SEARCHSKILL: TEACHING LLMS TO USE SEARCH TOOLS WITH EVOLVING SKILL BANKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Teaching language models to use search tools is not only a question of whether they search, but also of whether they issue good queries. This is especially important in open-domain question answering, where broad or copied queries often waste retrieval budget and derail later reasoning. We propose SEARCHSKILL, a framework that makes query planning explicit through reusable search skills. At each step, the model first selects a skill, then generates a search or answer action conditioned on the selected skill card. The skill inventory itself is not fixed: SearchSkill maintains an evolving SkillBank, expands or refines it from recurrent failure patterns, and reconstructs affected trajectories before supervised training. The resulting two-stage SFT recipe aligns training with the inference-time protocol of skill selection followed by skill-grounded execution. Across open-source and closed-source models, SearchSkill improves exact match on knowledge-intensive QA benchmarks and yields better retrieval behavior, including fewer copied first queries, more atomic hop-focused queries, and more correct answers within a small search budget. These results suggest that explicit skill-conditioned query planning is a lightweight alternative to treating search as an undifferentiated action. All training code and configurations are available here.

1 INTRODUCTION

Large language models (LLMs) exhibit strong reasoning ability and broad parametric knowledge, but that knowledge remains fundamentally static: it can be outdated, incomplete, or simply missing the long-tail facts required by knowledge-intensive tasks. This limitation is especially visible in open-domain and multi-hop question answering, where solving a problem often requires gathering fresh evidence, following hidden bridge entities, and verifying intermediate conclusions against external sources. Search tools therefore play an indispensable role in extending the knowledge boundary of LLMs and turning them into reliable systems for real-world information seeking.

Recent years have seen rapid progress on retrieval-augmented and tool-using LLMs. Retrieval-augmented generation (RAG) augments generation with external passages for knowledge-intensive prediction (Lewis et al., 2020). Tool-use methods such as Toolformer (Schick et al., 2023) and ReAct (Yao et al., 2022) further enable models to interleave reasoning with external actions. More recent search-centered systems train this capability more directly: Search-R1 (Jin et al., 2025) optimizes long-horizon search behavior with reinforcement learning, while ZeroSearch (Sun et al., 2025) incentivizes search capability without relying on live search during training. These advances have substantially improved LLM–retriever interaction, but they also expose a practical bottleneck.

Challenge 1: most existing methods teach the model to search, but devote much less modeling capacity to how to formulate high-quality search queries. In multi-hop settings, poor queries lead to longer interaction traces, redundant tool calls, noisy evidence, and lower final answer accuracy. The issue is often not whether a model uses a retriever, but whether it does so intelligently.

Recent work on agent skills offers a complementary angle. Anthropic's Agent Skills package instructions, scripts, and resources into reusable modules that agents can discover and load on demand (Zhang et al., 2025). Analyses of skill ecosystems show that skills rapidly became a practical mechanism for extending model functionality, while also raising ecosystem-level questions about organization and safe reuse (Ling et al., 2026; Liu et al., 2026). AgentSkillOS studies selection and benchmarking over large skill ecosystems (Li et al., 2026), while Reinforcement Learning for

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.

Self-Improving Agent with Skill Library, MemSkill, and SkillRL explore how agents can maintain and evolve skill libraries or skill banks over training (Wang et al., 2025a; Zhang et al., 2026; Xia et al., 2026). However, applying this intuition to search tool use is still non-trivial.

Challenge 2: current skill frameworks do not directly tell us how skills should be represented, invoked, and updated inside search trajectories. A useful search skill must shape not only a high-level strategy, but also the next query, the arguments passed to the retriever, the verification behavior after retrieval, and the stopping decision before answering.

To address these challenges, we propose SEARCHSKILL, a framework that teaches LLMs to use search tools through an evolving bank of reusable search skills. Each interaction follows *select–read–act*: the model selects skill ids, reads their procedural cards, and emits an explicit skill declaration followed by a search query or final answer. Retrieved evidence updates the context for the next decision. This token-level interface makes skill selection and execution explicit, trainable parts of the model's search trajectory. The cards describe reusable operations such as resolving a bridge entity, comparing attributes, checking conflicting evidence, and extracting a grounded answer.

The same interface connects skill discovery to policy training. A skill generator proposes and revises cards using retrieval experience, retaining procedures supported by successful rollouts. The updated bank guides trajectory reconstruction, and two-stage SFT teaches the policy to select and execute its procedures. The target backbone or a stronger model can serve as the skill generator. With Qwen-generated skills, the 7B policy reaches 38.60 macro EM, comparable to Search-R1's 38.68, while the 3B policy reaches 34.87, as reported in Table 1. Using GPT-5.4 as the skill generator yields the best 7B and 3B scores of 42.34 and 36.94.

Our main contributions are as follows:

- We propose a skill-conditioned search-tool interface that improves query formulation by making the model choose a reusable search strategy before each tool call, addressing the common failure of issuing broad or copied multi-hop queries.
- We introduce an evolvable Search SkillBank that is updated through failure analysis, add/refine operations, and trajectory reconstruction, providing a reusable tool-use prior that can extend beyond the QA setting.
- We show that Qwen-generated skills and two-stage SFT yield performance comparable to RL-trained search agents, suggesting that self-directed skill discovery can elicit the model's latent capabilities and enable more efficient and accurate search.
- We demonstrate effectiveness on both open-source and closed-source models: fine-tuned Qwen2.5 3B/7B models improve QA performance, and the frozen SkillBank also benefits proprietary models as an external planning guide.

2 PRELIMINARIES

2.1 AGENTIC SEARCH

Agentic search is a sequential decision process in which a language model interleaves reasoning with search and uses retrieved information to guide subsequent actions (Yao et al., 2022; Jin et al., 2025).

Formally, given a question q, a policy π_θ chooses an action a_t conditioned on the interaction history h_t, which records q, prior actions, and retrieved passages. A search action issues a query u_t and receives passages $e_t = \mathcal{R}(u_t)$ from a retriever $\mathcal{R}$; an answer action returns y and ends the interaction. After a search, h_{t+1} includes the query and returned passages, allowing subsequent actions to depend on the newly retrieved information.

The objective is to answer correctly within a retrieval budget T, where each search consumes one unit. For multi-hop questions, retrieved passages may establish intermediate facts without resolving the entire question. The policy must therefore track resolved facts and remaining information needs, using earlier findings to determine subsequent retrieval targets. It must also decide when the available information is sufficient to answer, since redundant or uninformative queries consume the budget.

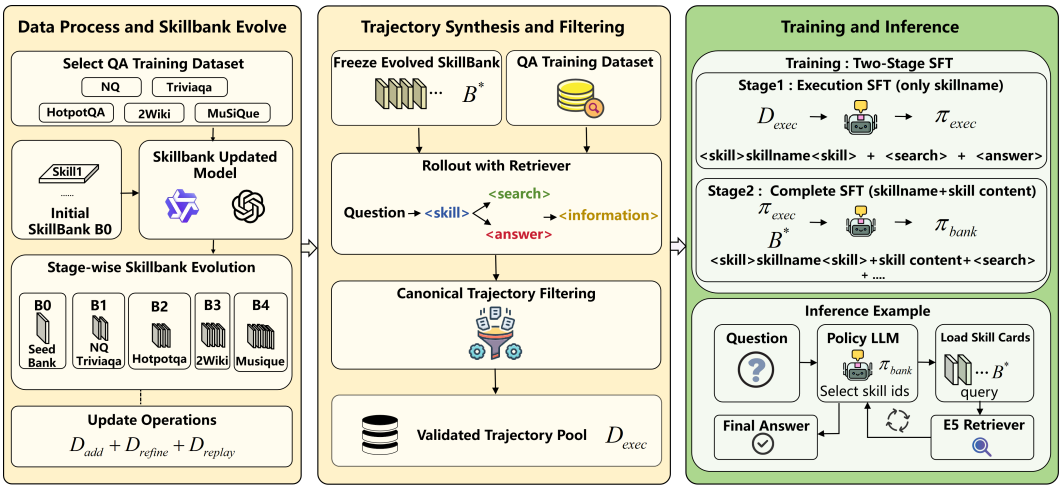

Figure 1: **Overview of SEARCHSKILL.** A skill generator builds and refines the SkillBank through retrieval feedback. The evolved bank guides trajectory construction, and two-stage SFT teaches skill execution and skill selection. Given a question, the trained policy selects skills, reads their cards, and gathers evidence through retrieval before producing a final answer.

2.2 SEARCH SKILLS AND THE SKILLBANK

A skill is a reusable procedure specifying when and how to perform an operation (Zheng et al., 2025; Wang et al., 2025a; Zhang et al., 2026; Xia et al., 2026). We represent a *search skill* by an identifier i and a natural-language card c_i. The card states the conditions for applying the skill, its procedure for querying and using retrieved information, the intermediate facts to establish, and when to continue searching or produce an answer.

A *SkillBank* $B = \{(i, c_i)\}_{i \in \mathcal{I}_B}$ is a collection of such skills, where $\mathcal{I}_B$ denotes the available identifiers. Its compact index $\mathrm{Index}(B)$ summarizes the skills for selection, while the corresponding cards provide detailed instructions for execution. This representation allows a search procedure to be reused across questions.

2.3 SKILL-CONDITIONED INTERACTION

SearchSkill organizes each decision through a *select–read–act* interface, illustrated by the inference example in Figure 1. Given h_t, the policy selects one or more skill identifiers $s_t \subseteq \mathcal{I}_B$ from $\mathrm{Index}(B)$. The system loads their cards $C_B(s_t)$, and the policy uses these instructions to generate a search or answer action a_t. The joint probability factorizes as

$$p_\theta(s_t, a_t \mid h_t, B) = p_\theta(s_t \mid h_t, \mathrm{Index}(B))\, p_\theta(a_t \mid h_t, s_t, C_B(s_t)). \tag{1}$$

The first factor models skill selection; the second models action generation conditioned on the selected cards. Search actions and retrieved passages update the history for the next selection; an answer ends the interaction. Section 3 describes how the bank and policy are learned; Appendix A provides additional related work.

3 METHODOLOGY

As shown in Figure 1, SearchSkill first evolves a SkillBank from retrieval experience, then constructs skill-guided trajectories and trains the policy in two supervised stages. Bank evolution updates the cards while model weights remain fixed; SFT updates the policy while the final bank remains fixed.

3.1 STAGE-WISE SKILLBANK EVOLUTION

We begin from a compact seed bank B_0 containing six general retrieval and grounding skills, and expand it through four rounds: B_1 from NQ and TriviaQA, B_2 from HotpotQA, B_3 from

2WikiMultiHopQA, and B_4 from MuSiQue. This ordering establishes direct lookup before extending the bank to bridge resolution, comparison, and longer reasoning chains.

Construct candidate skills. In each round, we group training questions using available dataset metadata and model-based grouping to preserve coverage. Representative question–answer pairs and the current bank are provided to a skill generator $\mathcal{G}$. The generator identifies recurring search procedures that existing cards do not cover, then proposes new cards or refines existing ones. Skill identifiers and procedures are generated from the examples; the grouping labels serve only to organize coverage. Each card specifies an observable trigger, retrieval steps, facts to establish, failure modes to avoid, and conditions for continuing or answering. This yields a candidate bank.

Test and refine. The generator uses the candidate cards to search and answer training questions. Reference answers are withheld from the rollout context and used to score its final answer. Exact match identifies successful trajectories; the skills used in them are retained and protected. Protection preserves their identifiers and core procedures in later updates. Aggregate outcomes and representative success and failure traces are returned to $\mathcal{G}$, which can revise, merge, or remove unprotected cards. The revised bank enters the next round, carrying forward previously validated skills. After four rounds, we fix $B^\star = B_4$ for trajectory construction and training. Grouping procedures, card formats, and update details appear in Appendices B and D.2.

