that LLM outputs can vary substantially under sampling randomness, and that inconsistency across generations can reveal uncertainty or hallucination risk (Kuhn et al., 2023; Manakul et al., 2023; Farquhar et al., 2024). Another line of work studies prompt robustness, showing that semantically similar or adversarially perturbed prompts may lead to different model predictions (Gan & Mori, 2023; Zhu et al., 2023). These findings suggest that a model's behavior on a query cannot be fully characterized by a single generated response. We bring this insight to LLM routing supervision: instead of constructing routing labels from isolated sampled outcomes, we analyze how input-side and output-side uncertainty affect model correctness, escalation labels, and the routing policies learned from those labels.

3. Measuring Input–Output Uncertainty in Routing Supervision

3.1. Single-shot Supervision for LLM Routing

LLM routing considers a candidate model pool $\mathcal{M} = \{m_1, \ldots, m_K\}$ and aims to select an appropriate model for each input query x. Given a routing function $r(\cdot)$, the selected model is $r(x) \in \mathcal{M}$. The goal is to achieve a favorable trade-off between response quality and inference cost: difficult queries should be assigned to more capable models when necessary, while queries that can be solved by cheaper models should avoid unnecessary escalation.

In a typical routing benchmark, each model m produces a response a for query x. The response is evaluated by a task-specific scoring function $s(\cdot)$, such as exact-match accuracy for multiple-choice tasks or F1 for reading-comprehension tasks. The same generation also incurs an inference cost measured by $\kappa(\cdot)$, which may depend on the model, the query, the output length, and the decoding process. Thus, a query–model pair (x_i, m) induces both a quality observation and a cost observation.

Most existing routing datasets construct these observations from a single sampled response. For each query x_i and model m, one response is generated:

$$a_{i,m}^{\text{single}} \sim P(a \mid x_i, m),$$

and the resulting single-shot observations are recorded as

$$q_{i,m}^{\text{single}} = s(a_{i,m}^{\text{single}}, x_i), \quad c_{i,m}^{\text{single}} = \kappa(a_{i,m}^{\text{single}}, x_i, m).$$

Here, $q_{i,m}^{\text{single}}$ denotes the observed quality of model m on query x_i, while $c_{i,m}^{\text{single}}$ denotes the observed cost of this particular generation. These values are then treated as supervision for training or evaluating routing policies.

Given a dataset $\mathcal{D} = \{x_i\}_{i=1}^N$ and a cost budget $C_{\max}$, the routing objective can be written as

$$\max_r \frac{1}{N} \sum_{i=1}^{N} q_{i,r(x_i)}^{\text{single}} \quad \text{s.t.} \quad \frac{1}{N} \sum_{i=1}^{N} c_{i,r(x_i)}^{\text{single}} \leq C_{\max}.$$

Equivalently, in a two-model setting with a cheaper model m_s and a stronger model m_l, a single-shot escalation label can be defined as

$$z_i^{\text{single}} = \mathbb{I}\left[q_{i,m_l}^{\text{single}} - q_{i,m_s}^{\text{single}} > \lambda\right],$$

where λ denotes the cost-adjusted threshold for using the stronger model. This label indicates whether the observed quality gain of the stronger model is large enough to justify escalation.

This protocol is simple and scalable, but it relies on a strong assumption: one sampled response is treated as a stable characterization of the query–model pair. For generative LLMs, this assumption is questionable. Both the input formulation and the stochastic generation process can change the observed response, quality score, and therefore the induced routing label. The central question of this work is whether single-shot observations are reliable enough to support LLM routing supervision.

3.2. Input–Output Uncertainty in Query–Model Behavior

To analyze the reliability of routing supervision, we view the observed quality of a query–model pair as a random outcome rather than a deterministic label. For each original query x_i, let $x_i^{(p)}$ denote the p-th semantically equivalent input reformulation, where $p \in \{1, \ldots, P\}$. For each reformulated input, we perform D independent decoding runs. The d-th response from model m under input $x_i^{(p)}$ is denoted as

$$a_{i,m}^{(p,d)} \sim P(a \mid x_i^{(p)}, m),$$

and its observed quality is

$$Q_{i,m}^{(p,d)} = s(a_{i,m}^{(p,d)}, x_i).$$

This produces a repeated-observation matrix for each query–model pair:

$$\mathbf{Q}_{i,m} = \left[Q_{i,m}^{(p,d)}\right]_{p=1,d=1}^{P,D} \in [0,1]^{P \times D}.$$

Single-shot supervision corresponds to observing only one entry of this matrix, while repeated evaluation allows us to estimate the stability of the query–model behavior.

Input-side uncertainty. Input-side uncertainty measures how much model behavior changes when the same underlying query is expressed in different but semantically equivalent forms. We first average over decoding samples for each input reformulation and then compute the variance across reformulations:

$$U_{i,m}^{\text{in}} = \text{Var}_p \left(\frac{1}{D} \sum_{d=1}^{D} Q_{i,m}^{(p,d)} \right).$$

A large value indicates that the model's apparent capability depends strongly on the surface form of the input.

Output-side uncertainty. Output-side uncertainty measures how much model behavior changes across stochastic generations when the input is fixed. We compute the within-input variance across decoding samples and average it over input reformulations:

$$U_{i,m}^{\text{out}} = \frac{1}{P} \sum_{p=1}^{P} \text{Var}_d \left(Q_{i,m}^{(p,d)} \right).$$

A large value indicates that even for the same input formulation, the model may alternate between correct and incorrect responses across decoding runs.

Outcome instability. In addition to variance, we measure whether a query–model pair exhibits any correctness flip across repeated observations:

$$I_{i,m}^{\text{flip}} = \mathbb{I} \left[\min_{p,d} Q_{i,m}^{(p,d)} \neq \max_{p,d} Q_{i,m}^{(p,d)} \right].$$

For binary-scored tasks such as multiple-choice QA, this quantity directly indicates whether the same query–model pair can be both correct and incorrect under valid input or output variations. At the model level, we report the average flip rate, mean quality, mean variance, and mean prediction range across queries.

