WHAT DOES ANSWER-BASED CO-TRAINING TEACH LLM SEARCH AGENTS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic search enables large language models to solve knowledge-intensive tasks through repeated retrieval and reasoning. Reinforcement learning trains these agents with final-answer rewards, but sparse feedback gives limited guidance for intermediate search decisions. Answer-based co-training addresses this problem by training a model to answer at intermediate steps and using improvements in answer quality as process rewards. However, how to combine these two training signals into a stronger search agent remains unclear. We investigate four design questions through controlled comparisons with full co-training. First, for parameter sharing, we compare shared and separate models: sharing parameters improves the deployed policy's answers on the same evidence by applying intermediate-answer updates to the policy itself. Second, for training signals, we separate intermediate-answer training from process rewards: intermediate-answer training improves early responses and robustness to changed answering instructions. Third, for reward placement, we compare stepwise delivery with terminal aggregation: accumulating process rewards at the trajectory end provides an effective alternative while preserving their sum. Fourth, for feedback information, we compare agent performance after training with or without prior reasoning in the answer model's input. We separately compare scoring criteria: continued-search scoring identifies useful clues better than immediate-answer scoring by testing whether the policy can follow them to the answer.[1]

1 INTRODUCTION

Agentic search enables large language models (LLMs) to solve knowledge-intensive problems by acquiring external evidence over multiple rounds of retrieval and reasoning (Yao et al., 2023; Trivedi et al., 2022a). A search agent formulates queries, interprets the returned passages, and decides whether to gather more information or answer the question. This interaction makes both information acquisition and answering central to success: the agent must find useful evidence and convert it into a correct response.

Reinforcement learning (RL) offers a way to train these behaviors using verifiable final-answer rewards (Jin et al., 2025; Song et al., 2025). Such rewards evaluate the completed interaction, while the agent makes many intermediate decisions about what to search for and how to use the results. Sparse outcome feedback makes credit assignment difficult, motivating process supervision that supplies feedback throughout the search trajectory (Zhang et al., 2025; Xie et al., 2026).

Answer-based co-training provides process supervision through intermediate answers. In OASES (Zhang et al., 2026), an answer model uses the evidence available at each search step, and an external scorer checks its answer against the reference. These scores serve two purposes: intermediate-answer training updates the answer model using its own answer score, while process rewards update the search policy using improvements in that score across search steps. Both shared and separate parameter configurations provide process rewards. Sharing parameters additionally lets intermediate-answer training update the search policy itself.

This combination raises four design questions. First, should the search policy and answer model share parameters? Second, what do intermediate-answer training and process rewards each con-

[1]The anonymous code is available at https://anonymous.4open.science/r/answer-based-cotraining-2646/

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.

tribute? Third, should process rewards be delivered after each retrieval or accumulated at the trajectory end? Fourth, what information should search feedback evaluate? We compare agent performance after training with or without prior reasoning in the answer model's input. Separately, we compare how immediate-answer scoring and continued-search scoring identify useful retrieved evidence.

In this paper, we study the complete shared-parameter configuration, denoted *Full co-training* (Full), through four corresponding comparisons. We first compare shared and separate parameter configurations on the same questions and retrieved passages. We then retain intermediate-answer training while removing process rewards, and compare this control with both Full and outcome-only training to identify the two signals' contributions. Next, we move the same process-reward sum from retrieval steps to the trajectory end. Finally, we compare Full with Reasoning-masked to measure how removing prior reasoning from the answer model's training input affects agent performance. With trained policies held fixed, we also compare immediate-answer scoring with success after further search, using proposed queries on natural questions and synthetic search tasks that require following a sequence of clues.

Our contributions are summarized as follows:

- **Parameter sharing.** We compare Full with separate training and show that shared intermediate-answer updates improve the deployed search policy's answers on common evidence.
- **Training signals.** We separate intermediate-answer training from process rewards and show that intermediate-answer training improves early answering and robustness to changed instructions.
- **Reward placement.** We compare rewards after each retrieval with their terminal sum, showing that terminal aggregation preserves the trajectory-reward objective and remains effective in practice.
- **Feedback information.** We measure how removing prior reasoning from the answer model's training input affects agent performance. We also compare immediate-answer scoring with continued-search scoring, showing why subsequent task completion better recognizes useful search clues.

2 Answer-Based Co-Training for Search

Answering knowledge-intensive questions often requires gathering evidence through multiple rounds of search. Given a question, a search policy π_θ generates queries, reasons over the retrieved passages, and decides whether to continue searching or produce a final answer. Its *search context* contains the question, previous reasoning and queries, and retrieved passages available at the current step. State s_t combines this context with the remaining interaction budget. A search adds an observation; an answer ends the episode. Outcome-only reinforcement learning optimizes an external score $R(a, y)$ of the final answer against reference y.

Two training signals from intermediate answers. At retrieval step t, an answer generator g_ψ answers from s_t under its training prompt. External scoring gives

$$\Phi_\psi(s_t) = \mathbb{E}_{a\sim g_\psi(\cdot|s_t)}[R(a, y)], \qquad r_t^{\text{ans}} = \alpha\big(\widehat{\Phi}_\psi(s_t) - \widehat{\Phi}_\psi(s_{t-1})\big). \tag{1}$$

Here $\widehat{\Phi}$ denotes a sampled estimate of answer quality. *Intermediate-answer training* uses each generated answer's own score to update g_ψ with PPO. *Process rewards* use the score difference r_t^{ans} to update the search policy, in addition to its final-answer reward. The reference is available only to the scorer. We study these two uses of intermediate answers in OASES (Zhang et al., 2026).

Full co-training and its variants. *Full co-training* (Full) is the reference configuration: it combines shared parameters, intermediate-answer training, and process rewards after each retrieval. *Separate* removes parameter sharing, training an independent answer generator whose scores supply the policy's process rewards. *Auxiliary-only* (Aux-only) removes Full's process rewards by setting $\alpha = 0$, retaining shared intermediate-answer training. Outcome-only removes both intermediate-answer training and process rewards. *Terminal* moves Full's process rewards to the trajectory end

Configuration	Intermediate-answer training	Shared parameters	Process-reward placement	Answer input	Research question
Outcome-only	No	—	None	—	Training signals
Aux-only ($\alpha = 0$)	Yes	Yes	None	All context	Training signals
Separate	Yes	No	After retrieval	All context	Parameter sharing
Terminal	Yes	Yes	End of trajectory	All context	Reward placement
Reasoning-masked	Yes	Yes	After retrieval	Queries/evidence	Feedback information
Full co-training	Yes	Yes	After retrieval	All context	All questions

Table 1: Training configurations and their corresponding research questions. Full co-training (Full), highlighted in the final row, is the reference configuration. Aux-only versus Outcome-only assesses intermediate-answer training; Full versus Aux-only assesses process rewards. All configurations receive the final-answer reward; process rewards use Eq. (1) with coefficient 0.5.

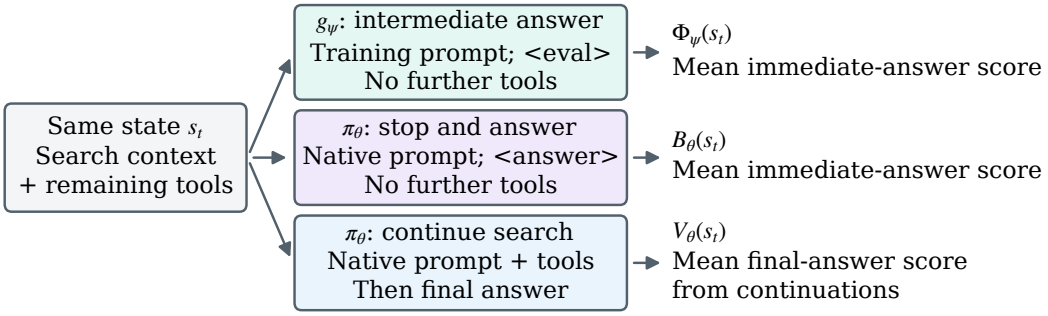

Figure 1: Three evaluations starting from the same retrieval state s_t. The first two answer from the available evidence under different instructions; the third allows further search. Each score averages the resulting answers.

while preserving their sum. *Reasoning-masked* removes tagged reasoning from the answer generator's training input, retaining queries and retrieved evidence. The policy keeps its full search context. Table 1 summarizes the variants. Complete-task evaluation uses only the search policy; diagnostics also evaluate the answer generator.

