SkillOrchestra: Learning to Route Agents via Skill Transfer

Jiayu Wang[1]* **Yifei Ming**[2] **Zixuan Ke**[2] **Shafiq Joty**[2] **Aws Albarghouthi**[1] **Frederic Sala**[1]

[1]University of Wisconsin-Madison [2]Salesforce AI Research

Abstract

Compound AI systems promise capabilities beyond those of individual models, yet their success depends critically on effective orchestration. Existing routing approaches face two limitations: (1) input-level routers make coarse query-level decisions that ignore evolving task requirements; (2) RL-trained orchestrators are expensive to adapt and often suffer from *routing collapse*, repeatedly invoking one option in multi-turn scenarios. We introduce `SkillOrchestra`, a framework for **skill-aware orchestration**. Instead of directly learning a routing policy end-to-end, `SkillOrchestra` learns fine-grained skills from execution experience and models agent-specific competence and cost under those skills. At deployment, the orchestrator infers the skill demands of the current interaction and selects agents that best satisfy them under an explicit performance-cost trade-off. Extensive experiments across ten benchmarks demonstrate that `SkillOrchestra` outperforms SoTA RL-based orchestrators by up to 27.5% with 700× and 300× learning cost reduction compared to Router-R1 and ToolOrchestra, respectively. These results show that explicit skill modeling enables scalable, interpretable, and sample-efficient orchestration, offering a principled alternative to data-intensive RL-based approaches. The code is available at: https://github.com/jiayuww/SkillOrchestra.

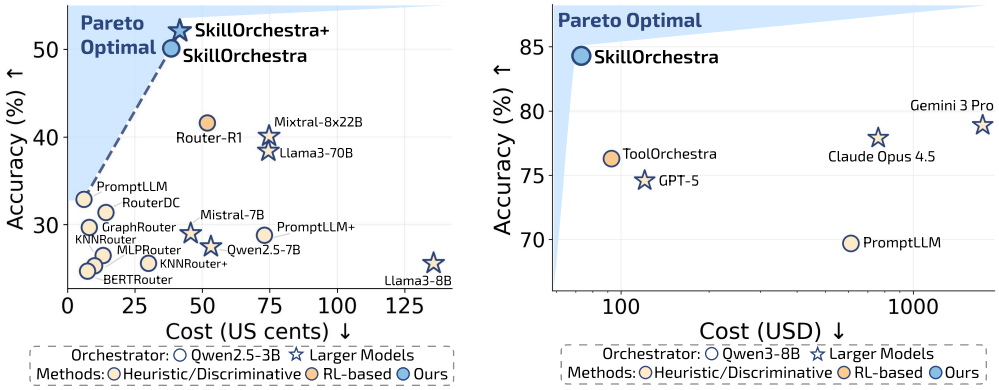


Figure 1: Performance-cost tradeoffs in multi-turn model routing (left) and agent orchestration (right). SkillOrchestra achieves higher performance at lower or comparable cost than all baselines.

1 Introduction

Modern AI systems are increasingly built as compound agents that coordinate multiple large language models (LLMs) and tools to solve complex, multi-step tasks such as deep research [Gemini,

*Correspondence: Jiayu Wang (milawang@cs.wisc.edu).

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

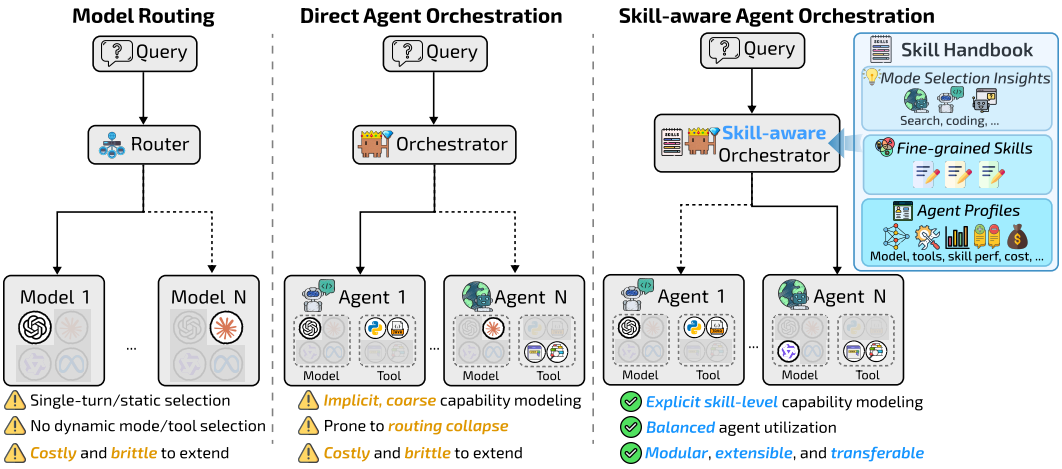


Figure 2: Comparison of orchestration approaches. **(Left)** Model routing performs static, query-level model selection without dynamic mode or tool reasoning. **(Middle)** Direct agent orchestration learns routing end-to-end with implicit capability modeling and is prone to routing collapse. **(Right)** Skill-aware agent orchestration uses a reusable Skill Handbook with explicit skill-level competence and cost estimates, enabling balanced agent utilization, transferability, and extensibility.

2024, OpenAI, 2025b] and scientific discovery [Gottweis et al., 2025]. Rather than relying on a single model, these systems interleave operations such as web search, code execution, and answer synthesis, dynamically invoking models with different strengths and costs [Ke et al., 2025]. In this setting, *orchestration*, the process deciding what capability is needed at each interaction state and which model–tool combination should execute it, is central to both performance and efficiency.

A common approach to efficient orchestration is model routing, where a controller selects a model from a candidate pool [Chen et al., 2024a, Hu et al., 2024, Ong et al., 2025]. However, most routing methods make a single query-level decision, implicitly assuming that one model is sufficient for the entire task. This assumption breaks down in agentic workflows, where different interaction states often require different operational modes and capabilities. For example, a task may require retrieving external evidence, performing symbolic or numerical computation, and synthesizing a final answer. Routing should therefore be state-conditioned and capability-aware, rather than treating the original query as a single static decision unit (Figure 2, left).

Recent RL-based orchestration methods [Zhang et al., 2025a, Su et al., 2025] address this by learning sequential routing policies with LLMs. While more flexible than single-shot routing, these methods introduce a different set of challenges. They require expensive policy training, are difficult to adapt when the model or tool pool changes, and can suffer from *routing collapse*: the learned policy repeatedly selects a single option at one or more decision levels, despite the availability of alternatives with better accuracy–cost trade-offs (Figure 2, middle). This collapse is especially problematic in compound systems, where the value of orchestration comes precisely from matching heterogeneous agents to heterogeneous task demands.

We introduce **SkillOrchestra**, a framework for *skill-aware orchestration*. Instead of directly learning a monolithic routing policy end-to-end, SkillOrchestra learns a reusable *Skill Handbook* from execution experience. The handbook externalizes orchestration knowledge into three components: (i) mode-level execution insights that guide high-level decisions about what operation to perform, (ii) a hierarchical registry of fine-grained skills that characterize capability requirements within each mode, and (iii) agent profiles that estimate skill-conditioned competence and execution cost. At deployment, the orchestrator uses the selected handbook to identify skills required by the current state and route to agents that best satisfy those skills under an explicit performance–cost trade-off (Figure 2, right).

This skill-centric perspective provides three advantages. First, it enables state-conditioned orchestration: the system can route different steps of the same task to different agents according to the skills required at each state. Second, it improves routing stability by making competence estimates explicit, reducing the tendency of end-to-end policies to collapse onto a single strong but costly

agent. Third, it makes orchestration knowledge reusable: the learned handbook can transfer across orchestrator backbones and can be updated as model and tool pools evolve, without retraining a full routing policy.

We evaluate SkillOrchestra in both multi-turn model routing and full agent orchestration settings. As shown in Figure 1, SkillOrchestra and SkillOrchestra+ improve the performance–cost frontier over heuristic, discriminative, and RL-based baselines. Across ten benchmarks, SkillOrchestra consistently improves accuracy, reduces cost, and yields more balanced routing behavior. For example, compared to state-of-the-art RL-trained orchestrators, SkillOrchestra achieves up to 27.5% absolute improvement while reducing learning cost by 700× and 300× relative to Router-R1 [Zhang et al., 2025a] and ToolOrchestra [Su et al., 2025], respectively. The learned handbook also transfers across orchestrator models without retraining. We summarize our contributions as follows:

❶ **Skill-aware orchestration.** We propose SkillOrchestra, a framework that structures orchestration around explicit skills and agent profiles, enabling state-conditioned, performance–cost-aware routing.

❷ **Skill Handbook learning.** We introduce a data-efficient procedure to discover and refine reusable skills from execution traces, while estimating skill-specific agent competence and cost.

❸ **Granularity-aware handbook selection.** We show that the useful granularity of a handbook depends on the target orchestrator, and develop a validation-based selection strategy that balances expressiveness, reliability, and cost.

❹ **Empirical gains and transferability.** Across ten benchmarks, SkillOrchestra improves accuracy, efficiency, and routing stability over strong RL-tuned baselines, mitigates routing collapse, and transfers across orchestrator backbones without retraining.

2 Related Works

Model Routing. Model routing aims to select the most appropriate model from a pool to balance performance and inference cost. Early approaches use heuristic or cascade strategies [Chen et al., 2024a] that escalate queries based on predicted difficulty or budget constraints [Ding et al., 2024, Šakota et al., 2024]. More recent approaches learn query–model matching functions using similarity-based retrieval [Hu et al., 2024, Ong et al., 2025], neural classifiers or ensembles [Jiang et al., 2023b, Lu et al., 2024], and graph-based formulations [Feng et al., 2025, Chen et al., 2024b, Stripelis et al., 2024]. These methods are effective for single-turn model selection, but they typically make one routing decision from input-level features. They therefore do not model how capability requirements evolve across intermediate states, nor how different operational modes may require different model–tool compositions. SkillOrchestra extends routing from query-level model selection to state-conditioned orchestration over modes, skills, agents, and costs.

RL-based Routing and Orchestration. To support multi-step decision making, recent work formulates routing as a sequential decision process and trains an LLM-based router with reinforcement learning [Schulman et al., 2017, Shao et al., 2024]. Systems such as Router-R1 [Zhang et al., 2025a] and ToolOrchestra [Su et al., 2025] that interleave reasoning and routing, optimizing trajectory-level performance–cost rewards. While more expressive than single-shot routers, they are costly to train, brittle to changes in tasks, models, or tools, and can suffer from routing collapse, repeatedly invoking a single agent instead of learning fine-grained specialization. SkillOrchestra instead externalizes orchestration knowledge into a reusable Skill Handbook, learning skill-level competence and cost estimates from execution traces without end-to-end RL finetuning.

3 Preliminaries

Agent Orchestration. We consider an agentic task environment where a user instruction $q \in \mathcal{Q}$ initiates a multi-step reasoning process. At each step, the system must decide both *what kind of operation to perform* and *which agent should perform it*. The system consists of an orchestrator $\mathcal{O}$, a set of operational modes Ψ, a model pool $\mathcal{M}$, and a tool pool $\mathcal{T}$.

Operational modes represent high-level action types, such as `search`, `code`, or `answer`. Each mode $\psi \in \Psi$ restricts the tools that may be used, inducing a mode-specific tool set $\mathcal{T}_\psi \subseteq \mathcal{T}$. An agent is a

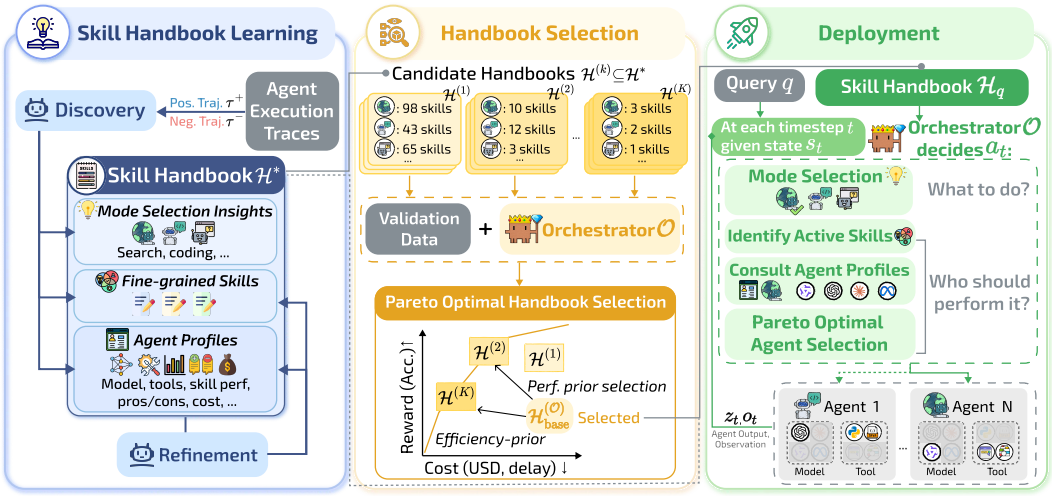


Figure 3: Overview of SkillOrchestra. **(Left)** SkillOrchestra learns a global Skill Handbook from agent traces by discovering reusable skills, refining the skill hierarchy, and estimating each agent's skill-level competence and cost. **(Middle)** It selects an orchestrator-specific subhandbook via Pareto Optimal validation to match the target orchestrator's capability. **(Right)** At deployment, the orchestrator uses the selected handbook for mode-aware, skill-grounded agent routing.

model–tool composition $A = (m, \mathcal{T}_A)$, where $m \in \mathcal{M}$ and $\mathcal{T}_A \subseteq \mathcal{T}$. For a mode ψ, the valid agent set is $\mathcal{A}_\psi = \{(m, \mathcal{T}_A) \mid m \in \mathcal{M},\ \mathcal{T}_A \subseteq \mathcal{T}_\psi\}$. Thus, orchestration consists of selecting a mode ψ_t and a valid agent $A_t \in \mathcal{A}_{\psi_t}$ at each interaction step.

Task Execution Workflow. Given query q, the interaction proceeds over turns $t = 0, \ldots, T$. Let s_t denote the state at turn t, consisting of the original query and the accumulated interaction history. At each turn, the orchestrator selects an action $a_t = (\psi_t, A_t)$, where $\psi_t \in \Psi$ specifies the operation to perform and $A_t \in \mathcal{A}_{\psi_t}$ specifies the agent to execute it. The selected agent produces an execution trace z_t, such as search results or generated code. The environment then returns an observation o_t, such as retrieved evidence or code execution output, which updates the state to s_{t+1}. This process induces a trajectory $\tau = (s_0, a_0, z_0, o_0, \ldots, s_T)$. Figure 3 (right) illustrates this multi-step workflow.

Objective. The orchestrator aims to maximize task success while controlling execution cost. The task objective is: $\max_\pi \mathbb{E}_{\tau\sim\pi}\left[R(\tau) - \lambda \sum_{t=0}^{T} C(A_t, z_t)\right]$, where $R(\tau)$ denotes task reward, $C(A_t, z_t)$ is the cost incurred by agent A_t when producing trace z_t (e.g., token usage or latency), and λ controls the performance–cost tradeoff. We factorize the orchestration policy into mode selection and agent routing: $\pi(a_t \mid s_t) = \pi_{\text{mode}}(\psi_t \mid s_t) \cdot \pi_{\text{route}}(A_t \mid s_t, \psi_t)$. Here, π_{mode} decides the next operational mode, while π_{route} selects an agent conditioned on the current state and mode.

Traditional model routing [Hu et al., 2024, Chen et al., 2024b] can be viewed as a special case of this formulation: there is a single step, a single mode $\Psi = \{\texttt{answer}\}$, and no external tools. The problem then reduces to selecting one model to answer the original query. In contrast, agent orchestration requires repeated state-conditioned decisions across modes, tools, and agents.