3.3. Diagnostic Analysis on GPQA

We conduct a diagnostic analysis on GPQA, a graduate-level multiple-choice question answering benchmark covering biology, physics, and chemistry (Rein et al., 2023). For each query, we generate five semantically preserving input reformulations using GPT-4o (Hurst et al., 2024), while keeping the gold answer and candidate choices unchanged. For each reformulated input and each model, we perform five independent decoding runs under the same evaluation protocol. This yields a 5×5 observation matrix for each query–model pair. Our model pool includes Gemma-3-12B-IT (Team, 2025), Mistral-Small-3.2-24B-Instruct (Mistral AI, 2025), Qwen3-32B (Yang et al., 2025a), Llama-3.3-70B-Instruct (Meta AI, 2024), Gemini-2.5-Flash-Lite (Google, 2025), and DeepSeek-Chat-V3.1 (DeepSeek-AI et al., 2024). The goal of this section is not to propose a new routing algorithm, but to examine whether the single-shot observations commonly used as routing supervision are stable.

Observation 1: Single-shot quality does not characterize model stability.

Figure 1a compares the mean quality and outcome flip rate of each model. The models differ substantially in average quality: for example, Llama-70B and DeepSeek-V3.1 achieve higher mean quality than Qwen3-32B and Gemini-Flash. However, all models exhibit high outcome instability, with flip rates concentrated around 0.68–0.78. This means that for a large fraction of query–model pairs, repeated valid observations contain both correct and incorrect outcomes. Therefore, a single sampled correctness label can easily misrepresent the model's behavior on the same query.

This finding is important for routing because model selection depends on relative model behavior. If the observed correctness of a model changes across input reformulations or decoding samples, then the estimated advantage of one model over another can also change. Thus, high mean quality does not imply reliable routing supervision, and low mean quality does not imply that the model is consistently wrong. A routing benchmark that records only one response per query–model pair loses this distinction.

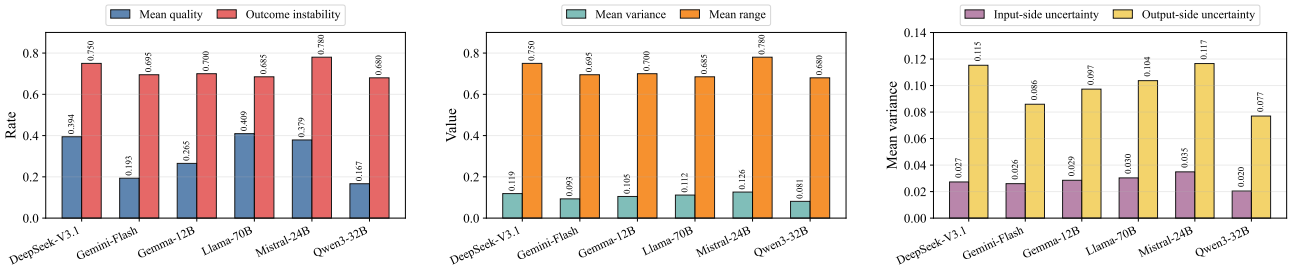
Observation 2: Query–model outcomes have broad distributions, not only occasional flips.

Figure 1b further reports the mean variance and mean prediction range of repeated observations for each model. The non-negligible variance shows that instability is not limited to a small number of outlier queries. Instead, many query–model pairs have dispersed outcome distributions under repeated evaluation. For binary-scored GPQA, the prediction range is closely related to correctness flipping: a large range indicates that the same query–model pair can move between failure and success across valid observations.

This result strengthens the limitation of single-shot supervision. A single response provides only one point sample from a broader outcome distribution. When the distribution has large variance or a large range, the sampled label can be highly sensitive to which input formulation or decoding run happens to be selected. Consequently, routing labels derived from single-shot outcomes may encode sampling artifacts rather than stable model capability.

Observation 3: Input-side and output-side uncertainty are both present but contribute differently.

Figure 1c decomposes the repeated-observation variance



(a) Mean quality vs. outcome flip rate. *(b)* Outcome variance and prediction range. *(c)* Decomposed input- and output-side uncertainty.

Figure 1. Diagnosing the limitations of single-shot routing supervision on GPQA. Each subfigure reports model-level statistics computed from repeated observations over semantically equivalent input reformulations and stochastic decoding samples. The results show that single-shot quality is not a stable characterization of query–model behavior: models exhibit high outcome flip rates, broad per-query outcome distributions, and non-negligible uncertainty from both input-side and output-side variation.

into input-side and output-side uncertainty. Across all models, output-side uncertainty is larger than input-side uncertainty, indicating that stochastic decoding under a fixed input is a major source of outcome variability on GPQA. At the same time, input-side uncertainty remains consistently nonzero, showing that semantically equivalent reformulations of the same query can also change model behavior. For example, models with relatively strong average quality still exhibit visible uncertainty from both sources, which suggests that scaling model capability alone does not eliminate supervision instability.

This decomposition shows why routing-label reliability cannot be assessed from a single axis alone. Evaluating only one prompt with repeated decoding captures output-side variability but misses sensitivity to input formulation. Evaluating multiple reformulations with deterministic decoding captures input-side variability but misses stochastic response instability. A reliable analysis of routing supervision should therefore account for both sides of uncertainty, since both can affect model correctness and the escalation labels used to train routers.

Observation 4: Single-shot label uncertainty propagates to substantial router-level instability.

To examine whether uncertainty in single-shot supervision affects the learned routing policy, we repeatedly sample one observation for each query–model pair from the 5×5 repeated-evaluation matrix and use these sampled labels to train routers under the same architecture, query representation, and train–test split. We construct 100 such random label sets and train one single-shot router from each set.

Figure 2 shows the resulting performance distribution on GPQA. Each scatter point corresponds to one independently trained router, while the violin, boxplot, and mean marker summarize the overall distribution. We report the average

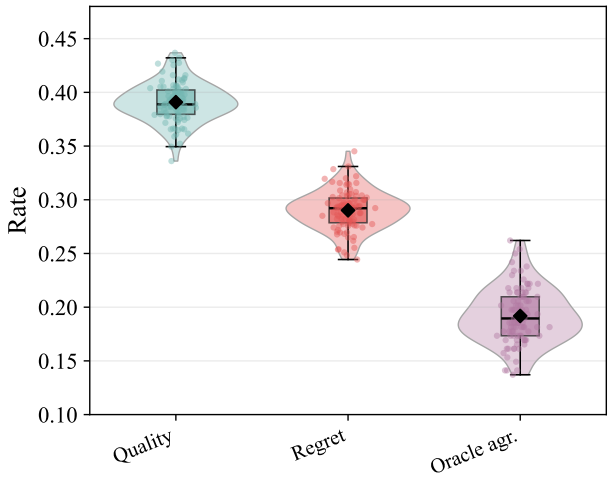


Figure 2. Performance distribution of single-shot routers trained from 100 random label samples on GPQA. Each point denotes one independently trained router. The violin plots show the distribution density, the boxplots indicate the median and interquartile range, and the black diamonds mark the mean. We report three metrics: routed quality (**Quality**), performance loss relative to the oracle router (**Regret**), and decision consistency with the oracle router (**Oracle agreement**). The broad spread across all three metrics shows that single-shot router performance is highly sensitive to label sampling and does not stably approach oracle behavior.

routed quality, the regret relative to the oracle router, and the agreement rate with oracle routing decisions. Overall, the training outcomes exhibit substantial variation across random label samples. Quality does not concentrate tightly around a single stable level, regret remains consistently nontrivial, and oracle agreement stays relatively low. These results indicate that the learned routing policy is highly sensitive to which single-shot labels happen to be observed during training.

