RadialRouter: Structured Representation for Efficient and Robust Large Language Models Routing

Anonymous ACL submission

Abstract

The rapid advancements in large language models (LLMs) have led to the emergence of routing techniques, which aim to efficiently select the optimal LLM from diverse candidates to tackle specific tasks, optimizing performance while reducing costs. Current LLM routing methods are limited in effectiveness due to insufficient exploration of the intrinsic connection between user queries and the characteristics of LLMs. To address this issue, in this paper, we present **RadialRouter**, a novel framework for LLM routing which employs a lightweight Transformer-based backbone with a radial structure named **RadialFormer** to articulate the query-LLMs relationship. The optimal LLM selection is performed based on the final states of RadialFormer. The pipeline is further refined by an objective function that combines Kullback-Leibler divergence with the query-query contrastive loss to enhance robustness. Experimental results on RouterBench show that RadialRouter significantly outperforms existing routing methods by 9.2% and 5.8% in the *Balance* and *Cost First* scenarios, respectively. Additionally, its adaptability toward different performance-cost trade-offs and the dynamic LLM pool demonstrates practical application potential.

1 Introduction

Recent advances in natural language processing, significantly driven by the development of large language models (LLMs), has opened new frontiers across numerous applications. LLMs demonstrate outstanding performance on various tasks, including mathematical problem-solving (Romera-Paredes et al., 2024), commonsense reasoning (Zhao et al., 2023), and code generation (Wu et al., 2023). The growing reliance on LLMs gives rise to the concept of *LLM ensemble* (Chen et al., 2025), which integrates multiple LLMs to establish a system capable of multitasking, thereby generating more accurate and robust responses to user inputs. Through this collaborative approach, practitioners can leverage the unique strengths of different LLMs, potentially improving the overall performance and reliability in addressing diverse requirements. However, as the size and complexity of LLMs increase, the challenges of deploying LLM ensemble —such as computational cost, latency, and scalability —also intensify.

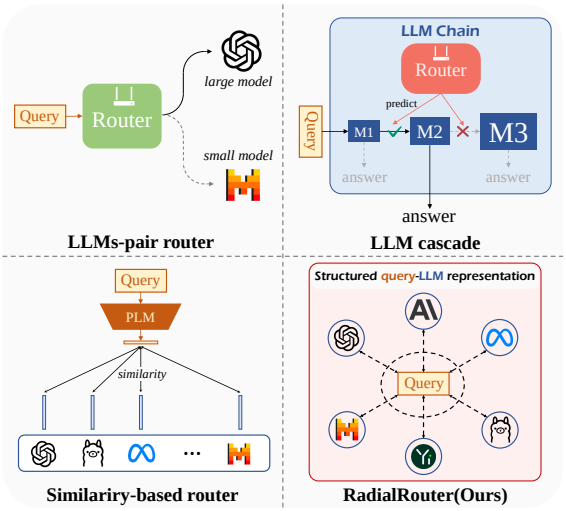


Figure 1: **Paradigm comparison between different LLM routing methods.** Existing methods lack the modeling of the interrelation between the query and LLMs, while the proposed RadialRouter unifies the routing process in a structured representation.

To address these challenges, *LLM routing* is explored to dynamically assign a specific LLM in the LLM ensemble to user queries. As shown in Fig.1, early attempts (Ding et al., 2024; Ong et al., 2024) employ a binary score router to decide whether to select a smaller LLM or a larger one for a given query. (Chen et al., 2023) utilizes a router to guide cascaded LLMs for generating responses. Further research (Chen et al., 2024) introduces similarity matching to align the query with LLMs and select the most appropriate LLM. However, these methods are limited in the following aspects: 1) They

narrow the routing sorely to identify the optimal LLM, failing to capture the intrinsic connection between the query and LLMs, which undermines their effectiveness in achieving ideal routing results. 2) The exclusive dependence on the features extracted by the BERT-based text encoder hinders their ability to leverage the contextual information and prevents a nuanced understanding of the underlying relationships and constraining the efficiency of routing. 3) Existing methods that limit routing to a fixed number of LLMs struggle to adapt to a dynamically evolving pool of LLMs. 4) Certain approaches neglect the actual requirements of the task, rendering them ineffective in scenarios that necessitate a simultaneous consideration of both performance and cost.

In this paper, we propose a novel LLM routing approach named **RadialRouter**, which leverages a Transformer-based architecture to enhance performance for efficient and robust LLM routing. To represent the interrelationship between the query and LLMs, we propose **RadialFormer** as the backbone of RadialRouter and incorporate the structure consisting of a relay node and n satellite nodes. During update, these nodes are processed through the multi-head attention mechanism. Compared with the standard Transformer, RadialFormer reduces the computational complexity from $O(n^2d)$ to $O(nd)$. The optimal LLM selection is performed based on the final states of satellite nodes. To guide the selection and enhance comprehensive representation, we employ a Kullback-Leibler divergence loss for supervision. Additionally, we cluster the queries into groups and introduce a query-query contrastive loss, which fosters the generation of similar embeddings for semantically related queries, facilitating robust LLM routing.

We conduct experiments on challenging RouterBench (Hu et al., 2024) encompassing 4 task domains (commonsense reasoning, knowledge-based language understanding, math, and coding) to evaluate the proposed RadialRouter in three scenarios. Extensive experiments demonstrate that RadialRouter efficiently harnesses the interrelationship of routing and outperforms existing routing methods by a large margin. Furthermore, the adaptability of RadialRouter toward different performance-cost trade-offs and the dynamic LLM pool is verified through extended experiments.

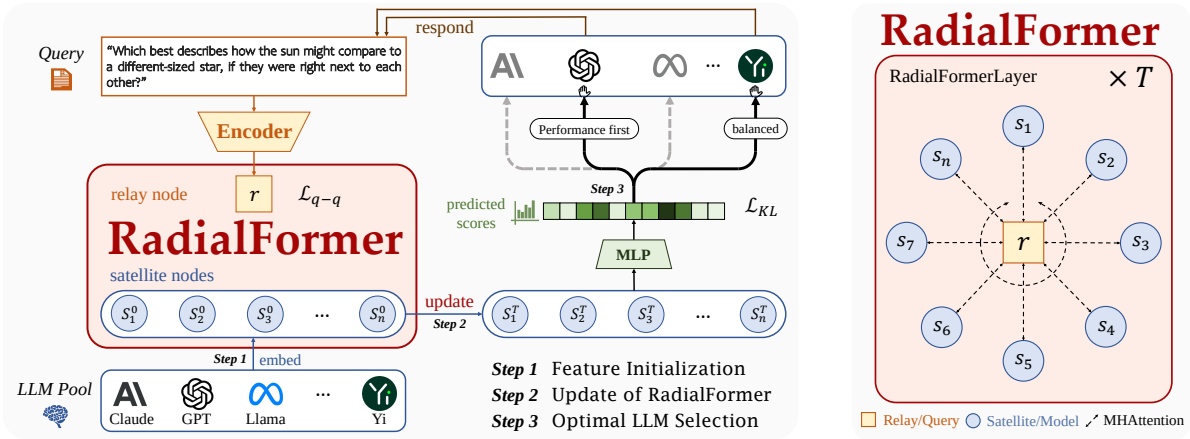
Our contributions are summarized as follows:

- We propose RadialRouter, a novel framework that leverages a Transformer-based architecture to dynamically route user queries to suitable LLMs.
- We introduce a lightweight architecture RadialFormer as the backbone of RadialRouter to capture the interrelationship between the query and LLMs in routing. To improve the robustness of routing, we incorporate contrastive loss in the optimization of RadialRouter.
- Experimental results show that RadialRouter outperforms baseline routing methods and achieves efficient and robust routing in three scenarios with different performance-cost trade-offs.

