TALENTRL: STRATEGY-GUIDED REINFORCEMENT LEARNING FOR HIERARCHICAL SKILL ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Natural-language skills let large language model (LLM) agents reuse procedures across tasks, yet most systems retrieve individual Skills directly from a flat SkillBank without explicitly modeling the high-level strategy that governs their selection and use. We introduce TalentRL, a strategy-guided reinforcement learning framework for hierarchical routing over procedural memory. Coarse-grained *Talents* summarize recurring skill requirements to guide routing, while fine-grained *Skill* cards provide actionable procedures. For each task, the policy selects a Talent to guide execution and softly prioritize its associated Skills during similarity-based retrieval. However, because task outcomes depend jointly on Talent selection, retrieved Skill context, and downstream execution, terminal rewards cannot directly credit the routing decision. TalentRL therefore trains Talent routing by comparing rewards across matched intervention rollouts, while optimizing downstream execution with group-relative advantages. Across seven open-domain and multi-hop question-answering benchmarks, Talent-aware supervised fine-tuning improves average success rate by 7.53 and 6.71 percentage points over the zero-shot baselines for Qwen2.5-7B and Qwen3-8B, respectively. Reinforcement learning further improves the Talent-aware SFT checkpoint, enabling TalentRL to achieve performance competitive with state-of-the-art SkillRL across the seven benchmarks. These gains extend to the agentic benchmarks ALFWorld and WebShop, where TalentRL consistently improves over SkillRL.

1 INTRODUCTION

Large language model (LLM) agents solve interactive tasks by interleaving reasoning, retrieval, and environment actions (Yao et al., 2023). To improve behavior across episodes, verbal-memory methods retain linguistic reflections from prior interactions (Shinn et al., 2023), while outcome-based reinforcement learning (RL) optimizes complete trajectories using task rewards (Jin et al., 2025). However, most procedural-memory systems retrieve individual Skills from a flat SkillBank, leaving the high-level strategy governing their selection and use unmodeled by the policy. Explicitly training the policy model to select a strategy raises a credit-assignment challenge, since the final reward reflects both routing and execution and cannot directly evaluate the routing decision. This motivates a central question: how can an agent learn effective routing over structured procedural memory from task-level outcomes?

Prior work has progressed from extracting reusable insights across trajectories to distilling successful and failed interactions into verified, evolving libraries of procedural Skills (Zhao et al., 2024; Mi et al., 2026; Zhang et al., 2026; Xia et al., 2026). Recent methods couple Skill generation or curation with hierarchical, joint, or cross-task RL (Li et al., 2026; Shi et al., 2026; Yao et al., 2026). These studies show that procedural memory can be distilled, evolved, and coupled to policy learning, but three challenges remain. First, direct task-to-Skill matching treats procedural memory as a flat collection, leaving the strategic roles of individual Skills implicit. As the SkillBank grows, similarity alone may struggle to prioritize among semantically related Skills under a limited context budget. Second, embedding-based retrieval does not rely on the policy model, whereas learned selectors typically act directly on individual Skills without first selecting a higher-level strategy. Third, terminal rewards reflect the joint effects of strategy selection, Skill retrieval, and execution, providing no direct credit signal for the routing decision.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To this end, we propose TalentRL, which separates strategic routing, procedural retrieval, and downstream execution through a predefined hierarchical procedural memory. Coarse-grained *Talents* summarize recurring patterns of Skill requirements and serve as reusable routing strategies, while fine-grained *Skill cards* encode executable procedures. For each task, the policy model selects a Talent to guide execution and softly prioritize its associated Skills. The predefined retriever then combines task–skill similarity with the Talent prior and supplies the top-ranked Skill cards to the policy model during execution. Before RL, we construct Talent-aware SFT demonstrations from paired rollouts to initialize both strategy selection and task execution.

However, since task outcomes jointly reflect Talent selection, retrieved Skill context, and downstream execution, using terminal rewards to train Talent routing poses a credit-assignment challenge in RL. TalentRL addresses this challenge by using reward contrasts across matched intervention rollouts to train Talent routing and group-relative advantages to optimize downstream execution. Specifically, the interventions hold the task fixed while withholding retrieved Skill cards, reducing the retrieval budget, or removing both Talent guidance and its retrieval prior. The resulting reward differences capture the joint contribution of Talent guidance and Talent-conditioned Skill retrieval, rather than isolating the effect of the Talent label itself. TalentRL applies the intervention-derived routing signal to the Talent response, while subsequent execution actions receive group-relative advantages.

Our contributions are threefold:

- We propose an outcome-guided Talent-aware SFT data generation pipeline that compares candidate Talents through paired rollouts and converts positive-utility Talent-conditioned trajectories into routing and execution supervision.
- We introduce TalentRL, a strategy-guided RL framework that combines hierarchical Skill routing with intervention-based credit for Talent selection and group-relative optimization of downstream execution.
- We demonstrate that Talent-aware SFT improves performance across seven open-domain and multi-hop question-answering benchmarks, while TalentRL surpasses state-of-the-art SkillRL after RL and delivers consistent gains on ALFWorld and WebShop.

2 Related Work

2.1 Skill Memory and Representation

Procedural memory enables LLM agents to transfer experience across tasks by storing reusable knowledge of how to act. In the CoALA memory taxonomy (Sumers et al., 2024), procedural knowledge governs agent behavior rather than recording observations or factual knowledge. Existing methods represent such knowledge in different forms and at different levels of abstraction. Natural-language representations range from verbal feedback in Reflexion (Shinn et al., 2023) and cross-trajectory insights in ExpeL (Zhao et al., 2024) to context-conditioned guidelines in AutoGuide (Fu et al., 2024) and step-level instructions or higher-level scripts in MemP (Fang et al., 2025). At coarser levels of abstraction, Buffer of Thoughts (Yang et al., 2024b) retains reusable reasoning templates, while AWM (Wang et al., 2025b) abstracts trajectories into workflows.

Executable procedural memory represents reusable behavior as programs or callable tools, as in Voyager (Wang et al., 2023), LATM (Cai et al., 2024), and CRAFT (Yuan et al., 2024). Together with natural-language memory, these approaches maintain procedural knowledge outside the model, where it can be inspected, retrieved, invoked, or composed at inference time. Other methods encode procedural knowledge in model parameters or continuous representations, as in SKILL0 (Lu et al., 2026) and SoftSkill (Tao et al., 2026). TalentRL adopts the explicit natural-language setting and organizes it hierarchically by linking fine-grained Skill cards to coarse-grained Talents that summarize recurring patterns of Skill requirements.

2.2 Self-Evolving Skill Learning

Research on self-evolving Skill memory studies how Skill libraries improve as agents accumulate experience. One line of work treats external Skills as editable state while keeping the executor fixed. Skill-Pro (Mi et al., 2026) revises executable procedures under score-based verification, SkillOpt

(Yang et al., 2026) applies bounded edits to a Skill document and retains only validation-improving updates, and Trace2Skill (Ni et al., 2026) consolidates recurring lessons from execution trajectories into portable Skill directories. By separating procedural adaptation from parameter updates, these methods keep memory revisions inspectable and facilitate their reuse across tasks. Validation is therefore central: an update should improve performance beyond the trajectories that produced it.

