Learning to Route LLMs from Implicit Cost-Performance Preferences via Meta-Learning

Jiahao Zeng[1], **Ming Tang**[2] and **Ningning Ding**[1]*
[1]Hong Kong University of Science and Technology (Guangzhou)
[2]Southern University of Science and Technology
jzeng110@connect.hkust-gz.edu.cn, tangm3@sustech.edu.cn, ningningding@hkust-gz.edu.cn

Abstract

Large language models (LLMs) present a trade-off between performance and cost, where more powerful models incur greater expense. LLM routing aims to mitigate expenses while maintaining performance by sending queries to the most suitable model. However, existing methods cannot perform well for different user cost-performance preferences. To address this gap, we introduce a novel perceptive LLM routing paradigm for personalized and user-centric cost-performance optimization, which efficiently learns users' implicit preferences through little interaction. To handle the challenge of heterogeneous user needs, we formulate preference profiles as a set of distinct tasks in contextual bandit and propose MetaRouter, a meta-learning framework designed for preference-aware LLM routing. Experimental results show that MetaRouter outperforms strong baselines on both in-distribution and out-of-distribution tasks. Furthermore, it exhibits high efficiency in learning user preferences, robustness to changes in the routable LLMs, and scalability to multi-model routing.

1 Introduction

The rapid evolution of Large Language Models (LLMs) like GPT series [OpenAI, 2025] marks a significant breakthrough in artificial intelligence. These models excel at a wide array of natural language processing tasks, including complex reasoning and sophisticated problem-solving [Bubeck *et al.*, 2023; Wei *et al.*, 2022]. However, their immense power comes at a cost. The substantial computational and memory requirements of these state-of-the-art LLMs necessitate cloud-based deployment and expensive API access. In response to these challenges, a new class of smaller and more efficient LLMs has emerged [Touvron *et al.*, 2023; Abdin *et al.*, 2024]. These models are specifically designed to operate on edge devices such as personal computers. This new paradigm offers compelling advantages, including low-cost deployment, reduced latency, and enhanced data privacy.

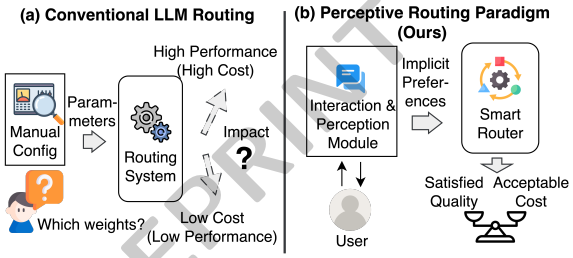

Figure 1: Perceptive LLM routing paradigm.

However, it also introduces a critical trade-off: while constantly improving, the performance of these smaller models still lags behind the large LLMs [Ding *et al.*, 2024].

To balance performance and cost, a common and effective strategy is to employ a hybrid system that balances the use of large and powerful models with smaller and more cost-effective ones. Some approaches employ a cascade method which sequentially queries different LLMs until a reliable response is found [Chen *et al.*, 2023; Aggarwal *et al.*, 2023]. However, this usually needs to query multiple times and leads to high latency. A more promising solution is to use routing methods, which direct each query to a single and optimal LLM. Similar to a small model, the router can be deployed on edge devices. It can route simple queries to a local small LLM and only use an API call for difficult queries. This strategy provides a flexible way to maximize LLM power for various budgets and quality needs.

Existing research has explored various methods for routing queries. [Ding *et al.*, 2024] used a router that assigns queries to a small or large model based on the predicted query difficulty and the desired quality level. [Ong *et al.*, 2025] proposed several efficient router models and developed a training framework that leverages human preference data. [Feng *et al.*, 2025] introduced a graph-based framework that utilized contextual information among tasks, queries, and LLMs to improve selection. However, a significant limitation of existing approaches is their reliance on manually configured parameters to strike a balance between performance and cost, as shown in Figure 1(a). Moreover, some advanced methods like [Feng *et al.*, 2025] that attempted to learn these trade-offs require retraining a new model for each distinct user preference,

*Corresponding author.

which is both inefficient and computationally expensive.

To address these limitations, we introduce a perceptive LLM routing paradigm. As shown in Figure 1(b), the router learns users' implicit preferences through interaction. However, realizing this paradigm presents two core challenges: (a) *Feedback Collection*: How to collect user feedback unobtrusively while ensuring it fully reflects users' preferences? (b) *Preference Integration*: How to infer preferences from this feedback and incorporate them into routing decisions?

In this paper, we propose MetaRouter, an end-to-end meta-learning framework for preference-aware LLM routing. We formulate the routing decision as a contextual bandit problem and treat distinct preferences as different tasks. During the meta-training phase, we train the router across diverse preference profiles to enable rapid adaptation. Specifically, our method initiates an adaptation phase where users provide pairwise comparisons of LLM responses. From this comparative feedback, we infer a latent preference representation that captures their cost-performance trade-offs. This representation is combined with users' query as input for the routing policy, allowing the system to intelligently select the optimal model for each query. By conditioning decisions on both the queries and learned preferences, our approach offers a highly personalized experience. Overall, our contributions include:

- *Novel Paradigm for LLM Routing.* We introduce *perceptive* LLM routing, a novel paradigm that goes beyond traditional *manual* configurations or model retraining. Our approach learns users' implicit preferences for cost–performance trade-offs through interaction, enabling the system to adaptively select the most suitable LLM for each query based on individual user needs.
- *Meta-Learning Framework for Preference-Awareness.* We introduce MetaRouter, a meta-learning-based approach that solves the preference-aware routing problem. To our knowledge, this is the first work to build LLM router based on meta-learning, providing insights into advancing more intelligent LLM systems.
- *Comprehensive Evaluation.* We validate MetaRouter on both in-distribution and out-of-distribution datasets. The results show that our method outperforms strong baselines across multiple performance metrics. Experiments also show its efficiency in learning user preferences, robustness to changes in the routable LLMs, and scalability to multi-model routing.