2 Related Work

LLM Ensemble The remarkable performance exhibited by a single LLM, coupled with the increasing demand for enhanced cross-domain capabilities, has catalyzed the emergence of the concept of LLM ensemble. Majority voting (Wang et al., 2022; Li et al., 2024) is a simple yet effective method to achieve the LLM ensemble. (Jiang et al., 2023) proposes a supervised ensembling method to produce enhanced results by synthesizing the outputs of all LLMs. (Du et al., 2023) develops a debate framework for LLM collaboration. LLM cascading (Chen et al., 2023; Yue et al., 2023; Gupta et al., 2024; Nie et al., 2024) adopts a serialized model architecture and halts when the output quality is satisfied. (Wang et al., 2023) tackles the fusion-of-experts problem by combining outputs from models with diverse knowledge domains. These approaches frequently exhibit considerable computational complexity, leading to substantial time latency and cost in practical applications. In contrast, our RadialRouter is highly efficient, as it requires only a single invocation of the routed LLM.

LLM Routing Similar to Mixture-of-Expert (MoE) approaches (Jacobs et al., 1991; Collobert et al., 2003; Jiang et al., 2024), LLM routing is designed to identify the most suitable LLM for a query, allowing the lightweight activation of LLM ensemble. (Shnitzer et al., 2023) addresses the LLM selection problem with a series of binary classification tasks. ZOOTER (Lu et al., 2023) develops a reward-guided method to train the routing function. HybridLLM (Ding et al., 2024) proposes a hybrid approach to save cost and maintain quality



(a) The framework of our proposed **RadialRouter**. RadialRouter captures the interrelation between the query and LLMs through a lightweight Transformer-based architecture. It selects the optimal LLM based on the final states of satellite nodes, while optimizing the pipeline through an objective function that combines Kullback-Leibler divergence and query-query contrastive loss.

(b) Connections of one layer in **RadialFormer**, where each satellite node is exclusively connected to the relay node to form a radial configuration.

Figure 2: Overview of **RadialRouter** methodology.

leveraging LLM pairs. The framework proposed in RouteLLM (Ong et al., 2024) dynamically routes between a strong model and a weak model. RouterDC (Chen et al., 2024) improves the routing performance by introducing dual contrastive learning. Recent advancements in LLM routing utilizing graph neural networks (Feng et al., 2024) and reinforcement learning (Sikeridis et al., 2024; Yue et al., 2025) show significant promise. Different from the aforementioned methods, the proposed RadialRouter leverages a Transformer-based backbone to achieve performance-cost balanced LLM routing.

3 Method

We consider a set of candidate LLMs $\{\text{LLM}_i : i = 1, \ldots n\}$, which can include local open-source LLMs and off-the-shelf LLMs hosted on cloud platforms. Our goal is to learn a router to select the most suitable LLM for each user query. For each round, the router receives the user query x as input and chooses the optimal $\text{LLM}_{\hat{i}}$ for response, balancing high performance with minimal cost.

In this section, we propose **RadialRouter**, a framework that leverages a Transformer-based architecture for query-based LLM routing. Fig.2a shows the framework of the proposed method. We present **RadialFormer**, a lightweight Transformer-based architecture as the backbone of routing (Sec.3.1). Based on the update of RadialFormer, we perform optimal LLM selection (Sec.3.2) and introduce a contrastive loss to optimize the router (Sec.3.3). The overall algorithm of RadialRouter is illustrated in Alg.2 of Appendix B.

3.1 RadialFormer Architecture

The key to LLM routing lies in designing a scoring mechanism to measure the potential capacity of LLMs on user queries. The backbone of the router necessitates the efficient representation of the query and LLMs. In this work, we present a novel Transformer-based architecture named RadialFormer, which builds upon the foundational design of the Star-Transformer (Guo et al., 2019), incorporating specific enhancements tailored to LLM routing tasks.

The RadialFormer consists of one relay node and n satellite nodes, representing the input user query and n candidate LLMs, respectively. The topology of the model is simplified for computational efficiency, where each satellite node is exclusively connected to the relay node, forming a radial configuration as shown in Fig.2b. Through the computational mechanisms employed by RadialFormer, the interrelationship between the query and LLMs is comprehensively captured, providing valuable insights for effective routing.

Update of RadialFormer Let $\mathbf{r}^t \in \mathbb{R}^{1\times d}$ denotes the state of the relay node, and $\mathbf{S}^t \in \mathbb{R}^{n\times d}$ denote the states of the n satellite nodes at time step t. Given a query x, we initialize the relay node with the query embedding encoded by a pre-trained

language model as $\mathbf{q} = \mathcal{E}(\mathrm{x}) \in \mathbb{R}^{1\times d}$. The satellite nodes are initialized with n learnable model embeddings $\{\mathbf{m}_i : i = 1, \ldots, n\}$.

Based on the multi-head attention mechanism (Vaswani et al., 2017), the update of RadialFormer focuses on processing the relay node and the satellite nodes. Details of the multi-head attention mechanism are introduced in Appendix A. The satellite node $\mathbf{s}_i$ is updated from the contextual information considering the relay node $\mathbf{r}^{t-1}$, the previous state $\mathbf{s}_i^{t-1}$, and the initial state $\mathbf{m}_i$:

$$\mathbf{C}_i^t = [\mathbf{s}_i^{t-1}; \mathbf{m}_i; \mathbf{r}^{t-1}], \tag{1}$$

$$\mathbf{s}_i^t = \mathrm{MHAttn}(\mathbf{s}_i^{t-1}, \mathbf{C}_i^t), \tag{2}$$

where $\mathbf{C}_i^t$ denotes the contextual information of the i-th satellite node. The relay node $\mathbf{r}$ is updated from all the satellite nodes and its previous state:

$$\mathbf{r}^t = \mathrm{MHAttn}(\mathbf{r}^{t-1}, [\mathbf{r}^{t-1}; \mathbf{S}^t]). \tag{3}$$

The updated satellite nodes are regularized through layer normalization (Ba et al., 2016). The update algorithm of RadialFormer is shown in Alg.1. Given the sequence length n and the dimension of the hidden state d, RadialFormer reduces the computational complexity of the standard Transformer from $O(n^2d)$ to $O(nd)$. The specific design of RadialFormer integrates both lightweight and structured representations of the routing. Subsequent experiments validate that RadialFormer facilitates efficient and effective LLM routing.