3.2 Skill-guided trajectory construction

With $B^\star$ fixed, we sample questions across the five training datasets and generate trajectories through interaction with the retriever. At each step, the rollout model predicts a primary skill, optional supporting skills, a search or answer action, and a checkpoint describing the resolved intermediate state. The retriever executes search queries, and returned passages are appended to the history until the model produces a final answer.

We keep only canonical trajectories that satisfy answer correctness, at least one retrieval step, legal skill identifiers, and consistency between the declared skill and executed action. These form the trajectory set $\mathcal{D}_{\text{exec}}$, with explicit `<skill>` declarations before `<search>` or `<answer>` actions. Appendix D.3 details data construction.

3.3 Two-stage supervised fine-tuning

Stage I: skill execution. The first stage starts from the target backbone and teaches the basic Search-Skill action format: each assistant turn emits `<skill>` followed by `<search>` or `<answer>`. For a trajectory τ, let h_t be the context before action turn a_t, and $\mathcal{M}(\tau)$ index the supervised steps. We optimize

$$\mathcal{L}_{\text{exec}}(\theta) = -\sum_{\tau \in \mathcal{D}_{\text{exec}}} \sum_{t \in \mathcal{M}(\tau)} \lambda(a_t) \sum_{k=1}^{|a_t|} \log p_\theta(a_{t,k} \mid h_t, a_{t,<k}), \tag{2}$$

where λ weights the targets. Full-trajectory records supervise all assistant turns; closure-focused records retain the preceding interaction as context but supervise only the final answer. This stage teaches skill-tagged query planning, evidence use, and stopping behavior.

Stage II: skill selection. The second stage continues from the first-stage checkpoint and repacks the same trajectory supervision into the select–read–act protocol. Each original action is rewritten as a selection turn s_t, a user message containing the selected cards $C_{B^\star}(s_t)$, and the original action turn a_t. The policy selects from the bank index without a question-specific recommended-skill hint. Let $\mathcal{D}_{\text{bank}} = \mathcal{R}_{B^\star}(\mathcal{D}_{\text{exec}})$ denote the rewritten dataset, h_t^{sel} the context before selection, and $h_t^{\text{act}} = h_t^{\text{sel}} \oplus s_t \oplus C_{B^\star}(s_t)$ the context after card insertion, where $\oplus$ concatenates messages. We optimize

$$\begin{aligned}\mathcal{L}_{\text{bank}}(\theta) = -\sum_{\tau' \in \mathcal{D}_{\text{bank}}} \sum_{t \in \mathcal{M}(\tau')} \Bigg[& \lambda(s_t) \sum_{k=1}^{|s_t|} \log p_\theta(s_{t,k} \mid h_t^{\text{sel}}, s_{t,<k}) \\ & + \lambda(a_t) \sum_{k=1}^{|a_t|} \log p_\theta(a_{t,k} \mid h_t^{\text{act}}, a_{t,<k}) \Bigg].\end{aligned} \tag{3}$$

Here λ is zero for unsupervised targets; supplied cards and retrieved passages are context only. Thus the first objective teaches execution, while the second teaches selection and card-conditioned execution under the inference-time protocol. At inference, both the policy and bank are fixed. Algorithm 1 gives the full interaction loop, and Appendix D.4 lists training settings.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate on seven QA benchmarks: Natural Questions (NQ), TriviaQA, and PopQA for single-hop retrieval (Kwiatkowski et al., 2019; Joshi et al., 2017; Mallen et al., 2023); HotpotQA, 2WikiMultiHopQA, MuSiQue, and Bamboogle for multi-hop reasoning (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022; Press et al., 2023). Together, they test factual lookup and the ability to decompose questions requiring multiple retrieval steps.

Baselines. We compare with direct inference, chain-of-thought prompting (Wei et al., 2022), RAG (Lewis et al., 2020), and Search-o1 (Li et al., 2025), as well as RL-based search agents: ZeroSearch (Sun et al., 2025), Search-R1 (Jin et al., 2025), StepSearch (Wang et al., 2025b), SmartSearcher (R1-Searcher++) (Song et al., 2025b), and s3 (Jiang et al., 2025b). These baselines compare language-only reasoning, retrieval access, and methods trained to search.

Models and training. We use Qwen2.5-7B and Qwen2.5-3B (Qwen Team, 2025), with base and instruct variants, to examine both model scale and initialization. The labels *7B skills*, *3B skills*, and *GPT-5.4 skills* identify the model that evolves the SkillBank and generates training trajectories through the workflow in Section 3. Each policy is trained with two-stage SFT and evaluated with its frozen bank.

Training questions come from the training splits of NQ, TriviaQA, HotpotQA, 2WikiMultiHopQA, and MuSiQue. Sampling preserves coverage across question types and retrieval demands. Successful trajectories provide SFT targets, while failed trajectories guide skill revision and are excluded from the SFT targets. Appendices B and D give data construction and training details.

Evaluation. Search-based methods share an E5 retriever (Wang et al., 2022) returning the top-3 passages per query, a five-search budget, and greedy decoding. Retrieved passages are appended to the interaction history for subsequent decisions. We report normalized exact match (EM) as a percentage; *Avg.* is the unweighted mean over seven datasets. Our main SEARCHSKILL results use SFT alone; post-SFT GRPO is analyzed separately in Section 4.9.

4.2 MAIN RESULTS

Self-generated skills elicit competitive search capabilities. Table 1 shows that SEARCHSKILL with Qwen-generated skills reaches 38.60 macro EM on 7B, comparable to Search-R1 at 38.68 and exceeding StepSearch at 38.13. The 3B model also reaches 34.87, surpassing Search-R1-Instruct by 1.73 points. These results highlight the potential of self-directed skill discovery: by discovering and refining its own search procedures and learning to apply them through two-stage SFT, a 7B model can develop search capabilities competitive with those obtained through large-scale reinforcement learning.

Stronger skill generators further improve performance. With GPT-5.4 as the skill generator, SEARCHSKILL obtains the best macro averages in both blocks, reaching 42.34 on 7B and 36.94 on 3B. Compared with the strongest RL baseline in each block, the average gains are 2.66 and 3.80 points. **Search skills yield substantial gains on multi-hop QA.** For example, on 2Wiki, where the model must decompose relations and carry intermediate entities across searches, 7B-Base and 3B-Instruct improve over their corresponding Search-R1 variants by 6.50 and 10.50 points, respectively. **Single-hop QA is not sacrificed for this gain**: with only SkillBank-guided SFT, SEARCHSKILL remains on par with or better than RL-trained search agents on factual lookup benchmarks, while avoiding their large online rollout cost. **The effect is stable across scale and initialization**: both 7B and 3B blocks reach the best macro average, and the gains appear for base and instruct variants, indicating that the evolved SkillBank provides a reusable search-control prior rather than a model-specific prompt advantage. These results suggest that stronger skill generators offer a practical route to strong search performance without expensive, time-consuming training.

Table 1: Exact match (%) on seven QA benchmarks; Avg. is their unweighted mean. Per model scale, best results are bold and second best underlined. Parentheses identify the skill generator.

Method	Single-Hop QA			Multi-Hop QA				Avg.
	NQ	Trivia	Pop	Hotpot	2Wiki	MuSiQ.	Bamb.	
Qwen2.5-7B-Base/Instruct								
Direct Inference	16.79	38.98	15.36	17.00	26.00	1.00	8.00	17.59
CoT	12.19	34.03	16.11	17.00	18.50	5.00	22.40	17.89
RAG	38.00	58.01	39.10	33.50	23.00	7.00	8.80	29.63
Search-o1	30.94	57.20	36.02	31.50	27.50	11.00	36.80	32.99
ZeroSearch-Base	36.70	58.90	47.30	29.00	31.00	7.50	22.40	33.26
SearchR1-Base (GRPO)	39.50	56.00	38.80	35.50	40.50	9.00	37.60	36.70
SearchR1-Instruct (GRPO)	39.66	60.51	41.09	40.00	37.50	**16.00**	36.00	38.68
StepSearch	33.21	52.49	45.90	37.00	40.00	13.50	**44.80**	38.13
Smart-Searcher (R1-Searcher++)	38.42	59.65	47.20	37.50	37.50	13.50	44.00	39.68
s3	6.65	30.08	17.10	21.50	14.50	4.00	11.20	15.00
SEARCHSKILL-Instruct (7B skills)	37.60	60.00	47.40	39.50	40.50	14.00	31.20	38.60
SEARCHSKILL-Base (GPT-5.4 skills)	40.89	60.01	**52.80**	42.00	**47.00**	14.50	39.20	**42.34**
SEARCHSKILL-Instruct (GPT-5.4 skills)	**41.60**	**61.04**	51.40	**43.00**	44.50	15.50	38.40	42.21
Qwen2.5-3B-Base/Instruct								
Direct Inference	11.88	33.31	11.64	14.50	27.00	2.00	4.00	14.90
CoT	11.33	33.07	11.90	15.00	15.50	2.50	20.80	15.73
RAG	33.16	53.24	38.30	27.50	26.00	3.00	6.40	26.80
Search-o1	28.86	47.49	31.29	24.50	24.00	6.00	20.80	26.13
ZeroSearch-Base	34.00	57.50	41.40	26.00	30.00	6.50	16.00	30.20
SearchR1-Base (GRPO)	36.10	**58.00**	41.30	33.00	31.50	7.50	14.40	31.69
SearchR1-Instruct (GRPO)	33.60	56.00	43.90	29.00	31.00	10.50	**28.00**	33.14
SEARCHSKILL-Instruct (3B skills)	35.80	53.90	43.90	**39.50**	35.50	11.50	24.00	34.87
SEARCHSKILL-Base (GPT-5.4 skills)	36.20	53.90	**47.00**	34.00	37.50	13.50	**28.00**	35.73
SEARCHSKILL-Instruct (GPT-5.4 skills)	**37.50**	54.30	46.60	35.50	**41.50**	**16.00**	27.20	**36.94**

Table 2: Web-search accuracy (%). Avg.: mean over levels; Overall: all 340 multi-source questions.

(a) GAIA

Method	Level 1	Level 2	Avg.
Base	9.52	0.00	4.76
Search-R1	16.67	4.55	10.61
SEARCHSKILL	11.90	**6.06**	8.98

(b) WebWalkerQA

Method	Easy	Medium	Hard	Overall
Base	8.75	6.43	4.17	6.18
Reflexion	13.75	6.43	2.50	6.76
SEARCHSKILL	12.50	**8.57**	**6.67**	**8.82**

4.3 GENERALIZATION TO WEB-SEARCH TASKS

Generalization of QA skills. In our evaluation on GAIA (Mialon et al., 2023) Level 1 and Level 2, we deploy the QA-trained 7B policy with its existing SkillBank and Serper web search, without additional training. As shown in Table 2a, SEARCHSKILL approaches Search-R1's average accuracy across the two levels and performs better on the more challenging Level 2, reaching 6.06% versus 4.55%. These results show that QA-derived skills generalize to evidence gathering in unseen web-search environments.

Extending the skill-learning workflow. Starting from a seed SkillBank B_0, Qwen2.5-7B-Instruct evolves its own SkillBank for WebWalkerQA (Wu et al., 2025). We then apply the same two-stage SFT procedure using 200 validated trajectories and evaluate 340 multi-source questions with a 32K context and 15 additional webpage visits. As shown in Table 2b, SEARCHSKILL outperforms Base and Reflexion (Shinn et al., 2023) on Medium and Hard questions and achieves the highest overall accuracy at 8.82%. These results demonstrate that the same workflow can be reapplied to a new web-search task, enabling a 7B model to develop effective skills for navigating and integrating evidence across pages.

