Priors in LLM Routing: From Initialization to Regularization

Anonymous authors
Paper under double-blind review

Abstract

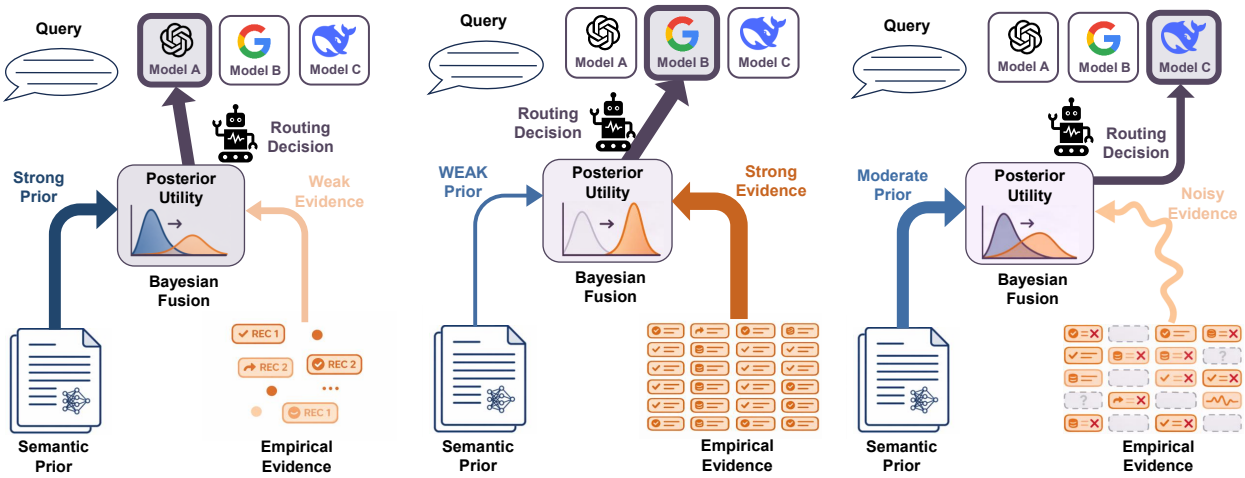
LLM routing is commonly treated as a data-driven utility estimation problem, where routing decisions are learned primarily from historical interactions. In this work, we argue that such a likelihood-dominant view is fundamentally incomplete. The key challenge in LLM routing is not only how to fit utility from data, but also how to make stable decisions when available evidence is insufficient or unreliable. In these regimes, prior information is indispensable: it initializes utility estimation when observations are scarce, and regularizes it when feedback is noisy or incomplete. Based on this insight, we propose BayesRouter, a lightweight posterior utility inference framework that integrates semantic priors from standardized model descriptions with empirical evidence from retrieved historical interactions via closed-form Bayesian fusion. Experiments on multiple public benchmarks demonstrate that BayesRouter achieves a better accuracy–cost trade-off than strong routing baselines, improves performance in low-data settings, and exhibits stronger robustness under noisy and missing supervision. Our results show that priors are not merely auxiliary in LLM routing, but a key ingredient for stable utility estimation.

1 Introduction

Large Language Models (LLMs) have been widely adopted in real-world applications and have demonstrated strong capabilities across a wide range of tasks (Narayanan Venkit et al., 2025; Li et al., 2025; Kim et al., 2024). However, always invoking the strongest model for every query incurs substantial computational cost and latency, which limits the scalability and deployment efficiency of LLM systems (Bang, 2023; Feng et al., 2025; Chen & Varoquaux, 2024). To address this issue, LLM routing has emerged as an effective system-level solution that dynamically assigns each query to an appropriate model from a candidate pool (Chen et al., 2023; Ong et al., 2025; Shen et al., 2023). In essence, routing seeks to optimize the trade-off between response quality and inference cost by estimating which model is most suitable for each input. In this way, effective routing can substantially reduce average cost and latency while preserving overall system performance (Srivatsa et al., 2024; Behera et al., 2025; Aggarwal et al., 2024).

Existing LLM routers typically rely on historical interactions to estimate the utility of candidate models, or equivalently, to predict which model is likely to perform best for a given query. While differing in architecture and supervision signals, these methods often share an implicit assumption: past observations provide sufficiently informative and reliable evidence for utility estimation. Under this assumption, routing can be treated largely as an empirical prediction problem. However, such an assumption is often fragile in practice. In realistic deployment scenarios, the available evidence may be limited, noisy, incomplete, or even mismatched with the current query distribution. As a result, routers that rely heavily on empirical observations can become unstable under scarce evidence and brittle under imperfect feedback.

From a Bayesian perspective, this issue can be understood more clearly. Since routing ultimately aims to choose the model with the highest expected utility, the key quantity of interest is the *posterior utility* of each model for a given query. However, most existing routers estimate this quantity almost entirely from observed evidence, making them effectively *likelihood-dominant* estimators of posterior utility. In other words, they emphasize what can be inferred from historical interactions, while largely overlooking what should be assumed *a priori* before evidence becomes sufficient or trustworthy. This viewpoint suggests that the limitation of



(a) Prior provides initialization (b) Likelihood dominates (c) Prior provides regularization

Figure 1: Three evidence regimes in LLM routing. With scarce evidence, priors provide initialization; with abundant reliable evidence, the likelihood dominates; with noisy or incomplete evidence, priors provide regularization. Together, these regimes show that priors play complementary roles in robust routing.

current routers is not merely algorithmic, but statistical: they underutilize the role of prior information in posterior utility estimation.

Figure 1 illustrates this perspective through three evidence regimes in LLM routing. Figure 1(b) corresponds to the idealized regime implicitly assumed by many existing methods: when empirical evidence is abundant and reliable, the likelihood dominates posterior estimation, and routing decisions can indeed be driven primarily by historical observations. However, practical routing often departs from this ideal case. In Figure 1(a), only weak evidence is available, such as in cold-start settings or for rarely seen queries. In this regime, empirical estimates are high-variance and poorly initialized, so semantic priors provide a necessary starting point for utility estimation. In Figure 1(c), evidence is available but noisy, incomplete, or otherwise unreliable. Here, directly trusting the likelihood can lead to overfitting spurious observations, whereas priors help stabilize posterior estimation against corrupted evidence. Together, these three regimes show that prior information is not merely an auxiliary feature, but a fundamental component of robust routing.

This leads to a simple but important insight: priors play two complementary statistical roles in LLM routing. When evidence is insufficient, priors provide *initialization*; when evidence is unreliable, priors provide *regularization*. Motivated by this view, we propose **BayesRouter**, a lightweight Bayesian routing framework that estimates posterior utility by fusing a semantic prior with an experiential likelihood. Specifically, BayesRouter derives model-side priors from query–model semantic descriptions and constructs likelihoods from retrieved historical interactions, then combines them through closed-form Bayesian fusion to obtain uncertainty-aware utility estimates for cost-sensitive routing. This design naturally emphasizes priors when evidence is scarce, allows likelihoods to dominate when feedback is abundant and reliable, and remains robust when observations are noisy or missing.

Our main contributions are as follows:

- We revisit LLM routing from a Bayesian perspective and show that many existing routers are effectively likelihood-dominant posterior utility estimators.
- We identify two complementary statistical roles of priors in routing: initialization under insufficient evidence and regularization under unreliable evidence.
- We propose **BayesRouter**, a lightweight Bayesian routing framework that fuses semantic priors with experiential likelihoods for uncertainty-aware and cost-sensitive routing.

- We conduct extensive experiments on multiple benchmarks and show that BayesRouter achieves strong accuracy–cost trade-offs, especially under cold-start, and noisy-feedback settings.