3.2 Optimal LLM Selection

Following the update of RadialFormer, we sent the final states $\mathbf{S}^T$ to an MLP network $\mathcal{M}$ to predict the potential score of the corresponding LLM as $\mathbf{s}_i^{\mathrm{pr}} = \mathcal{M}(\mathbf{s}_i^T)$, which comprehensively integrates both the information of the query and LLMs through RadialFormer. The optimal LLM selection is performed based on the predicted scores as $\hat{i} = \arg\max_i(p_i)$, where $p_i = \mathrm{softmax}(\mathbf{s}_i^{\mathrm{pr}})$ denotes the routing probability.

Given a query x_j, the process of LLM selection is supervised by the scores of the candidate LLMs $\mathrm{score}_{\mathrm{x}_j} = \{s_j^{(i)} : i = 1, \ldots, n\}$, which we identify in advance (described in Sec.4.3). We employ the Kullback-Leibler divergence loss (Kullback and Leibler, 1951) for supervision to guide the routing probability toward the probability derived from the exponent of true scores, which is defined as

$$\mathcal{L}_{\mathrm{KL}}(\mathrm{x}; \boldsymbol{\theta}) = D_{\mathrm{KL}}(p \| q) = \sum_{i=0}^{n} p_i \log \frac{p_i}{q_i}, \tag{4}$$

Algorithm 1 Update of RadialFormer

Input: number of layers T, model embeddings $\mathbf{m}_1, \cdots, \mathbf{m}_n$, and query embedding $\mathbf{q}$.

```
// Feature Initialization
s_1^0, ..., s_n^0 ← m_1, ..., m_n
r^0 ← q
for t = 1 to T do
    // Update the satellite nodes
    for i = 1 to n do
        C_i^t ← [s_i^{t-1}; m_i; r^{t-1}]
        s_i^t ← MHAttn(s_i^{t-1}, C_i^t)
        s_i^t ← LayerNorm(ReLU(s_i^t))
    // Update the relay node
    r^t ← MHAttn(r^{t-1}, [r^{t-1}; S^t])
    r^t ← LayerNorm(ReLU(r^t))
```

where $\boldsymbol{\theta}$ denotes the parameters in RadialRouter, p and $q = \mathrm{softmax}(\mathrm{score}_x)$ denote the predicted routing probability and the ground truth probability, respectively.

3.3 Optimization with Contrastive Loss

Inspired by (Chen et al., 2024), we leverage a contrastive loss to provide additional supervision for the optimization of RadialFormer.

Query-Query Contrastive Loss To enhance the robustness of LLM routing, we introduce a query-query contrastive loss, which promotes the ability of the language encoder in RadialRouter to generate analogous embeddings for semantically similar queries. Following (Chen et al., 2024), we transform the query embeddings encoder by a pre-trained language model into low-dimensional vectors by the t-SNE algorithm (Van der Maaten and Hinton, 2008) and perform k-means clustering (MacQueen, 1967) to obtain N semantic groups $\{\mathcal{K}_1, \cdots, \mathcal{K}_N\}$. We use the sample-sample contrastive loss to promote the generation of embeddings, formulated as:

$$\mathcal{L}_{\text{q-q}}(\mathrm{x}; \boldsymbol{\theta}) = -\log \frac{e^{\mathrm{sim}\langle \mathcal{E}(\mathrm{x}), \mathcal{E}(\mathrm{x}^+)\rangle}}{e^{\mathrm{sim}\langle \mathcal{E}(\mathrm{x}), \mathcal{E}(\mathrm{x}^+)\rangle} + \sum_t e^{\mathrm{sim}\langle \mathcal{E}(\mathrm{x}), \mathcal{E}(\mathrm{x}_t^-)\rangle}}, \tag{5}$$

where x^+ denotes in-group query, x_t^- denotes out-group queries, and $\mathrm{sim}\langle\cdot,\cdot\rangle$ denotes the cosine similarity.

Optimization Objective Finally, we learn the RadialRouter by minimizing a final objective that

combines the KL divergence and the query-query contrastive loss as

$$\boldsymbol{\theta}^* = \arg\min \mathbb{E}_{\mathrm{x}\sim\mathcal{D}_{\mathrm{train}}} \mathcal{L}_{\mathrm{KL}}(\mathrm{x};\boldsymbol{\theta}) + \lambda \mathcal{L}_{\text{q-q}}(\mathrm{x};\boldsymbol{\theta}), \tag{6}$$

where $\lambda > 0$ is a hyper-parameter.

4 Experimental Setup

4.1 Datasets and Candidate LLMs

We conduct experiments on RouterBench(Hu et al., 2024) to compare our RadialRouter model with baselines considering both performance and costs. We select user queries from 6 representative datasets in RouterBench across 4 task domains: (i) **Commonsense Reasoning**: Hellaswag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2021), ARC Challenge (Clark et al., 2018); (ii) **Knowledge-based Language Understanding**: MMLU (Hendrycks et al., 2021); (iii) **Math**: GSM8K (Cobbe et al., 2021); (iv) **Coding**: MBPP (Austin et al., 2021). A total of 11 candidate LLMs are involved in RouterBench, including both **open-source models**: Llama-70B-chat (Touvron et al., 2023), Mixtral-8x7B-chat (Aggarwal et al., 2024), Yi-34B-chat (Young et al., 2024), Code Llama-34B (Roziere et al., 2023), Mistral-7B-chat (Jiang et al., 2023), WizardLM-13B (Xu et al., 2024); and **proprietary models**: GPT-4, GPT-3.5-turbo (Achiam et al., 2023), Claude-instant-v1, Claude-v1, Claude-v2 (Anthropic, 2023).

4.2 Baselines

RadialRouter is compared with the following routing methods: (i) **CosineClassifier** trains a cosine classifier on the query embedding and performs a multi-class classification on candidate LLMs, which can be regarded as a simplified version of (Chen et al., 2024). (ii) **HybridLLM** (Ding et al., 2024) trains a language model to categorize queries to either small or large LLM. Mistral-7B-chat and GPT-4 are chosen as the small are large LLM, as they have the highest and lowest cost, respectively. We use DeBERTa (He et al., 2020) as the router model. (iii) **FrugalGPT** (Chen et al., 2023) uses a pre-trained language model to learn the scores of the generated results and guide the LLM cascade. We also use DeBERTa as the prediction model. (iv) **RouterDC** (Chen et al., 2024) learns a router to select the suitable LLM for user queries by dual contrastive learning. (v) **GraphRouter** (Feng et al., 2024) introduces a graph-based framework to leverage contextual information among tasks, queries, and LLMs for routing.

