CROSS-AGENT LEARNING SIGNALS ENABLE COORDINATED ROLE-DECOMPOSED LLM TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic search systems must coordinate evidence acquisition and response generation, yet existing approaches either couple both roles under a single agent objective or decompose them without disentangling their respective contributions to the final outcome. We introduce **DAC (Divide and Cooperate)**, a role-decomposed training framework that, given task-specific external verification signals, trains a searcher and a generator with role-specific cross-verification rewards. DAC allows the generator to abstain when the retrieved evidence appears insufficient, and uses this decision together with externally evaluated search sufficiency to assign appropriate credit to each role. To prevent degenerate over-abstention, we further introduce hard-positive evidence augmentation, which discourages abstaining on sufficient but hard evidence. Across seven general and multi-hop QA benchmarks and two model backbones, DAC consistently outperforms strong single-agent and multi-agent baselines. Controlled evaluations show that these gains arise from coordinated improvements in both search and generation. Our results highlight the importance of explicitly and properly assigning credit across interacting roles when training agentic search systems.

1 INTRODUCTION

Large language models (LLMs) increasingly solve knowledge-intensive tasks by actively searching for external information rather than relying solely on parametric knowledge. Building on retrieval-augmented generation and ReAct-style tool use (Yao et al., 2023), recent work trains such *agentic search* behavior with reinforcement learning (RL), from iterative search-and-reasoning agents (Jin et al., 2025b; Song et al., 2025; Chen et al., 2025; Xie et al., 2026) to long-horizon deep-research systems (Nguyen et al., 2025; Li et al., 2025a). A recent direction further decomposes these systems into multiple specialized agents (Jiang et al., 2025; Li et al., 2026; Chen et al., 2026), reflecting the distinct roles involved in acquiring evidence and producing a final response.

However, ***decomposing the roles does not by itself solve how to train them***. While role-decomposed and multi-agent frameworks have demonstrated inference-time gains on agentic tasks, reliably training such systems remains challenging due to error propagation and ambiguous credit assignment (Kim et al., 2026). In particular, the final task reward depends jointly on the quality of retrieved evidence and the generator's ability to use it. Assigning this outcome to both agents therefore creates ambiguous credit; a searcher may be penalized despite retrieving sufficient evidence when the generator fails, while a generator may be penalized for an unanswerable input caused by poor search. Likewise, independently rewarding search sufficiency overlooks that what constitutes useful evidence depends on the evolving capabilities of the generator.

We introduce **DAC (Divide and Cooperate)** (Figure 1), a multi-agent training framework that separates evidence acquisition from response generation while coupling their training through **cross-verification rewards**. DAC gives the generator an additional verification role: it may abstain when the retrieved evidence is insufficient, and this decision is incorporated into the searcher's reward. Consequently, the searcher is only rewarded for evidence that is both sufficient and usable by the current generator, while the generator is rewarded for either answering correctly on sufficient evidence or abstaining when the evidence is insufficient. This aligns their learning, inducing an adaptive curriculum in which search and generation capabilities progressively expand in step with each other, also avoiding penalties caused by failures of the other role.

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.

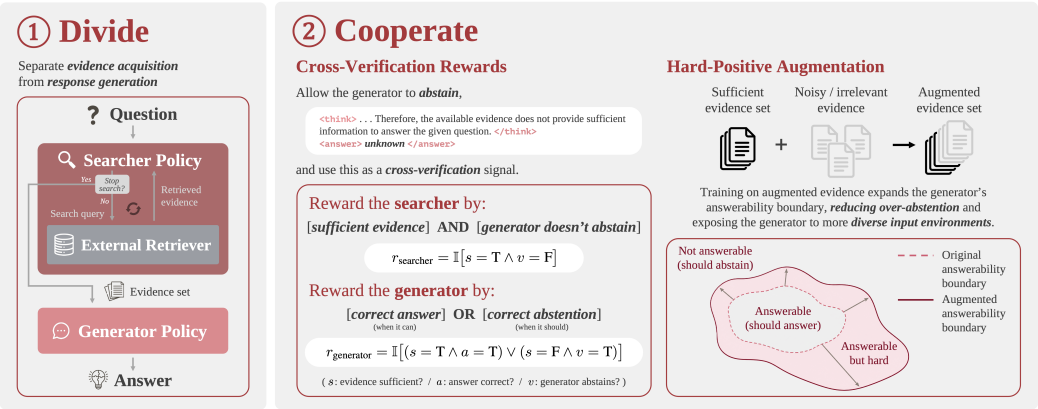


Figure 1: **Overview of DAC.** DAC decomposes agentic search into a searcher and a generator, and trains each with role-specific rewards. Using cross-verification rewards aligns searcher behavior with downstream generator answerability, while separating credits across agents. Hard-positive augmentation expands the generator's answerability boundary and mitigates over-abstention.

Yet allowing abstention introduces its own failure mode: the generator may become overly conservative and reject difficult but still answerable evidence. This resembles over-refusal in refusal-aware instruction tuning (Zhu et al., 2025b;a), where noisy or imbalanced abstention supervision can bias models toward refusing answerable inputs. We interpret this through a game-theoretic lens, suggesting that without enough hard but answerable training samples, the generator's verification boundary can become overly conservative. This motivates **hard-positive evidence augmentation**, which adds irrelevant or weakly related evidence to sufficient contexts, discouraging abstention while promoting correct answers despite the added noise.

We evaluate DAC on seven general and multi-hop question answering (QA) benchmarks. DAC consistently outperforms training-free, single-agent training, and prior multi-agent training approaches. Controlled experiments further disentangle these gains, showing that DAC improves in both search and generation separately. Our analyses show that cross-verification produces more balanced learning dynamics than using external verification signals alone. Furthermore, learned abstention provides a useful inference-time signal: inputs on which the generator abstain have substantially lower answer accuracy, allowing abstention to trigger further search or request user assistance when additional evidence is needed, despite the searcher's sufficiency decision.

Our main contributions are:

- We propose **DAC**, a cooperative training framework for role-decomposed agentic search. Given external search and answer verification, DAC combines these signals with the generator's evolving abstention decision to construct role-specific **cross-verification rewards**. This provides an adaptive curriculum for joint training, matching search capability to the generator's evolving answering capability while disentangling credit between search and generation.
- We analyze the training dynamics of this cooperative interaction through a theoretical lens, identifying over-abstention as a key failure mode and introducing **hard-positive evidence augmentation** to encourage the generator to accept and correctly answer on harder sufficient evidence.
- We evaluate DAC across **seven QA benchmarks** with two LLM backbones. DAC consistently outperforms strong single-agent and prior multi-agent baselines, while controlled analysis shows that the gains come from improvements in **both search quality and answer generation**.

2 Problem Setting: Agentic Search

We define **agentic search** as a broad class of tasks in which an LLM agent first acquires task-relevant evidence and then uses that evidence to produce a final response. This differs from interaction-centric tasks where the primary challenge is acting within an external environment and the final response mainly summarizes the completed actions. We decompose agentic search into two sequentially interacting roles, **evidence acquisition** and **response generation**. These roles may be implemented using either a shared policy or separately parameterized policies.

Under this decomposition, the agentic task proceeds as follows:

1. A task specification Q, also called a **question**, is provided to the system. The system consists of two agents: a **searcher agent** $\mathcal{S}$ and a **generator agent** $\mathcal{G}$.
2. $\mathcal{S}$ performs iterative evidence acquisition. At each iteration t, it observes Q and the **evidence** collected through iteration $t-1$, $\mathcal{E}_{t-1} = \bigcup_{i<t} e_i$, with initial condition $t = 1$ and $\mathcal{E}_0 = \varnothing$. Based on this information, $\mathcal{S}$ either **[continues the search loop]** or, if it determines that the collected evidence is sufficient, stops the search loop and sets the final accumulated evidence set $\mathcal{E} = \mathcal{E}_{t-1}$ before **[passing evidence to the generator]**.
3. **[Continue Searching]** $\mathcal{S}$ generates a query q_t and pass it to an external tool, which returns the search results that $\mathcal{S}$ processes into step-level evidence e_t. This evidence is accumulated into the evidence set, $\mathcal{E}_t = \mathcal{E}_{t-1} \cup e_t$. The process then returns to step 2 with $t \leftarrow t + 1$.
4. **[Pass Evidence to the Generator]** $\mathcal{G}$ receives the original question Q together with the final accumulated evidence set $\mathcal{E}$ and produces the final **answer** $\mathcal{A} = \mathcal{G}(Q, \mathcal{E})$.

We deliberately define **evidence** broadly as the accumulated task-relevant context rather than restricting it to raw tool outputs. Thus, $\mathcal{E}_t$ may include extracted facts, summaries, intermediate reasoning, or other processed information that the searcher and generator can utilize, depending on the task.

3 DAC

We introduce **DAC (Divide and Cooperate)**, a multi-agent training framework that optimizes evidence acquisition and response generation separately. DAC enables joint RL training with role-specific rewards that better address credit assignment between search and generation.

3.1 Reward design for joint optimization

Under the two-stage formulation, we aim to jointly optimize $\mathcal{S}$ and $\mathcal{G}$ using RL. The final objective of agentic search is the quality of the final response. For example, in question answering, this can be instantiated as the exact match (EM) score of the generated answer, as in Jin et al. (2025b).

A straightforward design is to assign this *answer-level reward* to both agents, but this creates a credit assignment problem, when one agent is penalized for the other's failure. An alternative is to reward the searcher directly using an external *evidence-sufficiency signal*. However, this signal alone does not indicate whether the evolving generator can utilize the retrieved evidence well. Directly optimizing external sufficiency may therefore improve retrieval faster than the generator's ability to exploit it. Moreover, it does not address how to avoid penalizing the generator when retrieval is insufficient.

To address this issue, we employ **cross-verification rewards**, by giving the generator an additional verification role. The generator is allowed to *abstain* when it judges that the provided evidence is insufficient. This abstention decision is then incorporated into the searcher's reward. At the same time, the generator is rewarded not only for answering correctly, but also for correctly abstaining when the evidence is indeed insufficient according to an external verifier. More formally, let

$$\begin{aligned} s(Q, \mathcal{E}) &= \mathbb{I}\big[\text{evidence } \mathcal{E} \text{ is sufficient to answer question } Q\big], \\ v(\mathcal{A}) &= \mathbb{I}\big[\text{generation } \mathcal{A} \text{ is an abstention}\big], \\ a(Q, \mathcal{A}) &= \mathbb{I}\big[\text{generation } \mathcal{A} \text{ is a qualified answer to question } Q\big], \end{aligned}$$

where s and a are task-specific signals judged by an external verifier.

Then, we may define the searcher and generator rewards as:

$$r_{\mathcal{S}}(Q, \mathcal{E}, \mathcal{A}) = s(Q, \mathcal{E})\big(1 - v(\mathcal{A})\big), \tag{1}$$

$$r_{\mathcal{G}}(Q, \mathcal{E}, \mathcal{A}) = s(Q, \mathcal{E})a(Q, \mathcal{A}) + \big(1 - s(Q, \mathcal{E})\big)v(\mathcal{A}) = \begin{cases} a(Q, \mathcal{A}), & \text{if } s(Q, \mathcal{E}) = 1, \\ v(\mathcal{A}), & \text{if } s(Q, \mathcal{E}) = 0. \end{cases} \tag{2}$$

Thus, the searcher is rewarded for retrieving evidence that is both externally sufficient and deemed sufficient by the generator, avoiding penalties due to generator-side answer failures. The generator is rewarded for either answering correctly on sufficient evidence or abstaining on insufficient evidence.

Both agents ultimately optimize the external verification criteria, regardless of whether cross-verification is used (see section A). Cross-verification mainly shapes and aligns how they reach this solution. For the searcher, it favors sufficient search modes that the current generator can likely use, providing a more focused exploration path through the multimodal exploration space. For the generator, the searcher defines its evolving RL environment; through abstention, the generator steers this environment toward evidence better matched to its current capability, making each intermediate learning problem easier. This interaction induces a mutually adaptive curriculum for both agents.