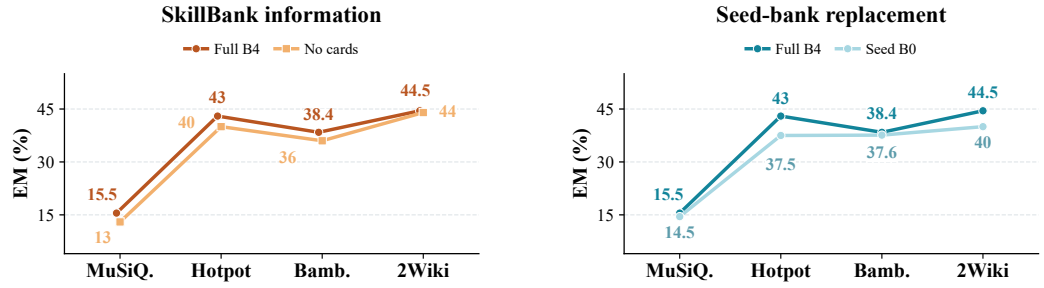

Figure 2: SkillBank information and evolution under the same SFT policy. Left: full B_4 versus skill IDs without cards. Right: evolved B_4 versus seed B_0.

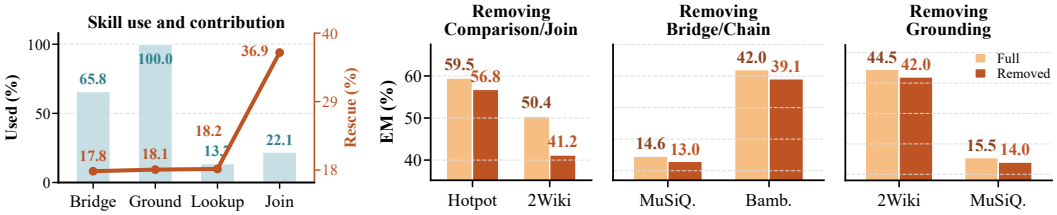

Figure 3: Skill use and contribution. Left: category activation and Rescue rates. Right: EM with the full bank (light) and after removing a skill category (dark).

Table 3: GPT-5.4 distillation control on Qwen2.5-7B-Instruct (EM, %).

Method	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboo.	Avg.
SEARCHSKILL	41.60	61.04	51.40	43.00	44.50	15.50	38.40	**42.21**
GPT-5.4 trajectories	38.64	58.20	46.60	38.00	32.50	18.00	33.60	**37.93**

4.4 SKILLBANK ABLATIONS

SkillBank information and evolution.

Figure 2 separates two sources of SkillBank benefit. The left plot keeps the selected B_4 ids but removes their card content (*No cards*), testing whether names alone are sufficient. The right plot keeps the same SFT policy but replaces the evolved B_4 with the seed bank B_0, testing the effect of bank evolution. Both controls generally reduce performance, showing that SearchSkill benefits from both readable skill-card content and iterative SkillBank refinement. The drops are not catastrophic because the trained backbone already acquires a basic query-decomposition prior from trajectory SFT, but explicit skill content and evolved routing still make the decomposition more reliable.

Skill contribution after activation.

We evaluate whether activated skills are functionally necessary rather than merely selected. Figure 3 groups B_4 cards into direct lookup, bridge/chain, comparison/join, and grounding/verification. *Activation* measures category use over multi-hop questions; *Rescue* is the percentage of activated questions answered correctly and judged to require that category. Removing one category at a time generally reduces EM, showing its contribution to answer construction. Appendix D.5 gives detailed definitions and scoring rules.

4.5 BEYOND TEACHER DISTILLATION

To test whether our gains come solely from teacher distillation, we ask GPT-5.4 to generate new retrieval trajectories without a SkillBank or skill actions, then fine-tune Qwen2.5-7B-Instruct with matched training questions, data size, and two-stage SFT budget. Table 3 shows that this control reaches 37.93 macro EM, compared with 42.21 for SEARCHSKILL, indicating that skill conditioning

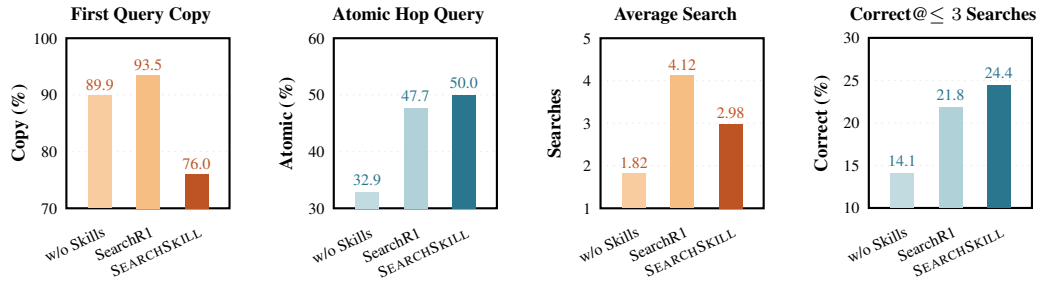

Figure 4: Query planning on four multi-hop benchmarks. Lower copy rate and fewer calls indicate less broad or costly search; higher atomic-query rate and Correct@≤ 3 indicate more focused and effective retrieval.

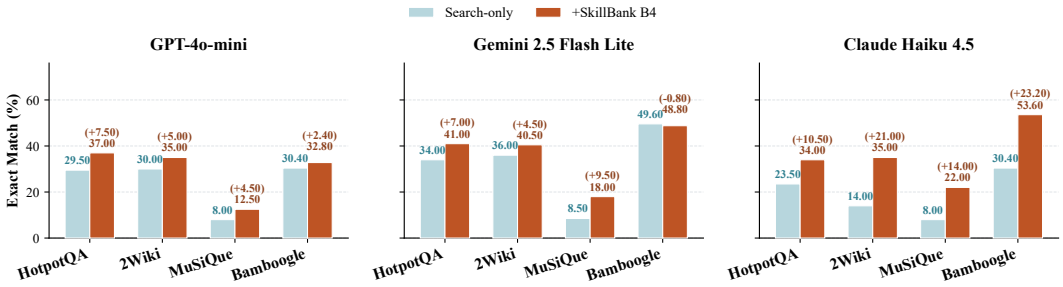

Figure 5: Closed-source transfer with frozen B_4. Bars show EM percentages; parentheses give percentage-point gains over search-only prompting.

contributes beyond teacher supervision alone. Together with Section 4.2, these results suggest that using stronger models to develop reusable skills offers a low-cost, broadly transferable approach to improving search performance and enabling more efficient search in practical applications.

4.6 Query planning and evidence efficiency

We further test whether SearchSkill addresses the query-quality failure that motivates our method: search policies often send the whole multi-hop question to the retriever instead of isolating the next missing entity or relation.

Figure 4 reports four diagnostics macro-averaged over HotpotQA, 2WikiMultiHopQA, MuSiQue, and Bamboogle. *First query copy* (↓) measures whether the first search is a near copy of the original question; *atomic-hop query* (↑) measures the fraction of searches that target one clear retrieval subgoal; *average search* (↓) counts the number of retrieval calls per example; and *Correct@*≤ 3 *searches* (↑) measures the fraction of examples answered correctly within three retrieval calls. Arrows indicate the preferred direction for each metric. See Appendix D.5 for details.

4.7 Closed-source model transfer

We also test whether the frozen SkillBank can serve as an external search prior for proprietary models without fine-tuning. Figure 5 compares a search-only workflow with the same workflow augmented by B_4 as an optional planning guide. The consistent macro gains across all three closed-source models show that learned search skills transfer beyond the trained open-source policies, and that exposing a structured SkillBank before retrieval can improve closed-source search planning.

4.8 Why build the SkillBank in stages?

Staging allows skills to be refined as new tasks are introduced. In the *one-shot* baseline, the 7B model instead receives all training questions and labels together and builds the entire bank at once.

Table 4: SkillBank construction (EM, %). Avg. averages all seven benchmarks.

Construction	HotpotQA	2Wiki	MuSiQue	NQ	TriviaQA	PopQA	Bamboo.	Avg.
Staged SkillBank	41.67	47.92	13.89	31.25	47.92	43.60	28.80	**36.44**
One-shot SkillBank	37.50	35.42	11.11	27.08	46.88	39.00	23.20	**31.46**

Table 5: RL results (EM fractions; Macro: unweighted mean).

Model	**HotpotQA**	**2Wiki**	**MuSiQue**	**Bamboogle**	**Macro**
Qwen2.5-7B-Instruct (GRPO)	0.440	0.455	0.145	0.328	0.342
Qwen2.5-7B-Base (GRPO)	0.400	0.475	0.100	0.336	0.328
Qwen2.5-3B-Instruct (GRPO)	0.355	0.415	0.160	0.280	0.303
Qwen2.5-3B-Base (GRPO)	0.365	0.390	0.135	0.256	0.287

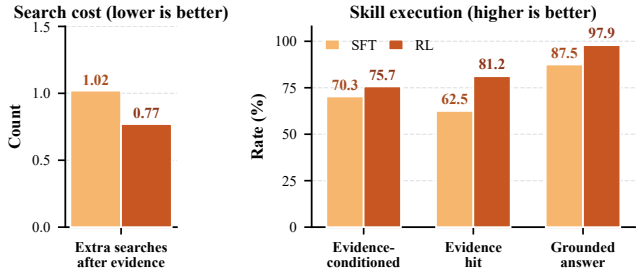

Figure 6: Paired execution diagnostics for 7B-Instruct questions corrected by GRPO. Rates measure evidence use; extra searches count calls after answer-containing evidence is first retrieved.

As shown in Table 4, staging improves all seven benchmarks under matched training and evaluation settings, raising average EM from 31.46% to 36.44%.

4.9 Reinforcement Learning

We further apply a short GRPO refinement stage after two-stage SFT while keeping the SkillBank interface fixed. The main RL results are reported in Table 5. Additional reward dynamics are provided in Appendix C; this experiment probes how GRPO behaves in the SkillBank select-read-act action space and whether reinforcement learning can further shape skill execution.

The diagnostic bars in Figure 6 show that, on SFT failures corrected by GRPO, RL conditions later actions on retrieved evidence more often (↑), hits answer-containing evidence more frequently (↑), grounds final answers more reliably (↑), and reduces extra searches after evidence is found (↓). These metrics suggest that GRPO mainly improves evidence use, answer grounding, and stopping behavior under the fixed SkillBank interface.

5 Conclusion

We present SearchSkill, a skill-conditioned framework that makes search-tool use explicit: the system evolves a reusable SkillBank, constructs bank-guided trajectories, and trains models to select skills before issuing retrieval or answer actions. Across open-source and closed-source models, our results show that explicit skill routing improves query planning, evidence use, and multi-hop answer grounding. The 7B model can also evolve new search skills on its own, eliciting its latent capabilities to achieve performance comparable to agents trained with large-scale reinforcement learning. Our auxiliary GRPO study does not yield large additional accuracy gains, but it noticeably changes execution behavior in the SkillBank action space, indicating that reinforcement learning can reshape how SFT policies select skills, search, and stop. Stronger RL algorithms and task-aligned rewards may further improve the framework.

AI USE STATEMENT

We used generative AI tools to help identify relevant literature, improve the manuscript's language, and debug and optimize a small portion of the code. Language models were also used to generate training trajectories as part of the proposed method, as detailed in Section 3. The authors take full responsibility for the final manuscript, code, and reported results.

REPRODUCIBILITY STATEMENT

All code and configurations required for training are available in the anonymous repository linked in the abstract. Section 3 specifies the framework and training objectives. Appendix D provides data processing, skill cards, rollout pseudocode, and optimization settings, and Appendix B documents the Qwen-generated implementation. Additional RL results, compute and API accounting, and case studies appear in Appendices C, G, and H. Data will be released through the same repository.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2023.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022.

Qinyuan Cheng, Xiaonan Li, Shimin Li, Qin Zhu, Zhangyue Yin, Yunfan Shao, Linyang Li, Tianxiang Sun, Hang Yan, and Xipeng Qiu. Unified active retrieval for retrieval augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 17153–17166, 2024.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pp. 3929–3938. PMLR, 2020.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

Xu Huang, Junwu Chen, Yuxing Fei, Zhuohan Li, Philippe Schwaller, and Gerbrand Ceder. Cascade: Cumulative agentic skill creation through autonomous development and evolution. *arXiv preprint arXiv:2512.23880*, 2025.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume*, pp. 874–880, 2021.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43, 2023.