4.3 Metrics

Metrics that consider both performance and cost are utilized to evaluate RadialRouter and baselines.

- **Performance** refers to the average accuracy of responses across user queries generated by LLM or LLM ensemble equipped with routing methods.
- **Cost** refers to the average LLM inference cost for generating responses to the queries, which is expressed in dollars per 1M tokens. The statistics of candidate LLMs on RouterBench are shown in Appendix C.
- **Score** is employed to assess how effectively a method balances performance and cost: for an input query x_j, the score for LLM_i is calculated via

$$\mathrm{score}_{ij} = \mathrm{performance}_{ij} - \alpha \cdot \mathrm{cost}_i, \tag{7}$$

  where α balances the performance-cost trade-off and a higher α indicates a preference for saving cost. We define three scenarios: *Performance First*, *Balance*, and *Cost First*, which correspond to different priorities between performance and cost. In three scenarios, we set the value of α to 0, 0.02, and 0.1, respectively.

4.4 Implementation Details

We adopt mDeBERTaV3-base (He et al., 2021) as the language encoder $\mathcal{E}(\mathrm{x})$. For RadialFormer, the number of layers T is set to 6, with a 768-dim hidden dimension. The head number of the multi-head attention is 4, with each head having a dimension of 32. The MLP for predicting the routing scores has a hidden layer dimension of 128. The training batch size is 64, and the maximum training epoch is 1000. The router is trained using the AdamW (Loshchilov and Hutter, 2019) optimizer with a learning rate of 5×10^{-5}. The hyperparameter λ is set to 0.5. All experiments are run on a single NVIDIA A100 80GB GPU.

5 Empirical Results

5.1 Comparison with Baselines

We compare RadialRouter with baselines in three scenarios. The results are shown in Tab.1. Here, '*Best candidate*' denotes the individual LLM that

Table 1: **Comparison of routing methods on RouterBench across three distinct performance-cost trade-off scenarios**. Bold and underline denote the best and second-best results. All methods are evaluated on Performance, Cost, and Score. The results are taken as the average of each dataset.

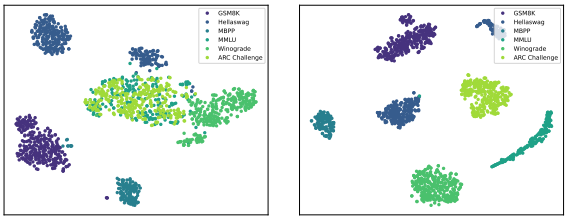
	Performance First			*Balance*			*Cost First*		
	Perf.↑	Cost↓	Score↑	Perf.↑	Cost↓	Score↑	Perf.↑	Cost↓	Score↑
Best candidate	0.813	7.185	0.813	0.709	0.562	0.698	0.704	0.439	0.660
Random	0.627	1.847	0.627	0.627	1.847	0.590	0.627	1.847	0.442
CosineClassifier	0.662	1.448	0.662	0.584	0.189	0.580	0.566	0.162	0.549
HybridLLM	0.801	6.869	0.801	0.791	6.612	0.659	0.517	0.107	0.506
FrugalGPT	0.813	7.185	0.813	0.671	0.336	0.664	0.549	0.124	0.536
RouterDC	0.815	6.768	<u>0.815</u>	0.716	1.313	0.690	0.718	0.418	<u>0.676</u>
GraphRouter	0.813	7.185	0.813	0.713	0.987	<u>0.693</u>	0.709	0.500	0.659
RadialRouter	0.816	6.759	**0.816**	0.781	1.179	**0.757**	0.763	0.476	**0.715**
Oracle	0.925	1.015	0.925	0.917	0.393	0.909	0.891	0.258	0.865

Table 2: **Ablation results on RadialRouter.** '*PF*', '*BA*', '*CF*' denote three trade-off scenarios. $\mathcal{RF}$, Star-$\mathcal{T}$, $\mathcal{T}$ denote RadialFormer, Star-Transformer and standard Transformer, respectively. 'Time' refers to the average routing time per batch in milliseconds. The best results are highlighted in **bold**.

Setting	*PF*	*BA*	*CF*	Time/ms
RadialRouter	**0.816**	**0.757**	**0.715**	10.7
$w/o\ \mathcal{RF}$				
+ Star-$\mathcal{T}$	0.813	0.751	0.709	13.5
+ $\mathcal{T}$	0.815	0.753	0.705	15.8
+ MLP	0.781	0.732	0.701	**4.6**
$w/o\ \mathcal{L}_{\text{KL}}$	0.548	0.442	0.017	-
$w/o\ \mathcal{L}_{\text{q-q}}$	0.813	0.740	0.711	-

achieves the highest score in the corresponding scenario. 'Random' denotes randomly selecting LLMs from the LLM pool to generate responses to testing queries. We conduct 50 independent selections and calculate the average of the results obtained. 'Oracle' denotes an ideal situation where all queries are routed to the optimal model, which defines the theoretical upper bound of the routing performance.

We can observe that RadialRouter substantially outperforms baseline methods in all three scenarios. In the *Performance First* scenario, a relatively singular optimal LLM (GPT-4) yields similar performance outcomes across the routing methods. The routing process becomes complex considering the performance-cost trade-off, leading to greater disparities among the methods. RadialRouter surpasses baselines by at least 9.2% and 5.8% in the *Balance* and *Cost First* scenarios, demonstrating the superiority of the framework. The adaptability of RadialRouter to different performance-cost trade-offs is further verified in the Sec.5.3.2. RadialRouter significantly exceeds the *Best candidate* and achieves at least 82.66% of the Oracle's score. This suggests that RadialRouter is capable of implementing flexible routing within the LLM pool to improve the routing ability. In contrast, the baseline methods struggle with ineffective representation of the routing process, which limits their overall scores. This underscores the importance of discerning the intrinsic connection between the query and LLMs in the routing task.



