RADAR: Routing Agents via Difficulty-Aware Recovery

Haizhou Du
Shanghai University of Electric Power
duhaizhou@shiep.edu.cn

Jinze Zhao
Shanghai University of Electric Power
y24208123@mail.shiep.edu.cn

Huaicheng Yan
Shanghai University of Electric Power
huaichengyan@shiep.edu.cn

Abstract

Multi-agent systems (MAS) have become a dominant paradigm, and routing each query to a suitable agent is key to balancing performance and cost. However, existing agent routing methods suffer from the prohibitive cost of exhaustive benchmark construction and the neglect of query complexity essential for mixed tasks. To address this issue, we propose the **RADAR** (**R**outing **A**gents via **D**ifficulty-**A**ware **R**ecovery) framework. RADAR enables accurate decision-making under limited evaluation cost through two complementary modules. The MatrixComp module employs structure-aware matrix completion to recover global performance profiles from sparse observations. Furthermore, the DiffMatch module extracts reasoning meta-features for fine-grained alignment via a difficulty-calibrated retrieval mechanism. Extensive experiments demonstrate that RADAR can significantly reduce the computational cost of benchmark construction and maintain performance equal with state-of-the-art methods.

1 Introduction

Recent advancements in Large Language Models (LLMs) drove remarkable progress in intelligent applications [1, 2], giving rise to multi-agent systems that achieve superior performance across diverse domains [3, 4, 5, 6]. The performance of individual agents varies drastically across queries [7, 8, 9, 10]. Agents that pair powerful models with high-effort prompting strategies incur expensive computational costs, whereas agents that pair lightweight models with low-effort prompting strategies are likely to fail on tough tasks. Therefore, it is a non-trivial issue how to dynamically select the suitable agent. A well-designed routing mechanism can significantly improve system efficiency in a wide range of settings [11].

Recent researches divide into two categories: *model routing* and *agent routing* to improve system efficiency. *Model routing* strategies fall into two paradigms: sequential and predictive. Sequential methods formalize cascading chains, invoking cost-effective models first and escalating to larger ones only when necessary [9, 12]. Predictive approaches select a single best model directly via semantic-based retrieval or supervised regression [7, 8, 13, 14]. In recent years, *agent routing* extends these principles to multi-agent systems, selecting not only the foundational backbone but also reasoning strategies and collaboration protocols [6, 15, 16, 17]. Unlike model routing, agent routing addresses task decomposition and specialized agent allocation, typically through semantic sub-task assignment, dynamic resource allocation with early-exit mechanisms, or cascaded controller networks for joint role and model selection [6, 18, 19].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Despite these advancements, adapting existing routing paradigms to the expansive multi-agent landscape encounters two challenges. *The key challenge is that the prohibitive cost of comprehensive benchmarking impedes practical deployment.* State-of-the-art routing methods generally require a dense performance matrix containing the ground-truth accuracy of all agents on all training queries to facilitate effective learning. As the action space expands into the Cartesian product of available backbones and strategies, the combinatorial explosion causes such a full performance matrix computationally infeasible. *Another important challenge is the insufficient difficulty awareness of traditional routing methods.* Existing methods heavily rely on semantic embeddings, which often fail to distinguish between superficial keyword overlap and intrinsic algorithmic complexity. Consequently, these methods struggle to differentiate between tasks solvable by lightweight strategies and those demanding expensive reasoning agents, thereby limiting routing precision on mixed tasks.

To address the above challenges, our core idea is a cost-effective and difficulty-aware routing mechanism tailored for the multi-agent landscape. Specifically, we address the data scarcity caused by prohibitive benchmarking costs with a structure-aware matrix completion protocol. This protocol characterizes low-rank latent factors of agent behaviors and clusters problems by their intrinsic cost-accuracy trade-offs. It accurately reconstructs the global performance manifold from extremely sparse observations. Then, we design a difficulty-aware matching mechanism to overcome the imprecision of semantic-only routing. This mechanism employs a lightweight LLM probe to extract multi-dimensional meta-features including logical constraints and required knowledge depth. These features serve as critical anchors for fine-grained performance approximation, enabling the system to distinguish superficial complexity from genuine reasoning challenges.

Our main contributions are summarized as follows:

- **Comprehensive Routing Framework.** We propose RADAR, a framework designed for the dynamic and cost-aware dispatching of optimal agent configurations. It establishes a unified paradigm that harmonizes the trade-off between system quality and inference costs in complex multi-agent environments.
- **Novel Algorithmic Modules.** We develop two complementary modules: MatrixComp, which recovers global performance profiles via structure-aware matrix completion, and DiffMatch, which performs difficulty-calibrated retrieval by combining difficulty meta-features with reranked neighbor voting.
- **Systematic Benchmark Construction.** We construct AgentRoutingBench, a comprehensive benchmark unifying four high-difficulty datasets. To reflect the frontier of LLM capabilities, we present extensive inference results across an action space comprising four backbone models of varying computational scales combined with three distinct prompting strategies, significantly expanding the empirical scope of agent routing.
- **Comprehensive Experimental Evaluation.** Extensive experiments demonstrate that RADAR matches full-matrix baselines while evaluating only 66% of the action space, and improves Average Improvement in Quality (AIQ) by 2.47% over the strongest baseline at the same budget.

2 Related Work

2.1 Model Routing

The primary objective of model routing is to optimize the trade-off between performance and inference cost by dynamically selecting the most appropriate LLM for a given query. As the ecosystem of LLMs expands, researchers develop various strategies to navigate this trade-off. These methods sequentially invoke models typically starting from smaller, cost-effective models and escalating to larger ones only when necessary. FrugalGPT [9] formalizes this as a cascading chain, where a query is passed to the next model in the sequence if the current model's response fails to meet a specific quality threshold. Similarly, AutoMix [12] introduces a self-verification mechanism where a smaller model assesses the reliability of its own output, routing to a larger verifier only when confidence is low. By matching easy instances with lightweight models, these methods significantly reduce the average cost per query. In contrast, predictive systems aim to select the single best model directly without sequential execution. LLM-Blender [20] employs a pairwise ranking mechanism to compare outputs from multiple models and select the best candidate. More recently, RouterBench

[8] and RouteLLM [7] utilize auxiliary components—such as BERT-based classifiers or lightweight regression networks—to map query embeddings to predicted model performance, achieving results comparable to frontiers like GPT-4 while reducing costs by over 50%. To further enhance granularity, "Attention, Please" [21] introduces a single-head cross-attention mechanism that jointly models query and model embeddings to predict both response quality and generation cost. Additionally, UniRoute [22] proposes a universal routing approach that represents each LLM as a feature vector based on its prediction errors on representative prompts.

However, these model-centric methods generally overlook the complexity of reasoning strategies, limiting their effectiveness in optimizing the diverse action space of multi-agent systems.