This finding complements the model-level analysis above. The previous observations showed that single-shot query–

model outcomes are unstable under both input-side and output-side variation. Figure 2 shows that this instability does not remain at the level of individual labels; it propagates to the router itself. Even when the training pipeline is fixed, changing only the sampled supervision can produce noticeably different routing behaviors and overall performance. Therefore, evaluating a single-shot router with only one training run can be misleading. A more reliable evaluation should report the distribution of router performance across repeated label samples, rather than relying on a single point estimate.

3.4. Repeated-Observation Sampling for Routing Supervision

To analyze the reliability of single-shot routing supervision, we construct a repeated-observation protocol that samples model behavior along both the input and output dimensions. This protocol is not intended as a new router architecture; rather, it provides a controlled way to expose how much supervision can vary when the same query–model pair is evaluated under valid input reformulations and stochastic generations.

For each original query x_i, we first construct P semantically equivalent input reformulations:

$$\mathcal{X}_i = \left\{ x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(P)} \right\},$$

where each $x_i^{(p)}$ preserves the original task semantics, gold answer, and evaluation target. These reformulations are used to measure input-side uncertainty: if a model's behavior changes across $\mathcal{X}_i$, then its observed capability depends on the surface form of the input rather than only on the underlying task.

For each reformulated input $x_i^{(p)}$ and each candidate model $m \in \mathcal{M}$, we then perform D independent decoding runs:

$$a_{i,m}^{(p,d)} \sim P(a \mid x_i^{(p)}, m), \quad d = 1, \ldots, D.$$

Each sampled response induces a quality observation and a cost observation:

$$q_{i,m}^{(p,d)} = s(a_{i,m}^{(p,d)}, x_i), \quad c_{i,m}^{(p,d)} = \kappa(a_{i,m}^{(p,d)}, x_i^{(p)}, m).$$

This gives a repeated-observation matrix for each query–model pair:

$$\mathbf{Q}_{i,m} = \left[q_{i,m}^{(p,d)} \right]_{p=1,d=1}^{P,D}, \quad \mathbf{C}_{i,m} = \left[c_{i,m}^{(p,d)} \right]_{p=1,d=1}^{P,D}.$$

The matrix $\mathbf{Q}_{i,m}$ captures the distribution of observed model quality under input–output variation, while $\mathbf{C}_{i,m}$ captures the corresponding variation in inference cost. In this view, conventional single-shot supervision corresponds to observing only one entry from each matrix.

We use this repeated-observation matrix in two ways. First, for uncertainty analysis, we compute statistics such as the mean quality, outcome variance, prediction range, and correctness flip rate across all $P \times D$ observations. These statistics quantify whether a query–model pair has stable behavior or whether its observed quality depends strongly on the sampled input formulation or decoding run.

Second, to study how label uncertainty affects router training, we simulate single-shot supervision by randomly sampling one observation from each query–model matrix. For the b-th sampled supervision set, we draw

$$(p_{i,m}^{(b)}, d_{i,m}^{(b)}) \sim \mathrm{Unif}\left(\{1, \ldots, P\} \times \{1, \ldots, D\}\right),$$

and define the sampled single-shot observations as

$$q_{i,m}^{(b)} = q_{i,m}^{(p_{i,m}^{(b)}, d_{i,m}^{(b)})}, \quad c_{i,m}^{(b)} = c_{i,m}^{(p_{i,m}^{(b)}, d_{i,m}^{(b)})}.$$

Each sampled supervision set therefore represents one possible benchmark that could have been obtained if each query–model pair had been evaluated only once. By repeating this sampling process and training one router on each sampled label set, we can measure how sensitive the learned routing policy is to the particular single-shot observations used for supervision.

For comparison, we also define an aggregated observation for each query–model pair:

$$\bar{q}_{i,m} = \frac{1}{PD} \sum_{p=1}^{P} \sum_{d=1}^{D} q_{i,m}^{(p,d)}, \quad \bar{c}_{i,m} = \frac{1}{PD} \sum_{p=1}^{P} \sum_{d=1}^{D} c_{i,m}^{(p,d)}.$$

These aggregated quantities estimate the expected quality and cost of model m on query x_i under the sampled input–output variations. They serve as a diagnostic reference for asking whether repeated evaluation provides more stable routing supervision than a single sampled response.

4. Experiments

4.1. Experimental Setup

We evaluate whether repeated-evaluation supervision leads to more reliable routing behavior on the GPQA setting used in Section 3.3. GPQA is a scientific multiple-choice reasoning benchmark, where model correctness can be directly evaluated against the gold answer. The candidate model pool contains six heterogeneous LLMs with different capability and cost profiles. This setting allows us to examine whether supervision constructed from input–output observations improves routing behavior beyond conventional single-shot labels.

Training supervision. For each query–model pair in the training set, we use the $P \times D$ repeated-observation matrix

defined in Section 3.4. The *single-shot* setting samples one observation from this matrix and treats the sampled quality and cost as fixed supervision. The *aggregated* setting instead estimates the expected quality and cost by averaging over repeated observations across input reformulations and decoding samples. This comparison isolates the effect of supervision construction: the router architecture, query representation, model pool, and train–test split are kept fixed.

Test-time observation protocol. For evaluation, we construct two observation groups for each test query–model pair. First, we generate three semantically preserving reformulations of each test query to evaluate robustness under input-side variation. Second, we perform three stochastic decoding runs on the original query to evaluate robustness under output-side variation. We report two corresponding utilities:

$$U_{\text{in}} \quad \text{and} \quad U_{\text{out}},$$

where U_{in} denotes the utility measured over input reformulations, and U_{out} denotes the utility measured over repeated stochastic generations. The former tests whether the selected model remains effective when the same task is expressed in different forms, while the latter tests whether the selected model remains reliable under generation randomness.