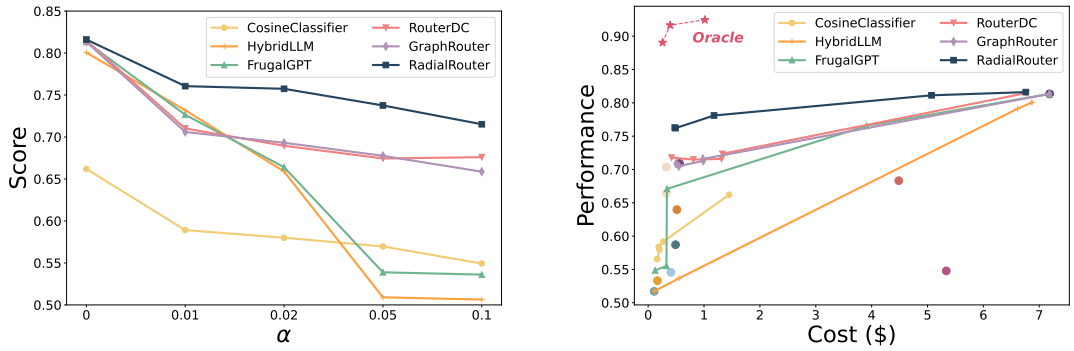
(a) $w/o\ \mathcal{L}_{\text{query-query}}$. (b) $w/\ \mathcal{L}_{\text{query-query}}$.

Figure 3: t-SNE visualization of test query embeddings extracted by the learned language encoder of RadialRouter.

5.2 Ablation Studies

We conduct ablation studies to investigate the contribution of each component in RadialRouter as shown in Tab.2. Here, 'w/o' denotes variants with specific components removed, and '+' denotes replacing RadialFormer with alternative architectures.

Replacing RadialFormer with alternative architectures (Star-Transformer (Guo et al., 2019), standard Transformer (Vaswani et al., 2017), and MLP) all leads to performance degradation, indicating the effectiveness of the attention mechanism design in RadialFormer for the structured representation. In construct, the architecture of Star-Transformer introduces unnecessary connections (e.g., Ring Connections), causing interference to the routing process. We further compare the routing efficiencies of different architectures. RadialFormer demonstrates lower time consumption than both Star-Transformer and standard Transformer, validating



(a) Effects of α on routing methods. All methods are evaluated on Score.

(b) Performance-cost trade-offs for routing methods. Scatter points denote candidate LLMs.

Figure 4: Routing results on RouterBench within different performance-cost trade-offs.

Table 3: **Comparison on different loss functions for LLM selection.** '*PF*', '*BA*', '*CF*' denote three trade-off scenarios. All settings are evaluated on Score. The best results are highlighted in **bold**.

Model Setting	*PF*	*BA*	*CF*
RadialRouter $w/\ \mathcal{L}_{\text{KL}}$	**0.816**	**0.757**	**0.715**
RadialRouter $w/\ \mathcal{L}_{\text{ce}}$	0.533	0.530	0.520
RadialRouter $w/\ \mathcal{L}_{\text{q-L}}$	0.815	0.714	0.676

the effectiveness of its lightweight design.

Eliminating the KL divergence loss leads to a substantial drop in metrics, further experiments in Sec.5.3.1 investigate the impact of loss functions on the optimal LLM selection. Eliminating the query-query contrastive loss also leads to a performance decline. Fig.3 exhibits the t-SNE visualization of test query embeddings extracted by the learned language encoder of RadialRouter. We can observe that the absence of the query-query contrastive loss results in mixed query embeddings across different datasets. By incorporating the contrastive loss, we achieve well-separated query embeddings, thereby establishing a robust foundation for effective routing. Detailed results of ablation studies are shown in Tab.8 of Appendix D.

5.3 Analysis

5.3.1 Loss Function for LLM Selection

We further compare the Kullback-Leibler divergence loss with two different loss functions for supervision the optimal LLM selection.

Cross-Entropy Loss Viewing LLM routing as a multi-class classification problem, the cross-entropy loss is introduced. In this approach, the LLM that receives the highest true score is assigned '1', while all other LLMs are assigned '0'. The

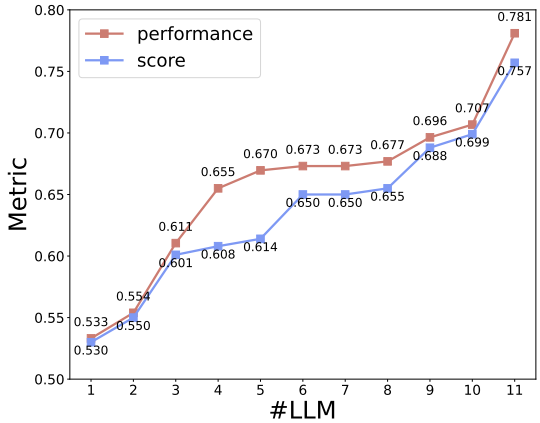


Figure 5: Effects of different numbers of LLMs.

cross-entropy loss function is defined as

$$\mathcal{L}_{\text{ce}}(\mathrm{x};\boldsymbol{\theta}) = -\sum_{i=1}^{n} y_i \log(p_i), \tag{8}$$

where y_i denotes the label for LLM_i, and p denotes the predicted routing probability.

Query-LLM Contrastive Loss Considering the objective of routing is to allocate the query to top-performing LLMs, rather than merely identifying the optimal model, (Chen et al., 2024) introduces the sample-LLM contrastive loss to learn the router. We make minor modifications to it under the RadialRouter framework. Specifically, based on the true scores, we construct the LLMs index set $\mathcal{I}^+$ and $\mathcal{I}^-$ as the indices of LLMs corresponding to the top-K and bottom-K scores, respectively. The query-LLM contrastive loss is then defined as

$$\mathcal{L}_{\text{q-L}}(\mathrm{x};\boldsymbol{\theta}) = \sum_{i^+\in\mathcal{I}^+} -\log \frac{e^{p_{i^+}}}{e^{p_{i^+}} + \sum_{i^-\in\mathcal{I}^-} e^{p_{i^-}}}, \tag{9}$$

where p denotes the predicted routing probability.

Table 4: **Comparison on routing to an increasing number of candidate LLMs** in the *Balance* scenario. The results are taken as the average of each dataset.

	#LLM	**Performance↑**	**Cost↓**	**Score↑**
WizardLM-13B-V1.2	1	0.5331	0.166	0.530
+ code-llama-34b-chat	2	0.5539	0.178	0.550
+ llama-2-70b-chat	3	0.6105	0.468	0.601
+ claude-v2	4	0.6550	2.348	0.608
+ claude-v1	5	0.6696	2.758	0.614
+ claude-instant-v1	6	0.6731	1.134	0.650
+ mistral-7b-chat	7	0.6731	1.134	0.650
+ mixtral-8x7b-chat	8	0.6769	1.109	0.655
+ Yi-34B-Chat	9	0.6964	0.421	0.688
+ gpt-3.5-turbo-1106	10	0.7068	0.404	0.699
+ gpt-4-1106-preview	11	0.7810	1.179	0.757

Tab.3 displays the results of comparison. Training RadialRouter with $\mathcal{L}_{\text{KL}}$ yields the highest scores in all three scenarios, as can be observed. From a macroscopic perspective, $\mathcal{L}_{\text{KL}}$, $\mathcal{L}_{\text{ce}}$ and $\mathcal{L}_{\text{q-L}}$ address the routing problem out of distinct viewpoints: probabilistic distribution fitting, multi-class classification, and similarity matching, respectively. The superiority of the KL divergence underscores the effectiveness of framing the routing problem through the lens of probabilistic distribution fitting, which fosters a comprehensive understanding of the intrinsic connection between the query and LLMs in the routing process, as opposed to focusing merely on a limited number of dominant LLMs. This perspective aligns seamlessly with the design principles of RadialFormer, allowing for the accomplishment of both efficient and robust routing.