2.2 Agent Routing

While single-model routing focuses on selecting a standalone LLM, recent advancements in MAS shift the challenge toward task decomposition and specialized agent allocation. In many MAS frameworks, routing is treated as an orchestration process. AOP [18] introduces an agent-oriented planning framework that decomposes complex queries into sub-tasks and utilizes semantic matching to assign these tasks to specialized agents based on their descriptions. This allows for high-level dispatching where agents act as modular experts tailored to specific sub-domains. Recent works began to formalize agent selection as a distinct optimization challenge. MaAS [19] addresses the dynamic allocation of inference resources by sampling query-dependent execution paths, incorporating an early-exit mechanism that routes simple queries to a single lightweight model to bypass complex multi-agent overhead. Furthermore, MasRouter [6] employs a cascaded controller network to jointly determine role allocation and model selection, ensuring each sub-task is routed to the most cost-effective backbone. These methods attempt to balance the synergy between multiple agents and the associated computational costs.

Despite these advancements, existing MAS routing methods often struggle to accurately perceive intrinsic task difficulty beyond surface-level semantics and typically rely on computationally expensive full-matrix benchmarks for training, lacking efficiency in sparse data scenarios.

3 Methodology

3.1 Problem Formulation

Action Space. We consider a compound AI system comprising a set of backbone LLMs $\mathcal{M}$ and a set of prompting strategies $\mathcal{S}$. An action a_j corresponds to a specific pair $(m, s) \in \mathcal{M} \times \mathcal{S}$. The complete action space is denoted as $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$, representing the Cartesian product of available models and strategies.

Objective. Let $\mathcal{Q}$ be the universe of queries. For each query $q_i \in \mathcal{Q}$ executed by action $a_j \in \mathcal{A}$, we define two intrinsic metrics: the response correctness $P_{i,j} \in [0, 1]$ and the associated inference cost $C_{i,j} \in \mathbb{R}^+$. The Oracle routing establishes a performance ceiling with perfect prior knowledge of these ground-truth outcomes. Given a user-defined coefficient λ that governs the trade-off between quality and budget, the Oracle selects the optimal action a_i^* by maximizing the utility function:

$$a_i^* = \arg\max_{a_j \in \mathcal{A}} \left(P_{i,j} - \lambda \cdot C_{i,j} \right). \tag{1}$$

In practice, we deploy a Predictive routing to estimate the expected performance ($\hat{P}_{i,j}$) and cost ($\hat{C}_{i,j}$) based on the input query. Our goal is to construct an accurate estimator to make a decision $\hat{a}_i^*$ that approximates the Oracle's choice:

$$\hat{a}_i^* = \arg\max_{a_j \in \mathcal{A}} \left(\hat{P}(q_i, a_j) - \lambda \cdot \hat{C}(q_i, a_j) \right), \tag{2}$$

thereby minimizing the regret between the predictive decision and the optimal solution.

3.2 MatrixComp module

The Sparse Evaluation Protocol The action space $\mathcal{A}$ expands combinatorially, making exhaustive evaluation computationally prohibitive. However, agent performances exhibit high redundancy

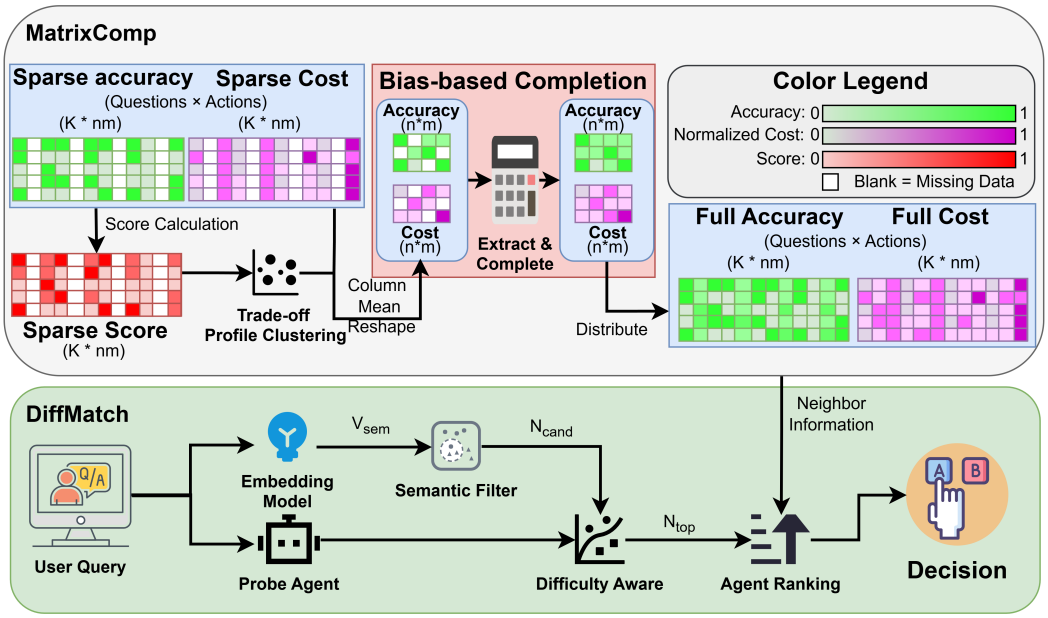

Figure 1: Overview of the proposed **RADAR** framework. The system comprises two decoupled phases: (1) **MatrixComp(TOP):** A structure-aware matrix completion module transforms sparsely evaluated accuracy and cost matrices into dense performance profiles. (2) **DiffMatch(BOTTOM):** An online routing module utilizes a difficulty-calibrated retrieval mechanism for precise decision-making.

due to shared underlying components. As shown in the top of Figure 1, we propose a sparse evaluation protocol that evaluates only a subset of anchor actions $\mathcal{A}_{obs} \subset \mathcal{A}$. The remaining actions $\mathcal{A}_{mask} = \mathcal{A} \setminus \mathcal{A}_{obs}$ remain completely unknown. This module utilizes the structural correlations in $\mathcal{A}_{obs}$ due to the missing entries in $\mathcal{A}_{mask}$.

Spectral Low-Rank Structure Our reconstruction methodology relies on the premise that the ground-truth performance matrix $\mathbf{P} \in \mathbb{R}^{|\mathcal{Q}| \times |\mathcal{A}|}$ possesses a low-rank structure. This is not an arbitrary assumption but a logical consequence of the intrinsic redundancy in agent behaviors. Specifically, LLMs share significant overlaps in their training corpora and architectural inductive biases [23, 24], leading to highly correlated performance patterns. For instance, if a specific agent configuration fails on a query due to a lack of logical reasoning capability, it is highly probable that weaker agents will also fail, while stronger agents may succeed. This implies that the rows (queries) and columns (actions) of $\mathbf{P}$ are not independent but are governed by a small number of latent variables—such as the agent's reasoning bandwidth and the query's intrinsic complexity. Mathematically, this behavioral correlation manifests as a rapid decay in the singular values of $\mathbf{P}$, meaning its effective rank r is significantly smaller than its dimensions ($r \ll \min(|\mathcal{Q}|, |\mathcal{A}|)$).

Leveraging this property, we invoke the Matrix Completion Theory [25], which states that exact recovery of a rank-r matrix is guaranteed with high probability if the number of observed entries $|\Omega|$ satisfies the information-theoretic lower bound:

$$|\Omega| \geq C \cdot \nu \cdot r \cdot n \log^2(n), \tag{3}$$

where $n = \max(|\mathcal{Q}|, |\mathcal{A}|)$, ν is the incoherence parameter, and C is a positive numerical constant.