3 Evaluating Answering and Search Decisions

At a selected retrieval state s_t, we evaluate three ways of answering the same question (Figure 1):

$$\begin{aligned} \Phi_\psi(s_t) &= \mathbb{E}[R(a, y) \mid s_t,\ g_\psi \text{ uses intermediate-answer instructions}], \\ B_\theta(s_t) &= \mathbb{E}[R(a, y) \mid s_t,\ \pi_\theta \text{ uses stop-and-answer instructions}], \\ V_\theta(s_t) &= \mathbb{E}[R(a_{\text{final}}, y) \mid s_t,\ \pi_\theta \text{ may use tools before answering}]. \end{aligned} \tag{2}$$

For Φ, the model uses the answer generator's training prompt: answer examples and a short `<eval>` response. For B, it retains the native search-policy prompt and context, receives a stop-search instruction, and can reason before producing an `<answer>`. Neither permits further tool use. For V, the policy can use its remaining tool budget before producing its final answer.

State scores and training rewards. All three quantities are defined at each retrieval state, including s_0 before search. A completed trajectory has one observed final-answer score; $V(s_t)$ averages such scores over continuations *from that state*. Different prefixes can therefore have different V values. Our diagnostics evaluate selected states. The process reward r_t^{ans} is the change in estimated Φ between consecutive states (Eq. (1)). For example, suppose a synthetic search task asks for an answer string, but retrieval reveals only "the answer is in record B." $\Phi(s_1)$ and $B(s_1)$ score answers without another search. For $V(s_1)$, the policy can retrieve B; we average final-answer scores over repeated continuations. At the resulting state s_2, all three evaluations can be repeated using the new evidence.

Answering instructions and parameter sharing. $\Phi_\psi(s_t)$ and $B_\theta(s_t)$ compare the two immediate-answer protocols. Full shares weights ($\psi = \theta$), but uses different prompts. Identical weights, complete inputs, decoding, and scoring would make these expectations equal. Separate additionally uses different weights. RQ1 measures the sharing benefit by comparing Full and Separate under each fixed answering protocol.

Selecting queries in RQ4. After training, we hold model parameters fixed and test whether feedback selects queries that lead to better final answers. We execute candidate queries from the same search state, then compare selection by immediate-answer scores with selection by scores after continued search. For $\alpha > 0$, choosing the query with the highest immediate-answer score also selects the highest process reward in Eq. (1): all candidates subtract the same starting-state score. The receiving policy continues from each candidate to measure final-answer quality; continued-search selection uses separate rollout samples for choosing and evaluating queries.

4 Experimental Setup

Models and benchmark evaluation. All agents start from Qwen2.5-7B-Instruct (Qwen Team, 2024) and use PPO (Schulman et al., 2017) trained on HotpotQA (Yang et al., 2018) and 2WikiMultihopQA (Ho et al., 2020). Evaluation adds Natural Questions (Kwiatkowski et al., 2019), MuSiQue (Trivedi et al., 2022b), and Bamboogle (Press et al., 2022), for 26,133 benchmark examples. Retrieval uses E5 (Wang et al., 2022) over Wikipedia. We evaluate all six training configurations at update 2,418, including Full versus Terminal for RQ3 and Full versus Reasoning-masked for RQ4. Aux-only uses $\alpha = 0$; configurations with process rewards use $\alpha = 0.5$. Initial denotes the model before PPO; Outcome-only at update 5,035 provides a training-duration comparison.

Answering from a fixed search context. After removing normalized exact matches with training questions, we select 200 diagnostic questions, 40 per dataset, independently of scores. Contexts from Outcome-only, Full, and Separate yield 991 states, including 791 observed-evidence states. Each model receives both intermediate-answer and policy stop-and-answer instructions. This includes the Separate generator and policies trained without intermediate-answer supervision. We compare question-only, first-observation, and last-observation states, then change answering instructions. These diagnostics measure parameter sharing (RQ1) and intermediate-answer training (RQ2).

Choosing queries on natural QA questions. For RQ4, we evaluate fixed checkpoints on a 75-question subset, taking 15 questions per benchmark from the diagnostic pool. From each question's fixed search context, the three source policies propose queries. Deduplication independent of scores retains 218 query-and-evidence candidates; 66 questions have multiple candidates. A *feedback provider* answers immediately from each candidate's retrieved evidence, and an external scorer evaluates its answer. A *receiving policy* continues searching from each candidate. We compare immediate-answer, uniform, and continued-search selection; the latter uses separate continuation samples for selection and evaluation. For tied maxima, we average the receiver's outcomes across tied candidates.

Synthetic search through a sequence of clues. We construct 40 two-record and 40 three-record tasks. A record is a short text entry pointing to the next record or storing a random *answer string* (a character sequence, not program code). Retrieving and copying this string tests search and answer production without relying on memorized facts. For example, A points to B, B points to C, and C stores `K7M2Q9`; the question asks for the string reached from A. These are readable aliases for a saved task. We supply a *useful clue* (A→B), a *distractor* (D→E), or the full chain including the *direct answer*. Continuations permit one or two further searches for two- or three-record tasks, respectively. RQ2 compares Full and Aux-only from a useful clue; RQ4 compares immediate-answer and continued-search scores (Table 5).

Measurements. Benchmarks use greedy decoding; diagnostics use temperature 1, with two answers per common state, four per candidate or synthetic-search state, and four independent continuations per candidate or synthetic-search state. We report token F1, EM, and output behavior. Immediate-answer F1 requires a unique tagged answer and rejects generated tool interactions.

Configuration	HotpotQA	2Wiki	NQ	MuSiQue	Bamboogle	Macro
Initial	38.20	23.85	33.31	18.59	42.79	31.35
Outcome-only	58.69	58.04	43.57	29.66	57.32	49.45
Aux-only	59.41	59.08	**43.88**	29.72	57.60	49.94
Separate	57.49	58.27	42.40	29.30	58.50	49.19
Terminal	60.16	**60.18**	43.80	30.86	59.23	**50.84**
Reasoning-masked	58.32	57.73	43.07	**32.07**	58.23	49.88
Full co-training	**60.53**	58.07	43.73	30.29	**59.84**	50.49

Table 2: Final-answer F1 at update 2,418 (Initial is untrained). The shaded row is Full co-training; bold numbers mark column maxima. Macro weights datasets equally.

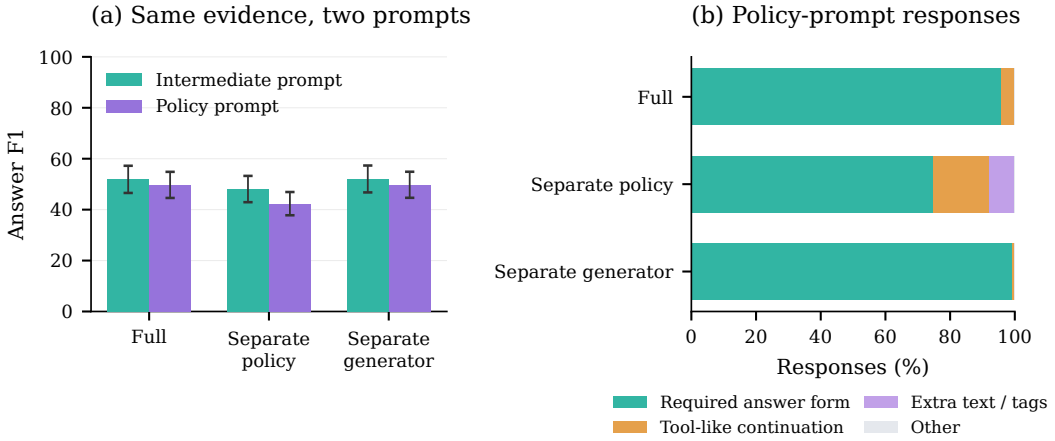

Figure 2: Parameter sharing on identical contexts from 200 questions: Full versus Separate's policy and independent answer generator. Left: F1 under both answering prompts, with 95% question-bootstrap intervals. Right: response types under policy stop-and-answer instructions. Tools are disabled. Figure 4 includes Initial, Outcome-only, and Aux-only.