5.3.2 Adaptability to Performance-Cost Trade-Offs

Beyond the three aforementioned scenarios, we assess the adaptability of RadialRouter and baseline routing methods to performance-cost trade-offs by varying the parameter α in Eq.7. Fig.4a shows the scores achieved by routing methods within different α, where RadialRouter achieves the highest scores in different trade-offs, significantly outperforming baseline routing methods. Fig.4b depicts the performance-cost curves of routing methods. Our observation indicate that RadialRouter is capable to obtain improved performance while maintaining comparable costs to baselines, leading to a robust performance-cost balance. This analysis substantiates the adaptability of RadialRouter to performance-cost trade-offs, which highlights its practical applicability. Detailed results are shown in Tab.9 of Appendix E.

5.3.3 Routing to Different Numbers of LLMs

We evaluate the efficiency of RadialRouter with a dynamic LLM pool by gradually increasing the number of candidate LLMs. We compare the results in the *Balance* scenario, as shown in Tab.4 and Fig.5. We can observe that increasing the number of candidate LLMs leads to improved performance, demonstrating RadialRouter's ability to effectively adapt to a dynamic LLM pool.

5.3.4 Effects of λ on Optimization

We study the impact of the contrastive loss weight λ in Eq.6 on the optimization of RadialRouter. Experiments are conducted in the *Balance* scenario, and the results are presented in Appendix F. As can be seen, the highest score is achieved when $\lambda = 0.5$. Therefore, we fix λ to 0.5 when training RadialRouter. Moreover, we can observe that RadialRouter demonstrates insensitivity across a wide range of $\lambda \in [0.25, 5]$, providing more flexibility for the selection of λ in practice.

6 Conclusions

In this paper, we introduce RadialRouter, a novel Transformer-based framework for efficient and robust LLM routing. We achieve a structured representation of the routing process with RadialFormer, which efficiently captures the interrelationship between the query and LLMs. The robustness of RadialRouter is enhanced by incorporating contrastive loss. Extensive experiments on RouterBench demonstrate that RadialRouter consistently outperforms baseline methods across various scenarios, exhibiting adaptability to performance-cost trade-offs and efficacy in routing queries within a dynamic LLM pool. These findings confirm the potential of RadialRouter as a superior solution for LLM deployment in practical applications.

Limitations

We acknowledge several limitations regarding our proposed method.

First, RadialRouter requires re-training whenever a new LLM is introduced to the LLM pool. Consequently, the ability to rapidly adapt and iterate in dynamic environments is hindered, particularly in scenarios where frequent updates are needed to address evolving tasks or domains. The router's implementation of training-free adaptation to the dynamic LLM pool relies on a general portrait or embedding for various LLMs, but that is beyond the scope of this paper.

Second, due to limited computational resources, we have not performed testing within multi-language and multi-modal LLM ensembles, which may restrict our ability to fully assess the framework's applicability across languages and modalities. We leave the investigation of such scenarios to future work.

References

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

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, et al. 2024. Automix: Automatically mixing language models. *Advances in Neural Information Processing Systems*, 37:131000–131034.

Anthropic. 2023. Model card and evaluations for claude models.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

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*.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. 2024. Routerdc: Query-based router by dual contrastive learning for assembling large language models. *Advances in Neural Information Processing Systems*, 37:66305–66328.

Zhijun Chen, Jingzheng Li, Pengpeng Chen, Zhuoran Li, Kai Sun, Yuankai Luo, Qianren Mao, Dingqi Yang, Hailong Sun, and Philip S Yu. 2025. Harnessing multiple large language models: A survey on llm ensemble. *arXiv preprint arXiv:2502.18036*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems, 2021. *arXiv preprint arXiv:2110.14168*, 9.

Ronan Collobert, Yoshua Bengio, and Samy Bengio. 2003. Scaling large learning problems with hard parallel mixtures. *International Journal of pattern recognition and artificial intelligence*, 17(03):349–365.

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*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Tao Feng, Yanzhen Shen, and Jiaxuan You. 2024. Graphrouter: A graph-based router for llm selections. In *The Thirteenth International Conference on Learning Representations*.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-transformer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1315–1325.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. 2024. Language model cascades: Token-level uncertainty and beyond. In *The Twelfth International Conference on Learning Representations*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *arXiv e-prints*, page arXiv:2310.06825.

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, et al. 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*.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. 2024. More agents is all you need. *arXiv preprint arXiv:2402.05120*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

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*.

James MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, volume 5, pages 281–298. University of California press.

Lunyiu Nie, Zhimin Ding, Erdong Hu, Christopher Jermaine, and Swarat Chaudhuri. 2024. Online cascade learning for efficient inference over streams. In *International Conference on Machine Learning*, pages 38071–38090. PMLR.

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 from preference data. In *The Thirteenth International Conference on Learning Representations*.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. 2024. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475.

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

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

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*.

Dimitrios Sikeridis, Dennis Ramdass, and Pranay Pareek. 2024. Pickllm: Context-aware rl-assisted large language model routing. *arXiv preprint arXiv:2412.12170*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Guoyin Wang, Heng Li, Jiangcheng Zhu, Jianqun Chen, et al. 2024. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. 2023. Large language model cascades with mixture of thoughts representations for cost-efficient reasoning. *arXiv preprint arXiv:2310.03094*.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. 2025. Masrouter: Learning to route llms for multi-agent systems. *arXiv preprint arXiv:2502.11133*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2023. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36:31967–31987.

A Details of Multi-Head Attention

The update of RadialFormer is based on the attention mechanism (Vaswani et al., 2017). Specifically, given the input sequence $\mathbf{H} \in \mathbb{R}^{m \times d}$ where m denotes the sequence length and d denotes the hidden dimension, we use a query sequence $\mathbf{q} \in \mathbb{R}^{1 \times d}$ to compute the relevant information based on the scaled dot-product attention:

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}(\frac{QK^{\mathsf{T}}}{\sqrt{d}})V, \quad (10)$$

where $[Q; K; V] = [\mathbf{q}W_q; \mathbf{H}W_k; \mathbf{H}W_v]$, and W_q, W_k, W_v denote learnable parameters.