This theorem provides the rigorous justification for our framework: as long as the agent correlations exist (ensuring low rank) and our sampling budget exceeds this theoretical threshold, we can accurately inference the unobserved entries $\mathcal{A}_{mask}$ with bounded error.

Trade-off Profile Clustering A critical difficulty in routing is that the intrinsic difficulty $\mu(q)$ varies significantly across queries. Traditional methods cluster queries based on semantic embeddings; however, semantic similarity often fails to capture difficulty alignment (*e.g.*, a simple calculation vs.

a complex proof in the same domain). To address this, we introduce Trade-off Profile Clustering. Our hypothesis is that queries exhibiting similar "Accuracy-Cost Trade-off" patterns across the observed actions $\mathcal{A}_{obs}$ share underlying structural complexity. Formally, for each query q_i, we construct a profile vector $\mathbf{S}_i \in \mathbb{R}^{|\mathcal{A}_{obs}|}$. To quantify the efficiency of an action relative to the problem's baseline difficulty, we first successfully identify the minimum cost required to solve q_i. Let $L_i^{\min} = \min_{k:P_{i,k}>0}(\log C_{i,k})$ denote this baseline log-cost. The efficiency score $S_{i,j}$ for action a_j is then defined as:

$$S_{i,j} = P_{i,j} \cdot \left(1 - \frac{\log C_{i,j} - L_i^{\min}}{\max(\log \mathbf{C}_i) - L_i^{\min} + \epsilon}\right), \tag{4}$$

where $\mathbf{C}_i$ represents the vector of costs for all observed actions on q_i, and ϵ ensures numerical stability. This metric yields a normalized score in $[0, 1]$, assigning the highest value to actions that achieve correctness with minimal resource consumption. Finally, we apply K-Means clustering on these profile vectors $\{\mathbf{S}_i\}$ to partition the training set into K_{sco} clusters. Within each cluster, queries are considered difficulty-aligned, as they demonstrate homogeneous performance responses to varying model capabilities.

Bias-based Imputation. Following the clustering process, we perform structure-aware imputation within each identified partition to avoid the over-smoothing caused by standard global regularity algorithms. For any unobserved action $a_{target} = (m^*, s^*) \in \mathcal{A}_{mask}$, we estimate its performance for a query in cluster c by aggregating the global baseline with specific component biases:

$$\hat{P}_c(m^*, s^*) = \mu_c + (\bar{P}_{c,m^*} - \mu_c) + (\bar{P}_{c,s^*} - \mu_c), \tag{5}$$

where μ_c represents the global average accuracy of all observed actions within cluster c, and terms $\bar{P}_{c,m^*}$ and $\bar{P}_{c,s^*}$ denote the marginal average accuracies of the target backbone model m^* and the target strategy s^*, respectively. This imputation logic is symmetrically applied to the cost matrix. Mathematically, Equation (5) acts as a low-rank factorized representation that decomposes agent performance into independent model and strategy biases. By utilizing these deterministic bias terms within local clusters, we explicitly satisfy the theoretical minimal-rank premise while preserving the distinct high-performance signals necessary for effective argmax-based routing. By filling these voids, we transform the sparse observations into dense performance profiles, which serve as the comprehensive knowledge base for the subsequent online routing phase.

3.3 DiffMatch module

The Semantic Gap. The online routing predicts the performance of a new query q_{new} using its nearest neighbors in the completed knowledge base $\mathcal{K}_{aug}$ produced offline by MatrixComp. Standard routing solely rely on semantic embeddings $\mathbf{e}_{sem}(\cdot)$. Semantic similarity does not imply difficulty alignment. Relying on semantic retrieval introduces noise when the routing infers the performance of a hard task from simple tasks. As depicted in the bottom of Figure 1, we propose a difficulty-calibrated retrieval mechanism that incorporates difficulty meta-features to bridge this semantic gap.

Difficulty Meta-Feature Extraction. To bridge the semantic gap, we define the intrinsic complexity of a query using a structured 5-dimensional meta-feature vector $\mathbf{v}_{diff} \in \mathbb{R}^5$ that acts as relative coordinates for difficulty alignment. To obtain these features, we follow the paradigm proposed by [18], utilizing an LLM to directly generate the attribute scores. We initially explore the methodology of [26], which fits mathematical problem difficulty levels using the last hidden layer states of an LLM. However, we observe that its fitting capability degrades significantly when extended to our comprehensive dataset. Therefore, we employ a lightweight probe agent with greedy decoding ($T = 0$) to autonomously score the query across five dimensions: **Logic** (reasoning depth), **Knowledge** (domain specificity), **Constraints** (rule complexity), **Ambiguity** (intent clarity), and **Symbolic** (formal density). Formally, we obtain $\mathbf{v}_{diff}(q) = [v_L, v_{Kn}, v_C, v_A, v_{Sym}]^\top$, providing a structured representation of task hardness that is orthogonal to its semantic topic.

Learnable Two-Stage Retrieval To effectively combine semantic context and difficulty alignment, we implement a Retrieve-and-Rerank strategy (illustrated in the bottom half of Figure 1).

Stage 1: Semantic Coarse Filter. Given a new query q_{new}, we first retrieve a candidate set $\mathcal{N}_{cand}$ of size K_{cand} based on the cosine similarity of their semantic embeddings $\mathbf{e}_{sem}$. This ensures that the retrieved instances are topically relevant.

Stage 2: Difficulty-Aware Reranking. Within the candidate set, we need to identify which neighbors are most predictive of q_{new}'s performance. We formulate this as a regression problem. For each candidate $q_c \in \mathcal{N}_{cand}$, we construct an interaction feature vector $\phi(q_{new}, q_c)$:

$$\phi(q_{new}, q_c) = \text{Concat}\big(\mathbf{v}_{diff}(q_{new}), |\mathbf{v}_{diff}(q_{new}) - \mathbf{v}_{diff}(q_c)|, \mathcal{I}_{inter}\big), \tag{6}$$

where $|\cdot|$ denotes the element-wise absolute difference, capturing the *difficulty gap*, and $\mathcal{I}_{inter}$ represents multiplicative interaction terms. We train a Gradient Boosting Decision Tree (GBDT) regressor f_{rank} to predict the performance deviation between the query and the candidate. The candidates are then reranked by the GBDT regressor, which effectively captures high-order non-linear interactions among the meta-features, and the top-K_{top} neighbors are selected to form the final retrieval set $\mathcal{N}_{top} \subset \mathcal{N}_{cand}$.

Final Decision. Using the reranked neighbors in $\mathcal{N}_{top}$, we estimate the expected accuracy $\hat{P}$ and cost $\hat{C}$ for each action a_j as the average of the completed entries in $\mathcal{K}_{aug}$. The final action is selected according to the objective defined in Equation (7):

$$\hat{a}^* = \arg\max_{a_j} \left(\frac{1}{|\mathcal{N}_{top}|} \sum_{q_n \in \mathcal{N}_{top}} P_{aug}(q_n, a_j) - \lambda \cdot \frac{1}{|\mathcal{N}_{top}|} \sum_{q_n \in \mathcal{N}_{top}} C_{aug}(q_n, a_j) \right), \tag{7}$$

where P_{aug} and C_{aug} are the accuracy and cost entries of $\mathcal{K}_{aug}$. In this process, the Top-K neighbor voting acts as an ensemble mechanism and low-pass filter, effectively smoothing out individual scoring noise and local feature perturbations, so that the selected action better approximates the Oracle's choice.