Another line of work co-adapts procedural memory and the policy. SkillRL (Xia et al., 2026) recursively evolves a hierarchical SkillBank during reinforcement learning. Skill1 (Shi et al., 2026) trains a unified policy for Skill selection, utilization, and distillation, while SkillRise (Yao et al., 2026) credits Skill edits according to their downstream utility across tasks. Related methods specialize this joint adaptation through different architectures or domains: CoSkill (Feng et al., 2026) uses reasoning and meta-skill agents to evolve a hierarchical SkillBank, SAGE (Wang et al., 2025a) couples RL with a self-improving external Skill library, and SearchSkill (Hu et al., 2026) evolves search-specific Skills for tool-augmented reasoning.

2.3 Skill Routing and Credit Assignment

As Skill libraries grow, agents must determine which procedures to retrieve and how to use or combine them for a given task. One line of work develops structured mechanisms for Skill retrieval and composition. SkillRouter (Zheng et al., 2026) studies full-text retrieval and reranking at registry scale, while SkillLens (Miao et al., 2026) organizes strategies, procedures, and primitives in a multi-granularity graph. SkillComposer (Zhao et al., 2026) jointly predicts a Skill subset, its cardinality, and its execution order, and HyperSkill (Xu et al., 2026) connects subtasks and reusable Skills through a hypergraph. These methods explore how library organization, retrieval, and composition shape the procedural knowledge available to an agent.

Another line of work learns Skill selection from task outcomes. ARISE (Li et al., 2026) introduces a hierarchical manager–worker policy with Skill-level rewards. From an evaluation perspective, SkillCoach (Zhu et al., 2026) assesses Skill selection, following, composition, and reflection, providing a more detailed view of Skill use than final task success alone. When the policy model selects a high-level strategy to guide Skill retrieval, terminal rewards reflect both that selection and subsequent execution, making the routing decision difficult to evaluate on its own. To address this challenge, TalentRL trains the policy model to select a Talent that guides execution and softly prioritizes associated Skills through a predefined similarity-based retriever. TalentRL trains this routing decision by comparing rewards across matched rollouts under controlled context interventions, while optimizing downstream execution with group-relative advantages.

3 TalentRL: Strategy-Guided Skill Routing

Existing Skill-augmented LLM frameworks such as SkillRL (Xia et al., 2026) use similarity-based retrieval to supply task-relevant Skill cards to the policy model before execution, without explicitly involving the model in high-level strategy selection. In contrast, TalentRL introduces a policy-selected Talent that guides both Skill retrieval and subsequent task execution. We first formalize Skill-augmented task execution, then describe Talent organization, Talent-conditioned retrieval, and the construction of Talent-aware SFT data.

3.1 Preliminaries

LLM agents with Skill memory. Let $\mathcal{D}$ denote the task distribution and $x \sim \mathcal{D}$ a task description. The agent has access to a fixed SkillBank $\mathcal{S}$ of natural-language procedural cards. Given a task x, a similarity-based retriever scores each Skill $s \in \mathcal{S}$ using the embedding-similarity score $q(x, s)$ and retrieves the top-K cards:

$$\mathcal{S}_K(x) = \mathrm{TopK}_{s \in \mathcal{S}}\, q(x, s), \tag{1}$$

where K is the number of retrieved Skill cards.

Let $h_0 = x$. At execution turn $j \geq 1$, the agent observes $h_{j-1} = (x, a_1, o_1, \ldots, a_{j-1}, o_{j-1})$, where a_j is an action and o_j is the resulting environment observation. Conditioned on the interaction history and retrieved Skill cards, the policy model π_θ, parameterized by θ, samples the next action:

$$a_j \sim \pi_\theta(\cdot \mid h_{j-1}, \mathcal{S}_K(x)). \tag{2}$$

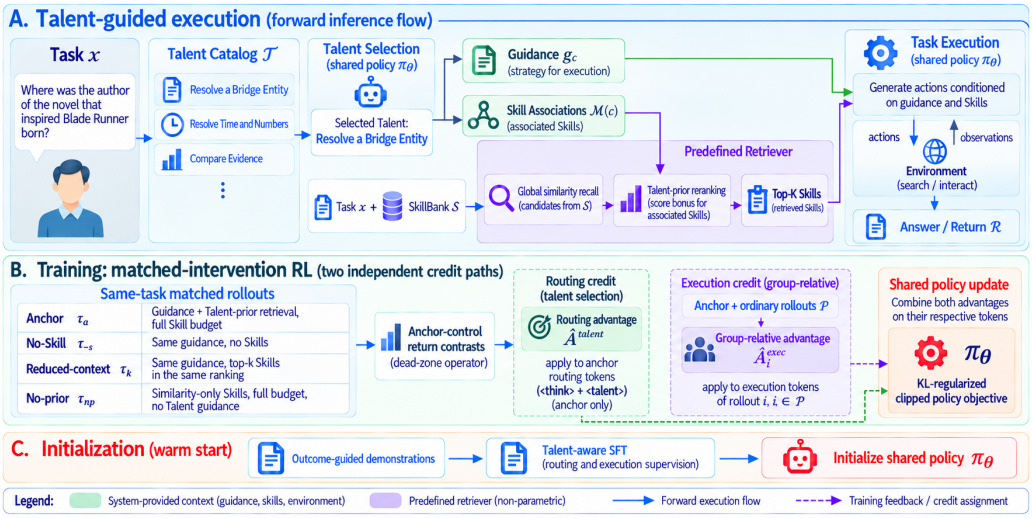

Figure 1: Overview of TalentRL. **(A)** The policy selects a Talent that guides execution and provides a soft prior for similarity-based Skill retrieval. **(B)** Matched-intervention return contrasts provide routing credit, while group-relative advantages provide execution credit; both optimize the shared policy through their respective response tokens. **(C)** Talent-aware SFT initializes routing and execution from outcome-guided demonstrations.

The environment then produces o_j and updates the history to $h_j = (h_{j-1}, a_j, o_j)$. Let $\tau = (x, a_1, o_1, \ldots, a_T, o_T)$ denote the interaction trajectory, where T is the number of execution turns. We optimize the expected episodic return

$$\max_\theta J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, \tau\sim\pi_\theta(\cdot|x)} \left[R(\tau)\right], \tag{3}$$

where $R(\tau)$ is the episodic return and τ is generated by the policy's interaction with the environment for task x. In this formulation, the retrieval mechanism is fixed, while the policy model learns to execute tasks using the retrieved Skill cards. We next introduce Talent selection as a policy action that guides both retrieval and execution.