2 Related Work

Hybrid LLM. To mitigate the expenses of using LLMs, an effective strategy is to implement a hybrid system that balances the use of large and small models. Early cascade methods [Chen *et al.*, 2023; Aggarwal *et al.*, 2023] query models sequentially from cheapest to most expensive, but this can introduce significant latency. In contrast, routing methods [Ding *et al.*, 2024; Ong *et al.*, 2025; Feng *et al.*, 2025] provide a more promising solution by directly routing queries to the most suitable model. For instance, [Ding *et al.*, 2024] used a router to assign queries based on predicted difficulty and desired quality. [Ong *et al.*, 2025] proposed several efficient router models and developed a training framework leveraging human preference data to enhance performance. [Feng *et al.*, 2025] introduced GraphRouter, a framework that utilized contextual information between tasks, queries, and LLMs to improve LLM selection. Despite these advances, a significant limitation of existing approaches is their reliance on manually configured parameters to balance performance and cost, which is not user-friendly. Moreover, solutions like GraphRouter [Feng *et al.*, 2025] require retraining a new model for each distinct user preference, which is both inefficient and computationally expensive. In this work, we introduce a perceptive LLM routing paradigm to learn users' preferences.

Meta Learning for Bandits. Meta-learning represents a paradigm that enables the model to learn a learning strategy itself. Instead of training a model to master a single task, meta-learning algorithms are trained on a distribution of tasks. Applying this principle to reinforcement learning gives rise to Meta-Reinforcement Learning (Meta-RL), which addresses poor generalization in traditional RL. Meta-RL aims to learn policies that can adapt to new tasks with minimal data. While extensively studied in general RL settings [Beck *et al.*, 2023], meta-learning in the bandit framework remains less explored. Prior work has investigated meta-learning for adversarial bandit feedback [Khodak *et al.*, 2023], learning policies in Bayesian bandits [Boutilier *et al.*, 2020], and developing exploration strategies [Sharaf and Daumé, 2021]. However, these methods are not applicable to our problem due to different settings. In this work, we frame the preference-aware LLM routing problem from a meta-learning perspective and propose an algorithm that learns a routing policy capable of quickly adapting to users' preferences.

3 Problem Formulation

3.1 LLM Routing

We use X and O to denote the user query space and the LLM answer space, respectively. Let $\mathcal{M} = \{M_1, M_2, \ldots, M_K\}$ represent a set of K candidate LLMs available for routing. While our proposed framework supports routing among multiple models (as validated in Section 5.4), we focus our problem formulation on the most common binary scenario ($K = 2$) for clarity and comparativity [Ding *et al.*, 2024; Ong *et al.*, 2025]. Specifically, we consider two distinct models: a large LLM, denoted as $L : X \to O$, and a small LLM, denoted as $S : X \to O$. Large models refer to state-of-the-art models such as the latest GPT series [OpenAI, 2025], which are often accessed via APIs. These models offer superior performance but typically incur higher latency and monetary costs. Conversely, small models refer to lightweight models with fewer parameters that can be deployed locally, such as the Phi family [Abdin *et al.*, 2024]. They have lower costs and weaker performance, but can achieve performance close to that of large models on many simple tasks.

LLM routing can be modeled as a sequential decision-making problem, which can be addressed using a contextual bandit framework. Specifically, at timestep t, a routing policy π observes a state (query x_t) and selects an action $a_t \in \{S, L\}$, corresponding to the choice of model. To avoid

ambiguity with "context" as used in task inference methods (introduced in the next subsection), we exclusively use the term "state" to refer to the contextual information in the bandit formulation. The reward for routing query x to model a is denoted by $R(x, a)$, a function that depends on user utility. The performance of the routing policy can be measured by:

$$J'(\pi) = \mathbb{E}_{x\sim X}\left[R(x, a) \mid a \sim \pi(x)\right]. \tag{1}$$

Model selection often involves a trade-off between performance and cost, and users have different preferences for the trade-off. To capture this, we define a general score function $s(o)$ to represent the user utility of an LLM-generated response o. Crucially, our framework is agnostic to the specific form of $s(o)$. This function constructs the reward that guides the policy's learning, as detailed in the next section.

User utility depends on both performance and cost of response o. Specifically, we use $q(o)$ to represent the performance of a response o, obtainable through evaluation methods such as BART score [Ding *et al.*, 2024] or a LLM judge [Ong *et al.*, 2025]. Concurrently, we use $p(o)$ to denote the inference cost of o, which is based on the token count and price. During the meta-training phase (detailed in Section 4.3), we can generate a diverse set of synthetic tasks (i.e., simulated user preferences) by varying the preference parameters (e.g., trade-off weights) inherent to the definition of $s(o)$. For example, the most common form is $s(o) = w \cdot q(o) - (1 - w) \cdot p(o)$ [Feng *et al.*, 2025; Zhang *et al.*, 2025a]. Note that to reduce the influence of their scales, both $p(o)$ and $q(o)$ are normalized.