2 Related Work

LLM routing has recently emerged as a promising paradigm for improving inference efficiency without sacrificing task performance (Varangot-Reille et al., 2025; Srivatsa et al., 2024). Most existing methods adopt a learning-based approach, training a router to directly infer model performance, utility, or routing decisions from the input, and have shown strong empirical results across a wide range of benchmarks and deployment settings (Ong et al., 2025; Shen et al., 2023).

Heuristic-based routing. Early LLM routing methods and some practical systems rely on heuristic or rule-based criteria to decide whether a query should be handled by a stronger or weaker model. A common strategy is to route based on confidence or uncertainty estimates, where the router assesses the reliability of a model's response and falls back to a stronger model when the estimated confidence is low (Zhang et al., 2025a; Shnitzer et al., 2023).

Learning-based routing. Most recent work formulates LLM routing as a learning problem, where a router is trained to predict model performance, utility, or routing decisions directly from the input (Ong et al., 2025; Shen et al., 2023). This paradigm has been extended to multi-model settings using a variety of lightweight predictors, including KNN, MLPs, linear models, and graph neural networks (Stripelis et al., 2024; Hari & Thomson, 2023; Wang et al., 2025; Feng et al., 2025). Other methods learn shared representations of model behaviors or input–output patterns, for example via clustering or contrastive learning, to support more fine-grained routing decisions (Jitkrittum et al., 2025; Zhang et al., 2025c; Chen et al., 2024). In addition, several works employ open-source LLMs themselves as routers (Lai & Ye, 2026; Mohammadshahi et al., 2024; Zhang et al., 2025b; Wang et al., 2026).

While these methods have demonstrated strong empirical performance, they still primarily rely on abundant and clean supervision to learn reliable scoring functions, and generally do not explicitly model posterior uncertainty or incorporate priors into utility estimation, which limits their robustness under cold-start, noisy, or incomplete feedback conditions.

3 Preliminary

3.1 Problem Definition

Let $\mathcal{M} = \{1, \dots, M\}$ denote a pool of M candidate LLMs. Given a query q, LLM routing aims to select the model with the highest cost-adjusted utility:

$$m^\star = \arg\max_{m \in \mathcal{M}} S(q, m), \tag{1}$$

where

$$S(q, m) = \mu(q, m) - \lambda \bar{c}_m. \tag{2}$$

Here, $\mu(q, m)$ denotes the target utility of model m on query q, $\bar{c}_m$ is the normalized inference cost of model m, and $\lambda \geq 0$ controls the utility–cost trade-off.

In conventional routing, $\mu(q, m)$ is typically estimated from a historical dataset with fully observed utility vectors:

$$\mathcal{D} = \{(q_i, \mathbf{y}_i)\}_{i=1}^n, \tag{3}$$

where $\mathbf{y}_i = (y_{i,1}, \dots, y_{i,M})^\top \in \mathbb{R}^M$ and $y_{i,m}$ denotes the utility of model m on query q_i.

In this work, we consider a more realistic setting where historical evidence is imperfect. Specifically, the router is given a support set

$$\mathcal{S} = \{(q_i, \tilde{\mathbf{y}}_i, \mathbf{a}_i)\}_{i=1}^n, \tag{4}$$

where $\tilde{\mathbf{y}}_i = (\tilde{y}_{i,1}, \ldots, \tilde{y}_{i,M})^\top \in (\mathbb{R} \cup \{\text{NaN}\})^M$ is the observed utility vector and $\mathbf{a}_i = (a_{i,1}, \ldots, a_{i,M})^\top \in \{0,1\}^M$ is the observation mask. Here, $a_{i,m} = 1$ indicates that $\tilde{y}_{i,m}$ is observed, while $a_{i,m} = 0$ means the corresponding entry is missing. Moreover, $\tilde{y}_{i,m}$ may be noisy, and should therefore be viewed as an imperfect observation of the latent utility $y_{i,m}$.

Accordingly, the problem studied in this work is to estimate $\mu(q, m)$ for each candidate model from imperfect historical evidence and make routing decisions using the cost-adjusted score above.

3.2 Bayesian Modeling of LLM Routing

Under the setting defined above, estimating $\mu(q, m)$ solely from historical interactions is fundamentally challenging. The support set $\mathcal{S}$ may contain missing entries and noisy observations, so the empirical evidence associated with a query–model pair can be sparse or unreliable. As a result, directly learning $\mu(q, m)$ from $\mathcal{S}$ alone may lead to unstable estimates. To alleviate this issue, we introduce *external knowledge* as an additional source of information for routing.

For a query q and a candidate model m, let $Y_{q,m}$ denote the latent utility of model m on query q. We assume that, besides the imperfect historical evidence, there exists an external semantic signal $z_{q,m}$ that reflects prior knowledge about the suitability of model m for query q. Such a signal may come from query–model semantic matching, model capability descriptions, or other side information independent of observed utilities. From a Bayesian perspective, this signal naturally induces a prior distribution over the latent utility: $p(Y_{q,m} \mid z_{q,m})$.

Meanwhile, the historical interactions in $\mathcal{S}$ provide empirical evidence about $Y_{q,m}$. Let $\mathcal{E}_{q,m}$ denote the evidence extracted from $\mathcal{S}$ that is relevant to query q and model m. This evidence may consist of observed utilities from similar historical queries, and is generally incomplete or noisy under our setting. We model its contribution through the likelihood $p(\mathcal{E}_{q,m} \mid Y_{q,m})$.

Combining prior knowledge and historical evidence, the utility of model m on query q can be inferred through the posterior distribution

$$p(Y_{q,m} \mid \mathcal{E}_{q,m}, z_{q,m}) \propto p(\mathcal{E}_{q,m} \mid Y_{q,m})\, p(Y_{q,m} \mid z_{q,m}). \tag{5}$$

Accordingly, we instantiate the target utility in Equation (2) by the posterior expectation

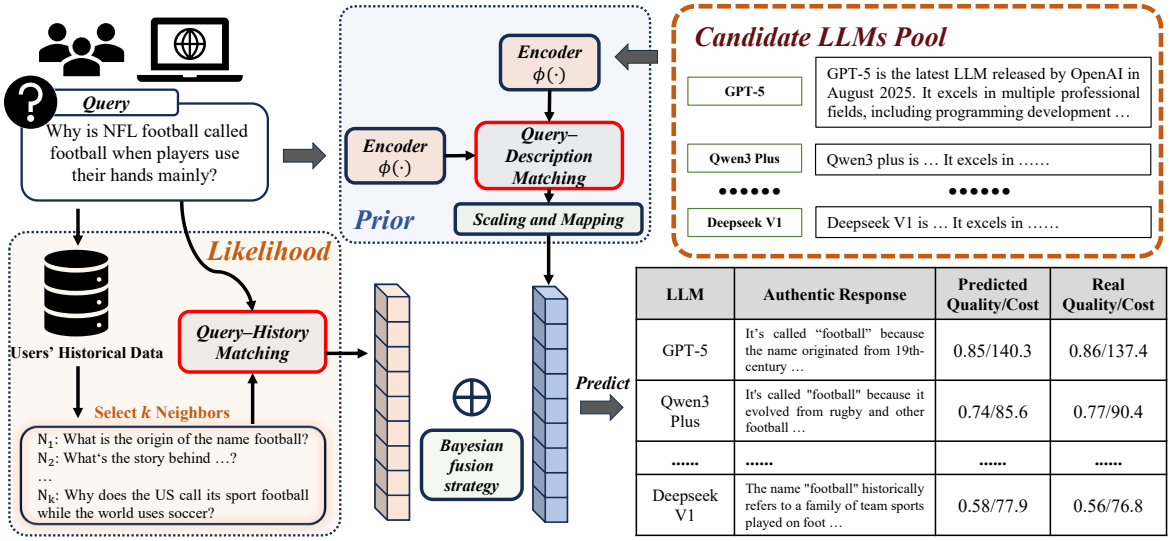
$$\mu(q, m) = \mathbb{E}[Y_{q,m} \mid \mathcal{E}_{q,m}, z_{q,m}]\,. \tag{6}$$

The routing decision is then made using the same cost-adjusted rule defined in Equation (2).

