LLM Bandit: Cost-Efficient LLM Generation via Preference-Conditioned Dynamic Routing

Anonymous ACL submission

Abstract

The rapid advancement in large language models (LLMs) has brought forth a diverse range of models with varying capabilities that excel in different tasks and domains. However, selecting the optimal LLM for user queries often involves a challenging trade-off between accuracy and cost, a problem exacerbated by the diverse demands of individual queries. In this work, we present a novel framework that formulates the LLM selection process as a multi-armed bandit problem, enabling dynamic and intelligent routing of queries to the most appropriate model. Our approach incorporates a preference-conditioned dynamic routing mechanism, allowing users to specify their preferences at inference time, thereby offering a customizable balance between performance and cost. Additionally, our selection policy is designed to generalize to unseen LLMs, ensuring adaptability to new models as they emerge. Experimental results demonstrate that our method achieves significant improvements in both accuracy and cost-effectiveness across various LLM platforms, showcasing the potential of our framework to adaptively optimize LLM selection in real-world scenarios.

1 Introduction

The rapid advancement in large language models (LLMs) has created a diverse ecosystem with varying capabilities and cost profiles. While larger models like GPT-4 demonstrate superior reasoning abilities, they come with substantial costs—often $0.03-0.10 per query—making them impractical for large-scale deployments (Achiam et al., 2023). In contrast, open-source models like Mixtral-8x7B offer competitive performance at roughly 1/10th the cost (Jiang et al., 2024), while domain-specialized models excel in specific areas while maintaining lower operational costs (Roziere et al., 2023; Singhal et al., 2023). This diversity creates a complex decision space where optimal model selection must balance performance, cost, and domain-specific requirements.

Existing approaches to address this performance-cost dilemma typically fall into three categories. Ensemble methods (Jiang et al., 2023; Wang et al., 2023) combine responses from multiple LLMs but require invoking multiple models per query, multiplying costs and latency. Cascading approaches like FrugalGPT (Chen et al., 2023) and AutoMix (Madaan et al., 2023) implement sequential strategies, starting with cheaper models and escalating only when necessary, but can increase latency for complex queries. Direct routing approaches (Ding et al., 2024; Ong et al., 2024; Nguyen et al., 2024) select the most appropriate model with a single inference, but current systems struggle with generalization and adaptation to new models.

Designing effective routing systems presents several fundamental challenges. First, LLMs encounter diverse queries ranging from simple factual questions to complex reasoning tasks, requiring accurate assessment of both query complexity and model capabilities. Second, the LLM landscape evolves rapidly, demanding adaptation to new models without extensive retraining. Third, applications have varying requirements—from cost-efficient customer service to accuracy-focused legal analysis—necessitating dynamic adjustment to different preferences. Finally, routing decisions must be lightweight to minimize processing overhead.

To address these challenges, we propose a preference-conditioned dynamic routing mechanism that frames LLM selection as a multi-armed bandit problem. Our approach introduces three key innovations: (1) model identity vectors that capture capabilities across different tasks and domains, enabling efficient comparison; (2) user-specified preference parameters for dynamic performance-cost trade-offs at inference time; and (3) efficient integration of new models using only 20-50 carefully selected benchmark prompts.

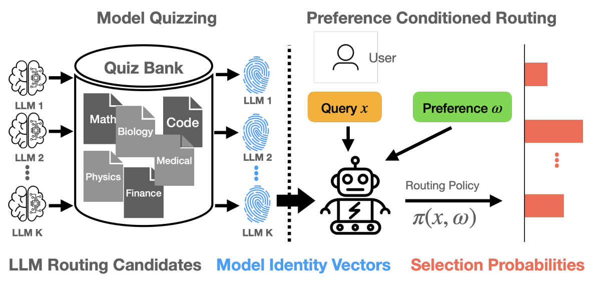


Figure 1: Overview of our preference-conditioned dynamic routing framework. Model quizzing (left) generates identity vectors capturing model capabilities, while routing policy (right) determines model selection based on user preferences and query.

Our contributions span both theoretical and practical aspects. We formulate routing as a multi-objective optimization task and develop a preference-conditioned mechanism that captures the entire Pareto front of performance-cost trade-offs. Our action-space aware policy generalizes to arbitrary sets of LLMs, demonstrated across various routing configurations. We introduce an efficient quizzing mechanism that characterizes new models with minimal evaluation, reducing integration overhead by 90% compared to full benchmark evaluation.

Experimental results across multiple benchmarks demonstrate that our method achieves up to 27% improvement in cost-efficiency while maintaining comparable performance. The framework proves especially effective in real-world scenarios where requirements vary across applications and users, enabling organizations to automatically select the most cost-effective model for each query while meeting specific performance demands.

2 Method

2.1 Problem Formulation

Let $\mathcal{X}$ denote the space of all possible queries and $\{M_k\}_{k=1}^{K}$ be a finite set of K large language models. Each model M_k is characterized by its generation capabilities and an associated cost $c_k \in \mathbb{R}_+$. For any query $x \in \mathcal{X}$ and model M_k, we define $s(x,k) \in [0,1]$ as a normalized score measuring the quality of M_k's response to query x. This score can be obtained through various evaluation metrics (e.g., accuracy, F1-score) depending on the task.

We aim to develop a routing policy $\pi : \mathcal{X} \to \mathcal{P}(K)$, where $\mathcal{P}(K)$ denotes the probability simplex over K models, that maps each query to a distribution over available models. When executing the policy, a model is sampled according to this distribution, i.e., $k \sim \pi(x)$. The routing decision results in a two-dimensional reward vector $\mathbf{r}(x,k) = [s(x,k), -c_k] \in \mathbb{R}^2$, capturing both the generation quality and the negative cost.

In the context of multi-objective optimization, we seek to maximize the expected reward vector:

$$\begin{aligned}\mathbf{J}_\pi &= \mathbb{E}_{x\sim p(x),k\sim\pi(x)}[\mathbf{r}(x,k)] \\ &= [\mathbb{E}_{x,\pi}[s(x,k)], -\mathbb{E}_{x,\pi}[c_k]],\end{aligned}$$

where $p(x)$ denotes the underlying query distribution and $\mathbb{E}_{x,\pi}$ is shorthand for the expectation over both $x \sim p(x)$ and $k \sim \pi(x)$. Given two policies π_1 and π_2, we say π_1 dominates π_2 if $\mathbf{J}_{\pi_1} \geq \mathbf{J}_{\pi_2}$ elementwise and the inequality is strict in at least one dimension. A policy π is Pareto optimal if it is not dominated by any other policy.

The set of all Pareto optimal policies forms the Pareto set Π^*, and their corresponding expected rewards $\{\mathbf{J}_\pi : \pi \in \Pi^*\}$ form the Pareto front. Due to the conflicting nature of performance and cost objectives, there typically exists no single policy that simultaneously maximizes both objectives. Instead, different policies in Π^* represent different trade-offs between performance and cost.

To navigate this trade-off, we introduce a preference parameter $\boldsymbol{\omega} = [\omega_1, \omega_2] \in \mathbb{R}^2_+$ that specifies the relative importance of performance versus cost. This allows us to define a scalarized reward: $r_{\boldsymbol{\omega}}(x,k) = \boldsymbol{\omega}^\top \mathbf{r}(x,k) = \omega_1 s(x,k) - \omega_2 c_k$. For any fixed preference $\boldsymbol{\omega}$, the optimal policy $\pi_{\boldsymbol{\omega}}$ maximizes the expected scalarized reward: $\pi_{\boldsymbol{\omega}} = \arg\max_\pi \mathbb{E}_{x\sim p(x),k\sim\pi(x)}[r_{\boldsymbol{\omega}}(x,k)]$. While the instantaneous reward $s(x,k)$ may be discrete (e.g., binary success/failure outcomes), the expected reward $\mathbb{E}_{x,\pi}[s(x,k)]$ is continuous in the policy parameters under mild regularity conditions on the policy class (see Theorem A.1 in Appendix). Specifically, when the policy π is parameterized by continuous functions (e.g., neural networks with softmax outputs), the expected reward surface remains continuous despite discrete individual rewards. This ensures the existence of optimal policies $\pi_{\boldsymbol{\omega}}$ for each preference vector $\boldsymbol{\omega}$. Moreover, as $\boldsymbol{\omega}$ varies across $\mathbb{R}^2_+$, the corresponding optimal policies $\{\pi_{\boldsymbol{\omega}} : \boldsymbol{\omega} \in \mathbb{R}^2_+\}$ trace out the complete Pareto front of achievable performance-cost trade-offs (Yang et al., 2019; Basaklar et al., 2022).

This formulation connects our problem to both multi-armed bandit (Katehakis and Veinott Jr, 1987; Bouneffouf and Rish, 2019) and multi-objective optimization (Sharma and Kumar, 2022) literature. The routing policy must learn to select models (arms) based on query-specific context, similar to contextual bandits. However, unlike traditional bandits that optimize a scalar reward, our setting involves vector-valued rewards and user-specified preferences, relating to multi-objective optimization. This combination presents unique challenges in policy learning and evaluation, which we address in subsequent sections.

2.2 Overall Framework

Given the formulation above, our framework addresses two key challenges: (1) how to efficiently characterize each model's capabilities to enable informed routing decisions, and (2) how to learn a preference-conditioned policy that generalizes across different models and queries. We propose a two-component solution: a model quizzing component that generates identity vectors capturing model capabilities, and a preference-conditioned routing policy that determines selection probabilities. Figure 1 illustrates this framework.

2.3 Model Identity Vector

To enable effective routing, we need a compact representation of each model's capabilities across different tasks and domains. Given a set of evaluation prompts $\mathcal{X} = \{x_n\}_{n=1}^N$ spanning various domains, we collect evaluation scores $\mathcal{Y}_k = \{y_{kn}\}_{n=1}^N$ for LLM M_k. Our goal is to learn a model identity vector $\mathbf{I}_k \in \mathbb{R}^d$ that predicts these evaluation scores.

We employ a variant of Item Response Theory (IRT) (Hambleton and Swaminathan, 2013) combined with deep neural networks. Unlike IRT, we leverage pretrained prompt embeddings $\mathbf{e}_n$ rather than learning explicit prompt representations, enabling generalization to unseen prompts. The score prediction model $f(\mathbf{e}_n, \mathbf{I}_k)$ outputs the probability of model M_k successfully handling prompt x_n.

For binary evaluation scores $\bar{y}_{kn}$, we optimize the binary cross-entropy loss: $\mathcal{L}_{\text{irt}} = \mathbb{E}[-\bar{y}_{kn} \log p_{kn} - (1 - \bar{y}_{kn}) \log(1 - p_{kn})]$, where $p_{kn} = \text{sigmoid}(f(\mathbf{e}_n, \mathbf{I}_k))$. For non-binary scores, we employ a thresholding mechanism in Appendix B.1.1.