3.2 Few-shot Preference Learning

Current methods [Ding *et al.*, 2024; Ong *et al.*, 2025; Zhang *et al.*, 2025a] require users to manually configure parameters to adjust their preferences for cost and performance. However, this paradigm is unintuitive and presents usability challenges. Primarily, users often do not know the actual impact of an abstract parameter. Furthermore, this paradigm assumes that users have a clear understanding of the optimal parameter for their specific needs. Consequently, this approach can lead to suboptimal outcomes for users. In this work, we propose a method to infer user preferences from their feedback.

Motivated by the success of task inference methods in Meta-RL [Beck *et al.*, 2023; Rakelly *et al.*, 2019], we formulate different preferences as distinct tasks and aim to identify the "task" by collecting users' preference contexts. However, unlike standard Meta-RL, which leverages continuous environmental interactions (states, actions, and rewards) to infer tasks, the router cannot rely on constant user interaction, because collecting persistent preference feedback would severely degrade the user experience. Consequently, we decouple the inputs to the task inference method from the outputs of the policy. The task inference method is conditioned only on preference contexts collected during a preliminary adaptation process. In this process, we return responses from both models for a query and require the user to choose the better one by considering prices and performance. This feedback constitutes a preference context c. For example, if the user feels that the small model S is better for query x, then $c = (x, S \succ L)$. The collected contexts are denoted by $c_{1:N}$.

Regarding the task inference, we represent each task $\mathcal{T}$, corresponding to a unique preference profile, with a d-dimensional latent task variable $z \in Z$. The policy $\pi_\theta(a|x, z)$, parameterized by θ, is conditioned on the task variable z, enabling end-to-end learning of the representation z alongside the policy, which aims to differentiate between task specifications. Additionally, we introduce an inference network (context encoder) $\mathcal{E}_\phi(z|c_{1:N})$, parameterized by ϕ, to infer the task variable z from preference contexts $c_{1:N}$. The policy's objective is to maximize the expected reward:

$$J(\pi_\theta) = \mathbb{E}_{x\sim X, z\sim\mathcal{E}_\phi}\big[R(x, a) \mid a \sim \pi_\theta(x, z)\big]. \tag{2}$$

In this case, our problem presents several challenges. First, it is hard to capture implicitly diverse user preferences through explicit tasks. Second, acquiring preference contexts from users' limited feedback results in a challenging data-scarce learning environment. Finally, it is difficult to learn an effective policy for a contextual bandit problem, which is only one-step RL. We will address them in the next section.

4 Methodology

As shown in Figure 2, MetaRouter comprises two core components: a context encoder $\mathcal{E}_\phi$ and a policy network π_θ. The context encoder $\mathcal{E}_\phi$ collects users' preference contexts during an initial adaptation phase to infer the latent task variable z. Subsequently, each query will be input into the policy network π_θ together with z for intelligent routing. The following subsections will detail each component and describe how they are trained jointly through meta-training.

4.1 Context Encoder

To adapt to diverse user preferences, our router employs a context encoder $\mathcal{E}_\phi(z|c_{1:N})$. This network infers a latent context vector z which encapsulates users' preference information. This inference is conditioned on a collection of feedback contexts $c_{1:N}$ gathered during an adaptation phase.

A key challenge in personalization is designing a feedback mechanism that is both informative for the router and unobtrusive for the user. To achieve this, we employ a lightweight yet effective feedback system based on pairwise comparisons. During the adaptation process, the system presents the user with responses from both models. The user simply chooses the preferred one, considering both performance and price. This feedback constitutes a context c. This approach is cognitively less demanding than asking for absolute scores and provides a direct signal of the user's latent trade-offs.

While the feedback contexts $c_{1:N}$ are collected sequentially, we assume the underlying user preference profile remains stationary during adaptation. Therefore, the final representation z must be independent of the arbitrary order in which feedback is received. To satisfy this property, we formulate $\mathcal{E}_\phi(z|c_{1:N})$ using a permutation-invariant function:

$$\mathcal{E}_\phi(z|c_{1:N}) = \frac{1}{N}\sum_{n=1}^{N} f_\phi(c_n), \tag{3}$$

where the function f is a neural network which outputs a d-dimensional vector. The parameters of $\mathcal{E}_\phi$ are learned during

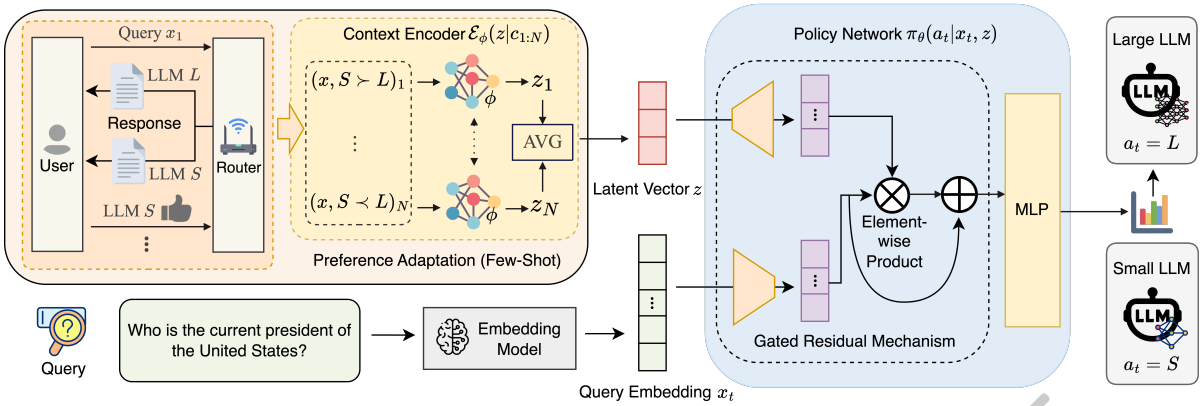

Figure 2: Overview of MetaRouter. MetaRouter consists of two components: a context encoder $\mathcal{E}_\phi$ that learns user preferences from their interactions, and a policy network π_θ that routes their queries based on those perceived preferences.