Compared routers. We compare several representative routing methods from different learning paradigms. MLPRouter (Stripelis et al., 2024), MIRT (Song et al., 2025), and EmbedLLM (Zhuang et al., 2024) represent regression- or representation-based routing methods. RM-Softmax (Tsiourvas et al., 2025) and GraphRouter (Feng et al., 2025) represent classification- or graph-based routing methods. AvengersPro (Zhang et al., 2025) is included as a clustering-based method, and kNNRouter (Stripelis et al., 2024) is included as a non-parametric retrieval-based router. For each method, we compare training with single-shot supervision against training with repeated-evaluation aggregated supervision under the same cost–quality trade-off setting.

4.2. Experimental Results

Overall comparison. Table 1 reports the GPQA results. Across different router families, aggregated supervision improves routing utility in almost all cases. For example, GraphRouter improves from 0.462 to 0.496 under input-side evaluation and from 0.487 to 0.516 under output-side evaluation. MIRT also benefits substantially, improving from 0.409 to 0.444 on U_{in} and from 0.433 to 0.468 on U_{out}. These gains are not limited to a single routing architecture: retrieval-based, graph-based, regression-based, and classification-based routers all benefit from replacing single-shot labels with aggregated supervision.

Table 1. Comparison between single-shot supervision and repeated-evaluation aggregated supervision on GPQA. U_{in} evaluates utility under input reformulations, while U_{out} evaluates utility under repeated stochastic decoding.

Router	GPQA	
	U_{in}	U_{out}
MLP	0.382	0.400
+ Aggregated	0.399	0.413
kNN	0.441	0.469
+ Aggregated	0.461	0.489
EmbedLLM	0.469	0.496
+ Aggregated	0.473	0.480
AvengersPro	0.464	0.490
+ Aggregated	0.480	0.496
GraphRouter	0.462	0.487
+ Aggregated	**0.496**	**0.516**
MIRT	0.409	0.433
+ Aggregated	0.444	0.468
RM-Softmax	0.453	0.475
+ Aggregated	0.477	0.512

The only exception is EmbedLLM under output-side evaluation, where the aggregated variant decreases from 0.496 to 0.480. This suggests that repeated-evaluation supervision is not a guaranteed improvement for every router and metric. Nevertheless, the dominant trend is clear: routing labels estimated from repeated input–output observations provide more reliable supervision than labels sampled from a single response. This supports the central claim that part of routing instability originates from the supervision protocol rather than from the router architecture alone.

Ablation analysis. Table 2 studies which components of repeated-evaluation supervision are most useful. Starting from the single-shot baseline, using only input-side aggregation improves both U_{in} and U_{out}, indicating that query reformulations provide a more stable estimate of model behavior than one surface form alone. Using only output-side aggregation also improves over single-shot supervision, showing that repeated decoding helps reduce sampling noise in model outcomes.

The comparison between input-only and output-only variants suggests that the two sources of uncertainty are complementary. Input-only aggregation performs better on U_{out}, while output-only aggregation performs slightly better on U_{in}. The full variant, which combines both input-side and output-side observations, achieves the best result on both metrics, reaching 0.444 on U_{in} and 0.468 on U_{out}. This confirms that reliable routing supervision should not rely on

Table 2. Component ablation on GPQA. U_{in} evaluates input-side robustness, while U_{out} evaluates output-side robustness.

Variant	GPQA	
	U_{in}	U_{out}
Single-shot	0.409	0.433
Input-only aggregation	0.423	0.452
Output-only aggregation	0.425	0.442
Aggregation w/o risk	0.413	0.441
Aggregation w/ input risk	0.431	0.456
Aggregation w/ output risk	0.430	0.449
Full aggregation	**0.444**	**0.468**

only one axis of variation.

The variants with risk terms further show that uncertainty information can improve the quality of the supervision signal. Compared with aggregation without risk, both input-side and output-side risk variants yield better utilities, and the full variant performs best overall. This result is consistent with the analysis in Section 3.3: query–model behavior is affected by both input formulation and stochastic generation, and routing supervision becomes more reliable when both sources are accounted for.

Cost–quality trade-off under different cost constraints. Figure 3 compares the cost–quality trade-off on GPQA under different routing cost constraints. Aggregated supervision consistently shifts the cost–quality frontier upward compared with single-shot supervision across router families. This indicates that the benefit of repeated-evaluation supervision is not limited to one fixed cost threshold, but remains visible under both strict and relaxed budget settings. In several cost regions, routers trained with aggregated supervision also surpass the best fixed-model baseline, suggesting that improved supervision can help learned routers achieve a better deployment trade-off than always using a single strong model.

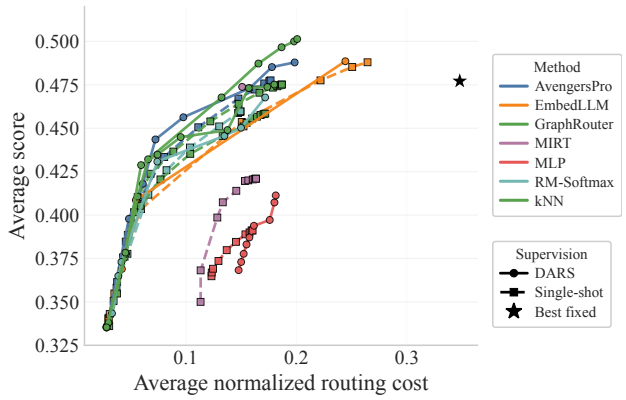


Figure 3. Cost–quality trade-off on GPQA under different cost constraints. Aggregated supervision improves the routing frontier over single-shot supervision, indicating more favorable cost–quality behavior across budget levels.

5. Conclusion and Future Work

This work studies input–output uncertainty in LLM routing supervision. Instead of treating single-shot query–model outcomes as deterministic labels, we view them as noisy observations affected by both input-side reformulations and output-side stochastic decoding. Through diagnostic analysis on GPQA, we show that these two sources of uncertainty can cause substantial correctness flips, induce unstable escalation labels, and propagate to learned routing policies, even when the router architecture, query representation, and train–test split are fixed. These findings suggest that the reliability of routing supervision is a critical but often overlooked factor in LLM routing systems. As a simple diagnostic remedy, we show that aggregating repeated evaluations provides more stable supervision and improves cost-adjusted routing behavior over single-shot labels. In future work, we plan to extend this analysis to broader agentic settings where routing decisions involve not only model selection but also tool invocation, query rewriting, response resampling, and multi-step reasoning modules. Another important direction is to develop more sample-efficient uncertainty estimation protocols, so that routing benchmarks can report label reliability without requiring expensive repeated evaluations for every query–model pair.