Usable-output rates and conditional content scores describe response production and answer quality. Diagnostics average samples within state and states within question. Intervals use 5,000 paired question-bootstrap draws; benchmark contrasts resample question clusters within dataset before an equal-weight macro average. Appendices B and C give training configurations, input provenance, protocols, and aggregation details.

5 Understanding Full Co-Training

The results follow four research questions. Table 1 maps configurations to questions, and Table 2 summarizes complete-task scores. Full is the reference configuration.

5.1 RQ1: Should the search policy and answer model share parameters?

Full gives the search policy better answers than its Separate variant on the same evidence. Under intermediate-answer instructions, Full scores 52.07 F1 and the Separate policy scores 48.21. The paired difference is 3.86 [1.22, 6.72] points. Under policy stop-and-answer instructions, Full leads by 7.33 [4.89, 9.85] points (Figure 2; Table 10).

The independent answer generator reaches 52.17 intermediate-answer F1 and exceeds its own policy under the same instructions by 3.95 [1.04, 7.03] points. Full and Separate both receive process rewards. The additional path in Full is intermediate-answer training: PPO updates from generating intermediate answers also train the search policy. In Separate, these updates train only the independent generator. Sharing parameters thus combines search training with practice in answering

at intermediate steps. The source-by-reader comparison shows the same pattern across contexts produced by different policies (Table 11).

Figure 5 makes this distinction concrete. The retrieved passage names Lacey Chabert as the original voice, alongside a distractor naming Alex Borstein. Under the same stop-and-answer instructions, Full answers Lacey Chabert and the Separate policy answers Alex Borstein. The Separate generator also answers correctly under intermediate-answer instructions.

5.2 RQ2: What do intermediate-answer training and process rewards contribute?

Intermediate-answer training provides a direct training benefit. Aux-only removes Full's process rewards while retaining shared intermediate-answer training. Comparing it with Outcome-only measures the answer-training contribution. It obtains 52.93 intermediate-answer F1, compared with 47.53 for Outcome-only, a paired gain of 5.40 [2.78, 8.14] points. Usable-output rates exceed 99.8% for both models under these instructions. Aux-only also improves policy stop-and-answer F1 by 3.00 [0.64, 5.30] points. Table 3 gives both removal comparisons.

Comparison	Intermediate F1	Policy F1	Task macro F1
Aux-only − Outcome-only	5.40 [2.78, 8.14]	3.00 [0.64, 5.30]	0.48 [-0.85, 1.80]
Aux-only − Full	0.86 [-1.15, 2.89]	-1.64 [-3.99, 0.56]	-0.55 [-1.92, 0.75]

Table 3: Auxiliary-only control: paired F1 differences with 95% question-bootstrap intervals. Answering uses 200 questions and identical search contexts; the complete task uses 26,133 examples across five datasets.

Early answering. Intermediate-answer training gives the policy practice responding before search is complete. Its benefit extends from question-only inputs to the last observation (Appendix D.4).

Under policy stop-and-answer instructions, the first-observation gain is 11.82 [6.87, 16.86] F1 points. The usable-output rate rises from 50.75% for Outcome-only to 85.13% for Aux-only. At the last observation, both rates exceed 99.9% and the F1 difference is -0.12 [-2.62, 2.21]. Intermediate-answer training is especially useful for producing a valid answer when the policy is asked to respond early.

Changed answering instructions. Removing answer examples and changing the requested tag yields 45.3 F1 for Aux-only and 46.4 for Full, compared with 29.1 for Outcome-only (Table 14). On the same questions, the Aux-only–Outcome gap grows by 11.73 [5.75, 18.60] points under this combined prompt change. Full shows the same direction of improvement. Intermediate-answer training thus provides stronger performance when the agent must answer under altered instructions.

Appendix D.2 separates answer production from conditional content; Appendices D.4 and D.5 provide the stage and instruction comparisons.

Process rewards: evidence acquisition and final-answer accuracy. Comparing Full with Aux-only adds process rewards while retaining shared intermediate-answer training. In the synthetic search task, a useful clue identifies the next record, and the policy must follow the sequence of clues to the requested answer string. Table 4 separates terminal-record acquisition from correct final-answer generation. On three-record tasks, Full completes every sampled continuation, compared with 90.63% for Aux-only. The paired Full–Aux-only difference is 9.38 [3.75, 16.88] points. Aux-only retrieves the terminal record in 97.50% of continuations; its remaining failures include 6.88 percentage points of incorrect answers after that record is available. The process-reward benefit in this task therefore includes more reliable conversion of acquired information into the final answer.

Complete-task performance. Full obtains 50.49 macro F1, Aux-only 49.94, and Outcome-only 49.45. The paired Aux-only–Outcome-only difference is 0.48 [-0.85, 1.80] points; Aux-only–Full is -0.55 [-1.92, 0.75]. Appendix D.3 reports the dataset-level F1 contrasts.

Policy	Terminal record seen	Seen but incorrect	Correct answer
Outcome-only	94.38	6.88	87.50
Aux-only	97.50	6.88	90.63
Full	100.00	0.00	100.00
Separate policy	90.63	0.00	90.63

Table 4: Three-record synthetic searches from a useful clue with two new searches (40 instances, 160 continuations per policy). All entries are percentages of all continuations. Correct equals terminal record seen minus seen but incorrect in every saved rollout. The Aux-only–Full success gap occurs mainly after the terminal information has been obtained.

Condition	Supplied evidence	Immediate answer $\widehat{\Phi}$	Continued search $\widehat{V}$
Useful clue	A points to B	B; correct: 0/4	Searches B, then C; returns `K7M2Q9`; correct: 4/4
Distractor	D points to E	D; correct: 0/4	Searches E; returns `R4T8W1`; correct: 0/4
Direct answer	A points to B; B points to C; C stores `K7M2Q9`	`K7M2Q9`; correct: 4/4	Returns `K7M2Q9`; correct: 4/4

Table 5: A saved synthetic-search task using Full. The question asks for the answer string reached from A. Record names and strings are readable aliases. Immediate answering uses `<eval>` without tools; continued search allows two further searches and ends with `<answer>`. Scores use the same target string, `K7M2Q9`. Counts show exact matches among four samples; dividing by four gives the score.

5.3 RQ3: Should process rewards be delivered stepwise or at the end?

Terminal retains Full's shared parameters, intermediate-answer training, and answer-score differences, but accumulates the process rewards at the final reward position. It reaches 50.84 macro F1, the highest macro score among the primary configurations, compared with Full's 50.49 (Table 2). Terminal placement is therefore an effective way to deliver the co-training reward.

The distinction is reward timing. With undiscounted returns and each sampled score reused in adjacent differences, both placements give the same total on a fixed trajectory:

$$\sum_{t=1}^{T} \alpha\big(\widehat{\Phi}(s_t) - \widehat{\Phi}(s_{t-1})\big) = \alpha\big(\widehat{\Phi}(s_T) - \widehat{\Phi}(s_0)\big). \tag{3}$$

Terminal still generates intermediate answers, trains on their scores, and uses the full sum of process rewards. It changes where that sum enters PPO's returns. Thus reward placement can be chosen separately from the intermediate-answer task and the trajectory-reward objective. Appendix A analyzes the corresponding returns and advantages, including the role of learned critics and finite-sample optimization.

5.4 RQ4: What information should search feedback evaluate?

RQ4 compares agent performance after changing the answer model's training input. It separately compares immediate-answer scoring with continued-search scoring while holding trained weights fixed.