We further incorporate pairwise model comparisons to enhance the identity vectors. Given responses from models M_{k_1} and M_{k_2} with annotations $z_n \in \{0, 1\}$ indicating the winner, we introduce a secondary network g that predicts winning probabilities: $\mathcal{L}_{\text{pair}} = \mathbb{E}[-z_n \log p_n - (1 - z_n) \log(1 - p_n)]$, where $p_n = \text{sigmoid}(g(\mathbf{e}_n, \mathbf{I}_{k_1}) - g(\mathbf{e}_n, \mathbf{I}_{k_2}))$.

To enhance generalization to unseen models, we employ variational inference, treating $\mathbf{I}_k$ as latent variables. This adds a KL-divergence term for regularization: $\mathcal{L}_{\text{KL}} = \mathbb{E}_k[D_{\text{KL}}(q(\mathbf{I}_k) \| p(\mathbf{I}_k))]$, where both prior $p(\mathbf{I}_k)$ and posterior $q(\mathbf{I}_k)$ are Gaussian distributions. Please see Appendix B.1 for details.

2.4 Preference-Conditioned Routing Policy

Building on our problem formulation, the core challenge is to develop a routing policy that can (1) generalize across different sets of LLMs and (2) adapt to varying user preferences $\boldsymbol{\omega}$. A natural approach would be to directly estimate the evaluation scores $s(x, k)$ using our IRT model $f(\mathbf{e}, \mathbf{I}_k)$ and select models that maximize the scalarized reward $r_{\boldsymbol{\omega}}(x, k)$. However, this direct estimation faces several limitations. The predicted scores may be inaccurate for specific query-model pairs, the estimation provides no uncertainty quantification, and most importantly, the deterministic selection strategy cannot balance exploration and exploitation.

We propose to learn a stochastic policy π_θ that maps queries to routing decisions while incorporating both the available models and user preferences as conditioning information: $\pi_\theta(k'|x, \mathcal{C}_K, \boldsymbol{\omega}) \propto \exp(\mathbf{I}_{k'}^\top h(x, \mathcal{C}_K, \boldsymbol{\omega}))$. Our formulation introduces three key innovations to address the core challenges. First, we enable generalization across model sets through action-space awareness. The policy is explicitly conditioned on model identity vectors $\{\mathbf{I}_k\}_{k=1}^K$, making it aware of available actions. The dot-product structure between model identities and network outputs allows the policy to work with arbitrary sets of models - once we compute a model's identity vector, it can be immediately incorporated into routing decisions. Second, we enhance routing decisions by incorporating comprehensive context $\mathcal{C}_K = \{(\mathbf{I}_k, c_k, \hat{p}_k)\}_{k=1}^K$, which includes not only identity vectors but also costs c_k and predicted scores $\hat{p}_k = \text{sigmoid}(f(x, \mathbf{I}_k))$. This context is processed through a permutation-invariant network $h(\cdot)$, enabling the policy to reason about relative strengths of different models for each specific query while maintaining consistency across different model orderings. Third, we enable dynamic preference adaptation by directly conditioning the policy on $\boldsymbol{\omega}$. This allows the policy to adjust its routing strategy at inference time without retraining, efficiently exploring different performance-cost trade-offs based on user requirements.

We optimize the policy following standard multi-objective policy gradient algorithms (Xu et al., 2020; Shu et al., 2024), where the gradient for updating the parameters θ is given by $\nabla_\theta[\boldsymbol{\omega}^T \mathbf{J}_{\pi_\theta}] = \mathbb{E}\left[\boldsymbol{\omega}^T \mathbf{A}(x, k')\nabla_\theta \log \pi_\theta(k' \mid x, C_K, \boldsymbol{\omega})\right]$, where $\mathbf{A}(x, k')$ indicates the advantage function estimated from sampled trajectories. The corresponding value function $\mathbf{V}_{\pi_\theta}(x, C_K)$ outputs a vector of expected returns under the current policy π_θ. The parameters of the value function are updated by a squared-error loss $\|\mathbf{V}_{\pi_\theta} - \mathbf{V}_{targ}\|^2$, where $\mathbf{V}_{targ}$ is the target value. Note the value function does not depend on the preference $\boldsymbol{\omega}$, which encourages shared values estimation across different user preferences. The vectorized value function is inspired by the core principles of multi-objective Q-learning algorithms (Yang et al., 2019; Basaklar et al., 2022). This value network and policy gradient extension can be seamlessly integrated into most existing policy gradient methods. In our implementation, we adapt Proximal Policy Optimization (PPO) (Schulman et al., 2017), where the clipped surrogate objective is used to update policy parameters. Additionally, Generalized Advantage Estimation (GAE) (Schulman et al., 2015) is employed to compute the advantage function $\mathbf{A}$ and target values $\mathbf{V}_{targ}$. For detailed derivations and implementation specifics, please refer to Appendix B.2.

A key advantage of our approach is its scalability. By leveraging model identity vectors and preference conditioning, the policy can seamlessly adapt to new models and varying user requirements without retraining from scratch. However, realizing these benefits requires careful consideration of training methodology. In the following sections, we explore techniques that ensure effective generalization across models, queries, and preferences.

2.5 Training for Generalization

While our policy architecture enables handling different models and preferences, realizing these capabilities requires careful training strategies. We identify three key generalization challenges: (1) handling arbitrary sets of models, (2) generalizing to unseen queries, and (3) maintaining consistent performance across preferences.

To handle arbitrary model sets, we employ two complementary strategies. First, we train the policy on dynamically sampled sets of models with varying sizes and capabilities. We leverage evaluation leaderboards like HELM (Liang et al., 2022) that provide scores for diverse models, randomly selecting different combinations during training. This exposure to diverse model combinations forces the policy to learn generalizable routing strategies rather than memorizing specific model relationships. Second, we address the challenge of varying score and cost scales across different model combinations. For instance, comparing GPT-4 with Mixtral-8x7B yields different scales than comparing two open-source models. We handle this through reward normalization within each set: $\bar{s}_k = s_k / \max(\{s_k\}_{k=1}^K)$, $\bar{c}_k = c_k / \max(\{c_k\}_{k=1}^K)$. This normalization ensures consistent reward scales regardless of the specific models, enabling stable optimization. Moreover, it maintains consistent interpretation of preference vectors - the same preference $\boldsymbol{\omega}$ represents similar trade-offs across different model combinations.

For query generalization, we employ two techniques. First, we perform large-scale pretraining on pairwise model comparison datasets, such as Nectar (Zhu et al., 2023) and Chatbot Arena (Zheng et al., 2023). While these datasets feature diverse user queries that help learn generalizable routing behaviors, they only provide binary winning labels rather than model-specific evaluation scores. To leverage this data, we first obtain predicted scores from our IRT model, then calibrate them using Platt scaling (Platt et al., 1999): $\bar{p}_k = \text{sigmoid}(\alpha f(x, \mathbf{I}_k) + \beta)$, where α and β are learned to align score predictions with human preferences (see Appendix B.3.1). The policy is pretrained to predict actions that maximize the calibrated reward: $\hat{a} = \arg\max_{k \in \{k_1, k_2\}} \boldsymbol{\omega}^T[\bar{p}_k, -c_k]$. Second, we introduce an on-manifold mixup regularization during the subsequent reinforcement learning phase. When sampling queries from the replay buffer, we interpolate each prompt embedding with its nearest neighbor. This neighborhood-based interpolation ensures the mixed embeddings remain meaningful, helping the policy learn smoother decision boundaries (see Appendix B.3.2).

For preference generalization, the key challenge is maintaining Pareto optimality while enabling efficient learning across different trade-offs. We leverage two complementary strategies. First, our decomposed value function $\mathbf{V}_{\pi_\theta}(x, C_K)$ estimates score and cost components independently. This decomposition enables value estimation sharing across preferences while maintaining separate tracking of objectives. Second, we train with dynamically sampled preferences $\boldsymbol{\omega} \sim U(\omega_{\min}, \omega_{\max})$, forcing the policy to learn consistent behaviors across different trade-offs. The preference range is chosen to cover practical trade-offs between quality and cost.

2.6 Cold Start for New Routing Candidates

A key advantage of our framework is its ability to efficiently incorporate new LLMs without retraining the routing policy. When a new model $\tilde{M}$ is introduced, we only need to compute its identity vector $\tilde{\mathbf{I}}$ to enable routing. While this vector could be obtained through full benchmark evaluation, such an approach would be prohibitively expensive and time-consuming. We propose instead an efficient characterization method that requires evaluating only 20-50 carefully selected prompts, reducing the integration overhead by 90% or more compared to full evaluation.

Our approach is based on the insight that not all evaluation prompts are equally informative for distinguishing model capabilities. Given existing prompts $\mathcal{X} = \{x_n\}_{n=1}^N$ and binary evaluation scores $\mathcal{Y}_k = \{\bar{y}_{kn}\}_{n=1}^N$ for existing models, we compute a discrimination score for each prompt:

$$\psi_n = \mathbb{E}_k[-\bar{y}_{kn} \log p_{kn} - (1 - \bar{y}_{kn}) \log(1 - p_{kn})]$$

where $p_{kn} = \text{sigmoid}(f(x_n, \mathbf{I}_k))$ indicates IRT model's prediction. The score ψ_n measures the average prediction error across models for prompt x_n. A high ψ_n indicates that our IRT model struggles to accurately predict model performance on this prompt, often because models with similar capabilities exhibit inconsistent performance. Conversely, a low ψ_n suggests model performance is highly predictable - either consistently successful or unsuccessful across models with similar capabilities.

Using these discrimination scores, we select a representative subset of prompts $\tilde{\mathcal{X}}$ through stratified sampling. By sampling from different strata of ψ_n values, we ensure our evaluation set covers prompts with varying discriminative power. Given the new model's evaluation scores $\tilde{Y}$ on these selected prompts, we compute its identity vector by:

$$\tilde{\mathbf{I}} = \arg\min_{\mathbf{I}} [\mathbb{E}_{\tilde{\mathcal{X}}} \mathcal{L}_{irt} + D_{\text{KL}}(q \| p)], \qquad (1)$$

where $\mathcal{L}_{irt} = -\bar{y} \log p - (1 - \bar{y}) \log(1 - p), p = \text{sigmoid}(f(x, \mathbf{I}))$. The KL term acts as a regularizer, encouraging the identity vector to remain close to our prior distribution over model capabilities. Importantly, this optimization updates only the identity vector $\tilde{\mathbf{I}}$ while keeping the IRT model fixed. Once computed, the identity vector $\tilde{\mathbf{I}}$ can be immediately used by our routing policy without any additional training or fine-tuning. This is enabled by our dot-product architecture that naturally extends to new models. As shown in Fig. 3, routing performance with these efficiently computed identity vectors closely matches that of vectors computed using full evaluation.

The combination of discriminative prompt selection and efficient identity vector computation provides a practical solution for maintaining an up-to-date routing system in the rapidly evolving LLM landscape. Implementation details and additional analysis can be found in Appendix B.3.4.