the meta-training phase and remain fixed during inference. This deterministic task variable z offers stable and clear guidance to the policy network.

4.2 Routing Policy

Optimization Objective

For the routing policy π, we employ the policy gradient (PG) method for optimization. Recall that we model the LLM routing problem as a contextual bandit, which is a one-step RL problem. At each step, the policy observes a state (the query x) and makes a decision (selects a model a) to maximize an immediate reward. Given this "one-step" nature, PG is sufficient because it does not need to account for long-term consequences or a sequence of states.

However, standard PG methods can suffer from premature convergence, where the policy becomes deterministic too quickly, failing to sufficiently explore the capabilities of different LLMs. To mitigate this and encourage exploration, we incorporate an entropy regularization term into the objective function. The overall objective is maximizing the expected reward augmented by the entropy of the policy:

$$J(\theta) = \mathbb{E}_{x,z,a}[R(x,a)] + \beta \mathbb{E}_{x,z}[H(\pi(\cdot|x,z))], \quad (4)$$

where $H(\pi(\cdot|x,z))$ is the entropy of the policy distribution, and β is a hyperparameter controlling regularization. This ensures the router maintains stochasticity during early training, preventing collapse to a trivial solution (e.g., always selecting the largest model).

Reward Design

To ensure scalability to multi-model scenarios and stabilize the training process, we adopt a baseline subtraction technique. Specifically, we define the reward as the deviation of the selected model's utility $s(o_a)$ from the average utility of all candidate models for the given query x. Furthermore, to normalize the reward scale and mitigate the impact of outliers, we apply a hyperbolic tangent function. The formulated reward function is:

$$R(x,a) = \tanh\left(\lambda\left(s(o_a) - \bar{s}(x)\right)\right), \quad (5)$$

where $\bar{s}(x) = \frac{1}{|\mathcal{A}|}\sum_{a' \in \mathcal{A}} s(o_{a'})$ represents the average utility of the candidate models, and λ is a scaling factor controlling the sensitivity of the reward. This formulation encourages the policy to select models that perform above the average, providing a robust signal for training.

Network Architecture

Conventional task inference approaches typically rely on concatenating the observation with the latent task variable. However, this strategy is suboptimal in our routing problem due to the large dimensionality disparity between the dense query embedding (e.g., 768 dimensions) and the compact task variable z (e.g., 10 dimensions). Such direct concatenation risks overshadowing the information in z. Furthermore, simple concatenation provides only implicit interaction, limiting the ability of z to explicitly modulate the query representation. To overcome these limitations, we introduce a Gated Residual Mechanism, as illustrated in the right component of Figure 2 and detailed in Supplementary Material.

Our architecture processes the inputs through two parallel branches. Specifically, we utilize an observation encoder (parameterized as an MLP) to project the query embedding into a feature representation h_{obs}. In parallel, the latent variable z is processed by a gating network to produce a modulation vector g. This process is formally defined as:

$$g = \tanh(\text{MLP}_{gate}(z)), \quad (6)$$

where MLP_{gate} denotes a multi-layer perceptron projecting z to the same dimension as h_{obs}, and the $\tanh$ activation constrains the gate values to the range $[-1, 1]$, allowing for both feature suppression and enhancement. Crucially, instead of simple multiplication, we employ a residual connection to stabilize the training:

$$h_{mod} = h_{obs} + (h_{obs} \odot g). \quad (7)$$

where $\odot$ denotes the Hadamard product. This formulation allows the user preference z to selectively enhance (positive g) or suppress (negative g) specific dimensions of the query while preserving the original query information flow. Finally, h_{mod} is fed into the MLP to output the routing probabilities.

Algorithm 1 Meta-training

Input: Batch of tasks $\mathbb{T} = \{\mathcal{T}_i\}_{i=1...T}$
Parameter: Learning rates α_1, α_2, Noise scale σ, Noise probability p_{noise}
Output: $\pi_\theta, \mathcal{E}_\phi$

1: **for** iteration in training iterations **do**
2: Sample a batch of training tasks $\mathbb{T}_{train}$.
3: **for** each $\mathcal{T}^i \in \mathbb{T}_{train}$ **do**
4: Sample contexts $c^i_{1:N} \sim S_c(X)$ and batch of queries $B^i \sim X$.
5: $\tilde{B}^i \leftarrow \{x + b_x \cdot \epsilon \mid x \in B^i\}$, where $b_x \sim$ Bernoulli(p_{noise}) and $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$.
6: Sample $z \sim \mathcal{E}_\phi(z|c^i_{1:N})$.
7: Calculate objective: $J^i_\theta = J_\theta(\tilde{B}^i, z)$.
8: **end for**
9: $\phi \leftarrow \phi + \alpha_1 \nabla_\phi \sum_i J^i_\theta$
10: $\theta \leftarrow \theta + \alpha_2 \nabla_\theta \sum_i J^i_\theta$
11: **end for**

4.3 Meta-training

The central goal of meta-training is to equip MetaRouter with the ability to rapidly adapt to diverse user preferences from minimal feedback. This is achieved by training the context encoder $\mathcal{E}_\phi$ and the routing policy π_θ across a wide distribution of simulated tasks, where each task represents a unique user preference profile, as detailed in Algorithm 1.