3.2 Talent Organization and Skill Retrieval

Let $\mathcal{T}$ denote a catalog of coarse-grained strategies. Each Talent $c \in \mathcal{T}$ has a description d_c for strategy selection, guidance g_c for task execution, and a set of associated Skills $\mathcal{M}(c) \subseteq \mathcal{S}$. Each Skill is associated with at least one Talent, and these associations may overlap across Talents. Following the trajectory-distillation strategy of SkillRL (Xia et al., 2026), we use an LLM as a judge to analyze successful and failed trajectories, distilling reusable procedures into Skill cards and summarizing recurring skill requirements into Talents. TalentRL also supports validation-driven SkillBank updates during RL, generating additional Skills from failed validation trajectories when the success rate falls below a configured threshold. However, all experiments reported in this paper keep the SkillBank and Talent–Skill associations fixed. Table 5 in Appendix B lists the eight Talents used for search tasks and their associated Skill cards.

Talent-conditioned Skill retrieval. For task x, the retriever first constructs a candidate pool $\mathcal{R}(x) \subseteq \mathcal{S}$ using task–skill similarity. Given the selected Talent c, it reranks these candidates using

$$\widetilde{q}(x, s \mid c) = q(x, s) + \lambda \mathbb{I}[s \in \mathcal{M}(c)], \qquad s \in \mathcal{R}(x), \tag{4}$$

where $\lambda > 0$ is the score bonus assigned to Skills associated with c, and $\mathbb{I}$ is the indicator function. The retriever then selects the top-K Skill cards for execution:

$$\mathcal{S}_K(x, c) = \mathrm{TopK}_{s\in\mathcal{R}(x)}\, \widetilde{q}(x, s \mid c). \tag{5}$$

Talent–Skill associations thus provide a soft preference rather than a hard filter, allowing highly relevant Skills outside $\mathcal{M}(c)$ to be selected from the candidate pool. The policy model guides Skill

selection through its Talent choice, while the predefined retriever selects individual Skill cards under a fixed budget K.

3.3 TALENT ROUTING ACTION

TalentRL uses a shared policy π_θ for both routing and execution. Before interacting with the environment, the policy observes the task and Talent catalog and samples

$$c \sim \pi_\theta(\cdot \mid x, \mathcal{T}), \qquad c \in \mathcal{T}. \tag{6}$$

The policy model expresses this choice in its first response, as illustrated in the simplified example below.

Talent-selection response

Task: `Where was the author of the novel that inspired Blade Runner born?`

Model Response:

```
<think>Identify the source novel and its author before retrieving the
author's birthplace.</think>
<talent>talent_bridge_entity</talent>
```

Talent Guidance (g_c):
Resolve the bridge entity first, preserve relation direction, then issue a targeted follow-up query for the requested attribute.

The `<think>` field provides a task-grounded routing rationale, and the `<talent>` field specifies the selected Talent. Together, these fields constitute the routing action. The selected Talent's guidance then provides high-level instructions for task execution.

Once a valid Talent identifier is parsed, the system retrieves the Skill cards $\mathcal{S}_K(x, c)$ according to Equation 5. The policy model then generates each execution action conditioned on the interaction history, the selected Talent's guidance, and the retrieved Skill cards:

$$a_j \sim \pi_\theta(\cdot \mid h_{j-1}, g_c, \mathcal{S}_K(x, c)). \tag{7}$$

Thus a TalentRL trajectory is $\tau = (x, c, \mathcal{S}_K(x, c), a_1, o_1, \ldots, a_T, o_T)$. The expectation in Equation 3 includes both Talent selection and subsequent execution actions.

Nevertheless, the base model may struggle to select an appropriate Talent without explicit training. To provide a useful starting point for RL, we first train the policy model on demonstrations that pair beneficial Talent choices with successful task execution. The next section describes how we construct these demonstrations through outcome-guided comparisons.

3.4 TALENT-AWARE SFT DATA CONSTRUCTION

Before RL, we use routing-aware SFT to teach Talent selection while preserving the policy's task-execution capabilities. We construct the training data from paired rollouts that evaluate the utility of each Talent-conditioned context. For every task x and Talent $c \in \mathcal{T}$, we collect n trajectories $\tau_{c,r}$ conditioned on g_c and the Skill cards retrieved using Equation 5, where r indexes the rollouts. For comparison, we collect reference trajectories $\tau_{\varnothing,r}$ for the same task using similarity-based Skill retrieval, without Talent guidance or a Talent-specific retrieval prior. We define the utility of each Talent-conditioned rollout as

$$u_{c,r}(x) = R(\tau_{c,r}) - R(\tau_{\varnothing,r}). \tag{8}$$

To account for variability across rollouts, we evaluate each task–Talent pair using its mean utility:

$$\overline{u}_c(x) = \frac{1}{n} \sum_{r=1}^{n} u_{c,r}(x). \tag{9}$$

We call $\overline{u}_c(x)$ joint context utility since the selected Talent determines both the execution guidance and the retrieval prior. Among Talents with positive mean utility, we select $c^* = \arg\max_c \overline{u}_c(x)$. For the selected Talent c^*, we retain one normally completed rollout with valid actions and positive utility, prioritizing higher utility, fewer tool calls, and higher action validity.

Each retained rollout provides supervision for both Talent selection and task execution, including a routing rationale, a Talent identifier, and the corresponding execution actions. Individual Skill

Algorithm 1 TalentRL: matched-intervention routing and policy optimization

Require: Base policy π_0; environment $\mathcal{E}$; task sets $\mathcal{D}_{\mathrm{SFT}}$, $\mathcal{D}_{\mathrm{RL}}$; SkillRL replay data $\mathcal{D}_{\mathrm{replay}}$; Talent catalog $\mathcal{T}$; SkillBank $\mathcal{S}$; rollout count n; Skill budgets $0 < k < K$
Ensure: Trained policy π_θ
Phase I: Talent-aware SFT
$\mathcal{D}_{\mathrm{route}} \leftarrow \varnothing$
for each task $x \in \mathcal{D}_{\mathrm{SFT}}$ **do**
  Collect n reference rollouts $\tau_{\varnothing,r}$ with similarity-only Skills and no Talent guidance.
  For each $c \in \mathcal{T}$, collect n rollouts $\tau_{c,r}$ using $(g_c, \mathcal{S}_K(x, c))$.
  Compute all $\overline{u}_c(x)$ (Eq. 9); $c^* \leftarrow \arg\max_c \overline{u}_c(x)$.
  if $\overline{u}_{c^*}(x) > 0$ **then**
    Retain a completed, valid $\tau_{c^*,r}$ with $u_{c^*,r}(x) > 0$ (Section 3.4).
    Add its routing and execution targets to $\mathcal{D}_{\mathrm{route}}$.
Fine-tune π_0 on $\mathcal{D}_{\mathrm{route}} \cup \mathcal{D}_{\mathrm{replay}}$ using $\mathcal{L}_{\mathrm{SFT}}$ (Eq. 10) to obtain π_θ.
Initialize and freeze $\pi_{\mathrm{ref}} \leftarrow \pi_\theta$.
Phase II: Matched-Intervention RL
for each optimization iteration **do**
  for each task x in a batch sampled from $\mathcal{D}_{\mathrm{RL}}$ **do**
    Sample $c_{\mathrm{a}} \sim \pi_\theta(\cdot \mid x, \mathcal{T})$; collect τ_{a} using $(g_{c_{\mathrm{a}}}, \mathcal{S}_K(x, c_{\mathrm{a}}))$.
    Collect controls $\tau_{-s}, \tau_k, \tau_{\mathrm{np}}$ as in Section 4.1.
    Collect ordinary rollouts; form $\mathcal{P}$ from their indices and the anchor's.
    Compute all R_i; obtain $\hat{A}_i^{\mathrm{exec}}$ for $i \in \mathcal{P}$ (Eq. 11).
    Compute $\hat{A}^{\mathrm{talent}}$ (Eq. 14) and assign $A_{i,t}$ (Eq. 15); mask input tokens.
  Update π_θ using the masked, KL-regularized $\mathcal{L}_{\mathrm{RL}}$ (Eq. 16).
return π_θ