This formulation also clarifies the limitation of many existing routing methods. Most of them estimate model utility using historical interactions alone, which is equivalent to relying only on the likelihood term $p(\mathcal{E}_{q,m} \mid Y_{q,m})$ while ignoring the prior information carried by $z_{q,m}$. In contrast, our formulation treats routing as utility estimation under imperfect evidence, where external knowledge provides a prior and historical interactions provide a likelihood. This Bayesian view will serve as the basis of our method in the following section.

4 BayesRouter: Prior–Likelihood Fusion for Cost-Aware LLM Routing

Motivated by the Bayesian formulation in Section 3.2, we propose **BayesRouter**, a routing framework that estimates model utility by fusing external semantic knowledge with historical interaction evidence. Given a user query and a set of candidate LLMs, BayesRouter first computes a semantic prior for each model via query–description compatibility scoring. It then retrieves the K nearest neighbors from historical interaction data to construct an experiential likelihood, yielding data-driven utility estimates together with uncertainty. Finally, the prior and likelihood are combined through Gaussian-conjugate Bayesian inference to produce posterior utility predictions, which are used for cost-aware routing.



LLM	Authentic Response	Predicted Quality/Cost	Real Quality/Cost
GPT-5	It's called "football" because the name originated from 19th-century …	0.85/140.3	0.86/137.4
Qwen3 Plus	It's called "football" because it evolved from rugby and other football …	0.74/85.6	0.77/90.4
……	……	……	……
Deepseek V1	The name "football" historically refers to a family of team sports played on foot …	0.58/77.9	0.56/76.8

Figure 2: Workflow of the proposed BayesRouter framework. Given a user query and candidate LLMs, BayesRouter constructs semantic priors via query–description compatibility scoring, estimates experiential likelihoods from retrieved historical neighbors, and then fuses the two through Bayesian inference to obtain posterior performance estimates.

4.1 Semantic Priors from Model Descriptions

As discussed in Section 3.2, robust routing under imperfect evidence requires information beyond historical interactions alone. In BayesRouter, we instantiate such external knowledge as a *semantic prior* derived from model descriptions.

Model descriptions are a natural source of prior information for three reasons. First, they are available before any query-specific interaction data are collected, making them particularly suitable for cold-start routing. Second, they often summarize model properties that are directly relevant to routing, such as modality support, training scope, architectural characteristics, and intended use cases. Third, unlike historical utilities, which may be sparse or noisy, textual descriptions provide a relatively stable and model-level signal that is independent of the current support set. Therefore, although such descriptions do not determine routing decisions by themselves, they offer a principled basis for forming initial beliefs about model suitability.

Concretely, assume each candidate model m is associated with a textual description d_m, obtained from publicly available metadata such as model cards on platforms like HuggingFace (Jain, 2022). Since these descriptions are written by different development teams and may vary substantially in style, granularity, and verbosity, we first standardize them into concise capability summaries. **Following the prompt in Section B**, we use a dedicated auxiliary LLM that is *not* included in the routing candidate pool to rewrite each raw description into a unified format, yielding a standardized description $\tilde{d}_m$. This standardization step reduces stylistic variation across models and improves the comparability of semantic priors in a heterogeneous model pool.

We then encode the standardized model descriptions and the user query into vector representations: $\mathbf{g}_m = \phi_{\text{desc}}(\tilde{d}_m) \in \mathbb{R}^{d_{\text{desc}}}, \mathbf{x}_q = \phi_{\text{query}}(q) \in \mathbb{R}^d$, where $\phi_{\text{desc}}(\cdot)$ and $\phi_{\text{query}}(\cdot)$ denote the description encoder and query encoder, respectively.

To measure the semantic compatibility between query q and model m, we project the two embeddings into a shared projection space: $\mathbf{q}_{\text{proj}} = W_Q \mathbf{x}_q + \mathbf{b}_Q, \mathbf{d}_{\text{proj},m} = W_D \mathbf{g}_m + \mathbf{b}_D$, where $W_Q \in \mathbb{R}^{d_{\text{proj}} \times d}$ and $W_D \in \mathbb{R}^{d_{\text{proj}} \times d_{\text{desc}}}$ are learnable projection matrices, and $\mathbf{b}_Q, \mathbf{b}_D \in \mathbb{R}^{d_{\text{proj}}}$ are bias terms.

The resulting prior similarity score is computed as

$$s_{q,m}^{(\text{prior})} = \frac{\mathbf{q}_{\text{proj}}^{\top} \mathbf{d}_{\text{proj},m}}{\sqrt{d_{\text{proj}}}}. \tag{7}$$

This module is used only to produce a scalar compatibility score between the query and the model description, rather than a query-conditioned aggregation over description representations.

We map this score to the prior mean utility through a scaled sigmoid transformation:

$$\mu_{0,q,m} = \sigma\left(\beta \cdot \frac{s_{q,m}^{(\text{prior})}}{T}\right), \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid function, β is a learnable scaling parameter, and T is a temperature hyperparameter.

We further parameterize the prior precision as $\tau_0 = \text{softplus}(\alpha)$, where α is a learnable parameter controlling the overall strength of the prior. Together, $(\mu_{0,q,m}, \tau_0)$ define the semantic prior for model m on query q, which will later be combined with experiential evidence through Bayesian fusion.

4.2 Constructing Experiential Likelihoods from Historical Interactions

Complementing the semantic prior, we construct an *experiential likelihood* from historical user–model interactions. The goal is to extract query-relevant empirical evidence from the support set and summarize it as a data-driven utility estimate together with its uncertainty.

Given an input query q, we first encode it as $\mathbf{x}_q$, and encode each historical query q_i in the support set $\mathcal{S}$ as $\mathbf{x}_i$. We then compute the similarity between q and each historical query using scaled dot-product similarity:

$$s_{q,i} = \frac{\mathbf{x}_q^{\top} \mathbf{x}_i}{\sqrt{d}}, \qquad i = 1, \ldots, n. \tag{9}$$

Based on these similarities, we then retrieve the top-K historical queries with the highest similarity scores and denote their indices by $\mathcal{N}_q = \{i_{q,1}, \ldots, i_{q,K}\}$.

To aggregate local evidence, we normalize the similarities within the retrieved neighborhood:

$$w_{q,k} = \frac{\exp(s_{q,i_{q,k}})}{\sum_{k'=1}^{K} \exp(s_{q,i_{q,k'}})}, \qquad k = 1, \ldots, K. \tag{10}$$

These weights reflect the relevance of historical query $q_{i_{q,k}}$ to the current query q.

Recall that each historical sample in $\mathcal{S}$ is associated with an observed utility vector $\tilde{\mathbf{y}}_i$ and an observation mask $\mathbf{a}_i$. For a candidate model m, we summarize the local empirical evidence by the weighted mean

$$\mu_{\ell,q,m} = \frac{\sum_{k=1}^{K} w_{q,k}\, a_{i_{q,k},m}\, \tilde{y}_{i_{q,k},m}}{\sum_{k=1}^{K} w_{q,k}\, a_{i_{q,k},m}}, \tag{11}$$

which aggregates only those neighbors for which the utility of model m is observed.