Agent performance under the input ablation. Full includes the question, prior reasoning, queries, and passages in the answer model's training input; Reasoning-masked removes only prior tagged reasoning. Both answer branches use the same tool-free `<eval>` output, scorer, shared parameters, and stepwise process rewards. The search policy retains its full context. Appendix E.1 illustrates the training inputs. On complete tasks, the resulting Reasoning-masked policy reaches 49.88 macro F1, compared with Full's 50.49 (Table 2). Full gives higher F1 on HotpotQA, 2Wiki, Natural Questions, and Bamboogle; Reasoning-masked gives higher F1 on MuSiQue.

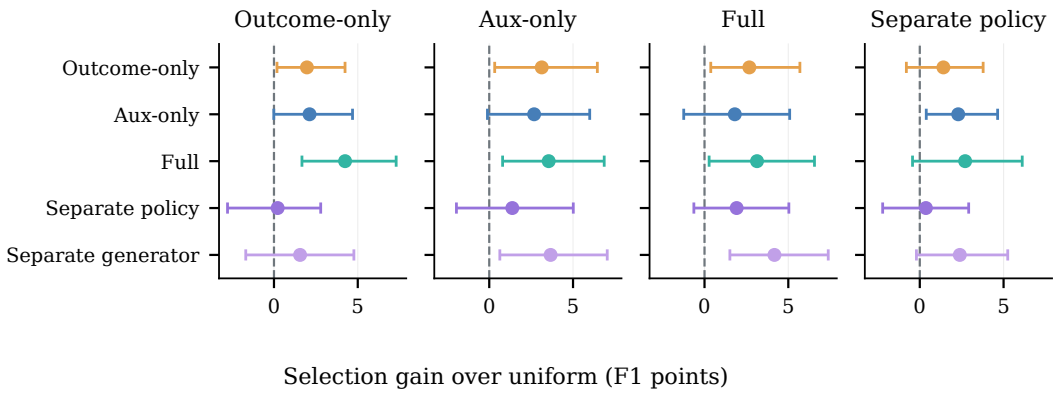

Figure 3: Selection gain over uniform on 66 common starting states with multiple candidate queries. Providers score the same observations, and receivers continue from every candidate. Maximum-score ties are averaged. Error bars are paired 95% question-bootstrap intervals.

Immediate answers or continued search. Table 5 evaluates the synthetic A→B→C task from Section 4. The useful clue supplies A→B; the distractor supplies the unrelated D→E. Full's immediate answers return record names B and D, both incorrect. With further search, it retrieves B and C to find the target answer string, but follows E to an unrelated string after the distractor. Supplying the complete relevant chain makes the answer available immediately. The two scores thus measure whether evidence already supports an answer or enables further search to find it. Table 16 aggregates both chain lengths; Appendix E.2 compares model behaviors on this task.

Candidate selection on natural questions. Each candidate is a query with its retrieved passages. Immediate-answer selection uses Full's answer scores; continued-search selection uses the receiver's eventual scores, with independent folds for selection and evaluation. Table 6 shows gains over uniform for all four receivers, and gains over immediate-answer scoring with intervals above zero for Aux-only and Separate.

Receiver	Over uniform	Over immediate-answer
Outcome-only	5.3 [2.4, 8.5]	1.02 [-0.54, 2.81]
Aux-only	7.4 [3.8, 11.4]	3.85 [1.52, 6.64]
Full	6.2 [2.8, 10.1]	3.04 [-0.11, 6.75]
Separate policy	6.1 [2.7, 10.2]	3.38 [0.21, 7.47]

Table 6: Continued-search selection gains over two baselines (F1 points), with paired 95% question-bootstrap intervals on 66 questions. The immediate-answer baseline uses Full as feedback provider.

Figure 3 compares gains over uniform query selection for each pairing of feedback provider and receiving policy. Feedback quality also depends on the receiving policy. The Separate-generator–Full provider difference changes by 3.72 [1.21, 6.88] points between Full and Outcome-only receivers. Feedback should therefore be evaluated with the policy that will use it. Appendix D.6 defines this interaction and compares the candidate-scoring criteria.

6 Design Implications

Parameter sharing: train the model that will answer. Full gives the deployed policy better answers than Separate on common evidence (RQ1). Both receive process rewards, while Full also applies intermediate-answer updates to the policy. These results favor parameter sharing when the search policy produces the final response.

Training signals: develop answering and search completion together. The removal controls in RQ2 distinguish answering behavior from complete search performance. Intermediate-answer

training improves early response production and performance under changed instructions. In the synthetic search task, Full converts acquired evidence into correct final answers more reliably than Aux-only: Aux-only sometimes retrieves the terminal record but returns an incorrect answer string. Evaluation should therefore distinguish finding the required evidence from using it to produce the final answer.

Reward placement: separate timing from the objective. RQ3 makes terminal aggregation a practical alternative to stepwise delivery. It retains intermediate-answer training and the same total process reward. The choice concerns PPO returns and credit assignment within a common trajectory objective, so it can be tuned without discarding intermediate-answer supervision.

Feedback information: evaluate evidence through the search it enables. In RQ4, retaining prior reasoning in the answer model's training input gives a higher macro score than removing it. Separately, for passages that reveal where to search next, continued search measures whether the receiving policy can follow the clue and finish the task. This motivates feedback that credits useful continuation alongside immediate answer quality, building on future-success supervision (Setlur et al., 2025).

7 Related Work

Learning to search. Retrieval-augmented generation conditions answers on external evidence (Lewis et al., 2020). WebGPT learns browsing from demonstrations and feedback (Nakano et al., 2021); ReAct and IRCoT interleave reasoning and information acquisition (Yao et al., 2023; Trivedi et al., 2022a). Search-R1 and R1-Searcher learn search with answer rewards (Jin et al., 2025; Song et al., 2025). We examine how shared intermediate-answer training improves the deployed search policy and evaluate the resulting answers on common evidence.

Coordinated roles. MMOA-RAG jointly optimizes retrieval and generation roles with an outcome reward (Chen et al., 2025a); MAO-ARAG learns query-dependent agent workflows (Chen et al., 2025b). M-ASK separates search from knowledge management (Chen et al., 2026a), while JADE jointly optimizes planning and execution through a shared backbone (Chen et al., 2026b). Our shared/separate comparison measures the benefit of applying intermediate-answer training to the deployed search policy. The auxiliary-only control further characterizes this training task under outcome-only search rewards.

Process supervision and policy-dependent feedback. Process supervision evaluates reasoning steps (Lightman et al., 2023); ReasonRAG constructs supervision for query generation, evidence extraction, and answering (Zhang et al., 2025). VSRM instead scores answers from partial reasoning (Yue et al., 2025), and OASES uses answer differences for search while co-training the answering role (Zhang et al., 2026). TIPS uses changes in reference-answer likelihood and studies teacher–policy compatibility (Xie et al., 2026). Rewarding Progress defines process advantage using future success under a prover and shows why a stronger prover need not provide better training feedback (Setlur et al., 2025). Our study connects parameter sharing, the two training signals, reward placement, and feedback information through configuration ablations and answering and search diagnostics.

8 Conclusion

We study four design questions in answer-based co-training for LLM agentic search. Sharing parameters improves the deployed policy's answers over separate training. Intermediate-answer training strengthens early responses and robustness to changed instructions. Terminal aggregation provides an effective alternative to stepwise reward delivery under the same trajectory-reward objective. Evaluating subsequent task completion better identifies evidence that supplies clues for further search. Together, these findings guide who receives intermediate-answer updates, which training signals to combine, when to deliver rewards, and how to evaluate search feedback.

AI USE STATEMENT

We used generative AI assistants to help implement and edit research code, analyze and interpret experimental results, and draft and edit parts of this paper. We reviewed AI-assisted code and manuscript text, checked mathematical arguments and reported results against the derivations and experiment records. The authors take responsibility for the final code, results, claims, and text.

ETHICS STATEMENT

This study uses public question-answering benchmarks and synthetic controlled search tasks to analyze the behavior of language-model search agents. The study introduces no human-subject data collection, and any redistribution of benchmark or retrieved material remains subject to the original data licenses.