Acknowledgements

This work was supported by the National Key R&D Program of China under Grant 2024YFE0202800.

Impact Statement

This work aims to improve the reliability of LLM routing systems by highlighting uncertainty in the supervision labels used to train and evaluate routers. More reliable routing can reduce unnecessary use of expensive models and improve cost–quality trade-offs in agentic systems. At the same time, routing decisions may affect downstream access to model capability, latency, and service quality. If label uncertainty is ignored, routers may make unstable or biased allocation decisions, especially for queries whose outcomes are sensitive to input formulation or stochastic generation. We therefore encourage future routing benchmarks and deployments to report label reliability and evaluate routing policies under repeated observations, rather than relying solely on single-shot outcomes.

References

Aggarwal, P., Madaan, A., Anand, A., Potharaju, S. P., Mishra, S., Zhou, P., Gupta, A., Rajagopal, D., Kappaganthu, K., Yang, Y., et al. Automix: Automatically mixing language models. In *NeurIPS*, 2024.

Chen, L., Zaharia, M., and Zou, J. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

DeepSeek-AI et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Rühle, V., Lakshmanan, L. V. S., and Awadallah, A. H. Hybrid LLM: Cost-efficient and quality-aware query routing. In *ICLR*, 2024.

Ding, D., Mallick, A., Zhang, S., Wang, C., Madrigal, D., Garcia, M. D. C. H., Xia, M., Lakshmanan, L. V., Wu, Q., and Rühle, V. Best-route: Adaptive llm routing with test-time optimal compute. In *ICML*, 2025.

Farquhar, S., Kossen, J., Kuhn, L., and Gal, Y. Detecting hallucinations in large language models using semantic entropy. *Nature*, 2024.

Feng, T., Shen, Y., and You, J. Graphrouter: A graph-based router for llm selections. In *ICLR*, 2025.

Gan, C. and Mori, T. Sensitivity and robustness of large language models to prompt template in japanese text classification tasks. In *Proceedings of PACLIC*, 2023.

Google. Gemini 2.5 flash-lite. Gemini API documentation, 2025. URL https://ai.google.dev/gemini-api/docs/pricing.

Hu, Q. J., Bieker, J., Li, X., Jiang, N., Keigwin, B., Ranganath, G., Keutzer, K., and Upadhyay, S. K. Routerbench: A benchmark for multi-LLM routing system. In *Agentic Markets Workshop at ICML 2024*, 2024.

Huang, Z., Ling, G., Lin, Y., Chen, Y., Zhong, S., Wu, H., and Lin, L. Routereval: A comprehensive benchmark for routing llms to explore model-level scaling up in llms. In *EMNLP*, 2025.

Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Kassem, A. M., Schölkopf, B., and Jin, Z. How robust are router-llms? analysis of the fragility of llm routing capabilities. *arXiv preprint arXiv:2504.07113*, 2025.

Kuhn, L., Gal, Y., and Farquhar, S. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *ICLR*, 2023.

Lai, G. and Ye, H.-J. When routing collapses: On the degenerate convergence of llm routers. *arXiv preprint arXiv:2602.03478*, 2026.

Lai, G., Hu, H., and Ye, H.-J. Routejudge: Preference-based evaluation of LLM routers under pluralistic user preferences. In *Pluralistic Alignment Workshop at ICML 2026*, 2026.

Li, Y. Rethinking predictive modeling for llm routing: When simple knn beats complex learned routers. *arXiv preprint arXiv:2505.12601*, 2025.

Lin, Q., Ji, X., Zhai, S., Shen, Q., Zhang, Z., Fang, Y., and Gao, Y. Life-cycle routing vulnerabilities of llm router. *arXiv preprint arXiv:2503.08704*, 2025.

Ma, H., Lai, G., and Ye, H.-J. Mmr-bench: A comprehensive benchmark for multimodal llm routing. *arXiv preprint arXiv:2601.17814*, 2026.

Manakul, P., Liusie, A., and Gales, M. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of EMNLP*, 2023.

Mei, K., Xu, W., Lin, S., and Zhang, Y. Omnirouter: Budget and performance controllable multi-llm routing. *arXiv preprint arXiv:2502.20576*, 2025.

Meta AI. Llama 3.3 70b instruct. Model card, 2024. URL https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_3/.

Mistral AI. Mistral-small-3.2-24b-instruct-2506. Hugging Face model card, 2025. URL https://huggingface.co/mistralai/Mistral-Small-3.2-24B-Instruct-2506.

Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., and Stoica, I. Routellm: Learning to route llms from preference data. In *ICLR*, 2025.

Piskala, D. B., Raajaa, V., Mishra, S., and Bozza, B. Dynamic llm routing and selection based on user preferences: Balancing performance, cost, and ethics. *arXiv preprint arXiv:2502.16696*, 2025.

Ramírez, G., Lindemann, M., Birch, A., and Titov, I. Cache & distil: Optimising api calls to large language models. *arXiv preprint arXiv:2310.13561*, 2023.

Ramírez, G., Birch, A., and Titov, I. Optimising calls to large language models with uncertainty-based two-tier selection. *arXiv preprint arXiv:2405.02134*, 2024.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Shafran, A., Schuster, R., Ristenpart, T., and Shmatikov, V. Rerouting llm routers. *arXiv preprint arXiv:2501.01818*, 2025.

Shen, Y., Song, K., Tan, X., Li, D., Lu, W., and Zhuang, Y. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. In *NeurIPS*, 2023.

Shnitzer, T., Ou, A., Silva, M., Soule, K., Sun, Y., Solomon, J., Thompson, N., and Yurochkin, M. Llm routing with benchmark datasets. In *NeurIPS 2023 Workshop on Distribution Shifts: New Frontiers with Foundation Models*, 2023.

Song, W., Huang, Z., Cheng, C., Gao, W., Xu, B., Zhao, G., Wang, F., and Wu, R. IRT-router: Effective and interpretable multi-LLM routing via item response theory. In *ACL*, 2025.

