Enhancing Search-Tool Policies for Small LLMs via Process Distillation and Post-Distillation RL

Yuyang Bai ybai
Shuning Gu shuning
Zhuofeng Li zhuofengli
Yi Wen cyberwenyi2357

Abstract

This document is a supplement to the general instructions for *ACL authors. It contains instructions for using the LaTeX style files for ACL conferences. The document itself conforms to its own specifications, and is therefore an example of what your manuscript should look like. These instructions should be used both for papers submitted for review and for final versions of accepted papers.

1 Introduction

Large language models (LLMs) increasingly rely on external tools such as search engines to answer knowledge-intensive questions. For small LLMs (e.g., $\approx$ 3B parameters), learning when to call a search tool and whether search is needed is particularly challenging due to their limited parametric knowledge and weaker planning ability. Enabling these models to learn effective search-tool policies that determine if, when, and how they should retrieve external evidence—is therefore essential for democratizing search-augmented LLM capabilities (Alzubi et al., 2025).

Prior approaches like ReAct (Yao et al., 2023) and Toolformer (Schick et al., 2023) encourage tool use, and RL methods like Search-R1 (Jin et al., 2025) teach search behavior through reward. However, RL-first pipelines try to learn tool use, reasoning, and decision timing all at once, leading to sample inefficiency and unstable behavior. What is missing is an explicit decision prior before RL —something that tells the model when searching is warranted and why.

This gap motivates our central research question: Can small LLMs learn effective search-tool policies by first acquiring structured reasoning traces and only then refining the policy with reinforcement learning—outperforming both distillation-only and RL-only paradigms?

To investigate this question, we propose a three-stage training regimen. First, we curate teacher-generated reasoning trajectories on Natural Questions that explicitly annotate <think>, <search>, <information>, and <answer> actions, providing a clean process prior for tool decision-making. Second, we perform process-supervised fine-tuning (SFT) using LoRA to distill these iterative reasoning behaviors into the student model. This step already teaches the model how to conduct a multi-step search process and substantially improves its ability to decide when search is necessary. Finally, we apply post-distillation reinforcement learning using PPO in a controlled search environment to refine the agent's decision boundaries for search timing, stopping conditions, and answer production. This SFT$\rightarrow$RL pipeline stabilizes early exploration, reduces wasted search calls, and improves reward-aligned behaviors such as accuracy and efficient evidence use.

Together, these contributions demonstrate that small LLMs benefit substantially from process priors before RL, yielding more reliable, cost-efficient, and higher-accuracy search-tool policies than existing approaches.

2 Related Work

Prior work has prompted tool use by coupling reasoning with actions (e.g., ReAct) (Yao et al., 2023) and has explored self-supervised generation of tool-use data (e.g., Toolformer) (Schick et al., 2023). RL methods such as *Proximal Policy Optimization (PPO)* (Schulman et al., 2017) have been applied to interleaved tool-use settings, and recent work such as *Search-R1* trains LLMs to reason and search jointly (Jin et al., 2025). *Process supervision* and *knowledge distillation* are established techniques for transferring guidance from stronger to weaker models (Hinton et al., 2015).
What remains insufficiently addressed is an explicit

process-level prior on the *decision to search* that is injected *before* outcome-based RL. Existing RL-first pipelines typically rely on sparse rewards to teach both content generation and tool operation simultaneously, which can be sample-inefficient and brittle. It is therefore still unclear whether pre-RL process distillation of search decisions measurably improves sample efficiency, query quality, and the overall cost–quality trade-off.

3 Supervised Fine-tuning+Reinforcement Learning pipeline

3.1 SFT

3.1.1 Objective

The goal of the SFT stage is to teach the student model to perform **iterative reasoning and search actions** rather than generating a full answer in a single pass. Specifically, the model must learn to (1) produce a `<think>` step, (2) decide when a `<search>` action is needed, (3) integrate retrieved `<information>` blocks, and (4) eventually output an `<answer>` once sufficient evidence is accumulated.

3.1.2 Data Preparation

The teacher data consists of full multi-step trajectories with explicitly annotated `<think>`, `<search>`, `<information>`, and `<answer>` tags. To enable step-wise supervision, we convert each trajectory into multiple training samples. For a trajectory of the form:

`<think>` T_1 `</think>`

`<search>` S_1 `</search>`

`<information>` I_1 `</information>`

`<think>` T_2 `</think>`

`<answer>` A `</answer>`

we generate two SFT samples:

- **Sample 1:** Instruction contains the system prompt and the user question; Output is the next action `<think>` T_1 `</think>` `<search>` S_1 `</search>`.
- **Sample 2:** Instruction additionally includes the history (`<think>` T_1 `</think><search>` S_1 `</search><information>` I_1 `</information>`); Output is the next-step prediction `<think>` T_2 `</think><answer>` A `</answer>`.

Each resulting example follows the Alpaca format used by `LLaMA-Factory`, where the `instruction` contains the system prompt, question, and all past reasoning/search steps, `input` is empty, and `output` contains only the next model action. This preprocessing transforms each trajectory into several supervised samples, yielding the iterative dataset `iterative_sft_data.json`.

3.1.3 Training Configuration

We fine-tune `Qwen2.5-3B-Instruct` using LoRA within the `LLaMA-Factory` framework. LoRA modules are applied to all linear layers to efficiently adapt the model. Training was performed on an A100 GPU and converged to a loss of approximately 0.89.

3.1.4 Outcome

After SFT, the student model learns a stable process prior: it can generate step-wise `<think>` reasoning, invoke `<search>` when needed, incorporate external `<information>`, and output `<answer>` once sufficient evidence is collected. This checkpoint serves as the initialization for the RL stage, which further refines **when** to search, **how long** to search, and **when** to stop and answer.