selection requires no supervision since the same Talent-conditioned retriever handles it during data collection, training, and inference. We mix these Talent-aware demonstrations with the original SkillRL (Xia et al., 2026) trajectories to preserve the policy model's general execution capabilities. This procedure changes both routing supervision and the quality of execution demonstrations.

Starting from the base model, we minimize the cross-entropy loss over response tokens:

$$\mathcal{L}_{\mathrm{SFT}} = -\sum_{t \in \mathcal{I}_{\mathrm{asst}}} \log \pi_\theta(y_t \mid \xi, y_{<t}), \tag{10}$$

where ξ denotes the input context for the current response, y_t is the target token at position t, $y_{<t}$ denotes the preceding tokens, and $\mathcal{I}_{\mathrm{asst}}$ is the set of token indices in the model's response. Tokens in the prompt and system-provided context are excluded from the loss. The resulting policy initializes the matched-intervention RL stage described in Section 4.

4 Reinforcement learning with token-specific intervention credit

Since an episodic return reflects both Talent routing and subsequent task execution, applying it uniformly to all policy tokens does not distinguish their respective contributions. To address this issue, TalentRL compares matched rollouts of the same task under different routing contexts and assigns separate credit to routing and execution tokens.

4.1 Paired rollouts and execution credit

For each task, we generate an anchor rollout τ_{a} using the policy-selected Talent c_{a}, its guidance $g_{c_{\mathrm{a}}}$, and the full Skill budget K. We construct three matched controls:

(a) **No-Skill control** τ_{-s} retains the same Talent and guidance but withholds all retrieved Skills.

(b) **Reduced-context control** τ_k, where $0 < k < K$, retains the same Talent, guidance, and retrieval ordering but supplies only the top-k Skills.

(c) **No-prior control** τ_{np} removes both Talent guidance and the Talent-specific retrieval bonus while retaining the full budget of similarity-retrieved Skills.

We collect these matched rollouts alongside ordinary, independently sampled policy rollouts for the same task. The matched controls vary distinct components of the routing context.

Let R_i denote the task return of rollout i. Let $\mathcal{P}$ denote the index set of the anchor and ordinary policy rollouts for the same task, excluding the matched control rollouts. Following GRPO (Shao et al., 2024), the execution advantage for each $i \in \mathcal{P}$ is

$$\hat{A}_i^{\text{exec}} = \frac{R_i - \mu_{\mathcal{P}}}{\sigma_{\mathcal{P}} + \epsilon}, \quad \mu_{\mathcal{P}} = \frac{1}{|\mathcal{P}|} \sum_{i' \in \mathcal{P}} R_{i'}, \quad \sigma_{\mathcal{P}} = \text{Std}_{i' \in \mathcal{P}}(R_{i'}). \tag{11}$$

Here, $\mu_{\mathcal{P}}$ and $\sigma_{\mathcal{P}}$ are the mean and standard deviation of returns in $\mathcal{P}$, and $\epsilon > 0$ is a numerical stability constant. The matched control rollouts instead provide reference returns for comparison with the anchor return to compute routing credit, as described in the next section.

4.2 Matched-intervention routing credit

Let R_{a}, R_{-s}, R_k, and R_{np} denote the task returns of the anchor, no-Skill, reduced-context, and no-prior rollouts, respectively. We define a dead-zone operator with threshold $\eta \geq 0$:

$$D_\eta(z) = \begin{cases} 0, & |z| < \eta, \\ z, & \text{otherwise} \end{cases}. \tag{12}$$

We first compute three unpenalized return contrasts:

$$\delta_{\text{skill}} = D_\eta(R_{\text{a}} - R_{-s}), \qquad \delta_{\text{budget}} = D_\eta(R_{\text{a}} - R_k), \qquad \delta_{\text{talent}} = D_\eta(R_{\text{a}} - R_{\text{np}}). \tag{13}$$

We combine these contrasts into a routing advantage:

$$\hat{A}^{\text{talent}} = \text{clip}(\alpha \left(w_s \delta_{\text{skill}} + w_k \delta_{\text{budget}} + w_t \delta_{\text{talent}}\right), -C, C), \tag{14}$$

where $w_s, w_k, w_t \geq 0$ and $w_s + w_k + w_t = 1$. The weights control the relative contributions of the three contrasts, while $\alpha > 0$ scales the combined signal and $C > 0$ sets its clipping bound.

The three contrasts measure the return differences associated with providing Skills (δ_{skill}), increasing the Skill budget (δ_{budget}), and jointly applying Talent guidance and its retrieval prior (δ_{talent}). Together, these contrasts provide task-specific feedback on the utility of the Talent-conditioned context to train the routing policy.

4.3 Token-level policy objective

We optimize the shared policy model by assigning routing and execution advantages to their corresponding response tokens. Let $m_{i,t}$ be the loss mask for token t in rollout i: it equals one for policy-generated tokens and zero for input context, including system-provided Talent guidance, Skills, and observations. We define the token-specific advantage as

$$A_{i,t} = \begin{cases} \hat{A}^{\text{talent}}, & i \text{ is the anchor and } t \text{ is a routing token}, \\ \hat{A}_i^{\text{exec}}, & i \in \mathcal{P} \text{ and } t \text{ is an execution token}, \\ 0, & \text{otherwise}. \end{cases} \tag{15}$$

Let $r_{i,t}(\theta)$ denote the token-level importance sampling ratio of the current policy to the behavior policy that generated the rollouts. We minimize the following KL-regularized clipped surrogate objective:

$$\mathcal{L}_{\text{RL}}(\theta) = -\frac{1}{N} \sum_{i,t} m_{i,t} \min(r_{i,t}(\theta) A_{i,t}, \text{clip}(r_{i,t}(\theta), 1 - \epsilon_c, 1 + \epsilon_c) A_{i,t}) + \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}). \tag{16}$$

Here, $N = \sum_{i,t} m_{i,t}$ counts the unmasked tokens, ϵ_c controls probability-ratio clipping, and β controls the strength of KL regularization toward the reference policy π_{ref}.

Routing and execution share policy parameters but receive phase-specific advantages. The routing rationale and Talent identifier receive the same routing advantage since together they form the routing action. Algorithm 1 summarizes the complete training procedure, from Talent-aware SFT to matched-intervention RL.