Stripelis, D., Hu, Z., Zhang, J., Xu, Z., Shah, A. D., Jin, H., Yao, Y., Avestimehr, S., and He, C. Tensoropera router: A multi-model router for efficient llm inference. In *EMNLP*, 2024.

Team, G. Gemma 3 technical report, 2025.

Tsiourvas, A., Sun, W., and Perakis, G. Causal llm routing: End-to-end regret minimization from observational data. In *NeurIPS*, 2025.

Wang, X., Liu, Y., Cheng, W., Zhao, X., Chen, Z., Yu, W., Fu, Y., and Chen, H. Mixllm: Dynamic routing in mixed large language models. *arXiv preprint arXiv:2502.18482*, 2025.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Yang, J., Wu, Q., Feng, Z., Zhou, Z., Guo, D., and Chen, X. Quality-of-service aware llm routing for edge computing with multiple experts. *IEEE Transactions on Mobile Computing*, 2025b.

Yue, M., Zhao, J., Zhang, M., Du, L., and Yao, Z. Large language model cascades with mixture of thoughts representations for cost-efficient reasoning. *arXiv preprint arXiv:2310.03094*, 2023.

Zhang, J., Krishna, R., Awadallah, A. H., and Wang, C. Ecoassistant: Using llm assistant more affordably and accurately. *arXiv preprint arXiv:2310.03046*, 2023.

Zhang, Y., Li, H., Chen, J., Zhang, H., Ye, P., Bai, L., and Hu, S. Beyond gpt-5: Making llms cheaper and better via performance-efficiency optimized routing. In *DAI*, 2025.

Zhao, Z., Jin, S., and Mao, Z. M. Eagle: Efficient training-free router for multi-llm inference. *arXiv preprint arXiv:2409.15518*, 2024.

Zhu, K., Wang, J., Zhou, J., Wang, Z., Chen, H., Wang, Y., Yang, L., Ye, W., Zhang, Y., Gong, N. Z., and Xie, X. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. *arXiv preprint arXiv:2306.04528*, 2023.

Zhuang, R., Wu, T., Wen, Z., Li, A., Jiao, J., and Ramchandran, K. Embedllm: Learning compact representations of large language models. In *ICLR*, 2024.

Appendix

A. Dataset and Model Details

Dataset. We conduct the main diagnostic and routing experiments on GPQA, a graduate-level multiple-choice question answering benchmark covering challenging scientific domains such as biology, physics, and chemistry (Rein et al., 2023). GPQA is suitable for our study because its questions require advanced reasoning, while the multiple-choice format enables reliable automatic scoring of model correctness. This allows us to directly analyze how input-side and output-side variations affect query–model outcomes and the routing labels derived from them.

Data split. We construct a training split for learning routing policies and a held-out test split for evaluation. Following the diagnostic analysis in the main paper, we sample 200 GPQA training queries to construct repeated observations for supervision. The remaining held-out queries are used for test-time evaluation. All routers are trained and evaluated using the same split, so that performance differences come from the supervision construction protocol rather than from different data partitions.

Training observation protocol. For each training query, we construct repeated observations for every candidate model. To capture input-side variation, we use a strong language model to generate five semantically preserving rewrites of the original question. The rewriting process preserves the original task semantics, gold answer, and answer choices. To capture output-side stochasticity, each candidate model is evaluated with five independent decoding runs for each rewritten query. This gives a 5×5 observation matrix for every query–model pair, where each entry records the response quality and the corresponding inference cost. These repeated observations are used to compute model-level outcome statistics and to construct aggregated supervision signals for router training.

Single-shot training protocol. For the single-shot baseline, each query–model pair is represented by one sampled observation. To reduce the effect of a particular random draw, we construct 100 independently sampled single-shot training sets. Each single-shot training set contains one randomly selected observation for every query–model pair, and one router is trained from each sampled set. This protocol allows us to measure how much learned routing behavior varies solely due to random supervision-label sampling.

Test-time evaluation protocol. At test time, the router receives only the original query and selects one model from the candidate pool. Repeated test observations are used only for evaluation and are not provided to the router. For each test query, we construct two evaluation groups. The first group contains three semantically preserving rewrites of the query, which is used to compute the input-side utility U_{in}. The second group contains three independent stochastic decoding runs on the original query, which is used to compute the output-side utility U_{out}. The former measures whether the routing decision is robust to input-side variation, while the latter measures whether the selected model remains reliable under output-side randomness.

Model pool. We use a heterogeneous pool of six LLMs with different capability and cost profiles: Gemma-3-12B-IT (Team, 2025), Mistral-Small-3.2-24B-Instruct (Mistral AI, 2025), Qwen3-32B (Yang et al., 2025a), Llama-3.3-70B-Instruct (Meta AI, 2024), Gemini-2.5-Flash-Lite (Google, 2025), and DeepSeek-Chat-V3.1 (DeepSeek-AI et al., 2024). This model pool contains both open-source and API-based models, as well as models with different sizes and deployment costs. Such heterogeneity is important for routing evaluation, since the router must learn when a cheaper model is sufficient and when a stronger model is worth the additional cost.

Evaluation metric and cost setting. For each generated response, we compute a task-specific quality score and an inference cost. For GPQA, quality is measured by multiple-choice accuracy after parsing the model's final answer. Inference cost is normalized before computing routing utilities. Unless otherwise specified, we use the cost coefficient $\lambda = 0.05$ in the main experiments.

B. Query Rewriting and Decoding Protocol

Query rewriting. For each GPQA query, we generate semantically preserving rewrites using a strong language model. The purpose of rewriting is to introduce controlled input-side variation while keeping the underlying task unchanged. Only

the question stem is rewritten, while the answer choices are kept unchanged. The rewritten question must preserve the same gold answer and must not add hints, remove constraints, or change the intended reasoning target.

GPQA rewriting instruction

This is a graduate-level multiple-choice science question. Rewrite only the question stem. Do not modify, reorder, remove, or paraphrase the answer choices. The correct option after rewriting must remain exactly the same. Do not add hints, simplify the question, or reveal the answer.

Prompt format for model generation. After obtaining rewritten queries, we construct prompts for candidate models using the rewritten question and the original answer choices. All prompts require the model to output a clearly marked `Final Answer:` line, so that responses can be parsed and evaluated consistently.

Shared system instruction for generation

You are answering benchmark questions for an LLM routing experiment. Follow the user instructions exactly. You may reason internally, but the final response must contain a clearly marked `Final Answer:` line. Do not mention that this is a paraphrase or dataset example.