3.2 Mitigating over-abstention with hard-positive evidence augmentation

A predictable failure mode of our framework is *over-abstention*. Because abstaining on insufficient evidence provides a relatively direct reward signal, whereas obtaining reward on sufficient evidence requires the generator to both accept the input and answer correctly, training can become biased toward a conservative abstention behavior when hard-positive examples are scarce. This can hurt even when abstention is disabled at test time; if the generator was too heavily biased toward abstention during training, it may not learn to answer well from hard, noisy, or imperfect evidence.

To mitigate this, we must pressure the generator to expand its answerable region rather than accepting only clean sufficient evidence. We therefore augment generator training with **hard-positive evidence** samples. For each sufficient pair $(Q, \mathcal{E})$ with $s(Q, \mathcal{E}) = 1$, we add irrelevant or weakly related evidence $\mathcal{E}_N$ to form $\mathcal{E}' = \mathcal{E} \cup \mathcal{E}_N$, as illustrated in Figure 1. Since $\mathcal{E}$ remains sufficient, this creates harder positive examples that encourage the generator to expand its answerable region rather than abstaining on non-ideal evidence. This mitigation is also motivated by a game-theoretic intuition. We provide a theoretical interpretation of this effect in section 6.5.

3.3 Turn-level difference rewards for searcher training

We also use temporal-difference turn-level rewards, following prior work (Xie et al., 2026; Zhang et al., 2026; Chen et al., 2026). Since the final search reward provides only trajectory-level feedback, we obtain a denser signal by evaluating the reward change after each search turn. When $\mathcal{E}_t$ denotes the evidence collected up to turn t, we run the generator on each intermediate evidence state and define the turn-level reward as the difference

$$s_{\mathcal{S}}^{(t)} = r_{\mathcal{S}}\big(Q, \mathcal{E}_t, \mathcal{G}(Q, \mathcal{E}_t)\big), \qquad r_{\mathcal{S}}^{(t)} = s_{\mathcal{S}}^{(t)} - s_{\mathcal{S}}^{(t-1)}.$$

Thus, turns are rewarded according to how they change the usefulness of the accumulated evidence. While prior methods rely on potentially unreliable metrics such as model perplexity to estimate $s_{\mathcal{S}}^{(t)}$, we benefit from using a decomposed design as it directly enables such intermediate-state feedback.

4 Experiments

We experiment on knowledge-based QA tasks. The system is equipped with a dense-retrieval system as the external tool. Full implementation details are provided in Appendix B.

Trajectory management We maintain evidence as the unprocessed searcher interaction trajectory; following Jin et al. (2025b), this is serialized as repeated `<think>...</think>`, `<search>...</search>`, and `<information>...</information>` segments, containing reasoning content, search queries, and retrieved documents, respectively. The loop terminates when the searcher emits `<stop>` after `</think>`. During training, `information` blocks are treated as environment outputs and masked out. For the generator, we pass only the retrieved documents as evidence, which keeps the input format simple and makes noise-based evidence augmentation straightforward.

Verifier instantiation DAC does not prescribe a particular implementation of the external verification signals. In our training runs, we use EM-based verification for both answer correctness and search sufficiency. Specifically, for search, we use substring EM (Zhao et al., 2024; Jin et al., 2025a), where the search result is considered sufficient if at least one document contains or the generated answer matches the gold answer. This is a lightweight choice for computational efficiency. Other semantic, learned, or task-specific verifiers can be substituted without changing the DAC objective.

Training We conduct experiments with two backbone models: `Qwen2.5-7B-Instruct` (Qwen Team, 2025a) and `Qwen3-8B (Non-Thinking)` (Qwen Team, 2025b). For each backbone, the

Table 1: **Exact match (EM) on QA benchmarks.** Datasets marked with * are in-domain datasets. **Bold** and underlined numbers indicate the best and second best results for each model, respectively.

Type	Method	General QA			Multi-Hop QA				Average
		NQ*	TriviaQA	PopQA	HotpotQA*	2Wiki	MuSiQue	Bamb.	
`Qwen2.5-7B-Instruct`									
Training-free	Direct	18.2	44.8	16.5	19.1	24.6	4.3	9.6	19.6
	CoT	19.3	47.0	16.0	20.8	21.2	5.3	25.6	22.2
	Naive RAG	36.9	58.9	38.9	29.3	17.9	6.4	24.0	30.3
	IRCoT	16.6	29.8	20.7	18.7	19.5	4.8	15.2	17.9
	Search-o1	31.3	52.3	33.6	29.0	26.0	11.6	33.6	31.1
Single-agent	Search-R1	**43.9**	62.0	42.6	38.5	34.8	15.6	40.0	39.6
	TIPS	39.8	62.2	42.2	37.9	36.6	13.4	35.2	38.2
Multi-agent	S3	7.3	36.0	19.0	17.3	12.6	4.0	1.6	14.0
	DART	42.4	61.6	41.0	39.9	**40.6**	15.7	40.8	40.3
	M-ASK	33.8	60.1	36.8	37.4	31.8	12.1	33.6	35.1
Multi-agent	**DAC**	**43.9**	**63.4**	**43.8**	**43.2**	39.8	**19.0**	**43.2**	**42.3**
`Qwen3-8B (Non-Thinking)`									
Training-free	Direct	16.3	42.4	15.4	18.2	25.2	3.4	10.4	18.8
	CoT	19.9	45.3	15.9	21.3	25.5	5.8	32.8	23.8
	Naive RAG	35.9	58.5	38.7	29.0	19.3	7.0	26.4	30.7
	IRCoT	35.6	57.9	36.3	36.9	26.3	12.0	36.8	34.4
	Search-o1	32.8	49.9	35.7	29.9	31.7	11.3	32.0	31.9
Single-agent	Search-R1	48.1	66.1	45.3	45.0	43.9	21.7	48.8	45.6
	TIPS	39.7	62.5	41.9	38.3	41.6	16.9	44.8	40.9
Multi-agent	S3	12.2	47.4	29.5	22.9	16.7	3.8	17.6	21.4
	DART	46.8	66.3	47.3	45.3	44.5	20.9	48.0	45.6
	M-ASK	33.5	64.1	38.0	41.8	37.4	13.8	36.0	37.8
Multi-agent	**DAC**	**48.4**	**67.2**	**47.9**	**46.6**	**48.7**	**22.3**	**49.6**	**47.2**

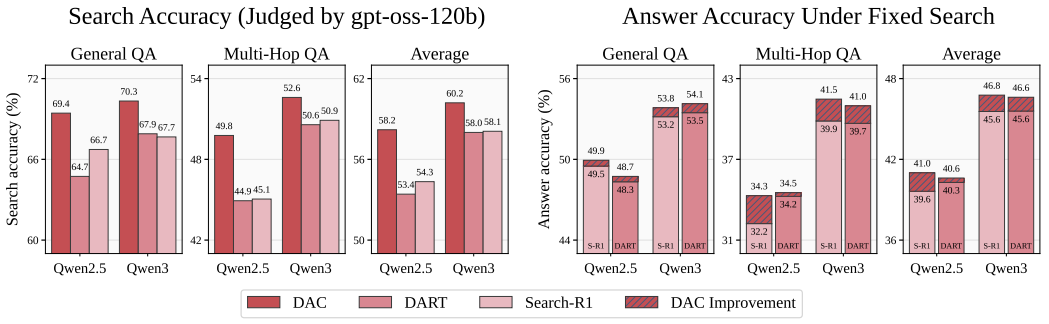


Figure 2: **Disentangling gains in search and generation.** *Left*: search accuracy of the retrieved evidence, judged by `gpt-oss-120b`. *Right*: answer EM when DAC generates from the same search results each produced by Search-R1 or DART, respectively. Hatched regions indicate DAC's improvement over each corresponding baseline.

searcher and generator are implemented using a single shared model. Their RL losses are computed separately and summed into a single objective used to update the shared parameters. We merge the training splits of NQ (Kwiatkowski et al., 2019) and HotpotQA (Yang et al., 2018) to construct the training dataset.

Benchmarks We evaluate on seven benchmarks. For general QA, we use NQ, TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023). For multi-hop QA, we use HotpotQA, 2WikiMultihopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). During DAC evaluation, generator abstention is disabled.

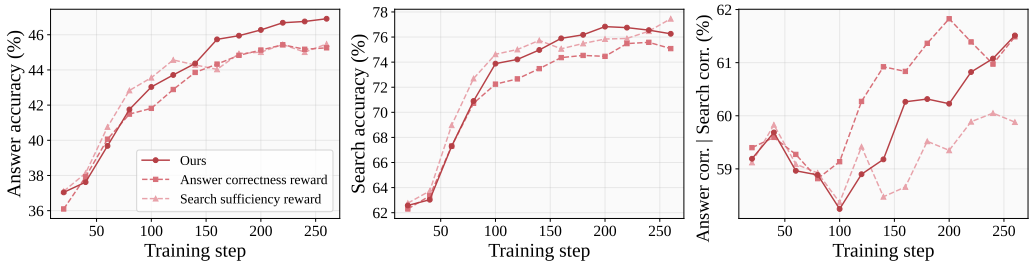


Figure 3: **Training dynamics under different searcher rewards.** Validation performance on NQ and HotpotQA. *Left*: answer EM. *Center* substring EM. *Right*: answer EM on correct search.

Baselines We compare against three groups of baselines: **(1) training-free methods**, including direct answering and chain-of-thought (CoT) prompting (Kojima et al., 2022), naive RAG (Lewis et al., 2020), IRCoT (Trivedi et al., 2023), and Search-o1 (Li et al., 2025b); **(2) single-agent training methods**, including Search-R1 (Jin et al., 2025b) and TIPS (Xie et al., 2026); **(3) multi-agent training methods**, including S3 (Jiang et al., 2025), DART (Li et al., 2026), and M-ASK (Chen et al., 2026). Detailed descriptions and reproduction procedures for each are provided in Appendix C.

5 Main Results

Table 1 presents the main results across seven QA benchmarks, with complementary LLM-judge evaluations in Appendix D.1. DAC consistently achieves the best overall performance on both model backbones. In particular, it outperforms naive single-agent optimization (Search-R1), as well as single-agent training augmented with turn-level rewards (TIPS), suggesting that denser feedback alone is insufficient. DAC also surpasses prior multi-agent approaches (S3, DART, and M-ASK), which rely primarily on final answer-correctness signals. Together, these results support DAC's use of role-specific, cross-verification rewards to better assign credit in decomposed agentic search.

To understand the source of these gains, we compare DAC with Search-R1 and DART, the strongest overall baselines, by disentangling search quality from answer generation. DAC's searcher outperforms both methods (Figure 2 left), and when given the same retrieved evidence as each baseline, DAC's generator still achieves higher answer accuracy (Figure 2 right). This shows that DAC improves both *search quality* and the *generator's ability to use the retrieved evidence*.

6 Analysis

6.1 Ablation Study I: Effect of Cross-Verification Rewards

We first examine the role of cross-verification by comparing DAC with two simpler searcher rewards discussed in section 3.1: *using only external signals*, answer correctness or only search sufficiency, without the generator's verification decision. As shown in Table 2, both variants underperform DAC. Figure 3 further reveals different training dynamics. With answer correctness as the searcher reward, conditional answer accuracy remains high, but search accuracy improves more slowly and overall answer accuracy consistently trails DAC. In contrast, the search sufficiency reward rapidly improves search, while conditional answer accuracy improves little, indicating that search advances faster than the generator's ability to exploit the retrieved evidence. DAC improves both components more coherently, suggesting that cross-verification helps align their training trajectories.

We further analyze this interaction through cross-checkpoint searcher–generator evaluations in Appendix D.4. The results suggest that DAC induces a gradual curriculum: as the generator improves, the searcher expands successful retrieval toward previously unresolved, harder questions, while preserving or improving evidence quality on questions that were already retrievable.

Takeaway 1. Cross-verification induces an adaptive curriculum in which search progressively expands with the generator's evolving capability, leading to better final performance.

Table 2: **Test results for training variants on `Qwen3-8B`.** Scores are reported as differences relative to DAC. **Highlighted text** marks a difference greater than +1, and **colored text** marks less than -1.