From Policy Learning to Skill Acquisition. Prior work typically instantiates the above objective by directly updating the orchestrator parameters θ to learn a routing policy [Su et al., 2025]. While flexible, this ties orchestration knowledge to the trained policy and makes adaptation costly when tasks, models, or tools change. It can also lead to routing collapse: under sparse trajectory-level rewards and cost penalties, the policy may settle on a locally effective strategy that repeatedly invokes a single agent, rather than learning when different agents are best suited for different skills.

`SkillOrchestra` takes a different view: instead of encoding orchestration knowledge only in policy parameters, we learn a reusable **Skill Handbook** $\mathcal{H}$ from execution experience. The handbook stores mode-level execution insights, a hierarchical registry of task-relevant skills, and agent profiles that estimate skill-level competence and cost. The orchestrator then uses $\mathcal{H}$ as external routing knowledge: $\mathcal{H}^* = \text{argmax}_{\mathcal{H}}\, \mathbb{E}_{\tau\sim\pi(\cdot|\mathcal{H})}\left[R(\tau) - \lambda \sum_{t=0}^{T} C(A_t, z_t)\right]$. This reframes orchestration

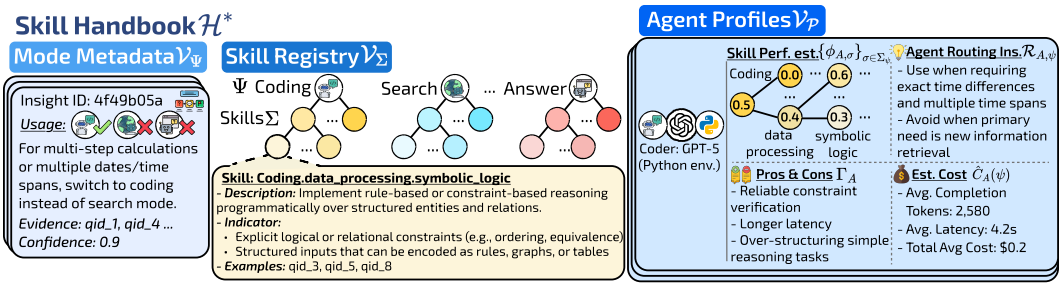


Figure 4: Example Skill Handbook. It decouples task requirements from agent identity through three components: **(left)** mode-level routing insights, **(middle)** a hierarchical registry of reusable skills, and **(right)** agent profiles with skill-specific competence estimates and execution cost statistics.

as skill acquisition: the system learns reusable capability and cost estimates that can transfer across orchestrators and updated model pools, rather than repeatedly training a monolithic routing policy.

4 SkillOrchestra

`SkillOrchestra` is a skill-aware orchestration framework that learns reusable routing knowledge from execution experience. Instead of directly training an end-to-end routing policy, `SkillOrchestra` builds a *Skill Handbook* (Figure 4): an externalized knowledge base that records the skills required by different operational modes, the agents' competence on those skills, and their execution costs. As shown in Figure 3, the framework proceeds in three stages: (i) *Handbook Learning*, which discovers skills and estimates agent profiles from execution traces; (ii) *Handbook Selection*, which chooses an orchestrator-specific subhandbook at an appropriate skill granularity; and (iii) *Deployment*, where the orchestrator uses the selected handbook for mode selection and competence-aware agent routing.

Skills and Agent Profiles. A *skill* σ denotes a recurring capability needed to solve part of a task under an operational mode ψ, such as `search` or `code`. Skills provide an intermediate abstraction between modes and agents: a mode specifies the broad operation to perform, while skills describe the finer-grained capabilities needed to perform it well. For example, under `code`, a task may require symbolic reasoning, numerical computation, data processing, or constraint verification. By reasoning over skills, `SkillOrchestra` decouples task requirements from agent identity and enables capability-aware routing across interaction states.

For each agent A and mode ψ, `SkillOrchestra` maintains an *agent profile* $\mathcal{P}_{A,\psi}$ (Figure 4, right). This profile summarizes the agent's skill-level competence, expected execution cost, and practical routing characteristics under that mode. It includes competence estimates such as $\phi_{A,\sigma}$, the success probability of agent A on skill σ, together with expected cost $\hat{C}_A(\psi)$ and observed usage patterns such as strengths, weaknesses, constraints, or recurring failure modes. During inference, the orchestrator uses these profiles to compare candidate agents for the skills required by the current state. Thus, routing decisions account for both expected competence and cost, rather than relying only on model size or average benchmark performance. Formal notation is summarized in Appendix B.

4.1 Stage I: Skill Handbook Learning

The first stage constructs a global Skill Handbook $\mathcal{H}^*$ from execution traces (Figure 3, left). The goal is to turn raw experience, which agents succeeded or failed on which trajectories, into reusable skill-level insights. The learned handbook (Figure 4) contains three components: (i) mode-level execution insights $\mathcal{V}_\Psi$, (ii) a hierarchical skill registry $\mathcal{V}_\Sigma$, and (iii) agent profiles $\mathcal{V}_\mathcal{P}$. We organize these components as a graph $\mathcal{G}_\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{V}_\Psi \cup \mathcal{V}_\Sigma \cup \mathcal{V}_\mathcal{P}$ and edges $\mathcal{E}$ encode mode–skill associations. This structure allows the orchestrator to retrieve only the skills relevant to the selected mode, rather than searching over the full skill space at runtime.

Skill Discovery. We collect an exploratory dataset $\mathcal{D}_{\text{train}} = \{(q_i, \mathcal{B}_i)\}_{i=1}^N$, where $\mathcal{B}_i = \{\tau_i^{(1)}, \tau_i^{(2)}, \ldots\}$ contains execution traces by varying agent choices at specific modes. For each

query and mode, we contrast successful and failed trajectories to identify the capability gap that explains the failure. An LLM-based discoverer abstracts this gap into a reusable skill σ_{new}, which is added to the skill registry together with its associated mode mapping. For example, if one agent succeeds by executing symbolic constraint checks while another fails with approximate numerical reasoning, the discovered skill may correspond to constraint verification under the `code` mode.

Agent Profile Estimation. Agent profiles are estimated from aggregated outcomes. For each agent A, mode ψ, and skill $\sigma \in \Sigma_\psi$, we model the success probability as $\phi_{A,\sigma} \sim \text{Beta}(\alpha_{A,\sigma}, \beta_{A,\sigma})$. This is natural because each observed agent–skill outcome is binary. Let $s_{A,\sigma} = \sum_{\tau \in \mathcal{B}_i} \mathbb{I}[A \text{ succeeds on } \sigma \text{ in } \tau]$ and $f_{A,\sigma} = \sum_{\tau \in \mathcal{B}_i} \mathbb{I}[A \text{ fails on } \sigma \text{ in } \tau]$. Under the Beta–Bernoulli model, the posterior update is $\alpha_{A,\sigma}^{(t+1)} \leftarrow \alpha_{A,\sigma}^{(t)} + s_{A,\sigma}, \quad \beta_{A,\sigma}^{(t+1)} \leftarrow \beta_{A,\sigma}^{(t)} + f_{A,\sigma}$. A derivation is provided in Appendix E. Mode-level routing signals, such as frequent transitions, systematic failures and recovery insights, are distilled into reusable mode-selection insights $\mathcal{R}_\psi$.

Handbook Refinement. The initial discovery process may produce skills that are redundant, overly broad, or overly fragmented. We therefore refine the skill registry using agent profile statistics.

✦ **Splitting.** A skill σ is considered for splitting when agent performance varies substantially across its associated queries, suggesting that the skill may conflate multiple underlying capabilities. We mark σ as a split candidate when the current skill partition underperforms a reference achievable performance level, such as the best single agent without routing.

✦ **Merging.** A pair of sibling skills (σ_i, σ_j) is considered for merging when their agent performance profiles are indistinguishable for routing. In this work, we mark two sibling skills as merge candidates when their validation performance difference falls within a predefined threshold, set to 0 by default.

An LLM-based reflector reviews the proposed split and merge operations and generates revised skill descriptions. Approved refinements update both the skill registry and the corresponding competence statistics. The resulting handbook $\mathcal{H}^*$ serves as the global knowledge base for the next stage.

4.2 Stage II: Pareto-Optimal Handbook Selection

The useful granularity of a handbook depends on the target orchestrator. Fine-grained skills can improve routing when the orchestrator can reliably identify the active skill, but they can hurt weaker orchestrators by introducing skill-identification errors. Therefore, **`SkillOrchestra`** selects an orchestrator-specific subhandbook whose granularity matches the orchestrator's reasoning capacity and deployment budget (Figure 3, middle).

Given the learned handbook $\mathcal{H}^*$, we construct candidate subhandbooks by retaining different levels of the hierarchical skill registry, together with the corresponding agent profiles and routing metadata. Each candidate $\mathcal{H} \subseteq \mathcal{H}^*$ exposes a different level of skill detail to the orchestrator and induces a routing policy $\pi_{\mathcal{H}}$, which produces a trajectory $\tau_{\mathcal{H}}(q)$ for a query q. We evaluate candidates on a held-out validation set $\mathcal{D}_{\text{val}}$ using two trajectory-level quantities: expected task reward and cumulative execution cost. We first retain the non-dominated candidates under these two objectives, forming an empirical Pareto set $\mathcal{F}^{(\mathcal{O})}$ for the target orchestrator. The final handbook is then selected from this set according to a cost-sensitive deployment preference: $\mathcal{H}_{\text{base}}^{(\mathcal{O})} = \text{argmax}_{\mathcal{H} \in \mathcal{F}^{(\mathcal{O})}} \mathbb{E}_{q \sim \mathcal{D}_{\text{val}}} \left[R(\tau_{\mathcal{H}}(q)) - \lambda \sum_{t=0}^{|\tau_{\mathcal{H}}(q)|} C(\psi_t, A_t) \right]$. Here, $R(\tau_{\mathcal{H}}(q)) \in [0, 1]$ denotes task success, $C(\psi_t, A_t)$ is the execution cost at step t, and λ controls the final performance–cost preference. This avoids committing to a single trade-off point too early, preserving diverse candidates across performance–cost regimes before selecting the deployment-specific operating point. By evaluating candidates through end-to-end trajectories, the selected handbook is tailored to the orchestrator's actual runtime behavior.

4.3 Stage III: Deployment with Skill Handbook

At deployment (Figure 3, right), the orchestrator uses the selected handbook to guide a retrieval–execution cycle. Given a query q, the system retrieves the orchestrator-specific base handbook $\mathcal{H}_{\text{base}}^{(\mathcal{O})}$. Optionally, this base handbook can be augmented with query-relevant skills: $\mathcal{H}_q =$

Table 1: Results on QA datasets. **Bold** = best, underline = second best in each column. SkillOrchestra uses the same orchestrator as baselines. SkillOrchestra+ switches the orchestrator backbone within the same agent pool while reusing the same learned Skill Handbook.

Method	General QA			Multi-Hop QA				Avg.
	NQ	TriviaQA	PopQA	HotpotQA	2wiki	Musique	Bamboogle	
Vanilla	9.2	26.0	12.2	14.0	26.6	2.6	4.0	13.5
No Routing								
SFT	21.2	40.0	16.0	19.8	25.6	5.2	11.2	19.9
RAG	29.8	54.0	36.6	21.6	14.6	7.8	22.4	26.7
CoT [Wei et al., 2022]	12.6	35.8	16.0	16.8	20.8	4.6	22.4	18.4
Search-R1 [Jin et al., 2025]	32.8	51.0	32.4	23.6	27.8	9.0	27.2	29.1
Heuristic & Discriminative Routing								
Largest LLM	29.6	57.8	35.4	27.8	27.4	10.4	48.0	33.8
Prompt LLM	30.0	58.0	34.0	26.8	26.2	10.8	44.8	32.9
Prompt LLM+ (multi turn)	25.8	50.0	25.6	20.6	24.8	7.8	47.2	28.8
KNN Router [Hu et al., 2024]	26.2	52.8	22.2	22.4	19.6	6.6	36.0	26.5
KNN Router+ (multi turn)	23.6	47.8	23.2	15.4	23.4	7.2	38.4	25.6
MLP Router [Hu et al., 2024]	25.2	46.0	22.2	19.8	21.0	7.2	36.0	25.3
BERT Router [Ong et al., 2025]	23.0	51.6	19.2	21.6	20.6	5.8	31.2	24.7
RouterDC [Chen et al., 2024b]	27.8	59.2	28.2	24.4	21.8	8.0	50.4	31.4
GraphRouter [Feng et al., 2025]	27.6	58.6	28.0	23.4	18.0	7.6	44.8	29.7
FrugalGPT [Chen et al., 2024a]	26.5	56.2	36.2	23.4	26.8	10.3	43.0	31.8
RL-based Routing								
Router-R1 [Zhang et al., 2025a]	38.8	70.6	38.4	35.2	43.4	13.8	51.2	41.6
Ours								
`SkillOrchestra`	57.1	77.6	45.6	43.7	48.0	18.2	60.5	50.1
`SkillOrchestra+`	**58.4**	**80.2**	**48.8**	**44.2**	**49.6**	**20.6**	**63.2**	**52.1**

$\mathcal{H}^{(\mathcal{O})}_{\text{base}} \cup \bigcup_{\sigma \in \mathcal{N}_k(q)} (\{\sigma\} \cup \{\mathcal{P}_{A,\psi} \mid A \in \mathcal{A}_\psi\})$, where $\mathcal{N}_k(q)$ denotes the k nearest skills to the query in embedding space. The resulting handbook $\mathcal{H}_q$ is used for the current query.

Mode Selection. At each time step t, the orchestrator first selects the operational mode ψ_t using the current state and mode-level routing insights: $\psi_t \sim \pi_{\text{mode}}(\psi \mid s_t; \mathcal{R}_\psi)$. This determines whether the next step should retrieve information, execute code, or produce an answer.

Skill-grounded Agent Routing. Conditioned on the selected mode ψ_t, the orchestrator is instructed to perform skill-based reasoning before selecting an agent. It analyzes the current state in terms of the relevant capabilities $\Sigma_t \subseteq \Sigma_{\psi_t}$ and then uses the corresponding profile estimates to guide routing. The skill analysis can remain internal to the orchestrator's reasoning and is used to inform the final agent choice, rather than serving as a separate hard classification step. Empirically, this structured reasoning improves over directly asking the orchestrator to choose an agent from the pool; see Appendix I. Agent selection trades off skill-level competence against mode-specific cost. In practice, each $\phi_{A,\sigma}$ is estimated by the posterior mean stored in the agent profile: $A^*_t = \text{argmax}_{A \in \mathcal{A}_{\psi_t}} \left[\sum_{\sigma \in \Sigma_t} w_{t,\sigma} \frac{\alpha_{A,\sigma}}{\alpha_{A,\sigma}+\beta_{A,\sigma}} - \lambda_c \hat{C}_A(\psi_t) \right]$, where $w_{t,\sigma}$ denotes the relevance weight of skill σ at state s_t. This grounds each routing decision in the skills required by the current state, the empirical competence of each candidate agent, and the expected execution cost. The full routing algorithm is provided in Appendix D.

5 Experiments

We conduct extensive experiments to answer:

(RQ1) Effectiveness: Does a learned Skill Handbook improve end-to-end accuracy over heuristic, discriminative, and RL-based methods?

(RQ2) Efficiency: Does skill-based orchestration yield a better performance-cost trade-off?

(RQ3) Routing Behavior: Does skill-based orchestration reduce routing collapse and better match model capacity to task difficulty across modes?

(RQ4) Transferability: Can a Skill Handbook be reused across orchestrators without retraining?

(RQ5) Component Contribution: How do handbook components affect performance and cost?

5.1 SkillOrchestra for Model Routing

We first evaluate SkillOrchestra in the model routing setting [Chen et al., 2024a, Feng et al., 2025, Zhang et al., 2025a], where the system selects among models without external tools or knowledge bases. This setting isolates the quality of model orchestration.

Benchmarks. We consider a diverse suite of knowledge and reasoning-intensive tasks including (1) General QA: Natural Question [Kwiatkowski et al., 2019], TriviaQA [Joshi et al., 2017], PopQA [Mallen et al., 2023]; (2) Multi-hop QA: HotpotQA [Yang et al., 2018], 2WikiMultiHopQA [Ho et al., 2020], Musique [Trivedi et al., 2022], and Bamboogle [Press et al., 2023]; (3) Math Reasoning: MATH [Hendrycks et al., 2021] and AMC23 [MAA, 2023].