4 Experiments

4.1 Experimental Setup

Benchmark. We evaluate RADAR on **AgentRoutingBench**, a composite benchmark unifying **MMLU-Pro** [27], **BBH** [28], **MATH500** [29], and the hard subset of **SuperGPQA** [30].

Action Space. Following the formulation in Section 3.1, we instantiate the action space with $|\mathcal{A}| = 12$ distinct configurations, formed by the Cartesian product of 4 backbone models and 3 prompting strategies.

Backbone Models ($\mathcal{M}$). We select four LLMs spanning a wide cost-capability spectrum: `QwenFlash`, `QwenTurbo`, `QwenPlus`, and `DeepSeek-V3`.

Prompting Strategies ($\mathcal{S}$). We implement three strategies to modulate reasoning depth: (1) **Direct Answer** for minimal latency; (2) **Step-Back Reasoning** [31] for abstraction; and (3) **Chain-of-Thought (CoT)** [32] for complex deduction.

Baselines. We compare RADAR against standard routing paradigms in RouterBench [8]:

- **Zero routing (Lower Bound):** A query-agnostic baseline that constructs a *Non-Decreasing Convex Hull* from global statistics to determine the optimal probabilistic agent mix for a given cost.
- **Predictive routing (KNN, MLP):** Supervised regressors that predict agent performance based on query embeddings. We implement KNN ($k = 25$) and MLP (2-layer ReLU) as representative baselines.
- **Recent Routing Methods (Attention, Please [21], UniRoute [22]):** To broaden the comparison beyond RouterBench, we additionally reimplement two recent routing methods following their original papers. As their official code is not publicly released, these reimplementations may underestimate the original methods (see limitations).

Table 1: **AIQ Performance Comparison.** BCR is our primary efficiency metric: it denotes the fraction of the action space that must be evaluated to train a routing. The second line of each BCR label additionally reports the corresponding monetary benchmarking cost (USD), which we give only to convey the scale; since all routing methods with the same BCR are trained on the same sparsely evaluated matrix, they incur an identical cost. For the 66% group this cost is the *average* over the four balanced subsampling schemes, whose individual costs span \$363.61–\$406.03 (Appendix D).

BCR	Method	MMLU-Pro	MATH500	BBH	SuperGPQA	AgentRoutingBench
100% \$554.53	Zero routing	0.7630	0.9190	0.8502	0.4448	0.7275
	MLP routing	0.7892	0.9245	0.8773	0.4457	0.7249
	KNN routing	0.7914	0.9251	0.8789	0.4558	0.7306
	Attention, Please	0.7881	0.9249	0.8764	0.4473	0.7236
	UniRoute	0.7903	0.9217	0.8784	0.4492	0.7264
66% avg. \$381.11	Zero routing	0.7383	0.8900	0.8327	0.4228	0.6823
	MLP routing	0.7788	0.9003	0.8553	0.4163	0.7053
	KNN routing	0.7803	0.9120	0.8658	0.4284	0.7120
	Attention, Please	0.7775	0.9010	0.8575	0.4150	0.7065
	UniRoute	0.7780	0.8990	0.8560	0.4170	0.7045
	RADAR (Ours)	**0.7975**	**0.9194**	**0.8745**	**0.4544**	**0.7296**
Improvement vs. KNN (BCR=66%)		+2.20%	+0.81%	+1.00%	+6.07%	+2.47%
Difference vs. KNN (BCR=100%)		+0.77%	-0.61%	-0.50%	-0.29%	-0.15%

Implementation Details. We utilize BAAI/bge-m3 [33] as the semantic encoder. We employ `qwen3-4B` as the lightweight probe agent for difficulty meta-feature extraction, prompting it to score queries across 5 dimensions. For MatrixComp, we set the trade-off profile cluster count to $K_{sco} = 20$. For DiffMatch, the GBDT reranker selects from $K_{cand} = 50$ candidates, with the top $K_{top} = 25$ used for voting. Regarding operational latency, the online routing process executes locally in under 100ms with zero API cost. This negligible overhead contrasts sharply with the 32.0s average inference time of frontier models.

Evaluation Metrics. We employ a dual-metric protocol to assess both outcome quality and data efficiency:

- **Average Improvement in Quality (AIQ) [8]:** Following the standard protocol, we quantify routing efficacy by calculating the area under the Accuracy-Cost Pareto curve, normalized by the operational cost range. This metric represents the routing's expected accuracy across the entire cost spectrum:
$$\text{AIQ} = \frac{\int_{c_{\min}}^{c_{\max}} \text{Accuracy}(c)\, dc}{c_{\max} - c_{\min}}. \tag{8}$$
- **Benchmarking Cost Ratio (BCR):** Highlights data efficiency by quantifying the percentage of the action space required for training:
$$\text{BCR} = \frac{|\mathcal{A}_{obs}|}{|\mathcal{A}|} \times 100\%. \tag{9}$$
Our evaluation goal is to achieve AIQ Parity with full-observation baselines while operating under a constrained BCR.

4.2 Experimental Results and Analysis

Results Presentation Table 1 presents the comparative analysis of RADAR against traditional routing baselines. As evidenced in the AgentRoutingBench column, RADAR achieves an AIQ of 0.7296 with a 66% BCR. This performance is statistically indistinguishable from the KNN routing trained on the full 100% dataset (0.7306). Under the identical 66% BCR constraint, RADAR consistently outperforms standard semantic baselines across all benchmarks, yielding a comprehensive 2.47% improvement in overall AIQ compared to the sparse KNN baseline. The superiority of RADAR is most pronounced on tasks requiring deep domain expertise and complex reasoning. On the SuperGPQA subset, RADAR surpasses the sparse KNN routing by a substantial margin of 6.07%, and we observe a 2.20% improvement on the MMLU-Pro benchmark. Furthermore, a counter-intuitive observation emerges on the MMLU-Pro benchmark, where RADAR trained with a 66% BCR outperforms even the full-observation KNN baseline trained with a 100% BCR by 0.77%.

Result Analysis These empirical achievements stem from the specific architectural designs of RADAR. First, the parity with full-observation baselines validates that the global performance manifold of multiagent systems possesses a low-rank structure. MatrixComp effectively recovers this structure, allowing the routing to approximate full-observation performance without incurring the exhaustive benchmarking overhead. Second, the pronounced gains on complex tasks are driven by the joint modeling of backbones and strategies rather than isolating either dimension. Standard routing often suffer from a semantic collapse in these domains because they rely solely on surface-level embedding similarity. They fail to distinguish between superficially similar queries that require vastly different reasoning strategies. RADAR mitigates this issue by leveraging structural correlations between agent configurations to identify the latent capabilities required for the solution. Finally, the phenomenon of outperforming baselines trained on more data is attributed to the denoising effect of our DiffMatch difficulty matching mechanism. Traditional semantic retrieval often introduces distractor neighbors, which are historical queries that relate topically to the target but possess mismatched intrinsic difficulty. By explicitly aligning queries based on a 5-dimensional difficulty meta-feature vector, RADAR actively filters out these difficulty-misaligned neighbors, proving that alignment quality is more critical than sheer data quantity for routing precision.