Table 1: SFT cold-start Search7 evaluation for Qwen2.5-7B and Qwen3-8B. Values are strict one-prediction-per-question exact match (%); the best result within each model is bold.

Model	Method	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Average
Qwen2.5-7B	Zero-shot	25.32	50.19	25.07	27.63	25.13	9.76	27.20	27.19
	SkillRL-SFT	34.07	52.13	32.87	32.91	23.71	12.62	32.80	31.59
	TalentRL-SFT	**36.29**	**53.67**	**34.26**	**35.30**	**27.32**	**14.60**	**41.60**	**34.72**
Qwen3-8B	Zero-shot	26.12	54.72	28.51	26.17	29.73	6.70	35.20	29.60
	SkillRL-SFT	37.23	55.03	34.05	35.84	27.70	**15.02**	**40.00**	34.98
	TalentRL-SFT	**39.39**	**56.52**	**35.52**	**36.35**	**33.06**	14.11	39.20	**36.31**

Table 2: Search7 success rate (%) after RL. Evaluation details are provided in Appendix A.

Model	Method	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Average
Qwen2.5-7B	CoSkill	44.24	61.39	38.09	37.87	39.09	14.81	39.20	39.24
	SAGE	46.43	**63.60**	39.30	41.67	**41.57**	16.92	42.40	41.70
	Skill1	46.15	62.66	38.70	39.41	39.44	15.52	38.40	40.04
	SKILL0	47.01	62.95	39.75	41.11	37.82	17.50	40.00	40.88
	SkillRL	**48.01**	63.24	40.70	**43.04**	36.90	17.38	44.00	41.90
	TalentRL	45.87	61.99	**41.61**	42.17	40.05	**18.41**	**48.80**	**42.70**
Qwen3-8B	CoSkill	44.32	62.66	39.30	39.55	41.12	15.76	42.40	40.73
	SAGE	45.62	61.96	41.32	41.39	44.18	16.76	44.00	42.18
	Skill1	46.68	62.62	40.86	43.05	41.81	18.54	45.60	42.74
	SKILL0	47.84	64.56	41.68	43.82	39.36	17.71	41.60	42.37
	SkillRL	47.53	64.16	41.24	43.98	39.03	**18.08**	**45.60**	42.80
	TalentRL	**48.03**	**64.68**	**42.13**	**45.17**	**43.73**	17.71	44.80	**43.75**

5 EXPERIMENTS

5.1 GENERAL SETUP

We use Qwen2.5-7B-Instruct (Yang et al., 2024a) and Qwen3-8B (Yang et al., 2025) as our backbone models. We construct the Talent-aware SFT corpus from the Search split released by SkillRL (Xia et al., 2026), combining outcome-filtered routing and execution demonstrations with the original SkillRL examples as replay (Section 3.4). We perform full-parameter SFT with response-only cross-entropy and initialize both the trainable policy and frozen reference policy from the resulting checkpoint. RL follows GRPO with the token-specific advantages defined in Section 4, using sampled same-task rollout groups to compute group-relative execution advantages.

We evaluate seven open-domain and multi-hop question-answering benchmarks: Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). We refer to this suite as Search7 and report exact match (%) with one prediction per question, averaged equally across the seven benchmarks. We refer to this metric as the Search7 success rate; reported gains are absolute differences in percentage points (pp), not relative percentage changes. We also evaluate interactive task performance on ALFWorld (Shridhar et al., 2021), which involves household tasks, and WebShop (Yao et al., 2022), which involves online shopping.

We consider the following baselines. (a) **CoSkill** jointly trains reasoning and meta-skill agents to evolve a hierarchical SkillBank (Feng et al., 2026). (b) **SkillRL** recursively evolves and retrieves procedural Skills during RL (Xia et al., 2026). (c) **SAGE** couples RL with a self-improving external Skill library (Wang et al., 2025a). (d) **Skill1** unifies Skill selection, utilization, and distillation in one policy (Shi et al., 2026). (e) **SKILL0** internalizes Skills into model parameters rather than retrieving explicit Skill cards (Lu et al., 2026). Further experimental details are provided in Appendix A.

Table 3: ALFWorld success rate and WebShop average score/success rate (%). Best and second-best results are bold and underlined, respectively.

Method	ALFWorld Success Rate (%)							WebShop	
	Pick	Look	Clean	Heat	Cool	Pick2	Overall	Score	Succ.
MemRL	62.8	38.5	22.2	12.5	8.0	0.0	21.4	29.5	9.2
EvolveR	64.9	33.3	46.4	13.3	33.3	33.3	43.8	42.5	17.6
Mem0+GRPO	78.1	54.8	56.1	31.0	<u>65.0</u>	26.9	54.7	58.1	37.5
SimpleMem+GRPO	89.5	36.3	60.0	50.0	64.9	26.3	62.5	67.8	46.9
SkillRL	<u>97.9</u>	<u>71.4</u>	<u>90.0</u>	<u>90.0</u>	**95.5**	<u>87.5</u>	<u>89.9</u>	<u>85.2</u>	<u>72.7</u>
TalentRL-SFT	94.3	**100.0**	77.8	87.5	64.0	79.2	82.9	82.9	66.4
TalentRL	**100.0**	**100.0**	**92.6**	**93.7**	**95.5**	**91.6**	**95.6**	**87.6**	**80.2**

5.2 Main results

Talent-aware SFT strengthens policy initialization. As shown in Table 1, TalentRL-SFT reaches 34.72% and 36.31% average Search7 success rates on Qwen2.5-7B and Qwen3-8B, respectively, outperforming the zero-shot models by +7.53 pp and +6.71 pp. Relative to SkillRL-SFT, TalentRL-SFT improves the average success rate by +3.13 pp on 7B and +1.33 pp on 8B, outperforming it on all seven benchmarks on 7B and five of seven on 8B. The largest improvements over SkillRL-SFT occur on Bamboogle for 7B (+8.80 pp) and 2Wiki for 8B (+5.36 pp), although 8B performance declines slightly on MuSiQue and Bamboogle. Overall, the combined routing and execution supervision provides a stronger policy initialization for subsequent RL.

RL improves performance on both backbones. As shown in Table 2, TalentRL achieves average Search7 success rates of 42.70% on Qwen2.5-7B and 43.75% on Qwen3-8B after RL, improving over the respective TalentRL-SFT initializations by +7.98 pp and +7.44 pp. Among the listed 7B methods, the demonstrated TalentRL result achieves the highest average, exceeding SkillRL (41.90%) by +0.80 pp and leading on PopQA, MuSiQue, and Bamboogle. The largest 7B gains over SkillRL occur on Bamboogle (+4.80 pp) and 2Wiki (+3.15 pp), although TalentRL does not lead on every benchmark. On 8B, TalentRL exceeds SkillRL (42.80%) and SKILL0 (42.37%) by +0.95 pp and +1.38 pp, respectively. Relative to SkillRL, the 8B gains are largest on 2Wiki (+4.70 pp) and HotpotQA (+1.19 pp), highlighting improvements on multi-hop question answering.