Pengcheng Jiang, Jiacheng Lin, Lang Cao, Runchu Tian, SeongKu Kang, Zifeng Wang, Jimeng Sun, and Jiawei Han. Deepretrieval: Hacking real search engines and retrievers with large language models via reinforcement learning, 2025a. URL https://arxiv.org/abs/2503.00223.

Pengcheng Jiang, Xueqiang Xu, Jiacheng Lin, Jinfeng Xiao, Zifeng Wang, Jimeng Sun, and Jiawei Han. s3: You don't need that much data to train a search agent via RL. *arXiv preprint arXiv:2505.14146*, 2025b. URL https://arxiv.org/abs/2505.14146.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7969–7992, 2023.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2020.

Hao Li, Chunjiang Mu, Jianhao Chen, Siyue Ren, Zhiyao Cui, Yiqun Zhang, Lei Bai, and Shuyue Hu. Organizing, orchestrating, and benchmarking agent skills at ecosystem scale, 2026. URL https://arxiv.org/abs/2603.02176.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models, 2025. URL https://arxiv.org/abs/2501.05366.

George Ling, Shanshan Zhong, and Richard Huang. Agent skills: A data-driven analysis of claude skills for extending large language model functionality, 2026. URL https://arxiv.org/abs/2602.08004.

Yi Liu, Weizhe Wang, Ruitao Feng, Yao Zhang, Guangquan Xu, Gelei Deng, Yuekang Li, and Leo Zhang. Agent skills in the wild: An empirical study of security vulnerabilities at scale, 2026. URL https://arxiv.org/abs/2601.10338.

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 *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 9802–9822, 2023.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general ai assistants, 2023. URL https://arxiv.org/abs/2311.12983.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback, 2022. URL https://arxiv.org/abs/2112.09332.

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*, pp. 5687–5711, 2023.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. Tool learning with foundation models, 2024. URL https://arxiv.org/abs/2304.08354.

Qwen Team. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025a.

Huatong Song, Jinhao Jiang, Wenqing Tian, Zhipeng Chen, Yuhuan Wu, Jiahao Zhao, Yingqian Min, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-Searcher++: Incentivizing the dynamic knowledge acquisition of LLMs via reinforcement learning. *arXiv preprint arXiv:2505.17005*, 2025b. URL https://arxiv.org/abs/2505.17005.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without searching, 2025. URL https://arxiv.org/abs/2505.04588.

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.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 10014–10037, 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.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. StepSearch: Igniting LLMs search ability via step-wise proximal policy optimization. *arXiv preprint arXiv:2505.15107*, 2025b. URL https://arxiv.org/abs/2505.15107.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2022. URL https://arxiv.org/abs/2201.11903.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. WebWalker: Benchmarking LLMs in web traversal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10290–10305. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.508. URL https://aclanthology.org/2025.acl-long.508/.

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, 2026. URL https://arxiv.org/abs/2602.08234.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

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

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping agents for the real world with Agent Skills. Anthropic Engineering, October 2025. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills.

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.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills, 2025. URL https://arxiv.org/abs/2504.07079.

A Related work

A.1 RAG and RL

Retrieval-augmented generation for knowledge-intensive QA began with retrieval-aware pretraining and dense retrieval backbones such as REALM and DPR (Guu et al., 2020; Karpukhin et al., 2020). RAG and FiD then made retrieved passages central to generation and reading (Lewis et al., 2020; Izacard & Grave, 2021). Later systems such as RETRO, Atlas, and IRCoT further tightened the coupling between retrieval and reasoning (Borgeaud et al., 2022; Izacard et al., 2023; Trivedi et al., 2023). Dynamic retrieval policies were subsequently explored by Active RAG, Self-RAG, and unified active retrieval (Jiang et al., 2023; Asai et al., 2023; Cheng et al., 2024).

In parallel, search became an explicit action space rather than a passive back-end. WebGPT and ReAct framed browsing and search as sequential reasoning-and-acting (Nakano et al., 2022; Yao et al., 2022), while Toolformer and Tool Learning generalized this pattern to broader tool use (Schick et al., 2023; Qin et al., 2024). Search-o1 further embeds agentic search into long reasoning traces without reducing the problem to a static retrieve-then-read pipeline (Li et al., 2025). RL-based search agents then push this line further: R1-Searcher, DeepRetrieval, and Search-R1 directly optimize search behavior with reinforcement learning (Song et al., 2025a; Jiang et al., 2025a; Jin et al., 2025). ZeroSearch complements them by improving search capability without live search during training (Sun et al., 2025). Together, these works frame search as a trainable decision process rather than a fixed retrieval module.

A.2 Skills

Agent skills package instructions, resources, and executable code into reusable bundles that an agent can discover and load when relevant. A recent data-driven analysis of Claude Skills shows that such skills are already functioning as a practical extension layer for LLM agents (Ling et al., 2026). At the same time, Agent Skills in the Wild highlights that once skills become reusable infrastructure, their organization and controlled invocation matter at scale (Liu et al., 2026).

Recent research extends this idea from hand-authored skills to learned and evolving skill ecosystems. SkillWeaver shows that web agents can discover and hone reusable skills from interaction traces (Zheng et al., 2025). Reinforcement Learning for Self-Improving Agent with Skill Library studies iterative policy improvement with an explicit skill library (Wang et al., 2025a). CASCADE studies cumulative agentic skill creation through autonomous development and evolution, while AgentSkillOS investigates organization and orchestration at ecosystem scale (Huang et al., 2025; Li et al., 2026). MemSkill reframes memory operations as evolvable skills (Zhang et al., 2026), and SkillRL co-evolves a hierarchical SkillBank with the agent policy under reinforcement learning (Xia et al., 2026). Collectively, these works shift skills from hand-authored helpers to dynamic, reusable, and trainable components for agents, while study of skills for search-tool interaction remains limited.

B Self-generated skills and trajectories

The Qwen-generated setting uses the target Qwen2.5 backbone as $\mathcal{G}$ in Section 3. This appendix details the 7B run: Qwen2.5-7B-Instruct groups training questions, proposes and revises cards, executes search trajectories, and produces final answers. Bank evolution and trajectory generation use the fixed backbone; two-stage SFT then trains the search policy with the resulting bank and trajectories.

B.1 Data grouping and bank evolution

Training questions come from NQ, TriviaQA, HotpotQA, 2WikiMultiHopQA, and MuSiQue. Dataset-native metadata supplies coverage groups for multi-hop data. For NQ and TriviaQA, mean-pooled hidden states from the same 7B backbone are grouped by cosine k-means into eight clusters per dataset. Sampling balances these groups, with the number of questions per dataset allocated in proportion to the square root of the available pool size. Normalized question overlap with the available evaluation files is removed during manifest construction.

Table 6: Trajectory generation with the Qwen2.5-7B-Instruct generator.

Dataset	Training questions	Accepted trajectories
NQ	1,021	318
TriviaQA	1,021	419
HotpotQA	2,247	1,063
2WikiMultiHopQA	1,835	838
MuSiQue	1,876	221
Total	8,000	2,859

Starting from the six-card seed bank, each round uses 120 representative training questions to propose a candidate bank. The generator tests it through retrieval rollouts and receives their aggregate outcomes and compact success and failure traces. Skills used in successful trajectories are protected; unsuccessful candidates can be revised, merged, or rejected. Each card specifies When (trigger), Procedure (actions), Checkpoints (resolved facts), Continue (next-step conditions), Avoid (failure patterns), and Finalize (answer conditions). The four rounds produce banks of 8, 11, 12, and 13 skills. Their active datasets follow the NQ/TriviaQA, HotpotQA, 2Wiki, and MuSiQue order, with additional training questions from the other datasets included in each round.

B.2 Trajectory construction and SFT

The final 13-card bank guides rollouts over 8,000 training questions. The generator sees the question, its own prior actions, and retrieved passages when choosing queries and final answers. Exact match against the training answers, at least one retrieval action, and legal skill identifiers determine acceptance. This yields 2,859 successful trajectories.

Each accepted trajectory contributes a full-trajectory record and a closure-focused record that supervises only the final answer. An additional 21 records repeat trajectories using empirically rare successful skills. The resulting data contains 5,675 training records and 64 validation records. Stage I learns skill-tagged search and answer actions. Stage II rewrites each action into a selection turn, a selected-card context turn, and an execution turn, removing the question-specific recommended-skill hint. The stages use the same learning rates, sequence lengths, loss weights, and LoRA configuration listed in Table 12.

B.3 Evaluation

The seven-benchmark evaluation contains 200 questions each from HotpotQA, 2WikiMultiHopQA, and MuSiQue; 1,000 each from NQ, TriviaQA, and PopQA; and all 125 Bamboogle questions. The Qwen-generated 7B policy obtains 38.60 macro EM, and the reported Qwen-generated 3B policy obtains 34.87, as shown in Table 1.

C Reinforcement learning analysis

We further apply a short GRPO refinement stage after two-stage SFT while keeping the SkillBank interface fixed. As an auxiliary test of post-SFT optimization, we use a lightweight reward

$$r = r_{\mathrm{EM}} + \lambda_e r_{\mathrm{evi}} - \lambda_d r_{\mathrm{dup}},$$

where the terms reward final-answer correctness and answer-containing evidence while discouraging duplicate searches. The main RL results are reported in Table 5; for each run, we select the reported checkpoint according to training diagnostics, including reward trends, valid-search behavior, and development scores, rather than always using the final checkpoint. Additional reward dynamics are provided in Appendix D.6; this experiment probes how GRPO behaves in the SkillBank select-read-act action space and whether reinforcement learning can further shape skill execution.

Figure 7 reports reward and valid-search dynamics for the 7B GRPO runs. The 7B-Instruct policy shows a noisy but sustained increase in train reward while keeping valid search usage near one search

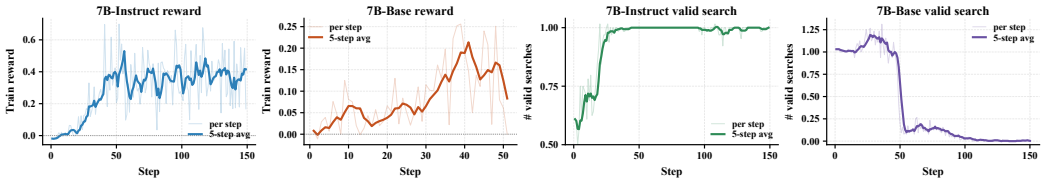

Figure 7: GRPO training diagnostics for 7B models. Panels show mean rollout reward and valid search calls per rollout. A valid call is a parsed search action executed by the retriever.

per rollout. The 7B-Base run initially improves, but its valid-search count later collapses, so we truncate its reward panel at the collapse point and treat this behavior as a reward-design limitation rather than a stable gain. This contrast motivates using both reward and action-validity diagnostics when selecting RL checkpoints.

Across all 725 paired 7B-Instruct evaluation questions, mean search calls decrease from 2.98 to 2.48 after GRPO. The mean number of calls after the first answer-containing passage falls from 0.60 to 0.42. Both means include every question; a trajectory that never retrieves answer-containing evidence contributes zero to the latter metric. Macro EM across the four datasets changes from 35.35% to 34.20%.

Figure 6 compares the SFT and RL trajectories on questions that change from wrong to correct. *Evidence-conditioned* tracks subsequent queries that use information from earlier retrieval. *Evidence hit* records retrieval of an answer-containing passage, and *Grounded answer* records agreement between the final answer and its supporting evidence. *Extra searches after evidence* counts retrieval calls after the first answer-containing passage is found. The first three are rates (↑); the last is a mean count (↓). On this corrected subset, GRPO improves evidence use and grounding while reducing additional searches.

D Experimental details

SearchSkill uses the same stage-wise bank evolution, trajectory construction, and two-stage SFT framework across generator choices. The detailed data-processing recipe and skill taxonomy below describe the GPT-5.4-generated instantiation; Appendix B records the Qwen-generated implementation.