The multi-head attention layer extends the scale dot-product attention by performing h paralleled attention operations and concatenating the information:

$$\mathrm{MHAttn}(\mathbf{q}, \mathbf{H}) = [\mathrm{head}_1, \cdots, \mathrm{head}_h]W_o, \quad (11)$$

$$\mathrm{head}_i = \mathrm{Attn}(Q_i, K_i, V_i), i \in [1, h], \quad (12)$$

where $[Q_i; K_i, ; V_i]$ are the i-th group from $[Q; K; V]$ with a dimension of d/h, and W_o denotes output learnable parameter.

In RadialFormer, the multi-head attention mechanism is utilized to update states of the relay node and the satellite nodes.

B Training and Inference Algorithm of RadialRouter

Alg.2 shows the training and inference procedures of RadialRouter. The algorithm presented in RadialRouter outlines a training and inference framework for efficient and robust LLM routing. During training, the learnable parameters in the router are updated through mini-batch sampling, supervised by the Kullback-Leibler divergence loss and the query-query contrastive loss. Given an input query, the inference of RadialRouter involves predicting the routing probabilities for the candidate LLMs and selecting the optimal one for response.

C Statistics of Candidate LLMs on RouterBench

Tab.5 shows the basic statistics of 11 LLMs in RouterBench as our candidate LLMs. Tab.7 shows the statistics of candidate LLMs on RouterBench considering performance and cost. The performance of each candidate LLM is assessed by averaging the accuracy across six different datasets.

Algorithm 2 The overall algorithm of **RadialRouter**

Input: training set $\mathcal{D}_{\text{train}}$, LLMs $\{\text{LLM}_i : i = 1, \ldots, n\}$, number of out-group queries H, number of clusters N, hyper-parameter λ, mini-batch size b, and learning rate η; learnable parameters $\boldsymbol{\theta}$: encoder $\mathcal{E}$, RadialFormer $\mathcal{RF}$, MLP $\mathcal{M}$, and learnable LLM embeddings $\{\mathbf{m}_i : i = 1, \ldots, n\}$;

Training:

Score LLMs for each query $(\text{x}_j, \text{y}_j) \in \mathcal{D}_{\text{train}}$ and obtain $\{s_j^{(i)} : i = 1, \ldots, n\}$ by Eq.7;
Cluster training queries $\{\text{x}_j : j = 1, \ldots, l\}$ into N groups $\{\mathcal{K}_1, \cdots, \mathcal{K}_N\}$;
repeat
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}_{\text{train}}$;
  for $(\text{x}_j, \text{y}_j) \in \mathcal{B}$ **do**
    $\mathbf{q} \leftarrow \mathcal{E}(\text{x}_j)$;
    Compute the updated state of RadialFormer by Alg.1:
    $\mathbf{r}^T, \mathbf{s}_1^T, \cdots, \mathbf{s}_n^T \leftarrow \mathcal{RF}(\mathbf{q}, \mathbf{m}_1, \cdots, \mathbf{m}_n)$;
    Compute the Kullback-Leibler divergence loss $\mathcal{L}_{\text{Kullback-Leibler}}(\text{x}_j; \boldsymbol{\theta})$ by Eq.4;
    Sample an in-group query x^+ and H out-group queries x_t^- from $\mathcal{B}$;
    Compute the query-query contrastive loss $\mathcal{L}_{\text{query-query}}(\text{x}_i; \boldsymbol{\theta})$ by Eq.5;
  $\mathcal{L}(\mathcal{B}; \boldsymbol{\theta}) \leftarrow \sum_{\text{x}_i \in \mathcal{B}} \mathcal{L}_{\text{Kullback-Leibler}}(\text{x}_i; \boldsymbol{\theta}) + \lambda \mathcal{L}_{\text{query-query}}(\text{x}_i; \boldsymbol{\theta})$;
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{B}; \boldsymbol{\theta})$;
until converged.

Inference:

Sample a testing query x';
Compute the predicted routing probability p' using the $\mathcal{E}$, $\mathcal{RF}$ and $\mathcal{M}$;
$i' \leftarrow \arg\max_{i \in \{1, \ldots, n\}} (p'_i)$;
$\hat{\text{y}}' \leftarrow \text{LLM}_{i'}(\text{x}')$.

Output: response $\hat{\text{y}}'$

Table 5: Statistics of different LLMs in RouterBench.

	LLM	Size
open-source	WizardLM-13B-V1.2	13B
	code-llama-34b-chat	34B
	llama-2-70b-chat	70B
	mistral-7b-chat	7B
	mixtral-8x7b-chat	47B
	Yi-34B-Chat	34B
proprietary	claude-instant-v1	-
	claude-v1	-
	claude-v2	-
	gpt-3.5-turbo-1106	-
	gpt-4-1106-preview	-

The cost is determined based on the pricing of the LLMs per million tokens, and is also averaged over the six datasets.

D Detailed Results of Ablation Studies

Tab.8 shows the detailed results of ablation studies on RadialRouter. Through ablation studies, we verify the rationality of RadialFormer and the necessity of introducing the KL divergence loss and the query-query contrastive loss.

Table 6: **Effects of** λ in the *Balance* scenario. The best results are highlighted in **bold**.

λ	Performance	Cost	Score
0	0.7492	0.455	0.740
0.25	0.7911	1.964	0.752
0.5	0.7810	1.179	**0.757**
0.75	**0.7923**	1.960	0.753
1	0.7911	1.964	0.752
2	0.7845	1.985	0.745
3	0.7911	1.964	0.752
4	0.7589	0.478	0.749
5	0.7620	0.488	0.752
6	0.7577	0.482	0.748
8	0.7581	0.481	0.748
10	0.7575	0.481	0.748

E Full Results of Performance-Cost Balance

Tab.9 is the full results of Fig.4. We set the value of α to 0, 0.01, 0.02, 0.05, and 0.1 to assess the adaptability of different routing methods to performance-cost trade-offs. We can see that RadialRouter is robust to a wide range of performance-cost trade-off scenarios ($\alpha \in [0, 0.1]$).

Table 7: Statistics of candidate LLMs on RouterBench.