Experimental Setup and Baselines. We use Qwen2.5-3B [Qwen, 2024] as the orchestrator and follow the Router-R1 setup for controlled comparison. Experimental details are in Appendix C.2. We compare SkillOrchestra against three categories of methods: **(1) No routing**: methods that do not dynamically consult different models, including supervised finetuning, RAG as in Zhang et al. [2025a], CoT [Wei et al., 2022], and Search-R1 [Jin et al., 2025]; **(2) Heuristic & Discriminative routing**: methods that select models based on input-level signals or learned classifiers, including Largest LLM, Prompt LLM, Prompt LLM+ with multi-turn decomposition, KNN Router [Hu et al., 2024], KNN Router+ with subtask-level decomposition, MLP Router [Hu et al., 2024], BERT Router [Ong et al., 2025], RouterDC [Chen et al., 2024b], GraphRouter [Feng et al., 2025], and FrugalGPT [Chen et al., 2024a]; **(3) RL-based routing**: Router-R1 [Zhang et al., 2025a], a strong PPO-trained [Schulman et al., 2017] multi-turn router trained with 14k samples.

Observation ❶ SkillOrchestra outperforms all routing baselines, including expensive RL-based methods (RQ1). As shown in Table 1, SkillOrchestra achieves the best performance among methods using the same orchestrator. Compared to Router-R1, SkillOrchestra improves average EM from 41.6 to 50.1 (+8.5), while SkillOrchestra+ reaches 52.1 (+10.5) by switching the orchestrator backbone within the same agent pool while reusing the same learned handbook. The gains are especially large on multi-hop tasks, including HotpotQA (35.2 → 43.7 → 44.2) and Bamboogle (51.2 → 60.5 → 63.2). Similar trends hold on math reasoning (Figure 5), where SkillOrchestra improves accuracy by up to 27.5 points over Router-R1 while using substantially less training data.

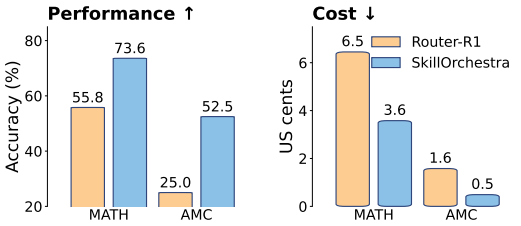


Figure 5: Performance and cost comparison: SkillOrchestra vs. Router-R1. SkillOrchestra achieves up to a 27.5 percentage-point improvement in accuracy while reducing inference cost by $\sim 2.0\times$.

Observation ❷ SkillOrchestra improves the performance–cost frontier (RQ2). Figure 1 (left) shows that SkillOrchestra and SkillOrchestra+ achieve higher accuracy at lower or comparable cost than heuristic, discriminative, and RL-based baselines. SkillOrchestra explicitly accounts for total inference cost, which depends not only on per-token price but also on output length and the number of routing steps. This enables the router to select capable yet cost-effective models rather than consistently escalating to the largest model. For example, Router-R1 attains 41.6 EM at 51.8ć, whereas SkillOrchestra achieves 50.1 EM at 38.4ć; SkillOrchestra+ further improves to 52.1 EM at 41.6ć. On math reasoning, SkillOrchestra similarly improves accuracy while reducing inference cost by about $2\times$ (Figure 5).

Observation ❸ Skill-based routing alleviates routing collapse (RQ3). Figure 6 (left) compares model selection distributions across nine benchmarks. Router-R1 exhibits clear routing collapse, selecting LLaMA-3.1-70B for 98.0% of calls while leaving other models almost unused. In contrast, SkillOrchestra distributes calls across models according to capability and cost: Mixtral-8x22B is selected for 44.5% of calls, Qwen2.5-7B for 26.0%, LLaMA-3.1-70B for 15.4%, and Qwen2.5-3B for 11.5%. This pattern suggests that skill-level competence estimates help the orchestrator reserve stronger models for harder steps while assigning simpler steps to cheaper models. In some cases, the orchestrator can answer directly without escalating to a larger model, further reducing unnecessary calls; an example is shown in Figure 8.

Observation ❹ The learned Skill Handbook transfers across orchestrator backbones without retraining (RQ4). We reuse the Skill Handbook learned with Qwen2.5-3B and apply it directly to

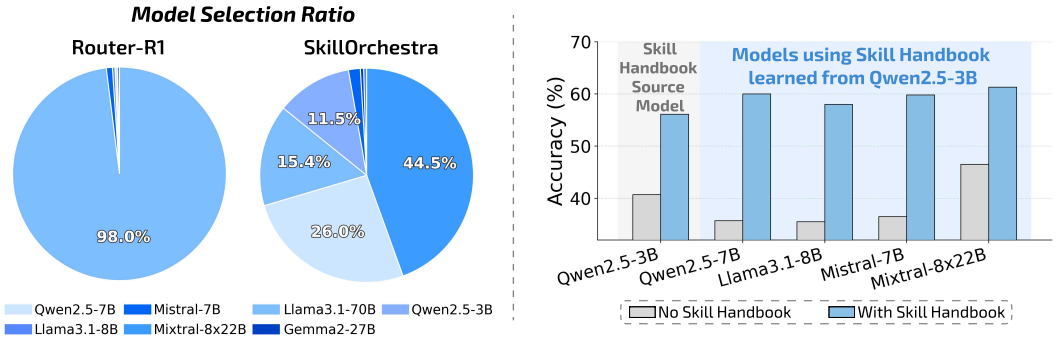


Figure 6: Skill-based orchestration mitigates routing collapse and transfers across orchestrators. **(Left)** Router-R1 collapses to a single model (98% Llama3.1-70B), while SkillOrchestra distributes calls according to skill-level competence and cost. **(Right)** A Skill Handbook learned with Qwen2.5-3B improves multiple orchestrator backbones without retraining.

other orchestrator backbones. Figure 6 (right) shows that the same handbook improves all tested orchestrators on three general QA datasets. Qwen2.5-3B improves from 40.7% to 56.1% (+15.4). The gains remain substantial for other backbones: Qwen2.5-7B improves from 35.7% to 60.0% (+24.3), LLaMA-3.1-8B from 35.5% to 58.0% (+22.5), Mistral-7B from 36.5% to 59.8% (+23.3), and Mixtral-8×22B from 46.5% to 61.3% (+14.8). These results indicate that the handbook captures transferable orchestration knowledge rather than overfitting to a single orchestrator.

5.2 SkillOrchestra for Agent Orchestration

We next evaluate whether SkillOrchestra extends beyond model routing to full agent orchestration, where the system must coordinate multiple operational modes and tools beyond model selection. We follow the ToolOrchestra setup [Su et al., 2025] for controlled comparison.

Experimental Setup and Baselines. We evaluate on FRAMES [Krishna et al., 2024] and consider three operational modes: `search` (web and local search), `code`, and `answer`. Each mode corresponds to a different model pool, detailed in Appendix C.3. The maximum interaction horizon is 50 turns. With Qwen3-8B as the orchestrator, we compare against ToolOrchestra [Su et al., 2025], which trains the orchestrator using GRPO. We also compare against strong proprietary model orchestrators such as GPT-5 [OpenAI, 2025a], Gemini-3-Pro [Google, 2025], or Claude-Opus-4.5 [Anthropic, 2025], while keeping modes, model pools, tools, and execution environments fixed.

Observation ❺ SkillOrchestra achieves better performance-cost tradeoffs in full agent orchestration (RQ1, RQ2). Figure 1 (right) shows that SkillOrchestra remains on the performance–cost frontier in the more complex agent orchestration setting. It achieves the highest accuracy (84.3%) and the lowest total cost ($72.7) among strong learned and proprietary-model baselines. Compared to ToolOrchestra (76.3%, $92.7), SkillOrchestra improves accuracy by +8.0 points while reducing cost by 21.6%. It also outperforms proprietary orchestrators such as GPT-5 (74.6%, $120.4), Claude Opus 4.5 (77.9%, $758.1), and Gemini 3 Pro (78.9%, $1729.3). These results show that strong orchestrator models alone are not sufficient for efficient agent execution; explicit skill modeling enables the system to coordinate specialized agents and tools more effectively.

Table 2: Analysis of SkillOrchestra's Skill Handbook design. HB: Has Handbook, Disc: Skill Discovery, Ref: Handbook Refinement, Sel: Handbook Selection, FG: Fine-grained Skills. Orchestrator: Qwen3-8B.

Setting	HB	Disc	Ref	Sel	FG	Acc %	Cost $
No HB	○	○	○	○	○	71.0	122.9
No Ref + Sel	✔	✔	○	○	✔	79.0	5.5
No Selection	✔	✔	✔	○	✔	79.3	3.4
No FG Skills	✔	✔	✔	✔	○	80.4	15.1
Full System	✔	✔	✔	✔	✔	85.0	9.3

Observation ❻ More skills are not always better; optimal performance-cost tradeoffs require refining and selecting skills to match the orchestrator's capability (RQ1, RQ3, RQ5). Table 2 shows a controlled ablation on 100 randomly sampled FRAMES tasks. Removing the Skill Hand-

book drops accuracy from 85.0% to 71.0% and increases cost from 9.3 to 122.9, confirming that structured skill guidance is important for both effectiveness and efficiency. Using discovered skills without refinement or selection still improves over no handbook (79.0%, cost 5.5), showing that even unrefined skills provide useful routing signals. Refinement reduces cost further (5.5 $\rightarrow$ 3.4) while maintaining similar accuracy, suggesting that merging redundant skills and reorganizing skill boundaries improves efficiency. Removing fine-grained skills hurts both accuracy (85.0% $\rightarrow$ 80.4%) and cost (9.3 $\rightarrow$ 15.1), indicating that appropriately detailed skills are useful when the orchestrator can exploit them. Overall, the best trade-off is achieved when skills are discovered, refined, and selected at a granularity matched to the orchestrator.

6 Conclusion

We introduced SkillOrchestra, a framework for skill-aware orchestration in compound AI systems. Instead of training an end-to-end routing policy, SkillOrchestra learns a reusable Skill Handbook that captures mode-level insights, skill-level agent competence, and execution cost. Across model routing and full agent orchestration settings, SkillOrchestra improves accuracy, reduces cost, mitigates routing collapse, and transfers across orchestrator backbones without retraining. These results suggest that externalizing orchestration knowledge as reusable skill profiles is a scalable alternative to data-intensive RL-based routing as model and tool pools continue to grow.

References

Anthropic. Introducing Claude Opus 4.5, 2025. URL https://www.anthropic.com/news/claude-opus-4-5.

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, 2024a. ISSN 2835-8856. URL https://openreview.net/forum?id=cSimKw5p6R. Featured Certification.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. RouterDC: Query-based router by dual contrastive learning for assembling large language models. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024b. URL https://openreview.net/forum?id=7RQvjayHrM.

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

Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for LLM selections. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=eU39PDsZtT.

Gemini. Gemini deep research, 2024. URL https://gemini.google/overview/deep-research/.

Team Gemma, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. arXiv preprint arXiv:2408.00118, 2024.

Google. A new era of intelligence with Gemini 3, 2025. URL https://blog.google/products-and-platforms/products/gemini/gemini-3/.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, Khaled Saab, Dan Popovici, Jacob Blum, Fan Zhang, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Pushmeet Kohli, Yossi Matias, Andrew Carroll, Kavita Kulkarni, Nenad Tomasev, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R D Costa, José R Penadés, Gary Peltz, Yunhan Xu, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan. Towards an ai co-scientist. 2025. URL https://arxiv.org/abs/2502.18864.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. NeurIPS, 2021.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel, and Chengqing Zong, editors, Proceedings of the 28th International Conference on Computational Linguistics, pages 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

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. arXiv preprint arXiv:2403.12031, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b, 2023a. URL https://arxiv.org/abs/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. Mixtral of experts. arXiv preprint arXiv:2401.04088, 2024.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 14165–14178, Toronto, Canada, July 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.792. URL https://aclanthology.org/2023.acl-long.792/.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In Second Conference on Language Modeling, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan, editors, Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, PeiFeng Wang, silvio savarese, Caiming Xiong, and Shafiq Joty. A survey of frontiers in LLM reasoning: Inference scaling, learning to reason, and agentic systems. Transactions on Machine Learning Research, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=SlsZZ25InC. Survey Certification.

Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation, 2024. URL https://arxiv.org/abs/2409.12941.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. Transactions of the Association for Computational Linguistics, 7:453–466, 2019.

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 Kevin Duh, Helena Gomez, and Steven Bethard, editors, Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 1964–1974, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.109. URL https://aclanthology.org/2024.naacl-long.109/.

MAA. American mathematics competitions 2023 problems (amc23), 2023. URL https://maa.org/student-programs/amc/.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9802–9822, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

OpenAI. Gpt-5 system card, 2025a. URL https://cdn.openai.com/gpt-5-system-card.pdf.

OpenAI. Introducing deep research, 2025b. URL https://openai.com/index/introducing-deep-research/.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, Findings of the Association for Computational Linguistics: EMNLP 2023, pages 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

Qwen. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

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

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024.

Dimitris Stripelis, Zhaozhuo Xu, Zijian Hu, Alay Dilipbhai Shah, Han Jin, Yuhang Yao, Jipeng Zhang, Tong Zhang, Salman Avestimehr, and Chaoyang He. TensorOpera router: A multi-model router for efficient LLM inference. In Franck Dernoncourt, Daniel Preoţiuc-Pietro, and Anastasia Shimorina, editors, Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 452–462, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-industry.34. URL https://aclanthology.org/2024.emnlp-industry.34/.

Hongjin Su, Shizhe Diao, Ximing Lu, Mingjie Liu, Jiacheng Xu, Xin Dong, Yonggan Fu, Peter Belcak, Hanrong Ye, Hongxu Yin, Yi Dong, Evelina Bakhturina, Tao Yu, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Toolorchestra: Elevating intelligence via efficient model and tool orchestration, 2025. URL https://arxiv.org/abs/2511.21689.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. Transactions of the Association for Computational Linguistics, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems, 35:24824–24837, 2022.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Conference on Empirical Methods in Natural Language Processing (EMNLP), 2018.

Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-r1: Teaching llms multi-round routing and aggregation via reinforcement learning. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025a.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. arXiv preprint arXiv:2506.05176, 2025b.

Appendix

A Limitations and Broader Impact **15**

B Notation **15**

C Experimental Details **15**

C.1 Implementation Details 15

C.2 Experimental Details for Model Routing 16

C.3 Experimental Details for Agent Orchestration 16

D Skill-Grounded Agent Routing Algorithm Pseudocode **16**

E Derivation of the Posterior Update **16**

F A Closer Look at Model Selection: SkillOrchestra vs. ToolOrchestra **17**

G Can a Handbook Learned on One Dataset Transfer to Other Tasks? **17**

H Does SkillOrchestra Remain Effective with a Smaller Skill Discoverer and Reflector? **18**

I Does Explicit Skill-Based Routing Improve Over Direct Routing? **18**

J How Robust is SkillOrchestra? **19**

K Demonstrations of Skill-Aware Orchestration **19**

A Limitations and Broader Impact

Limitations. The quality of the learned Skill Handbook depends on the coverage and reliability of the execution traces used to construct it. If the collected trajectories are noisy, the resulting handbook may provide noisy guidance for orchestration. In addition, our method assumes that the LLM-based orchestrator can reason about task requirements and follow the provided orchestration instructions. Although we construct subhandbooks that match the reasoning capability of the orchestrator through validation and collect execution traces using the corresponding orchestrator, the system may still be less effective when the orchestrator has a limited ability to interpret tasks or make use of handbook guidance.

Broader Impact. This work aims to make compound AI systems more capable, efficient, and reliable by improving how agents are selected and orchestrated across multi-step tasks. A positive impact is that skill-aware orchestration can learn fine-grained task requirements and route each step to agents that are better matched to the required capabilities. This can improve overall task performance while also reducing unnecessary computation and monetary cost, rather than repeatedly invoking a single strong or expensive model or agent. As a result, the proposed framework may make advanced AI systems more accessible, cost-effective, and environmentally efficient.