Method	General QA			Multi-Hop QA				Average
	NQ*	TriviaQA	PopQA	HotpotQA*	2Wiki	MuSiQue	Bamb.	
DAC (Base)	48.4	67.2	47.9	46.6	48.7	22.3	49.6	47.2
Answer correctness as searcher reward	+0.2	**–1.2**	**–1.1**	**–1.6**	**–3.4**	**–1.7**	**–2.4**	**–1.6**
Search sufficiency as searcher reward	–0.3	**–1.7**	–1.0	–1.0	**–1.1**	**–1.9**	**–8.8**	**–2.2**
w/o augmented samples	**–3.4**	**–1.5**	**–2.8**	**–2.4**	**–5.3**	**–1.9**	–0.8	**–2.6**
Non-abstention additional training	**+1.8**	–0.5	+0.9	+0.2	**+1.1**	+0.1	+0.8	+0.7

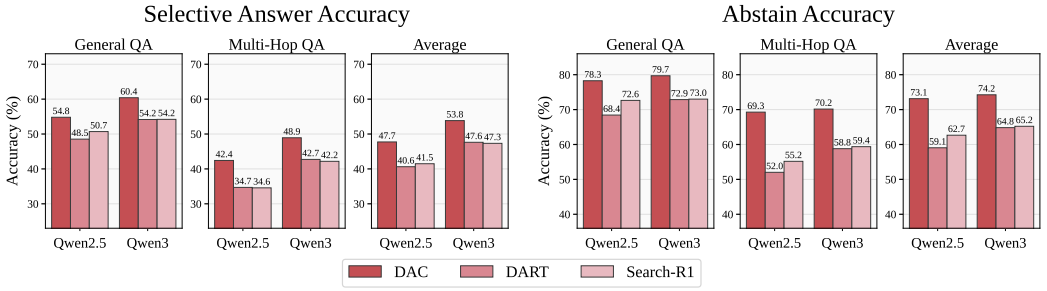


Figure 5: **Selective and abstain accuracy.** *Left*: selective accuracy measures answer EM among non-abstained examples. *Right*: abstain accuracy measures whether the model correctly decides to answer or abstain based on search accuracy, which is judged by `gpt-oss-120b`.

6.2 Ablation Study II: Effect of Hard-Positive Evidence Augmentation

We ablate hard-positive evidence augmentation by removing the noise documents added to sufficient evidence examples. As shown in Table 2, this greatly lowers overall test performance. The training dynamics in Figure 4 clarify the source of this difference. Without augmented samples, the generator has substantially lower recall for non-abstention, i.e., it answers less often in cases where it should answer. This is precisely the over-abstention failure mode discussed in section 3.2: the generator learns an overly strict verification boundary and rejects sufficient but imperfect evidence. Hard-positive augmentation mitigates this by exposing the generator to sufficient but noisy evidence, encouraging it to accept and answer correctly under noisy evidence rather than abstain too conservatively. We also analyze this from a theoretical perspective in section 6.5.

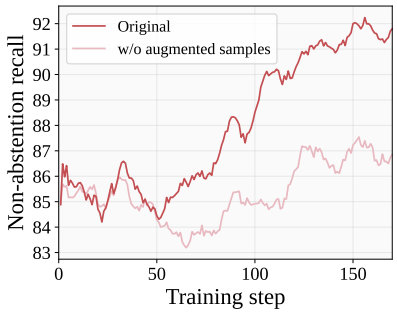


Figure 4: **Non-abstention recall of the generator.** We measure during training how often the generator correctly chooses to answer rather than abstain.

Takeaway 2. Hard-positive evidence augmentation prevents over-abstention during training, improving final performance even when abstention is disabled at test time.

6.3 Abstention at Inference Time

Abstention behavior learned by DAC is meaningful. In Appendix D.6, we find that when the generator is forced to answer on inputs it has abstained on, accuracy drops substantially, indicating that the generator recognizes when evidence is insufficient rather than hastily providing an inaccurate answer.

Ideally, DAC optimizes the searcher toward always retrieving sufficient evidence, in which case the generator would no longer need to abstain. In practice, however, search failures remain, making abstention a useful inference-time capability. As Figure 5 shows, DAC learns substantially better abstention behavior than Search-R1 and DART, while also maintaining higher selective answer

accuracy. For the baselines, we treat continuing to search until the maximum turn limit as abstention. Hence, beyond its role in training, abstention could also be exposed as a standalone functionality in deployed systems. For instance, when the generator abstains, the system could automatically trigger additional search rounds to collect more evidence before producing a final answer.

Yet, for direct comparison with the baselines under standard QA evaluation, abstention is disabled in our main evaluation. Despite being trained with an abstention option, the model readily follows the test-time instruction to always answer, and this simple strategy already yields strong performance. Nevertheless, if the resulting train–test mismatch is a concern, it can be further mitigated with a short post-training stage. Specifically, after the original joint training finishes, or once the searcher reward has largely plateaued, we continue training the shared policy with only the generator objective, while disabling abstention reward and only rewarding answer correctness. This preserves the learned search behavior while adapting the generator to the inference-time setting. On `Qwen3-8B`, roughly 100 additional training steps yield a further improvement in answer accuracy (see Figure 6 and the "Non-abstention additional training" row of Table 2), suggesting that only lightweight adaptation is needed.

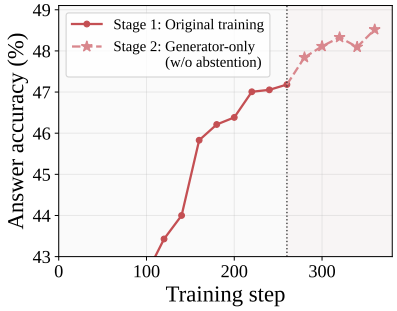


Figure 6: **Generator-only post-training without abstention.** Validation answer EM on NQ and HotpotQA, during joint training and the subsequent generator-only stage with abstention disabled.

6.4 Using dedicated policies for each agent

For computational efficiency, our main experiments optimize a single shared model for both the searcher and generator. To isolate unintended effects of parameter sharing, on `Qwen3-8B`, we additionally compare a shared LoRA policy against dedicated LoRA policies for the two roles. Full results are reported in Appendix D.5 and training dynamics are shown in Figure 7. Dedicated policies achieve better final performance overall.

Interestingly, the shared policy performs slightly better early in training, suggesting that parameter sharing may initially facilitate transfer between the two roles. As training progresses, however, the dedicated policies overtake the shared model and maintain a growing advantage, as the searcher and generator become more specialized and interference between their distinct objectives is reduced. These results suggest that the benefits of role specialization may extend to independently full-parameter fine-tuned policies, which we leave for future work.

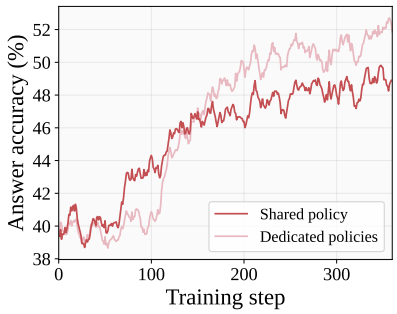


Figure 7: **Shared vs. dedicated LoRA policies.** Training answer EM with a shared policy or dedicated policies for the searcher and generator. Curves show a moving average over 10 training steps. Dedicated policies overtake the shared policy as training progresses.

6.5 Theoretical analysis of DAC

DAC's cross-verification relies on the generator accepting sufficient evidence so that retrieval useful to the generator receives a positive searcher reward. When sufficient training examples are concentrated on easy cases, the generator may instead favor a conservative acceptance rule that rejects harder but answerable evidence. Such over-abstention suppresses the searcher's reward on these inputs. Hard-positive augmentation supplies the corrective signal needed to counter this failure mode.

We study a threshold model in which the generator answers on an input evidence if and only if $h \leq \theta$, where $h \in [0, 1]$ indicates the evidence's scalar **difficulty score** and $\theta \in [0, 1]$ is the generator's chosen **acceptance threshold**. Let γ denote the training relative mixture weight of augmented sufficient evidence in the generator's training input distribution. Holding the evidence distributions and conditional answering ability fixed, we obtain the following result. Appendix A gives the full formulation and proof.

Proposition 1 (Conservative regime and its removal)**.** *Under the model assumptions in Appendix A.1, suppose ordinary sufficient evidence has difficulty at most $b < 1$, and any threshold above b causes the generator to accept additional insufficient evidence it would have abstained on at threshold b. If augmented evidence has positive mass above b on which the generator can answer correctly with positive probability, then there exists a finite $\Gamma > 0$ such that every optimal threshold $\theta^\star_\gamma$ satisfies*

$$\theta^\star_\gamma \le b \quad \text{if } \gamma = 0, \qquad \theta^\star_\gamma > b \quad \text{if } \gamma > \Gamma. \tag{3}$$

The first conclusion identifies an incentive failure: without hard-positive signal, even an optimal generator can remain confined to easy evidence, *over*-abstaining from answering slightly harder but still answerable inputs. The second shows that sufficient augmentation weight removes all optima in this conservative regime. Augmentation thus counteracts a failure of DAC's cross-verification by making acceptance of harder sufficient evidence rewarding. This mechanism is consistent with the reduced non-abstention recall and lower answer accuracy observed when augmentation is removed. The appendix relaxes the restriction on ordinary sufficient evidence, gives an explicit bound on Γ, and further relates the analysis to an idealized Stackelberg game (Stackelberg, 1934).

7 Related work

Learning to search with RL Retrieval-augmented reasoning is increasingly formulated as an agentic task, building on ReAct-style interleaving of reasoning and tool use (Yao et al., 2023). Early RL approaches optimize such trajectories with outcome-level rewards (Jin et al., 2025b; Song et al., 2025; Chen et al., 2025), while later work develops finer turn-level credit assignment from these signals (Wang et al., 2026; Xie et al., 2026; Zhang et al., 2026; Tao et al., 2026). This paradigm has been extended to complex, long-horizon deep-research settings, where models perform many rounds of search, browsing, and synthesis (Nguyen et al., 2025; Li et al., 2025a). A recent direction is to decompose the process across specialized agents (Jiang et al., 2025; Li et al., 2026; Chen et al., 2026). These methods vary in role decomposition and how each agent is trained, but largely rely on final outcome-level supervision. DAC follows this multi-agent direction while introducing role-specific credit assignment between search and generation.

Multi-agent LLM training Recent work has explored multi-agent LLM training through cooperative and adversarial interactions. Cooperative approaches train role-specialized agents to jointly solve tasks through reasoning, refinement, or discussion (Motwani et al., 2025; Park et al., 2025). Related work on verifier-guided RL jointly trains verifiers or critics with the policy, allowing the supervision signal to adapt as the policy evolves (ByteDance Seed, 2025; Kirchner et al., 2024; Wu et al., 2026); some further introduce adversarial interactions to expose failure modes and improve robustness. DAC is related to these ideas in that the generator acts as a critic of search, but it also differs in that both agents are task-performing components: the generator produces the final response while simultaneously providing a co-evolving verification signal that shapes the searcher's behavior.

8 Limitations and Future Work

While DAC performs consistently across our evaluated settings, several limitations remain and motivate future work. First, our hard-positive evidence augmentation is a simplified approximation that adds irrelevant or weakly related documents to sufficient contexts. Real settings may involve conflicting, misleading, partially sufficient, or only jointly sufficient information. Future work could construct more diverse hard-positives through structured perturbations or learned and adversarial evidence generation, potentially adapting the difficulty of evidence to the evolving generator.

Also, for efficiency, we instantiate search sufficiency verification using substring EM. This lightweight proxy can miss paraphrased or jointly sufficient evidence and may falsely accept incidental answer-string matches. DAC is agnostic to the implementation of the verifier, but it is not independent of verifier quality. Future work should evaluate semantic and task-specific verifiers, as well as robustness to verifier false positives and false negatives.

Finally, due to computational constraints, our dedicated-policy experiments are limited to LoRA adaptation. Whether the benefits of role specialization persist under independent full-parameter fine-tuning remains open. Future work could investigate full-parameter training of dedicated policies and characterize its trade-offs in specialization, training stability, and computational cost.