In each training iteration, we uniformly sample a batch of tasks $\mathbb{T}_{train}$ (line 2). For each sampled task $\mathcal{T}^i$, we generate a set of preference contexts $c^i_{1:N}$ and a batch of queries B^i (line 4). The contexts are derived by sampler S_c which applies the specific task's scoring function to LLM outputs, simulating the pairwise feedback we would get from a real user with that preference profile.

To enhance the router's robustness against input variations, we employ a noise injection strategy (line 5). Formally, for each query x in the batch B^i, we independently sample a binary mask $b_x \sim$ Bernoulli(p_{noise}). The perturbed query batch $\tilde{B}^i$ is constructed as:

$$\tilde{x} = x + b_x \cdot \epsilon, \quad \forall x \in B^i \tag{8}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ represents distinct Gaussian noise sampled for each query. Note that here x refers to the embedding vector of the query (encoded by the embedding model). This acts as implicit data augmentation to smooth the decision boundary, preventing overfitting in data-scarce scenarios and improves generalization to unseen queries. The inference network $\mathcal{E}_\phi$ encodes the contexts into a latent task variable z (line 6), which conditions the policy π_θ. The policy then processes the batch $\tilde{B}^i$ to compute the routing objective (line 7). Finally, the accumulated gradients are used to update the parameters of both networks (lines 9-10). The policy is updated to maximize routing performance, while the inference network is simultaneously updated by backpropagating the policy objective through the entire computational graph. This step trains $\mathcal{E}_\phi$ to produce latent variables z that are maximally informative for the policy, thereby directly linking the quality of the inference to the final routing performance.

5 Experiments

5.1 Experiment Setup

Datasets. We evaluate our method on three tasks: hybrid question answering (QA), code generation, and mathematical reasoning. For each task, we use one dataset to assess in-distribution (ID) performance and another to test out-of-distribution (OOD) performance: RouteLLM [Ong *et al.*, 2025] and AlpacaEval [Li *et al.*, 2023] for Hybrid QA; Magicoder [Wei *et al.*, 2024] and FullStackBench [Liu *et al.*, 2024] for code generation; and MATH [Hendrycks *et al.*, 2021] and Omni-MATH [Gao *et al.*, 2024] for mathematical reasoning. For efficiency and due to the large size of the original dataset, we primarily selected difficult queries where small models are less likely to perform as well as large models. The relevant statistics are summarized in Supplementary Material.

Candidate LLM. For Hybrid QA, as the original dataset already provides responses from gpt-4-1106-preview [OpenAI *et al.*, 2023] and Mixtral 8x7B [Jiang *et al.*, 2024], we directly use them respectively as the large model and the small model. For other datasets, we use DeepSeek-v3 [DeepSeek-AI *et al.*, 2025] as the large model and Qwen2.5-1.5B [Yang *et al.*, 2024] as the small model. Given that LLM judges have shown a high correlation with human judgment [Dubois *et al.*, 2023], we use the widely-adopted LLM judge to measure the response quality [Ong *et al.*, 2025]. We also test another quality metric BART score [Yuan *et al.*, 2021] in Section 5.5. Pricing information for the open-source models is referenced from TogetherAI [Together AI, 2025].

Baselines. We compare MetaRouter with the following strong baselines: *GraphRouter* [Feng *et al.*, 2025], which routes queries via a graph framework requiring separate training for different preferences; *Avengers-Pro* [Zhang *et al.*, 2025a; 2025b], which routes based on embedding-based clustering and a cluster-wise weighted score; and two methods from [Ong *et al.*, 2025], the training-free *Similarity-weighted (SW) ranking* and the recommended method *Matrix Factorization*, both using thresholds to control trade-offs. We also include *Oracle* to demonstrate the performance upper bound.

Implementation Details. Following common designs [Feng *et al.*, 2025; Zhang *et al.*, 2025a], we instantiate the score function as $s(o) = w \cdot q(o) - (1 - w) \cdot p(o)$, where $q(o)$ and $p(o)$ denote the normalized quality and price, respectively. We also test more complex score function in Section 5.5. For all routers, we set ten different levels of preference for balancing performance and costs. We adopt all-mpnet-base-v2 [Song *et al.*, 2020] as the embedding model to encode queries. Network architectures and other details are provided in the Supplementary Material.

Metrics. We use two standard metrics in multi-objective optimization: Hypervolume (HV) [Zitzler and Thiele, 1999] for overall quality, and Inverted Generational Distance (IGD) [Coello and Sierra, 2004] for convergence, using the Oracle's solution as the true Pareto front. To complement these geometric metrics, we report the Area Under the Cost-Performance Curve (AUC), which quantifies the router's global efficiency and robustness across the cost spectrum.