B Notation

Definition B.1 (Skill)**.** *A skill is a reusable capability abstraction that specifies the type of competence required to perform a task under an operational mode ψ. Skills form an intermediate layer between high-level modes (e.g.,* `search`*,* `code`*) and individual agents, enabling the system to decouple capability requirements from agent identity.*

Formally, a skill σ is represented as

$$\sigma \triangleq \langle \mathcal{D}, \mathcal{I} \rangle,$$

where $\mathcal{D}$ is a natural-language description of the capability, and $\mathcal{I}$ denotes contextual indicators, such as keywords, structural patterns, or exemplar queries, that signal when the skill is applicable.

Definition B.2 (Agent Profile)**.** *An agent profile summarizes an agent's mode-conditioned competence, cost, and routing characteristics for skill-aware orchestration. For agent A under operational mode ψ, the profile is defined as*

$$\mathcal{P}_{A,\psi} = \left(\{\phi_{A,\sigma}\}_{\sigma \in \Sigma_\psi}, \hat{C}_A(\psi), \mathcal{R}_{A,\psi}, \Gamma_A \right),$$

where Σ_ψ is the set of skills associated with mode ψ, $\phi_{A,\sigma}$ denotes the estimated success probability of agent A on skill σ, $\hat{C}_A(\psi)$ is the estimated execution cost under mode ψ, $\mathcal{R}_{A,\psi}$ encodes mode-conditioned routing signals such as usage constraints or systematic failure patterns, and Γ_A provides a high-level summary of the agent's strengths and weaknesses.

C Experimental Details

C.1 Implementation Details

For each task, we use 20 samples for learning the Skill Handbook and 20 samples for validation.

To construct the training set, we collect exploratory traces by varying model choices for each question while keeping the task, prompt format, and available tools fixed. In the model-routing setting, each training question is evaluated with every candidate model in the pool, yielding one trajectory per model. In the agent-orchestration setting, we perform mode-conditioned exploration: for a selected mode such as `search` or `code`, we override the model assigned to that mode with each candidate model while keeping the remaining modes fixed to a default model. This produces controlled trajectories that isolate how different models perform under the same task state and operational mode, which are then used to discover skills and estimate agent profiles. This small learning set is substantially smaller than the rollout data required by RL-based orchestrators such as Router-R1 and ToolOrchestra, contributing to the 300–700$\times$ training cost reduction observed in our experimental comparison.

The main results in the paper use GPT-5 as the skill discoverer and reflector, but we also include results using Qwen3-8B as the discoverer and reflector in Appendix H. The results show that SkillOrchestra still preserves its gains over RL-based baselines with a smaller open-source reflector.

C.2 Experimental Details for Model Routing

Implementation Details. We use the same evaluation protocol as Router-R1 for controlled comparison with all routing baselines. We use Qwen2.5-3B [Qwen, 2024] as the orchestrator, and the model pool consists of Qwen2.5-7B [Qwen, 2024], LLaMA-3.1-8B [Grattafiori et al., 2024], LLaMA-3.1-70B [Grattafiori et al., 2024], Mistral-7B [Jiang et al., 2023a], Mixtral-8x22B [Jiang et al., 2024], and Gemma-2-27B [Gemma et al., 2024]. Routing operates in two modes: (1) search mode, where the orchestrator selects a model from the pool to perform subtasks (provide knowledge or solve the subtask); and (2) answer mode, where the orchestrator aggregates intermediate results and produce the final answer. We set the max number of turns to 4. We evaluate the performance using Exact Match (EM) and efficiency using total completion cost. SkillOrchestra is trained in a low-data regime: by default, we select k ($k < 50$) samples from each dataset to train the Skill Handbook and k additional samples for validation and handbook retrieval. We use SkillOrchestra+ to denote the best performance obtained by switching among different orchestrator models within the same agent pool while using the same learned Skill Handbook.

C.3 Experimental Details for Agent Orchestration

Implementation Details. We follow the same evaluation protocol and experimental setup as ToolOrchestra to ensure a controlled and comparable evaluation. We consider three operational modes: For $\psi = \texttt{search}$, the allowable tools are $T_{\texttt{search}} = \{\texttt{WebSearch}, \texttt{LocalSearch}\}$, where WebSearch uses the Tavily API and LocalSearch uses a FAISS index built with Qwen3-Embedding-8B [Zhang et al., 2025b]. The model set is $\mathcal{M}_{\texttt{search}} = \{$GPT-5, GPT-5-mini, Qwen3-32B$\}$. Valid agents are compositions $(m, T_{\texttt{search}})$ with $m \in \mathcal{M}_{\texttt{search}}$. For $\psi = \texttt{code}$, the tool set is $T_{\texttt{code}} = \{\texttt{PythonExec}\}$ operating in a sandbox, and $\mathcal{M}_{\texttt{code}}$ = {GPT-5, GPT-5-mini, Qwen2.5-Coder-32B}. Valid agents are $(m, T_{\texttt{code}})$ with $m \in \mathcal{M}_{\texttt{code}}$. For $\psi = \texttt{answer}$, no external tools are used ($T_{\texttt{answer}} = \varnothing$), and $\mathcal{M}_{\texttt{answer}}$ = {GPT-5, GPT-5-mini, Llama-3.3-70B-Instruct, Qwen3-32B, Qwen2.5-Math-72B, Qwen2.5-Math-7B}. Valid agents are $(m, \varnothing)$ with $m \in \mathcal{M}_{\texttt{answer}}$. The maximum interaction horizon is 50 turns. Final answers are evaluated for accuracy using GPT-5-mini as a judge, and total system cost (USD) is measured.

D Skill-Grounded Agent Routing Algorithm Pseudocode

We present an algorithm block for Skill-grounded Agent Routing in Algorithm 1. A concrete illustration can be found in Figure 3 (Deployment).

E Derivation of the Posterior Update

In Section 4.1, we show that under the Beta–Bernoulli model, the posterior update has a closed form:

$$\alpha_{A,\sigma}^{(t+1)} \leftarrow \alpha_{A,\sigma}^{(t)} + \sum_{\tau \in \mathcal{B}_i} \mathbb{I}[A \text{ succeeds on } \sigma \text{ in } \tau],$$
$$\beta_{A,\sigma}^{(t+1)} \leftarrow \beta_{A,\sigma}^{(t)} + \sum_{\tau \in \mathcal{B}_i} \mathbb{I}[A \text{ fails on } \sigma \text{ in } \tau].$$

This update rule arises from modeling an unknown success probability under binary observations (success/failure). Specifically, for each agent-skill pair (A, σ), we place a Beta prior on the latent success probability and model each observed outcome as

$$y_j \mid \phi_{A,\sigma} \sim \text{Bernoulli}(\phi_{A,\sigma}), \qquad \phi_{A,\sigma} \sim \text{Beta}(\alpha, \beta),$$

where $y_j \in \{0, 1\}$ indicates success/failure. If a batch contains s successes and f failures, then the likelihood is

$$p(\mathcal{D} \mid \phi_{A,\sigma}) \propto \phi_{A,\sigma}^{s}(1 - \phi_{A,\sigma})^{f}.$$

Algorithm 1 Skill-Grounded Agent Routing by Orchestrator $\mathcal{O}$

Input : State s_t; query handbook $\mathcal{H}_q$; cost weight λ_c
Output : Selected mode ψ_t, agent A_t, trace z_t, observation o_t, updated state s_{t+1}

▷ `Mode selection`
Select operational mode $\psi_t \sim \pi_{\text{mode}}(\cdot \mid s_t;\, \mathcal{R}_\psi)$

▷ `Retrieve active skills`
Retrieve active skills $\Sigma_t \subseteq \Sigma_{\psi_t}$ from $\mathcal{H}_q$

▷ `Competence-aware routing`
foreach $A \in \mathcal{A}_{\psi_t}$ **do**
  ▷ `posterior-mean competence from estimated stats in the Handbook`
  $\widehat{P}(A) \leftarrow \sum_{\sigma \in \Sigma_t} w_{t,\sigma} \, \dfrac{\alpha_{A,\sigma}}{\alpha_{A,\sigma} + \beta_{A,\sigma}}$
  ▷ `utility = competence - mode-specific cost`
  $U(A) \leftarrow \widehat{P}(A) - \lambda_c \cdot \widehat{C}_A(\psi_t)$
$A_t \leftarrow \arg\max_{A \in \mathcal{A}_{\psi_t}} U(A)$

▷ `Execute + state transition`
$(z_t, o_t) \leftarrow \text{Execute}(A_t, \psi_t, s_t)$ `// $z_t$ = agent trace, $o_t$ = env observation`
$s_{t+1} \leftarrow \text{UpdateState}(s_t, \psi_t, A_t, z_t, o_t)$

Combining this likelihood with the Beta prior gives Beta posterior:

$$\phi_{A,\sigma} \mid \mathcal{D} \sim \text{Beta}(\alpha + s, \beta + f).$$

Therefore, this directly yields the following update rule:

$$\alpha_{A,\sigma}^{(t+1)} = \alpha_{A,\sigma}^{(t)} + s, \qquad \beta_{A,\sigma}^{(t+1)} = \beta_{A,\sigma}^{(t)} + f,$$

which is the posterior update implied by Beta-Bernoulli conjugacy.

F A Closer Look at Model Selection: SkillOrchestra vs. ToolOrchestra

Skill-grounded routing leads to more efficient tool-model allocation (RQ3). To understand the benefits of SkillOrchestra compared to ToolOrchestra, we also take a closer look at the model selection ratio at each operational mode. We found that the cost reduction of SkillOrchestra comes from smarter allocation of models across different operational modes, rather than simply reducing the number of calls. In search mode, ToolOrchestra routes 99.7% of calls to GPT-5-mini, whereas SkillOrchestra instead uses Qwen3-32B (also the cheapest) for 100% of search calls, identifying it as sufficiently capable and more cost-efficient for the search task. In answer mode, ToolOrchestra similarly exhibits routing collapse, routing 97.9% of calls to GPT-5. SkillOrchestra distributes answer generation more strategically: GPT-5 is used in 58.4% of calls, with the remainder handled by cheaper or specialized models such as GPT-5-mini (10.0%) and Qwen3-32B or math-expert models. This diversification allows the system to reserve expensive models for truly difficult reasoning steps while offloading simpler synthesis or domain-specific subtasks to more efficient models.

G Can a Handbook Learned on One Dataset Transfer to Other Tasks?

A key motivation of SkillOrchestra is that the learned handbook should capture reusable skill-level orchestration knowledge, rather than dataset-specific routing rules. To evaluate this, we train the Skill Handbook only on Natural Questions (NQ) and evaluate it directly on other QA datasets without additional handbook training.

As shown in Table 3, the NQ-trained handbook maintains strong performance on out-of-domain datasets. It improves the average score from 41.6 for Router-R1 to 49.2, a gain of 7.6 points, and even slightly exceeds the in-domain SkillOrchestra variant averaged across these QA tasks. These results suggest that the handbook learns transferable capability abstractions, such as entity lookup, multi-hop reasoning, and answer synthesis, rather than memorizing dataset-specific routing patterns.

Table 3: Cross-dataset transfer of the Skill Handbook. A handbook trained only on NQ transfers to other QA datasets without additional training.

Method	NQ	TriviaQA	PopQA	HpQA	2Wiki	Musique	Bamboogle	Avg.
Router-R1	38.8	70.6	38.4	35.2	43.4	13.8	51.2	41.6
SkillOrchestra (ID)	57.1	**77.6**	45.6	**43.7**	**48.0**	18.2	**60.5**	**50.1**
SkillOrchestra (NQ-only)	57.1	75.2	**48.6**	41.8	46.0	**18.8**	56.8	49.2

Table 4: Skill discovery and refinement with a smaller LLM. Replacing GPT-5 with Qwen3-8B as the discoverer/reflector preserves SkillOrchestra's gains over RL-based baselines.

Method	Discoverer/Reflector	QA							Math		Agent	Avg.
		NQ	TriviaQA	PopQA	HpQA	2Wiki	Musique	Bamb.	MATH	AMC	FRAMES	
Router-R1 / ToolOrchestra	–	38.8	70.6	38.4	35.2	43.4	13.8	51.2	55.8	25.0	76.3	44.9
SkillOrchestra	GPT-5	**57.1**	77.6	45.6	**43.7**	**48.0**	**18.2**	**60.5**	**73.6**	**52.5**	**84.3**	**56.1**
	Qwen3-8B	57.0	**79.6**	**47.6**	37.6	47.8	15.4	59.2	70.6	45.0	83.3	54.3

H Does SkillOrchestra Remain Effective with a Smaller Skill Discoverer and Reflector?

SkillOrchestra uses an LLM-based discoverer and reflector to propose skill definitions and split/merge refinements. While our main experiments use GPT-5 for this step, the pipeline does not require the reflector to be a proprietary frontier model. To evaluate whether a smaller LLM can support skill discovery and refinement, we replace GPT-5 with Qwen3-8B while keeping the rest of the pipeline unchanged.

Table 4 shows that a smaller open-source LLM can still support effective skill discovery and refinement. Using Qwen3-8B as the discoverer/reflector achieves an average score of 54.3, substantially above the RL-based baselines and close to the GPT-5-based discoverer/reflector. This suggests that SkillOrchestra's gains come primarily from the execution-grounded skill-learning pipeline, rather than from relying on a specific frontier model to construct the handbook.

This robustness comes from the role of the LLM in our pipeline. The LLM proposes candidate skill abstractions and refinements, but the final handbook is constrained by empirical evidence. Skill definitions are induced from observed success–failure contrasts; refinement operations are guided by performance statistics, such as indistinguishable behavior among sibling skills or poor downstream performance under the current skill partition; and handbook selection validates candidate subhandbooks using downstream performance and cost. We further enforce non-decreasing validation performance when selecting the final subhandbook. Thus, even when a smaller LLM is used for discovery and refinement, the deployed handbook is chosen according to measured orchestration utility.

I Does Explicit Skill-Based Routing Improve Over Direct Routing?

SkillOrchestra provides the orchestrator with a learned Skill Handbook. A natural question is whether the handbook is most useful as additional context, or whether the orchestrator benefits from being instructed to reason over skills before routing. To test this, we compare two routing strategies using the same learned Skill Handbook. In *skill-based routing*, the orchestrator is prompted to analyze the current state in terms of relevant skills before selecting an agent. This does not require matching an exact ground-truth skill label; rather, it encourages the orchestrator to explicitly reason about the capability requirements of the current step and then use the corresponding competence and cost estimates. In *direct routing*, the orchestrator receives the same handbook context but is asked to select an agent directly, without the intermediate skill-analysis step.

As shown in Table 5, directly providing the Skill Handbook already improves over Router-R1, increasing average EM from 41.6 to 44.0. However, instructing the orchestrator to perform skill-based reasoning further improves the average score to 50.1, a gain of 6.1 points over direct routing and

Table 5: Effect of explicit skill-based routing. Both SkillOrchestra variants use the same learned Skill Handbook; they differ only in whether the orchestrator performs an explicit skill-analysis step before routing.

Method	Routing	NQ	TriviaQA	PopQA	HpQA	2Wiki	Musique	Bamboogle	Avg.
Router-R1	RL-based	38.8	70.6	38.4	35.2	43.4	13.8	51.2	41.6
SkillOrchestra	Direct	45.2	71.0	39.6	38.2	45.6	14.2	54.4	44.0
	Skill-based	**57.1**	**77.6**	**45.6**	**43.7**	**48.0**	**18.2**	**60.5**	**50.1**

Table 6: Robustness of SkillOrchestra across repeated runs.