REPRODUCIBILITY STATEMENT

Appendix B specifies the training configurations and checkpoint selection; Appendix C describes the common-state interventions, candidate construction, decoding, and scoring. Saved model outputs and input pairing are checked before analysis. Appendix A derives the relationship between stepwise and terminal reward placement.

REFERENCES

Yiqun Chen, Lingyong Yan, Weiwei Sun, Xinyu Ma, Yi Zhang, Shuaiqiang Wang, Dawei Yin, Yiming Yang, and Jiaxin Mao. Improving retrieval-augmented generation through multi-agent reinforcement learning. *arXiv preprint arXiv:2501.15228*, 2025a. URL https://arxiv.org/abs/2501.15228.

Yiqun Chen, Erhan Zhang, Lingyong Yan, Shuaiqiang Wang, Jizhou Huang, Dawei Yin, and Jiaxin Mao. MAO-ARAG: Multi-agent orchestration for adaptive retrieval-augmented generation. *arXiv preprint arXiv:2508.01005*, 2025b. URL https://arxiv.org/abs/2508.01005.

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 preprint arXiv:2601.04703*, 2026a. URL https://arxiv.org/abs/2601.04703.

Yiqun Chen, Erhan Zhang, Tianyi Hu, Shijie Wang, Zixuan Yang, Meizhi Zhong, Xiaochi Wei, Yan Gao, Yi Wu, Yao Hu, and Jiaxin Mao. JADE: Bridging the strategic-operational gap in dynamic agentic RAG. *arXiv preprint arXiv:2601.21916*, 2026b. URL https://arxiv.org/abs/2601.21916.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*, 2020. URL https://arxiv.org/abs/2011.01060.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025. URL https://arxiv.org/abs/2503.09516.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019. URL https://aclanthology.org/Q19-1026/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021. URL https://arxiv.org/abs/2112.09332.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 278–287, 1999. URL https://people.eecs.berkeley.edu/~russell/papers/icml99-shaping.pdf.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022. URL https://arxiv.org/abs/2210.03350.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.08146.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-Searcher: Incentivizing the search capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025. URL https://arxiv.org/abs/2503.05592.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. *arXiv preprint arXiv:2212.10509*, 2022a. URL https://arxiv.org/abs/2212.10509.

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

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

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. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/fd881d3b625437354d4421818f81058f-Abstract-Conference.html.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018. URL https://arxiv.org/abs/1809.09600.

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

Chuhuai Yue, Chengqi Dong, Yinan Gao, Hang He, Jiajun Chai, Guojun Yin, and Wei Lin. Promoting efficient reasoning with verifiable stepwise reward. *arXiv preprint arXiv:2508.10293*, 2025. URL https://arxiv.org/abs/2508.10293.

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. In *Advances in Neural Information Processing Systems*, 2026. URL https://arxiv.org/abs/2604.03675.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, et al. Process vs. outcome reward: Which is better for agentic RAG reinforcement learning. *arXiv preprint arXiv:2505.14069*, 2025. URL https://arxiv.org/abs/2505.14069.

A RQ3: Stepwise and Terminal Reward Placement

Let $z_t = \widehat{\Phi}(s_t)$ be the sampled immediate-answer score at retrieval boundary t, with each sample reused in adjacent differences. Each difference $\alpha(z_t - z_{t-1})$ is a process reward. For a fixed trajectory with T retrieval boundaries, the sum of its answer-based rewards is

$$\sum_{t=1}^{T} \alpha(z_t - z_{t-1}) = \alpha(z_T - z_0). \tag{4}$$

The terminal-placement configuration generates and trains on the same answer branches, but moves the actual sum of these increments to the final reward position. With undiscounted returns, it preserves the trajectory's total reward:

$$G = R(a_T, y) + \alpha(z_T - z_0). \tag{5}$$

Here $R(a_T, y)$ scores the policy's final answer, while z_T scores the answer-generator branch at the last retrieval state. Both placements optimize the same trajectory reward, including this last branch score. This boundary term connects the construction to the conditions for policy-invariant potential shaping (Ng et al., 1999).

The placement change also has a simple relationship to credit assignment. At a generated token position j, let $C_{<j}$ be the sum of intermediate rewards already delivered under stepwise placement. Terminal placement increases the reward-to-go from that position by $C_{<j}$. When the value function conditions on the corresponding context, its exact value increases by the same amount, leaving the exact advantage unchanged. For sampled branch scores, this statement applies to a context augmented with those samples, or in conditional expectation when the critic does not observe them.

Learned critics, finite batches, normalization, and clipped PPO updates can produce different optimization trajectories under the two placements. The empirical comparison evaluates these two ways of delivering the same reward sum.

B Implementation and Evaluation Details

B.1 Models and training configurations

All trained models start from Qwen2.5-7B-Instruct. The search policy is an autoregressive model that emits reasoning, a tagged search query, or a final answer. Search results are inserted into the context by the environment, and their tokens are excluded from the policy loss. The answer generator receives a question and a partial search context and produces a short answer under a separate instruction template. It is trained on its own externally scored answers.

The training recipe uses PPO with a batch of 256 questions, an actor learning rate of 2×10^{-7}, a critic learning rate of 10^{-6}, a mini-batch size of 32, and one PPO epoch per rollout batch. The return discount and GAE parameter are both one. Process rewards have weight $\alpha = 0.5$ when present; Auxiliary-only uses $\alpha = 0$ while retaining the shared intermediate-answer training and terminal task reward. The configurations use token F1 as the answer reward and retain their format checks. The diagnostic experiments evaluate the trained checkpoints with their parameters held constant.

Each primary configuration is represented by one training run, evaluated at update 2,418. Full co-training generates and optimizes answer branches in the policy model. The separate configuration has independent actor, critic, and optimizer states for the two roles. The separate run uses two four-GPU pools and CPU offloading, whereas the Full run uses a joint pool. Appendix D.1 treats additional outcome-only training as a duration control within the same configuration.

Aux-only was selected at update 2,418 before evaluation. It reuses the original diagnostic states, candidate observations, prompts, and decoding settings. The primary comparison matches update count; branch generation, separate optimizers, and GPU allocation contribute to each configuration's compute.

Setting	Value
Benchmark maximum prompt tokens	2,500
Diagnostic maximum prompt tokens	8,192
Response or answer-generation token cap	2,000
Maximum search calls in benchmark evaluation	6
Candidate query token cap	256
Benchmark temperature	0
Diagnostic temperature / top-p	1 / 1
Answers per common state and protocol	2
Answers per candidate or synthetic-search state and protocol	4
Continuations per candidate and receiver	4 (two independent folds)
Maximum distinct candidate queries per parent	4
Data-parallel workers / tensor-parallel size	8 / 1

Table 7: Inference settings. The benchmark and diagnostic protocols use the same tokenizer, search interface, answer scorer, and model family.

B.2 Data and retrieval

The benchmark pool contains 7,405 HotpotQA questions, 12,576 2WikiMultihopQA questions, 3,610 Natural Questions, 2,417 MuSiQue questions, and 125 Bamboogle questions. HotpotQA was also used for validation during training. The remaining datasets expand evaluation across different question and evidence distributions.

Retrieval uses E5 (Wang et al., 2022) over Wikipedia through a shared service. Each query requests three passages, each truncated to 200 tokenizer tokens. Cached responses ensure that all models scoring a candidate receive the same observation. Failed requests are retried and are not counted as incorrect answers. Aux-only reuses the original cache and records any misses separately.

B.3 Decoding and execution

Table 7 summarizes inference settings. Benchmark rollouts use greedy decoding. A native rollout has a response-token budget that includes inserted tool observations; sampled forced answers have a generation-token cap. Continued-search interventions receive a fresh response budget and preserve the remaining number of tool calls of the candidate state. This makes continuation budgets equal within a candidate pool.

Inference uses eight NVIDIA L20Y GPUs, PyTorch 2.7.1, vLLM 0.10.0, and Transformers 4.57.6. Saved batches are checked against their inputs and settings before reuse.