Training decoding protocol. For each training query, we use $P = 5$ rewritten variants. For every rewritten variant and every candidate model, we perform $D = 5$ independent stochastic decoding runs. This produces 5×5 observations for each query–model pair. Each observation records the generated response, parsed answer, task-specific quality score, token usage, and inference cost. These observations are then used to compute single-shot samples, aggregated quality estimates, and uncertainty statistics.

Decoding configuration. All model generations are collected through the same decoding interface using the same sampling configuration unless otherwise specified. We use temperature 0.7 and top-p 0.95 to induce stochastic generations. The maximum generation length is capped at 768 tokens for GPQA. Independent decoding samples are obtained by issuing separate generation calls with the same prompt and decoding configuration.

Test-time decoding protocol. At test time, the router receives only the original query and selects a candidate model. The repeated test observations are used only for evaluation and are not provided to the router. For each selected model, we compute two evaluation views. The input-side utility U_{in} is evaluated using three semantically preserving rewrites of the test query, while the output-side utility U_{out} is evaluated using three stochastic decoding runs on the original query. This protocol allows us to separately evaluate robustness to input-side query variation and output-side generation randomness.

C. Baseline Implementation Details

This section provides implementation details for the routing baselines used in our experiments. All baselines are implemented in the same codebase and share the same data loader, query feature extraction pipeline, candidate model pool, supervision construction protocol, and evaluation procedure unless otherwise specified.

Common setup. All methods are evaluated on GPQA using the same six-model candidate pool described in Appendix A. Each method reads scored generations from the same training and test files. For query representation, we use one feature vector per query. By default, the feature text contains the question and answer choices. Neural and retrieval-based methods use the same query encoder interface. When a local sentence-transformer encoder is available, we use it to encode queries; otherwise, we fall back to TF-IDF features.

Single-shot and aggregated supervision. Each baseline is evaluated under two supervision modes. In the single-shot setting, each query–model pair is represented by one sampled scored observation. We repeat this process with 100 independently sampled single-shot training sets to estimate the variance induced by label sampling. In the aggregated setting, the router architecture is kept unchanged, but the supervision signal is replaced by statistics aggregated from repeated observations. For each query–model pair, we compute the mean score, mean cost, and score variability from the 5×5 observation matrix.

Table 3. Summary of baseline implementations and how repeated-observation supervision is incorporated.

Router	Learning paradigm	Aggregated supervision
MLP	Regression-based routing	Predicts mean score, mean cost, and score variability.
MIRT	IRT-style regression	Learns score, cost, and uncertainty through separate IRT-style heads.
EmbedLLM	Model-embedding regression	Uses aggregated performance targets with learned model embeddings.
RM-Softmax	Classification / regret minimization	Trains on utilities computed from aggregated observations.
GraphRouter	Query–model edge prediction	Predicts edge-level score, cost, and uncertainty for query–model pairs.
kNNRouter	Non-parametric retrieval	Aggregates targets from nearest training queries using repeated-observation statistics.
AvengersPro	Cluster-based routing	Aggregates repeated-observation statistics at the query-cluster level.

Regression-based routers. **MLP** trains separate regressors for model score, model cost, and, under aggregated supervision, score variability. Each regressor predicts a vector over the candidate models for a given query representation, and the router selects the model with the largest predicted utility.

MIRT implements a multi-dimensional item-response-theory router. It maps each query into a latent ability vector and learns model-specific discrimination and difficulty parameters. Separate prediction heads are used for score, cost, and score variability.

EmbedLLM learns compact model embeddings and a query projection to predict query–model performance. It jointly trains a performance head and a model-cost head. Under aggregated supervision, the performance target is estimated from repeated observations rather than from a single sampled response.

Classification-based routers. **RM-Softmax** implements a regret-minimization router. It augments each query embedding with a cost-weight value and trains a neural classifier over candidate models across a grid of cost coefficients. Under aggregated supervision, model preferences are computed from repeated-observation utilities.

GraphRouter treats each query–model pair as an edge between a query node and a model node. Query features and model descriptions are projected into a shared space, and an edge predictor estimates model-wise score, cost, and uncertainty.

Non-parametric and cluster-based routers. **kNNRouter** is a non-parametric retrieval-based router. It stores training query embeddings and their query–model target matrices. For a test query, it retrieves the nearest training queries and averages their model-wise score, cost, and uncertainty statistics using inverse-distance weights.

AvengersPro is implemented as a cluster-based router. Training queries are clustered in feature space, and each cluster stores the average score, cost, and uncertainty for each candidate model. For a new query, the router retrieves the nearest clusters and estimates model utility by aggregating their stored statistics.

D. Diagnostic Metrics

This section provides additional details on the diagnostic analysis in Section 3.3. The goal is to examine whether the standard single-shot supervision protocol provides reliable routing labels. We focus on three aspects: whether the observed outcome of a query–model pair is stable, whether the best model selected for a query is stable, and whether single-shot routing labels agree with labels estimated from repeated observations.

For each query x_i and candidate model $m \in \mathcal{M}$, let

$$\mathcal{O}_{i,m} = \{(q_{i,m}^{(p,d)}, c_{i,m}^{(p,d)})\}_{p=1,d=1}^{P,D}$$

denote the observations of model m on query x_i, where p indexes rewritten query variants and d indexes independent decoding runs. Here, $q_{i,m}^{(p,d)}$ is the task-specific quality score and $c_{i,m}^{(p,d)}$ is the corresponding normalized inference cost. The per-observation utility is

$$u_{i,m}^{(p,d)} = q_{i,m}^{(p,d)} - \lambda c_{i,m}^{(p,d)}.$$

Outcome instability. Outcome instability measures whether repeated observations of the same query–model pair lead to different observed performance scores. For each query–model pair, we check whether the quality scores vary across

repeated observations:

$$\mathrm{Instab}_{\mathrm{out}}(i,m) = \mathbb{I}\left[\max_{p,d} q_{i,m}^{(p,d)} - \min_{p,d} q_{i,m}^{(p,d)} > \epsilon\right],$$

where ϵ is a small tolerance used to avoid numerical artifacts. The overall outcome instability is computed by averaging this indicator over all queries and candidate models:

$$\mathrm{OutcomeInstability} = \frac{1}{|\mathcal{D}||\mathcal{M}|}\sum_{i=1}^{|\mathcal{D}|}\sum_{m\in\mathcal{M}} \mathrm{Instab}_{\mathrm{out}}(i,m).$$

A higher value means that the same model can receive different scores on the same underlying query under natural input or output variations.