Method	Statistic	NQ	TriviaQA	PopQA	HpQA	2Wiki	Musique	Bamboogle
SkillOrchestra	Mean	57.1	77.6	45.6	43.7	48.0	18.2	60.5
	Std.	1.6	3.4	3.0	2.7	2.0	0.7	3.0

8.5 points over Router-R1. The gains are consistent across datasets, with particularly large improvements over direct routing on NQ (+11.9), TriviaQA (+6.6), PopQA (+6.0), HotpotQA (+5.5), and Bamboogle (+6.1). These results indicate that the handbook is most effective when used as structured routing knowledge rather than generic prompt context: the skill-analysis step helps the orchestrator articulate the current capability requirements before comparing agents by their estimated competence and cost.

J How Robust is SkillOrchestra?

We evaluate the robustness of SkillOrchestra across three repeated runs. Since multi-step orchestration can produce different intermediate trajectories and agent choices, we report the mean and standard deviation.

As shown in Table 6, SkillOrchestra is stable across three runs. The standard deviations range from 0.7 to 3.4 points, which are small relative to the gains over Router-R1. This suggests that the improvements are consistent across runs rather than being driven by run-specific variance.

K Demonstrations of Skill-Aware Orchestration

We provide full execution traces of the skill-based router in Figures 7–9, along with the orchestration instruction template in Figure 10. The instruction integrates the task query, execution context, and the selected Skill Handbook used for routing decisions.

AMC Example: Skill-Based Router Corrects via Multi-turn Routing

✓ Skill Handbook | 2 external model calls (<search>) | ✓ Correct

Skill Router Instruction

```
You are a skill-based model router. You are selecting the best model to answer a question by
     analyzing a question to identify required skills and their importance related to this
     question.

## Learned Skill Definitions (from validation)

### Algebra and Functions
Symbolic manipulation and equation-solving across rational expressions, logarithms/exponents,
     polynomials via Vieta, and trigonometric identities and parameters.
  - Rational/linear manipulation (nested fractions, clearing denominators, simple systems):
     Evaluate nested or continued fractions, reduce to irreducible form, and set up/solve linear
     relations from worded constraints.
    Examples: Problem 1: Compute 3 + 1/(3 + 1/(3 + 1/3)) as an irreducible fraction and return m+n
     ., Problem 12: Three numbers sum to 96 with linear relations; find |first - second|.
```

```
  - Logarithm/exponent identities and metric constraints: Use log laws to simplify expressions and
      translate distance conditions on the number line into equations in log variables.
    Examples: Problem 3: Distance between log_6 x and log_6 9 equals twice another distance; find
      product of solutions., Problem 6: Evaluate (log 5)^3 + (log 20)^3 + (log 8)(log 0.25).
  - Vieta and symmetric sums with parameter shifts: Extract sums/products of roots from polynomial
      coefficients and evaluate expressions after shifting variables (e.g., edges increased by a
      constant).
    Examples: Problem 7: Roots are box dimensions; edges lengthened by 2; compute new volume via
      Vieta.
  - Trigonometric identity reduction and parameter interval analysis: Rewrite trig sums/products (e
      .g., sin x + sin 2x) and determine parameter ranges guaranteeing multiple solutions or
      specific solution behaviors.
    Examples: Problem 9: For a*(sin x + sin 2x) = sin 3x, find all a producing more than one
      solution and compute p+q+r.

### Geometry and Transformations
Spatial reasoning in 2D/3D using vectors, coordinates, isometries, and complex-plane
     interpretations of loci and areas.
  - 3D vector/coordinate geometry and dot products: Model regular solids with coordinates, use
      midpoints and vectors, and compute angles via dot products and norms.
    Examples: Problem 4: In regular tetrahedron ABCD with M midpoint of AB, find $\cos(\angle CMD)$
       as p/q.
  - Complex-plane geometry: loci, polygons, and area optimization: Interpret complex constraints as
       geometric loci (segments, disks), compute Minkowski sums (stadium/rounded rectangle areas),
      and analyze polygons formed by z and 1/z under quadratic relations.
    Examples: Problem 5: Region from sum of a segment [3, 4i] and a unit disk; find closest integer
       to area., Problem 14: For z^2 - cz + 10 = 0, quadrilateral with z1, z2, 1/z1, 1/z2 has
      maximal area; find $c=\sqrt{m}$.
  - Coordinate geometry with distance constraints in special quadrilaterals: Place figures in
      coordinates, encode equal-leg/parallel-side constraints, and solve using distance equations
      to find side ratios.
    Examples: Problem 13: Isosceles trapezoid with PA=1, PB=2, PC=3, PD=4; find BC/AD.
  - Composition of plane isometries and periodicity: Represent rotations and reflections (matrices
      or angle-line representations), compose varying-parameter isometries, and determine the least
       n returning a point to itself.
    Examples: Problem 10: Find minimal n so $T_1\circ T_2\circ \cdots \circ T_n$ sends (1,0) back
      to itself.

### Number Theory and Diophantine Analysis
Integer-structure problems involving Pell equations, valuations, and divisibility properties of
     rational sums.
  - Pell-type equations for figurate-number squares: Convert conditions like triangular numbers
      being squares into Pell equations, use fundamental solutions/recurrences to generate the next
       solutions.
    Examples: Problem 8: Find the fourth triangular number that is also a square; sum its digits.
  - Harmonic denominators vs LCM via p-adic valuations: Analyze reduced denominators of harmonic
      numbers, compare to LCM(1..n), and count n where strict inequality holds using prime-power
      valuations and cancellation.
    Examples: Problem 15: For 1 <= n <= 22, count n with k_n < L_n.

### Combinatorics and Discrete Structures
Counting and structural reasoning for pairings and permutations under process constraints.
  - Constrained pairing/matching counts: Model pairings with inequality or dominance constraints,
      assess feasibility (often via greedy/ordering), and count valid matchings.
    Examples: Problem 2: Number of ways to pair 1..14 so larger >= 2x smaller in each pair.
  - Permutation process modeling (passes/runs) and counting: Translate left-to-right multi-pass
      selection procedures into properties like increasing runs or pile counts and enumerate
      permutations achieving a given number of passes.
    Examples: Problem 11: Count orderings of 13 cards that are picked up in exactly two passes.

## Model Performance (learned from validation)

### LLaMA-3.1-70B-Instruct
Overall: 15% success (3/20)
Skill scores:
  - Pell-type equations for figurate-number squares: 100%
  - Harmonic denominators vs LCM via p-adic valuations: 50%
  - Complex-plane geometry: loci, polygons, and area optimization: 33%
  - Rational/linear manipulation (nested fractions, clearing denominators, simple systems): 0%
  - Constrained pairing/matching counts: 0%
Strengths: geometry.complex_plane_loci_area
Weaknesses: geometry.3d_dot_product, algebra.rational_linear_manipulation, math-heavy reasoning in
     this sample

### Gemma-2-27B-Instruct
Overall: 10% success (2/20)
Skill scores:
  - Harmonic denominators vs LCM via p-adic valuations: 50%
  - Rational/linear manipulation (nested fractions, clearing denominators, simple systems): 25%
```

```
  - Constrained pairing/matching counts: 0%
  - Logarithm/exponent identities and metric constraints: 0%
  - 3D vector/coordinate geometry and dot products: 0%
Strengths: algebra.rational_linear_manipulation, basic_algebra_word_problems
Weaknesses: algebra.logs_and_exponents, quantitative reasoning in this sample, geometry.
     complex_plane_loci_area

### Qwen2.5-7B-Instruct
Overall: 25% success (5/20)
Skill scores:
  - 3D vector/coordinate geometry and dot products: 100%
  - Harmonic denominators vs LCM via p-adic valuations: 100%
  - Complex-plane geometry: loci, polygons, and area optimization: 33%
  - Rational/linear manipulation (nested fractions, clearing denominators, simple systems): 25%
  - Constrained pairing/matching counts: 0%
Strengths: algebra.rational_linear_manipulation, geometry.3d_dot_product (observed), general_backup
Weaknesses: algebra.logs_and_exponents, combinatorics.constrained_pairing_matching, algebra.
     rational_linear_manipulation

### Mistral-7B-Instruct
Overall: 25% success (5/20)
Skill scores:
  - Harmonic denominators vs LCM via p-adic valuations: 100%
  - Logarithm/exponent identities and metric constraints: 50%
  - Coordinate geometry with distance constraints in special quadrilaterals: 50%
  - Rational/linear manipulation (nested fractions, clearing denominators, simple systems): 25%
  - Constrained pairing/matching counts: 0%
Strengths: algebra.logs_and_exponents (partial), geometry.coordinate_distance_quadrilaterals,
     algebra.rational_linear_manipulation
Weaknesses: geometry.3d_dot_product, algebra.rational_linear_manipulation, combinatorics.
     permutation_passes_runs

### Mixtral-8x22B-Instruct
Overall: 20% success (4/20)
Skill scores:
  - Pell-type equations for figurate-number squares: 100%
  - Rational/linear manipulation (nested fractions, clearing denominators, simple systems): 50%
  - Logarithm/exponent identities and metric constraints: 50%
  - Constrained pairing/matching counts: 0%
  - 3D vector/coordinate geometry and dot products: 0%
Strengths: algebra.logs_and_exponents, algebra.rational_linear_manipulation, general algebraic
     manipulation
Weaknesses: combinatorics.constrained_pairing_matching (no clear advantage), general, geometry.
     coordinate_distance_quadrilaterals

### LLaMA-3.1-8B-Instruct
Overall: 20% success (4/20)
Skill scores:
  - Vieta and symmetric sums with parameter shifts: 100%
  - Logarithm/exponent identities and metric constraints: 50%
  - Harmonic denominators vs LCM via p-adic valuations: 50%
  - Rational/linear manipulation (nested fractions, clearing denominators, simple systems): 25%
  - Constrained pairing/matching counts: 0%
Strengths: algebra.logs_and_exponents (partial), algebra.rational_linear_manipulation
Weaknesses: algebra.rational_linear_manipulation, geometry.coordinate_distance_quadrilaterals,
     combinatorics.permutation_passes_runs

## Cost Tiers (cheapest to most expensive)
- Cheap: Qwen2.5-7B-Instruct, LLaMA-3.1-8B-Instruct, Mistral-7B-Instruct
- Medium: Gemma-2-27B-Instruct
- Expensive: LLaMA-3.1-70B-Instruct, Mixtral-8x22B-Instruct

## Task
1. First, analyze the question below and identify which skills are needed, along with the
     percentage/weight of each skill (how important each skill is for answering this question).

Output your analysis in the following format inside <skill_analysis> tags:
<skill_analysis>
{
  "required_skills": [
    {"skill_id": "skill.id", "percentage": 50},
    {"skill_id": "skill.id", "percentage": 30},
    ...
  ],
  "reasoning": "Brief explanation of why these skills are needed"
}
</skill_analysis>
```

```
The percentages should sum to approximately 100 (they don't need to be exact, but should reflect
     relative importance).

2. After providing the skill analysis, reflect on which model is best suited based on the skills
     required and model performance data above.

3. Route to that model using <search> tags and provide final answer in <answer>...</answer>

Every time you receive new information, you must first conduct reasoning inside <think> ... </think
     >. After reasoning, if you find you lack some knowledge, you can call a specialized LLM by
     writing a query inside <search> LLM-Name:Your-Query </search>.
!!! STRICT FORMAT RULES for <search>: !!!
    + You MUST replace LLM-Name with the EXACT name of a model selected from
      [Qwen2.5-7B-Instruct, LLaMA-3.1-8B-Instruct, LLaMA-3.1-70B-Instruct,
       Mistral-7B-Instruct, Mixtral-8x22B-Instruct, Gemma-2-27B-Instruct].
    + You MUST replace Your-Query with the EXACT same question as the original question below (DO
      NOT CHANGE IT).
    + NEVER copy or paste model descriptions into <search>.
    + NEVER output the placeholder format <search> LLM-Name:Your-Query </search>. Always replace
      both parts correctly.

Before each LLM call, you MUST explicitly reason inside <think> ... </think> about:
    + Why external information is needed.
    + Which skills from the catalog are required for this question.
    + Which model is best suited based on the model performance data above.

When you call an LLM, the response will be returned between <information> and </information>.
You are encouraged to explore and utilize different LLMs to better understand their respective
     strengths and weaknesses.

If you find that no further external knowledge is needed, you can directly provide your final
     answer to the original question inside <answer> ... </answer>, without additional explanation
      or illustration.

If you think none of the models listed have the necessary skills to answer this question directly,
     you can route to the model with the highest overall pass rate of models in the pool to get
     more information.

Question: How many complex numbers satisfy the equation $z^5=\overline{z}$, where $\overline{z}$ is
      the conjugate of the complex number $z$?
```

Turn 1 -- skill analysis + first external call

```
<skill_analysis>
{
  "required_skills": [
    {"skill_id": "algebra.logs_and_exponents", "percentage": 70},
    {"skill_id": "algebra.rational_linear_manipulation", "percentage": 30}
  ],
  "reasoning": "Use polar form; equate magnitudes and angles; solve exponent/angle constraints."
}
</skill_analysis>
<think>
To solve this problem, we need to manipulate the given equation involving complex numbers and their
      conjugates. This requires skills in algebra, specifically dealing with logarithms and
     exponents, as well as rational/linear manipulation.
</think>
<search>
Mixtral-8x22B-Instruct: How many complex numbers satisfy the equation $z^5=\overline{z}$, where $\
     overline{z}$ is the conjugate of the complex number $z$?
</search>

<information>
The equation $z^5=\\overline{z}$ implies that $|z^5|=|\\overline{z}|$, which simplifies to $|z|^5=|
     z|$.
This implies that $|z|=0$ or $|z|=1$.
- If $|z|=0$, then $z=0$.
- If $|z|=1$, then $z=e^{i\\theta}$ for some $\\theta\\in\\mathbb{R}$.
Substituting this into the original equation, we get $e^{5i\\theta}=e^{-i\\theta}$.
This implies that $5\\theta=-\\theta+2k\\pi$ for some $k\\in\\mathbb{Z}$.
Solving for $\\theta$, we get $\\theta=\\frac{k\\pi}{3}$ for some $k\\in\\mathbb{Z}$.
Therefore, there are infinitely many complex numbers that satisfy the equation $z^5=\\overline{z}$.
      These are $z=0$ and $z=e^{i\\frac{k\\pi}{3}}$ for any $k\\in\\mathbb{Z}$.
</information>
```

Turn 2 -- second call to resolve counting

```
<skill_analysis>
{
```

```
  "required_skills": [
    {"skill_id": "algebra.logs_and_exponents", "percentage": 60},
    {"skill_id": "algebra.rational_linear_manipulation", "percentage": 40}
  ],
  "reasoning": "Need correct restriction on k from \theta in [0,2\pi)."
}
</skill_analysis>
<think>
We must count distinct solutions modulo 2\pi; query a higher-capability model to confirm.
</think>
<search>
LLaMA-3.1-70B-Instruct: How many complex numbers satisfy the equation $z^5=\overline{z}$, where $\
    overline{z}$ is the conjugate of the complex number $z$?
</search>

<information>
[...]
Substituting $z = re^{i\theta}$ into the equation $z^5=\overline{z}$, we get
$(re^{i\theta})^5 = re^{-i\theta}$, i.e., $r^5e^{5i\theta} = re^{-i\theta}$.
Equating magnitudes gives $r^5 = r$, so $r = 0$ or $r = 1$.
Equating arguments gives $5\theta = -\theta + 2k\pi$, so $6\theta = 2k\pi$ and
$\theta = \frac{k\pi}{3}$.
Since $\theta \in [0,2\pi)$, we have $0 \le \frac{k\pi}{3} < 2\pi$, so $k=0,1,2,3,4,5$.
For $r = 0$, there is exactly one solution ($z = 0$).
For $r = 1$, there are 6 possible values of $\\theta$ corresponding to the 6 possible values of $k$
    . Therefore, the total number of complex numbers that satisfy the equation is 1 (for $r = 0$)
     + 6 (for $r = 1$) = 7.
</information>
```

Turn 3 -- final answer

```
<answer>7</answer>
```

Figure 7: Full trace of an AMC example. The router first analyzes the required skills, calls Mixtral-8x22B-Instruct, which derives the correct form but miscounts the solutions; it then routes to LLaMA-3.1-70B-Instruct to correctly restrict $\theta \in [0, 2\pi)$ and count distinct solutions, producing the correct final answer.