Method	RouteLLM dataset			MATH			Magicoder dataset		
	HV ↑	IGD ↓	AUC ↑	HV ↑	IGD ↓	AUC ↑	HV ↑	IGD ↓	AUC ↑
Oracle	0.9242	0	2.701	0.8697	0	0.5920	0.8514	0	0.4021
GraphRouter	0.7755	0.1348	2.493	0.6134	0.1691	0.5416	0.7006	0.1281	0.3885
SW ranking	0.4778	0.3446	1.560	0.5820	0.2061	0.4186	0.5883	0.2031	0.3127
Matrix factorization	0.6590	0.2000	2.203	0.6060	0.1885	0.4974	0.6007	0.1901	0.3509
Avengers-Pro	0.6942	0.1716	2.276	0.6269	0.1615	0.5392	0.7175	0.1081	0.3860
MetaRouter (ours)	**0.8437**	**0.0801**	**2.583**	**0.6716**	**0.1419**	**0.5454**	**0.7536**	**0.0829**	**0.3903**

Table 1: Performance on in-distribution (ID) tasks. We highlight the best results in bold, and the second-best results with an underline. Note that AUC values are scaled by 100 ($\times 10^{-2}$) for better readability.

Method	AlpacaEval			Omni-MATH			FullStackBench		
	HV ↑	IGD ↓	AUC ↑	HV ↑	IGD ↓	AUC ↑	HV ↑	IGD ↓	AUC ↑
Oracle	0.8411	0	6.120	0.7548	0	1.193	0.8667	0	0.2345
GraphRouter	0.6697	0.1212	5.520	0.5093	0.1827	1.089	0.6156	0.1656	0.2180
SW ranking	0.6049	0.2198	4.252	0.5254	0.2406	0.794	0.6661	0.1434	0.2166
Matrix factorization	0.6243	0.1600	4.978	**0.6260**	**0.0949**	1.080	0.6385	0.1543	0.2152
Avengers-Pro	0.6259	0.1497	5.426	0.4383	0.2325	1.101	0.6548	0.1379	0.2185
MetaRouter (ours)	**0.6778**	**0.1176**	**5.949**	0.5667	0.1353	**1.115**	**0.6678**	**0.1334**	**0.2202**

Table 2: Performance on out-of-distribution (OOD) tasks. We highlight the best results in bold, and the second-best results with an underline. Note that AUC values are scaled by 100 ($\times 10^{-2}$) for better readability.

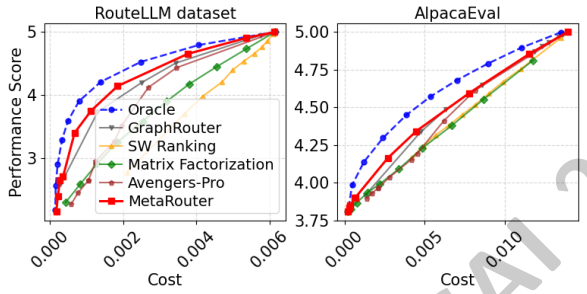

Figure 3: Performance on Hybrid QA task. The left and right figures show performance on the ID and OOD datasets, respectively. MetaRouter demonstrates better performance than other methods.

5.2 Main Results

Table 1 displays the in-distribution (ID) performance of all methods. As shown, MetaRouter consistently outperforms the baseline methods across all metrics on all ID datasets. For example, it improves the HV score by approximately 8.8% and reduces the IGD score by 40.6% compared to the second-best method on the RouteLLM dataset. Furthermore, MetaRouter achieves the highest AUC scores across all tasks. This suggests that MetaRouter is highly effective when the test data distribution matches the training distribution.

Table 2 evaluates out-of-distribution (OOD) generalization. MetaRouter demonstrates strong generalization capabilities, achieving the best performance across all metrics on AlpacaEval and FullStackBench. On Omni-MATH, while Matrix factorization achieves slightly better HV and IGD scores, MetaRouter outperforms all baselines in terms of AUC. Overall, MetaRouter maintains robust performance against baselines. We also observed a noticeable gap between the oracle and all methods in Omni-MATH and FullStackBench, which can be attributed to the limited size of the training dataset.

Figure 3 provides a visual example of the router's performance on both ID and OOD datasets for the Hybrid QA task. Other figures are available in the Supplementary Material.

5.3 Context Analysis

We need to collect contexts to infer preferences in the adaptation process. Obviously, the quantity of context significantly impacts performance. If many contexts are required for accurate inference, this method will impose a significant burden on the user, which is unacceptable. In this section, we want to answer the following question: *How many preference contexts c does the context encoder need to infer a reliable z?*

We conducted experiments on the RouteLLM dataset. To intuitively measure the impact of context quantity, we defined average task accuracy (ATA). This metric is the average accuracy computed over all test samples from all tasks. A prediction is deemed correct if our router's output matches the output from the oracle. Formally, ATA is defined as:

$$\text{ATA} = \frac{1}{T \cdot D_{\text{test}}} \sum_{j=1}^{T} \sum_{i=1}^{D_{\text{test}}} \mathbb{I}(a^{ji}_{\text{MetaRouter}} = a^{ji}_{\text{Oracle}}), \quad (9)$$

where D_{test} is the number of samples in the test set and T is the total number of tasks.

As shown in Figure 4, the experimental results indicate that only a few (around 6) contexts are needed to achieve strong performance. After that, accuracy continues to rise at a slower pace and appears to plateau around 0.86 after 18 contexts. MetaRouter reaches a saturation point where it has sufficient information for optimal performance.