3 Related Works

LLM Ensemble, Cascade and Routing As LLMs diversify, researchers have developed strategies to balance performance and cost. Ensemble methods (Jiang et al., 2023; Wang et al., 2023; Lu et al., 2024) aggregate multiple model outputs but multiply costs. Cascading approaches (Chen et al., 2023; Madaan et al., 2023; Ramírez et al., 2024) start with cheaper models but still require multiple inferences for complex queries. In contrast, routing methods direct queries to the most suitable model with a single inference, either by predicting evaluation scores (Shnitzer et al., 2023; Lu et al., 2023; Hari and Thomson, 2023; Šakota et al., 2024) or win rates between models (Ding et al., 2024; Ong et al., 2024). Most closely related, MetaLLM (Nguyen et al., 2024) frames routing as a bandit problem but optimizes for fixed preferences and predefined models, whereas ours generalizes to dynamic preferences and adapts to new LLMs without retraining.

Multi-objective Reinforcement Learning Multi-objective RL optimizes conflicting rewards to find Pareto-optimal policies. Traditional methods sample discrete policies (Van Moffaert and Nowé, 2014; Parisi et al., 2014; Xu et al., 2020) but face dimensionality challenges as objectives increase. Modern approaches use preference-conditioned networks (Yang et al., 2019; Abels et al., 2019; Basaklar et al., 2022) or hypernetworks (Chauhan et al., 2023; Shu et al., 2024) to represent the entire Pareto front with a single model. Our work employs preference conditioning specifically tailored for LLM routing, enabling efficient adaptation across performance-cost tradeoffs.

Generalization in Reinforcement Learning Zero-shot RL enables policies to handle unseen tasks without retraining (Korkmaz, 2024). Our approach adopts a representation-based paradigm where task characteristics are explicitly encoded as model identity vectors and costs, similar to action space generalization (Jain et al., 2020). For observation distribution generalization, we employ regularization techniques inspired by Cobbe et al. (2019) and Zhang and Guo (2021). Additional related work is discussed in Appendix D.

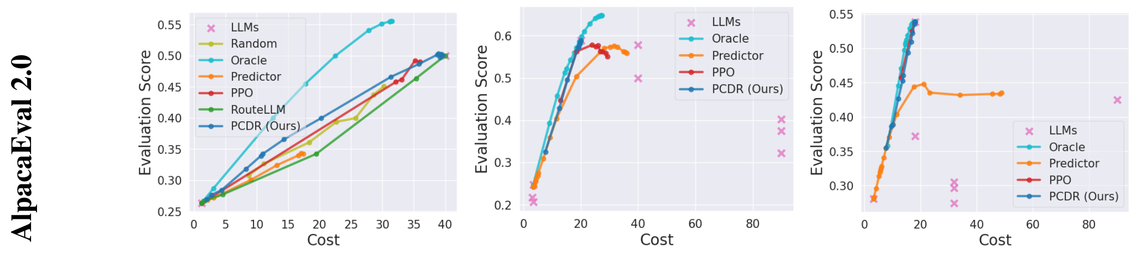
4 Experiments

We evaluate our routing policy on five popular LLM benchmarks: HELM-Lite, HELM-MMLU (Liang et al., 2022), HuggingFace OpenLLM Leaderboard, OpenLLM Leaderboard v2 (Beeching et al., 2023; Fourrier et al., 2024), and AlpacaEval 2.0 (Li et al., 2023). For each benchmark, we divide prompts into training and test splits, with the former used for policy training and the latter for evaluation. Our routing policy is pretrained on pairwise comparison datasets including Chatbot Arena (Zheng et al., 2023), Nectar (Zhu et al., 2023), and RouteLLM's synthetic dataset (Ong et al., 2024). The IRT model is trained on these same pairwise datasets plus training splits from all leaderboards. We approximate model costs based on processing and generating 1M tokens each (see Appendix E for details).

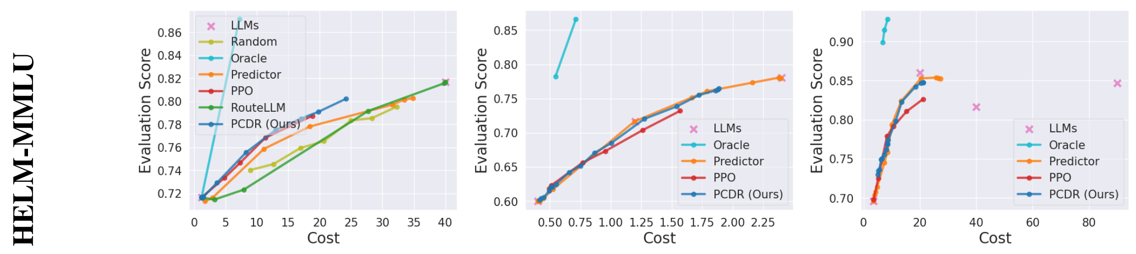
For comparative evaluation, we follow RouteLLM's setup with GPT-4 and Mixtral-8x7B as candidates, noting that while RouteLLM is specifically trained for this two-model scenario, our approach handles arbitrary model sets. To test generalization, we evaluate two additional multi-model configurations per dataset where RouteLLM is not applicable. We compare against several baselines: (1) RouteLLM (two-model scenarios only), (2) a random selection baseline (two-model only), (3) a Predictor baseline that uses predicted scores $\hat{p}_k$ to compute utility, (4) separate PPO policies trained for each LLM set and preference, and (5) an Oracle policy that selects models based on actual evaluation scores. For preference adjustment, RouteLLM uses thresholds while our method directly accepts preference parameters as inputs.

Results Figure 2 demonstrates our method's routing performance across five major LLM evaluation benchmarks with various model combinations. The results reveal several key advantages of our approach: First, our Predictor baseline consistently outperforms RouteLLM, validating the effectiveness of our model identity vector and score prediction framework. Second, our preference-conditioned routing policy further improves upon the Predictor baseline, particularly in challenging scenarios where score prediction is less reliable. This is especially evident in AlpacaEval 2.0 (c) and HELM-Lite (b), where the routing policy learns to compensate for prediction uncertainties by incorporating broader context about model capabilities and costs. When compared to RouteLLM, our policy demonstrates substantial cost savings - on AlpacaEval 2.0 with the GPT4/Mixtral-8x7B configuration, our approach achieves 46.35% accuracy at \$31 cost compared to RouteLLM's \$35, representing an 11% cost reduction. On MMLU, the improvement is even more significant, reducing costs from \$33 to \$24 (27% reduction) while maintaining 80% accuracy. Third, our single routing policy achieves comparable or better performance than separately trained PPO policies across all datasets and LLM configurations. This is a crucial advantage, as each PPO baseline requires specific training for its fixed set of models and preference settings, while our approach generalizes across arbitrary model combinations and preferences without retraining. This demonstration of robust generalization is particularly important for practical deployments where model sets and requirements frequently change. While these results demonstrate significant improvements over existing methods, the performance gap between all routing policies and the Oracle baseline indicates potential for further optimization. This gap suggests opportunities for future work in improving both prediction accuracy and routing strategy.

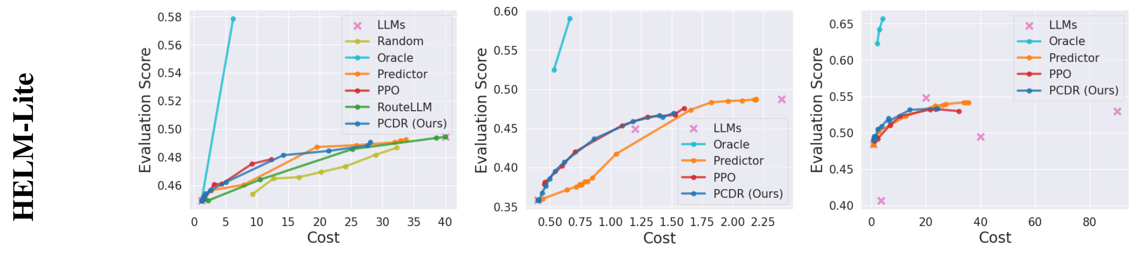
Cold Start for New Routing Candidates To simulate the scenario where new models are introduced into the routing system, we select several unseen models from the HuggingFace OpenLLM v2 benchmark. These models are not used for training either the IRT model or the routing policy. For detailed evaluation settings, please refer to Appendix E.9. The identity vectors for these models are obtained by optimize (1) over a selected subset of prompts from the OpenLLMv2 benchmark. We explore different evaluation budgets, selecting 10, 20 or 50 prompts to obtain the evaluation scores for these newly added models. The *Predictor* baseline utilizes the learned identity vectors to predict the evaluation scores, while the *PPO* baseline trains the routing policy using the same set of selected prompts. For our preference conditioned routing policy, we directly plug the identity vectors into the



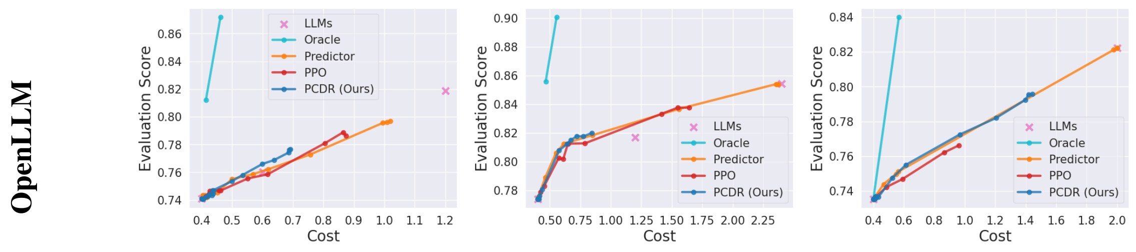
(a) GPT4/Mixtral-8x7B (b) GPT Family (c) Claude Family



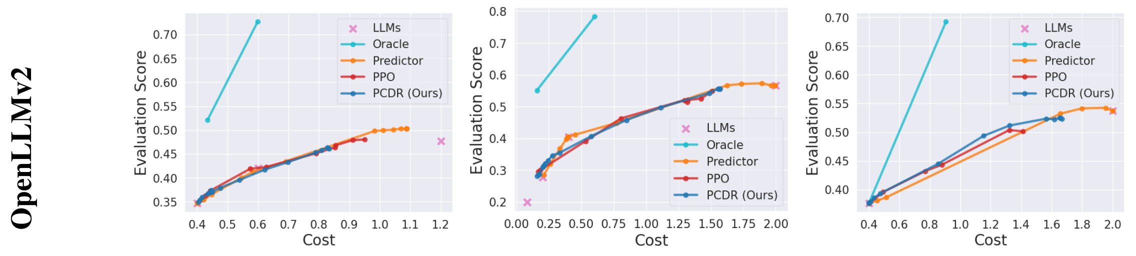
(a) GPT4/Mixtral-8x7B (b) Mistral Family (c) GPT Family



(a) GPT4/Mixtral-8x7B (b) Mistral Family (c) GPT Family

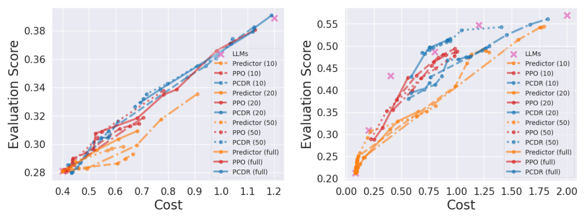


(a) Yi1.5 Family (b) Mistral Family (c) LLaMA3 Family



(a) Yi1.5 Family (b) Qwen2 Family (c) LLaMA3 Family

Figure 2: Evaluate the routing performance across 5 datasets and various sets of LLM candidates.