To quantify the reliability of this empirical estimate, we further compute the corresponding weighted variance over the observed neighbors of model m, where the weights are given by $w_{q,k}$. We also compute the effective sample size as

$$n_{\text{eff},q,m} = \frac{1}{\sum_{k=1}^{K} \left(w_{q,k}\, a_{i_{q,k},m}\right)^2}. \tag{12}$$

Intuitively, $n_{\text{eff},q,m}$ measures how much usable evidence is available for model m after accounting for both retrieval relevance and missing observations.

We then define the likelihood precision as

$$\tau_{\ell,q,m} = \frac{n_{\text{eff},q,m}}{\sigma^2_{\ell,q,m} + \epsilon}, \tag{13}$$

where $\epsilon > 0$ is a small constant for numerical stability. A larger $\tau_{\ell,q,m}$ indicates that the historical evidence is both more consistent and more abundant, while a smaller value reflects higher uncertainty due to sparse or noisy observations.

In this way, the retrieved neighborhood induces an experiential likelihood summarized by $(\mu_{\ell,q,m}, \tau_{\ell,q,m})$ for each candidate model. In the next subsection, these likelihood parameters are fused with the semantic prior through Bayesian inference to produce the final utility estimate for routing.

4.3 Bayesian Fusion for Utility Estimation

The semantic prior and experiential likelihood capture complementary aspects of routing. The prior provides a stable estimate when historical evidence is limited or unreliable, while the likelihood becomes more informative when sufficient relevant interactions are available. To combine these two sources in a principled manner, we adopt Gaussian-conjugate Bayesian fusion.

Specifically, for each query–model pair (q, m), we treat the semantic prior as a Gaussian distribution with mean $\mu_{0,q,m}$ and precision τ_0, and the experiential evidence as a Gaussian likelihood with mean $\mu_{\ell,q,m}$ and precision $\tau_{\ell,q,m}$. Under this formulation, the posterior remains Gaussian, with precision $\tau_{\text{post},q,m} = \tau_0 + \tau_{\ell,q,m}$ and mean

$$\mu_{\text{post},q,m} = \frac{\tau_0 \mu_{0,q,m} + \tau_{\ell,q,m} \mu_{\ell,q,m}}{\tau_{\text{post},q,m}}. \tag{14}$$

Thus, the posterior mean is a precision-weighted combination of the prior estimate and the empirical estimate, automatically adapting to their relative reliability.

This fusion behaves appropriately in boundary cases. When $\tau_{\ell,q,m} = 0$, the posterior reduces to the prior, corresponding to the absence of usable historical evidence. As $\tau_{\ell,q,m}$ increases, the posterior places more weight on the experiential estimate. We then instantiate the utility term in Equation (2) by $\mu(q, m) = \mu_{\text{post},q,m}$, and perform routing according to Equation (1).

4.4 Training Objective

We learn the model parameters by minimizing the masked mean squared error between posterior utility predictions and observed utilities. Let $\mathcal{S}_{\text{train}}$ denote the training support set. For each training query q_i, supervision is provided only on observed query–model pairs. The objective is

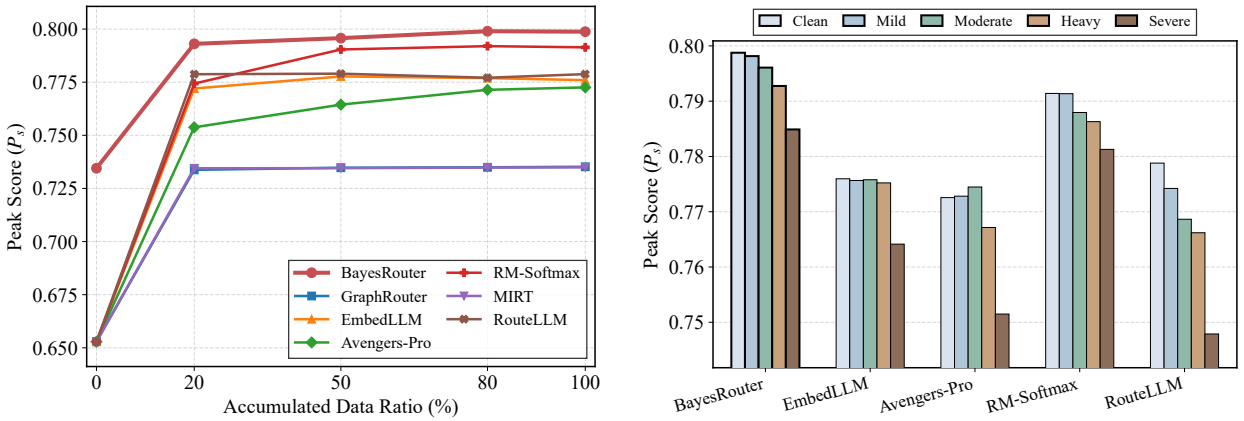
$$\mathcal{L} = \frac{1}{|\mathcal{S}_{\text{train}}|} \sum_{i=1}^{|\mathcal{S}_{\text{train}}|} \frac{\sum_{m=1}^{M} a_{i,m} \big(\mu_{\text{post},q_i,m} - \tilde{y}_{i,m}\big)^2}{\sum_{m=1}^{M} a_{i,m}}. \tag{15}$$

This objective ensures that missing entries do not contribute to the loss, while all observed utilities are used to supervise the posterior predictions. The learnable parameters are $\Theta = \{W_Q, W_D, \mathbf{b}_Q, \mathbf{b}_D, \alpha, \beta\}$, which are optimized by gradient-based training.

5 Experiment

5.1 Experimental Setup

Benchmarks and Splits. We evaluate BayesRouter on four benchmark datasets for LLM routing, including three text-only benchmarks, RouterEval (Huang et al., 2025), MixInstruct (Jiang et al., 2023), and RouterBench (Hu et al., 2024), as well as one multimodal benchmark, MMR-Bench (Ma et al., 2026). Detailed descriptions of the datasets, preprocessing procedures, and data splits are provided in Section C.



(a) P_s under continual data accumulation.

(b) P_s under different noise levels.

Figure 3: Robustness analysis under continual data accumulation and noisy supervision.

Evaluation Metrics. Following prior work on LLM routing, we evaluate each method by varying the cost trade-off parameter to obtain a quality–cost decision curve. From this curve, we report two summary metrics. P_s (*Peak Score*) denotes the maximum performance achieved along the curve, together with the corresponding inference cost (Jitkrittum et al., 2025; Zhuang et al., 2024). **QNC** (*Quality-Neutral Cost*) measures the relative cost required to match the performance of the single strongest test-time LLM, normalized by the cost of that model (Jitkrittum et al., 2025). A lower QNC indicates better routing efficiency, while a higher P_s indicates stronger achievable performance under budget-aware routing.

Comparison Methods. We compare BayesRouter with a diverse set of representative routing baselines, including GraphRouter (Feng et al., 2025), EmbedLLM (Zhuang et al., 2024), AvengersPro (Zhang et al., 2025c), RM-Softmax (Tsiourvas et al., 2025), MIRT (Song et al., 2025), and RouteLLM (Ong et al., 2025).

5.2 Analysis

Q1: When does the prior matter? To understand when semantic priors are most beneficial, we evaluate BayesRouter under two representative imperfect-evidence scenarios on RouterEval. The results are shown in Figure 3.

In Figure 3a, we simulate a continual accumulation setting in which historical interactions gradually become available. The initial point at step 0 is particularly important, since no historical interaction data are available and routing must rely entirely on prior information. In this case, BayesRouter already achieves a peak score of 0.734 using only the semantic prior, which is even higher than strong baselines such as MIRT and GraphRouter. As more interaction data accumulate, the likelihood gradually becomes more informative and routing decisions become increasingly data-driven. Nevertheless, BayesRouter remains consistently competitive throughout the process. This result shows that the prior provides a strong initialization under cold-start conditions, while naturally giving way to experiential evidence as sufficient data become available.