AI USE STATEMENT

In this work, we used generative AI tools to assist in implementing methods, specifically in writing and debugging parts of the training and evaluation code. We have not used generative AI tools to propose or refine hypotheses, design experiments, generate synthetic datasets, clean or reformat datasets, or interpret results. Translation and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools to search for and identify related work and to edit the manuscript for grammar and readability. We have reviewed all AI-assisted work; AI-assisted code was inspected and tested by the authors, all reported results come from our own training and evaluation runs, every cited work was checked by the authors, and all bibliographic entries were verified against the original sources. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All authors have read and adhere to the ICLR Code of Ethics. This work uses only publicly available open-weight models, a public Wikipedia corpus, and established public QA benchmarks, in accordance with their respective licenses. No human subjects were involved, and no personally identifiable information was collected. While our method aims to improve the reliability of retrieval-augmented systems, trained search agents may still inherit biases and factual errors from their retrieval corpus.

REPRODUCIBILITY STATEMENT

Anonymized source code is provided in the supplementary material and will be publicly released upon publication. The problem formulation and reward definitions are given in sections 2 and 3. The retrieval setup, model configurations, full training hyperparameters, RL objectives, checkpoint selection criteria, and the hard-positive augmentation procedure are detailed in Appendix B. Reproduction details for all baselines, including the official codebases we used and any deviations from the original setups, are provided in Appendix C. All prompts, including those for LLM-based evaluation, are listed in Appendix E.

REFERENCES

Cem Anil, Guodong Zhang, Yuhuai Wu, and Roger Grosse. Learning to Give Checkable Answers with Prover-Verifier Games. *arXiv:2108.12099 [cs.LG]*, 2021.

ByteDance Seed. Seed1.5-Thinking: Advancing Superb Reasoning Models with Reinforcement Learning. *arXiv:2504.13914 [cs.CL]*, 2025.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, sunhaoze, ZhouYijie, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. ReSearch: Learning to Reason with Search for LLMs via Reinforcement Learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Yiqun Chen, Lingyong Yan, Zixuan Yang, Erhan Zhang, Jiashu Zhao, Shuaiqiang Wang, Dawei Yin, and Jiaxin Mao. Beyond Monolithic Architectures: A Multi-Agent Search and Knowledge Optimization Framework for Agentic Search. *arXiv:2601.04703 [cs.AI]*, 2026.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In Donia Scott, Nuria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

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. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21599–21617, Suzhou, China, November 2025. Association for Computational Linguistics.

Bowen Jin, Jinsung Yoon, Priyanka Kargupta, Sercan O. Arik, and Jiawei Han. An Empirical Study on Reinforcement Learning for Reasoning-Search Interleaved LLM Agents. *arXiv:2505.15117 [cs.CL]*, 2025a.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. In *Second Conference on Language Modeling*, 2025b.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Yubin Kim, Ken Gu, Chanwoo Park, Chunjong Park, Samuel Schmidgall, A. Ali Heydari, Yao Yan, Zhihan Zhang, Yuchen Zhuang, Yun Liu, Mark Malhotra, Paul Pu Liang, Hae Won Park, Yuzhe Yang, Xuhai Xu, Yilun Du, Shwetak Patel, Tim Althoff, Daniel McDuff, and Xin Liu. Capable language models can outgrow the benefits of collaboration. *Nature Machine Intelligence*, 8(7): 1157–1172, 2026.

Jan Hendrik Kirchner, Yining Chen, Harri Edwards, Jan Leike, Nat McAleese, and Yuri Burda. Prover-Verifier Games improve legibility of LLM outputs. *arXiv:2407.13692 [cs.CL]*, 2024.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large Language Models are Zero-Shot Reasoners. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc., 2022.

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*, 7:452–466, 2019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, page 611–626, New York, NY, USA, 2023. Association for Computing Machinery.

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. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc., 2020.

Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, Kuan Li, Liangcai Su, Litu Ou, Liwen Zhang, Pengjun Xie, Rui Ye, Wenbiao Yin, Xinmiao Yu, Xinyu Wang, Xixi Wu, Xuanzhong Chen, Yida Zhao, Zhen Zhang, Zhengwei Tao, Zhongwang Zhang, Zile Qiao, Chenxi Wang, Donglei Yu, Gang Fu, Haiyang Shen, Jiayin Yang, Jun Lin, Junkai Zhang, Kui Zeng, Li Yang, Hailong Yin, Maojia Song, Ming Yan, Minpeng Liao, Peng Xia, Qian Xiao, Rui Min, Ruixue Ding, Runnan Fang, Shaowei Chen, Shen Huang, Shihang Wang, Shihao Cai, Weizhou Shen, Xiaobin Wang, Xin Guan, Xinyu Geng, Yingcheng Shi, Yuning Wu, Zhuo Chen, Zijian Li, and Yong Jiang. Tongyi DeepResearch Technical Report. *arXiv:2510.24701 [cs.CL]*, 2025a.

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. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5420–5438, Suzhou, China, November 2025b. Association for Computational Linguistics.

Yu Li, Mingyang Yi, Xiuyu Li, Ju Fan, Fuxin Jiang, Binbin Chen, Peng Li, Jie Song, and Tieying Zhang. Reasoning and Tool-use Compete in Agentic RL:From Quantifying Interference to Disentangled Tuning. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026. To appear.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada, July 2023. Association for Computational Linguistics.

Sumeet Ramesh Motwani, Chandler Smith, Rocktim Jyoti Das, Rafael Rafailov, Philip Torr, Ivan Laptev, Fabio Pizzati, Ronald Clark, and Christian Schroeder de Witt. MALT: Improving Reasoning with Multi-Agent LLM Training. In *Second Conference on Language Modeling*, 2025.

Xuan-Phi Nguyen, Shrey Pandit, Revanth Gangi Reddy, Austin Xu, Silvio Savarese, Caiming Xiong, and Shafiq Joty. SFR-DeepResearch: Towards Effective Reinforcement Learning for Autonomously Reasoning Single Agents. *arXiv:2509.06283 [cs.AI]*, 2025.

Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman E. Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. MAPoRL: Multi-Agent Post-Co-Training for Collaborative Large Language Models with Reinforcement Learning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30215–30248, Vienna, Austria, July 2025. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and Narrowing the Compositionality Gap in Language Models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore, December 2023. Association for Computational Linguistics.

Qwen Team. Qwen2.5 Technical Report. *arXiv:2412.15115 [cs.CL]*, 2025a.

Qwen Team. Qwen3 Technical Report. *arXiv:2505.09388 [cs.CL]*, 2025b.

John Schulman and Thinking Machines Lab. LoRA Without Regret. *Thinking Machines Lab: Connectionism*, 2025. URL `https://thinkingmachines.ai/blog/lora/`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs.LG]*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv:2402.03300 [cs.CL]*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A Flexible and Efficient RLHF Framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, page 1279–1297, New York, NY, USA, 2025. Association for Computing Machinery.

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:2503.05592 [cs.AI]*, 2025.

Heinrich Freiherr von Stackelberg. *Marktform und Gleichgewicht*. Julius Springer, Vienna, 1934.

Leitian Tao, Baolin Peng, Wenlin Yao, Tao Ge, Hao Cheng, Mike Hang Wang, Jianfeng Gao, and Sharon Li. TRACE: Turn-level Reward Assignment via Credit Estimation for Long-Horizon Agents. *arXiv:2607.13988 [cs.LG]*, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 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 Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada, July 2023. Association for Computational Linguistics.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information Gain-based Policy Optimization: A Simple and Effective Approach for Multi-Turn Search Agents. In *The Fourteenth International Conference on Learning Representations*, 2026.

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:2212.03533 [cs.CL]*, 2024.

Mian Wu, Gavin Zhang, Sewon Min, Sergey Levine, and Aviral Kumar. RLAC: Reinforcement Learning with Adversarial Critic for Free-Form Generation Tasks. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yutao Xie, Nathaniel Thomas, Nicklas Hansen, Yang Fu, Li Erran Li, and Xiaolong Wang. TIPS: Turn-level Information-Potential Reward Shaping for Search-Augmented LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.

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 Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Erhan Zhang, Yiqun Chen, Zechun Niu, Wei Yang, Xiaochi Wei, Yan Gao, Yi Wu, Yao Hu, and Jiaxin Mao. OASES: Outcome-Aligned Search-Evaluation Co-Training for Agentic Search. *arXiv:2604.03675 [cs.AI]*, 2026.

Qingfei Zhao, Ruobing Wang, Yukuo Cen, Daren Zha, Shicheng Tan, Yuxiao Dong, and Jie Tang. LongRAG: A Dual-Perspective Retrieval-Augmented Generation Paradigm for Long-Context Question Answering. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22600–22632, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel. *Proc. VLDB Endow.*, 16(12):3848–3860, August 2023.

Runchuan Zhu, Zinco Jiang, Jiang Wu, Zhipeng Ma, Jiahe Song, Fengshuo Bai, Dahua Lin, Lijun Wu, and Conghui He. GRAIT: Gradient-Driven Refusal-Aware Instruction Tuning for Effective Hallucination Mitigation. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4006–4021, Albuquerque, New Mexico, April 2025a. Association for Computational Linguistics.

Runchuan Zhu, Zhipeng Ma, Jiang Wu, Junyuan Gao, Jiaqi Wang, Dahua Lin, and Conghui He. Utilize the Flow Before Stepping into the Same River Twice: Certainty Represented Knowledge Flow for Refusal-Aware Instruction Tuning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(24):26157–26165, 2025b.

A THEORETICAL ANALYSIS OF DAC

A.1 SETUP AND GENERATOR OBJECTIVE

We analyze the generator reward

$$r_{\mathcal{G}} = sa + (1-s)v,$$

where s denotes search sufficiency, v abstention, and a answer correctness. Thus, on sufficient evidence ($s = 1$), the generator is rewarded only for answering correctly, whereas on insufficient evidence ($s = 0$), it is rewarded only for abstaining. We assume accurate sufficiency and correctness labels.

Each input $(Q, \mathcal{E})$ has a scalar **difficulty score** $h \in [0, 1]$, with larger scores representing inputs the generator is more inclined to abstain. The generator chooses an **acceptance threshold** $\theta \in [0, 1]$, and uses

$$v \leftarrow v_\theta(h) \coloneqq \mathbb{I}\big[h > \theta\big],$$

i.e., it answers on an evidence set of difficulty h if and only if $h \le \theta$.

Sufficient and insufficient inputs may have overlapping score distributions; their true sufficiency labels are not supplied to the acceptance rule. Conditional on the abstention decision, we assume the answer generation policy (when chosen to answer) is held fixed as θ changes. This is an idealized restriction used to isolate the acceptance decision.

In a batch training setting, the generator is updated on a mixture of evidence states. We abstract this mixture using three effective generator-side weights: α for **insufficient**-evidence examples, β for **ordinary sufficient** examples, and γ for **augmented sufficient** examples. Insufficient examples may arise from failed search trajectories or constructed insufficient inputs. We aggregate these cases into the generator-side weight α, rather than modeling search failure explicitly, since our analysis focuses on the impact of the acceptance threshold.

Let I, H, and S denote the three sources of inputs in order. For each sufficient source $j \in \{H, S\}$, let the measurable function $c_j(h) \in [0, 1]$ give the conditional probability of a correct answer when chosen to answer. No monotonicity assumption on the correctness functions is required. Also, let P_j be the probability distribution over difficulty scores for each evidence source. We assume that P_I has no atoms, while P_H and P_S may contain point masses. The former assumption holds for continuous score distributions and ensures continuity of the insufficient-input acceptance mass. The latter allowance includes the endpoint construction in Appendix A.5. We hold the evidence distributions, correctness functions, and the original source weights α and β fixed while varying only the augmentation weight γ.

Define the cumulative reward quantities

$$F_I(\theta) = P_I\big([0, \theta]\big),$$
$$C_j(\theta) = \int_{[0,\theta]} c_j(h)\, dP_j(h), \qquad j \in \{H, S\}.$$

All three are nondecreasing. F_I is the probability mass of insufficient inputs incorrectly accepted at threshold θ, while C_H and C_S are correctness rewards gained by answering sufficient inputs from the two sources.