In conclusion, beyond raw performance gains, these experiments underscore the resource efficiency of RADAR. Achieving performance parity with only 66% of the action space observations reduces the benchmarking overhead required for routing construction by over 33%, without any measurable loss in routing quality. This efficiency stems from the ability of RADAR to decouple semantic topics from reasoning complexity. By characterizing the latent factors of agent behaviors, RADAR transforms the combinatorial explosion of multiagent evaluation into a manageable matrix completion task. This makes the deployment of high-performing multiagent systems economically viable for real world applications where the action space expands dynamically.

4.3 Interaction between Model and Strategy

Table 2: **Interaction between backbone and prompting strategy.** (a) Distribution of the optimal strategy for each backbone. (b) Distribution of the optimal backbone for each strategy. All values are percentages of queries aggregated over the four benchmarks; in (b) the remaining mass belongs to `QwenFlash` (5.1% under *Direct*, 2.7% under *Step-Back*, 1.9% under *CoT*).

(a) Optimal strategy per backbone (%)

Backbone	**Direct**	**Step-Back**	**CoT**
`DeepSeek-V3`	55.5	36.2	8.3
`QwenTurbo`	67.8	24.9	7.3
`QwenFlash`	62.2	29.9	7.8
`QwenPlus`	58.0	31.1	10.8

(b) Optimal backbone per strategy (%)

Strategy	`QwenPlus`	`QwenTurbo`	`DeepSeek-V3`
Direct	64.0	21.3	9.6
Step-Back	75.8	14.4	7.1
CoT	76.4	6.7	15.0

Whether a backbone m and a prompting strategy s should be treated as a single joint action is an empirical question. If the two decisions were separable, a factored routing that independently chooses them would be sufficient, and the joint action space of Section 3.1 would be unnecessarily large. We therefore examine, for every query, which strategy is optimal under each backbone and which backbone is optimal under each strategy, aggregating the results over all four benchmarks.

Two effects emerge (Table 2). First, the value of a strategy is backbone-dependent: *Direct* is the most frequently optimal strategy for every backbone, yet its share ranges from 55.5% for `DeepSeek-V3` to 67.8% for `QwenTurbo`. Second, the relative ranking of backbones shifts with the strategy: although `QwenPlus` dominates under every strategy, `QwenTurbo` accounts for 21.3% of the optimal picks under *Direct* but only 6.7% under *CoT*. Such a cross-over cannot be represented by factored decisions that select a model and a strategy independently, which motivates routing over the joint action space in which RADAR operates.

4.4 Empirical Validation of Low-Rank Assumption.

To justify the theoretical feasibility of our MatrixComp module, we analyze the spectral properties of the agent performance manifold via singular value decomposition. Figure 2 compares the cumulative explained variance of the singular values for the global performance matrix versus the sub-matrices derived from our Trade-off Profile Clustering. The results empirically validate the low-rank hypothesis. Even the raw global data exhibits a rapid spectral decay, requiring only a rank of $r = 4$ to capture 90.7% of the system variance. Furthermore, the analysis reveals that our clustering mechanism enhances this property by partitioning queries into difficulty-aligned groups. The effective rank of the local manifolds is further reduced to $r = 3$ (capturing 91.6% variance). This confirms that the agent behavior matrix satisfies the low-rank condition required by matrix completion theory [25], and that our clustering strategy effectively regularizes the data structure to ensure robust recovery under sparse observations.

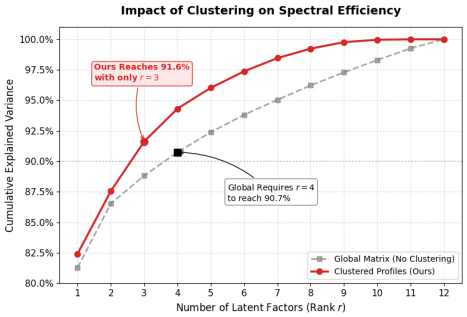

Figure 2: **Empirical Validation of Low-Rank Assumption.**

4.5 Ablation Study

To assess the individual contributions of each component in RADAR, we conduct an ablation study under the 66% BCR constraint. We evaluate three variants by systematically removing key modules:

- **w/o MatrixComp (w/o MC):** Removes the entire offline structure-aware matrix completion phase. The routing operates directly on the sparse observations $\mathcal{A}_{obs}$, verifying the necessity of recovering global performance profiles via MatrixComp.
- **w/o DiffMatch (w/o DM):** Retains the completed matrix but replaces the DiffMatch module (specifically the difficulty-calibrated retrieval) with standard semantic retrieval based solely on embedding similarity. This tests the hypothesis that semantic alignment is insufficient for routing hard tasks.
- **w/o Profile Clustering (w/o PC):** A variant of MatrixComp that performs matrix completion globally on the entire dataset without partitioning queries based on their Trade-off Profiles. This isolates the benefit of the structure-aware clustering strategy described in Section 3.2.

Table 3: **Ablation Study of RADAR.**

Configuration	MMLU-Pro	MATH500	BBH	SuperGPQA	AgentRoutingBench
RADAR (Full)	**0.7975**	**0.9194**	**0.8745**	**0.4544**	**0.7296**
w/o MC	0.7820	0.9130	0.8670	0.4315	0.7150
w/o DM	0.7860	0.9150	0.8695	0.4390	0.7215
w/o PC	0.7895	0.9175	0.8725	0.4480	0.7268
Ref: KNN (BCR=66%)	*0.7803*	*0.9120*	*0.8658*	*0.4284*	*0.7120*
Ref: KNN (BCR=100%)	*0.7914*	*0.9251*	*0.8789*	*0.4558*	*0.7306*

Table 3 summarizes the results. We observe that all components contribute positively to the final performance, with the following insights:

Impact of MatrixComp. Removing the MatrixComp module (*w/o MC*) results in the most significant performance drop, with the Average AIQ decreasing from 0.7296 to 0.7150. This confirms that sparse evaluation data severely limits the routing's ability to generalize. Without the dense performance profiles reconstructed by MatrixComp, the routing effectively reverts to a standard sparse retrieval method, struggling to find reliable neighbors for unseen queries.

Necessity of DiffMatch. The removal of the DiffMatch module (*w/o DM*) leads to a notable decline, bringing the AIQ down to 0.7215. This drop is particularly pronounced on reasoning-intensive benchmarks like SuperGPQA, where performance falls from 0.4544 to 0.4390. This

validates our claim regarding the "Semantic Gap": even with a completed matrix, relying solely on semantic similarity causes the system to mismatch queries with superficially similar but structurally different historical examples, thereby selecting suboptimal agents.

Effectiveness of Structure-Aware Clustering. Finally, the *w/o PC* variant achieves an AIQ of 0.7268, which is superior to the other variants but still lower than the full RADAR framework. This demonstrates that the Trade-off Profile Clustering within MatrixComp contributes to finer-grained recovery. By grouping difficulty-aligned queries, the clustering mechanism lowers the effective rank of local manifolds, allowing for more accurate imputation than a global low-rank approximation.