AMC Example: Skill-Based Router Recognizes No External Model Is Needed and Answers Autonomously

✓ Skill Handbook | 0 external model calls | ✓ Correct

Skill Router Instruction

```
You are a skill-based model router. You are selecting the best model to answer a question by
    analyzing a question to identify required skills and their importance related to this
    question.

## Learned Skill Definitions (from validation)

### Algebraic Modeling and Equation Solving
Setting up and solving algebraic relationships, including polynomial constraints, radical systems,
    and vector-rate models.
  - Polynomial interpolation from structural constraints: Use given values and leading coefficients
     to determine polynomial constants or invariant sums (e.g., P(0)+Q(0)) without full
    reconstruction.
    Examples: Problem 1: Quadratic polynomials P and Q with opposite leading coefficients both pass
     through two points; find P(0)+Q(0).
  - Linearization of symmetric radical systems: Transform symmetric systems with square roots via
    substitutions, factoring, and controlled squaring to solve for variables or symmetric
    expressions.
    Examples: Problem 7: Solve a 3-equation symmetric radical system and compute [(1-x)(1-y)(1-z)
    ]^2.
  - Vector kinematics in flowing media: Model motion with currents/winds by decomposing velocities
    into components relative to ground and medium; impose geometric/temporal constraints to solve
     for unknowns.
    Examples: Problem 11: Two swimmers in a flowing river head to a common point on the opposite
    bank; determine the downstream offset D.

### Euclidean Geometry and Transformations
Reasoning about plane and spatial geometry using angle bisectors, tangencies, homothety/inversion,
    and 3D-to-2D section relations.
```

```
  - Circle tangency via homothety/inversion: Exploit homothety or inversion to relate radii,
      centers, and intersection loci of tangent circles and inscribed figures.
    Examples: Problem 14: Three equal tangent circles inside a circumcircle around an equilateral
      triangle; find side of the inner equilateral from circle intersections.
  - Angle bisectors and (ex)center configurations: Use properties of angle bisectors, incenter/
      excenters, and their intersections to compute distances in polygons, especially trapezoids
      and triangles.
    Examples: Problem 9: In an isosceles trapezoid, angle bisectors meet at P and Q; find PQ given
      side lengths.
  - Sphere-plane sections and power-of-a-point relations: Relate congruent circular cross-sections
      of spheres to sphere radii and plane offsets; compute distances between projected centers
      using tangency and power.
    Examples: Problem 2: Three tangent spheres cut by a plane in congruent circles with centers A,B
      ,C; given AB^2, find AC^2.
  - Equal-perimeter splitting lines via reflection: Model equal-perimeter partitions by reflecting
      across sides to convert perimeter conditions to straight-line distance constraints.
    Examples: Problem 6: Splitting lines through midpoints M and N in a triangle intersect; deduce
      side lengths/structure from equal-perimeter property.

### Combinatorics and Discrete Structures
Counting and optimization in discrete settings with indicator methods, parity constraints, pattern
      avoidance, and digit-product arrangements.
  - Indicator variables and linearity over subsets: Compute sums over families of sets by summing
      elementwise contributions with indicator variables and symmetry constraints like |A|=|B|.
    Examples: Problem 4: Evaluate $S_n = \sum |A \cap B|$ over ordered pairs of subsets with equal
      cardinalities.
  - AP-free (pattern-avoidance) sequence design: Select or count integers under monotonicity while
      forbidding k-term arithmetic progressions; use modular classes and structural constraints.
    Examples: Problem 12: Count integer pairs (a,b) so the 8-term increasing sequence contains no
      4-term arithmetic progression.
  - Parity-constrained arrangements with identical pairs: Count permutations where identical items
      must be separated by an even/odd number of positions; apply parity classes and inclusion-
      exclusion or structural bijections.
    Examples: Problem 15: Probability a random arrangement of two each of six colors is 'even' (
      parity gap constraint).
  - Digit-based discrete optimization of product ratios: Assign distinct digits to maximize/
      minimize product differences/ratios under constraints using inequalities, rearrangement, and
      greedy swaps.
    Examples: Problem 13: Minimize a ratio of product difference over product using digits 1-9
      exactly once; report m+n.

### Number Theory and Modular Reasoning
Reasoning with modular arithmetic, base representations, reduced fractions of repeating decimals,
      and roots-of-unity congruences.
  - Repeating decimal reduction via gcd/phi structure: Convert repeating decimals to fractions with
       denominators of the form 9, 99, 9999..., reduce by gcd, and count distinct reduced
      numerators leveraging multiplicative structure.
    Examples: Problem 5: Count distinct numerators obtained from all 4-digit repeating decimals
      when reduced to lowest terms.
  - Base conversion as digit-constrained Diophantine equations: Translate cross-base digit
      identities into linear equations with digit bounds; solve for digits satisfying both base
      systems.
    Examples: Problem 8: Find three-digit base-10 abc whose base-9 representation is bca_9.
- Roots-of-unity angle congruences and counting: Interpret complex constants as $e^{i\theta}$;
      equate powers via modular congruences on arguments and count solutions within given bounds.
  Examples: Problem 10: With $w = \frac{\sqrt{3}+i}{2}$ and $z = \frac{-1+i\sqrt{3}}{2}$, count $(r
      ,s) \le 100$ satisfying $i w^r = z^s$.

## Model Performance (learned from validation)

### LLaMA-3.1-70B-Instruct
Overall: 30% success (6/20)
Skill scores:
  - Sphere-plane sections and power-of-a-point relations: 100%
  - Base conversion as digit-constrained Diophantine equations: 100%
  - Parity-constrained arrangements with identical pairs: 100%
  - Circle tangency via homothety/inversion: 50%
  - Indicator variables and linearity over subsets: 50%
Strengths: 3D geometry with spheres/planes and Power of a Point, Number base representation and
      digit-constraint puzzles, Best overall reliability in this batch
Weaknesses: Circle tangency/homothety/inversion geometry, Algebra with polynomial interpolation/
      constraint reasoning, Subset-sum/inclusion-exclusion style combinatorics

### Mistral-7B-Instruct
Overall: 5% success (1/20)
Skill scores:
  - Indicator variables and linearity over subsets: 50%
  - Polynomial interpolation from structural constraints: 0%
  - Sphere-plane sections and power-of-a-point relations: 0%
```

```
  - Circle tangency via homothety/inversion: 0%
  - Angle bisectors and (ex)center configurations: 0%
Strengths: Occasional general reasoning success (inconsistent)
Weaknesses: Euclidean geometry (especially tangency/homothety/inversion), 3D geometry with spheres/
     planes, Polynomial interpolation constraints

### Gemma-2-27B-Instruct
Overall: 0% success (0/20)
Skill scores:
  - Polynomial interpolation from structural constraints: 0%
  - Sphere-plane sections and power-of-a-point relations: 0%
  - Circle tangency via homothety/inversion: 0%
  - Angle bisectors and (ex)center configurations: 0%
  - Indicator variables and linearity over subsets: 0%
Weaknesses: 3D sphere-plane geometry, Circle tangency/homothety/inversion, Polynomial interpolation
      constraints

### Mixtral-8x22B-Instruct
Overall: 0% success (0/20)
Skill scores:
  - Polynomial interpolation from structural constraints: 0%
  - Sphere-plane sections and power-of-a-point relations: 0%
  - Circle tangency via homothety/inversion: 0%
  - Angle bisectors and (ex)center configurations: 0%
  - Indicator variables and linearity over subsets: 0%
Weaknesses: 3D sphere-plane geometry, Circle tangency/homothety/inversion, Polynomial interpolation
      constraints

### Qwen2.5-7B-Instruct
Overall: 0% success (0/20)
Skill scores:
  - Polynomial interpolation from structural constraints: 0%
  - Sphere-plane sections and power-of-a-point relations: 0%
  - Circle tangency via homothety/inversion: 0%
  - Angle bisectors and (ex)center configurations: 0%
  - Indicator variables and linearity over subsets: 0%
Weaknesses: Geometry-heavy tasks, Algebraic modeling/constraints, 0% on tested tasks

### LLaMA-3.1-8B-Instruct
Overall: 0% success (0/20)
Skill scores:
  - Polynomial interpolation from structural constraints: 0%
  - Sphere-plane sections and power-of-a-point relations: 0%
  - Circle tangency via homothety/inversion: 0%
  - Angle bisectors and (ex)center configurations: 0%
  - Indicator variables and linearity over subsets: 0%
Weaknesses: All observed olympiad-style math skills in this batch, All tested math skills in this
     batch (0% accuracy)

## Cost Tiers (cheapest to most expensive)
- Cheap: Qwen2.5-7B-Instruct, LLaMA-3.1-8B-Instruct, Mistral-7B-Instruct
- Medium: Gemma-2-27B-Instruct
- Expensive: LLaMA-3.1-70B-Instruct, Mixtral-8x22B-Instruct

## Task
1. First, analyze the question below and identify which skills are needed, along with the
     percentage/weight of each skill (how important each skill is for answering this question).

Output your analysis in the following format inside <skill_analysis> tags:
<skill_analysis>
{
  "required_skills": [
    {"skill_id": "skill.id", "percentage": 50},
    {"skill_id": "skill.id", "percentage": 30},
    ...
  ],
  "reasoning": "Brief explanation of why these skills are needed"
}
</skill_analysis>

The percentages should sum to approximately 100 (they don't need to be exact, but should reflect
     relative importance).

2. After providing the skill analysis, reflect on which model is best suited based on the skills
     required and model performance data above.

3. Route to that model using <search> tags and provide final answer in <answer>...</answer>
```

```
Every time you receive new information, you must first conduct reasoning inside <think> ... </think
     >. After reasoning, if you find you lack some knowledge, you can call a specialized LLM by
     writing a query inside <search> LLM-Name:Your-Query </search>.
!!! STRICT FORMAT RULES for <search>: !!!
    + You MUST replace LLM-Name with the EXACT name of a model selected from [Qwen2.5-7B-Instruct,
     LLaMA-3.1-8B-Instruct, LLaMA-3.1-70B-Instruct, Mistral-7B-Instruct, Mixtral-8x22B-Instruct,
     Gemma-2-27B-Instruct].
    + You MUST replace Your-Query with the EXACT same question as the original question below (DO
     NOT CHANGE IT).
    + NEVER copy or paste model descriptions into <search>.
    + NEVER output the placeholder format <search> LLM-Name:Your-Query </search>. Always replace
     both parts correctly.
Before each LLM call, you MUST explicitly reason inside <think> ... </think> about:
    + Why external information is needed.
    + Which skills from the catalog are required for this question.
    + Which model is best suited based on the model performance data above.
When you call an LLM, the response will be returned between <information> and </information>. You
     are encouraged to explore and utilize different LLMs to better understand their respective
     strengths and weaknesses.
If you find that no further external knowledge is needed, you can directly provide your final
     answer to the original question inside <answer> ... </answer>, without additional explanation
      or illustration. For example: <answer> Beijing </answer>.
+ Important: You must not output the placeholder text "<answer> and </answer>" alone.
+ You must insert your actual answer between <answer> and </answer>, following the correct format.
+ You must not output the model name or query between <answer> and </answer>.
If you think none of the models listed have the necessary skills to answer this question directly,
     you can route to the model with the highest overall pass rate of models in the pool to get
     more information.

Question: A $3-4-5$ right triangle is inscribed in circle $A$, and a $5-12-13$ right triangle is
     inscribed in circle $B$. Find the ratio of the area of circle $A$ to the area of circle $B$.
     The final answer can be written in the form $\frac{m}{n}$, where $m$ and $n$ are relatively
     prime positive integers. What is $m+n$?
```

Turn 1 - direct answer

```
<think>
To solve this problem, we need to find the radii of the circles in which the right triangles are
     inscribed. For a right triangle, the hypotenuse is the diameter of the circumscribed circle.
     Therefore, we can find the radius of each circle by dividing the length of the hypotenuse by
     2.

For the $3-4-5$ right triangle, the hypotenuse is 5, so the radius of circle $A$ is $\frac{5}{2}$.
     For the $5-12-13$ right triangle, the hypotenuse is 13, so the radius of circle $B$ is $\frac
     {13}{2}$.

The area of a circle is given by $\pi r^2$. Therefore, the area of circle $A$ is $\pi \left(\frac
     {5}{2}\right)^2 = \pi \frac{25}{4}$, and the area of circle $B$ is $\pi \left(\frac{13}{2}\
     right)^2 = \pi \frac{169}{4}$.

The ratio of the area of circle $A$ to the area of circle $B$ is:
\[
\frac{\pi \frac{25}{4}}{\pi \frac{169}{4}} = \frac{25}{169}
\]
Since 25 and 169 are relatively prime, the ratio is $\frac{25}{169}$. Therefore, $m+n = 25+169 =
     194$.

No further information is needed, so I can directly provide the answer.
</think>
<answer>194</answer>
```

Figure 8: Full trace of an AMC example. While the router (Qwen2.5-3B) has access to the Skill Handbook, it autonomously determines that the problem can be solved using its internal capability, refrains from issuing any `<search>` calls, and produces the correct final answer. This example highlights the flexibility of skill-based routing, where external model invocation is optional rather than mandatory.

PopQA Example: Adaptive Skill-Based Routing Recovers from a Noisy Initial Response