For an insufficient input at score h, answering receives reward 0 and abstaining receives reward 1. Its expected reward is therefore $v_\theta(h)$. For a sufficient input from source $j \in \{H, S\}$, the expected reward is $\big(1 - v_\theta(h)\big)c_j(h)$. Integrating over the three sources gives the expected generator reward

$$\mathbb{E}\big[r_{\mathcal{G}} \mid \theta\big] = \frac{\alpha\big(1 - F_I(\theta)\big) + \beta C_H(\theta) + \gamma C_S(\theta)}{\alpha + \beta + \gamma}.$$

The denominator is positive and independent of θ. Hence the following unnormalized objective has the same maximizers for each fixed augmentation weight γ:

$$U_\gamma(\theta) = \alpha\big(1 - F_I(\theta)\big) + \beta C_H(\theta) + \gamma C_S(\theta). \tag{4}$$

A.2 Formal statement and proof of Proposition 1

Theorem 1 (Formal restatement of Proposition 1)**.** *Under the assumptions in Appendix A.1, hold P_I, P_H, P_S, c_H, c_S, $\alpha > 0$, and $\beta \geq 0$ fixed. For each $\gamma \geq 0$, let*

$$\Theta(\gamma) = \operatorname*{argmax}_{\theta \in [0,1]} U_\gamma(\theta).$$

Suppose there exists $b \in [0, 1)$ such that

$$P_H\big((b, 1]\big) = 0, \qquad F_I(\theta) > F_I(b) \quad \text{for every} \quad \theta > b, \qquad C_S(1) > C_S(b). \tag{5}$$

Write $U_\gamma(\theta) = U_0(\theta) + \gamma C_S(\theta)$ and define

$$M_b = \max_{0 \leq u \leq b} U_0(u), \qquad \Gamma_b = \frac{M_b - U_0(1)}{C_S(1) - C_S(b)}.$$

Then, the following hold.

(i) *$\Theta(\gamma)$ is nonempty and compact for every $\gamma \geq 0$.*

(ii) *$0 < \Gamma_b < \infty$.*

(iii) *$\Theta(0) \subseteq [0, b]$, while $\Theta(\gamma) \subseteq (b, 1]$ for every $\gamma > \Gamma_b$.*

(iv) *For every $\gamma > \Gamma_b$ and every $\theta^\star \in \Theta(\gamma)$,*

$$C_S(\theta^\star) - C_S(b) \geq \frac{U_0(b) - U_0(\theta^\star)}{\gamma} > 0. \tag{6}$$

In particular, $P_S\big((b, \theta^\star]\big) > 0$, so non-abstention recall on the fixed augmented source strictly exceeds that of every unaugmented optimum.

The first condition in eq. (5) restricts ordinary sufficient inputs to difficulty at most b. The second means that any broader threshold admits additional insufficient inputs and therefore loses reward. The third requires that there exist augmented sufficient evidence above b on which the generator can answer correctly with positive probability. These conditions describe a shortage of hard-positive signal in the original mixture that can be addressed through augmentation. Proposition 1 in the main text follows by taking $\Gamma = \Gamma_b$.

Proof. First, optimal thresholds exist. Because P_I is atomless, its cumulative function F_I is continuous. Also, C_H and C_S are both cumulative functions of nonnegative finite measures and are upper semicontinuous. Thus U_γ is upper semicontinuous, and its maximizing set on $[0, 1]$ is nonempty and compact. Likewise, U_0 attains its maximum on $[0, b]$, so M_b is well defined.

Next, the numerator of Γ_b is positive because

$$M_b \geq U_0(b) > U_0(1),$$

where the strict inequality follows from $P_H\big((b, 1]\big) = 0$ and $F_I(1) > F_I(b)$. Its denominator is positive by assumption. All terms are finite, proving $0 < \Gamma_b < \infty$.

The first condition of eq. (5) gives $C_H(\theta) = C_H(b)$ for every $\theta > b$. Consequently, without augmentation, or if $\gamma = 0$,

$$U_\gamma(\theta) - U_\gamma(b) = U_0(\theta) - U_0(b) = -\alpha\big(F_I(\theta) - F_I(b)\big) < 0, \qquad \theta > b. \tag{7}$$

Hence no threshold larger than b can maximize U_γ, proving $\Theta(0) \subseteq [0, b]$. Note that since $C_S(1) > C_S(b)$, these thresholds in $\Theta(0)$ reject inputs with positive correctness-weighted mass under P_S. The rejected mass is at least $C_S(1) - C_S(b) > 0$.

Also, for any $u \leq b$ and $\gamma > \Gamma_b$,

$$\begin{aligned} U_\gamma(1) - U_\gamma(u) &= U_0(1) - U_0(u) + \gamma\big(C_S(1) - C_S(u)\big) \\ &\geq U_0(1) - M_b + \gamma\big(C_S(1) - C_S(b)\big) > 0. \end{aligned}$$

Hence $\Theta(\gamma) \subseteq (b, 1]$.

Finally, choose any $\theta^\star \in \Theta(\gamma)$, then optimality gives $U_\gamma(\theta^\star) \geq U_\gamma(b)$. If $\gamma > \Gamma_b$, then $\theta^\star > b$ and $U_0(\theta^\star) < U_0(b)$ by eq. (7), so

$$\gamma\big(C_S(\theta^\star) - C_S(b)\big) \geq U_0(b) - U_0(\theta^\star) > 0.$$

Dividing by $\gamma > 0$ proves eq. (6). Thus $C_S(\theta^\star) > C_S(b)$, i.e., the optimum accepts additional sufficient inputs with positive correctness reward. In particular, $P_S\big((b, \theta^\star]\big) > 0$. Every unaugmented optimum is at most b, so its acceptance probability under P_S is at most $P_S\big([0, b]\big)$, which is strictly smaller than $P_S\big([0, \theta^\star]\big)$. This explains the increase in non-abstention recall when using augmented samples. □

Beyond the support restriction Previously, we assumed that there is no ordinary sufficient evidence with difficulty above b. This is a sufficient condition for the conservative regime to exist, but it is not necessary. The same conclusions and proof hold under the more general condition

$$\beta\big(C_H(\theta) - C_H(b)\big) < \alpha\big(F_I(\theta) - F_I(b)\big), \qquad \text{for every } \ \theta > b.$$

This allows ordinary hard-positives, provided their correctness reward is too weak to offset the reward lost from accepting additional insufficient inputs. In either case, the optimal acceptance threshold is at most b without augmentation, and strictly larger than b with sufficient augmentation weight. The bound Γ_b is sufficient but not necessary for the regime change; a smaller weight may suffice if the ordinary sufficient evidence above b is weak enough.

Implication for cross-verification On sufficient evidence, the searcher reward equals the generator's acceptance indicator, $1 - v$. A threshold at or below b therefore gives the searcher zero reward on sufficient inputs above b, even when their conditional answering accuracy is positive. The augmented regime restores that reward on additional answerable sufficient inputs. Yet this is an incentive-level consequence of the reward definition, and it does not by itself prove that searcher training reaches those inputs or that the coupled training process converges.

A.3 Monotonicity of the optimal acceptance threshold

We further analyze the optimal acceptance threshold as a function of the augmentation weight γ. Define

$$\underline{\theta}(\gamma) = \min \Theta(\gamma), \qquad \overline{\theta}(\gamma) = \max \Theta(\gamma).$$

The following result shows that both extremal selections of optimal thresholds are nondecreasing in γ. This extends Theorem 1 and does not require the support restriction on ordinary sufficient evidence.

Proposition 2 (Monotone acceptance boundary)**.** *Under the assumptions in Appendix A.1, hold the evidence distributions, conditional correctness functions, α, and β fixed. Then, both $\underline{\theta}(\gamma)$ and $\overline{\theta}(\gamma)$ are nondecreasing in γ. Under either consistent choice, non-abstention recall is nondecreasing on any fixed distribution of difficulty scores for sufficient inputs.*

Proof. Existence and compactness of the maximizing set were established in Theorem 1, so both extrema are well defined.

Fix $\gamma_2 > \gamma_1$ and choose $\theta_i \in \Theta(\gamma_i)$. Their optimality implies

$$U_0(\theta_1) + \gamma_1 C_S(\theta_1) \geq U_0(\theta_2) + \gamma_1 C_S(\theta_2),$$
$$U_0(\theta_2) + \gamma_2 C_S(\theta_2) \geq U_0(\theta_1) + \gamma_2 C_S(\theta_1).$$

Adding these inequalities yields

$$(\gamma_2 - \gamma_1)\big(C_S(\theta_2) - C_S(\theta_1)\big) \geq 0. \tag{8}$$

Suppose $\theta_2 < \theta_1$. Because C_S is nondecreasing, eq. (8) forces $C_S(\theta_2) = C_S(\theta_1)$. The optimality inequalities then give $U_0(\theta_2) = U_0(\theta_1)$. Both thresholds therefore maximize both objectives. If $\theta_i = \underline{\theta}(\gamma_i)$, this contradicts the minimality of θ_1, because the smaller θ_2 also maximizes U_{γ_1}. If $\theta_i = \overline{\theta}(\gamma_i)$, it contradicts the maximality of θ_2, because the larger θ_1 also maximizes U_{γ_2}. Both extremal selections are consequently nondecreasing.

For any fixed sufficient-input difficulty score distribution P^+, non-abstention recall equals $P^+\big([0, \theta]\big)$. This is nondecreasing in θ, proving the recall statement. □

Therefore, we can conclude that with larger augmentation weight, the generator never reduces its acceptance threshold. Furthermore, if C_S is strictly increasing, eq. (8) also rules out $\theta_2 < \theta_1$ for arbitrary choices of maximizers, implying that the entire set of maximizers is nondecreasing in γ.

A.4 Conditions for strict expansion

We now quantify conditions under which increasing the augmentation weight *strictly* increases the optimal acceptance threshold. Suppose the three score distributions have densities f_I, f_H, f_S and the products $c_H f_H$ and $c_S f_S$ are continuous. Then the derivative of the objective is

$$U'_\gamma(\theta) = -\alpha f_I(\theta) + \beta c_H(\theta) f_H(\theta) + \gamma c_S(\theta) f_S(\theta). \tag{9}$$

Any interior optimum $\theta^\star$ therefore satisfies

$$\beta c_H(\theta^\star) f_H(\theta^\star) + \gamma c_S(\theta^\star) f_S(\theta^\star) = \alpha f_I(\theta^\star).$$

The left-hand side is the correctness reward gained by admitting marginal sufficient inputs, and the right-hand side is the reward lost by admitting marginal insufficient inputs. This is a necessary condition. If U'_γ is strictly decreasing and changes sign from positive to negative, its unique root is the unique global optimum.

Proposition 3 (Strict local expansion)**.** *Suppose the terms in eq. (9) are continuously differentiable. Fix $\gamma_0 > 0$ and assume the global optimum*

$$\theta^\star(\gamma_0) = \operatorname*{argmax}_{\theta \in [0,1]} U_{\gamma_0}(\theta)$$

is unique and interior throughout a neighborhood of γ_0, with $U''_{\gamma_0}\big(\theta^\star(\gamma_0)\big) < 0$. Then

$$\left.\frac{d\theta^\star}{d\gamma}\right|_{\gamma=\gamma_0} = \frac{c_S\big(\theta^\star(\gamma_0)\big) f_S\big(\theta^\star(\gamma_0)\big)}{-U''_{\gamma_0}\big(\theta^\star(\gamma_0)\big)} \geq 0. \tag{10}$$

The derivative is strictly positive if its numerator is positive.