(a) Cohere (b) Qwen2.5

Figure 3: Evaluate routing performance on two sets of new models. the identity vectors are obtained using 10, 20 or 50 selected prompts, respectively.

routing policy trained on OpenLLMv2 (as shown in the last row of Figure 2), without further tuning on these newly added models. Figure 3 presents the evaluation results. Overall, our routing policy outperforms the *Predictor* baseline and performs comparably to the *PPO* policy, despite the latter being specifically trained on the new models. Additionally, our approach maintains effectiveness even with very limited evaluation data - using just 50 prompts achieves performance nearly matching that of identity vectors computed from the full set.

Computational Overhead The routing overhead is minimal compared to model inference time. Our policy requires approximately 5ms per routing decision on a single GPU, negligible compared to typical LLM inference times (100ms-1s). Memory requirements are also modest: the identity vectors and routing policy together require less than 100MB of GPU memory.

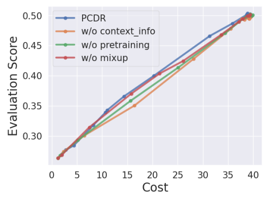


Figure 4: Ablation studies on the routing policy components.

Ablation Studies Our routing policy consists of a supervised pretraining stage followed by a RL training stage. During training, we incorporate on-manifold mixup regularization to improve generalization to unseen prompts. Additionally, our policy leverages the predicted scores $\hat{p}_k$ as contextual information. In this section, we perform ablation studies to assess the contributions of these components. Figure 4 presents the results when each component is removed. The results indicate that the context information, pretraining stage, and mixup regularization all contribute to learning a more effective routing policy.

5 Conclusion

In this work, we present a novel preference-conditioned dynamic routing framework for large language models that addresses three key challenges in LLM deployment: balancing performance and cost, adapting to user preferences, and incorporating new models efficiently. We formulate LLM routing as a multi-objective optimization problem and develop a preference-conditioned policy that dynamically adapts to user requirements at inference time. Our approach introduces a model identity vector framework that enables efficient integration of new LLMs without policy retraining, reducing adaptation time from hours to minutes. Through comprehensive experiments on five major benchmarks, we demonstrate significant improvements over existing methods, achieving up to 27% cost reduction while maintaining comparable performance.

Our results highlight the potential of intelligent routing systems in making LLM deployments more efficient and adaptable. However, several promising directions remain for future research. While our current approach operates in an offline setting with pre-computed evaluation scores, extending to online learning could improve policy robustness through real-time feedback. This would enable continuous adaptation to changing user needs and model performance patterns. Our framework currently assumes fixed costs per model, but real-world costs vary with input length and computation requirements. Developing adaptive cost models that account for query-specific characteristics could enable more precise optimization of the performance-cost trade-off.

Future work could also expand the routing capability to leverage external tools and API calls that many modern LLMs support. This could include incorporating tool use, online search results, and other augmentations into the routing decision process. Additionally, while our preference-based approach offers flexibility, expressing trade-offs through numerical parameters may not be intuitive for all users. Developing more natural interfaces for preference specification and automated preference learning from user feedback could improve usability.

As the LLM ecosystem continues to evolve with new models and capabilities, efficient routing systems will become increasingly critical for practical applications. Our framework provides a foundation for building more sophisticated, adaptive, and user-friendly LLM routing systems that can meet the diverse needs of real-world deployments.

Limitations

Our work on efficient LLM routing has several potential societal implications. On the positive side, by enabling more cost-efficient use of LLMs, our approach could help democratize access to advanced AI capabilities, allowing organizations with limited resources to leverage these technologies more effectively. The ability to balance performance and cost dynamically could make AI applications more sustainable and economically viable for a broader range of users.

However, this work also raises important considerations. By making LLM deployments more efficient, we could accelerate the adoption of these technologies, potentially exacerbating existing concerns about AI's impact on privacy, misinformation, and labor markets. Additionally, while our routing system aims to optimize resource allocation, it could inadvertently reinforce biases present in the underlying models if not carefully monitored.

To address these concerns, we emphasize that our framework is designed to be transparent in its decision-making process and configurable to align with organizational policies and ethical guidelines. We encourage users of this technology to carefully consider their specific use cases and implement appropriate safeguards, particularly when deploying in sensitive domains.

References

Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. 2019. Dynamic weights in multi-objective deep reinforcement learning. In *International conference on machine learning*, pages 11–20. PMLR.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rishabh Agarwal, Marlos C Machado, Pablo Samuel Castro, and Marc G Bellemare. 2021. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. *arXiv preprint arXiv:2101.05265*.

Toygun Basaklar, Suat Gumussoy, and Umit Y Ogras. 2022. Pd-morl: Preference-driven multi-objective reinforcement learning algorithm. *arXiv preprint arXiv:2208.07914*.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open llm leaderboard. https://huggingface.co/spaces/open-llm-leaderboard-old/open_llm_leaderboard.

Carolin Benjamins, Theresa Eimer, Frederik Schubert, Aditya Mohan, Sebastian Döhler, André Biedenkapp, Bodo Rosenhahn, Frank Hutter, and Marius Lindauer. 2022. Contextualize me–the case for context in reinforcement learning. *arXiv preprint arXiv:2202.04500*.

Djallel Bouneffouf and Irina Rish. 2019. A survey on practical applications of multi-armed and contextual bandits. *arXiv preprint arXiv:1904.10040*.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, and 1 others. 2022. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*.

Vinod Kumar Chauhan, Jiandong Zhou, Ping Lu, Soheila Molaei, and David A Clifton. 2023. A brief review of hypernetworks in deep learning. *arXiv preprint arXiv:2306.06955*.

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

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. 2019. Quantifying generalization in reinforcement learning. In *International conference on machine learning*, pages 1282–1289. PMLR.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks VS Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid llm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*.

Juncheng Dong, Hao-Lun Hsu, Qitong Gao, Vahid Tarokh, and Miroslav Pajic. 2023. Robust reinforcement learning through efficient adversarial herding. *arXiv preprint arXiv:2306.07408*.

Arpad E Elo. 1967. The proposed uscf rating system, its development, theory, and applications. *Chess life*, 22(8):242–247.

Jesse Farebrother, Marlos C Machado, and Michael Bowling. 2018. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*.

Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. 2024. Open llm leaderboard v2. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard.

Ronald K Hambleton and Hariharan Swaminathan. 2013. *Item response theory: Principles and applications*. Springer Science & Business Media.

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

Tyler Ingebrand, Amy Zhang, and Ufuk Topcu. 2024. Zero-shot reinforcement learning via function encoders. *arXiv preprint arXiv:2401.17173*.

Ayush Jain, Andrew Szot, and Joseph J Lim. 2020. Generalization to new actions in reinforcement learning. *arXiv preprint arXiv:2011.01928*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. *arXiv preprint arXiv:2306.02561*.

Michael N Katehakis and Arthur F Veinott Jr. 1987. The multi-armed bandit problem: decomposition and computation. *Mathematics of Operations Research*, 12(2):262–268.

George Konidaris and Finale Doshi-Velez. 2014. Hidden parameter markov decision processes: an emerging paradigm for modeling families of related tasks. In *2014 AAAI Fall Symposium Series*.

Ezgi Korkmaz. 2022. Deep reinforcement learning policies learn shared adversarial features across mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7229–7238.

Ezgi Korkmaz. 2024. A survey analyzing generalization in deep reinforcement learning. *arXiv preprint arXiv:2401.02349*.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. 2020. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019a. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR.

Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. 2019b. Network randomization: A simple technique for generalization in deep reinforcement learning. *arXiv preprint arXiv:1910.05396*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, and 1 others. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. Routing to the expert: Efficient reward-guided ensemble of large language models. *arXiv preprint arXiv:2311.08692*.

Xiaoding Lu, Adian Liusie, Vyas Raina, Yuwen Zhang, and William Beauchamp. 2024. Blending is all you need: Cheaper, better alternative to trillion-parameters llm. *arXiv preprint arXiv:2401.02994*.

Aman Madaan, Pranjal Aggarwal, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, and 1 others. 2023. Automix: Automatically mixing language models. *arXiv preprint arXiv:2310.12963*.

Luckeciano C Melo. 2022. Transformers are meta-reinforcement learners. In *international conference on machine learning*, pages 15340–15359. PMLR.

Janosch Moos, Kay Hansel, Hany Abdulsamad, Svenja Stark, Debora Clever, and Jan Peters. 2022. Robust reinforcement learning: A review of foundations and recent advances. *Machine Learning and Knowledge Extraction*, 4(1):276–315.

Quang H Nguyen, Duy C Hoang, Juliette Decugis, Saurav Manchanda, Nitesh V Chawla, and Khoa D Doan. 2024. Metallm: A high-performant and cost-efficient dynamic framework for wrapping llms. *arXiv preprint arXiv:2407.10834*.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*.

Simone Parisi, Matteo Pirotta, Nicola Smacchia, Luca Bascetta, and Marcello Restelli. 2014. Policy gradient approaches for multi-objective sequential decision making. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 2323–2330. IEEE.

John Platt and 1 others. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

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

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, and 1 others. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Marija Šakota, Maxime Peyrard, and Robert West. 2024. Fly-swat or cannon? cost-effective language model choice via meta-modeling. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 606–615.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shubhkirti Sharma and Vijay Kumar. 2022. A comprehensive review on multi-objective optimization techniques: Past, present and future. *Archives of Computational Methods in Engineering*, 29(7):5605–5633.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2023. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*.

Tianye Shu, Ke Shang, Cheng Gong, Yang Nan, and Hisao Ishibuchi. 2024. Learning pareto set for multi-objective continuous robot control. *arXiv preprint arXiv:2406.18924*.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, and 1 others. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Remi Tachet, Philip Bachman, and Harm van Seijen. 2018. Learning invariances for policy generalization. *arXiv preprint arXiv:1809.02591*.

Ahmed Touati and Yann Ollivier. 2021. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23.

Kristof Van Moffaert and Ann Nowé. 2014. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512.