D.1 Data process

We construct the training pool with a coverage-oriented data processing pipeline rather than uniform random sampling. The pipeline first reads the cached training splits and removes direct overlap with the corresponding evaluation ids when such ids are available. It then converts each example into a lightweight profile containing the question, answer form, question type, coarse entity count, length bucket, dataset-native type or hop count, and heuristic retrieval cues such as temporal, numerical, comparison, alias, verification, and relation-chain indicators. These profiles are grouped into coarse reasoning signatures, which serve as the sampling units for preserving diversity.

For single-hop datasets, Natural Questions and TriviaQA are sampled with a signature-capped strategy. Rare signatures are protected, frequent signatures are capped, and the remaining examples are ranked to improve label coverage over answer forms, WH types, question length, entity density, and retrieval-related cues. This produces 3,000 examples from Natural Questions and 3,000 examples from TriviaQA. For multi-hop datasets, HotpotQA, 2WikiMultiHopQA, and MuSiQue are first grouped by signatures that include native question type and hop count. We then ask GPT-5.4 to annotate representative groups with reasoning types, skill demands, answer form, entity pattern, difficulty, and coverage priority. The sampler uses these annotations to enforce coverage over bridge reasoning, comparison, relation chains, temporal and numerical reasoning, set constraints, verification, and answer composition. After initial selection, the pipeline removes duplicate or conflicting questions, reviews suspicious records with the same model, and refills dropped items from the ranked candidate pool.

Table 7: Coverage-oriented training subsets produced by the data processing pipeline.

Dataset	Original examples	Reasoning signatures	Selected examples
Natural Questions	79,168	1,086	3,000
TriviaQA	78,785	5,130	3,000
HotpotQA	90,447	3,094	12,445
2WikiMultiHopQA	15,000	410	8,300
MuSiQue	19,938	2,118	8,594

Table 8: Curriculum used to evolve the SearchSkill SkillBank. New skills are added only when the corresponding data stage exposes a reusable search or verification behavior.

Bank	Evolution data	Added	Main purpose of the update
B_0	Seed bank	6	Core bridge search, comparison, temporal extraction, conflict checking, evidence extraction, and grounding.
B_1	NQ + TriviaQA	5	Add a single-hop spine for direct relation lookup, surface-name resolution, anchored queries, ranking lookup, and forced-choice answers.
B_2	HotpotQA	3	Add reusable two-hop bridge planning, bridge disambiguation, relation-chain decomposition, and multihop yes/no verification.
B_3	2WikiMultiHopQA	3	Add bridge-comparison planning, derived-kinship inference joins, and checkpointed longer-hop retrieval.
B_4	MuSiQue	3	Add long-hop re-anchoring, temporal-anchor carry-forward, and full-chain reconstruction before final answering.

The output of this stage consists of lightweight profiles, group annotations, sampling reports, and final full-record JSONL files. The sampled examples are then passed to the trajectory construction pipeline, where teacher rollouts, retriever interactions, and positive-trajectory filtering produce the SFT supervision described in Section 3.

D.2 SkillBank evolution

The SkillBank is evolved before trajectory SFT because the search interface should expose reusable retrieval policies rather than dataset-specific answer shortcuts. If the bank is too small, the model must express distinct behaviors with the same generic search action; if it is expanded without structure, the model faces a noisy routing problem. We therefore grow the bank in curriculum order, starting from a compact B_0 that contains the core bridge, comparison, temporal, conflict-checking, evidence-span, and answer-grounding skills, and then adding only skills that correspond to stable behavior patterns observed in the sampled data.

Each evolution round uses the processed samples from Appendix D.1. The examples are grouped into discovery packets by reasoning signature, and each packet keeps a small set of representative questions together with metadata such as dataset, group size, native type, hop count, answer form, entity pattern, and heuristic flags. These packets are summarized into coverage buckets, then GPT-5.4 proposes a new SkillBank version from the previous bank, the bucket summaries, and representative examples. The update is constrained to preserve existing useful skills, add skills only when they change search planning or verification behavior, and refine old skills when their routing boundary is too broad. This design makes the bank an explicit external prior for later teacher synthesis, rather than something the student must rediscover from SFT labels alone.

This staged evolution is important for two reasons. First, it separates skill discovery from policy optimization: the bank is built by analyzing broad coverage packets, while the student later learns to invoke the resulting skills through explicit tags. Second, it avoids asking the model to choose from a large long-hop bank before it has stable simpler skills. Direct lookup and answer grounding are established first, bridge and comparison skills are added next, and only then do we introduce long-hop decomposition and recovery skills for MuSiQue-style questions. The final B_4 is therefore a cumulative skill prior that covers single-hop lookup, two-hop bridge reasoning, compositional comparison, and longer decomposition under one consistent interface.

Table 9: GPT-5.4-generated B_4, grouped by the four functional categories in Section 4.4. Summaries are condensed from the cards; the Qwen2.5-7B-generated bank is described in Appendix B.

Category	Skill	Skill-card summary
Direct lookup	`single-entity-relation-lookup`	Query one resolved entity plus the requested relation or attribute, then verify that the evidence explicitly attaches the answer to that entity.
	`surface-name-resolution`	Resolve real names, alternate names, former names, nicknames, or character-to-actor mappings from contextual name-linking evidence.
	`superlative-ranking-match`	Retrieve and verify a superlative claim such as first, largest, highest, oldest, or top-ranked within the specified set and timeframe.
	`forced-choice-option-resolution`	Treat an explicit option set in the question as answer anchors, verify which option matches the evidence, and answer with that option span.
	`temporal-range-extract`	Extract a date, year, count, measurement, rank, or range for an already identified target while preserving unit, scope, and timeframe.
Bridge/chain	`bridge-entity-search`	Find one hidden intermediate entity from a named clue, then search the final attribute on that confirmed bridge entity.
	`bridge-disambiguate-then-hop`	Resolve an alias-heavy or underspecified bridge candidate with rare anchors before launching the second-hop attribute search.
	`relation-chain-decomposition`	Follow an explicit ordered relation chain one edge at a time, verifying each intermediate before continuing to the next relation.
	`multi-constraint-query-anchoring`	Build a query from several rare constraints when the next target is clue-heavy or alias-prone, then re-anchor after the candidate is verified.
	`sequential-hop-checkpointing`	Preserve multiple intermediate checkpoints in 3-plus-hop or mirrored multi-step questions, restarting from the last secure checkpoint when needed.
	`re-anchored-long-hop-decomposition`	Solve long clue-heavy paths by making each confirmed entity the new anchor for the unresolved remainder of the question.

Table 9: GPT-5.4-generated B_4 (continued).

Category	Skill	Skill-card summary
	temporal-anchor-carry-forward	Attach a temporal qualifier to the hop it constrains, carry the time-scoped checkpoint forward, and verify downstream evidence under that timeframe.
Comparison/join	parallel-attribute-compare	Retrieve matched evidence for two named entities under the same attribute, unit, definition, and timeframe before comparing or selecting.
	bridge-comparison-planning	Resolve hidden bridge entities on both sides of a comparison before retrieving endpoint attributes and making the final comparison.
	derived-kinship-inference-join	Derive family roles such as paternal grandmother or in-law relations by joining immediate relative and second-edge family evidence.
	multihop-yes-no-verification	Decompose a yes/no claim into required subclaims, gather matched evidence for each, and answer yes only when all are explicitly supported.
Grounding/ verification	conflict-check	Run a targeted disambiguation search when retrieved evidence surfaces competing entities, titles, dates, numbers, or inconsistent checkpoints.
	verbatim-evidence-span	Copy the shortest faithful answer span directly supported by evidence, preserving names, titles, units, quotations, or list structure.
	answer-grounding-check	Apply a final grounding checklist over answer support, relation match, answer type, units, timeframe, and all bridge or comparison edges.
	reconstructed-chain-verification	Replay confirmed checkpoints before answering long chains, ensuring the endpoint answer belongs to the intended final target rather than a nearby entity.

D.3 Trajectory construction

Trajectory construction turns the sampled examples and the evolved SkillBank into executable supervision. We first build a manifest from the processed training pools in Appendix D.1. Each manifest item stores the question, gold answers, dataset family, metadata summary, candidate primary skills, and suggested support skills. The manifest also mixes a small number of failure-replay examples from earlier evaluations, so the teacher can reconstruct trajectories for cases where previous policies selected poor skills, stopped too early, or answered with an intermediate entity.

For each manifest item, GPT-5.4 acts as a teacher policy under the current SkillBank. At every step, the teacher returns a structured JSON action containing a primary skill, optional support skills, an action type, a search query or draft answer, and a checkpoint of the current resolved state. If the action is `search` or `verify`, the local retriever is called with top-3 retrieval and the returned passages are appended to the trace. If the action is `answer`, a finalizer prompt normalizes the draft into the shortest answer span that is explicitly supported by the collected evidence, typically using `verbatim-evidence-span` together with `answer-grounding-check`. This produces raw trajectories with interleaved skills, actions, retrieved evidence, checkpoints, and final answers.

Table 10: Final trajectory-format SFT pool. The train split reuses validated replay trajectories and adds cleaned coverage-supplement trajectories.

Source	Records	Role in the SFT pool
Replay train records	1,445	Preserve earlier validated SearchSkill supervision
Cleaned supplement trajectories	786	Add broader coverage after filtering and deduplication
Closure-expanded supplement records	1,572	Add action and closure supervision from the supplement
Final train split	3,017	Trajectory-format SFT training data
Final eval split	16	Held-out SFT validation records

Algorithm 1 Two-phase SkillBank interface in SearchSkill

Require: Question x, policy model, SkillBank $B^\star$, retriever, budget T
Ensure: Final answer and skill-annotated interaction trace

Build compact index $I = \mathrm{Index}(B^\star)$
Initialize evidence $E_0 \leftarrow \emptyset$, prompt state $H_0 \leftarrow [x, I]$, trace $\Gamma \leftarrow \emptyset$
for $t = 1, \ldots, T$ **do**
  Generate skill-selection turn and parse s_t from <select_skill>...</select_skill>
  Load selected cards $C_t \leftarrow \mathrm{Cards}(B^\star, s_t)$
  Generate action turn <skill>$\hat{s}_t$</skill> followed by one action tag
  Parse $o_t \in \{$<search>, <answer>$\}$ and content z_t
  Update trace $\Gamma \leftarrow \Gamma \cup \{(s_t, C_t, o_t, z_t)\}$
  if $o_t =$ SEARCH **then**
    Execute retriever $R_t \leftarrow \mathrm{Search}(z_t)$
    Update evidence $E_{t+1} \leftarrow E_t \cup R_t$
    Update prompt $H_{t+1} \leftarrow H_t + (s_t, C_t, o_t, z_t) +$ <information>R_t</information> $+ I$
  else if $o_t =$ ANSWER **then**
    return final answer z_t and trace Γ
  else
    Stop with invalid action
  end if
end for
Stop when the search budget is exhausted

We keep only trajectories that pass executable validation. A trajectory must answer correctly under exact-match normalization, contain at least one retrieval step, use only legal SkillBank identifiers, and avoid using support-only verification skills as the primary skill before the final answer. When multiple rollout sources contain the same example, we keep the higher-quality row according to validation success, runtime status, nonempty final answer, legal skill usage, route consistency, and trace completeness. The early canonical teacher set contains 360 validated trajectories from 757 unique candidates, but the final SFT pool is larger because we continue with coverage-supplement rollouts and replay earlier validated data.

The final SFT data is therefore not limited to the 360 early canonical trajectories. The trajectory-format training split contains 3,017 records: 1,445 records from the earlier validated replay set and 1,572 additional records derived from 786 cleaned coverage-supplement trajectories after closure expansion. These records supervise the model to emit explicit <skill>...</skill> declarations before each <search>...</search> or <answer>...</answer> action, while the held-out SFT validation split contains 16 records.