4.6 Hyperparameter Sensitivity

We determine the optimal hyperparameters strictly via the validation set to ensure no leakage. For matrix completion, the strategy clustering count K_{sco} is selected to minimize the validation MAE of recovered values. For reranking, the candidate pool size K_{cand} and neighbor count K_{top} are tuned to maximize the validation AIQ. We search $K_{sco} \in [5, 100]$, $K_{cand} \in [10, 100]$, and $K_{top} \in [5, 50]$. Optimal performance is observed within broad plateaus around $K_{sco} \in [15, 30]$, $K_{cand} \in [40, 60]$, and $K_{top} \in [20, 40]$. The final values are $K_{sco} = 20$, $K_{cand} = 50$, and $K_{top} = 25$, selected from these plateaus.

5 Conclusion

In this paper, we introduce RADAR, a cost-effective framework designed to tackle the prohibitive benchmarking costs and semantic limitations in complex multi-agent routing. By synergizing structure-aware matrix completion with a difficulty-calibrated retrieval mechanism, RADAR effectively reconstructs global agent performance profiles from extremely sparse observations. This work establishes a scalable new paradigm for efficient agent orchestration, paving the way for more economically viable and robust intelligent multi-agent systems.

Acknowledgments and Disclosure of Funding

This work received no specific funding.

We are grateful to the members of our laboratory for insightful discussions and generous help throughout this project, and to our families for their unwavering support and encouragement.

The authors declare no competing interests.

References

[1] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[2] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023.

[3] Qingyun Wu et al. Autogen: Enabling next-gen LLM applications via multi-agent conversation. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

[4] Chen Qian et al. ChatDev: Communicative agents for software development. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[5] Sirui Hong et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024.

[6] Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. MasRouter: Learning to route LLMs for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, Vienna, Austria, 2025. Association for Computational Linguistics.

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

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

[9] Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024.

[10] Marah Abdin et al. Phi-3 technical report: A highly capable language model locally on your phone, 2024.

[11] Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Association for Computational Linguistics, June 2024.

[12] Aman Madaan et al. Automix: Automatically mixing language models, 2023.

[13] Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing, 2024.

[14] Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for llm selections, 2025.

[15] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate, 2023.

[16] Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. Multi-agent collaboration via evolving orchestration. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[17] Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *Proceedings of the 42nd International Conference on Machine Learning*. PMLR, 2025.

[18] Ao Li, Yuexiang Xie, Songze Li, Fugee Tsung, Bolin Ding, and Yaliang Li. Agent-oriented planning in multi-agent systems. In *The Thirteenth International Conference on Learning Representations*, 2025.

[19] Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. *arXiv preprint arXiv:2502.04180*, 2025.

[20] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise comparison and generative fusion. In *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics (ACL 2023)*, 2023.

[21] Roshini Pulishetty et al. Attention, please: Single-head cross-attention for unified LLM routing. In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*, 2025.

[22] Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. Universal model routing for efficient LLM inference. In *International Conference on Learning Representations (ICLR)*, 2026.

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

[24] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, et al. Gpt-4 technical report, 2024.

[25] Emmanuel J. Candes and Benjamin Recht. Exact matrix completion via convex optimization, 2008.

[26] Sunbowen Lee, Qingyu Yin, Chak Tou Leong, Jialiang Zhang, Yicheng Gong, Shiwen Ni, Min Yang, and Xiaoyu Shen. Probing the difficulty perception mechanism of large language models, 2025.

[27] Yubo Wang et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

[28] Mirac Suzgun et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

[29] Hunter Lightman et al. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[30] Xinrun Du et al. SuperGPQA: Scaling LLM evaluation across 285 graduate disciplines. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025.

[31] Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. Take a step back: Evoking reasoning via abstraction in large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[32] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[33] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024.

A Prompt Configurations

In this section, we detail the comprehensive set of system and user prompts employed within the RADAR framework. This includes the prompts utilized by the lightweight probe agent for extracting difficulty meta-features, as well as the prompting strategies applied to the backbone Large Language Models to modulate reasoning depth.

A.1 Meta-Feature Extraction Prompt

To bridge the semantic gap, RADAR employs a lightweight probe agent to score each query along five difficulty dimensions. The complete probe prompt is reproduced below. The probe returns a five-element score vector and nothing else, which makes the extracted meta-features directly usable by the reranking stage described in Section 3.3.

Prompt: Five-Dimensional Difficulty Scoring

```
[System Prompt]
Analyze the question across 5 dimensions (Score 1-5):
L (Logic): 1:Simple fact, 3:Multi-step reasoning, 5:Complex logical proof.
K (Knowledge): 1:Common sense, 3:Academic/College level, 5:Highly specialized/
    Professional.
C (Constraints): 1:No limits, 3:2-3 strict rules, 5:Complex/Harsh constraints.

A (Ambiguity): 1:Crystal clear, 3:Implied intent, 5:Vague/Trap-filled.
S (Symbolic): 1:Pure text, 3:Simple Math/Code, 5:Dense formulas/Algorithms.
Output format ONLY: [L,K,C,A,S]

### Problem Content:
{input_question}

### Difficulty Scores:
```

A.2 Reasoning Strategy Prompts

To populate the dynamic action space, RADAR combines the backbone models with three distinct reasoning strategies: Direct Answer, Step-Back Reasoning, and Chain-of-Thought (CoT).

A.2.1 Direct Answer

Designed for minimal latency and computational overhead, forcing a zero-shot extraction without intermediate steps.

Prompt Strategy: Direct Answer

```
[System Prompt]
You are a strictly automated answer extraction engine.
Your mandate is to provide the final result in the most concise format
    possible.

OPERATIONAL PROTOCOLS:
1. **Zero-Shot Extraction**: Do not compute, derive, or explain. Output the
    result immediately.
2. **Adaptive Formatting**:
   - If the user asks a Multiple Choice Question: Output ONLY the option
    letter (e.g., "A", "B").
   - If the user asks a Math Question: Output ONLY the numeric value or
    simplified expression.
   - If the user asks for a Code Snippet: Output ONLY the code block.
```

```
   - For Open Questions: Output the core entity, date, or key phrase without
     full sentences.
3. **Silence Protocol**: Never use phrases like "The answer is", "Sure", "
    Here is", or "Based on".

[User Prompt]
Question: {question}

Target Answer:
```

A.2.2 Step-Back Reasoning

Aimed at eliciting reasoning via abstraction by identifying core principles before specific application.

Prompt Strategy: Step-Back

```
[System Prompt]
You are an expert reasoning engine using **Step-Back Prompting** methodology.
STRICT INSTRUCTIONS:
1. **Identify Core Principle**: Extract ONLY the fundamental concept
    governing this problem (1 sentence maximum).
2. **Direct Application**: Apply the principle directly to derive the answer
    in minimal steps.
3. **Token Efficiency**: Be extremely concise - no explanations,
    justifications, or examples.

RESPONSE FORMAT:
**Principle**: [Single core concept]
**Application**: [few step application]
**Answer**: [Final answer only]

[User Prompt]
Question: {question}

Apply Step-Back reasoning:
```

A.2.3 Chain-of-Thought (CoT)

A rigorous, systematic generation prompt designed for complex logic and extensive deductive steps.