Proof. An interior maximizer $\theta^\star$ satisfies $U'_\gamma(\theta^\star) = 0$. Continuity and compactness imply that unique maximizers converge to $\theta^\star(\gamma_0)$ as $\gamma \to \gamma_0$, and the root is locally unique. Also, since $U''_{\gamma_0}\big(\theta^\star(\gamma_0)\big) < 0$, we can apply the implicit function theorem to the stationarity equation $U'_\gamma(\theta^\star) = 0$ in a neighborhood of γ_0. Differentiating the stationarity equation with respect to γ gives

$$U''_\gamma(\theta^\star)\frac{d\theta^\star}{d\gamma} + c_S(\theta^\star) f_S(\theta^\star) = 0,$$

which yields eq. (10). The denominator is strictly positive and the numerator is nonnegative, proving the nonnegativity of the derivative. If $c_S(\theta^\star) f_S(\theta^\star) > 0$, then the derivative is strictly positive. □

Therefore, we can state that the optimal threshold strictly increases until its movement offsets the extra marginal reward obtained from the augmented source. In other words, the threshold continues to increase until either the marginal correctness reward from the augmented source becomes zero or the optimum reaches the boundary of the feasible interval. The former may occur if augmented sufficient evidence has no mass near the current optimum, while the latter occurs when the optimum reaches 1.

A.5 An idealized Stackelberg special case

We can simplify the analysis by considering a special case in which the evidence distributions are concentrated at a point mass. This allows us to view the system as a Stackelberg game, especially a lightweight analogue of a *prover-verifier game* (Kirchner et al., 2024; Anil et al., 2021). Kirchner et al. (2024) study a game where the main agent, the *prover*, is trained to produce correct answers, while a secondary agent, the *verifier*, is trained to accept or reject the prover's answers. To guide the verifier to learn a robust decision boundary, the prover is trained in two tracks: a *helpful prover* that produces correct answers that should be accepted and a *sneaky prover* that produces incorrect answers that may fool the verifier. This setup encourages the verifier to learn a decision boundary that can distinguish between correct and incorrect answers, even in the presence of adversarial examples.

Our augmentation targets the opposite failure mode: *rejecting correct but noisy answerable cases*. Thus, our sneaky agent should target examples that are correct and answerable but likely to be rejected because the evidence is noisy or imperfect. Instead of training a separate adversarial agent to produce such hard-positives, we approximate this effect with noise-based evidence augmentation. In more complex settings where noise sampling is difficult, one could instead train an adversarial searcher to generate hard sufficient evidence.

Under this interpretation, we have a game of three players: a **helpful searcher** $\mathcal{S}_H$, a **sneaky searcher** $\mathcal{S}_S$, and a **generator** $\mathcal{G}$. The helpful searcher aims to produce sufficient evidence that the generator accepts and answers from, whereas the sneaky searcher aims to produce sufficient but difficult evidence that the generator incorrectly rejects by abstaining. The generator becomes the leading agent and commits to an acceptance threshold $\theta \in [0, 1]$. The two searchers then choose their evidence difficulties $h_H, h_S \in [0, 1]$ in response to the generator's commitment. Under the source weights α, β, γ, we can write the utility functions of the three players as

$$\begin{aligned} U_H(h_H, \theta) &= \mathbb{I}\big[h_H \le \theta\big], \\ U_S(h_S, \theta) &= \mathbb{I}\big[h_S > \theta\big], \\ U_G(h_H, h_S, \theta) &= \alpha(1-\theta) + \beta c_G(h_H)\mathbb{I}\big[h_H \le \theta\big] + \gamma c_G(h_S)\mathbb{I}\big[h_S \le \theta\big], \end{aligned} \tag{11}$$

where $\alpha > 0$, $\beta, \gamma \ge 0$. Here, $c_G : [0, 1] \to [0, 1]$ is the probability that the generator answers correctly when given a sufficient evidence set of difficulty h. We assume $c_G(0) = 1$, or equivalently, that the generator always answers correctly on the easiest sufficient evidence. Insufficient evidence contributes the first term, and it is not chosen by the two searchers in this simplified game. Among best responses, let the helpful searcher choose the smallest difficulty and the adversarial searcher the largest.

A generator-leader Stackelberg equilibrium is a triple $(h_H^\star, h_S^\star, \theta^\star)$ in which the searchers follow these best-response rules and $\theta^\star$ maximizes the resulting generator utility. We can characterize the equilibrium triples in closed form.

Proposition 4 (Endpoint equilibria)**.** *Under the above assumptions, the set of Stackelberg equilibria of the induced game is given by:*

$$(h_H^\star, h_S^\star, \theta^\star) = \begin{cases} (0, 1, 0), & \text{if } \alpha > \gamma c_{\mathcal{G}}(1), \\ (0, 1, 1), & \text{if } \gamma c_{\mathcal{G}}(1) > \alpha, \\ (0, 1, 0) \text{ or } (0, 1, 1), & \text{if } \alpha = \gamma c_{\mathcal{G}}(1). \end{cases} \tag{12}$$

Proof. For any θ, the helpful searcher's best-response set is $[0, \theta]$, so its tie-breaking rule selects $h_H = 0$. For $\theta < 1$, the sneaky searcher's best-response set is $(\theta, 1]$, so it selects $h_S = 1$. At $\theta = 1$, every difficulty gives it utility zero, and the same rule again selects $h_S = 1$. The induced generator objective then becomes

$$U_G(0, 1, \theta) = \beta + \alpha(1-\theta) + \gamma c_G(1)\mathbb{I}\big[\theta = 1\big].$$

It is strictly decreasing on $[0, 1)$, so only 0 and 1 can be optimal. Comparing their values $\beta + \alpha$ and $\beta + \gamma c_G(1)$ proves eq. (12), including the equality case. □

This induced objective is exactly eq. (4) with P_I uniform, $P_H = \delta_0$, $P_S = \delta_1$, $c_H(0) = 1$, and $c_S(1) = c_G(1)$. This special case illustrates how the sneaky searcher plays a central role in determining the generator's equilibrium acceptance threshold. At equilibrium, the helpful searcher always chooses the easiest sufficient evidence, $h_H^\star = 0$, while the sneaky searcher chooses the hardest sufficient evidence, $h_S^\star = 1$. The generator's threshold is determined by a trade-off between the reward for abstaining on insufficient evidence, weighted by α, and the reward for answering on hard but sufficient evidence, weighted by $\gamma c_{\mathcal{G}}(1)$. That is, when the generator is sufficiently accurate on hard-evidence inputs and the effective proportion of hard-sufficient examples is large relative to insufficient examples, the equilibrium shifts toward a larger acceptance threshold. In this regime, we expect the generator to abstain less often on hard but sufficient evidence. Also, without the sneaky searcher, i.e., when $\gamma = 0$, the equilibrium deterministically collapses to the trivial easy-positive regime $(h_H^\star, \theta^\star) = (0, 0)$. In this regime, the generator has no incentive to expand its acceptance threshold beyond the easiest sufficient evidence.

Scope and limitations of our analysis Our general analysis establishes no convergence guarantee for actual training. It also does not imply higher forced-answer accuracy or selective accuracy, and we assume the generator's answering ability is fixed. Moreover, expanding acceptance can admit additional insufficient inputs, but we do not model any adverse effect on the searcher.

B Further implementation details

Retrieval setup The dense retrieval system uses the December 2018 Wikipedia dump as its corpus, with `E5-base-v2` (Wang et al., 2024) as the embedding model. Each query retrieves the top-3 passages, and the searcher is allowed at most four search turns.

Models and parameterization We train DAC with `Qwen2.5-7B-Instruct` and `Qwen3-8B`. While `Qwen3-8B` supports both thinking and non-thinking modes, we use the non-thinking mode to avoid starting from a model with previously shaped reasoning behavior. The main experiments use full-parameter fine-tuning of a single shared policy. For the policy-sharing analysis in section 6.4, we instead use LoRA. The shared-policy setting uses a rank-64 adapter with $\alpha = 128$, while the dedicated-policy setting uses separate rank-32 adapters with $\alpha = 64$ for the searcher and generator.

Training setup RL training is implemented with verl (Sheng et al., 2025). We use FSDP2 (Zhao et al., 2023) with CPU offloading and gradient checkpointing, and generate rollouts with vLLM (Kwon et al., 2023) using tensor parallel size 1. Rollouts use temperature 1.0 and top-p 1.0. We train on 8 H100 GPUs with a global batch size of 512, mini-batch size of 256, and dynamic micro-batch sizing. We use AdamW (Loshchilov and Hutter, 2019) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay 0.01. The main experiments use a learning rate of 1×10^{-6}. For shared-policy LoRA training, we apply a $10\times$ learning-rate scaling following Schulman and Thinking Machines Lab (2025). For dedicated-policy LoRA training, both agents use a learning rate of 1×10^{-6} on `Qwen2.5-7B-Instruct`. On `Qwen3-8B`, we use 2×10^{-6} for the searcher and 1×10^{-5} for the generator based on stabilization tuning. We train for 500 steps and save checkpoints every 20 steps. If training diverges, we report the last stable checkpoint based on the generator reward curve.

RL objectives Because the searcher receives intermediate rewards across multiple search turns, we optimize it with PPO (Schulman et al., 2017). In contrast, generator training can be viewed as a single-turn LLM RL problem: given $(Q, \mathcal{E})$, the generator produces reasoning followed by either an answer or an abstention decision. We therefore optimize the generator with GRPO (Shao et al., 2024). For both objectives, we use token-level policy-gradient losses (Yu et al., 2025):

$$\mathcal{J}_{\text{PPO}}(\theta) = \mathbb{E}_{\substack{x\sim\mathcal{D} \\ y\sim\pi_0(\cdot|x)}} \left[\frac{1}{|y|} \sum_{t=1}^{|y|} \mathcal{C}_\epsilon\left(\frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_0(y_t \mid x, y_{<t})}, A_t^{\text{GAE}} \right) \right],$$

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\substack{x\sim\mathcal{D} \\ \{y_i\}_{i=1}^{G}\sim\pi_0(\cdot|x)}} \left[\frac{1}{\sum_{i=1}^{G}|y_i|} \sum_{i=1}^{G} \sum_{t=1}^{|y_i|} \mathcal{C}_\epsilon\left(\frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_0(y_{i,t} \mid x, y_{i,<t})}, A_i^{\text{GRPO}} \right) \right],$$

where $\mathcal{C}_\epsilon$ is the clipping function $\mathcal{C}_\epsilon(\rho, A) = \min\big(\rho A, \text{clip}\,(\rho, 1-\epsilon, 1+\epsilon)A\big)$. We use no KL penalty and set the clipping parameter to $\epsilon = 0.2$. For PPO, we set $\gamma = 1$ and $\lambda = 0.995$. For generator training with GRPO, we use group size $G = 5$. Here, $\mathcal{D}$ denotes the prompt dataset, π_θ the current policy, π_0 the rollout policy, and A_t^{GAE} and A_i^{GRPO} the corresponding GAE and GRPO advantages. For mini-batch and micro-batch aggregation, losses are normalized by the total number of tokens within each per-GPU micro-batch and then averaged across micro-batches.

Hard-positive evidence augmentation For noise-based augmentation, we retrieve the top-15 passages using the original question, remove passages already present in the evidence set, and select three remaining passages in reverse rank order as distractors. These are appended to the original evidence to form an augmented input used for generator training. Augmented samples only increase the number of samples contributing to the same update, and do not introduce extra optimization steps.

C Baselines

We provide brief descriptions and reproduction details for each baseline below. Unless otherwise noted, all methods use the same backbone models, retrieval corpus, and evaluation setup as DAC.