C Diagnostic Protocols

C.1 Question and state sampling

We normalize question text using Unicode normalization, case folding, and whitespace normalization, then remove exact matches with the training-question pool. A fixed, score-independent ordering determines question selection within each dataset. The first 40 per dataset form the common-state pool, and the first 15 of those form the candidate-query pool. The resulting pools contain 200 and 75 distinct questions. The overlap filter uses normalized exact matching.

The three trained search policies generate benchmark trajectories on these questions. We retain contexts immediately after a retrieved observation and before the next model response. Prefixes already containing a generated answer tag are excluded. A fixed rule assigns each question to one of the three source policies, independently of model outputs. We keep that source's question-only state and first observation, together with the last observed state from every source. Duplicate states are removed. The resulting common-state pool contains 991 states. This samples early and late evidence while avoiding three copies of the question-only input.

All readers receive the same selected states. Question-only, first-observation, and last-observation summaries are computed separately; observed-state averages first average within question. The

source-by-reader comparison conditions on the last state reached by each source. Each source contributes its own stopping context, including the number of observations it produced. The extended answering comparison also includes the initial model as an untrained reference. Auxiliary-only reads and continues from the same original state and candidate pools. The diagnostic input distribution is shared across all evaluated models.

C.2 Answering instructions and scoring

The *intermediate-answer* protocol uses the training-time answer-generator prompt, including its answer examples and <eval> tags. The *policy stop-and-answer* (native) protocol retains the native search-policy system instruction and observed assistant context, then appends the native tool-limit instruction. This permits reasoning followed by a final <answer> response, but further tools are disabled. These protocols preserve the same observed information while changing the model's assigned role and context serialization. We apply both protocols to every model in the answering diagnostics. Thus prompt labels describe answering instructions, while model labels identify the weights used. For Φ, Full and Aux-only use their shared policy weights; Separate uses its independent generator. Outcome-only and Initial use their policy weights under intermediate-answer instructions for evaluation. Shared weights can produce different scores under the two prompts; equality of the expectations requires the complete input, decoding, and scoring to match.

The primary content score extracts a unique answer in the required tag and evaluates it using the original token-F1 and EM scorer. Extra outer prose does not invalidate a unique answer. Responses with ambiguous answer tags or generated search/result blocks receive no content credit. Strict scoring additionally requires the prescribed outer format. For sensitivity analysis, a tag-agnostic extractor accepts either one <eval> answer or one <answer> answer, after removing complete reasoning blocks. It rejects tool tags, ambiguous answers, and untagged prose. It does not search the response for the reference answer. This isolates answer-tag choice from the other differences between the roles.

Native rollouts are scored from generated tokens selected by the response mask. Tool observations and inserted system warnings are removed before extracting the final answer. This avoids confusing the tool-limit instruction's answer-tag example with the model's output.

C.3 Changed answering instructions

The prompt control uses a fixed quarter of questions selected independently of model scores. On their selected observed states, it removes the intermediate prompt's answer examples and uses <answer> tags. Its paired comparisons use the original intermediate-answer outputs for those same states. This is a joint example-and-tag control; the tag-agnostic rescoring check separately tests tag choice without changing the input.

C.4 Candidate queries and feedback selection

For each candidate-pool question, we choose the earliest legal observed state with at least two remaining tool calls, preferring the assigned source policy and using an eligible alternative if needed. Each of the three search policies proposes two queries. Proposals must contain a single valid search block. They are normalized and deduplicated, and at most four are selected by a fixed rule independent of answer scores and continuation outcomes. All selected queries are executed, and each resulting candidate consumes one tool call.

Every reader generates four answers under each answering protocol for every candidate. Every receiver independently generates four continuations, divided into selection and confirmation folds. For a state s_t, the mean immediate-answer scores estimate $\Phi(s_t)$ or $B(s_t)$ according to the model and prompt. Continued-search value is estimated by

$$\widehat{V}_\theta(s_t) = \frac{1}{K}\sum_{k=1}^{K} R(a_{\text{final}}^{(k)}, y), \qquad K = 4. \tag{6}$$

Each $a^{(k)}_{\text{final}}$ comes from an independent continuation of the same saved state with its remaining tool budget. Repeating this evaluation at another state gives a separate estimate for that state. These values are diagnostic scores; training process rewards use the consecutive-state differences in Eq. (1).

A feedback provider's score is the mean immediate-answer F1. Its selection value is the receiver's mean continuation F1 across the candidates tied for the highest feedback score. Uniform selection averages over all candidates. For the continuation-based reference, one fold identifies the highest-valued candidates and the other evaluates their mean; the folds are then exchanged. Ties in this reference are also averaged. Parents with only one distinct valid query are omitted from query-selection comparisons.

Different queries can return the same passages. The external scorer uses reference answers; models receive only the question and search context.

The pool contains 218 candidate states from 75 questions. Nine have only one valid distinct query, leaving 66 for selection comparisons. Of 150 proposals per source policy, 150 are valid for Full, 149 for Outcome-only, and 131 for Separate. Invalid proposals are excluded before deduplication and scoring; no selected state is excluded for prompt length.

C.5 Synthetic search and answer acquisition

We construct 80 synthetic questions, separately from the natural QA subsets: 40 ask for the answer at the end of a two-record chain and 40 at the end of a three-record chain. A record is a short retrievable text entry containing either the identifier of the next record or the final answer string. Both identifiers and target strings are randomly generated, making success depend on retrieving the records rather than recalling factual knowledge. From a useful clue, the policy must retrieve the terminal entry and copy its stored string into the final answer. In the raw prompts and records, this string is called a `code`; it is an arbitrary identifier, not executable code.

For the saved task in Table 5, readable aliases give the following collection: A says "next: B," B says "next: C," and C says "answer: `K7M2Q9`." The question asks for the answer string reached from A. The search tool returns an entry when the query contains its identifier. Supplying only A→B leaves two searches: retrieving B reveals C, and retrieving C reveals `K7M2Q9`.

We evaluate three supplied-evidence conditions: a *useful clue* gives the first relevant record, a *distractor* gives an unrelated record, and a *direct answer* gives the complete relevant chain and its answer string. Across these three conditions, the 80 questions yield 240 starting contexts. Each condition occupies one initial observation. Within each question, continuation budgets are identical across conditions: one new search for two-record tasks and two for three-record tasks. Immediate-answer evaluation disables tools. Both evaluations score against the same target string, averaging exact match over four samples. Table 5 displays one question, Table 16 aggregates the conditions, and Table 4 compares acquisition and answering from the useful clue.

For each continuation, we record whether the terminal record was retrieved and whether the final answer matches its stored string. "Seen but incorrect" includes any incorrect response after acquisition, including copying errors; it measures a different failure from not finding the terminal record. Saved queries, observations, and responses identify these cases, with one copying error shown in Appendix E.2.

C.6 Aggregation and uncertainty

Repeated answers are averaged within state, and repeated states or candidates are averaged within question before reporting aggregate diagnostic statistics. The candidate experiment has one parent per question. Confidence intervals use 5,000 paired question-bootstrap draws with a fixed seed. Benchmark macro averages weight the five datasets equally; diagnostic question sampling is balanced across datasets. For the auxiliary-only benchmark contrasts, we resample paired normalized-question clusters within each dataset, preserve all repeated rows in a sampled cluster, and then average the five dataset means. The 26,133 examples contain 26,127 within-dataset question clusters: MuSiQue has 2,417 rows and 2,411 distinct normalized questions. Bamboogle contributes 125 examples and receives one fifth of the macro weight. Bootstrap intervals describe variation across questions, with paired predictions for each model.

Configuration	HotpotQA	2Wiki	NQ	MuSiQue	Bamboogle	Macro
Initial	25.15	13.32	15.54	10.26	32.00	19.25
Outcome-only	46.21	51.97	22.52	18.82	43.20	36.55
Aux-only	46.21	53.43	22.38	17.46	44.00	36.70
Separate	45.10	52.27	22.38	19.32	44.80	36.78
Terminal	47.13	54.59	22.33	19.40	47.20	38.13
Reasoning-masked	45.37	51.88	21.47	20.36	42.40	36.30
Full co-training	47.09	50.41	22.60	18.70	50.40	37.84

Table 8: Exact match for the primary benchmark comparison; the shaded row is Full co-training.