Hongyi Wang, Felipe Maia Polo, Yuekai Sun, Souvik Kundu, Eric Xing, and Mikhail Yurochkin. 2023. Fusing models with complementary expertise. *arXiv preprint arXiv:2310.01542*.

Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. 2020. Improving generalization in reinforcement learning with mixture regularization. *Advances in Neural Information Processing Systems*, 33:7968–7978.

Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. 2020. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *International conference on machine learning*, pages 10607–10616. PMLR.

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. 2019. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *Advances in neural information processing systems*, 32.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. 2021. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International conference on learning representations*.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. 2020. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*.

Hanping Zhang and Yuhong Guo. 2021. Generalization of reinforcement learning with policy-aware adversarial data augmentation. *arXiv preprint arXiv:2106.15587*.

Hongyi Zhang. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, and Jiantao Jiao. 2023. Starling-7b: Improving llm helpfulness & harmlessness with rlaif.

A Theoretical Analysis

A.1 Continuity and Existence of Optimal Policies

Consider a policy π_θ parameterized by $\theta \in \Theta$, where Θ is a compact subset of $\mathbb{R}^d$. Let $s(x, k) \in \{0, 1\}$ be a binary reward function and $p(x)$ be the query distribution.

Theorem A.1. *If the policy $\pi_\theta(k|x)$ is continuous in θ for all x and k, then the expected reward $J(\theta) = \mathbb{E}_{x\sim p(x),k\sim\pi_\theta(x)}[s(x, k)]$ is continuous in θ.*

Proof. For any $\theta, \theta' \in \Theta$:

$$
\begin{aligned}
&|J(\theta) - J(\theta')| \\
&= \left| \int_{\mathcal{X}} \sum_k s(x,k)(\pi_\theta(k|x) - \pi_{\theta'}(k|x))p(x)dx \right| \\
&\leq \int_{\mathcal{X}} \sum_k |s(x,k)||\pi_\theta(k|x) - \pi_{\theta'}(k|x)|p(x)dx \\
&\leq \int_{\mathcal{X}} \sum_k |\pi_\theta(k|x) - \pi_{\theta'}(k|x)|p(x)dx
\end{aligned}
$$

Since $\pi_\theta(k|x)$ is continuous in θ, for any $\epsilon > 0$, there exists $\delta > 0$ such that $\|\theta - \theta'\| < \delta$ implies $|\pi_\theta(k|x) - \pi_{\theta'}(k|x)| < \epsilon/K$ for all k and x. Therefore, $|J(\theta) - J(\theta')| < \epsilon$ when $\|\theta - \theta'\| < \delta$, proving continuity. □

Corollary A.2. *For any preference vector $\boldsymbol{\omega}$, there exists an optimal policy $\pi_{\boldsymbol{\omega}}$ that maximizes the expected scalarized reward.*

This follows from the extreme value theorem, as we are maximizing a continuous function over a compact set. For the relationship between the preference vectors and the Pareto front, we refer readers to Yang et al. (2019) who provide a detailed analysis in the context of multi-objective reinforcement learning.

B Method

B.1 Model Identity Vector

We learn the model identity vector $\mathbf{I}_k$ following a variational variant of the IRT model. Given evaluation scores $Y_k = \{y_{kn}\}_{n=1}^N$ for model M_k on a set of prompts $X = \{x_n\}_{n=1}^N$, we maximize the following variational lower bound of the log-likelihood:

$$
\begin{aligned}
&\log p(y_{kn} \mid x_n) \\
&= \log \int p(y_{kn}, \mathbf{I}_k \mid x_n) d\mathbf{I}_k \\
&\geq \mathbb{E}_{q(\mathbf{I}_k)} [\log p(y_{kn} \mid x_n, \mathbf{I}_k)] - D_{\mathrm{KL}}(q(\mathbf{I}_k) \| p(\mathbf{I}_k)).
\end{aligned}
$$

Here, the model embedding $\mathbf{I}_k$ is treated as a latent variable, with the posterior and prior distributions over $\mathbf{I}_k$ denoted by $q(\mathbf{I}_k)$ and $p(\mathbf{I}_k)$, respectively. In practice, both distributions are modeled as Gaussians, with the posterior $q(\mathbf{I}_k) = \mathcal{N}(\mu_k, \Sigma_k)$ and the prior $p(\mathbf{I}_k) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. The posterior mean μ_k and variance Σ_k are represented as embedding vectors of dimension d, with the variance assumed to be diagonal. The predictive distribution $p(y_{kn} \mid x_n, \mathbf{I}_k)$ is implemented as a neural network that concatenates of prompt and model embeddings as input and outputs the score prediction logits. During training, the loss is computed over the entire evaluation benchmarks, involving multiple prompts and models, i.e., $-\mathbb{E}_{x,k} \log p(y_{kn} \mid x_n)$.

B.1.1 Training with Real-valued Evaluation Scores

Certain evaluation datasets produce real-valued evaluation scores, such as F1 and RougeL. In order to unify the training procedure, we propose to binarize the real-valued scores. Specifically, given a set of real-valued scores $Y = \{y_n\}_{n=1}^N$, where $y_n \in [0, 1]$, we find an optimal threshold η^* so that the average performance across instances are close to the original scores, that is

$$
\eta^* = \arg\min_\eta \left(\frac{1}{N} \sum_{n=1}^N \mathbb{I}(y_n > \eta) - \frac{1}{N} \sum_{n=1}^N y_n \right)^2,
$$

where $\mathbb{I}(y_n > \eta)$ is the indicator function, which equals to 1 only when the condition $y_n > \eta$ is true. Therefore, the binarized evaluation scores are derived as $\bar{Y} = \{\mathbb{I}(y_n > \eta^*)\}_{n=1}^N$.

B.2 Preference Conditioned Routing Policy

In the main text, we derived the routing policy as

$$
\begin{aligned}
&\pi_\theta(k' \mid x, \{(\mathbf{I}_k, c_k, \hat{p}_k)\}_{k=1}^K, \boldsymbol{\omega}) \\
&\propto I_{k'}^T h(x, \{(\mathbf{I}_k, c_k, \hat{p}_k)\}_{k=1}^K, \boldsymbol{\omega}),
\end{aligned}
$$

where $h(\cdot)$ is a neural network that is permutation invariant to the set $\{(\mathbf{I}_k, c_k, \hat{p}_k)\}_{k=1}^K$. We achieve the permutation invariance by using a permutation invariant embeddings of the set, implemented via the SetTransformer architecture(Lee et al., 2019a).

The prompt x is encoded using pretrained prompt embeddings. The preference vector $\boldsymbol{\omega}$ is projected through a linear layer for integration into the routing policy. The neural network then concatenates the embeddings and passes them through several linear layers, resulting in a vector representation in $\mathbb{R}^d$. The inner product between $h(\cdot)$ and each model embedding $I_{k'}$ determines which model to select based on the policy. Specifically, the routing probability for selecting model $M_{k'}$ follows the softmax distribution:

$$\pi_\theta(k' \mid x, \{(\mathbf{I}_k, c_k, \hat{p}_k)\}_{k=1}^K, \boldsymbol{\omega}) = \frac{\exp\left(I_{k'}^T h(x, \{(\mathbf{I}_k, c_k, \hat{p}_k)\}_{k=1}^K, \boldsymbol{\omega})\right)}{\sum_{k''=1}^K \exp\left(I_{k''}^T h(x, \{(\mathbf{I}_k, c_k, \hat{p}_k)\}_{k=1}^K, \boldsymbol{\omega})\right)}.$$

We train the routing policy following the multi-objective PPO algorithm, where the gradient for updating the policy parameters θ is given by

$$\nabla_\theta[\boldsymbol{\omega}^T \mathbf{J}_{\pi_\theta}] = \mathbb{E}_{x,k'}\left[\boldsymbol{\omega}^T \mathbf{A}(x,k') \nabla_\theta \log \pi_\theta(k' \mid \cdot)\right],$$

where $\mathbf{A}(x,k')$ indicates the advantage function estimated via GAE (Schulman et al., 2015). The PPO algorithm also requires a value estimation to reduce the gradient variance. Following multi-objective RL literature (Xu et al., 2020; Shu et al., 2024), we define a value network $\mathbf{V}_{\pi_\theta}(x, \{(\mathbf{I}_k, c_k, \hat{p}_k)\}_{k=1}^K)$ that outputs a vector of expected returns under the current policy π_θ. The value estimation is not conditioned on the preference, therefore, it can be shared across different user preferences. We train the values network by optimizing a MSE loss $\|\mathbf{V}_{\pi_\theta} - \mathbf{V}_{targ}\|^2$, where $\mathbf{V}_{targ}$ indicates the target values estimated via GAE.

B.3 Generalization of the Routing Policy

In this section, we discuss the training procedure of the dynamic routing policy, which is designed to enhance the generalizability of the policy to various scenarios.

B.3.1 Supervised Pretraining

The supervised pretraining stage leverages diverse prompts from pairwise comparison datasets to enhance generalization to unseen prompts. Given a pairwise comparison dataset $\mathcal{V}$, where each example consists of a prompt x_n, a pair of models M_{k1} and M_{k2}, and a winning label $z_n \in \{0,1\}$, we first train a logistic regression model to calibrate the predicted evaluation scores, $\hat{p} = \text{sigmoid}(f(x, \mathbf{I}_k))$, using the winning label z_n. Specifically, the logical regression model predicts the wining probability as $p(z_n = 1) = \text{sigmoid}(\alpha(f(x_n, I_{k1}) - f(x_n, I_{k2})) + \beta)$, where α and β are learnable parameters. After training, the calibrated evaluation scores are given by $\bar{p} = \text{sigmoid}(\alpha f(x, \mathbf{I}_k) + \beta)$. The calibration follows the well-known Platt scaling (Platt et al., 1999) algorithm, which refines the evaluation scores using human-labeled winning labels to produce more accurate predictions.

With the calibrated evaluation scores $\bar{p}$ on a prompt x and a user preference vector $\boldsymbol{\omega}$, the routing action is determined by $\hat{a} = \arg\max_{k \in \{k1,k2\}} \boldsymbol{\omega}^T[\bar{p}_k, -c_k]$. We then pretrain the routing policy in a supervised manner using the following negative log-likelihood loss:

$$\mathcal{L}_{pretrain} = -\log \pi(\hat{a} \mid x, \{(\mathbf{I}_k, c_k, \hat{p}_k)\}_{k \in \{k1,k2\}}, \boldsymbol{\omega}). \quad \text{(B.1)}$$

It is important to note that the policy utilizes the original predicted scores $\hat{p}$ as input, rather than the calibrated scores, to maintain consistency with the subsequent RL training stage.