In Figure 3b, we further examine robustness under noisy supervision by injecting five levels of noise into historical interaction data (see Section D.1 for details). We observe that existing routing methods deteriorate substantially as the noise level increases, indicating their sensitivity to corrupted historical evidence. In contrast, BayesRouter exhibits clear robustness: even under the most severe noise setting, it still maintains a peak score of 0.785. This suggests that the semantic prior not only helps when evidence is absent, but also acts as an effective regularizer when observed interactions are unreliable.

We further provide several representative **cases** in Section E to qualitatively illustrate how semantic priors help BayesRouter make more suitable routing decisions.

Table 1: Comparison of routing methods on four benchmarks under peak score (P_s) and quality-neutral cost (QNC). Higher P_s is better, while lower QNC is better. The best results are shown in bold and the second-best results are underlined.

Method	RouterEval		RouterBench		MixInstruct		MMR-Bench	
	$P_s \uparrow$	QNC $\downarrow$	$P_s \uparrow$	QNC $\downarrow$	$P_s \uparrow$	QNC $\downarrow$	$P_s \uparrow$	QNC $\downarrow$
RouteLLM	0.7788	∞	0.8014	0.9999	0.7477	1.0000	0.7290	∞
MIRT	0.7345	∞	0.8046	0.9999	0.7477	1.0000	<u>0.7482</u>	<u>1.0000</u>
EmbedLLM	0.7760	∞	<u>0.8051</u>	<u>0.9639</u>	0.7488	0.9927	0.7283	∞
GraphRouter	0.7336	∞	0.8046	1.0000	0.7467	∞	0.7143	∞
Avengers-Pro	0.7726	∞	0.8006	∞	0.7493	0.9878	0.7283	∞
RM-Softmax	<u>0.7913</u>	<u>0.9164</u>	0.8043	0.9936	<u>0.7495</u>	<u>0.9876</u>	0.7395	∞
BayesRouter (Ours)	**0.7988**	**0.8024**	**0.8054**	**0.9353**	**0.7497**	**0.9816**	**0.7502**	**0.8867**

Q2: Does BayesRouter still perform well when the likelihood dominates? A natural question is whether priors remain beneficial when historical evidence is abundant and reliable, namely in the standard routing regime where likelihood-based estimation is expected to dominate. To answer this, we report the summary results in Table 1.

As shown in Table 1, BayesRouter achieves the best overall performance on all four benchmarks, consistently outperforming existing baselines in both P_s and QNC. In particular, it attains the highest P_s on all four datasets, showing that Bayesian fusion does not compromise performance even when experiential evidence is already strong. At the same time, it also achieves the lowest QNC on all benchmarks, indicating that it can match the quality of the strongest test-time model at substantially lower cost.

These results suggest that the proposed prior–likelihood fusion is beneficial not only under imperfect-evidence settings, but also in the conventional routing regime. When evidence is scarce or noisy, the prior provides initialization and regularization; when evidence is abundant, the likelihood naturally dominates the posterior, allowing BayesRouter to retain strong data-driven routing decisions without sacrificing efficiency.

Q3: How efficient is BayesRouter? Does incorporating priors slow down routing? A natural concern is that introducing semantic priors may increase routing latency and reduce the practical efficiency of the system. To examine this, we compare the inference time of all methods on the four benchmarks, and report their runtime relative to BayesRouter in Figure 4. The results show that BayesRouter remains highly efficient in practice. Among all compared methods, it is second only to EmbedLLM in inference speed, while consistently outperforming the other baselines across all benchmarks. This indicates that the proposed prior construction and Bayesian fusion introduce only modest computational overhead. Therefore, the performance gains of BayesRouter are not achieved at the expense of substantial additional latency, making it a practical solution for cost-aware routing.

Q4: How sensitive is BayesRouter to hyperparameters, and is it robust? Since BayesRouter contains both prior and likelihood components, we further study its sensitivity to key hyperparameters. On RouterBench, we vary the number of neighbors K, the projection dimension d_{proj}, the temperature T, the batch size, and the learning rate; detailed results are reported in Figure 5.

Overall, BayesRouter is robust across most hyperparameters. The main sensitivity lies in K, which is related to dataset size and can usually be handled by choosing a moderately large value. In contrast, performance varies only slightly with respect to d_{proj}, T, batch size, and learning rate, indicating that BayesRouter does not require delicate hyperparameter tuning.

6 Conclusion

This paper studies LLM routing under imperfect historical evidence, where interaction data can be scarce, missing, or noisy. To address this issue, we propose BayesRouter, a Bayesian routing framework that combines semantic priors from model descriptions with experiential likelihoods from historical interactions. Experiments on text-only and multimodal benchmarks show that BayesRouter achieves better cost–performance trade-offs than existing routing methods and remains robust in challenging settings such as cold-start and noisy supervision.

References

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, et al. Automix: Automatically mixing language models. In *NeurIPS*, 2024.

Fu Bang. Gptcache: An open-source semantic cache for llm applications enabling faster answers and cost savings. In *NLP-OSS*, 2023.

Adarsh Prasad Behera, Jaya Prakash Champati, Roberto Morabito, Sasu Tarkoma, and James Gross. Towards efficient multi-llm inference: Characterization and analysis of llm routing and hierarchical techniques. *arXiv preprint arXiv:2506.06579*, 2025.

Lihu Chen and Gaël Varoquaux. What is the role of small models in the llm era: A survey. In *arXiv preprint arXiv:2409.06857*, 2024.

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

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 *NeurIPS*, 2024.

Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for llm selections. In *ICLR*, 2025.

Surya Narayanan Hari and Matt Thomson. Tryage: Real-time, intelligent routing of user prompts to large language models. *arXiv preprint arXiv:2308.11601*, 2023.

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. In *arXiv preprint arXiv:2403.12031*, 2024.

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. Routereval: A comprehensive benchmark for routing llms to explore model-level scaling up in llms. In *EMNLP*, 2025.

Shashank Mohan Jain. Hugging face. In *Introduction to transformers for NLP: With the hugging face library and models to solve problems*. Springer, 2022.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *ACL*, 2023.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Zifeng Wang, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. Universal llm routing with correctness-based representation. In *First Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models*, 2025.

Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In *ACM SIGKDD*, 2024.

Guannan Lai and Han-Jia Ye. When routing collapses: On the degenerate convergence of llm routers. *arXiv preprint arXiv:2602.03478*, 2026.

Zhipeng Li, Binglin Wu, Yingyi Zhang, Xianneng Li, Kai Li, and Weizhi Chen. Cusmer: Multimodal intent recognition in customer service via data augment and llm merge. In *Companion Proceedings of the ACM on Web Conference 2025*, 2025.

Haoxuan Ma, Guannan Lai, and Han-Jia Ye. Mmr-bench: A comprehensive benchmark for multimodal llm routing. *arXiv preprint arXiv:2601.17814*, 2026.

Alireza Mohammadshahi, Arshad Rafiq Shaikh, and Majid Yazdani. Routoo: Learning to route to large language models effectively. *arXiv preprint arXiv:2401.13979*, 2024.

Pranav Narayanan Venkit, Philippe Laban, Yilun Zhou, Yixin Mao, and Chien-Sheng Wu. Search engines in the ai era: A qualitative understanding to the false promise of factual and verifiable source-cited responses in llm-based search. In *Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency*, 2025.