Stage, prompt, candidate-selection, and acquisition contrasts use the same paired outputs. Each model-by-prompt or provider-by-receiver interaction is formed within question before bootstrap resampling. The Full–Separate answering comparison resamples the same 200 questions under each protocol. Intervals for secondary contrasts are unadjusted.

D Additional Results

D.1 Benchmark scores and training duration

Table 8 complements the main F1 comparison with exact match for all primary configurations. These comparisons match the training update count at 2,418. Continuing Outcome-only training to update 5,035 raises its macro F1 from 49.45 to 51.17, providing a duration control within that configuration.

D.2 RQ1–RQ2: Answering across models and prompts

Figure 4 extends the Full–Separate comparison with Initial, Outcome-only, and Aux-only. Table 9 separates answer scores, format compliance, and answer production. A usable answer is nonempty and extractable without a tool violation. Conditional F1 divides the weighted answer score by the usable rate, with the same question–state–sample weights; each model's conditional score uses its own usable-output subset.

Under intermediate-answer instructions, usable rates exceed 99.8% for Outcome-only, Aux-only, and Full. Under policy stop-and-answer instructions, Aux-only improves answer production over Outcome-only, while its conditional F1 is lower. This distinguishes producing a valid response from improving content within each model's usable responses. Table 10 gives the parameter-sharing contrasts, and Table 11 checks them across source policies.

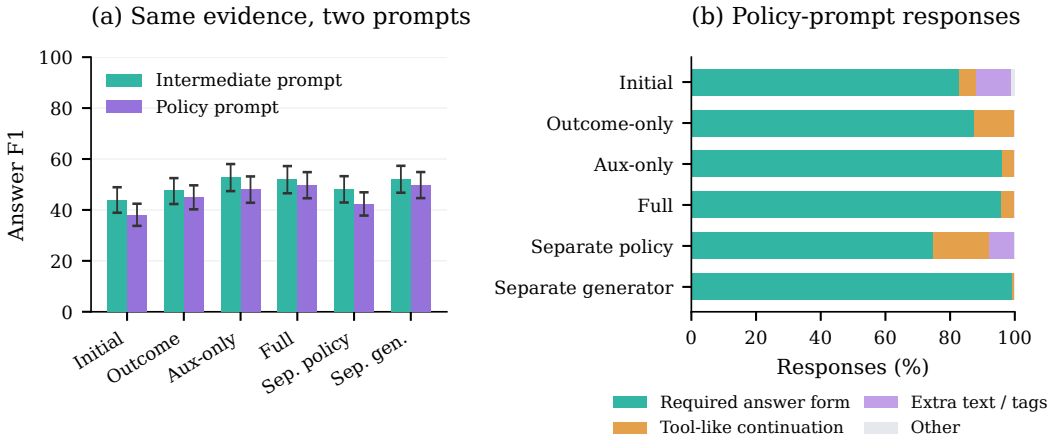

Figure 4: Extended answering comparison on identical contexts from 200 questions. Left: F1 under intermediate-answer and policy stop-and-answer instructions, with 95% question-bootstrap intervals. Right: response types under policy stop-and-answer instructions. Tools are disabled. Full and Separate address RQ1; Aux-only and Outcome-only address RQ2.

(a) Answer scores and output format

Model	Prompt	F1	Strict F1	Format (%)
Outcome-only	Intermediate	47.5	47.5	99.9
Outcome-only	Policy	45.1	45.1	87.6
Aux-only	Intermediate	52.9	52.9	99.9
Aux-only	Policy	48.1	48.1	96.2
Full	Intermediate	52.1	52.1	100.0
Full	Policy	49.7	49.7	95.8
Separate policy	Intermediate	48.2	48.2	100.0
Separate policy	Policy	42.4	39.3	74.7
Separate generator	Intermediate	52.2	52.2	100.0
Separate generator	Policy	49.8	49.8	99.2

(b) Answer production and conditional content

Model	Prompt	Total F1	Usable (%)	Conditional F1
Outcome-only	Intermediate	47.53	99.88	47.59
Outcome-only	Policy	45.09	87.63	51.46
Aux-only	Intermediate	52.93	99.94	52.97
Aux-only	Policy	48.09	96.25	49.96
Full	Intermediate	52.07	100.00	52.07
Full	Policy	49.73	95.75	51.94

Table 9: Answering on common evidence from 200 questions. Intermediate and Policy denote intermediate-answer and policy stop-and-answer instructions. Strict F1 additionally requires the prescribed outer format; accepting either answer tag leaves F1 unchanged for these rows. A usable answer is nonempty, extractable, and free of tool violations. Conditional F1 divides total F1 by the usable rate, using each model's own usable-output subset. All scores use the same question–state–sample weights.

Comparison	Intermediate prompt	Policy prompt
Full − Separate policy	3.9 [1.2, 6.7]	7.3 [4.9, 9.8]
Separate generator − Separate policy	4.0 [1.0, 7.0]	7.4 [5.2, 9.7]
Separate generator − Full	0.1 [-1.6, 1.8]	0.1 [-2.0, 2.2]

Table 10: Parameter-sharing comparisons on identical search contexts. Answer-F1 differences have paired 95% question-bootstrap intervals; positive values favor the first model.

Reader	Outcome contexts	Full contexts	Separate contexts
Full	53.4	55.1	56.5
Separate policy	50.4	51.0	50.7
Separate generator	53.1	55.8	55.9

Table 11: Intermediate-answer F1 on the last observed contexts produced by each source policy. Every reader receives the same contexts within a column; sources may stop after different numbers of observations.

D.3 RQ2: Training-signal benchmark contrasts

Table 12 reports paired F1 differences for every dataset and their equally weighted macro average. Aux-only exceeds Full on 2Wiki and falls below it on HotpotQA; the macro comparison incorporates this variation. The Aux-only–Outcome-only and Aux-only–Full macro intervals both include zero.

Dataset	Aux-only − Outcome-only	Aux-only − Full
HotpotQA	0.73 [-0.05, 1.50]	-1.12 [-1.85, -0.40]
2Wiki	1.04 [0.32, 1.77]	1.00 [0.32, 1.69]
NQ	0.32 [-0.67, 1.32]	0.15 [-0.78, 1.11]
MuSiQue	0.06 [-1.27, 1.35]	-0.57 [-1.89, 0.72]
Bamboogle	0.28 [-6.13, 6.35]	-2.24 [-8.82, 4.16]
Macro	0.48 [-0.85, 1.80]	-0.55 [-1.92, 0.75]

Table 12: Paired F1 differences with 95% question-cluster bootstrap intervals. The macro interval resamples within each dataset before equal-weight averaging.

D.4 RQ2: Early and late answering

Table 13 combines scores, paired differences, and usable-answer rates at three stages. Intermediate-answer training improves F1 under intermediate instructions at every stage. Under policy instructions, its first-observation gain is 11.82 [6.87, 16.86] points, accompanied by a usable-rate increase from 50.75% to 85.13%. At the last observation, both usable rates exceed 99.9% and the F1 difference is -0.12 [-2.62, 2.21]. The policy-prompt benefit is concentrated in early answering.

Context	Prompt	Outcome F1	Aux F1	Aux − Outcome	Usable (%) Outcome / Aux
Question only	I	23.45	28.75	5.30 [1.33, 9.26]	99.25 / 100.00
Question only	P	0.00	25.76	25.76 [20.81, 30.97]	0.25 / 90.00
First retrieval	I	38.80	45.20	6.40 [2.11, 10.94]	99.50 / 100.00
First retrieval	P	21.59	33.41	11.82 [6.87, 16.86]	50.75 / 85.13
Last retrieval	I	50.54	55.63	5.09 [2.39, 7.96]	100.00 / 99.92
Last retrieval	P	52.98	52.86	-0.12 [-2.62, 2.21]	99.92 / 100.00

Table 13: Early and late answering on 200 questions per stage. I: intermediate-answer instructions; P: policy stop-and-answer instructions. Differences have paired 95% question-bootstrap intervals. Usable rates measure answer production, separately from answer correctness. First and last contexts coincide for one-retrieval trajectories.