Interactive-task results. Table 3 presents ALFWorld and WebShop results. TalentRL-SFT records an ALFWorld success rate of 82.86% and a WebShop score of 82.95 with a success rate of 66.4%. After RL, TalentRL exceeds the reported SkillRL scores in five of six ALFWorld task categories, reaching 100.0% success on Pick and Look and matching SkillRL's 95.5% on Cool. Its category-level success rates remain above 90% across all six task types, with the largest gap over SkillRL on Look. On WebShop, TalentRL achieves a score of 87.6 and a success rate of 80.2%, compared with 85.2 and 72.7% for SkillRL, respectively. These correspond to differences of +2.4 in average score and +7.5 percentage points in success rate, with higher values on both reward and task-completion metrics.

6 Conclusion

We propose TalentRL, a strategy-guided reinforcement learning framework that learns high-level Talent routing and downstream Skill execution within a shared policy. Talents connect strategy selection to procedural memory by guiding execution and providing a soft prior for Skill retrieval. Talent-aware SFT initializes this routing-and-execution protocol, while matched-intervention contrasts and group-relative advantages provide distinct training signals for routing and execution, respectively. Experiments on Search7 show improved policy initialization and further gains through RL on both backbones, complemented by evaluations on ALFWorld and WebShop. Future work will explore jointly evolving the Talent taxonomy, SkillBank, and their associations, allowing high-level strategies and reusable procedures to adapt together.

Ethics Statement

TalentRL studies strategy-guided routing and execution for LLM agents on established benchmarks. Retrieved Skills may contain errors or biases inherited from the underlying models and data, and improved benchmark performance does not guarantee safe behavior in deployment. For agents that interact with files or external tools, deployment should include appropriate access controls, human oversight for consequential actions, and safeguards against unintended modifications or disclosure of sensitive information. Use of the models and datasets should follow their applicable licenses and terms.

AI Use Statement

Generative AI tools assisted with identifying potentially relevant literature and refining search queries for related work. They also helped improve the grammar, clarity, and readability of the manuscript. The authors take responsibility for the final content of the paper.

References

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2305.17126.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. MemP: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*, 2025. URL https://arxiv.org/abs/2508.06433.

Jinyuan Feng, Dongmin Li, Yiqun Chen, Yang Gao, Xing Chen, Huimu Wang, and Zhiqiang Pu. Coskill: Joint reinforcement learning of reasoning and meta-skill agents for hierarchical skill evolution. *arXiv preprint arXiv:2609.04865*, 2026. URL https://arxiv.org/abs/2609.04865.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. *arXiv preprint arXiv:2403.08978*, 2024. URL https://arxiv.org/abs/2403.08978.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *International Conference on Computational Linguistics*, 2020.

Jinchao Hu, Meizhi Zhong, Kehai Chen, and Min Zhang. Searchskill: Teaching llms to use search tools with evolving skill banks. *arXiv preprint arXiv:2605.09038*, 2026.

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. *arXiv preprint arXiv:2503.09516*, 2025. URL https://arxiv.org/abs/2503.09516.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Annual Meeting of the Association for Computational Linguistics*, 2017.

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.

Yu Li, Rui Miao, Zhengling Qi, and Tian Lan. ARISE: Agent reasoning with intrinsic skill evolution in hierarchical reinforcement learning. *arXiv preprint arXiv:2603.16060*, 2026.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. SKILL0: In-context agentic reinforcement learning for skill internalization. *arXiv preprint arXiv:2604.02268*, 2026.

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 *Annual Meeting of the Association for Computational Linguistics*, 2023.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-pro: Learning reusable skills from experience via non-parametric ppo for llm agents. *arXiv preprint arXiv:2602.01869*, 2026. URL https://arxiv.org/abs/2602.01869.

Yongliang Miao, Ziyang Yu, Liang Zhao, Bowen Zhu, and Hasibul Haque. Skilllens: Adaptive multi-granularity skill reuse for cost-efficient llm agents. *arXiv preprint arXiv:2605.08386*, 2026.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026. URL https://arxiv.org/abs/2603.25158.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP*, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi Gu, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning. *arXiv preprint arXiv:2605.06130*, 2026.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2023. URL https://arxiv.org/abs/2303.11366.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2309.02427.

Xijia Tao, Yihua Teng, Xinyu Fu, Ziru Liu, Kecheng Chen, Yuzhi Zhao, Suiyun Zhang, Rui Liu, and Lingpeng Kong. Softskill: Behavioral compression for contextual adaptation. *arXiv preprint arXiv:2606.20333*, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 2022.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. *arXiv preprint arXiv:2512.17102*, 2025a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *International Conference on Machine Learning*, 2025b. URL https://arxiv.org/abs/2409.07429.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026. URL https://arxiv.org/abs/2602.08234.

Ruiyao Xu, Tiankai Yang, and Wei-Chieh Huang. Hyperskill: Self-evolving llm agents via hypergraph-structured skill memory. *arXiv preprint arXiv:2608.16114*, 2026. URL https://arxiv.org/abs/2608.16114.

An Yang, Baosong Yang, Beichen Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. In *Advances in Neural Information Processing Systems*, 2024b. URL https://arxiv.org/abs/2406.04271.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026.

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*, 2018.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. *arXiv preprint arXiv:2207.01206*, 2022. URL https://arxiv.org/abs/2207.01206.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Zhiyuan Yao, Yuxin Chen, Zhengxi Lu, Zishan Xu, Yueqing Sun, Yifu Guo, Yuquan Lu, Zhengzhou Cai, Kangning Zhang, Zhuowen Han, Zi-Han Wang, Ziang Ye, Qi Gu, Xunliang Cai, Weiwen Liu, and Yongliang Shen. Skillrise: Agentic reinforcement learning for cross-task skill evolution. *arXiv preprint arXiv:2607.26784*, 2026. URL https://arxiv.org/abs/2607.26784.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi R. Fung, Hao Peng, and Heng Ji. CRAFT: Customizing LLMs by creating and retrieving from specialized toolsets. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.17428.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026. URL https://arxiv.org/abs/2602.02474.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *AAAI Conference on Artificial Intelligence*, 2024. URL https://arxiv.org/abs/2308.10144.

Xinyu Zhao, Zhen Tan, Vaishnav Tadiparthi, Nakul Agarwal, Kwonjoon Lee, Ehsan Moradi Pari, Hossein Nourkhiz Mahjoub, and Tianlong Chen. Generative skill composition for llm agents. *arXiv preprint arXiv:2606.32025*, 2026.

Yanzhao Zheng, Zhentao Zhang, Chao Ma, Yuanqiang Yu, Jihuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, and Gang Yu. Skillrouter: Skill routing for llm agents at scale. *arXiv preprint arXiv:2603.22455*, 2026.

Jiayin Zhu, Kelong Mao, Yudong Guo, Dengbo He, Sulong Xu, Simiu Gu, and Yutao Yue. Skill-coach: Self-evolving rubrics for evaluating and enhancing agentic skill-use. *arXiv preprint arXiv:2607.01874*, 2026.