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

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 *NeurIPS*, 2023.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Llm routing with benchmark datasets. In *NeurIPS 2023 Workshop on Distribution Shifts: New Frontiers with Foundation Models*, 2023.

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 *ACL*, 2025.

KV Srivatsa, Kaushal Kumar Maurya, and Ekaterina Kochmar. Harnessing the power of multiple minds: Lessons learned from llm routing. In *NAACLW*, 2024.

Dimitris Stripelis, Zijian Hu, Jipeng Zhang, Zhaozhuo Xu, Alay Dilipbhai Shah, Han Jin, Yuhang Yao, Salman Avestimehr, and Chaoyang He. Tensoropera router: A multi-model router for efficient llm inference. In *EMNLP*, 2024.

Asterios Tsiourvas, Wei Sun, and Georgia Perakis. Causal llm routing: End-to-end regret minimization from observational data. In *NeurIPS*, 2025.

Clovis Varangot-Reille, Christophe Bouvard, Antoine Gourru, Mathieu Ciancone, Marion Schaeffer, and François Jacquenet. Doing more with less–implementing routing strategies in large language model-based systems: An extended survey. *arXiv preprint arXiv:2502.00409*, 2025.

Chenxu Wang, Hao Li, Yiqun Zhang, Linyao Chen, Jianhao Chen, Ping Jian, Peng Ye, Qiaosheng Zhang, and Shuyue Hu. Icl-router: In-context learned model representations for llm routing. In *AAAI*, 2026.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. Mixllm: Dynamic routing in mixed large language models. *arXiv preprint arXiv:2502.18482*, 2025.

Tuo Zhang, Asal Mehradfar, Dimitrios Dimitriadis, and Salman Avestimehr. Leveraging uncertainty estimation for efficient llm routing. *arXiv preprint arXiv:2502.11021*, 2025a.

Yi-Kai Zhang, De-Chuan Zhan, and Han-Jia Ye. Capability instruction tuning: A new paradigm for dynamic llm routing. In *AAAI*, 2025b.

Yiqun Zhang, Hao Li, Jianhao Chen, Hangfan Zhang, Peng Ye, Lei Bai, and Shuyue Hu. Beyond gpt-5: Making llms cheaper and better via performance-efficiency optimized routing. In *DAI*, 2025c.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. Embedllm: Learning compact representations of large language models. In *ICLR*, 2024.

Appendix

Organization of the Appendix. This appendix is organized as follows. In Table 2, we summarize the main notations used in BayesRouter. In Section B, we provide the prompt templates used for model description standardization. In Section C, we describe the benchmark datasets, data splits, and evaluation metrics. In Section D, we present supplementary experimental results, including additional details on the noise injection experiments, inference time comparison, and hyperparameter robustness. Finally, in Section E, we provide qualitative case studies to further illustrate the routing behavior of BayesRouter.

A Notation

Table 2: Main notations used in BayesRouter.

Notation	Description
q	Input query.
m	Index of a candidate LLM.
$\mathcal{M}$, M	Candidate LLM pool and its size, respectively.
$S(q, m)$	Cost-adjusted routing score for query q and model m.
$\mu(q, m)$	Target utility of model m on query q.
$\bar{c}_m$	Normalized inference cost of model m.
λ	Trade-off coefficient between utility and cost.
$\mathcal{S} = \{(q_i, \tilde{\mathbf{y}}_i, \mathbf{a}_i)\}_{i=1}^n$	Historical support set used for routing.
$\tilde{y}_{i,m}$	Observed utility of model m on historical query q_i; it can be noisy or missing.
$a_{i,m}$	Observation mask, where $a_{i,m} = 1$ means $\tilde{y}_{i,m}$ is observed.
$Y_{q,m}$	Latent utility random variable of model m on query q.
$z_{q,m}$	External semantic signal used to construct the prior.
$\tilde{d}_m$	Standardized textual description of model m.
$\phi_{\text{query}}(\cdot)$, $\phi_{\text{desc}}(\cdot)$	Query encoder and description encoder.
x_q, g_m	Encoded representations of query q and description $\tilde{d}_m$.
$\mu_{0,q,m}$, τ_0	Mean of the semantic prior for query–model pair (q, m) and the global prior precision.
$\mathcal{N}_q$	Index set of the top-K retrieved historical neighbors for query q.
$w_{q,k}$	Normalized relevance weight of the k-th retrieved neighbor.
$\mu_{\ell,q,m}$, $\sigma^2_{\ell,q,m}$	Weighted empirical mean and variance estimated from retrieved neighbors.
$n_{\text{eff},q,m}$	Effective sample size for model m under the retrieved neighborhood of q.
$\tau_{\ell,q,m}$	Precision of the experiential likelihood.
$\mu_{\text{post},q,m}$, $\tau_{\text{post},q,m}$	Mean and precision of the posterior utility after Bayesian fusion.
K	Number of retrieved neighbors used to construct the likelihood.

B Prompt Templates for Model Description Standardization

To construct semantic priors from model descriptions, we first standardize the raw model cards or metadata into concise, routing-oriented capability summaries. In our implementation, this is done by querying an auxiliary LLM with a fixed system prompt and a model-specific user prompt. The auxiliary model is used only for description generation and is never included in the routing candidate pool.

B.1 System Prompt

System Prompt

You generate cold-start routing descriptions for AI models.
Your job is to write a compact, conservative, routing-oriented description for a TARGET MODEL NAME.
You MUST use web search results available in context. Prefer these sources in this order:
1. Official model docs / official blog / official announcements
2. Official GitHub or Hugging Face model card
3. Official paper abstract or project page
Rules:
- The target model name may be a local alias, shorthand, or slightly non-standard.
- First resolve the most likely canonical model identity from evidence.
- If identity is ambiguous, say so explicitly and keep the description cautious.
- Do NOT invent benchmark scores, release dates, or unsupported capabilities.
- Do NOT use marketing language like "state-of-the-art", "best", or "leading" unless directly supported by an official source, and even then avoid repeating the slogan.
- Focus on what helps routing: modality support, family/size if clear, likely strengths, likely limitations, and where evidence is uncertain.
- Description length: 70 to 120 words.
- Mention uncertainty when evidence is limited.
- Return valid JSON that exactly matches the schema.

B.2 User Prompt

User Prompt Template

Target model name: `{model_name}`
Task:
1. Search the web for the most likely official information about this model.
2. Resolve the canonical model identity as carefully as possible.
3. Produce a routing-oriented description suitable for use as a cold-start prior.
The description should help decide when to route queries to this model. Include:
- whether it is text-only or multimodal if known
- family / size / variant if confidently known
- major strengths (for example coding, reasoning, document understanding, vision, long context, multilingual, tool use) only if supported
- notable limitations or uncertainty if supported or unresolved
Do not include citations in the description text itself.

B.3 Output Schema

Structured Output Fields

The auxiliary LLM is required to return a JSON object with the following fields:

- `canonical_name`: the resolved canonical model identity;
- `confidence`: one of `high`, `medium`, or `low`;
- `description`: a conservative routing-oriented description of 70–120 words;
- `evidence`: a list of key sources used to support the identification and description.

The resulting description is then used as the standardized semantic description $\tilde{d}_m$ for model m, which serves as the basis for prior construction in BayesRouter.

C Benchmark

In this section, we briefly describe the benchmark datasets used in our experiments and the corresponding data splits.

RouterEval is a large-scale text-only benchmark for LLM routing. It contains over 200 million performance records collected from 12 popular LLM evaluations, based on more than 8,500 candidate LLMs. Since its model pool is exceptionally large, we do not enumerate the individual models in the appendix. Instead, we only report its overall scale and use it as a representative large-candidate routing benchmark.

We next summarize the remaining three benchmarks and list their candidate models for completeness.