- **Direct prompting** asks the backbone model to answer the question directly without external retrieval.
- **CoT** (Kojima et al., 2022) additionally elicits chain-of-thought reasoning by appending `Let's think step by step.` to the prompt.
- **Naive RAG** (Lewis et al., 2020) supplies the model with the top-3 passages retrieved by our dense-retrieval system and asks it to generate the final answer.
- **IRCoT** (Trivedi et al., 2023) interleaves chain-of-thought reasoning with retrieval, using intermediate reasoning to guide subsequent searches. We use the official codebase.[1] For the `Qwen2.5-7B-Instruct` setting, we instead use `Qwen2.5-7B-Base`, since the instruction-tuned model performs poorly under the generation format assumed by IRCoT.
- **Search-o1** (Li et al., 2025b) dynamically invokes search during reasoning and uses a separate Reason-in-Documents component to process retrieved information. We reproduce it using the official codebase.[2]
- **Search-R1** (Jin et al., 2025b;a) trains a single model with RL to interleave reasoning and multi-turn search. We consider four variants combining PPO or GRPO with either outcome-only rewards or outcome and format rewards. All variants are reproduced using the official codebase,[3] except for three `Qwen2.5-7B-Instruct` configurations for which official checkpoints are available: PPO with outcome-only reward, GRPO with outcome-only reward, and GRPO with outcome and format rewards.[4] For the format-reward variants, we use a coefficient of 0.2. We train for up to 500 steps, saving checkpoints every 20 steps and using the last stable checkpoint if training collapses.
- **TIPS** (Xie et al., 2026) augments single-agent search training with turn-level reward shaping based on the information contribution of each turn. We use the official codebase.[5] Following the paper, we train for 500 steps and re-synchronize the teacher at step 200. We use dynamic α with a target information-reward range of 0.05–0.3, which is reported as the best-performing configuration.
- **S3** (Jiang et al., 2025) separates search from answer generation, keeps the generator fixed, and trains the searcher based on downstream answer quality. We use the official codebase.[6] S3 introduces its own answer correctness verifier (`GenAcc`), but for consistency with the other methods, we instead use EM as the verifier during training. We train for 100 steps, exceeding the paper's observation that roughly 15 steps are sufficient for convergence. In our reproduction, however, neither training nor test performance improved meaningfully over the course of training. We therefore report the final trained checkpoint for consistency with the other baselines.
- **DART** (Li et al., 2026) disentangles reasoning and tool-use behavior using separate LoRA modules and optimizes them toward the final task outcome. We use the associated codebase provided by one of the authors.[7] The originally reported DART results additionally use an SFT warm-up stage and add special tags such as `<search>` to the tokenizer. To keep the training setup comparable to the other baselines, we omit these steps and train DART directly from the original backbone. Switching between the two roles is instead triggered whenever the generated string `<search>` appears, even though it is not represented as a single token. We train for up to 500 steps, saving checkpoints every 20 steps and using the last stable checkpoint if training collapses.
- **M-ASK** (Chen et al., 2026) decomposes agentic search into specialized components for search behavior and knowledge management, including planning, retrieval, summarization, and context updating. We reproduce M-ASK using the official codebase.[8] We train for up to 500 steps, saving checkpoints every 20 steps and using the last stable checkpoint if training collapses.

[1] https://github.com/StonyBrookNLP/ircot
[2] https://github.com/RUC-NLPIR/Search-o1
[3] https://github.com/PeterGriffinJin/Search-R1
[4] SearchR1-nq_hotpotqa_train-qwen2.5-7b-it-em-ppo-v0.2, SearchR1-nq_hotpotqa_train-qwen2.5-7b-it-em-grpo-v0.2, SearchR1-nq_hotpotqa_train-qwen2.5-7b-it-em-grpo-v0.3
[5] https://github.com/ucsd-wang-lab-lm/tips
[6] https://github.com/pat-jj/s3
[7] https://github.com/sheriyuo/DART
[8] https://github.com/chenyiqun/M-ASK

Table 3: **LLM judgement on QA benchmarks.** We use `gpt-oss-120b` as the judge. Datasets marked with * are in-domain datasets. **Bold** and underlined numbers indicate the best and second best results for each model, respectively.

Type	Method	General QA			Multi-Hop QA				Average
		NQ*	TriviaQA	PopQA	HotpotQA*	2Wiki	MuSiQue	Bamb.	
`Qwen2.5-7B-Instruct`									
Training-free	Direct	30.5	52.3	19.2	28.0	25.8	7.9	11.2	25.0
	CoT	33.7	56.2	19.0	30.5	22.8	10.2	30.4	29.0
	Naive RAG	55.4	71.4	46.8	42.7	21.2	9.9	28.0	39.4
	IRCoT	46.8	59.4	45.6	40.9	31.1	11.1	26.4	37.3
	Search-o1	51.8	65.2	41.1	44.8	32.9	18.5	41.6	42.3
Single-agent	Search-R1	57.2	72.0	47.5	54.3	39.5	21.8	50.4	49.0
	TIPS	57.7	73.8	48.1	53.9	42.1	20.4	45.6	48.8
Multi-agent	S3	**64.5**	74.7	47.7	55.6	36.4	19.4	44.0	48.9
	DART	56.0	72.0	45.7	56.1	**45.6**	22.1	49.6	49.6
	M-ASK	58.6	74.6	45.5	55.2	40.4	21.6	44.0	48.6
Multi-agent	**DAC**	60.2	**74.8**	**49.2**	**59.9**	45.2	**25.5**	**54.4**	**52.7**
`Qwen3-8B (Non-Thinking)`									
Training-free	Direct	32.5	52.5	20.5	28.1	27.1	7.8	12.8	25.9
	CoT	36.0	53.8	19.9	31.3	27.6	10.9	40.8	31.5
	Naive RAG	54.9	70.3	45.5	42.9	22.7	11.0	34.4	40.2
	IRCoT	57.9	74.5	50.0	53.5	34.4	17.7	47.2	47.9
	Search-o1	56.3	72.1	45.3	51.3	40.9	22.1	48.0	48.0
Single-agent	Search-R1	61.8	77.1	50.4	62.3	50.1	**29.6**	59.2	55.8
	TIPS	59.4	74.5	49.1	55.7	47.7	26.4	56.0	52.7
Multi-agent	S3	62.0	78.0	51.9	53.5	36.4	17.9	41.6	48.8
	DART	60.2	77.1	51.9	62.4	50.7	27.9	**60.0**	55.7
	M-ASK	61.3	77.9	48.1	62.0	49.1	26.2	50.4	53.6
Multi-agent	**DAC**	**63.5**	**78.8**	**52.5**	**64.2**	**54.1**	**29.6**	59.2	**57.4**

Table 4: **Matched comparison with DART.** We compare DAC with DART under matched conditions (SFT warmup, two LoRA adapters). **Bold** numbers indicate the best results for each model.

Method	General QA						Multi-Hop QA								Average	
	NQ*		TriviaQA		PopQA		HotpotQA*		2Wiki		MuSiQue		Bamb.			
	EM	LLM	EM	LLM	EM	LLM	EM	LLM	EM	LLM	EM	LLM	EM	LLM	EM	LLM
`Qwen2.5-7B-Instruct`																
DART	47.1	60.0	64.8	75.7	**48.7**	**53.3**	40.9	60.1	43.7	45.2	18.1	26.1	37.6	48.0	43.0	52.6
DAC	**48.8**	**61.6**	**65.5**	**76.7**	46.2	51.0	**45.7**	**63.1**	**43.9**	**50.0**	**19.5**	**27.0**	**41.6**	**49.6**	**44.5**	**54.1**
`Qwen3-8B (Non-Thinking)`																
DART	44.4	58.8	64.9	76.7	**48.1**	**53.9**	47.0	64.3	41.4	47.9	22.4	30.0	45.6	53.6	45.2	55.0
DAC	**48.7**	**60.3**	**65.4**	**77.0**	47.8	51.5	**47.3**	**65.5**	**44.7**	**51.0**	**23.0**	**30.4**	**45.6**	**54.4**	**46.1**	**55.7**

D Additional results and analysis

D.1 Main results evaluated by LLM judgement

Since answer EM may fail to capture semantically correct responses that do not exactly match the reference answer, we additionally evaluate the main results using LLM-based judgment with `gpt-oss-120b` as the judge model. We use reasoning effort `medium`, and provide the full judging prompt in Appendix E. The results are reported in Table 3 and show the same overall trend as the EM evaluation, with DAC outperforming all baselines overall.

Table 5: **Search accuracy by benchmark.** Search accuracy of retrieved evidence for Search-R1, DART, and DAC, judged by `gpt-oss-120b`. **Bold** numbers indicate the best results for each model.

Method	General QA			Multi-Hop QA				Average
	NQ*	TriviaQA	PopQA	HotpotQA*	2Wiki	MuSiQue	Bamb.	
`Qwen2.5-7B-Instruct`								
Search-R1	69.7	75.4	55.1	58.7	36.5	30.7	54.4	54.3
DART	67.2	74.9	52.1	59.9	**42.3**	28.7	48.8	53.4
DAC	**73.5**	**78.5**	**56.3**	**66.1**	42.1	**34.1**	**56.8**	**58.2**
`Qwen3-8B (Non-Thinking)`								
Search-R1	71.7	76.8	54.5	63.9	42.1	36.0	**61.6**	58.1
DART	71.3	77.2	55.2	63.0	42.9	36.4	60.0	58.0
DAC	**74.5**	**79.8**	**56.7**	**67.1**	**44.0**	**37.7**	**61.6**	**60.2**

Table 6: **Answer accuracy under fixed retrieval by benchmark.** Answer EM when the baseline or DAC generator uses the same retrieval results produced by Search-R1 or DART. **Bold** numbers indicate the best results in each block.

Method	General QA			Multi-Hop QA				Average
	NQ*	TriviaQA	PopQA	HotpotQA*	2Wiki	MuSiQue	Bamb.	
`Qwen2.5-7B-Instruct`								
Search-R1	**43.9**	62.0	42.6	38.5	34.8	15.6	40.0	39.6
DAC	43.4	**63.2**	**43.2**	**41.1**	**37.4**	**16.3**	**42.4**	**41.0**
DART	42.4	61.6	41.0	39.9	**40.6**	**15.7**	40.8	40.3
DAC	**42.8**	**61.8**	**41.6**	**40.8**	39.7	15.2	**42.4**	**40.6**
`Qwen3-8B (Non-Thinking)`								
Search-R1	48.1	66.1	45.3	45.0	43.9	**21.7**	48.8	45.6
DAC	**48.8**	**66.2**	**46.5**	**45.7**	**46.7**	21.5	**52.0**	**46.8**
DART	46.8	66.3	**47.3**	45.3	44.5	20.9	48.0	45.6
DAC	**48.6**	**66.5**	**47.3**	**46.0**	**47.3**	**21.8**	**48.8**	**46.6**

D.2 Comparison with DART under matched training conditions

DART relies on an SFT warm-up stage and special search tag tokens as part of its original training pipeline. To control for these differences, we additionally compare DART and DAC under matched conditions, following the training order specified in the DART codebase. Specifically, we add the tag tokens `<search>` and `</search>` to the tokenizer, perform a lightweight SFT warm-up on 30K self-distilled trajectories, and then train both methods using two LoRA adapters with rank 32 and $\alpha = 64$. The self-distilled trajectories were sampled from the initial model and filtered by final-answer EM correctness. The results are reported in Table 4. DAC continues to outperform DART under this matched setup, suggesting that the gains are not primarily due to differences in initialization, tokenizer design, or adaptation capacity, but instead arise from DAC's training design.

D.3 Full results for disentangled search and generation

We provide the per-benchmark results underlying Figure 2. For search accuracy, we use LLM-based verification with `gpt-oss-120b` (reasoning effort: `medium`), which provides a more reliable measure than substring EM and avoids directly evaluating the baselines with the same lightweight verifier used during DAC training. The resulting search accuracy scores are reported in Table 5. We further report answer EM when DAC generates from fixed retrieval results produced by Search-R1 or DART in Table 6, isolating generation quality from retrieval quality. As discussed in section 5, DAC improves both search and generation quality.