B.3.2 On-Manifold Mixup Regularization

The mixup regularization technique was initially introduced for supervised learning tasks (Zhang, 2017), where new input-output pairs are generated by taking convex combinations of pairs of training samples. Wang et al. (2020) extended this approach to RL, where observations and their associated supervision signals from two transitions are combined convexly. In our case, the observation corresponds to the prompt embeddings. However, naively combining two prompt embeddings may produce vectors that lie outside the prompt manifold. To address this, we use the nearest neighbor from the replay buffer for each prompt x. Given the embedding e for prompt x and the embedding e_n for its nearest neighbor, the interpolated prompt embedding is obtained as:

$$\hat{e} = \lambda e + (1-\lambda) e_n, \quad \text{(B.2)}$$

where $\lambda \sim \text{Beta}(\xi, \xi)$, and ξ is a hyperparameter, set to 0.2 as recommended in the original mixup paper. To train the routing policy on the interpolated prompt embeddings using PPO, we similarly interpolate the associated supervision signals:

$$\begin{aligned} \hat{\pi}_{old} &= \lambda \pi_{old} + (1-\lambda)\pi_{old}^{(n)} \\ \hat{\mathbf{A}} &= \lambda \mathbf{A} + (1-\lambda)\mathbf{A}_n \\ \hat{\mathbf{V}}_{targ} &= \lambda \mathbf{V}_{targ} + (1-\lambda)\mathbf{V}_{targ}^{(n)} \end{aligned} \quad \text{(B.3)}$$

The interpolated routing action $\hat{a}$ is chosen as a if $\lambda > 0.5$, otherwise a_n. Similarly, routing-relevant parameters, including $\mathbf{I}_k$ and $\boldsymbol{\omega}$ are chosen based on λ as well.

B.3.3 Reward Normalization

Our routing policy is designed to generalize across different sets of LLM candidates. However, the varying score and cost scales across these sets can pose challenges. For instance, routing decisions involving proprietary API models often involve higher costs compared to open-source models, where the cost is significantly lower. These discrepancies in scale can complicate the training of the routing policy, as the preference vector must be adjusted to suit each scenario. Moreover, the same preference vector might favor higher costs for one set of models while preferring lower costs for another, introducing inconsistency and instability during training. To address this, we propose normalizing both the scores and costs across all LLM sets. Given a set of LLMs $\{M_k\}_{k=1}^K$ with scores $\{s_k\}_{k=1}^K$ and costs $\{c_k\}_{k=1}^K$, we normalize the scores and costs by

$$\begin{aligned} \bar{s}_k &= s_k / \max(\{s_k\}_{k=1}^K), \\ \bar{c}_k &= c_k / \max(\{c_k\}_{k=1}^K). \end{aligned} \tag{B.4}$$

This normalization ensures that both scores and costs are scaled such that their maximum value is 1.0. By standardizing the range of values, the policy can learn a consistent mapping from user preferences to routing decisions across various LLM sets. This approach prevents the policy from disproportionately favoring either high-cost or low-cost models based purely on their relative scales, promoting more balanced decisions that accurately reflect trade-offs between performance and cost.

In theory, the preference vector $\boldsymbol{\omega}$ can take any value in the range of $[0, \infty)$. However, for simplicity, we define it as $\boldsymbol{\omega} = [1, \omega]$, fixing the preference weight for scores at 1 and only varying the weight for cost. When $\omega = 0$, the model selection prioritizes high scores regardless of cost, while $\omega = \infty$ indicates a preference for the lowest-cost model. In practice, we found that sampling ω from the range $[0, 2]$ effectively captures the Pareto front.

B.3.4 Stratified Sampling

Generalizing the routing policy to a new model $\tilde{M}$ requires to obtain its identity vector $\tilde{I}$, which captures the model's unique strengths and weaknesses. However, evaluating the model on all available prompts is often prohibitively expensive, especially when new models are frequently introduced. In order to reduce the evaluation cost, we propose selecting a subset of informative prompts that effectively assess the model's capabilities. Specifically, given a set of prompts $X = \{x_n\}_{n=1}^N$ and the binarized evaluation scores $Y_k = \{\bar{y}_{kn}\}_{n=1}^N$ for each available LLM M_k, we assess the difficulty of each prompt based on the average prediction accuracy across all models M_k, i.e.,

$$\psi_n = \mathbb{E}_k \left[-\bar{y}_{kn} \log p_{kn} - (1 - \bar{y}_{kn}) \log(1 - p_{kn}) \right].$$

We then apply stratified sampling using the difficulty ψ_n as the strata. The stratified sampling ensures the selected prompts covers a range of difficulties, from easy to hard, providing a more balanced and informative assessment of the model's strengths and weaknesses. Once the subsets $\tilde{X}$ is selected, the model identity vector is computed as:

$$\tilde{I} = \arg\min_I \left[\mathbb{E}_{\tilde{x}} \mathcal{L}_{irt} + D_{\mathrm{KL}} \left(q(\tilde{I}) \| p(\tilde{I}) \right) \right],$$

where

$$\begin{aligned} &\mathcal{L}_{irt} = -\bar{y} \log p - (1 - \bar{y}) \log(1 - p), \\ &p = \mathrm{sigmoid}(f(\tilde{e}, \tilde{I})), \end{aligned}$$

and $\tilde{e}$ is the prompt embedding for prompts $\tilde{x} \in \tilde{X}$.

The stratified sampling approach described above can also be extended to sample prompts from pairwise comparison datasets. Given a pairwise comparison dataset $\mathcal{V}$, where each example consists of a prompt x_n, a pair of models M_{k1} and M_{k2}, and a winning label $z_n \in \{0, 1\}$. We first assess each model's capability using Elo score (Elo, 1967). The Elo scores are then used as strata to sample a set of models as the comparison baselines. For each baseline M_k, we uniformly select a set of prompts X_k on which to run inference with the new model $\tilde{M}$ and compare its performance to the baseline M_k. After obtaining the baseline models and pairwise comparison labels, the model identity vector is computed as:

$$\tilde{I} = \arg\min_I \mathcal{L}_{pair} + \mathcal{L}_{KL}. \tag{B.5}$$

In our experiments, we opted to sample prompts from existing evaluation benchmarks for simplicity. We leave the exploration of sampling from pairwise comparison datasets as future work.

C Training Algorithm

Algorithm 1 Training Procedure of Preference Conditioned Dynamic Routing

```
Require: Model identify vectors $I$, Pretraining steps $T1$, Training steps $T2$
Require: Comparison dataset $\mathcal{V}$ for pretraining, Evaluation leaderboard $\mathcal{D}$ for training
Require: Evaluation score predictor $f(x, \mathbf{I}_k)$, Calibration parameters $\alpha$ and $\beta$
Require: Routing policy $\pi_\theta$, Preference range $[\omega_{min}, \omega_{max}]$, RL training procedure $\mathbb{P}$
 1: ## Pretraining Stage
 2: for step in $[1, \dots, T1]$ do
 3:     sample a batch of pretraining data $(x, (k1, k2), (c1, c2)) \sim \mathcal{V}$
 4:     sample a batch of preference $\boldsymbol{\omega} = [1, \omega]$ and $\omega \sim U(\omega_{min}, \omega_{max})$ {uniform for cost}
 5:     compute score predictions $\hat{p}_k = \text{sigmoid}(f(x, \mathbf{I}_k))$ {auxiliary info to the policy}
 6:     calibrate the score predictions $\bar{p}_k = \text{sigmoid}(\alpha f(x, \mathbf{I}_k) + \beta)$ {only used to predict action}
 7:     normalize scores $\bar{p}_k = \bar{p}_k / \max(\{\bar{p}_k\}_{k \in \{k1,k2\}})$ and costs $\bar{c}_k = c_k / \max(\{c_k\}_{k \in \{k1,k2\}})$
 8:     obtain routing action $\hat{a} = \arg\max_{k \in \{k1,k2\}} \boldsymbol{\omega}^T [\bar{p}_k, -\bar{c}_k]$ {maximize scalarized reward}
 9:     pretrain the policy by optimizing $-\log \pi(\hat{a} \mid x, \{(\mathbf{I}_k, \bar{c}_k, \hat{p}_k)\}_{k \in \{k1,k2\}}, \boldsymbol{\omega})$
10: end for
11: ## Training Stage
12: Initialize a replay buffer $\mathbb{B}$ {with on-manifold mixup regularization}
13: for step in $[1, \dots, T2]$ do
14:     sample a batch of training data $(x, \{(M_k, c_k, s_k)\}_{k=1}^K) \sim \mathcal{D}$ {$K$ is different across batches}
15:     sample a batch of preference $\boldsymbol{\omega} = [1, \omega]$ and $\omega \sim U(\omega_{min}, \omega_{max})$ {uniform for cost}
16:     normalize scores $\bar{s}_k = s_k / \max(\{s_k\}_{k=1}^K)$ and costs $\bar{c}_k = c_k / \max(\{c_k\}_{k=1}^K)$
17:     compute score predictions $\hat{p}_k = \text{sigmoid}(f(x, \mathbf{I}_k))$ {auxiliary info to the policy}
18:     run the current policy $a \sim \pi(x, \{(\mathbf{I}_k, \bar{c}_k, \hat{p}_k)\}_{k=1}^K, \boldsymbol{\omega})$ and obtain reward $[\bar{s}_a, -\bar{c}_a]$
19:     update replay buffer $\mathbb{B} \leftarrow (x, a, \{(M_k, \bar{c}_k, \bar{s}_k)\}_{k=1}^K, \boldsymbol{\omega})$
20:     RL training on data sampled from the replay buffer $\mathbb{P}(\pi_\theta, \mathbb{B})$ {with mixup interpolation}
21: end for
```

D Additional Related Works

LLM Ensemble, Cascade and Routing As the number of LLMs grows, there is increasing interest in combining them to optimize performance and balance costs. LLM ensemble methods improve response quality by aggregating outputs from multiple LLMs but incur high computational costs since they require running inference on multiple models (Jiang et al., 2023; Wang et al., 2023; Lu et al., 2024). LLM cascading reduces costs by invoking LLMs sequentially, starting with the least expensive model and progressing to more costly ones until a satisfactory response (Chen et al., 2023; Madaan et al., 2023; Ramírez et al., 2024). While effective in reducing costs, cascading still requires multiple inferences, especially for complex queries, and often depends on an additional model to assess the response quality.

In contrast, LLM routing sends queries directly to the most appropriate model, requiring only a single inference and thus offering a more cost-efficient solution. Typical routing methods rely on performance prediction models to guide the selection of LLM. These methods either predict downstream evaluation or reward scores for a given query (Shnitzer et al., 2023; Lu et al., 2023; Hari and Thomson, 2023; Šakota et al., 2024), or estimate win rates between pairs of models (Ding et al., 2024; Ong et al., 2024). The chosen LLM is then selected based on predicted performance and any additional constraints, such as cost or latency.

The most relevant work to ours is MetaLLM (Nguyen et al., 2024), which also frames the routing task as a multi-armed bandit problem. However, MetaLLM optimizes a scalarized reward and operates on a fixed set of LLMs, limiting the learned policy to specific user preferences and a predefined set of models. Our approach, by contrast, generalizes to varied user preferences and dynamically adapts to new LLMs added to the system, ensuring broader applicability and greater flexibility.