RouterBench. RouterBench is a text-only routing benchmark with 405,467 samples, covering 8 datasets and 64 tasks. Its candidate models include:
WizardLM/WizardLM-13B-V1.2, claude-instant-v1, claude-v1, claude-v2, gpt-3.5-turbo-1106, gpt-4-1106-preview, meta/code-llama-instruct-34b-chat, meta/llama-2-70b-chat, mistralai/mistral-7b-chat, mistralai/mixtral-8x7b-chat, zero-one-ai/Yi-34B-Chat.

MixInstruct. MixInstruct is an instruction-following benchmark with 110,000 samples. Its candidate models include:
Alpaca-native, ChatGLM-6B, Dolly-v2-12b, Flan-T5-xxl, koala-7B-HF, llama-7b-hf-baize-lora-bf16, Moss-moon-003-sft, MPT-7B, mpt-7b-instruct, oasst-sft-4-pythia-12b-epoch-3.5, stablelm-tuned-alpha-7b, vicuna-13b-1.1.

MMR-Bench. MMR-Bench is a multimodal routing benchmark containing 11,000 instances over 8 datasets and 3 scenarios. Its candidate models include:
Open-weight: InternVL3-78B, Qwen2.5-VL-3B, Qwen2.5-VL-7B, Qwen2.5-VL-72B, Gemma3-4B;
Commercial: GPT-5-0807, GPT-5-Nano-0807, Claude 3.7 Sonnet, Gemini 2.5 Pro, and Gemini 2.5 Flash.

Data split. For RouterEval, RouterBench, and MixInstruct, we randomly split the data into training and test sets with a ratio of 2:8. For MMR-Bench, we follow its official frozen 2:8 split. We adopt this deliberately low-data protocol for two reasons. First, routing supervision is expensive, since obtaining one labeled training example requires evaluating multiple candidate models on the same query. Second, using a larger test split provides more stable estimates of the quality–cost curves and their summary metrics, leading to more reliable comparison across methods. Overall, this setting better reflects realistic routing scenarios, where only limited historical interactions are available for training the router.

Evaluation Metrics Following standard practice in routing and model selection Jitkrittum et al. (2025); Zhuang et al. (2024), we evaluate routers by tracing a performance–cost curve induced by sweeping the cost-sensitivity hyperparameter λ in the cost-adjusted routing score $S_{q,m}$. Concretely, for each λ, the router selects

$$m^\star = \arg\max_{m \in \{1,\dots,M\}} S_{q,m}, \tag{16}$$

and we define the corresponding test performance and expected cost as

$$A(\lambda) = \mathbb{E}_{q\sim\mathcal{D}_{\text{test}}}[u_{m^\star}(q)], \qquad \text{Cost}(\lambda) = \mathbb{E}_{q\sim\mathcal{D}_{\text{test}}}[c_{m^\star}]. \tag{17}$$

We evaluate $\{(\text{Cost}(\lambda), A(\lambda))\}$ over a grid of λ values to obtain the performance–cost curve, and report the following curve-level summary metrics.

(1) P_s (Peak Score). We define the peak performance on the curve as

$$P_s = \max_{t\in\{1,\dots,T\}} y_t, \tag{18}$$

and report it together with the corresponding cost $x_{t^\star}$, where $t^\star \in \arg\max_t y_t$.

(2) QNC (Quality–Neutral Cost). Let $A_{\max}$ denote the test performance achieved by the single best standalone model in the candidate pool (evaluated on $\mathcal{D}_{\text{test}}$), and let $x_{\max}$ denote its expected cost (i.e., the cost of always selecting that model). We define the quality–neutral cost as the minimal expected cost

required by the router to match this quality:

$$\mathrm{QNC} = \min_{t:\, y_t \geq A_{\max}} x_t, \tag{19}$$

and report the relative cost $\mathrm{QNC}/x_{\max}$. If the router never achieves $A_{\max}$ on the evaluated curve, we report QNC as ∞.

D Supplementary experimental results

D.1 Details of Noise Injection Experiments

To evaluate the robustness of different routing methods under corrupted historical evidence, we inject Gaussian noise into the performance values (`perf`) of the historical interactions. Specifically, for each noise level, we randomly select a proportion of historical records and perturb their performance values with Gaussian noise of a predefined standard deviation.

We consider five noise levels, characterized by the noise ratio and the noise standard deviation, as summarized in Table 3.

Table 3: Noise settings used in the robustness experiments.

Noise Level	**Noise Ratio**	**Noise Std**
Clean	0.0	0.0
Mild	0.3	0.1
Moderate	0.5	0.2
Heavy	0.7	0.3
Severe	0.9	0.5

As the noise level increases, existing routing methods deteriorate substantially, indicating their sensitivity to corrupted historical evidence. In contrast, BayesRouter remains notably robust: even under the most severe noise setting, it still achieves a peak score of 0.785. This result suggests that the semantic prior not only improves routing when historical evidence is limited, but also serves as an effective regularizer when the observed interactions are noisy or unreliable.

D.2 Inference Time

We further evaluate the practical efficiency of different routing methods by comparing their inference time on the four benchmarks. For a fair comparison, all methods are evaluated under the same hardware and software environment, using the same train/test split as in the main experiments. We measure the wall-clock time of the *routing stage*, i.e., the time required for the router to produce the selected model for each test query, and exclude the execution time of the downstream LLM itself. For each method, we report its average inference time on the test set and normalize it by that of BayesRouter. Therefore, a relative time larger than 1 indicates that the method is slower than BayesRouter, while a value smaller than 1 indicates that it is faster.

Figure 4 shows that BayesRouter remains highly efficient across all four benchmarks. Most existing methods are substantially slower, with particularly large overhead observed for RouteLLM, GraphRouter, and RM-Softmax on several benchmarks. In contrast, BayesRouter stays close to the fastest group, and is overall second only to EmbedLLM in inference speed. This is notable because BayesRouter additionally incorporates semantic prior construction and Bayesian fusion, yet introduces only modest computational overhead in practice.

Overall, these results show that the performance gains of BayesRouter do not come at the cost of substantial additional latency. The proposed prior-based design preserves strong practical efficiency, making BayesRouter a favorable choice for real-world cost-aware routing scenarios where both routing quality and runtime matter.

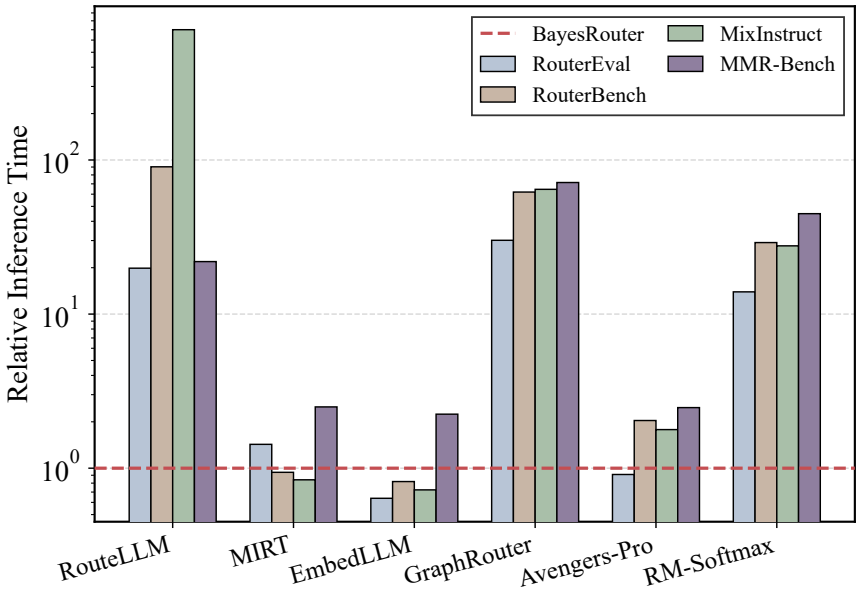


Figure 4: Relative inference time of different routing methods on the four benchmarks, normalized by BayesRouter. The red dashed line denotes the runtime of BayesRouter (set to 1), and lower values indicate better efficiency.