Prompt Strategy: Chain-of-Thought (CoT)

```
[System Prompt]
You are a rigorous logical reasoning engine designed for complex problem
    solving.
Your output must follow a strict **Linear Derivation Path**.

DERIVATION RULES:
1. **Plan & Execute**: briefly scan the constraints, then execute step-by-
    step.
2. **No Internal Monologue**: Do NOT output your drafting process (e.g.,
    avoid "Wait," "Let me check," "Maybe"). If a step is invalid, correct it
    internally before outputting text.
3. **Standard Assumptions**: Unless explicitly stated otherwise, assume
    standard academic contexts (e.g., standard biological states, Euclidean
    geometry).
4. **Step Numbering**: Use "Step 1:", "Step 2:" structure for clarity.

REQUIRED ENDING:
```

```
You must conclude your response exactly with: "[Final Answer Only]"

[User Prompt]
Question: {question}

Provide a comprehensive step-by-step derivation:
```

B Action Leaderboard and Difficulty Analysis

In this section, we present a granular empirical analysis of the action space performance and the intrinsic difficulty across various problem categories.

Figure 3 displays a comprehensive heatmap that evaluates the effectiveness of different agent configurations. The numeric values in the action columns indicate the probability of a specific action being ranked within the top 3 most cost-effective and accurate solutions for a given domain. Furthermore, to quantify the absolute hardness of the tasks, the analysis incorporates a `<No_Solution>` metric. This specific column represents the probability that all available models in the action space completely failed to produce a correct answer for the queries in that category.

For optimal visualization, the actions (columns) are ordered sequentially by their global popularity across all datasets. Concurrently, the problem categories (rows) are sorted in descending order based on the highest frequency value achieved within each category, thereby providing a clear visual hierarchy of domain-specific challenges and model capabilities.

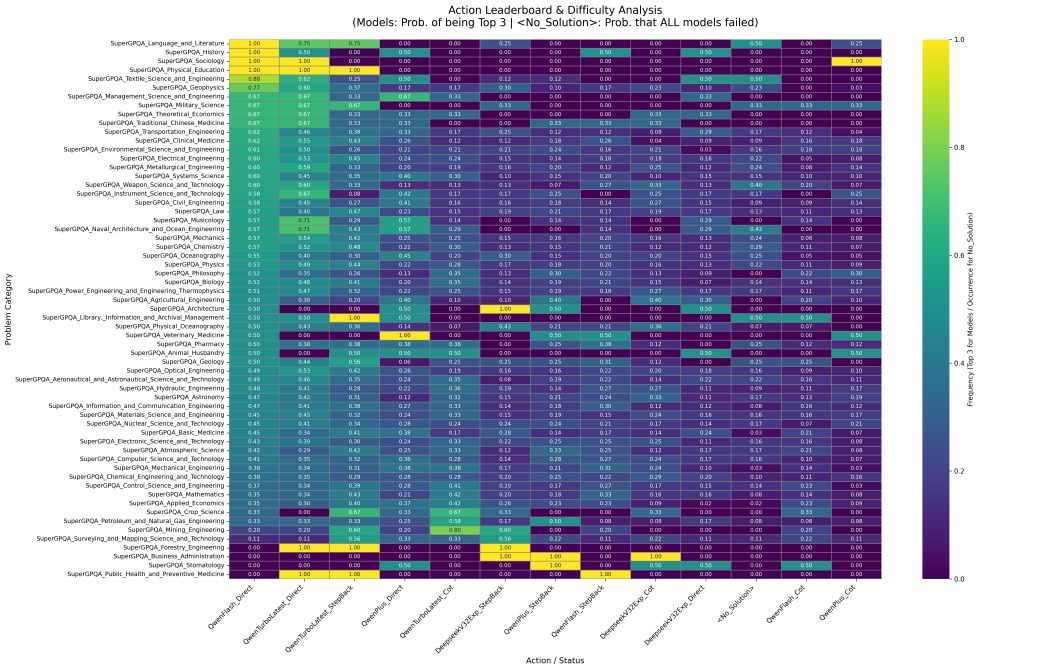

Figure 3: Action Leaderboard and Difficulty Analysis. The heatmap illustrates the probability of specific actions ranking in the top 3, alongside the probability of complete system failure (`<No_Solution>`) across different problem categories.

C Visualization of the Semantic Gap

As discussed in Section 3.3, routing models based solely on semantic embedding similarity introduces a critical bottleneck, which we define as the "Semantic Gap". To empirically validate this phenomenon

and explain why direct performance prediction using standard hidden layer states struggles on comprehensive multi-agent datasets, we project the semantic space of the user queries.

Figure 4 presents a 2D projection of the semantic embeddings generated via Uniform Manifold Approximation and Projection (UMAP). Each data point represents an individual query, and the color coding denotes its corresponding *Oracle Label*—the optimal action configuration (model and strategy pair) that successfully solves the query with minimal computational cost.

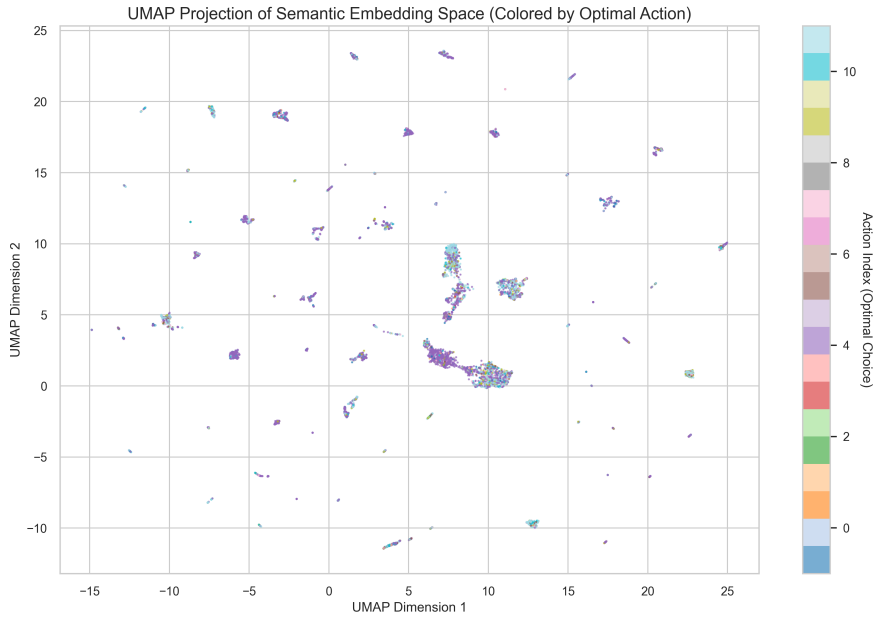

Figure 4: UMAP Projection of the Semantic Embedding Space. Points represent individual queries, colored by their theoretically optimal action index. The distinct spatial clusters represent broad topical domains. However, the severe color mixing within these clusters indicates that semantically similar queries frequently require vastly different computational resources and reasoning strategies.

The visualization reveals a highly characteristic distribution. While the UMAP algorithm successfully organizes the queries into distinct topological clusters (islands representing different broad topics or domains), the intra-cluster distribution is highly heterogeneous. Rather than forming homogeneous regions where specific agents uniformly excel, the colors are heavily intermixed within almost every single cluster.