LLM	GSM8K	Hellaswag	MBPP	MMLU	winograde	ARC	Perf.↑	Cost↓
WizardLM-13B-V1.2	0.5054	0.6004	0.3906	0.5253	0.5289	0.6476	0.5331	0.166
claude-instant-v1	0.6281	0.7690	0.6250	0.4529	0.5211	0.8421	0.6397	0.514
claude-v1	0.6520	0.8187	0.6094	0.5281	0.5711	0.9199	0.6832	4.486
claude-v2	0.6671	0.3130	0.6406	0.5652	0.4763	0.6247	0.5478	5.336
gpt-3.5-turbo-1106	0.6094	0.7843	0.6875	0.6667	0.6632	0.8444	0.7092	0.562
gpt-4-1106-preview	0.6589	0.9057	0.6875	0.8162	0.8552	0.9565	0.8134	7.185
code-llama-34b-chat	0.4548	0.5194	0.5156	0.5284	0.5921	0.6636	0.5457	0.407
llama-2-70b-chat	0.5252	0.7046	0.3750	0.6034	0.4974	0.8169	0.5871	0.490
mistral-7b-chat	0.4151	0.5410	0.3828	0.5198	0.5737	0.6705	0.5171	0.107
mixtral-8x7b-chat	0.5214	0.6960	0.5391	0.6822	0.6842	0.8627	0.6642	0.324
Yi-34B-Chat	0.5517	0.8782	0.4141	0.7187	0.7421	0.9176	0.7037	0.439

Table 8: **Detailed ablation results on RadialRouter.** The best results are highlighted in **bold**.

	Performance First			*Balance*			*Cost First*		
	Perf.	Cost	Score	Perf.	Cost	Score	Perf.	Cost	Score
RadialRouter	0.816	6.759	**0.816**	0.781	1.179	**0.757**	0.763	0.476	**0.715**
w/o RadialFormer									
+ Star-Transformer	0.813	7.185	0.813	0.794	2.170	0.751	0.758	0.491	0.709
+ Transformer	0.815	6.768	0.815	0.792	1.960	0.753	0.752	0.478	0.705
+ MLP	0.781	4.362	0.781	0.770	1.940	0.732	0.751	0.496	0.701
$w/o\ \mathcal{L}_{\text{KL}}$	0.548	5.308	0.548	0.548	5.308	0.442	0.548	5.308	0.017
$w/o\ \mathcal{L}_{\text{q-q}}$	0.813	7.185	0.813	0.759	0.519	0.749	0.759	0.478	0.711

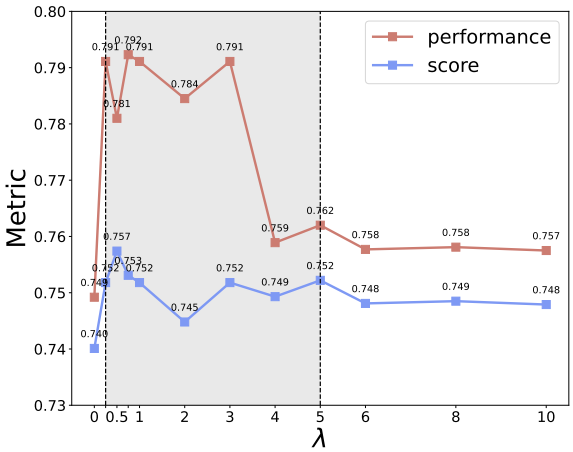


Figure 6: Effects of λ in the *Balance* scenario.

F Detailed Results of Effects of λ on Optimization

Tab.6 shows the effects of the contrastive loss weight λ on the optimization of RadialRouter. Experiments are conducted in the *Balance* scenario, visualized in Fig.6. As can be seen, the highest score is achieved when $\lambda = 0.5$ and RadialRouter demonstrates insensitivity across a wide range of $\lambda \in [0.25, 5]$, offering greater flexibility for the selection of λ in practice.

Table 9: Performance and cost of routing methods on RouterBench with different α.

Method	α	GSM8K	Hellaswag	MBPP	MMLU	winograde	ARC	**Perf.↑**	**Cost↓**
CosineClassifier	0	0.6062	0.7046	0.6250	0.6798	0.5395	0.8169	0.6620	1.448
	0.01	0.4671	0.6004	0.4922	0.5730	0.5553	0.8627	0.5918	0.271
	0.02	0.4725	0.5410	0.4766	0.5765	0.5737	0.8627	0.5838	0.189
	0.05	0.4320	0.5410	0.5000	0.5663	0.5763	0.8627	0.5797	0.201
	0.1	0.4945	0.5410	0.5703	0.5352	0.5816	0.6705	0.5655	0.162
HybridLLM	0	0.6489	0.8898	0.6719	0.8054	0.8474	0.9405	0.8006	6.869
	0.01	0.6489	0.8898	0.6719	0.8054	0.8474	0.9405	0.8006	6.869
	0.02	0.6371	0.8715	0.6719	0.7904	0.8474	0.9291	0.7912	6.612
	0.05	0.4294	0.5586	0.4219	0.5393	0.5868	0.6842	0.5367	0.553
	0.1	0.4151	0.5410	0.3828	0.5198	0.5737	0.6705	0.5171	0.107
FrugalGPT	0	0.6589	0.9057	0.6875	0.8162	0.8552	0.9565	0.8134	7.185
	0.01	0.6317	0.8437	0.6953	0.7422	0.7816	0.8993	0.7656	3.910
	0.02	0.5229	0.7172	0.5312	0.6888	0.6947	0.8696	0.6708	0.336
	0.05	0.5056	0.6004	0.5234	0.5253	0.5289	0.6476	0.5552	0.327
	0.1	0.4154	0.5410	0.5703	0.5198	0.5737	0.6705	0.5485	0.124
RouterDC	0	0.6671	0.9057	0.6875	0.8163	0.8553	0.9565	0.8147	6.768
	0.01	0.6671	0.8782	0.5078	0.6860	0.6842	0.9176	0.7235	1.329
	0.02	0.6671	0.8782	0.5156	0.6869	0.6842	0.8627	0.7158	1.313
	0.05	0.6094	0.8782	0.6016	0.5822	0.7553	0.8627	0.7149	0.810
	0.1	0.6281	0.8782	0.5078	0.6910	0.6842	0.9176	0.7178	0.418
GraphRouter	0	0.6589	0.9057	0.6875	0.8162	0.8552	0.9565	0.8134	7.185
	0.01	0.6121	0.7902	0.6953	0.6751	0.6711	0.8513	0.7158	0.980
	0.02	0.6121	0.7886	0.6875	0.6739	0.6711	0.8444	0.7129	0.987
	0.05	0.6013	0.7896	0.6719	0.6698	0.6553	0.8421	0.7050	0.548
	0.1	0.5802	0.8321	0.5859	0.6926	0.6895	0.8719	0.7087	0.500
RadialRouter	0	0.6672	0.9057	0.6953	0.8163	0.8553	0.9565	0.8161	6.759
	0.01	0.6671	0.8782	0.7031	0.8077	0.8553	0.9565	0.8113	5.072
	0.02	0.6281	0.8782	0.6797	0.7270	0.8553	0.9176	0.7810	1.179
	0.05	0.6281	0.8782	0.6875	0.7187	0.7421	0.9176	0.7620	0.488
	0.1	0.6281	0.8782	0.6875	0.7235	0.7421	0.9176	0.7628	0.476