✓ Skill Handbook | 3 external calls | ✓ Correct

Skill Router Instruction

```
You are a skill-based model router. You are selecting the best model to answer a question by
      analyzing a question to identify required skills and their importance related to this
      question.

## Learned Skill Definitions (from validation)

### Entity Resolution and Attributes
Identify and link entities from descriptive phrases and retrieve their attributes or locations.
  - Descriptor-based Entity Linking: Resolve a specific entity from multi-clause descriptors (
      titles, awards, roles) without the name explicitly given.
    Examples: Problem 12: Who is the American singer-songwriter, who won an award for Best Female
      Video at the 2009 MTV Video Music Awards, and wrote a song for the 'AT&T Team USA Soundtrack
      '?, Problem 1: Who is the English Professional Footballer that is the niece of a former
      footballer born in the same place as where he plays?
  - Membership/Role Resolution: Identify an entity based on group membership and an additional role
       or credit.
    Examples: Problem 4: What member of the South Korean-Chinese boy group EXO stars in the
      upcoming movie "The Underdog"?, Problem 9: Maha Kali is an EP by what band from Stromstad
      that was formed in 1989?
  - Attribute/Location Retrieval: Extract a specific attribute or location associated with an
      entity from knowledge bases.
    Examples: Problem 15: Japanese Weekend School of New York has offices in the entertainment
      complex in what section of New Rochelle?

### Relational Composition and Constraints
Answer questions requiring chaining multiple relations and satisfying constraints across entities.
  - Multi-hop Bridge Reasoning: Combine two or more linked facts (A->B->C) to derive the answer.
    Examples: Problem 6: The Battle of Cambrai took place during a series of offensives that ended
      on what date?, Problem 3: Liz Rose has co-written songs with which artist including "White
      Horse" and "You Belong with Me"?
  - Relational Constraint Satisfaction: Apply constraints such as shared attributes (same
      birthplace/place) or time/place filters across relations to identify the correct entity.
    Examples: Problem 1: Who is the English Professional Footballer that is the niece of a former
      footballer born in the same place as where he plays?, Problem 9: Maha Kali is an EP by what
      band from Stromstad that was formed in 1989?
  - Reverse Relation Traversal: Start from works or properties and infer the originating entity via
       reverse edges (e.g., song -> artist).
    Examples: Problem 3: Liz Rose has co-written songs with which artist including "White Horse"
      and "You Belong with Me"?, Problem 10: A Pair of Brown Eyes and Wild Mountain Thyme is based
      from what artists song?

### Temporal and Ordinal Reasoning
Interpret dates, periods, and ordinal descriptors to resolve time-based queries.
  - Event-Date Alignment: Map events to their dates directly or via higher-level campaigns/series.
    Examples: Problem 6: The Battle of Cambrai took place during a series of offensives that ended
      on what date?
  - Ordinal Title/Sequence Disambiguation: Use ordinal descriptors (e.g., third, first) within a
      known sequence to identify the correct entity and associated dates.
    Examples: Problem 5: When did the the Antonine who was known as the third of the five good
      emperors live?
  - Temporal Qualifier Interpretation: Interpret temporal qualifiers such as 'first', 'upcoming',
      or year references to locate the relevant time frame.
    Examples: Problem 4: What member of the South Korean-Chinese boy group EXO stars in the
      upcoming movie "The Underdog"?, Problem 8: Shirley Breeden won her first Senate term in a
      narrow upset over the politician who was a member of what branch of the armed forces?

### Comparative, Classification, and Set Reasoning
Make comparisons, classify options by properties, and find common attributes across entities.
  - Property-based Option Selection: Choose among given options based on a specific property filter
      .
    Examples: Problem 2: Which is a black-and-white film, Flying Padre or Inside Job?
  - Numeric/Quantitative Comparison: Compare numeric attributes (counts, totals) across entities to
       determine the greater/lesser.
    Examples: Problem 14: Which player won more Grand Slam titles, Kevin Ullyett or Billie Jean
      King?
  - Set Intersection/Common Attribute Identification: Find a shared attribute (e.g., profession)
      between two entities.
    Examples: Problem 13: What profession does Leonty Magnitsky and Leonid Khachiyan have in common
      ?

### Language Robustness and Causal Inference
Handle noisy phrasing and infer causal or thematic relations from text.
  - Noisy/Paraphrase Normalization: Parse ungrammatical or awkward phrasing and map it to a
      coherent structured query.
    Examples: Problem 10: A Pair of Brown Eyes and Wild Mountain Thyme is based from what artists
      song?, Problem 11: In which song was written by singer-songwriter Taylor Swift and shares the
       optimistic lyrical message to a song called "Yodel It!"?
  - Causal/Narrative Why-Reasoning: Answer why-questions by identifying causes, motives, or
      precipitating events in historical narratives.
```

```
    Examples: Problem 7: Why did Rudolf Hess stop serving Hitler in 1941?
  - Thematic/Message Similarity Inference: Infer thematic similarity (e.g., optimistic message)
      between works when not explicitly linked by facts.
    Examples: Problem 11: In which song was written by singer-songwriter Taylor Swift and shares
      the optimistic lyrical message to a song called "Yodel It!"?

## Model Performance (learned from validation)

### LLaMA-3.1-70B-Instruct
Overall: 43% success (26/60)
Skill scores:
  - Property-based Option Selection: 100%
  - Numeric/Quantitative Comparison: 100%
  - Membership/Role Resolution: 67%
  - Thematic/Message Similarity Inference: 67%
  - Noisy/Paraphrase Normalization: 60%
Strengths: Property-based option selection, Stronger-than-peers multi-hop bridge reasoning,
     Paraphrase/noise normalization
Weaknesses: Descriptor-based entity linking, Relational constraint satisfaction, Relational
     constraint satisfaction across multiple entities

### Gemma-2-27B-Instruct
Overall: 48% success (29/60)
Skill scores:
  - Property-based Option Selection: 100%
  - Set Intersection/Common Attribute Identification: 100%
  - Numeric/Quantitative Comparison: 100%
  - Thematic/Message Similarity Inference: 67%
  - Attribute/Location Retrieval: 50%
Strengths: Property-based option selection (top-tier), Paraphrase/noise normalization, Thematic
     similarity inference
Weaknesses: Multi-hop bridge reasoning (mid), Descriptor-based entity linking, Relational
     constraint satisfaction

### Mistral-7B-Instruct
Overall: 30% success (18/60)
Skill scores:
  - Property-based Option Selection: 100%
  - Numeric/Quantitative Comparison: 100%
  - Attribute/Location Retrieval: 39%
  - Reverse Relation Traversal: 30%
  - Multi-hop Bridge Reasoning: 27%
Strengths: Occasional success on simple entity lookups, No demonstrated strengths on measured
     skills in this set, Cost/latency for very simple, single-hop queries
Weaknesses: Multi-hop bridge reasoning (low), Descriptor-based entity linking, Relational
     constraint satisfaction

### Mixtral-8x22B-Instruct
Overall: 45% success (27/60)
Skill scores:
  - Property-based Option Selection: 100%
  - Membership/Role Resolution: 100%
  - Set Intersection/Common Attribute Identification: 100%
  - Numeric/Quantitative Comparison: 100%
  - Attribute/Location Retrieval: 44%
Strengths: Multi-hop bridge reasoning (strong among evaluated models), Descriptor-based entity
     linking (partial strength), Multi-hop bridge reasoning (large-model tier)
Weaknesses: Descriptor-based entity linking, Relational constraint satisfaction, Paraphrase/noise
     normalization

### LLaMA-3.1-8B-Instruct
Overall: 25% success (15/60)
Skill scores:
  - Property-based Option Selection: 100%
  - Set Intersection/Common Attribute Identification: 67%
  - Numeric/Quantitative Comparison: 50%
  - Event-Date Alignment: 29%
  - Attribute/Location Retrieval: 22%
Strengths: Basic multi-hop (limited), No demonstrated strengths on measured skills in this set,
     Basic single-hop factoids
Weaknesses: Multi-hop bridge reasoning (below larger models), Descriptor-based entity linking,
     Relational constraint satisfaction

### Qwen2.5-7B-Instruct
Overall: 33% success (20/60)
Skill scores:
  - Property-based Option Selection: 100%
  - Set Intersection/Common Attribute Identification: 100%
  - Numeric/Quantitative Comparison: 100%
```

```
  - Attribute/Location Retrieval: 56%
  - Multi-hop Bridge Reasoning: 21%
Strengths: None clearly demonstrated in this sample, Partial capability in paraphrase/noise
     normalization, Simple fact lookup with low latency
Weaknesses: Multi-hop bridge reasoning (0%), Descriptor-based entity linking, Relational constraint
      satisfaction

## Cost Tiers (cheapest to most expensive)
- Cheap: Qwen2.5-7B-Instruct, LLaMA-3.1-8B-Instruct, Mistral-7B-Instruct
- Medium: Gemma-2-27B-Instruct
- Expensive: LLaMA-3.1-70B-Instruct, Mixtral-8x22B-Instruct

## Task
1. First, analyze the question below and identify which skills are needed, along with the
     percentage/weight of each skill (how important each skill is for answering this question).

Output your analysis in the following format inside <skill_analysis> tags:
<skill_analysis>
{
  "required_skills": [
    {"skill_id": "skill.id", "percentage": 50},
    {"skill_id": "skill.id", "percentage": 30},
    ...
  ],
  "reasoning": "Brief explanation of why these skills are needed"
}
</skill_analysis>

2. After providing the skill analysis, reflect on which model is best suited based on the skills
     required and model performance data above.

3. Route to that model using <search> tags and provide final answer in <answer>...</answer>

Every time you receive new information, you must first conduct reasoning inside <think> ... </think
     >. After reasoning, if you find you lack some knowledge, you can call a specialized LLM by
     writing a query inside <search> LLM-Name:Your-Query </search>.
!!! STRICT FORMAT RULES for <search>: !!!
    + You MUST replace LLM-Name with the EXACT name of a model selected from
      [Qwen2.5-7B-Instruct, LLaMA-3.1-8B-Instruct, LLaMA-3.1-70B-Instruct, Mistral-7B-Instruct,
       Mixtral-8x22B-Instruct, Gemma-2-27B-Instruct].
    + You MUST replace Your-Query with the EXACT same question as the original question below (DO
     NOT CHANGE IT).
    + NEVER copy or paste model descriptions into <search>.
    + NEVER output the placeholder format <search> LLM-Name:Your-Query </search>. Always replace
     both parts correctly.
Before each LLM call, you MUST explicitly reason inside <think> ... </think> about:
    + Why external information is needed.
    + Which skills from the catalog are required for this question.
    + Which model is best suited based on the model performance data above.
When you call an LLM, the response will be returned between <information> and </information>.
If you find that no further external knowledge is needed, you can directly provide your final
     answer to the original question inside <answer> ... </answer>, without additional explanation.


Question: Who was the composer of It's Going to Take Some Time?
```

Turn 1 -- noisy lookup

```
<think>
To answer this question, we need to identify the composer of the song \"It's Going to Take Some
     Time\". This requires the skill of Reverse Relation Traversal, which involves starting from a
      work (in this case, the song) and inferring the originating entity (the composer). We also
     need the skill of Attribute/Location Retrieval, as we are looking for a specific attribute (
     composer) associated with the song.
</think>

<skill_analysis>
{
  "required_skills": [
    {"skill_id": "Reverse Relation Traversal", "percentage": 60},
    {"skill_id": "Attribute/Location Retrieval", "percentage": 40}
  ],
  "reasoning": "We need to start from the song and infer the composer, and then retrieve the
     composer's attribute."
}
</skill_analysis>

<search>Qwen2.5-7B-Instruct: Who was the composer of It's Going to Take Some Time?</search>
```

```
<information>
The song "It's Going to Take Some Time" is a popular track from the musical "The Wedding Singer."
The composer of this song is Frank Wildhorn. Frank Wildhorn is an American composer known for his
     work in musical theater, including other notable works such as "Jekyll & Hyde" and "The Last
     5 Years."
He collaborated with lyricist Michael Gore on this particular song. The song was featured in the
     1998 film adaptation of "The Wedding Singer," which starred Adam Sandler.
Would you like more information about Frank Wildhorn or the musical "The Wedding Singer"? If not, I
      can provide insights on the song's significance in the context of the musical. Let me know
     your next step.
[... noisy repetitive content omitted: the same "let me know" lines repeat many times ...]
</information>
```

Turn 2 - uncertain attribution

```
<think>
The Qwen2.5-7B-Instruct model couldn't provide a definitive answer and includes excessive filler
     content. I need to consult another model with stronger factual knowledge about music.
</think>

<search>LLaMA-3.1-70B-Instruct: Who was the composer of It's Going to Take Some Time?</search>

<information>
"It's Going to Take Some Time" is a song written by Toni Stern and Carole King. It was recorded by
     Carole King in 1971.
However, I am not sure if she was the composer of the song.
</information>
```

Turn 3 -- cross verification

```
<think>
LLaMA-3.1-70B identifies Toni Stern and Carole King as the writers, but explicitly expresses
     uncertainty about whether Carole King should be treated as the "composer" requested. Cross-
     verify with another model.
</think>

<search>Mixtral-8x22B-Instruct: Who was the composer of It's Going to Take Some Time?</search>

<information>
The composer of "It's Going to Take Some Time" is Carole King. She co-wrote the song with Toni
     Stern.
The song was first released on her 1971 album "Music" and later became a hit single for The
     Carpenters in 1972.
</information>
```

Turn 4 -- final answer

```
<answer>Carole King</answer>
```

Figure 9: Full trace for a PopQA example. Guided by skill-based analysis, the router detects that the initial model response is incomplete or ambiguous, re-routes to alternative models, and performs cross-model verification before finalizing the answer. This demonstrates robust recovery from intermediate errors.