This structural chaos visualizes the precise nature of the Semantic Gap. It demonstrates that queries sharing high topical or semantic proximity often demand completely divergent reasoning capabilities. For instance, a basic factual calculation and an advanced theoretical proof may be situated adjacently within the embedding space due to shared vocabulary, yet they require fundamentally different routing decisions. Consequently, standard semantic-based routing methods (such as generic KNN or MLP baselines) are prone to severe misclassifications in these dense regions. This severe misalignment underscores the necessity of the proposed **DiffMatch** module, proving that extracting orthogonal difficulty meta-features is critical for untangling semantic overlap and ensuring precise agent routing on complex tasks.

D Benchmarking Cost Analysis

Cost of a full evaluation. Table 4 reports the cost of exhaustively evaluating all 12 actions on each constituent dataset of AgentRoutingBench(25,857 queries in total).

Cost at 66% BCR. Because the absolute cost of a 66% sample depends on *which* actions are observed, we adopt BCR as the primary, sampling-ratio-based metric and report the corresponding monetary cost alongside it. Table 5 lists the four balanced subsampling schemes used in the main

Table 4: **Benchmarking cost at full evaluation (BCR = 100%).**

Dataset	Queries	Cost (USD)
MMLU-Pro	12,032	$194.42
BBH	6,511	$109.03
MATH500	500	$6.19
SuperGPQA	6,814	$244.90
Total	**25,857**	**$554.53**

experiments: in each scheme every backbone drops exactly one strategy, and every strategy is dropped by at least one backbone. The average cost is $381.11 (range $363.61–$406.03), i.e., a saving of $173.42 (31.3%) relative to the full $554.53 evaluation, at a cost of less than 0.15% AIQ. Note that this 31.3% is a *monetary* saving and should not be conflated with the reduction in evaluation volume: measured in the number of evaluated action–query pairs, BCR drops from 100% to 66%, i.e., a 34% reduction. We report BCR rather than the monetary figure as the primary metric because the latter depends on which specific actions are sampled.

Table 5: **Benchmarking cost of the four balanced 66% subsampling schemes.**

Scheme	Dropped (backbone:strategy) pairs	Cost (USD)
1	DS:StepBack, QF:CoT, QP:Direct, QT:Direct	$374.12
2	DS:CoT, QF:StepBack, QP:StepBack, QT:Direct	$380.67
3	DS:Direct, QF:StepBack, QP:CoT, QT:StepBack	$363.61
4	DS:StepBack, QF:Direct, QP:StepBack, QT:CoT	$406.03
Average (RADAR)		**$381.11**

E Reranker Choice

Choice of reranker. Table 6 compares several regressors for the difficulty-aware reranking stage on MMLU-Pro, using the same interaction features and the same retrieved candidate sets. GBDT attains the best AIQ, although the margin over the other tree-based methods is modest. This is expected: the interaction features already encode part of the non-linearity, so the reranker mainly needs to combine them robustly across heterogeneous scales—which is exactly where tree ensembles are strong.

Table 6: **Reranker comparison on MMLU-Pro.**

Regressor	AIQ (MMLU-Pro)
Linear Regression	0.7938
2-layer MLP	0.7955
Random Forest ($n = 200$)	0.7967
GBDT (XGBoost)	**0.7975**
GBDT (LightGBM)	0.7972

F Limitations

While the proposed RADAR framework demonstrates robust empirical performance and significantly mitigates the prohibitive costs of comprehensive benchmarking, it presents certain limitations that warrant further investigation:

Reimplementation of Recent Baselines. Two recent routing methods, *Attention, Please* [21] and *UniRoute* [22], are included in our comparison, but their official implementations are not publicly available. We therefore reimplemented both methods following their original papers, without access to the authors' tuning recipes. These reimplementations may consequently underestimate the performance of the original methods, and the comparison against them should be interpreted with this

caveat in mind; the comparison against the open-source RouterBench baselines (Zero, MLP, KNN) is unaffected.

Dependency on the Low-Rank Assumption. The efficacy of the *MatrixComp* module fundamentally relies on the premise that the agent performance matrix possesses a low-rank structure. Although Section 4.4 empirically validates this assumption across current foundational models and reasoning strategies, the emergence of highly specialized or completely orthogonal agents lacking shared architectural inductive biases may diminish this structural redundancy. In such edge cases, the accuracy of the bias-based matrix imputation could degrade.

Probe Agent Capacity Bottleneck. To bridge the semantic gap, the *DiffMatch* module employs a lightweight probe agent (e.g., `qwen3-4B`) to extract 5-dimensional difficulty meta-features via zero-shot prompting. Consequently, the routing precision is upper-bounded by the comprehension capacity of this probe agent. For queries involving extremely esoteric disciplines or deliberately convoluted logical constraints, the probe might fail to accurately assess the intrinsic hardness, generating noisy meta-features that could mislead the retrieval and reranking process.

Adaptability to Dynamic Action Spaces. RADAR operates on a two-phase paradigm requiring an offline sparse evaluation protocol. While it successfully reduces the benchmarking budget to 66% of the full action space $\mathcal{A}$, introducing a completely novel LLM backbone or prompting strategy post-deployment currently necessitates a partial re-evaluation of the anchor actions $\mathcal{A}_{obs}$ and a re-execution of the *MatrixComp* clustering pipeline. Future research will explore incremental matrix completion and online learning techniques to facilitate a more seamless, plug-and-play multi-agent ecosystem.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the research claims, including the proposed RADAR framework, MatrixComp, DiffMatch, and AgentRoutingBench, which are consistent with the experimental results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the limitations of the proposed RADAR framework, including its dependency on the low-rank assumption, probe agent capacity bottlenecks, and its adaptability to dynamic action spaces, in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper explicitly states the low-rank assumption of the performance matrix and provides the theoretical lower bound formula based on Matrix Completion Theory.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper fully discloses the experimental settings, including the benchmark datasets used, the backbone models with pricing, prompting strategies, baselines, and hyperparameters.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: While the paper details the experimental configurations, it does not currently provide open access links to the code and data.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setup specifies all necessary training and test details, such as the cluster counts and candidate pool sizes, which were strictly selected via a validation set.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper reports Average Improvement in Quality (AIQ) as the main metric but does not include error bars or statistical significance tests in the results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The paper mentions the online routing execution time and API costs, but lacks details on the specific hardware used for offline matrix completion and GBDT training.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research focuses on multi-agent routing algorithms and conforms to standard ethical guidelines without violating any NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper mentions the positive impact of making multi-agent systems more economically viable in the conclusion, but does not explicitly discuss potential negative societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The proposed RADAR framework is a routing mechanism and does not involve releasing pre-trained language models, image generators, or datasets with a high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The paper references existing assets like MMLU-Pro and Qwen models, but does not explicitly list their license information or terms of use in the text.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper introduces a new composite benchmark named AgentRoutingBench and documents its composition of four high-difficulty datasets in the experimental setup.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This research is strictly computational and focuses on LLM agent routing; it does not involve any crowdsourcing experiments or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: Since the research does not involve human subjects, Institutional Review Board (IRB) approvals are not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The paper clearly describes the usage of an LLM as a lightweight probe agent to extract difficulty meta-features, which is a core methodological component.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.