D.5 RQ2: Changed answering instructions

The prompt control jointly removes answer examples and changes the requested answer tag. Table 14 reports scores and paired model-by-prompt contrasts on the same 46 questions and 180 states. The Aux-only–Outcome-only gap grows from 4.52 [-0.96, 9.86] to 16.25 [8.47, 24.67] F1 points, a paired change of 11.73 [5.75, 18.60]. Full shows the same direction. Both configurations with shared intermediate-answer training maintain stronger answers under the changed instructions.

(a) Performance with changed instructions

Model	Changed-prompt F1	Format (%)	Change from original
Outcome-only	29.1	100.0	-13.1 [-19.5, -7.6]
Aux-only	45.3	100.0	-1.4 [-3.6, 0.6]
Full	46.4	100.0	-2.1 [-6.2, 1.5]

(b) Paired model contrasts

Contrast	Original prompt	Changed prompt	Change in gap
Aux-only − Outcome-only	4.52 [-0.96, 9.86]	16.25 [8.47, 24.67]	11.73 [5.75, 18.60]
Full − Outcome-only	6.38 [1.11, 12.03]	17.39 [9.24, 26.58]	11.01 [4.30, 19.08]

Table 14: Answering-instruction control on the same 46 questions (180 states). The changed prompt jointly removes the intermediate-answer examples and requests `<answer>` tags. Intervals are paired 95% question-bootstrap intervals. Panel (b) measures the model gap under each prompt and its paired change.

D.6 RQ4: Candidate selection for each receiving policy

Table 15 gives the selected F1 scores behind the paired gains in Table 6. All three criteria use the same candidate pools. Continued-search selection uses independent rollout folds for selection and evaluation; maximum-score ties are averaged.

Receiver	Uniform	Immediate-answer	Continued-search
Outcome-only	50.4	54.63	55.64
Aux-only	49.1	52.64	56.49
Full	46.3	49.48	52.52
Separate policy	51.2	53.92	57.30

Table 15: Selected continuation F1 on 66 questions. Immediate-answer selection uses Full as the feedback provider. Continued-search selection splits the receiver's four rollouts per candidate into two equal folds for independent selection and evaluation, then exchanges the folds. Maximum-score ties are averaged. Table 6 reports paired gains.

Feedback provider and receiving policy. Figure 3 reports the provider–receiver comparisons. Let $Q_{g,r}(q)$ be the selected continuation F1 for provider g and receiver r on question q. The interaction averages

$$[Q_{\text{SepGen,Full}} - Q_{\text{Full,Full}}] - [Q_{\text{SepGen,Outcome}} - Q_{\text{Full,Outcome}}]. \tag{7}$$

It is 3.72 [1.21, 6.88] F1 points across the same 66 questions. The Separate-generator–Full provider difference is -2.7 [-5.6, -0.4] with Outcome-only as receiver and $+1.0$ [-0.2, 2.7] with Full as receiver. The interaction measures how the relative selection benefit changes with the policy that uses the feedback.

D.7 RQ2 and RQ4: Following a useful clue to the answer

Table 16 reports all three supplied-evidence conditions. A useful clue identifies the next relevant record, a distractor points to an unrelated record, and the direct-answer condition supplies the relevant chain and its answer string. Immediate-answer scores are zero for both clues and distractors. Continued-search scores distinguish them through the receiving policy's ability to reach the requested answer.

For the three-record useful-clue task, Full's completion advantage over Aux-only is 9.38 [3.75, 16.88] percentage points (Table 4). Every correct saved continuation first retrieves the terminal record. The failure gap decomposes into 2.50 [0.00, 7.50] points before terminal-record acquisition and 6.88 [2.50, 11.88] points after acquisition. The larger contribution after acquisition locates most of the observed difference in producing the final answer once its evidence is available.

Policy	Records	Clue I	Clue V	Distr. I	Distr. V	Direct I	Direct V
Outcome-only	2	0.0	93.8	0.0	0.0	99.4	100.0
Outcome-only	3	0.0	87.5	0.0	0.0	100.0	98.1
Aux-only	2	0.0	97.5	0.0	0.0	100.0	96.2
Aux-only	3	0.0	90.6	0.0	0.0	100.0	95.6
Full	2	0.0	100.0	0.0	0.0	100.0	100.0
Full	3	0.0	100.0	0.0	0.0	100.0	100.0
Separate policy	2	0.0	95.6	0.0	0.0	100.0	100.0
Separate policy	3	0.0	90.6	0.0	0.0	100.0	96.9

Table 16: Synthetic tasks under useful-clue, distractor, and direct-answer conditions, with 40 questions per chain length and four samples per state and protocol. I: immediate-answer exact match under intermediate-answer instructions, without tools. V: final-answer exact match after continued search from the same supplied observation. Two- and three-record tasks permit one and two additional searches, respectively. All entries are percentages.

E Illustrative Model Traces

E.1 The same evidence: answering and training inputs

Question. Who did the original voice of the character played by mila kunis in the cleveland show?

Retrieved excerpts. "Originally voiced by Lacey Chabert during the first season, Meg has been voiced by Mila Kunis since season 2." The same result list describes Alex Borstein as "best known for voicing the role of Lois Griffin."

Model and role	Generated answer
Full policy	Lacey Chabert
Outcome-only policy	Lacey Chabert
Separate answer generator	Lacey Chabert
Separate policy	Alex Borstein

Figure 5: A saved example on common evidence. All policies use stop-and-answer instructions; the Separate generator uses intermediate-answer instructions. The Separate policy selects the distractor, while Full and the Separate generator identify the original voice. Both samples agree with each displayed answer.

Figure 5 shows a content disagreement on the same evidence. Both Full policy samples and both Separate-generator samples answer correctly; both Separate-policy samples select the distractor.

The training-input ablation on this context. Full retains prior reasoning in the answer model's training input; Reasoning-masked removes reasoning blocks outside retrieved passages. The question and passages in Figure 5 remain in both inputs, as do the search queries. For example:

> **Retained in Full; removed in Reasoning-masked:** `<think>` To answer this question, we need to identify the character played by Mila Kunis in the Cleveland Show and then determine who the original voice of that character was. `</think>`
>
> **Retained in both:** `<search>` original voice of Meg Griffin in the animated series Family Guy `</search>`

Both answer branches use intermediate-answer instructions and generate an `<eval>` answer without tools. Its external answer score trains the branch, and its change from the previous retrieval state supplies the process reward. Parameter sharing, reward coefficient, and placement match; each search policy retains its full context. Table 2 compares the resulting trained policies. In contrast, the immediate-answer versus continued-search comparison in Table 5 holds trained weights fixed and changes whether the model can search before producing the scored answer.

E.2 FOLLOWING CLUES AND RETURNING THE ANSWER

The question asks for the answer string reached from A. The hidden collection contains A pointing to B, B pointing to C, and C storing `K7M2Q9`. Only A pointing to B is supplied initially: searching B reveals the next clue, and searching C reveals the answer string. Record names and answer strings are readable aliases for the saved random identifiers. The raw record field `code` holds this string; it does not denote program code.

Policy	Queries and returned records	Final answer
Full	`B` → `Record B. next: C.;` `C` → `Record C. code: K7M2Q9.`	`K7M2Q9`
Outcome-only	`B` → `Record B. next: C.;` `C` → `Record C. code: K7M2Q9.`	`K7M2Q8`
Separate policy	`B code record` → `Record B. next: C.`	`B`

Table 17: Saved actions and answers for the same synthetic task as Table 5, starting from the useful clue. Each arrow maps a query to its returned record. Aliases distinguish the target answer string from an incorrect copy.

Full follows both clues and returns the target string. Separate retrieves B and returns its identifier, stopping before the answer. Outcome-only reaches C but copies its answer string incorrectly, shown as `K7M2Q8`. The case separates premature stopping from a copying error after retrieval; Table 4 measures acquisition and answer accuracy across all tasks.