Table 11: Question-type distribution of the trajectory-format SFT training data. "Unknown" mainly corresponds to the single-hop NQ and TriviaQA records, where no dataset-native multihop type is defined.

Question type	Records	Main source
Bridge	682	HotpotQA
Comparison	561	HotpotQA / 2WikiMultiHopQA
Compositional	371	2WikiMultiHopQA
MuSiQue decomposition-2	201	MuSiQue
MuSiQue decomposition-3	217	MuSiQue
MuSiQue decomposition-4	156	MuSiQue
Bridge comparison	104	2WikiMultiHopQA
Inference	95	2WikiMultiHopQA
Single-hop / unlabeled	630	NQ / TriviaQA / residual records

Table 12: Two-stage SFT configuration for the Qwen2.5-7B-Instruct SearchSkill policy.

Item	Stage I: execution SFT	Stage II: SkillBank-card SFT
Initialization	Qwen2.5-7B-Instruct	Stage-I adapter on Qwen2.5-7B-Instruct
Supervision format	<skill> plus one <search> or <answer> action	<select_skill>, selected skill cards, then <skill> plus one action
Train / eval records	3,017 / 16	3,017 / 16
Max sequence length	8,192	12,288
Truncation side	Left	Left
Learning rate	7×10^{-5}	1×10^{-5}
Epochs	2.0	1.0
Warmup ratio	0.05	0.03
GPUs	4	4
Per-device train batch	1	1
Gradient accumulation	4	4
Effective train batch	16	16
LoRA setting	$r = 16$, $\alpha = 32$, dropout 0.05	$r = 16$, $\alpha = 32$, dropout 0.05
Loss weights	answer 2.5, search 0.8, other 1.0	answer 2.0, search 0.8, other 1.0
Scheduler / precision	cosine; bf16 when supported, otherwise fp16	cosine; bf16 when supported, otherwise fp16
Logging / eval / save steps	5 / 40 / 40	5 / 40 / 40

D.4 TWO-STAGE SFT TRAINING

This section reports the 7B-Instruct SFT configuration used for the main SearchSkill policy. Both stages use Qwen2.5-7B-Instruct as the backbone and train LoRA adapters with the same target modules: `q_proj`, `k_proj`, `v_proj`, `o_proj`, `up_proj`, `down_proj`, and `gate_proj`. The first stage trains the execution format from the validated trajectory-format supervision. The second stage initializes from the first-stage adapter and trains the SkillBank-card protocol, where each original action turn is rewritten into a skill-selection turn, a selected-card context turn, and the original skill-grounded action turn.

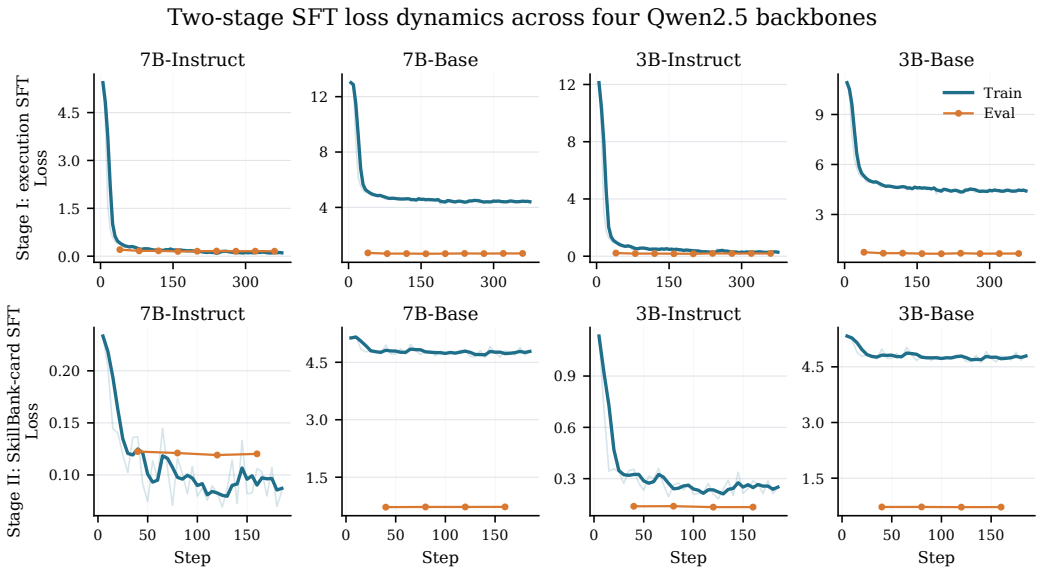

Figure 8: Training and validation loss curves for the two-stage SFT runs across four Qwen2.5 backbones. Each row corresponds to one SFT stage, and columns correspond to model variants. Thin lines show raw training loss and thick lines show a three-point moving average.

During preprocessing, user and system tokens are masked out and only assistant outputs are supervised. For records marked as full-trajectory supervision, every assistant turn contributes to the loss; for closure-focused records, only the final assistant turn is supervised. The weighted token loss assigns larger weight to answer turns to improve grounded stopping, while search turns receive a smaller weight to avoid over-optimizing surface query strings.

SFT loss dynamics. Figure 8 summarizes the training curves for the two supervised stages. Across both stages, instruction-tuned backbones absorb the SearchSkill format more easily than base backbones: their validation losses are lower, and the Stage-II SkillBank-card objective remains stable after initializing from Stage I. Base models still reduce loss under the same supervision, but their much higher training loss suggests that learning the structured search protocol is harder without prior instruction-following behavior. This supports using the two-stage recipe as a portable training protocol while expecting stronger and cleaner convergence from instruction-tuned initializations.

D.5 Definitions of diagnostic metrics

Skill activation and Rescue in Figure 3. The evaluation set $\mathcal{D}$ contains 725 questions: 200 each from HotpotQA, 2WikiMultiHopQA, and MuSiQue, and 125 from Bamboogle. For category c, let A_c contain the questions whose trajectory uses at least one skill in that category. Each question is counted once per category, even if the category is used repeatedly. Let $N_c \subseteq A_c$ contain the questions with an EM-correct answer for which the judge identifies a necessary contribution from that category. We report

$$\text{Activation}(c) = 100\frac{|A_c|}{|\mathcal{D}|}, \qquad \text{Rescue}(c) = 100\frac{|N_c|}{|A_c|}.$$

Activation measures how often a category is used; Rescue measures how often its use is judged necessary in a successful answer. Incorrect activated examples remain in the denominator and contribute zero to the numerator.

Only correct activated traces are sent to GPT-4o-mini at temperature 0. The judge receives the question, reference and predicted answers, skill ids, and ordered actions with queries and retrieved evidence. It checks whether removing the category's steps would lose a required fact, intermediate entity, comparison decision, or evidence-based correction. A positive judgment must cite concrete step ids. A decisive lookup, required bridge, comparison, or correction qualifies; routine formatting

Table 13: GRPO configuration for the 7B diagnostic runs. Figure 10 reports the available cumulative 7B-Base training-reward curve.

Setting	7B-Instruct	7B-Base
Initialization	Two-stage SFT policy	Two-stage SFT policy
Training / validation examples	1200 / 80	1200 / 80
GPUs / parallelism	4 H20, FSDP offload	4 H20, FSDP offload
GRPO samples	8 rollouts per prompt	8 rollouts per prompt
Train / validation batch	8 / 80	8 / 80
PPO mini / micro batch	32 / 4	32 / 4
Actor learning rate	1×10^{-6}	1×10^{-6}
KL regularization	low-variance KL, coefficient 0.001	low-variance KL, coefficient 0.001
Entropy coefficient / grad clip	0.001 / 1.0	0.001 / 1.0
Rollout decoding	temperature 1.1, top-$p = 0.98$	temperature 1.1, top-$p = 0.98$
Search interface	top-3 retriever, max 5 turns	top-3 retriever, max 5 turns
Length limits	prompt 8192, response 192, observation 1400	prompt 8192, response 192, observation 1400
Eval / save frequency	every 15 steps / every 15 steps	every 15 steps / every 15 steps
Planned optimization steps	150	150

and repeated information do not. Outputs contain `rescued`, `step_ids`, and `reason`. Rescue is a judgment about the recorded trajectory. The category-removal experiment instead removes the corresponding cards from the available bank and reruns inference to measure the change in EM.

Query planning in Figure 4. *First query copy* measures the fraction of first queries that reproduce or paraphrase the question as a whole, retaining its unresolved constraints instead of choosing a retrieval target. *Atomic-hop query* measures the fraction of all search calls directed at one retrieval subgoal: an entity, relation, attribute, bridge, comparison operand, or verification fact. For example, a birthplace query about an already resolved person is atomic; repeating a question that still requires resolving that person's identity is not. *Average search* is the number of retrieval calls divided by the number of questions. *Correct@*≤ 3 *searches* is the percentage of all questions for which EM is correct and the complete trajectory uses at most three retrieval calls. It includes both correctness and cost in its numerator; it is not accuracy conditioned on short trajectories. Each diagnostic is first computed within a dataset and then averaged equally across the four multi-hop datasets.

D.6 GRPO REWARD DYNAMICS

Our RL action space follows the same fixed SkillBank interface used at inference time: the policy selects skill ids, reads the corresponding card content, and then emits either a search query or a final answer. We use this section only to report auxiliary reward dynamics rather than to introduce a separate RL method; the reward used for the post-SFT test is summarized in Section 4.9.

Figure 9 shows the 7B-Instruct GRPO trajectory initialized from two-stage SFT: training reward rises after early skill-space exploration while valid search usage remains stable, but the validation curves remain noisy across benchmarks. Figure 10 reports the cumulative 7B-Base training-reward curve; reward rises during the early and middle training phase, then becomes unstable and decays toward the end of the run. These diagnostics support treating RL as a promising but still exploratory extension for directly optimizing skill selection, search execution, and grounded stopping beyond SFT imitation.

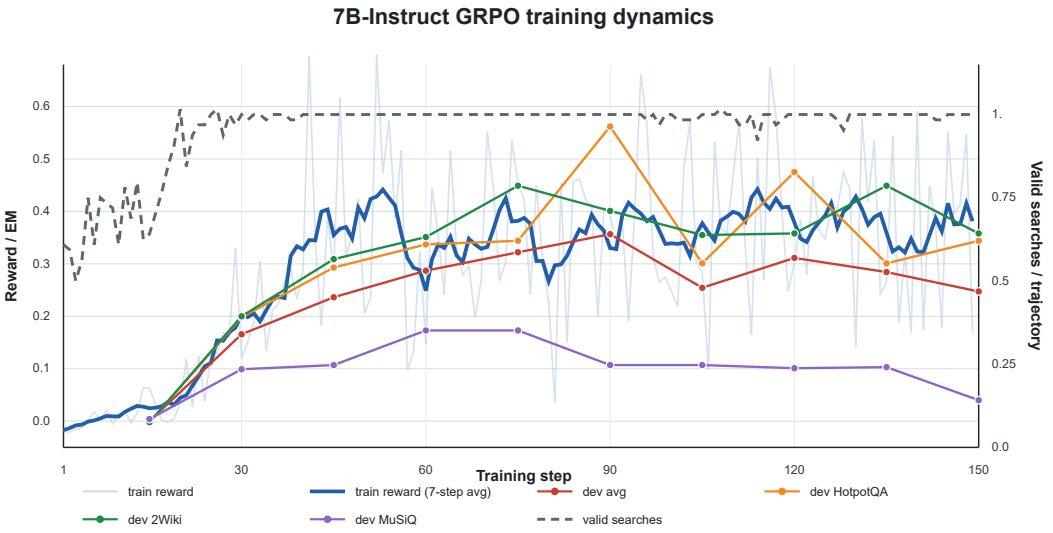

Figure 9: GRPO training dynamics for the 7B-Instruct SFT-initialized policy. The plot shows train reward, smoothed train reward, validation exact-match curves, and valid-search usage.