Winner flip rate. Outcome instability at the query–model level does not necessarily imply that the routing label changes. Therefore, we further measure whether the model selected as the best candidate for a query changes across repeated single-shot observations. For each observation index (p,d), we construct a single-shot routing label by selecting the model with the largest observed utility:

$$w_i^{(p,d)} = \arg\max_{m\in\mathcal{M}} u_{i,m}^{(p,d)}.$$

The winner flip indicator for query x_i is then defined as

$$\mathrm{Flip}(i) = \mathbb{I}\left[\left|\{w_i^{(p,d)} : 1 \le p \le P, 1 \le d \le D\}\right| > 1\right].$$

The winner flip rate is the average of this indicator over all queries:

$$\mathrm{WinnerFlipRate} = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\mathrm{Flip}(i).$$

This metric captures whether different single-shot samples can lead to different best-model labels for the same query.

Single-shot versus aggregated-label disagreement. We also compare the single-shot routing label with an aggregated label estimated from all repeated observations. For each query–model pair, we first compute the expected quality and expected cost:

$$\mu_q(i,m) = \frac{1}{PD}\sum_{p=1}^{P}\sum_{d=1}^{D} q_{i,m}^{(p,d)}, \quad \mu_c(i,m) = \frac{1}{PD}\sum_{p=1}^{P}\sum_{d=1}^{D} c_{i,m}^{(p,d)}.$$

The aggregated utility is

$$\bar{u}_{i,m} = \mu_q(i,m) - \lambda\mu_c(i,m),$$

and the corresponding aggregated routing label is

$$w_i^{\mathrm{agg}} = \arg\max_{m\in\mathcal{M}} \bar{u}_{i,m}.$$

For each single-shot observation (p,d), we compute whether its selected model disagrees with the aggregated label:

$$\mathrm{Disagree}^{(p,d)}(i) = \mathbb{I}\left[w_i^{(p,d)} \neq w_i^{\mathrm{agg}}\right].$$

The final disagreement rate is averaged over all queries and single-shot observations:

$$\mathrm{Disagreement} = \frac{1}{|\mathcal{D}|PD}\sum_{i=1}^{|\mathcal{D}|}\sum_{p=1}^{P}\sum_{d=1}^{D}\mathrm{Disagree}^{(p,d)}(i).$$

This metric measures how often a routing label constructed from one sampled observation differs from the label obtained by aggregating repeated observations.

Table 4. Automatic validity check for GPQA query rewrites. Complete denotes the fraction of original queries for which all rewrites are generated. Original duplicate and rewrite duplicate denote duplicate rates with the original query and among rewrites, respectively. Structural pass checks whether answer choices are preserved. Review flag marks rewrites selected for further inspection due to low similarity or numeric mismatch.

Split	Records	Rewrites	Complete	Orig. dup.	Rew. dup.	Sim. mean	Sim. p05	Struct. pass	Review flag
Train	200	1000	1.0000	0.0000	0.0000	0.8723	0.6267	1.0000	0.2910
Test	248	1240	1.0000	0.0000	0.0000	0.8815	0.6656	1.0000	0.3016

Table 5. Validity check for the automatic GPQA scoring pipeline. Parse success measures whether a generated response can be parsed into the expected answer format. Finite and in $[0, 1]$ check numerical validity. Score type checks whether the scorer returns the expected score type. Rescore agreement measures agreement between stored scores and scores recomputed by the unified scorer.

Split	Rows	Parse success	Finite	In $[0, 1]$	Score type	Rescore agreement	Rescore diff.	Duplicate keys
Train	30000	0.9996	1.0000	1.0000	1.0000	0.9998	7	0
Test	8928	0.9999	1.0000	1.0000	1.0000	0.9998	2	0

Router variance under single-shot supervision. Finally, we examine whether the instability of single-shot labels propagates to the learned router. We construct multiple single-shot training sets by randomly sampling one observation for each query–model pair. Each sampled training set is used to train the same router architecture under the same data split and model pool. Since the only changing factor is the sampled supervision signal, the variation in test utility reflects the sensitivity of learned routing policies to single-shot supervision noise.

E. Validity Checks

Our analysis relies on two preprocessing components: query rewriting for constructing input-side variations and automatic scoring for evaluating repeated model outputs. We therefore conduct additional validity checks to examine whether these components introduce obvious artifacts. These checks are intended as lightweight sanity checks rather than formal guarantees of semantic equivalence or scoring correctness.

E.1. Rewrite Validity Check

The purpose of query rewriting is to introduce controlled input-side variations while preserving the underlying task semantics. To assess the quality of the generated rewrites, we perform an automatic validity check over all rewritten GPQA queries used in our experiments. The check covers five aspects: (1) whether all required rewrites are successfully generated, (2) whether rewritten queries duplicate the original query or duplicate each other, (3) whether answer choices are preserved, (4) whether the rewritten query remains close to the original query in sentence-embedding space, and (5) whether numeric tokens are preserved. Sentence similarity is computed using the same sentence-transformer backend as in our query representation pipeline. We additionally flag rewrites with low semantic similarity or numeric-token mismatch for manual inspection. These flags are conservative: a flagged rewrite is not necessarily invalid, since scientific questions often contain symbols, quantities, abbreviations, and domain-specific expressions whose surface forms may change under paraphrasing.

The results show that GPQA achieves full rewrite completion, no duplicate rewrites are detected, and all rewrites pass the structural checks. Sentence similarity is also high on average. The review-flag rate is mainly caused by numeric-token or symbol changes, which are common in scientific questions and are therefore treated as conservative diagnostic flags rather than direct evidence of invalid rewrites.

E.2. Scoring Validity Check

We also verify the automatic scoring pipeline used to evaluate generated responses. For each stored generation, we recompute the score using the unified GPQA scoring script and check whether the output can be parsed, whether the score is finite, whether the score lies in the valid range $[0, 1]$, whether the expected score type is produced, whether rescoring agrees with the stored score, and whether duplicate generation keys exist.

The scoring pipeline achieves near-perfect parse coverage on GPQA, and all parsed scores are finite and lie within the valid range. We also find no duplicate generation keys. The small number of rescoring differences is caused by scorer-version

inconsistencies in intermediate stored files. To avoid propagating such inconsistencies, all reported experimental results are computed after applying the same unified scoring pipeline to the stored generations.