Agent Orchestration Instruction (Full Specification)

```
You are a skill-based orchestrator for multi-step question answering.
You select the best tool (search|code|answer) and model by analyzing required skills.

## Problem to Solve
{{problem}}

## Current Context
{{context}}

python code and execution outputs:
{{execution_results}}

---

## Quick Reference: What You Need to Do

CRITICAL REQUIREMENT: Before making ANY tool call, you MUST:

1. Inside <think>...</think>, analyze required skills and output in <skill_analysis> tags.
2. Then choose the appropriate tool with the selected model based on the skill analysis.

The context above may be long - scroll back to see the problem and context, then follow the
     instructions below.

---
```

```
## Available Tools and Models (Baseline)

Tool: search
Models: search-1 ($10/M), search-2 ($2/M), search-3 ($0.8/M)
Description: Search for missing information

Tool: code | enhance_reasoning
Models: reasoner-1 ($10/M), reasoner-2 ($2/M), reasoner-3 ($0.8/M)
Description: Write and execute Python code to solve the problem

Tool: answer
Models: answer-1 ($10/M), answer-2 ($2/M), answer-3 ($0.9/M),
        answer-4 ($0.8/M), answer-math-1 ($0.9/M), answer-math-2 ($0.2/M)
Description: Extract the final answer if you think you have enough information to answer the
     problem
```

```
## Learned Skill Definitions

### Search Skills

- search.control.zero_result_guardrail: Detect zero-result searches and prevent escalation to non-
     search stages until several structured reformulations and decomposition attempts have been
     tried.
- search.control.stage_adherence: Adhere to the search stage's intended tools and behaviors; avoid
     misusing the code tool when the task requires query generation and retrieval.
- search.control.answer_block_on_zero_results: Prevent advancing to the answer stage when searches
     return zero results; require iterative reformulation attempts first.
- search.query_planning.multi-hop.list_then_filter.attribute_match: Plan retrieval by first listing
      entities that satisfy a temporal/role constraint, then filtering them by a specific
     attribute or name match (e.g., middle name matching a category like Scottish monarchs).
- search.tool_choice.scope_awareness: Avoid search for well-known facts that can be confidently
     recalled; reserve search for low-confidence or obscure queries.
- search.efficiency.tool_call_minimization: Avoids unnecessary tool invocations (search or code)
     when they do not contribute to retrieval, reducing cost and noise.
- search.tool_choice.code_for_computation_only: Uses the code tool solely for numerical or data-
     processing steps after retrieval, not for information retrieval or scraping.
- search.retrieval.robust_authoritative_search: Plan and execute single- and multi-hop web
     retrieval with entity/event disambiguation and numeric/temporal reasoning while explicitly
     prioritizing authoritative sources (e.g., Britannica/Wikipedia for general facts, IOC/FIFA/
     World Athletics for sports, NOAA/NASA and peer-reviewed literature for science, official
     corporate and regulator sites for filings). Construct resilient queries using operators and
     filters (quotes, OR, -, site:, filetype:, language, date ranges); align dates, ranges, and
     units; iteratively broaden/narrow via synonyms and result feedback; detect empty/malformed or
      low-quality-result queries and systematically retry/refine (including minimal viable queries
     ) until at least one high-quality supporting source is obtained or escalate for clarification
      if recovery fails.
Examples: Compute 10 years after Neil Armstrong's birth, then search nasa.gov with date filters for
      events on that date; refine queries if initial filters yield no results., Who won the women'
     s marathon at the 2024 Olympics? Prioritize olympics.com or World Athletics results; apply
     year filters and exact-match athlete names; retry with synonyms or broader terms if initial
     query is sparse.
- search.strategy.query_formulation_refinement_and_verification: Formulate clear, targeted search
     queries using key entities and relations; when results are weak or conflicting, rapidly
     iterate with synonyms, paraphrases, broader/narrower terms, constraints, operators, time
     filters, and source scoping. Perform lightweight verification searches even when an answer is
      recalled, corroborate with credible sources, and adjust conclusions based on evidence.
Examples: Need: When was the Kyoto Protocol adopted? Start with 'Kyoto Protocol adopted date';
     iterate with 'Kyoto Protocol adoption 1997', add site:unfccc.int and time filters to confirm.,
      Recall an author of a concept; perform a brief search to confirm the name and cite the
     seminal paper, revising if results conflict.
# Additional skills with performance data available:
- retrieval.date_lookup
- retrieval.entity_lookup
- retrieval.multi_hop
- search.basic_retrieval
- search.control.stage_adherence.strict
- search.cost_effectiveness
- search.cost_efficiency
- search.date_lookup
- search.disambiguation
- search.entity_lookup
- search.entity_retrieval
- search.fact_retrieval
- search.factual.lookup
- search.factual_single_hop
- search.lookup.dates
- search.lookup.entities
- search.multi_hop
- search.multi_hop.light
```

```
- search.multi_hop.simple
- search.multi_hop_facts
- search.multi_hop_light
- search.multihop
- search.precision.general
- search.query_planning.multi-hop
- search.recall.general
- search.retrieval.cost_efficiency
- search.retrieval.dates
- search.retrieval.factoid
- search.retrieval.facts
- search.retrieval.multi_hop
- search.retrieval.multi_hop_light
- search.retrieval.precision
- search.retrieval.recall
- search.retrieval.single_fact
- search.retrieval.single_hop
- search.single_hop
- search.strategy.source_prioritization
- search.web.fact_lookup
- search.web.lookup
- search.web.multi_hop

## Reasoning Skills

- code.generation.codeblock_integrity: Maintain clean code blocks containing only valid Python,
    avoiding mixed prose or unrelated text.
- code.math.date_calculation: Computing differences between dates, handling month/day boundaries
    and leap years
- code.data.aggregation: Summing, averaging, grouping, and comparing values across lists or tables
- code.math.multi_step: Accurate computation across multiple operations or items where mental math
    is error-prone.\n- code.execution.iteration_budgeting: Limits repeated tool invocations;
    detects diminishing returns and switches strategy (e.g., back to search/answer) to avoid
    wasted cost.
- code.execution.token_economy: Generates concise code and avoids unnecessary large literals or
    pasted text to minimize token usage and cost.
- code.stage_adherence: Ensure the code stage produces valid, executable Python only; avoid
    narrative text or non-code dumps.
- code.execution.strategy_handoff_on_failure: After detecting that code execution cannot progress (
    e.g., needs external data or repeated non-executable outputs), emit a clear handoff signal to
     search/answer instead of retrying.
- code.data.tabular_parsing_ops_ranking: Extract structured tables from semi-structured text (lists
    , textual/markdown/CSV-like tables) and perform tabular filtering, multi-key sorting, stable
    tie-aware ranking (top/bottom N with ties), grouping, and aggregation. Supports producing
    rank annotations and operating on small tables or tables returned by search.
Examples: Parse this markdown table of product sales, then return the top 10 products by revenue (
    include ties) and compute average revenue per category., From this bullet list of shipped
    orders, extract fields (order_id, customer, ship_date, total), filter status='shipped', sort
    by ship_date descending, then group by customer and sum total_spent.
- code.execution.offline_minimal_inline_structures: Decide whether code execution is warranted for
    a task. When appropriate, produce the smallest self-contained, offline-safe Python snippet.
    Replace external I/O with tiny in-memory structures (lists/dicts/DataFrames) to perform
    lookups and computations. Validate outputs against requested type/format/schema/units and
    iterate until compliant. When code adds no value or violates constraints, abstain and answer
    directly while avoiding noisy outputs.
Examples: A request to map a few codes to names without using files or APIs; construct a small
    inline dict and return the mapped results., Summarize a CSV hosted at a URL in a no-network
    environment; create a minimal inline DataFrame with representative rows, compute summary
    stats offline, and print validated outputs.
- code.datetime.date_difference: Compute accurate differences between dates (years/months/days)
    with proper inclusivity and leap-year handling, and generate/execute Python datetime code (
    including parsing) to achieve the same.
Examples: What is the difference between 2020-02-29 and 2021-03-01 in years, months, and days (
    inclusive)?, Write Python code to parse two dates and return the year, month, and day deltas,
     correctly handling leap years.
# Additional skills with performance data available:
- code.date.arithmetic
- code.execution.syntax_self_check
- code.math.arithmetic

## Answer Skills
No dedicated skills defined.
Use general performance of answer models to select the best model for the answer stage if you think
     you have enough information to answer the problem.

---

## Model Performance (learned from validation)

### Search Models
```

```
#### search-1 (gpt-5)
Overall: 55% success (11/20)
Skill scores:
  - search.factual_single_hop: 100%
  - search.multi_hop: 100%
  - search.retrieval.multi_hop: 100%
  - search.basic_retrieval: 100%
  - search.retrieval.single_hop: 67%
  - search.tool_choice.scope_awareness: 67%
  - search.factual.lookup: 60%
  - search.fact_retrieval: 60%
  - search.retrieval.cost_efficiency: 50%
  - search.cost_effectiveness: 50%
  - search.cost_efficiency: 40%
  - search.retrieval.precision: 40%
  - search.retrieval.recall: 40%
  - search.control.stage_adherence: 17%
  - search.query_planning.multi-hop: 0%
  - search.control.zero_result_guardrail: 0%
  - search.control.answer_block_on_zero_results: 0%
  - search.control.stage_adherence.strict: 0%
  - search.tool_choice.code_for_computation_only: 0%
  - search.efficiency.tool_call_minimization: 0%
  - search.retrieval.robust_authoritative_search: 0%
  - search.strategy.query_formulation_refinement_and_verification: 0%
Strengths:
  - answered from recall despite zero search results
  - robust multi-hop retrieval
  - maintained stage separation (did not misuse code tool)
Weaknesses:
  - invoked code tool with no effect (unnecessary tool usage)
  - no effective search results and no reformulation
  - issued a search call but returned no results

#### search-2 (gpt-5-mini)
Overall: 50% success (10/20)
Skill scores:
  - search.cost_efficiency: 100%
  - search.basic_retrieval: 100%
  - search.fact_retrieval: 100%
  - search.multi_hop: 67%
  - search.retrieval.single_fact: 67%
  - search.retrieval.precision: 60%
  - search.retrieval.recall: 60%
  - search.retrieval.multi_hop: 60%
  - search.tool_choice.scope_awareness: 50%
  - search.control.stage_adherence: 20%
  - search.query_planning.multi-hop: 0%
  - search.strategy.source_prioritization: 0%
  - search.control.zero_result_guardrail: 0%
  - search.control.answer_block_on_zero_results: 0%
  - search.control.stage_adherence.strict: 0%
  - search.tool_choice.code_for_computation_only: 0%
  - search.efficiency.tool_call_minimization: 0%
  - search.retrieval.robust_authoritative_search: 0%
  - search.strategy.query_formulation_refinement_and_verification: 0%
Strengths:
  - accurate recall of both anchor years and mental subtraction
  - concise final answer
  - maintained stage separation (did not misuse code tool)
Weaknesses:
  - no effective search results
  - no reformulation after zero results
  - relied on an unverified answer without retrieval

#### search-3 (Qwen/Qwen3-32B)
Overall: 65% success (13/20)
Skill scores:
  - search.retrieval.factoid: 100%
  - search.factual_single_hop: 100%
  - search.cost_efficiency: 100%
  - search.date_lookup: 100%
  - search.recall.general: 100%
  - search.precision.general: 100%
  - search.single_hop: 100%
  - search.retrieval.facts: 100%
  - search.retrieval.dates: 100%
  - search.lookup.entities: 100%
  - search.lookup.dates: 100%
```

```
  - search.multi_hop.light: 100%
  - search.factual.lookup: 100%
  - search.multi_hop.simple: 100%
  - search.basic_retrieval: 100%
  - search.fact_retrieval: 100%
  - search.disambiguation: 100%
  - search.web.lookup: 100%
  - search.entity_retrieval: 100%
  - search.multi_hop_facts: 100%
  - search.retrieval.single_fact: 100%
  - search.web.fact_lookup: 100%
  - search.web.multi_hop: 100%
  - search.cost_effectiveness: 100%
  - search.tool_choice.scope_awareness: 75%
  - search.retrieval.multi_hop: 67%
  - search.entity_lookup: 67%
  - search.retrieval.single_hop: 67%
  - search.retrieval.multi_hop_light: 67%
  - search.multi_hop_light: 67%
  - search.retrieval.precision: 67%
  - search.retrieval.recall: 67%
  - search.multi_hop: 60%
  - search.multihop: 60%
  - search.efficiency.tool_call_minimization: 50%
  - search.query_planning.multi-hop: 0%
  - search.strategy.source_prioritization: 0%
  - search.control.zero_result_guardrail: 0%
  - search.control.stage_adherence: 0%
  - search.control.answer_block_on_zero_results: 0%
  - search.control.stage_adherence.strict: 0%
  - search.tool_choice.code_for_computation_only: 0%
  - search.retrieval.robust_authoritative_search: 0%
  - search.strategy.query_formulation_refinement_and_verification: 0%
Strengths:
  - answered from recall despite zero search results
  - kept cost low with a single lightweight search attempt and no unnecessary tools
  - low-cost factual retrieval
Weaknesses:
  - no effective search query produced results
  - did not reformulate after zero results
  - issued a search call but returned no results

### Reasoning Models

#### reasoner-1 (gpt-5)
Overall: 45% success (10/22)
Skill scores:
  - code.math.date_calculation: 60%
  - code.math.arithmetic: 60%
  - code.data.aggregation: 60%
  - code.execution.iteration_budgeting: 8%
  - code.generation.codeblock_integrity: 0%
  - code.execution.token_economy: 0%
  - code.execution.syntax_self_check: 0%
  - code.stage_adherence: 0%
  - code.execution.strategy_handoff_on_failure: 0%
  - code.execution.offline_minimal_inline_structures: 0%
  - code.datetime.date_difference: 0%
Strengths:
  - kept code attempts to a single run (iteration budgeting)
  - lowest orchestration cost among peers
  - eventual handoff to answer after code provided no value
Weaknesses:
  - no executable code produced
  - ineffective search (no useful retrieval)
  - relied on non-code answer without tool-aligned computation/output

#### reasoner-2 (gpt-5-mini)
Overall: 62% success (13/21)
Skill scores:
  - code.math.multi_step: 100%
  - code.data.aggregation: 100%
  - code.date.arithmetic: 100%
  - code.execution.iteration_budgeting: 38%
  - code.execution.token_economy: 10%
  - code.generation.codeblock_integrity: 6%
  - code.execution.syntax_self_check: 0%
  - code.stage_adherence: 0%
  - code.execution.strategy_handoff_on_failure: 0%
```

```
  - code.execution.offline_minimal_inline_structures: 0%
  - code.datetime.date_difference: 0%
Strengths:
  - arrived at the correct concise numeric answer
  - iterated with multiple tool calls
  - kept Python syntactically valid (no execution errors)
Weaknesses:
  - inserted non-code/prose into a code block
  - no printed/output result from code executions
  - ineffective search (no useful retrieval)

#### reasoner-3 (Qwen/Qwen2.5-Coder-32B-Instruct)
Overall: 50% success (11/22)
Skill scores:
  - code.math.date_calculation: 100%
  - code.math.arithmetic: 100%
  - code.data.aggregation: 100%
  - code.data.tabular_parsing_ops_ranking: 100%
  - code.execution.strategy_handoff_on_failure: 50%
  - code.execution.iteration_budgeting: 31%
  - code.execution.token_economy: 30%
  - code.generation.codeblock_integrity: 0%
  - code.execution.syntax_self_check: 0%
  - code.stage_adherence: 0%
  - code.execution.offline_minimal_inline_structures: 0%
  - code.datetime.date_difference: 0%
Strengths:
  - produced the correct concise numeric answer
  - cost-efficient with a single code call and low token usage
  - date calculations
Weaknesses:
  - did not produce or display any code output (no executable/printed result)
  - overkill for simple factual lookups
  - produced no observable output from the code tool

### Answer Models

#### answer-1 (gpt-5)
Overall: 75% success (21/28)
Strengths:
  - returned a single exact number
  - used direct recall without retrieval
  - directly recalled the cultural fact
Weaknesses:
  - did not perform the required multi-hop reasoning
  - no verification / disambiguation checks
  - higher cost than answer-3

#### answer-2 (gpt-5-mini)
Overall: 87% success (13/15)
Strengths:
  - returned a single exact number
  - direct recall of well-known event
  - concise, exact output
Weaknesses:
  - ineffective query led to no results and no refinement attempt
  - defaulted to an unverified memorized answer
  - did not execute the multi-hop chain

#### answer-3 (meta-llama/Llama-3.3-70B-Instruct)
Overall: 62% success (8/13)
Strengths:
  - returned a single exact number
  - low overall overhead
  - immediate direct recall
Weaknesses:
  - performed a search that yielded no results and did not iterate/refine
  - failed to disambiguate role timing
  - did not verify the attribute definition

#### answer-4 (Qwen/Qwen3-32B)
Overall: 47% success (7/15)
Strengths:
  - returned a single exact number
  - correct direct recall
  - no tool usage
Weaknesses:
  - higher token usage compared to peers
  - no-hit search without query refinement or alternate strategy
```

```
  - no multi-hop reasoning or validation before finalizing

#### answer-math-1 (Qwen/Qwen2.5-Math-72B-Instruct)
Overall: 43% success (6/14)
Strengths:
  - returned a single exact number
  - correct direct recall beyond math domain
  - concise exact answer
Weaknesses:
  - misused the code tool by emitting non-Python content under the code stage
  - retrieved irrelevant/low-authority content and failed to process it
  - no error handling or validation pipeline

#### answer-math-2 (Qwen/Qwen2.5-Math-7B-Instruct)
Overall: 15% success (2/13)
Strengths:
  - attempted search before answering
  - produced the correct content via recall
  - concise single-entity output
Weaknesses:
  - produced an incorrect numeric answer
  - did not correctly align the two event dates before computing the difference
  - likely exact-match formatting or evaluator compliance issue
```

```
## Detailed Instructions

STEP 1 - REQUIRED
Based on the Problem and Context shown at the top, determine the next stage:
(search | code | answer).

- search: retrieve missing external information needed to answer the problem.
- code: write and execute Python code to compute or process data.
- answer: synthesize gathered information into the final answer.

STEP 2 - REQUIRED FORMAT
After deciding the next stage:

1. Reason inside <think>...</think>:
   - Why this stage is needed.
   - Which skills are required.
   - The relative importance of each skill.

2. Output the structured skill analysis inside <skill_analysis> tags:

<skill_analysis>
{
  "required_skills": [
    {"skill_id": "skill.id", "percentage": 50},
    {"skill_id": "skill.id", "percentage": 30}
  ],
  "reasoning": "Brief explanation of why these skills are needed."
}
</skill_analysis>

STEP 3
Choose the appropriate tool and model based on the skill analysis.

---

## Final Reminders

CRITICAL:
- The <skill_analysis> block is MANDATORY.
- It MUST appear BEFORE any tool call.

IMPORTANT:
- When calling a tool, you MUST specify the model parameter using the model alias (e.g., \"answer
    -1\", \"search-1\", \"reasoner-1\").
- Use exact model names from the Available Models section above.

Now, based on the Problem and Context shown at the top, analyze what should be the next stage (
    search|code|answer), provide the detailed skill analysis needed for the next stage in the <
    skill_analysis> tags and then choose an appropriate tool.
```

Figure 10: Agent orchestration instruction used for FRAMES, integrating the task query, execution history context, and the selected Skill Handbook to enable skill-aware orchestration.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.
Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.
The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.
IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

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 paper's main contributions: a skill-aware orchestration framework, the construction of reusable skill-level knowledge from execution experience, agent profiles that estimate skill-specific competence and cost, and deployment-time routing using this information. The claims are supported by the empirical results in Section 5.

   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 discussed limitations in Appendix A.

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 uses the standard Beta–Bernoulli conjugacy result for estimating skill-specific success probabilities. We provide the proof in Appendix E.

   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 describes the experimental setups in Section 5 with additional details in Appendix C to support reproducibility.

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: [Yes]

   Justification: The paper uses open-source datasets and benchmarks, and the code is publicly available at https://github.com/jiayuww/SkillOrchestra. We provide implementation details in Appendix C.

   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: We provide the details in Appendix C.

   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: [Yes]

   Justification: All results are averaged over three runs. We report the standard deviation in Appendix J.

   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: [Yes]

   Justification: We include the compute resources usage in Appendix C.

   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: The research conforms to the 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: [Yes]

    Justification: We discuss the societal impacts in Appendix A.

    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 paper does not release pretrained language models, image generators, scraped datasets, or other high-risk assets.

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: [Yes]

Justification: The paper cites the existing benchmarks, datasets, models, and software packages used in the experiments.

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 new experimental artifacts, including the learned skill handbook, prompts, configurations, and/or evaluation scripts. These assets are documented in the Appendix K.

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: The paper does not involve crowdsourcing experiments or research with 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: The paper does not involve crowdsourcing experiments or research with human subjects, so IRB approval or equivalent review is 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: [N/A]

    Justification: LLMs were used only for writing assistance, such as polishing and formatting the manuscript. They were not used as an important, original, or non-standard component of the core methodology, experiments, or scientific claims.

    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.