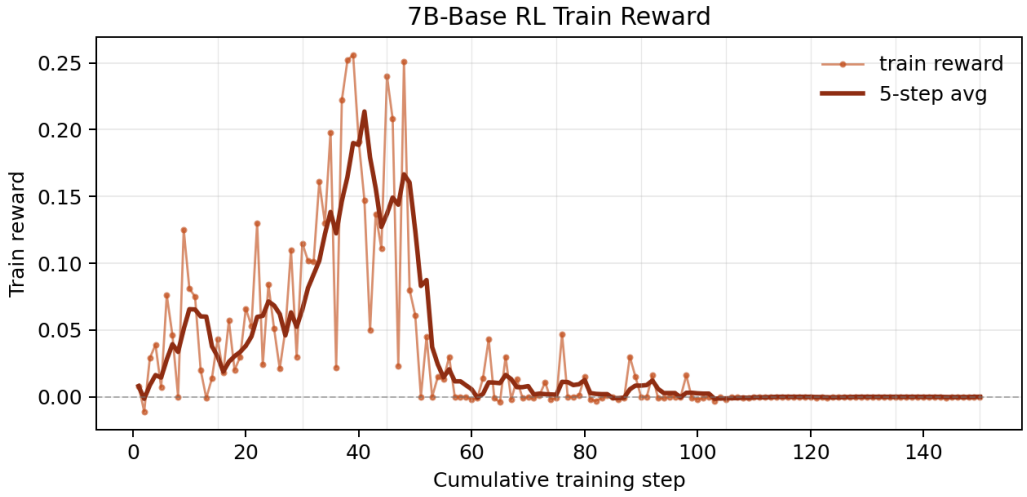

Figure 10: Cumulative GRPO training reward for the 7B-Base SFT-initialized policy. The plot shows raw train reward and a five-step moving average over cumulative training steps.

E LIMITATIONS AND FUTURE DIRECTIONS

SearchSkill currently evolves its SkillBank for multi-hop question answering from a finite, coverage-oriented training pool. This makes the learned bank effective for the reasoning patterns seen in our data, but it may miss skills that only appear in new domains, new tools, or rare reasoning forms. Extending the framework therefore requires more complete mechanisms for continual skill discovery, validation, merging, and revision when the task distribution changes.

Our reinforcement-learning results also reveal both flexibility and risk. GRPO can substantially change the action habits of an SFT policy in the select-read-act space, but it can also disturb behavior on multi-hop questions that the SFT model already solves, for example by changing search frequency or weakening stable stopping behavior. Better algorithms and rewards that are explicitly matched to SkillBank routing, retrieval execution, and grounded answering are needed; with such alignment, reinforcement learning may further improve the capability of the overall workflow.

F EXISTING ASSETS AND LICENSES

SearchSkill uses publicly released benchmarks, model checkpoints, retrievers, and baseline implementations only for academic research. Tables 14 and 15 record the main existing assets used or cited by the experimental setup; the corresponding source papers are cited in the main text and bibliography. Our released code and data package will not relicense these upstream assets; it will preserve the original source attribution, license names, and terms of use, and will provide scripts or pointers for obtaining assets from their original providers when redistribution is not appropriate.

Table 14: Existing benchmark assets and license or terms notes.

Asset	**Role in this paper**	**License / terms note**
Natural Questions	Single-hop benchmark and training pool source	Apache-2.0 in the official repository.
TriviaQA	Single-hop benchmark and training pool source	Apache-2.0 in the official repository.
PopQA	Single-hop benchmark	MIT in the official PopQA/adaptive-retrieval repository.
HotpotQA	Multi-hop benchmark and training pool source	CC BY-SA 4.0 for the dataset and processed Wikipedia release; Apache-2.0 for the official code repository.
2WikiMultiHopQA	Multi-hop benchmark and training pool source	Apache-2.0 in the official repository.
MuSiQue	Multi-hop benchmark and training pool source	CC BY 4.0 in the official repository.
Bamboogle	Multi-hop benchmark	MIT in the official Self-Ask/Bamboogle repository.

Table 15: Existing model, retriever, and baseline assets and license or terms notes.

Asset	Role in this paper	License / terms note
Qwen2.5-7B base/instruct	Backbone models	Apache-2.0 on the Hugging Face model cards.
Qwen2.5-3B base/instruct	Backbone models	Qwen Research License on the Hugging Face model cards.
E5 retriever	Local retrieval model family used by the search environment	MIT on the Hugging Face model card and UniLM/E5 release.
Search-o1	Search-agent baseline	MIT in the official repository.
Search-R1	Search-agent baseline	Apache-2.0 in the official repository.
ZeroSearch	Search-agent baseline	Apache-2.0 in the official repository.
Direct, CoT, and RAG baselines	Locally implemented baseline protocols	No external baseline code package is redistributed; implementations follow the cited papers and use the assets listed above.

G Cost analysis

G.1 Compute budget

Table 16 gives a coarse compute accounting for the GPT-5.4-generated SearchSkill experiments. We report the budget in reserved H20 GPU-hours because some long-running data-construction jobs reserve an 8-H20 node, while SFT and GRPO training use four H20 GPUs and benchmark evaluation uses two H20 GPUs for the retriever plus two H20 GPUs for dev-set evaluation. The numbers should therefore be read as an approximate end-to-end engineering cost rather than a hardware-efficiency benchmark.

For the four GRPO runs used in Section 4.9, the logged wall-clock times are approximately 10.39 h for 7B-Instruct, 7.19 h for 7B-Base, 2.96 h for 3B-Instruct, and 3.17 h for 3B-Base. The two-stage SFT cost is much smaller: the logged 7B-Instruct stages take about 1.21 h in total. Overall, the dominant cost is not LoRA SFT itself, but search-conditioned rollout generation and GRPO, where each optimization step repeatedly samples tool-use trajectories.

G.2 API budget

Table 17 summarizes external API usage for the GPT-5.4-generated experiments, judging, and closed-source transfer. We separate data synthesis, automatic judging, and closed-source transfer because they serve different roles in the pipeline. Data synthesis is the dominant API cost because it asks GPT-5.4 to build profiles, evolve SkillBank candidates, and synthesize or revise trajectories. All judge calls use GPT-4o-mini. Closed-source transfer counts only the three final comparison models: Claude Haiku 4.5, GPT-4o-mini, and Gemini 2.5 Flash-Lite.

For the GPT-5.4-generated setting, most API cost is incurred during skill and trajectory construction. The resulting cards and training data are reused for SFT and evaluation. With Qwen-generated skills, the target backbone performs this generation locally; its data-construction procedure is described in Appendix B.

Table 16: Approximate compute budget for data construction, SFT, RL, and evaluation. Wall time is based on available run logs when present; otherwise we report a conservative range from the observed pipeline runs.

Stage	Main resource	Wall time	Reserved H20-hours	Scope
Data profiling and sampling	CPU/API, light retrieval	3–5 h	< 5	Build dataset profiles, group question packets, and select coverage-oriented pools.
SkillBank evolution	CPU/API	1–2 h	0	Generate and validate staged bank updates from B_0 to B_4.
Teacher rollout and trajectory validation	8 H20 node	6–10 h	48–80	Generate candidate traces, run search-tool validation, filter positives, and keep diagnostic failures.
Stage-I execution SFT	4 H20 training	0.4–0.5 h per backbone	3–4 per backbone	Train the basic `<skill>` plus action format; the logged 7B-Instruct run takes 0.44 h.
Stage-II SkillBank-card SFT	4 H20 training	0.75–0.80 h per backbone	6–7 per backbone	Continue from Stage I with `<select_skill>` and selected SkillBank cards; the logged 7B-Instruct run takes 0.77 h.
GRPO refinement	4 H20 training	2.7–10.4 h per run	22–83 per run	Run fixed-bank GRPO with policy, reference, and rollout devices; 7B runs are longer than 3B runs.
Benchmark evaluation	2 H20 retriever + 2 H20 dev evaluation	2–4 h per checkpoint suite	4–16	Evaluate checkpoints on single-hop and multi-hop QA splits with the same search interface.

Table 17: Approximate API usage by pipeline component. Values are read from the provider usage dashboards and rounded for readability.

Component	Model / service	Requests	Usage scale	Cost
Data synthesis	GPT-5.4	27.0K	101.96M tokens	$416.13
Automatic judging	GPT-4O-MINI	2.712K	4.107M tokens	$0.88
Closed-source transfer	Claude Haiku 4.5; GPT-4o-mini; Gemini 2.5 Flash-Lite	16.015K	Dashboard billed usage	$17.20

H CASE STUDIES

The following examples use the same display style as the retrieval-quality examples in Section 4.6: each case shows the question, final answer, query sequence, and retrieved output. We keep the text compact enough for readability, but retain the actual multi-turn operating process.

H.1 CASE 1: RECOVERING THE HIDDEN EMU BRIDGE

Question: When was conscription introduced in country having Emus during the war setting *The Things They Carried*?
Ground Truth: 1964
Search-R1 Answer: malformed `</search>`
SEARCHSKILL Answer: `<answer>1964</answer>`

– –

Search-R1 Trace
Query 1: *country having Emus during the war setting The Things They Carried*
Output:
Doc 1: *Emu War*. The Emu War was a wildlife-management military operation in Australia in 1932.
Doc 2: *Emu*. The emu is a large flightless bird native to Australia.
Observation: The query copies the surface clue "Emus" and retrieves bird/Emu-War evidence, not the intended acronym bridge.
Query 2: *when was conscription introduced in Australia*
Output:
Doc 1: *Conscription in Australia*. The page gives a broad history of mandatory military service across multiple periods.
Doc 2: The evidence mentions early defence legislation and later national-service schemes.
Query 3: *when was conscription introduced in Australia during WWI*
Query 4: *exact year of conscription introduced in Australia WWI*
Failure: The model has moved from the Vietnam-War setting to WWI and keeps searching variants of the wrong temporal frame.

– –

SEARCHSKILL Trace (SkillBank + Retriever)
Selection: `<select_skill>bridge-entity-search</select_skill>`
Skill meaning: first recover the hidden bridge entity, then ask the final attribute question using that resolved entity.

Action 1:
`<skill>bridge-entity-search</skill>`
`<search>"The Things They Carried" emus country</search>`
Output:
Doc 1: *The Things They Carried* is a collection of linked stories about American soldiers in the Vietnam War.
Doc 2: The page anchors the setting to Vietnam rather than a generic animal clue.
State update: the relevant war setting is Vietnam.

Action 2:
`<skill>bridge-entity-search</skill>`
`<search>"Emu" Vietnam War country</search>`
Output:
Doc 1: *Experimental Military Unit*. EMU was an Australian-American helicopter assault force during the Vietnam War.
Doc 2: The unit was created after a U.S. request for Australia to send more helicopter pilots to the conflict.
State update: EMU $\rightarrow$ Australia.

Action 3:
<skill>bridge-entity-search</skill>
<search>Australia conscription introduced</search>
Output:
Doc 1: *Conscription in Australia*. The national-service scheme relevant to the Vietnam era was introduced in 1964.
Doc 2: The page distinguishes Vietnam-era national service from earlier home-defence or WWI debates.
Final:
<skill>verbatim-evidence-span|bridge-entity-search</skill>
<answer>1964</answer>
Why this case matters: SearchSkill does not ask the retriever to solve the whole riddle. It turns a misleading lexical cue into a controlled bridge chain: *The Things They Carried* → Vietnam War → EMU → Australia → conscription date.

H.2 Case 2: Decomposing a broad mayor question

Question: Who was the first African American mayor of the most populous city in the United States?
Ground Truth: David Dinkins
Search-R1 Answer: malformed </search>
SearchSkill Answer: <answer>David Dinkins</answer>

- -

Search-R1 Trace
Query 1: *first African American mayor of the most populous city in the United States*
Output:
Doc 1: *Doris A. Davis*. Davis was a mayor of Compton and is described as the first African-American woman mayor of a metropolitan city.
Doc 2: Other returned snippets discuss African-American mayors but do not first resolve the target city.
Observation: The first query is too broad; the retriever matches the "first African American mayor" phrase without grounding "most populous city".
Query 2: *most populous city in the United States*
Output:
Doc 1: *New York City*. New York City is the most populous city in the United States.
Query 3: *first African American mayor of New York City*
Output:
Doc 1: an unrelated mayor result appears before the correct David Dinkins evidence.
Query 4: *when did David Dinkins become mayor of New York City*
Query 5: *who was the first African American mayor of the most populous city in the United States*
Failure: Even after finding the right entity, the trace keeps querying and ends without a clean answer.