3.2 RL

We further refine the student model through reinforcement learning using the **Search-R1** (Jin et al., 2025) framework built on the VeRL infrastructure. The goal of this stage is to optimize the model's decision policy regarding *when to search*, *how long to search*, and *when to stop and answer*, starting from the process-supervised SFT initialization.

3.2.1 Training Configuration

RL training is conducted on the Natural Questions (NQ) dataset (Kwiatkowski et al., 2019)(train split) using Proximal Policy Optimization (PPO). The agent interacts with a retrieval-augmented environment in a multi-step setting: at each turn, it may generate a `<search>` query, receive retrieved `<information>`, continue `<think>` reasoning, or output the final `<answer>`. Rewards are outcome-based and computed using Exact Match (EM). A local retriever provides top-$k = 3$ documents for each search query.

3.2.2 Hyperparameters

Table 1 summarizes the PPO hyperparameters used in training.

Parameter	Value
Total training steps	600
Batch size	32
Mini-batch size	32
Micro-batch size	8
Learning rate (actor)	1×10^{-6}
Learning rate (critic)	1×10^{-5}
KL coefficient	0.001
Max turns (train)	2
Max prompt length	4096
Max response length	500
Advantage estimator	GAE

Table 1: PPO hyperparameters used for RL training.

4 Evaluation

We evaluate the RL-trained `Qwen2.5-3B-Instruct` model on the NQ test split using a multi-turn search-and-generation process.

4.1 Metrics

We evaluate model performance using two standard metrics in open-domain question answering: **Exact Match (EM)** and **ROUGE-L**.

Exact Match (EM). EM measures whether the model's predicted answer string exactly matches the gold answer. This metric is strict and reflects whether the model recovers the precise answer required by the dataset.

ROUGE-L. ROUGE-L computes the longest common subsequence (LCS) between the prediction and reference, capturing token-level recall and precision of their overlap. Given prediction $\hat{y}$ and reference y with LCS length $\mathrm{LCS}(\hat{y}, y)$, the precision and recall are defined as:

$$P = \frac{\mathrm{LCS}(\hat{y}, y)}{|\hat{y}|}, \qquad R = \frac{\mathrm{LCS}(\hat{y}, y)}{|y|}.$$

ROUGE-L combines them using an F-score:

$$\mathrm{ROUGE{-}L} = \frac{(1+\beta^2)\, PR}{R + \beta^2 P},$$

where β is typically set to 1 to weight precision and recall equally. ROUGE-L reflects semantic similarity and partial correctness even when the prediction does not fully match the gold answer.

4.2 Baselines

We compare our approach against three alternative configurations:

1. **Baseline**: `Qwen2.5-3B-Instruct` (no SFT, no RL).
2. **SFT Only**: `Qwen2.5-3B-Instruct` + supervised fine-tuning (SFT).
3. **RL Only**: `Qwen2.5-3B-Instruct` + reinforcement learning (RL), using the publicly available checkpoint `PeterJinGo/SearchR1-nq_hotpotqa_train-qwen2.5-3b-it-em-ppo`.

5 Experiment

5.1 Setting

Table 2 lists the decoding and interaction settings used during evaluation.

Parameter	Value
Temperature	0.7
Top-p	0.9
Max turns (eval)	5
Retrieval top-k	3
Max new tokens	1024

Table 2: Inference and evaluation parameters.

5.2 Results

We compare four configurations of our system to evaluate the contributions of SFT and RL individually as well as their combined effect. All metrics are reported on the NQ-Test split.

Group	Configuration	EM	ROUGE-L
1	Baseline (no SFT, no RL)	0.2374	0.3179
2	SFT Only	0.3654	0.4515
3	RL Only	0.3108	0.3906
4	SFT + RL (Ours)	0.4127	0.4890

Table 3: Comparison of four training configurations on NQ-Test.

Overall performance. The combined approach (**SFT + RL**) achieves the best performance across both EM and ROUGE-L, outperforming all other configurations by a substantial margin. This demonstrates that reinforcement learning is most effective when applied on top of a strong process-supervised SFT initialization rather than as a standalone method.

Effect of SFT. SFT alone significantly improves performance over the baseline (**+12.8 EM**, **+13.4 ROUGE-L**), highlighting the importance of learning structured search-and-answer protocols prior to interacting with the RL environment. The baseline and RL-only models lag notably behind, suggesting that exploration alone is insufficient for recovering procedural knowledge such as multi-step reasoning, consistent formatting, and appropriate search invocation.

Effect of RL. Adding RL on top of SFT yields an additional **+4.7 EM** and **+3.7 ROUGE-L** over the SFT-only model. PPO fine-tuning sharpens the agent's decision boundary on when to query the retriever, when to refine a query, and when to stop and produce an answer. This indicates that online feedback compensates for coverage limitations in the static supervised corpus and reinforces reward-aligned behaviors that improve final answer quality.

The results suggest a clear division of labor: SFT establishes core process priors for iterative reasoning and tool use, while RL refines these behaviors to maximize task reward in the interactive environment. Their synergy makes the full pipeline substantially stronger than either stage alone.

References

Salaheddin Alzubi, Creston Brooks, Purva Chiniya, Edoardo Contente, Chiara von Gerlach, Lucas Irwin, Yihan Jiang, Arda Kaz, Windsor Nguyen, Sewoong Oh, and 1 others. 2025. Open deep search: Democratizing search with open-source reasoning agents. *arXiv preprint arXiv:2503.20201*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Preprint*, arXiv:1503.02531.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *Preprint*, arXiv:2503.09516.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.