A EXPERIMENTAL DETAILS

Table 4: Primary Qwen2.5-7B training, routing, and retrieval configuration.

Stage	Hyperparameter	Value
SFT	Epochs	2
SFT	Learning rate	1×10^{-5}
SFT	Effective batch size	16
SFT	Maximum sequence length	8,192
RL	Policy updates	200
RL	Learning rate	1×10^{-6}
RL	Task batch size	64
RL	Rollouts per task	8
RL	KL coefficient	0.01
RL	Policy-ratio clipping radius	0.2
RL	Maximum interaction steps	4
Routing	Reduced Skill budget k	2
Routing	Dead-zone threshold η	0.1
Routing	Skill-use penalty c_s	0.02
Routing	Advantage scale α	2.0
Routing	Advantage clipping bound C	2.0
Retrieval	Embedding model	Qwen3-Embedding-0.6B
Retrieval	Similarity recall size	20
Retrieval	Execution Skill budget	5
Retrieval	Similarity threshold	0.2
Retrieval	Talent-prior weight λ	0.1

Training rollouts are sampled to produce within-task diversity for group-relative optimization; zero-variance groups receive an unnormalized zero-centered advantage. Final Search7 evaluation uses deterministic decoding. Skill and query embeddings are L2-normalized, so $q(x, s)$ is cosine similarity. The three routing contrasts are equally weighted in the mixed variant. The full generated Talent response receives the routing advantage.

Search7 baseline configurations. The 7B CoSkill, SAGE, and Skill1 results in Table 2 use our Search7 adaptations aligned with the official implementations and trained for 200 steps. The 7B SKILL0 result uses a single step-200 run, with external Skill context annealed to zero at final inference.

Interactive-task baseline results. Baseline values in Table 3 are reported by SkillRL (Xia et al., 2026). Highlighting includes all rows: the highest and second-highest distinct values for each metric are bold and underlined, respectively, with ties receiving the same formatting.

Interactive-task evaluation protocols. The TalentRL-SFT evaluations use Qwen2.5-7B without interactive RL. On ALFWorld, evaluation uses 140 held-out `valid_seen` tasks, fixed category-assigned Skills, and state-feedback prompts without learned Talent routing. The limits are 50 steps, an 8,192-token context, and 2,048 generated tokens per action.

On WebShop, evaluation uses the step-112 SFT checkpoint with learned Talent routing and 500 goals in a fixed seed-42 ordering within the 1,000-product environment, using a 15-step limit, an 8,192-token context, up to 512 generated tokens per response, and temperature 0.4. The denominator includes seven invalid-routing failures and two exact overlaps with SFT training tasks.

B TALENT TAXONOMY

The current search instantiation uses eight strategy classes: task decomposition, bridge-entity resolution, entity disambiguation, constraint application, evidence comparison, long-tail query expansion, temporal and numeric reasoning, and answer verification. Each Talent contains a natural-language description, generation guidance, and references to a subset of reusable Skill cards.

Table 5: Talent-specific generation guidance and preset Talent–Skill associations for Search7. The selected Talent provides natural-language guidance for execution and a soft prior for Skill retrieval. Associated cards are not necessarily all included in the execution context. Table 6 provides the referenced card details.

Talent	Generation guidance	Associated Skills
Decompose the Task	Identify the dependency chain before searching, solve one subproblem at a time, and carry forward only verified intermediate facts.	gen_001, mul_001, mul_002, gen_007, gen_010
Resolve a Bridge Entity	Resolve the bridge entity first, preserve relation direction, then issue a targeted follow-up query for the requested attribute.	gen_007, mul_001, mul_002, gen_004, gen_010
Disambiguate the Entity	Use type, date, location, occupation, or related-work qualifiers and reject evidence that belongs to another referent.	gen_006, mul_004, ent_002, ent_003, gen_005
Apply All Constraints	Translate every clause into a checkable constraint, narrow candidates by intersection, and verify all constraints before answering.	com_004, mul_001, mul_002, gen_005, gen_010
Compare Evidence	Collect the same attribute for every entity, normalize dates or units, and perform the comparison only after all values are verified.	gen_008, mul_003, com_001, com_002, com_003, com_005, gen_004
Expand a Long-Tail Search	When direct retrieval is weak, try aliases, related works, distinctive phrases, and progressively refined contextual queries.	dir_002, dir_003, ent_004, gen_003, mul_005, gen_006
Resolve Time and Numbers	Retrieve authoritative dated or numeric evidence, normalize formats and units, and explicitly compute or compare the required values.	gen_009, dir_004, gen_008, com_003, com_005, gen_005
Verify Before Answering	Do not guess from memory or a weak snippet; obtain explicit supporting evidence, cross-check critical facts, and stop once the answer is solid.	gen_004, gen_005, gen_010, dir_005, ent_003

B.1 Preset Talent-to-Skill assignments

Table 5 reports the Talent-specific generation guidance and initial Talent–Skill associations. The guidance conditions execution, while the associations supply a soft prior rather than a hard retrieval boundary. The retriever recalls up to 20 globally relevant candidates, adds a prior bonus of 0.1 to Skills associated with the policy-selected Talent, and supplies the top five reranked cards during execution. The retriever uses a similarity threshold of 0.2 and no graph-expansion slots. Skill cards and associations remain fixed during the reported RL runs.

B.2 SKILL CARD DETAILS

Table 6 lists the principles and applicability conditions of the 26 distinct Skill cards referenced in Table 5. Each card is listed once, even when it is associated with multiple Talents.

Table 6: Details of the preset SkillBank cards referenced by the Search7 Talent catalog. Principles and applicability conditions reproduce the corresponding card fields; only typography is normalized.

Skill ID	Principle	When to apply
gen_001	Break the question into minimal sub-questions (entities, attributes, relations) and handle each with its own query before synthesizing the final answer.	Any complex or multi-part question, especially multi-hop or comparative tasks.
gen_003	If top results don't yield direct evidence, rephrase by adding qualifiers (dates, roles, context) or alternate names instead of repeating the same query.	After scanning first result set and finding no definitive evidence.
gen_004	Only state an answer once the supporting evidence (reliable source snippet) has been located and read; refrain from guessing or summarizing from memory.	Before committing to a final answer, especially for contentious or less-known facts.
gen_005	Validate critical facts (dates, numbers, names) against at least two independent sources to avoid outdated or incorrect information.	When the fact may have changed over time or when initial sources disagree.
gen_006	Detect ambiguous entity mentions and disambiguate by adding clarifiers (e.g., occupation, year, nationality) in the query or by exploring disambiguation pages.	When an entity name maps to multiple possible referents or results appear mixed.
gen_007	For multi-hop tasks, first retrieve the intermediate entity (e.g., a film's director) then run a second targeted query for that entity's attribute (e.g., birthplace) before synthesizing.	Whenever the answer depends on an attribute of a related entity rather than the entity in the question.
gen_008	When comparing entities (earlier/later, larger/smaller), collect each relevant value separately, list them side by side, then decide using explicit comparison logic.	Any comparative question involving dates, counts, or quantitative attributes.
gen_009	For time-sensitive queries (e.g., upcoming releases, current versions), include temporal cues like "latest", year, or "release date 2023" and prioritize most recent authoritative sources.	Questions asking for "current", "next", or "latest" information.
gen_010	Stop issuing further queries once you have clear, corroborated evidence; conversely, avoid premature termination if no source explicitly supports an answer.	After each read step—decide to answer only if confidence is justified; otherwise refine search.
dir_002	If first search yields weak/no hits, instantly reformulate using synonyms, alternate names, dates, or quoted phrases instead of repeating the same query.	After an initial search returns no clear answer or only tangential results.
dir_003	For song titles, quotes, episode names, etc., wrap the unique phrase in quotation marks to pull exact-match sources.	When the query contains distinctive phrases, lyrics, book/film titles, or direct quotations.
dir_004	Include recency cues (e.g., "current", year) in the search and verify publication date to avoid outdated or speculative info.	For questions about 'current', 'latest', or future events/releases.
dir_005	Only state an answer that is explicitly supported by the retrieved text; if unclear, continue searching rather than guess or hallucinate.	Before finalizing any factoid answer.