D.3 Hyperparameter Robustness

We further study the sensitivity of BayesRouter to its key hyperparameters on RouterBench. Specifically, we examine the number of retrieved neighbors K, the projection dimension d_{proj}, the temperature T, the batch size, and the learning rate. For each hyperparameter, we vary it over a range of candidate values while keeping all other hyperparameters fixed to their default settings used in the main experiments. We then evaluate the routing performance under each setting using the same training and test split as in the main results.

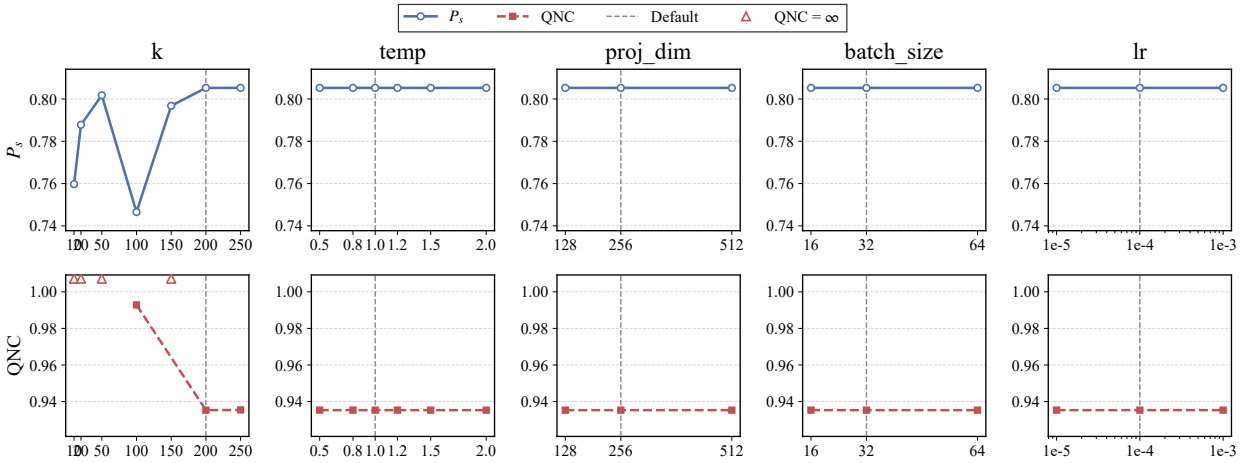


Figure 5: Hyperparameter sensitivity of BayesRouter on RouterBench. We vary one hyperparameter at a time while fixing the others to their default values in the main experiments.

Figure 5 summarizes the results. Among all hyperparameters, the main sensitivity comes from the number of neighbors K. This is expected, since K directly controls how much historical evidence is used to construct

the experiential likelihood. When K is too small, the local evidence may be insufficient or unstable; using a moderately larger K generally leads to more reliable performance. This also explains why the preferred value of K is related to the dataset size and the density of available historical interactions.

In contrast, BayesRouter is highly robust with respect to the remaining hyperparameters. The performance changes only slightly when varying the projection dimension d_{proj}, the temperature T, the batch size, or the learning rate. This indicates that the proposed method does not rely on delicate hyperparameter tuning, and that its empirical gains are stable across a broad range of configurations.

Overall, these results suggest that BayesRouter is robust in practice. Except for the expected dependence on K, which can usually be handled by choosing a moderately large value, the method remains stable across different architectural and optimization settings.

E Case Study

We further provide three representative case studies to qualitatively analyze the behavior of BayesRouter across different tasks. These examples show that BayesRouter consistently selects models whose strengths align with the input-specific requirements, rather than relying on a single model across all cases. In image QA, this leads to better visual grounding and more reliable numerical reasoning; in text QA, it yields cleaner and more accurate outputs with fewer formatting failures. In contrast, the baseline-selected models often fail due to mismatched capabilities, such as unsupported local associations, arithmetic mistakes, or output contamination. These cases complement the quantitative results by illustrating how routing improves prediction quality through better model–instance matching.

Case 1. Ferry Connection (Image QA) **Ground Truth:** `D. Oakland Jack London`

Question. Which station on the route map indicates a ferry connection available?

Input image

BayesRouter-selected Model

Selected model: `GPT-5-nano`
Prediction: `D. Oakland Jack London`
Key evidence. The model first checks the legend and identifies the light-blue dotted line as the ferry connection symbol. It then locates this symbol on the map and correctly associates it with Oakland Jack London.

Baseline-selected Model

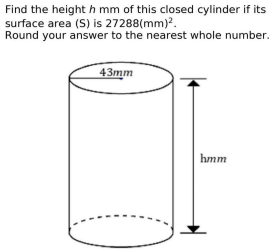
Selected model: `Gemma3-4B`
Prediction: `B. Suisun-Fairfield`
Failure. The model gives an incorrect station and does not ground its answer in the legend or the ferry symbol.

Takeaway. BayesRouter selects a model that grounds the answer in the visual legend and symbol matching, whereas the baseline-selected model makes an unsupported local association.

Figure 6: A representative image-based case where BayesRouter selects a better-matched model.

Case 2. Concrete Pipe Volume (Image QA) **Ground Truth:** `1633.63`

Question. According to the question shown in the image, please directly answer the question and provide the final value, e.g., 1, 2.5, 300.



Input image

BayesRouter-selected Model

Selected model: `Claude3-7V` **Prediction:** `1633.63`

Key evidence. The model correctly identifies the task as computing the volume of a hollow cylinder, extracts the outer radius (7 m), inner radius (3 m), and length (13 m), and derives $V = \pi(7^2 - 3^2) \times 13 = 520\pi \approx 1633.63$.

Baseline-selected Model

Selected model: `InternVL3-78B` **Prediction:** `1628.05`

Failure. The model derives the same symbolic expression 520π, but makes an arithmetic error in the final numerical conversion, leading to an incorrect answer.

Takeaway. BayesRouter selects a model that not only identifies the correct geometric formula, but also executes the final numerical computation correctly. In contrast, the baseline-selected model reaches the correct intermediate form yet fails at the last step.

Figure 7: An image-based case where BayesRouter selects a model with more reliable numerical execution.

Case 3. Soccer Team Origin Classification (Text QA) Ground Truth: `European, American, American, European, American, European`

Question. Tell me whether each of these soccer teams are American or European: Leeds United, Seattle Sounders, Portland Timbers, Arsenal Gunners, Sporting KC, Bayern Munich.

BayesRouter-selected Model

Selected model: `ChatGLM-6B`
Prediction.

- Leeds United: European
- Seattle Sounders: American
- Portland Timbers: American
- Arsenal Gunners: European
- Sporting KC: American
- Bayern Munich: European

Observation. The model gives a clean, structured response and correctly classifies all six teams.

Baseline-selected Model

Selected model: `MPT-7B`
Output excerpt. `American American European European European ... #include <iostream> ...`
Failure. The response is both incorrect and severely contaminated by unrelated code generation, making it unusable even beyond the classification errors.

Takeaway. BayesRouter selects a model that matches the task format and produces accurate, structured classifications, whereas the baseline-selected model exhibits both label errors and severe output contamination.

Figure 8: A text-based case where BayesRouter selects a model with cleaner and more reliable outputs.