- -

SearchSkill Trace (SkillBank + Retriever)
Selection: <select_skill>relation-chain-decomposition</select_skill>
Skill meaning: resolve each relation in order and carry the intermediate entity into the next query.

Action 1:
<skill>relation-chain-decomposition</skill>
<search>most populous city in the United States</search>
Output:
Doc 1: *New York City*. The city of New York is identified as the most populous city in the United States.
Doc 2: Additional snippets confirm New York City's population ranking.
State update: target city = New York City.

Action 2:
<skill>relation-chain-decomposition</skill>
<search>New York City first African American mayor</search>
Output:
Doc 1: *David Dinkins*. David Norman Dinkins served as the 106th Mayor of New York City from 1990 to 1993.
Doc 2: The same passage states that he was the first African American to hold that office.
Final:
<skill>verbatim-evidence-span|relation-chain-decomposition</skill>
<answer>David Dinkins</answer>
Why this case matters: The SkillBank-selected decomposition prevents a single overloaded query from matching the wrong "first mayor" fact. The retriever receives one query for the city and one query for the city-specific mayor relation.

H.3 Case 3: Avoiding a plausible but wrong president

Question: When did the president who set the precedent of a two term limit enter office?
Ground Truth: April 30, 1789
Search-R1 Answer: 1933
SearchSkill Answer: <answer>April 30, 1789</answer>

- -

Search-R1 Trace
Query 1: *president who set the precedent of a two term limit*
Output:
Doc 1: *Term limits in the United States*. The page explains the two-term tradition and also notes that Franklin D. Roosevelt broke the tradition by winning a third and fourth term.
Doc 2: Other snippets discuss constitutional term limits after Roosevelt.
Observation: The evidence contains both the original precedent and the later exception; the model latches onto Roosevelt.
Query 2: *Franklin D. Roosevelt entered office year*
Output:
Doc 1: *First and second terms of Franklin D. Roosevelt*. Roosevelt's first term began in 1933.
Failure: The second query is precise but follows the wrong entity. The error happens at the entity-selection step, not at answer extraction.

- -

SearchSkill Trace (SkillBank + Retriever)
Selection: <select_skill>relation-chain-decomposition</select_skill>
Skill meaning: first identify the entity satisfying the description, then retrieve the requested attribute.

Action 1:
<skill>relation-chain-decomposition</skill>
<search>two-term-limit president</search>
Output:
Doc 1: *Term limits in the United States*. The tradition of not seeking a third term is associated with George Washington.
Doc 2: The same evidence separates Washington's precedent from Roosevelt's later decision to break it.
State update: target president = George Washington.

Action 2:
<skill>relation-chain-decomposition</skill>
<search>George Washington entered office</search>
Output:
Doc 1: *Presidency of George Washington*. Washington's presidency began on April 30, 1789.

Doc 2: The passage also gives the end date, March 4, 1797, confirming the presidential-office context.
Final:
<skill>verbatim-evidence-span|relation-chain-decomposition</skill>
<answer>April 30, 1789</answer>
Why this case matters: The skill forces an explicit entity-resolution state before asking for the date. This avoids a common retrieval trap where the top evidence contains a salient but wrong entity.

H.4 Case 4: Carrying an intermediate entity across hops

Question: Where was the place of death of Suzanne Farrington's mother?
Ground Truth: London
Search-R1 Answer: malformed <search>Vivien Leigh death place</search>
SearchSkill Answer: <answer>London</answer>

Search-R1 Trace
Query 1: *Suzanne Farrington*
Output:
Doc 1: *Suzanne Farrington*. The passage describes Farrington and includes facts about her own death in Wiltshire.
Doc 2: Additional snippets discuss her family and career context.
Observation: The first retrieval is useful for identifying the person, but it also contains a distractor death location for Farrington herself.
Query 2: *Vivien Leigh death place*
Output:
Doc 1: *Vivien Leigh*. The passage contains the relevant death-context evidence and central-London theatre references.
Query 3: *Vivien Leigh funeral location*
Query 4: *Vivien Leigh place of death*
Query 5–8: repeated variants of *Vivien Leigh death place*.
Failure: The trajectory already has the right intermediate entity and enough evidence, but continues to search and finally outputs a search tag rather than the answer.

SearchSkill Trace (SkillBank + Retriever)
Selection: <select_skill>relation-chain-decomposition</select_skill>
Skill meaning: follow the relation chain *Suzanne Farrington* → *mother* → *place of death*.

Action 1:
<skill>relation-chain-decomposition</skill>
<search>Suzanne Farrington mother</search>
Output:
Doc 1: *Suzanne Farrington*. The passage links Suzanne Farrington to Vivien Leigh as her mother.
Doc 2: The surrounding evidence distinguishes Farrington's own biography from the target relation.
State update: target mother = Vivien Leigh.

Action 2:
<skill>relation-chain-decomposition</skill>
<search>Vivien Leigh place of death</search>
Output:
Doc 1: *Vivien Leigh*. The passage describes the circumstances around Leigh's death and links the event to London.
Doc 2: Related snippets discuss funeral arrangements and central-London theatre tributes.
State update: candidate answer = London.

Action 3:
<skill>relation-chain-decomposition</skill>

<search>Vivien Leigh death place London</search>
Output:
Doc 1: The retrieved passage verifies the same entity and answer span, supporting London as the place of death.
Final:
<skill>verbatim-evidence-span|relation-chain-decomposition</skill>
<answer>London</answer>
Why this case matters: The model keeps the intermediate entity explicit, verifies the answer span, and stops instead of repeatedly searching around the same evidence.

H.5 Case 5: Joining an in-law relation

Question: Who is Maria Luisa Of Savoy's father-in-law?
Ground Truth: Louis, Grand Dauphin
Search-R1 Answer: Louis XIV
SearchSkill Answer: <answer>Louis, Grand Dauphin</answer>

Search-R1 Trace
Query 1: *Maria Luisa of Savoy husband*
Output:
Doc 1: *Maria Luisa of Savoy*. The passage states that Maria Luisa was queen consort of Spain by marriage to Philip V of Spain.
Query 2: *Philip V of Spain father*
Output:
Doc 1: *Philip V of Spain*. The passage states that Philip's father was Louis, Grand Dauphin, and also mentions that Philip was a grandson of King Louis XIV.
Query 3: *Louis Grand Dauphin father*
Failure: The trajectory follows one edge too far and answers Louis XIV, the father of the father-in-law, rather than the father-in-law itself.

SearchSkill Trace (SkillBank + Retriever)
Selection: <select_skill>derived-kinship-inference-join</select_skill>
Skill meaning: compose an in-law relation by joining immediate spouse evidence with the spouse's parent evidence, then stop at the derived role.

Action 1:
<skill>derived-kinship-inference-join</skill>
<search>Maria Luisa of Savoy husband</search>
Output:
Doc 1: *Maria Luisa of Savoy*. Maria Luisa was queen consort of Spain by marriage to Philip V of Spain.
State update: husband = Philip V of Spain.

Action 2:
<skill>derived-kinship-inference-join</skill>
<search>Philip V of Spain father</search>
Output:
Doc 1: *Philip V of Spain*. The passage states that Philip's father was Louis, Grand Dauphin.
State update: father-in-law = Louis, Grand Dauphin.

Final:
<skill>verbatim-evidence-span|derived-kinship-inference-join</skill>
<answer>Louis, Grand Dauphin</answer>
Why this case matters: The SkillBank makes the join boundary explicit. Search-R1 retrieves the

needed evidence but continues to a neighboring family edge; SearchSkill stops at the composed in-law relation.

H.6 Case 6: Comparing matched attributes

Question: Who died first, Olaf Everson or Abdul Wahab Khan Tarzi?
Ground Truth: Abdul Wahab Khan Tarzi
Search-R1 Answer: Olaf Everson
SearchSkill Answer: <answer>Abdul Wahab Khan Tarzi</answer>

Search-R1 Trace
Query 1: *Olaf Everson date of death*
Output:
Doc 1: The top result is *Olaf Skoogfors*, an unrelated Olaf entity, and says he died in 1975.
Query 2: *Abdul Wahab Khan Tarzi date of death*
Output:
Doc 1: *Abdul Wahab Khan Tarzi*. The passage gives 20 January 1994 as the death date.
Query 3–4: repeated variants of *Olaf Everson date of death exact*.
Failure: The comparison is corrupted by an entity mismatch on the Olaf side, so the final answer chooses the wrong option.

SearchSkill Trace (SkillBank + Retriever)
Selection: <select_skill>parallel-attribute-compare</select_skill>
Skill meaning: retrieve the same attribute for both candidate entities, check entity identity, and compare the matched values.

Action 1:
<skill>parallel-attribute-compare</skill>
<search>Olaf Everson death date</search>
Output:
Doc 1: The top evidence is noisy and points to an Olaf entity with a 1975 death date, requiring caution before comparison.
State update: candidate date for Olaf side is uncertain/noisy.

Action 2:
<skill>parallel-attribute-compare</skill>
<search>Abdul Wahab Khan Tarzi death date</search>
Output:
Doc 1: *Abdul Wahab Khan Tarzi*. The passage gives 24 November 1903–20 January 1994.
State update: Tarzi death date = 20 January 1994.

Action 3:
<skill>parallel-attribute-compare</skill>
<search>Olaf Everson death date</search>
Output:
Doc 1: The repeated evidence remains about a different Olaf entity, so the model does not treat it as a clean support for Olaf Everson.
Final:
<skill>answer-grounding-check|parallel-attribute-compare</skill>
<answer>Abdul Wahab Khan Tarzi</answer>
Why this case matters: The skill frames the problem as matched-attribute comparison rather than independent lookup. This makes entity mismatch visible before the final comparison.

H.7 CASE 7: RESOLVING A SUPERLATIVE TARGET

Question: What was the death toll of the second largest volcanic eruption in the 20th century?
Ground Truth: 847
Search-R1 Answer: 847
SEARCHSKILL Answer: <answer>847</answer>

- -

Search-R1 Trace
Query 1: *second largest volcanic eruption in the 20th century*
Output:
Doc 1: *Ring of Fire*. The passage identifies the 1991 eruption of Mount Pinatubo as the world's second-largest terrestrial eruption of the 20th century.
Query 2: *death toll of Mount Pinatubo eruption*
Output:
Doc 1: *Mount Pinatubo*. The passage reports that 847 people were killed by the eruption.
Observation: Search-R1 also answers correctly, but the trajectory treats the superlative resolution as ordinary query rewriting.

- -

SEARCHSKILL Trace (SkillBank + Retriever)
Selection: <select_skill>superlative-ranking-match</select_skill>
Skill meaning: first resolve the ranked entity under the specified scope, then retrieve the requested attribute for that entity.

Action 1:
<skill>superlative-ranking-match</skill>
<search>second largest volcanic eruption 20th century</search>
Output:
Doc 1: *Ring of Fire*. The passage identifies the 1991 Mount Pinatubo eruption as the world's second-largest terrestrial eruption of the 20th century.
State update: ranked target = Mount Pinatubo eruption.

Action 2:
<skill>superlative-ranking-match</skill>
<search>Mount Pinatubo death toll</search>
Output:
Doc 1: *Mount Pinatubo*. The passage states that a reported 847 people were killed by the eruption.
Final:
<skill>verbatim-evidence-span|superlative-ranking-match</skill>
<answer>847</answer>
Why this case matters: This case shows a non-bridge skill: SearchSkill exposes the superlative resolution step explicitly before asking for the final attribute, making the query plan easier to inspect even when a baseline also reaches the answer.