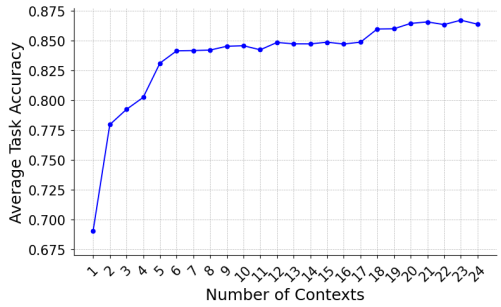

Figure 4: Effect of the number of contexts on MetaRouter's performance on the RouteLLM dataset. High performance can be achieved with only a small number of contexts.

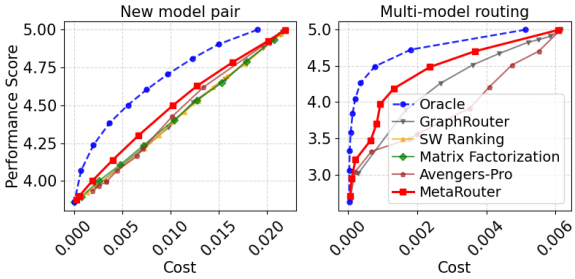

Figure 5: **Generalization and Scalability Analysis. (Left)** Performance of MetaRouter when transferring to a new model pair (Claude 3 Opus and Llama 3 8B) without retraining. **(Right)** Performance when scaling to a multi-model routing scenario with five LLMs.

5.4 Generalization and Scalability

Generalization to New Model Pair. To evaluate generalization, we tested the router on AlpacaEval using a new model pair (Claude 3 Opus [Anthropic, 2024] and Llama 3 8B [AI@Meta, 2024]) instead of the original training pair (GPT-4 [OpenAI *et al.*, 2023] and Mixtral-8x7B [Jiang *et al.*, 2024]), without retraining. As shown in Figure 5 (left), MetaRouter consistently outperforms baselines, demonstrating strong generalization capabilities.

Scalability to Multi-Model Routing. We further extended our framework to a multi-model setting involving five candidates on the RouteLLM dataset: Mixtral-8x7B, Llama 3 8B, Yi 34B [AI *et al.*, 2024], Llama 3 70B, and GPT-4. To adapt to this scenario, we maintain the pairwise comparison mechanism during the preference adaptation phase by sampling response pairs from the candidate pool. Crucially, our reward formulation (Eq. (5)) naturally scales to this setting, requiring no architectural changes other than adjusting the policy's output dimension to 5. Note that baselines strictly limited to binary settings (SW Ranking and Matrix Factorization) were excluded. Figure 5 (right) shows that MetaRouter significantly outperforms other baselines, indicating that our formulation scales naturally to larger action spaces.

5.5 Sensitivity Analysis

Utility Functions. To assess adaptability to complex user preferences, we employed the non-linear Cobb-Douglas utility function [Cobb and Douglas, 1928], defined as $s(o) = q(o)^{1-w} \cdot p(o)^{-w}$. We exclude threshold-based baselines as they do not explicitly optimize utility. Figure 6 (Left) shows that MetaRouter maintains its performance advantage, confirming its robustness to diverse preference structures.

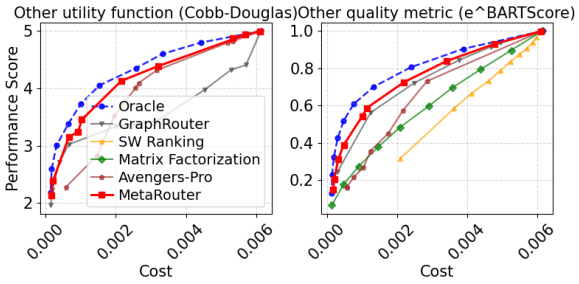

Figure 6: **Sensitivity Analysis. (Left)** Performance comparison under the non-linear Cobb-Douglas utility function. MetaRouter (red) remains closest to the Oracle (blue). **(Right)** Evaluation using transformed BART scores as the quality metric. The proposed method consistently outperforms baselines across varying cost constraints.

Quality Metric for Responses. We substituted the LLM judge with the BART score [Yuan *et al.*, 2021]. To align this metric with our utility formulation, we transformed the original log-probability scores into the $[0, 1]$ range using an exponential function($e^{\text{BART Score}}$). Figure 6 (Right) confirms that MetaRouter consistently outperforms the baselines, regardless of the metric employed.

5.6 Ablation Study

We conducted ablation studies to validate the contribution of Entropy Regularization (ER), the tanh operation in reward design, the Gated Residual Mechanism (GRM), and Noise Injection (NI). The proposed MetaRouter yields the best performance with an HV of 0.8437. The removal of ER and NI decreases the HV to 0.8236 and 0.8233, respectively. Furthermore, the IGD metric worsens from 0.0801 to 0.0935 when GRM is excluded (replacing it with a direct concatenation of z and x). This consistent performance drop across different metrics confirms the effectiveness of our design. Please refer to the Supplementary Material for the complete table.

6 Conclusion

In this paper, we study the problem of balancing cost and performance in LLM applications. To this end, we introduce perceptive LLM routing, a novel user-centric paradigm. To realize this paradigm, we present MetaRouter which is the first framework to leverage meta-learning for LLM routing. Experimental results demonstrate that MetaRouter significantly outperforms strong baselines on both in-distribution and out-of-distribution tasks. Furthermore, it exhibits strong generalization to unseen model pairs and scalability to multi-model routing. Future work will aim to expand the scope of user preferences to include other critical dimensions such as latency and privacy.