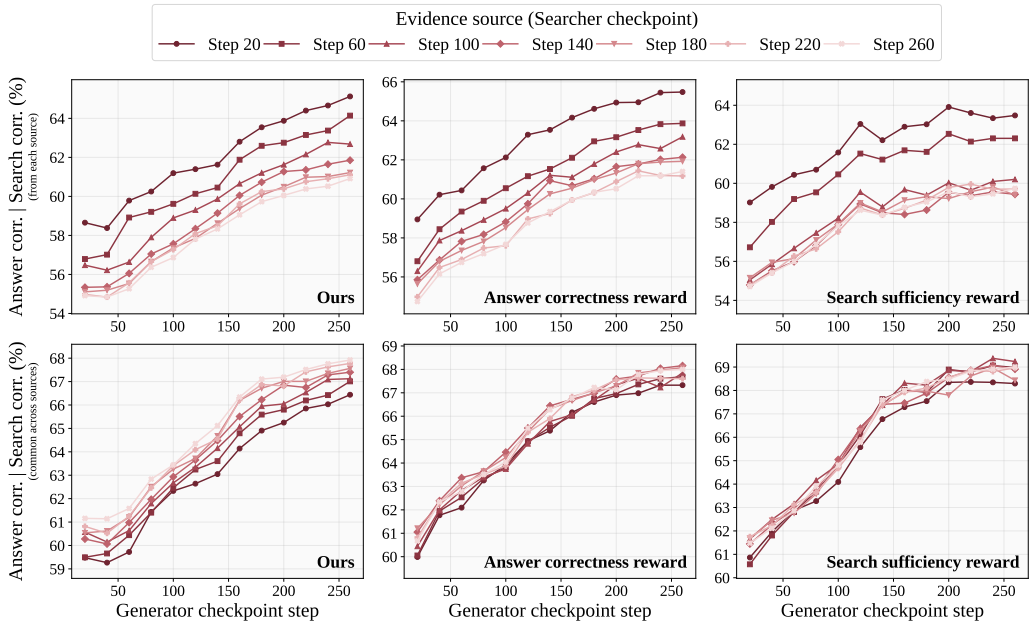


Figure 8: **Cross-checkpoint conditional answer accuracy.** Each column corresponds to a different searcher reward. We evaluate generator checkpoints on evidence produced by different searcher checkpoints, with abstention disabled. *Top*: answer accuracy conditioned on the search-correct examples of each evidence source. *Bottom*: the same evaluation restricted to the intersection of questions retrieved correctly by all evidence source checkpoints, keeping the evaluated question set fixed. Under DAC, later generators perform better on fixed evidence, while the apparent difficulty increase of later search evidence in the top row disappears when question composition is controlled.

D.4 Further analysis of the effect of cross-verification

D.4.1 Cross-checkpoint searcher–generator evaluation

To further examine how the searcher and generator co-adapt during training, we evaluate generators from different checkpoints on evidence produced by searchers from different checkpoints. Abstention is disabled in this analysis, and we report answer accuracy conditioned on correct search. The top row of Figure 8 conditions separately on questions where each searcher checkpoint successfully retrieves correct evidence, whereas the bottom row restricts evaluation to the intersection of questions that are retrieved correctly by all compared searcher checkpoints, thereby keeping the question set fixed. The three searcher reward variants follow the definitions in section 6.1.

Under DAC, the top row exhibits a clear monotonic, ordered structure. For a fixed evidence source, later generators achieve higher answer accuracy, confirming that *the generator becomes better at exploiting the same retrieved evidence*. Conversely, for a fixed generator, evidence from later search checkpoints yields lower aggregate accuracy, yet the bottom row clarifies that this does not arise because later searchers degrade evidence on already-solved questions. When the question set is held fixed, later searchers instead yield comparable or higher answer accuracy. The aggregate decline therefore primarily reflects that *later searchers retrieve correctly on additional, harder questions that earlier searchers fail to cover*. This is consistent with a curriculum in which retrieval progressively expands toward more difficult parts of the task as generation capability improves.

However, the searcher reward ablations show weaker forms of this structure. Answer correctness reward retains some cross-evidence monotonic ordering in the aggregate evaluation, consistent with the searcher still receiving a signal tied to downstream generation. However, this ordering largely disappears on the fixed-question subset, suggesting less room for continued improvement once the searcher already retrieves evidence from which the generator can answer correctly. Search sufficiency reward shows substantially weaker alignment across both evaluations, consistent with its searcher objective being misaligned to the evolving generator.

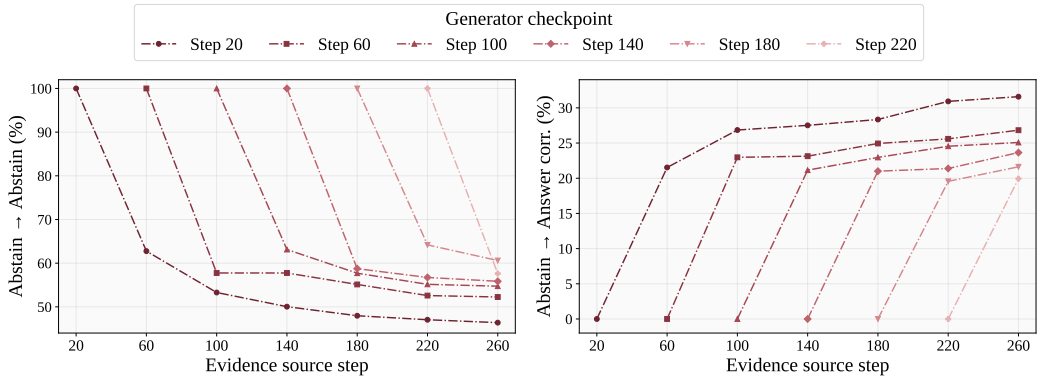


Figure 9: **Effect of later search evidence on previously abstained examples.** For each generator checkpoint, we select questions for which its matched searcher retrieves sufficient evidence but the generator abstains, then keep the generator fixed while replacing the evidence with that produced by later searcher checkpoints. *Left*: fraction that remain abstained. *Right*: fraction for which the fixed generator produces the correct answer. Later search evidence increasingly converts abstained cases into answerable ones.

Table 7: **Shared vs. dedicated LoRA policies.** Answer EM for a shared policy or dedicated policies for the searcher and generator. **Bold** numbers indicate the best results in each block.

Method	General QA			Multi-Hop QA				Average
	NQ*	TriviaQA	PopQA	HotpotQA*	2Wiki	MuSiQue	Bamb.	
`Qwen2.5-7B-Instruct`								
Shared policy	39.6	60.2	38.9	**40.7**	34.1	16.3	40.0	38.5
Dedicated policies	**41.9**	**61.2**	**41.1**	40.3	**35.4**	**17.2**	**41.6**	**39.8**
`Qwen3-8B (Non-Thinking)`								
Shared policy	45.5	65.0	46.5	45.3	44.6	**22.2**	47.2	45.2
Dedicated policies	**49.2**	**66.4**	**47.9**	**45.7**	**46.0**	21.1	**49.6**	**46.6**

D.4.2 How cross-verification reshapes search behavior after abstention

We next focus on cases where, at a given checkpoint, the searcher retrieves sufficient evidence but the corresponding generator abstains. For each such checkpoint, we keep the generator fixed and replace its evidence with that produced by progressively later searchers on the same questions. As shown in Figure 9, later searchers substantially reduce the probability that the fixed generator continues to abstain, while increasing the probability that it produces the correct answer. Thus, an abstention decision not only reflects the generator's current limitation but also provides a learning signal that pushes the searcher toward alternative sufficient evidence that the generator can more readily use. This illustrates how cross-verification can improve cases where retrieval is already sufficient but the generator cannot yet make use of the retrieved evidence.

D.5 Full results for shared vs. dedicated policies

Table 7 reports the full per-benchmark comparison for section 6.4. Dedicated LoRA policies improve average EM on both backbones, from 38.5 to 39.8 on `Qwen2.5-7B-Instruct` and from 45.2 to 46.6 on `Qwen3-8B`, with gains on most benchmarks. This is consistent with the training dynamics in Figure 7, where dedicated policies eventually outperform the shared policy.

D.6 Accuracy when forcing abstained examples to answer

When the generator chooses to abstain and is forced to produce an answer, accuracy drops substantially compared to non-abstained examples. As shown in Table 8, for `Qwen3-8B`, the forced answer accuracy

Table 8: **Answer accuracy by abstention decision.** *Non-abstain* reports answer EM on examples the generator chooses to answer, while *Abstain* reports forced-answer EM on examples it chooses to abstain from.

Benchmark	`Qwen2.5-7B-Instruct`		`Qwen3-8B (Non-Thinking)`	
	Non-abstain	Abstain	Non-abstain	Abstain
NQ	46.09%	13.13%	54.11%	11.68%
TriviaQA	67.77%	13.81%	72.00%	11.22%
PopQA	50.53%	14.42%	55.11%	12.90%
HotpotQA	49.57%	13.30%	52.58%	9.98%
2WikiMultihopQA	47.52%	17.63%	59.14%	20.74%
MuSiQue	26.47%	4.90%	27.86%	4.51%
Bamboogle	46.08%	13.04%	56.07%	5.56%
Average	47.72%	12.89%	53.84%	10.94%

on abstained examples averages 10.94% across benchmarks, compared to 53.84% selective accuracy on non-abstained examples. This confirms that abstention decisions reflect genuine uncertainty about evidence sufficiency rather than arbitrary behavior.

E PROMPT TEMPLATES

This section presents the prompts used in our various experiments.

Red-colored text in brackets indicates placeholders to be filled in. Both agents are prompted through a single user-role message. For readability, we present each prompt as an instruction followed by the runtime input fields. The `[think tag]` is set to `think` for `Qwen2.5-7B-Instruct` and `reason` for `Qwen3-8B`, since `<think>` is a reserved token in `Qwen3`'s native thinking mechanism and would conflict with the model's built-in thinking mode.

Prompts 1 to 3 correspond to the prompts used for the **searcher agent**, the **generator agent** during *training*, and the **generator agent** during **validation**, respectively. The searcher agent is initialized with an instruction and the question. After each retrieval turn, the model's reasoning, search query, and retrieved documents are appended to the context as `<[think tag]>...</[think tag]><search>...</search><information>...</information>` blocks. The training and test prompts for the generator differ in that the training prompt instructs the model to output `unknown` when the answer cannot be found from the provided documents, enabling abstention, while the test prompt omits this instruction since abstention is disabled at test time.

Prompt 1: Searcher agent

Instruction:

You are a research assistant that plans retrieval. Your job is to collect information to answer the given question. You must conduct reasoning inside `<[think tag]>` and `</[think tag]>` first every time you get new information. After reasoning, if you find you lack some knowledge, you can call a search engine by `<search>` query `</search>` and it will return the top searched results between `<information>` and `</information>`. You can repeat this process as many times as you want. If you find no further external knowledge needed, simply output `<stop>`.

Runtime input:

Question: `[Question]`
`[Search history appended after each turn]`

Prompt 2: Generator agent (Training)

Instruction:

You are a knowledgeable assistant that answers questions. You will be provided with a question and some relevant documents. First, think step-by-step about how to answer the question based on the provided documents inside <[think tag]> and </[think tag]>. Then, provide a concise and accurate answer inside <answer> and </answer>. The answer should be in a short phrase or single word if possible. If you cannot find the answer from the provided documents, simply answer unknown inside <answer> and </answer>.

Runtime input:

Question: [Question]
Documents:
[Retrieved documents]

Prompt 3: Generator agent (Test)

Instruction:

You are a knowledgeable assistant that answers questions. You will be provided with a question and some relevant documents. First, think step-by-step about how to answer the question based on the provided documents inside <[think tag]> and </[think tag]>. Then, provide a concise and accurate answer inside <answer> and </answer>. The answer should be in a short phrase or single word if possible.

Runtime input:

Question: [Question]
Documents:
[Retrieved documents]

Prompts 4 and 5 correspond to the prompts used for LLM-based judgement, for answer accuracy and search accuracy, respectively. Since we prompt gpt-oss-120b with reasoning effort medium, we do not explicitly ask the judge to provide reasoning in the prompt.

Prompt 4: LLM-based judging for answer accuracy

Instruction: You will be given a question, answer, and gold answer triple. Your task is to provide a yes/no answer indicating whether the answer is correct compared to one of the gold answers. The answer is considered correct if it matches any of the gold answers within a reasonable degree of similarity. However, the answer should be considered wrong if it is too broad, too narrow, or only partially correct compared to the gold answers. Please answer with 'yes' or 'no' only.

Runtime input:

Question: [Question]
Answer: [Generated answer]
Gold answers: [Gold answers]

Prompt 5: LLM-based judging for search accuracy

Instruction: You will be given a question, some documents, and gold answer triple. Your task is to provide a yes/no answer indicating whether the provided documents contain sufficient information to answer the question with one of the given gold answers. Please answer with 'yes' or 'no' only.

Runtime input:

Question: `[Question]`
Documents: `[Retrieved documents]`
Gold answers: `[Gold answers]`