Multi-objective Reinforcement Learning Multi-objective RL seeks to optimize multiple, often conflicting reward signals within a Markov decision process, resulting in a set of Pareto-optimal policies known as the Pareto set rather than a single optimal policy. Traditional algorithms typically aim to approximate this Pareto set by searching for a finite number of policies (Van Moffaert and Nowé, 2014; Parisi et al., 2014; Xu et al., 2020). However, these methods face the curse of dimensionality, where the number of policies needed to accurately approximate the Pareto set grows exponentially with the number of objectives. To address this, recent approaches have proposed using a single deep neural network conditioned on preferences to represent the entire Pareto set (Yang et al., 2019; Abels et al., 2019; Basaklar et al., 2022). Another approach involves using hypernetworks (Chauhan et al., 2023), which map user preferences to the parameters of the policy network (Shu et al., 2024). Our routing policy aligns with the conditional neural network framework, where a single model is conditioned on user preferences to adapt to different user requirements. We further tailor this conditional architecture specifically for routing LLMs, allowing for efficient decision-making across a diverse and expanding set of models.

Generalization in Reinforcement Learning Generalizing RL policies to new tasks, often referred to as zero-shot RL, is a growing area of research focused on enabling policies to handle unseen tasks without retraining (Korkmaz, 2024). Approaches typically fall into three categories: The first category focuses on maximizing worst-case performance across tasks, often using adversarial training (Moos et al., 2022; Dong et al., 2023). This approach is commonly used when no data is available to identify the current task. The second category aims to compute task representations from data, allowing agents to adapt their policies to the specific task at hand. This approach is commonly employed in multi-task RL and hidden-parameter MDPs (Konidaris and Doshi-Velez, 2014), where task representations are inferred from exploration data within the task environment (Touati and Ollivier, 2021; Agarwal et al., 2021; Benjamins et al., 2022; Ingebrand et al., 2024). The third category leverages in-context learning by feeding data from the current task directly into a pretrained transformer as context (Melo, 2022; Brohan et al., 2022). Although transformers have demonstrated effectiveness, their high memory consumption, training instability, and data inefficiency present challenges to their broader application. Our routing policy falls into the second category, where the task representation is explicitly provided as a set of LLMs and their associated costs. In a similar vein, Jain et al. (2020) explore RL generalization to new action spaces using a VAE to learn action representations, whereas we capture LLM capabilities via identity vectors.

In addition to task generalization, research has also explored generalizing RL policies to new observation distributions. Techniques include data augmentation (Cobbe et al., 2019; Yarats et al., 2021; Laskin et al., 2020), specialized architectures (Lee et al., 2019b), regularization methods (Farebrother et al., 2018; Wang et al., 2020), invariant representation learning (Tachet et al., 2018; Zhang et al., 2020; Agarwal et al., 2021), and adversarial observation perturbations (Zhang and Guo, 2021; Korkmaz, 2022). Our approach explores a simple regularization technique that encourage smoothness across prompt distributions.

E Experiment

E.1 Model Cost

In Table E.1, we list the costs for each model. For proprietary APIs, the costs are based on their official API pricing, while for open-source models, we reference pricing from TogetherAI[1]. All costs are normalized by estimating the expense of processing 1 million input tokens and generating 1 million output tokens.

E.2 Dataset Statistics

Our framework consists of three training stages: First, we train the IRT model to obtain model identity vectors I and the evaluation score prediction model f. Second, we perform supervised pretraining of the routing policy on diverse prompts. Third, we train the routing policy using a reinforcement learning procedure. Below, we summarize the datasets used in each training stage.

The datasets used in this work fall into two categories: First, pairwise comparison datasets, where annotations indicate which of two models provides a higher-quality response. Second, LLM evaluation datasets, which provide evaluation scores for various models on a set of prompts. Table E.2 summarizes the statistics of these two types of datasets. We apply basic preprocessing, such as removing multi-turn prompts and excluding ties from pairwise comparisons. For LLM evaluation benchmarks, we select a subset of popular LLMs. Please see Table E.3 for the full list of LLMs involved in this work.

[1]https://www.together.ai/pricing

Table E.1: The estimated cost of invoking the models for processing 1M input tokens and generating 1M output tokens.

Model	Cost ($)
gpt-3.5-turbo-0125	2
gpt-3.5-turbo-0301	3.5
gpt-3.5-turbo-0613	3.5
gpt-3.5-turbo-1106	3
gpt-4-0125-preview	40
gpt-4o-2024-05-13	20
gpt-4o-mini-2024-07-18	0.75
gpt-4	90
gpt-4-1106-preview	40
gpt-4-turbo-2024-04-09	40
gpt-4-turbo	40
claude-3-opus	90
claude-3.5-sonnet	18
claude-3-sonnet	18
claude-3-haiku	1.5
claude-2.1	32
claude-2	32
claude-instant	3.2
claude-1	32
gemini-pro-1.5	14
gemini-flash-1.5	0.375
llama3.1-405b	9
llama3.1-70b	1.584
llama3.1-8b	0.324
llama3-70b	1.584
llama3-8b	0.324
mistral-large	12
mistral-medium	10.8
mistral-small	8
mixtral-8x22b	2.16
mixtral-8x7b	1.08
mixtral-7b	0.36
command-r-plus	18
command-r	2
command	3
command-light	0.9
qwen-1.5-110b	3.24
qwen-1.5-72b	1.62
yi-large	6

The IRT model is trained using the pairwise comparison datasets and the training splits of the evaluation datasets. The pretraining stage also uses these pairwise comparison datasets. For the policy training stage, the routing policy is trained separately on each LLM evaluation dataset. We do not train the policy across different evaluation benchmarks, as they employ different scoring mechanisms, leading to variations in score scales.

E.3 Training the IRT Model

The IRT model for evaluation outcome prediction consist of four component: the prompt representation e, the model identity vector I, the evaluation score predictor $f(e, I)$, and the pairwise winner predictor $g(e, I)$. The prompt representation e is obtained using a pretrained text embedding model, meaning it contains no learnable parameters. The model identity vector is initialized as random embeddings for each model listed in Table E.3, with the embedding dimension set to 128. The two neural networks, f and g, share a common backbone, differing only in their final linear layer. This shared architecture encourages the model identity vector to capture both types of evaluation outcomes, enabling more accurate representation of each model's strengths and weaknesses.

The IRT model is trained using both the pairwise comparison datasets and the training splits of the evaluation datasets, with a combined loss function, $\mathcal{L}_{irt} + \mathcal{L}_{pair}$. The model is trained for 10 epochs with a batch size of 256. We use the Adam optimizer with a learning rate of 0.001. The learning rate is decayed by 0.95 after each epoch. We did not conduct extensive hyperparameter tuning, and no signs of overfitting were observed during preliminary experiments. Additionally, training beyond 10 epochs did not lead to further improvements in validation performance and downstream routing performance.

E.4 Supervised Pretraining of the Routing Policy

The supervised pretraining stage for the routing policy optimizes the following negative log-likelihood $-\log \pi(\hat{a} \mid x, \{(\mathbf{I}_k, \bar{c}_k, \hat{p}_k)\}_{k\in\{k1,k2\}}, \boldsymbol{\omega})$ using the pairwise comparison dataset $\mathcal{V}$. Given two models M_{k1} and M_{k2} compared on the prompt x, we first sample a preference vector $\boldsymbol{\omega} = [1, \omega]$, where ω is uniformly sampled from the predefined distribution $U(\omega_{min}, \omega_{max})$. The routing decision is then estimated as $\hat{a} =$

Table E.2: Two types of datasets used in the training process.

Category	Dataset	# Prompts	# Models
Pairwise Model Comparison	berkeley-nest/Nectar[2]	182954	39
	lmsys/lmsys-arena-human-preference-55k[3]	39716	64
	lmsys/chatbot_arena_conversations[4]	18320	20
	lmsys/mt_bench_human_judgments[5]	894	6
	routellm/gpt4_judge_battles[6]	84864	2
Single Model Evaluation	AlpacaEval 2.0[7]	805	61
	HELM-Lite[8]	13021	61
	HELM-MMLU[9]	14042	45
	OpenLLM Leaderboard[10]	14617	41
	OpenLLM Leaderboard v2[11]	21606	39

Table E.3: The models used in this work for training and evaluating the routing policy.

ai21_j2-grande	ai21_j2-jumbo	ai21_jamba-instruct	alpaca-7b
alpaca-13b	chatglm-6b	chatglm2-6b	chatglm3-6b
claude-1	claude-2.0	claude-2.1	claude-instant-1
claude-instant-1.2	claude-3-5-sonnet-20240620	claude-3-opus-20240229	claude-3-sonnet-20240229
claude-3-haiku-20240307	cohere_command-r	cohere_command	cohere_command-r-plus
cohere_command-light	cohere_command-xlarge	codellama-7b-instruct	codellama-13b-instruct
codellama-34b-instruct	codellama-70b-instruct	deepseek-llm-67b-chat	dolly-v2-12b
dolphin-2.2.1-mistral-7b	dialogpt-large	falcon-180b-chat	falcon-40b-instruct
falcon-7b-instruct	fastchat-t5-3b	flat-t5-small	gemini-1.0-pro
gemini-1.5-pro	gemini-1.5-flash	gemini-pro-dev-api	gemma-2-9b-it
gemma-2-27b-it	gemma-2b-it	gemma-7b-it	recurrentgemma-2b-it
recurrentgemma-9b-it	google-text-unicorn	google-text-bison	gpt2
gpt2-large	gpt2-medium	gpt2-xl	gpt-3.5-turbo-0125
gpt-3.5-turbo-0314	gpt-3.5-turbo-0613	gpt-3.5-turbo-1106	gpt-4-0125-preview
gpt-4	gpt-4-0314	gpt-4-0613	gpt-4-1106-preview
gpt-4-turbo-2024-04-09	gpt-4o-2024-05-13	gpt-4o-mini-2024-07-18	gpt4all-13b-snoozy
guanaco-13b	guanaco-33b	guanaco-65b	guanaco-7b
koala-13b	llama-13b	llama-65b	llama-2-13b-chat
llama-2-70b-chat	llama-2-7b-chat	llama2-70b-steerlm-chat	llama-3-70b-instruct
llama-3-8b-instruct	llama-3.1-405b-instruct-turbo	llama-3.1-70b-instruct-turbo	llama-3.1-8b-instruct-turbo
luminous-base	luminous-supreme	luminous-extended	mamba-gpt-7b-v2
metamath-13b	metamath-70b	mistral-7b-instruct-v0.1	mistral-7b-instruct-v0.2
mistral-7b-instruct-v0.3	mistral-large	mistral-medium	mistral-small
mixtral-8x7b-instruct-v0.1	mixtral-8x22b-instruct-v0.1	mpt-30b-chat	mpt-7b-chat
nous-hermes-2-mixtral-8x7b-dpo	oasst-pythia-12b	opt-1.3b	opt-2.7b
opt-350m	opt-6.7b	opt-iml-max-1.3b	opt-iml-max-30b
openchat-3.5	openchat-3.5-0106	openhermes-2.5-mistral-7b	phi-2
phi-2-dpo	phi-2-sft	phi-3-medium	phi-3-small
phi-3-mini	palm-2	pythia-12b	pplx-70b-online
pplx-7b-online	palmyra-x-v3	palmyra-x-v2	qwen-14b-chat
qwen1.5-0.5b-chat	qwen1.5-1.8b-chat	qwen1.5-4b-chat	qwen1.5-72b-chat
qwen1.5-14b-chat	qwen1.5-32b-chat	qwen1.5-7b-chat	qwen1.5-110b-chat
qwen1.5-moe-a2.7b-chat	qwen2-0.5b-instruct	qwen2-1.5b-instruct	qwen2-7b-instruct
qwen2-72b-instruct	rwkv-4-raven-1b5	rwkv-4-raven-3b	rwkv-4-raven-7b
rwkv-4-raven-14b	solar-10.7b-instruct-v1.0	stablelm-tuned-alpha-7b	starling-lm-7b-alpha
stripedhyena-nous-7b	text_davinci_001	text_davinci_002	text_davinci_003
tulu-2-dpo-7b	tulu-2-dpo-13b	tulu-2-dpo-70b	ultralm-13b
ultralm-65b	vicuna-13b	vicuna-33b	vicuna-7b
wizardlm-7b	wizardlm-13b	wizardlm-70b	yi-6b-chat
yi-34b-chat	yi-large	yi1.5-6b-chat	yi1.5-9b-chat
yi1.5-34b-chat	zephyr-7b-alpha	zephyr-7b-beta	