Acknowledgements

This work is supported by National Natural Science Foundation of China (Project 62502412), Guangdong Province (Project 2024QN11X097), and HKUST-HKUST(GZ) Cross-campus Collaborative Research Scheme under the "1+1+1" Joint Funding Program (G081).

References

[Abdin *et al.*, 2024] Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, et al. Phi-3 technical report: A highly capable language model locally on your phone, 2024.

[Aggarwal *et al.*, 2023] Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. Automix: Automatically mixing language models, 2023.

[AI *et al.*, 2024] 01. AI, :, Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, et al. Yi: Open foundation models by 01.ai, 2024.

[AI@Meta, 2024] AI@Meta. Llama 3 model card. https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md, 2024. Accessed: 2025-07-11.

[Anthropic, 2024] Anthropic. Introducing the next generation of claude. https://www.anthropic.com/news/claude-3-family, 2024. Accessed: 2025-07-11.

[Beck *et al.*, 2023] Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning, 2023.

[Boutilier *et al.*, 2020] Craig Boutilier, Chih-wei Hsu, Branislav Kveton, Martin Mladenov, Csaba Szepesvari, and Manzil Zaheer. Differentiable meta-learning of bandit policies. In *Advances in Neural Information Processing Systems*, pages 2122–2134, Virtual Event, 2020. Curran Associates, Inc.

[Bubeck *et al.*, 2023] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023.

[Chen *et al.*, 2023] Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance, 2023.

[Cobb and Douglas, 1928] Charles W Cobb and Paul H Douglas. A theory of production. *The American economic review*, 18(1):139–165, 1928.

[Coello and Sierra, 2004] Carlos A. Coello Coello and Margarita Reyes Sierra. A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In *Third Mexican International Conference on Artificial Intelligence*, pages 688–697, Mexico City, Mexico, 2004. Springer.

[DeepSeek-AI *et al.*, 2025] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, et al. Deepseek-v3 technical report, 2025.

[Ding *et al.*, 2024] Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations*, Vienna, Austria, 2024. OpenReview.net.

[Dubois *et al.*, 2023] Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback, 2023.

[Feng *et al.*, 2025] Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for LLM selections. In *The Thirteenth International Conference on Learning Representations*, Singapore, 2025. OpenReview.net.

[Gao *et al.*, 2024] Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-math: A universal olympiad level mathematic benchmark for large language models, 2024.

[Hendrycks *et al.*, 2021] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems*, Virtual Event, 2021. Curran Associates, Inc.

[Jiang *et al.*, 2024] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, et al. Mixtral of experts, 2024.

[Khodak *et al.*, 2023] Misha Khodak, Ilya Osadchiy, Keegan Harris, Maria-Florina F Balcan, Kfir Y. Levy, Ron Meir, and Steven Z. Wu. Meta-learning adversarial bandit algorithms. In *Advances in Neural Information Processing Systems*, pages 35441–35471, New Orleans, LA, USA, 2023. Curran Associates, Inc.

[Li *et al.*, 2023] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 2023. Accessed: 2025-07-11.

[Liu *et al.*, 2024] Siyao Liu, He Zhu, Jerry Liu, Shulin Xin, Aoyan Li, Rui Long, Li Chen, Jack Yang, Jinxiang Xia, Z. Y. Peng, Shukai Liu, Zhaoxiang Zhang, Ge Zhang,

Wenhao Huang, Kai Shen, and Liang Xiang. Fullstack bench: Evaluating llms as full stack coders, 2024.

[Ong *et al.*, 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 *The Thirteenth International Conference on Learning Representations*, Singapore, 2025. OpenReview.net.

[OpenAI *et al.*, 2023] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report, 2023.

[OpenAI, 2025] OpenAI. Research. https://openai.com/research/, 2025. Accessed: 2025-07-11.

[Rakelly *et al.*, 2019] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5331–5340, Long Beach, California, USA, 2019. PMLR.

[Sharaf and Daumé, 2021] Amr Sharaf and Hal Daumé, III. Meta-learning effective exploration strategies for contextual bandits. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 9541–9548, Virtual Event, 2021. AAAI Press.

[Song *et al.*, 2020] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems*, pages 16857–16867, Virtual Event, 2020. Curran Associates, Inc.

[Together AI, 2025] Together AI. Pricing. https://www.together.ai/pricing, 2025. Accessed: 2025-07-11.

[Touvron *et al.*, 2023] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[Wei *et al.*, 2022] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, pages 24824–24837, New Orleans, LA, USA, 2022. Curran Associates, Inc.

[Wei *et al.*, 2024] Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering code generation with OSS-instruct. In *Proceedings of the 41st International Conference on Machine Learning*, pages 52632–52657, Vienna, Austria, 2024. PMLR.

[Yang *et al.*, 2024] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, et al. Qwen2 technical report, 2024.

[Yuan *et al.*, 2021] Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: Evaluating generated text as text generation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27263–27277. Curran Associates, Inc., 2021.

[Zhang *et al.*, 2025a] 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, 2025.

[Zhang *et al.*, 2025b] Yiqun Zhang, Hao Li, Chenxu Wang, Linyao Chen, Qiaosheng Zhang, Peng Ye, Shi Feng, Daling Wang, Zhen Wang, Xinrun Wang, Jia Xu, Lei Bai, Wanli Ouyang, and Shuyue Hu. The avengers: A simple recipe for uniting smaller language models to challenge proprietary giants, 2025.

[Zitzler and Thiele, 1999] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.