Continued on the next page

Table 6 continued from the previous page

Skill ID	Principle	When to apply
mul_001	Split the main query into explicit sub-questions to identify each entity and the specific attribute or relationship needed before searching.	Any multi-hop question that links two or more entities/facts (e.g., director's birthplace, album vs film comparisons).
mul_002	Issue separate, focused searches for each sub-question or entity instead of one broad query to gather precise intermediate facts.	After decomposition, when distinct pieces of information must be collected individually.
mul_003	Retrieve concrete values for all items involved (dates, places, relations) before performing any comparison or conclusion to avoid premature or unsupported answers.	Comparative tasks (earlier/later, older/younger, bigger/smaller, etc.).
mul_004	Add clarifying descriptors (profession, year, medium, location) to queries to distinguish between entities with identical or similar names and prevent mixing facts.	Whenever an entity name is generic or shared by multiple subjects (e.g., common personal names, titles appearing in different media).
mul_005	If initial search fails or returns ambiguous results, promptly rephrase using synonyms, alternate titles, or additional context instead of repeating the same query.	After a search yields no relevant results or conflicting information.
ent_002	If the name is common or ambiguous, append clarifiers (birth year, nationality, known work) to the query or scan result snippets for matching context before selecting a source.	When multiple people/characters share the same name or early results mix different entities.
ent_003	Confirm the attribute in at least two independent, reputable sources (e.g., Wikipedia infobox + biography site) to avoid hallucinations.	After the first plausible answer appears or when the attribute seems uncommon or uncertain.
ent_004	If initial search returns irrelevant hits, adjust the query instead of repeating it—use alternative spellings, middle initials, or related works/titles.	When consecutive top results don't mention the sought attribute for the intended entity.
com_001	First split the question into (a) each entity and (b) the single attribute to be compared; this ensures every subsequent search is targeted and comparable.	At the moment you read any comparison-type question.
com_002	Independently retrieve the identical attribute for each entity—run separate, attribute-focused searches (e.g., "[Entity] headquarters", "[Entity] release date") and store the raw values.	Immediately after identifying entities and the comparison attribute.
com_003	Convert retrieved values to a common, directly comparable form (e.g., standardize dates, compute ages, map demonyms to countries) before judging equality or ordering.	After gathering each entity's attribute but before drawing any conclusion.
com_004	If a search yields no clear or multiple results, refine the query with context (profession, year, language) or acknowledge data insufficiency—never infer without evidence.	Whenever an attribute lookup returns ambiguous, missing, or conflicting information.
com_005	State each normalized value and perform the logical comparison (equal, earlier, larger, etc.) explicitly to derive the answer, ensuring traceability and avoiding hallucination.	As the final synthesis step before outputting the comparison result.

C SFT DATA CONSTRUCTION

We start from the SkillRL Search SFT split, group its step-level rows by question, and reconstruct task trajectories. For each task, repeated rollouts compare every forced Talent with a no-prior reference. We compute the mean reward improvement for each task–Talent pair and retain only Talents whose mean improvement is positive. We select the Talent with the largest mean improvement, breaking ties by mean conditioned score, positive-outcome rate, fewer tool calls, and a deterministic task–Talent hash. We then choose one normally completed, valid, positive-utility trajectory for that Talent, prioritizing reward improvement, fewer tool calls, and action validity.

Each accepted paired trajectory yields a Talent-selection row and one ReAct row for every search or answer action. The routing target contains a task-grounded rationale and the selected Talent identifier. Skills are selected by the same Talent-conditioned similarity retriever used at inference and therefore require no policy target. The ReAct targets preserve the accepted Talent-conditioned rollout rather than the original SkillRL actions. Tasks without a positive Talent do not produce routing-aware rows. Original SkillRL rows for the source tasks are retained as replay, and the two sources are concatenated without enforcing a fixed ratio.

Training and validation are separated by a deterministic task hash, ensuring that no task appears in both partitions. The final Search7 evaluation sets are not used to construct routing targets. We initialize from the base instruction model and run two epochs of full-parameter, response-only SFT with a low learning rate. Prompt tokens and injected Skill context are excluded from the loss. Replay rows retain the original search protocol without adding a Talent-selection target.

D MATCHED-INTERVENTION RL DETAILS

The RL source row contains the task prompt and benchmark metadata. For each task group, the system creates a routing anchor, matched zero-Skill, two-Skill, and no-prior controls, and ordinary policy trajectories. Retrieved observations, Talent guidance, and injected Skill cards are environment context and are masked out of the policy loss.

D.1 POLICY-ROUTED AND CONTROL TRAJECTORIES

The anchor uses the policy-selected Talent, its guidance, and five Talent-conditioned Skill cards. The zero-Skill control retains the Talent and guidance but withholds Skill cards. The two-Skill control retains the same Talent, guidance, and retrieval ordering while supplying only the top two cards. The no-prior control removes Talent guidance and the retrieval bonus, supplying five cards retrieved by similarity alone. The matched controls are retained for routing-credit estimation, but their Talent tokens are excluded from policy credit. The anchor Talent response receives the combined intervention advantage. Execution tokens receive the group-relative advantage computed from the anchor and ordinary policy trajectories only.

D.2 REWARD AND CREDIT DETAILS

The active runs use only the task score; all execution, balance, label, and gate-shaping weights are zero. Execution responses receive the normalized group-relative advantage from non-intervention trajectories. Talent routing credit averages three dead-zoned reward differences: the anchor's gains over the zero-Skill, two-Skill, and no-prior controls. In the reported runs, we replace the unpenalized Skill contrast by $\delta_{\text{skill}} - c_s$ before weighting, where $c_s \geq 0$ is a fixed Skill-use penalty rather than a measured computation cost. The combined advantage is clipped before being applied to the anchor Talent response. We use $\eta = 0.1$, $c_s = 0.02$, equal contrast weights, $\alpha = 2.0$, and $C = 2.0$. The intervention contrasts are auxiliary routing-utility signals rather than unbiased estimators of the original episodic-return advantage.