$\arg\max_{k\in\{k1,k2\}} \boldsymbol{\omega}^T[\bar{p}_k, -\bar{c}_k]$, where $\bar{c}_k$ represents the normalized cost $\bar{c}_k = c_k / \max(c_{k1}, c_{k2})$, and $\bar{p}_k$ represent the calibrated evaluation score $\bar{p}_k = \text{sigmoid}(\alpha f(x, \mathbf{I}_k) + \beta)$. The evaluation scores are further normalized by dividing by the maximum calibrated scores, ensuring consistency with the scale used in the RL training stage. Note that these calibrated and normalized scores are used only for routing action estimation during pretraining, the policy takes in the original score prediction $\hat{p}_k = \text{sigmoid}(f(x, \mathbf{I}_k))$ as auxiliary inputs, since the normalized scores are not available during the test phase.

The pretraining stage runs for 500 steps with a batch size of 1024, using the Adam optimizer with a learning rate of 0.001. The calibration parameters, α and β, are learned by fitting a logistic regression model. Again, we did not conduct extensive hyperparameter tuning, further tuning may improve the performance.

E.5 RL Training of the Routing Policy

The RL training stage follows a modified PPO procedure tailored for the multi-objective optimization task. Specifically, from the evaluation leaderboard $\mathcal{D}$, we sample K models, $\{M_k\}_{k=1}^K$, as routing candidates. The costs c_k of these models are normalized by $\bar{c}_k = c_k / \max(\{c_k\}_{k=1}^K)$, and the their evaluation scores s_k are normalized by $\bar{s}_k = s_k / \max(\{s_k\}_{k=1}^K)$. The user preference $\boldsymbol{\omega} = [1, \omega]$ and ω is sampled from the distribution $U(\omega_{min}, \omega_{max})$. The training process starts with generating the trajectories following the current policy $a \sim \pi(x, \{(\mathbf{I}_k, \bar{c}_k, \hat{p}_k)\}_{k=1}^K, \boldsymbol{\omega})$. The multi-objective reward for action a is represented as a vector $[\bar{s}_a, -\bar{c}_a]$. We update the replay buffer with these sampled trajectories and use samples from the buffer to train the policy. For mixup regularization, we identify the nearest neighbor for each sampled prompt and perform a weighted linear combination of the prompt embedding and its neighbors, where the weights are drawn from $\text{Beta}(0.2, 0.2)$. Both the reward and the advantage are linearly combined using the same weights.

At each training step, we sample 256 new prompts, along with their routing candidates and preference vectors, to obtain the routing trajectories and update the replay buffer. The training stage runs for 500 steps with a batch size of 256, using Adam optimizer with learning rate of 0.001.

E.6 Evaluation Setup

We evaluate the routing performance on 5 LLM evaluation benchmarks and various sets of routing candidates. Table E.4 presents the detailed evaluation settings.

E.7 Baselines

In this section, we describe the implementation details of the baseline methods.

E.7.1 RouteLLM

RouteLLM (Ong et al., 2024) develops a model that predicts the winning label between a pair of LLMs and selects the model based on a threshold applied to the predicted probability. To account for varying user preferences, we evaluate RouteLLM using a range of different thresholds.

E.7.2 Predictor

The predicted evaluation scores $\hat{p}_k = \text{sigmoid}(f(x, \mathbf{I}_k))$ can be used directly to compute the scalarized reward for an LLM M_k as $r_{\boldsymbol{\omega}}(x, k) = \boldsymbol{\omega}^T[\hat{p}_k, -c_k]$. The routing decision is then made by selecting $\hat{a} = \arg\max_k r_{\boldsymbol{\omega}}(x, k)$.

E.7.3 Random

The random routing policy selects models based on predefined probabilities for each model. Different user preferences are reflected by adjusting these probabilities. However, when there are more than two LLM candidates, specifying the probabilities becomes non-trivial, so we omit the random baseline in these scenarios.

E.7.4 Oracle

The oracle routing policy selects the model based on the actual evaluation scores, making the routing decision as $\hat{a} = \arg\max_k r_{\boldsymbol{\omega}}(x, k) = \arg\max_k \boldsymbol{\omega}^T[s(x, k), -c_k]$, where $s(x, k)$ represents the true performance score for model M_k on prompt x.

E.7.5 PPO

For each LLM candidate set and each user preference, we train a separate PPO routing policy to maximize the scalarized reward $r_{\boldsymbol{\omega}}(x, k) = \boldsymbol{\omega}^T[s(x, k), -c_k]$.

E.8 Additional Evaluation Results

We also evaluate our preference-conditioned dynamic routing (PCDR) approach on MT-Bench, a

Table E.4: Evaluation settings.

Benchmark	Setting	Models
AlpacaEval 2.0	GPT4/Mixtral-8x7B	gpt4_1106_preview Mixtral-8x7B-Instruct-v0.1
	GPT Family	gpt-3.5-turbo-0301 gpt-3.5-turbo-0613 gpt-3.5-turbo-1106 gpt-4-0125-preview gpt-4o-2024-05-13 gpt4 gpt4_0314 gpt4_0613 gpt4_1106_preview
	Claude Family	claude claude-2 claude-2.1 claude-3-5-sonnet-20240620 claude-3-opus-20240229 claude-3-sonnet-20240229 claude-instant-1.2
HELM-MMLU	GPT4/Mixtral-8x7B	gpt4_1106_preview mixtral-8x7b-32kseqlen
	Mistral Family	mistral-7b-instruct-v0.3 mixtral-8x22b mixtral-8x7b-32kseqlen
	GPT Family	gpt-3.5-turbo-0613 gpt-4-0613 gpt-4-1106-preview gpt-4o-2024-05-13
HELM-Lite	GPT4/Mixtral-8x7B	gpt4_1106_preview mixtral-8x7b-32kseqlen
	Mistral Family	mistral-7b-instruct-v0.3 mixtral-8x7b-32kseqlen mixtral-8x22b
	GPT Family	gpt-4o-2024-05-13 gpt-4o-mini-2024-07-18 gpt-3.5-turbo-0613 gpt-4-0613 gpt-4-1106-preview
OpenLLM	Yi1.5 Family	Yi-1.5-34B-Chat Yi-1.5-6B-Chat Yi-1.5-9B-Chat
	Mistral Family	Mistral-7B-Instruct-v0.2 Mixtral-8x22B-Instruct-v0.1 Mixtral-8x7B-Instruct-v0.1
	LLaMA3 Family	Llama-3-70B-Instruct Llama-3-8B-Instruct
OpenLLMv2	Yi1.5 Family	Yi-1.5-34B-Chat Yi-1.5-6B-Chat Yi-1.5-9B-Chat
	Qwen2 Family	Qwen2-0.5B-Instruct Qwen2-1.5B-Instruct Qwen2-72B-Instruct Qwen2-7B-Instruct
	LLaMA3 Family	Llama-3-70B-Instruct Llama-3-8B-Instruct

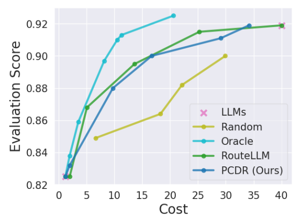


Figure E.1: Performance-cost trade-off on MTBench dataset.

Table E.5: Evaluation setting fro new routing candidates.

Benchmark	Setting	Models
OpenLLMv2	Cohere	aya-23-35B
		aya-23-8B
	Qwen2.5	Qwen2.5-0.5B-Instruct
		Qwen2.5-1.5B-Instruct
		Qwen2.5-7B-Instruct
		Qwen2.5-14B-Instruct
		Qwen2.5-32B-Instruct
		Qwen2.5-72B-Instruct

widely-used benchmark for assessing LLM performance. Figure E.1 shows the performance-cost trade-off curves for different routing methods. While the Oracle policy achieves the best performance-cost trade-off as expected, our PCDR approach performs competitively with RouteLLM, particularly in the mid-to-high cost regime ($20-40). Both methods significantly outperform random routing. Again, the gap between all routing policies and the Oracle baseline suggests potential room for improvement in routing decisions.

E.9 Cold Start for New Routing Candidates

To simulate the scenario where new models are introduced into the routing system, we select several unseen models from the HuggingFace Open-LLM v2 benchmark. These models are not used for training either the IRT model or the routing policy. Table E.5 shows the detailed evaluation settings.

[2] https://huggingface.co/datasets/berkeley-nest/Nectar

[3] https://huggingface.co/datasets/lmsys/lmsys-arena-human-preference-55k

[4] https://huggingface.co/datasets/lmsys/chatbot_arena_conversations

[5] https://huggingface.co/datasets/lmsys/mt_bench_human_judgments

[6] https://huggingface.co/datasets/routellm/gpt4_judge_battles

[7] https://tatsu-lab.github.io/alpaca_eval/

[8] https://crfm.stanford.edu/helm/lite/latest/

[9] https://crfm.stanford.edu/helm/mmlu/latest/

[10] https://huggingface.co/spaces/open-llm-leaderboard-old/open_llm_leaderboard

[11] https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard