PRIMESEEKER: CAPABILITY-ORIENTED SUPERVISION FOR DEEP SEARCH AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model search agents are often trained with synthetic questions whose difficulty is increased through larger evidence graphs, additional hops, and longer trajectories. These global properties, however, are only indirect proxies for the local retrieval capabilities required during search. To address this mismatch, we introduce latent anchor reasoning, which consists of resolving an unnamed retrieval anchor from descriptive specifications and transferring the recovered anchor into a subsequent information demand. This primitive retrieval unit decomposes deep search into chains of coupled operations and organizes question construction around anchor resolution and relation transfer, without prescribing a canonical search path. Based on this formulation, we propose PrimeSeeker, a capability-oriented framework that constructs web-grounded anchor structures and jointly derives a question and a reference evidence skeleton. The skeleton preserves supporting evidence from construction and guides expert generation through extractive highlights of current tool observations. These highlights are removed before supervised fine-tuning, while the skeleton is subsequently reused to audit reference-step coverage for reinforcement-learning rewards. We construct 9,221 expert trajectories, training a 30B search agent. Across five deep-search benchmarks, PrimeSeeker achieves strong performance, while reference-step optimization further improves the supervised policy. The resulting trajectories exhibit low retrieval redundancy, and fixed-budget evaluation shows strong solution coverage with substantially fewer tool calls than long-horizon systems. Data and code are available at https://anonymous.4open.science/r/PrimeSeeker-445D.

1 INTRODUCTION

Large language models (LLMs) are increasingly used as tool-augmented agents that interleave reasoning with external retrieval to solve knowledge-intensive tasks (Lewis et al., 2020; Yao et al., 2023). Recent systems acquire such behavior through iterative-retrieval supervision, outcome-based reinforcement learning, and multi-stage agentic training (Wang et al., 2025; Jin et al., 2025; Song et al., 2025; Team et al., 2026b). Although these approaches have substantially advanced long-horizon information seeking, they leave open what specific search capabilities a constructed question and its expert trajectory are intended to teach.

Most data construction pipelines model search capability through two indirect routes. The first increases the difficulty of the task by expanding the evidence graphs, obscuring entities, and dispersing the supporting evidence (Li et al., 2025; 2026; Du et al., 2026b; Chu et al., 2026). The second removes questions that can be solved through simple shortcuts or simplifies trajectories by pruning redundant evidence. (Deng et al., 2026; Wang et al., 2026). By improving task difficulty or trajectory, they refine the surface properties of training examples. Yet they fall short of specifying the search capability—a reusable behavioral operation invariant across different content instances. A complex question may be solved using parametric knowledge or a single discriminative query, while a short question may still require identifying an unnamed target. The same retrieval ability may also appear in questions of different complexity and be expressed through different search paths. Therefore, complexity and trajectory properties alone cannot specify the capability being supervised.

We address this mismatch by viewing deep search as a composition of local retrieval capabilities. In many web search questions, the next useful target is not explicit and is specified only through

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.

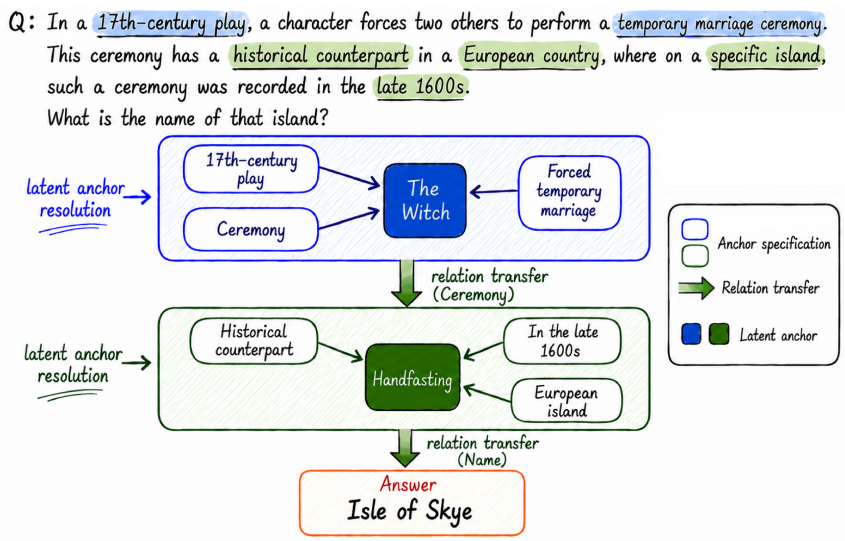


Figure 1: Illustration of latent anchor reasoning. Descriptive specifications first identify an unnamed play; the recovered anchor makes the ceremony relation queryable, leading to a second latent anchor and ultimately the name of the requested location. The decomposition specifies local retrieval capabilities without prescribing a unique search trajectory.

descriptive clues, which we call *anchor specifications*. The agent must first recover a usable retrieval anchor from these specifications and then use the recovered anchor to formulate a subsequent information demand. We refer to these coupled operations as *latent anchor resolution* and *relation transfer*, and to the overall capability as *latent anchor reasoning*. Figure 1 illustrates how multiple such local units compose a deep search problem without determining the model's realized search path. This decomposition defines the targeted retrieval capabilities without enforcing a canonical question or search path.

We propose PRIMESEEKER, a capability-oriented framework that constructs questions and expert trajectories around these local operations. PrimeSeeker builds web-grounded anchor structures by collecting a small number of independently sourced and validated specifications for each anchor and following substantive semantic relations. From the same construction structure, it derives both a question q and a reference evidence skeleton D_q. During expert generation, D_q highlights matching evidence only after it appears in the current tool observation, without introducing unseen information or determining the expert's next action. All skeleton-dependent guidance is removed before supervised fine-tuning, while the same skeleton remains external to the policy and can subsequently audit reference-step coverage during reinforcement learning.

Using this pipeline, we construct 9,221 expert trajectories and train PrimeSeeker from Qwen3-30B-A3B-Thinking-2507. Across two expert models, reference-aligned D_q guidance consistently improves expert accuracy, whereas random observation-derived highlights sharply degrade performance. The generated demonstrations exhibit lower duplicate-search and duplicate-visit rates than the compared search-agent corpora. Across five deep search benchmarks, PrimeSeeker-D_q consistently improves over PrimeSeeker-SFT by reusing the construction-derived skeleton for reference-step coverage. Under a controlled interaction budget, PrimeSeeker also matches or exceeds strong long-horizon systems while using substantially fewer tool calls. These results demonstrate the utility of reusing construction-derived evidence across expert generation and policy optimization.

Our contributions are summarized as follows:

- We reformulate deep search from a capability-oriented perspective and introduce latent anchor reasoning, characterizing each question through latent anchor resolution and relation transfer. This enables an operational basis for organizing search-task construction.
- We propose PrimeSeeker, which jointly derives a question and a reference evidence skeleton D_q from the same anchor structure. The skeleton guides expert generation and supplies

reference-step rewards, providing consistent supervision from task construction to policy optimization while preserving diverse valid search strategies.

- We evaluate PrimeSeeker across five deep search benchmarks. The resulting demonstrations exhibit low retrieval redundancy, D_q-guided generation consistently improves expert accuracy, reference-step optimization improves the supervised agent, and fixed-budget evaluation shows strong solution coverage with substantially lower tool consumption.

2 Problem Formulation

We study deep search tasks that require an agent to interleave reasoning with multi-turn retrieval. Given a question q, a retrieval environment Σ, and a gold answer $y^\star$, the agent must acquire external evidence and use it to support the final answer. Instead of treating the length or global structure of a complete trajectory as the basic unit of supervision, we focus on the local retrieval capabilities from which such trajectories are composed. We represent the task-relevant evidence structure behind q as

$$\mathcal{G}_q = (\mathcal{A}_q, \mathcal{C}_q, \mathcal{R}_q, \mathcal{E}_q, y^\star).$$

Here, $\mathcal{A}_q$ contains the task-relevant retrieval anchors. Anchors may be named entities, rules, attribute slots, or other intermediate objects that enable subsequent retrieval. $\mathcal{C}_q$ collects the descriptive specification sets $C(a_i)$ associated with these anchors, $\mathcal{R}_q$ contains the queryable relation demands over them, and $\mathcal{E}_q$ contains the external webpages supporting the specifications and relations.

For an anchor $a_i \in \mathcal{A}_q$, let

$$C(a_i) = \{c_{i1}, \ldots, c_{ik}\}$$

denote its specification set. Each c_{ij} describes a property, function, scenario, context, or partial relation of a_i, without necessarily exposing a directly searchable name. The specification therefore induces a candidate space

$$\mathrm{Cand}_\Sigma(C(a_i))$$

within the retrieval environment. We write the primitive local retrieval unit as

$$C(a_i) \Rightarrow a_i \xrightarrow{r_i} z_i.$$

The first operation, $C(a_i) \Rightarrow a_i$, is *latent anchor resolution*. The agent forms candidate anchors from the specifications, retrieves discriminative evidence, and identifies the corresponding explicit anchor. Resolution may be immediate when a strong prior or discriminative query sharply narrows the candidate space, or may require iterative comparison and verification. If the anchor is already *explicit*, $C(a_i)$ degenerates to $\{a_i\}$.

The second operation, $a_i \xrightarrow{r_i} z_i$, is *relation transfer*. Once a_i is available, the agent formulates an information demand conditioned on that anchor. The result takes one of three forms:

$$z_i \in \{y^\star,\ a_{i+1},\ C(a_{i+1})\}.$$

It may directly yield the final answer $y^\star$, expose the explicit anchor a_{i+1}, or produce a set of specifications $C(a_{i+1})$ that defines a new latent anchor. Semantic migration occurs when relation transfer moves the search to z_i, thereby composing multiple local retrieval units into a deep search task.

By specifying capabilities rather than execution routes, the formulation allows the same anchor to be recovered through prior knowledge, a discriminative query, or iterative evidence comparison. Supervision therefore targets identifiable instances of latent anchor resolution and relation transfer, together with their grounding in external evidence, rather than adherence to a fixed search path.

3 Method

PrimeSeeker operationalizes the capability-oriented formulation in Section 2 through capability-oriented QA construction and D_q-guided expert generation. The recovered trajectories are used for supervised policy training, while the evidence skeleton D_q can optionally be reused to audit reference-step coverage during reinforcement learning. Figure 2 presents the complete pipeline.

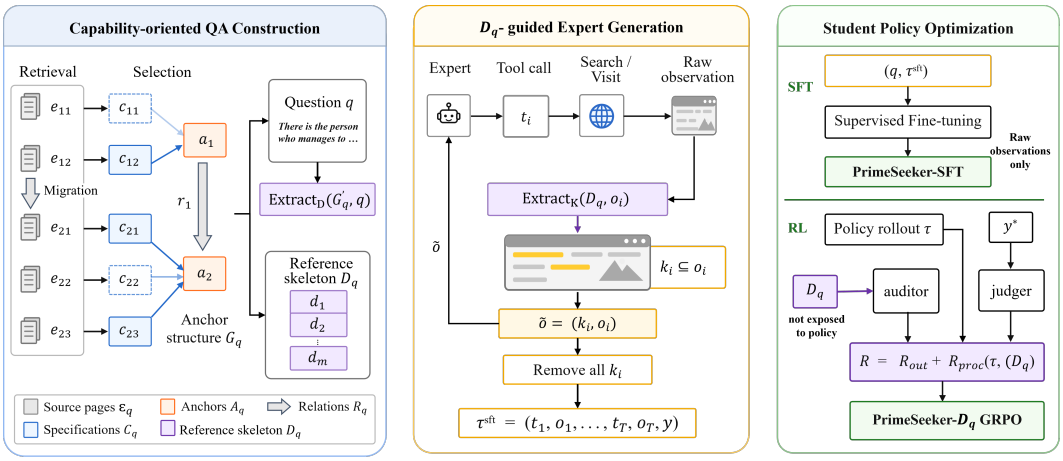


Figure 2: Overview of PRIMESEEKER. The constructed anchor structure yields both a question q and a reference evidence skeleton D_q. During expert generation, D_q extracts matching evidence from the current observation. These annotations are removed before SFT, while D_q remains external to the policy and later provides reference-step coverage signals during reinforcement learning.

3.1 CAPABILITY-ORIENTED QA CONSTRUCTION

PrimeSeeker constructs questions around explicit instances of latent anchor resolution and relation transfer, rather than increasing global question complexity and relying on the resulting trajectory to expose useful search behavior.

Constructing the anchor structure. Starting from a seed page or topic, the construction agent expands the current anchor a_i by retrieving independent webpages. It extracts one factual specification c_{ij} from each source page $e_{ij} \in \mathcal{E}_q$ and retains the corresponding provenance, so that every specification can be traced to supporting evidence.

A language model validator retains a specification only if it is supported by the source page, adds information not already covered by the collected specifications, and remains relevant to the relation chain. Through ambiguity checking validation, we ensure that the retained anchor is unique and the specifications are correct. We collect two or three specifications for each anchor from different webpages. The combined description is then checked for ambiguity and retained only when it is sufficiently discriminative to identify a single anchor in practice.

The construction agent then follows a substantive semantic relation

$$a_i \xrightarrow{r_i} a_{i+1},$$

such as a functional, historical, causal, compositional, or institutional relation. Iterative descriptive specification collection and semantic migration produce the construction record

$$\mathcal{G}'_q = (\mathcal{A}_q, \mathcal{C}_q, \mathcal{R}_q),$$

together with the source provenance associated with each specification.

Question and reference skeleton derivation. The question generator selects a connected substructure of $\mathcal{G}'_q$ and verbalizes its specifications and relation demands. For anchors intended to train latent anchor resolution, the question includes only their descriptive specifications, without revealing their identities. The selected relation demands are retained so that each recovered anchor can support the intended subsequent query. The resulting substructure is then rendered as a natural language question q with target answer $y^\star$.

From the same construction record, PrimeSeeker extracts a reference evidence skeleton

$$D_q = \text{Extract}_D(q, \mathcal{G}'_q) = \{d_1, \ldots, d_m\}.$$

Each reference step d_i links a supporting fact to the anchor or relation established by that fact. Together, the steps describe the evidence milestones that a successful search is expected to establish,

progressing from intermediate anchors and relations to evidence for the target answer. They specify what evidence is relevant without prescribing the queries, actions, or order used to retrieve it. A reference step is treated as answer evidence when the anchor or relation it verifies directly establishes $y^\star$ and the remaining steps capture intermediate evidence progress.

Quality control. We remove candidates that the designated expert answers correctly without external tools under the closed-book filtering configuration. This excludes questions already solved by that expert from parametric knowledge alone.

3.2 D_q-GUIDED EXPERT GENERATION

During expert generation, PrimeSeeker uses D_q to identify when a retrieved observation supports an anchor or relation specified during construction and temporarily highlights relevant evidence only when it appears in the current observation, while excluding spans that directly reveal the target answer. The expert then continues its own reasoning and tool selection; all auxiliary guidance is removed before SFT, ensuring the student model is trained solely on the original interaction history.

Evidence extraction and guidance. At turn i, the expert issues a tool call t_i and receives a raw observation o_i. PrimeSeeker extracts evidence from the current observation that is relevant to the question and its reference steps:

$$k_i = \mathrm{Extract}_k(q, o_i, D_q), \qquad \tilde{o}_i = (o_i, k_i).$$

The guidance is strictly extractive:

$$k_i \subseteq o_i.$$

Every highlighted span must occur verbatim in the current raw observation. If no relevant evidence is found, k_i is empty and o_i is left unchanged. Spans that directly reveal the target answer $y^\star$ are excluded, and the reference skeleton itself is never inserted into the observation. The expert remains responsible for forming hypotheses, selecting tools, and choosing the search path, without any objective on trajectory length or repetition. The guidance therefore directs attention to relevant evidence without constraining how the answer is reached.

Guidance removal and consistency filtering. The expert is instructed not to reference guidance markers or hidden annotations in its reasoning. After generation, all D_q-dependent annotations and extracted highlights are removed, yielding

$$\tau^{\mathrm{sft}} = (t_1, o_1, \ldots, t_T, o_T, \hat{y}_\tau),$$

where free-form assistant reasoning is omitted from the notation.

A prompt-based verifier then checks whether the retained reasoning and tool calls are supportable from q and the raw interaction history. Trajectories that explicitly depend on hidden annotations or contain actions unsupported by the available history are discarded. Removing the guidance ensures that no additional factual content appears in the student input, although the expert policy may still benefit from evidence highlighting during synthesis.

3.3 STUDENT POLICY OPTIMIZATION

Supervised fine-tuning. We train the student on the recovered trajectories using standard next-token prediction over assistant reasoning, tool calls, and final answers. Raw tool observations remain in the context but are masked from the loss. The student therefore receives neither the reference evidence skeleton D_q nor the extracted highlights k_i.

Reference-step coverage reward. A reward based only on final correctness cannot distinguish an unsuccessful rollout that has recovered useful evidence from one that has made no relevant progress. We therefore use D_q during reward computation to measure how much of the reference evidence skeleton is established by the rollout, while keeping D_q hidden from the policy.

Given a sampled trajectory τ, an auditor agent evaluates whether each reference step is covered:

$$A(\tau, d_i) \in \{0, 1\}.$$

The auditor sets $A(\tau, d_i) = 1$ only when the public interaction history contains evidence that establishes the fact, anchor, or relation described by d_i. Tool observations provide the primary evidence. Assistant reasoning is accepted only when it accurately cites or paraphrases evidence retrieved earlier in the trajectory. Search queries, topical overlap, unsupported model claims, and the final answer alone do not establish coverage, and each reference step contributes at most once.

For reward computation, we separate steps that directly verify the gold answer from those that establish intermediate anchors or relations:

$$D_q^{\text{answer}} = \{d_i \in D_q : d_i \text{ directly verifies } y^\star\}, \qquad D_q^{\text{chain}} = D_q \setminus D_q^{\text{answer}}.$$

For process reward, let

$$n_{\text{chain}}(\tau) = \sum_{d_i \in D_q^{\text{chain}}} A(\tau, d_i), \qquad n_{\text{answer}}(\tau) = \sum_{d_i \in D_q^{\text{answer}}} A(\tau, d_i)$$

denote the numbers of covered intermediate and answer-supporting reference steps, respectively. We define the process reward as

$$R_{\text{proc}}(\tau; D_q) = \min\left\{\alpha n_{\text{chain}}(\tau) + \beta\, \mathbb{I}[\text{correct}(\tau)] n_{\text{answer}}(\tau), C\right\}.$$

Intermediate coverage receives partial credit regardless of the final outcome, whereas answer-supporting coverage contributes only when the rollout is correct. The constant C bounds the total process contribution. Numerical settings are provided in Appendix C.

The full reward used by PrimeSeeker-D_q is

$$R(\tau; D_q) = R_{\text{out}}(\tau) + R_{\text{proc}}(\tau; D_q),$$

where R_{out} measures final-answer correctness. PrimeSeeker-Outcome uses R_{out} alone, isolating the effect of reference-step coverage. Rewards are computed independently for each rollout and normalized within each sampled group following standard GRPO. The auditor prompt and remaining optimization details are provided in Appendix C.

4 Experiments

We evaluate PrimeSeeker through benchmark performance, robustness under varying inference budgets, efficiency under a common interaction budget, and the generated expert trajectories.

4.1 Experimental Setup

Benchmarks. We evaluate on BrowseComp-EN and BrowseComp-ZH (Wei et al., 2025; Zhou et al., 2025), DeepSearch-2505 and DeepSearch-2510 from xbench-DeepSearch (Chen et al., 2025), and GAIA (Mialon et al., 2024). BrowseComp evaluates target identification from sparse indirect clues, xbench-DeepSearch emphasizes multi-step information seeking, and GAIA covers broader retrieval and agentic reasoning. Due to evaluation cost, we use fixed 200-example subsets of BrowseComp-EN and BrowseComp-ZH for all PrimeSeeker variants.

Baselines. Table 1 includes advanced tool-augmented models OpenAI o3, GPT-5, DeepSeek-V3.1, DeepSeek-V3.2, and GLM-4.7 (OpenAI, 2025b;a; DeepSeek-AI et al., 2025a;b; Z.ai, 2025). We also compare with selected open-source search agents of at least 30B parameters, including Tongyi DeepResearch (Team et al., 2026b), DeepMiner (Tang et al., 2025), E-GRPO (Zhao et al., 2026), WebSailor-V2 (Li et al., 2026), and OpenSeeker (Du et al., 2026b;a). MiroThinker is evaluated separately under a shared interaction budget in Section 4.3.

Implementation details. All PrimeSeeker variants use Qwen3-30B-A3B-Thinking-2507 (Yang et al., 2025) and the OpenSeeker live-web environment, with Serper (Serper, 2025) for search and Jina Reader (Jina AI, 2025) for page retrieval. **SFT.** We train for two epochs on 9,221 PrimeSeeker trajectories and 902 correct Chinese OpenSeeker trajectories. Tool observations are masked from the loss, and all D_q-dependent guidance is removed before training. **RL.** PrimeSeeker-Outcome and PrimeSeeker-D_q start from the same SFT checkpoint and use the same questions and four rollouts per prompt, differing only in whether reference-step coverage is included in the reward. Each inference run is limited to 100 tool calls. Further details are provided in Appendix C.

Table 1: Reported performance on five deep-search benchmarks. Underlined values are the best among reported models in each block; bold values are the best among PrimeSeeker variants.

Model	BS-EN		BS-ZH		DS2505		DS2510		GAIA	
	P@1	P@3	P@1	P@3	P@1	P@3	P@1	P@3	P@1	P@3
Advanced tool-augmented models										
OpenAI o3	49.7	–	58.1	–	67.0	–	–	–	70.5	–
DeepSeek-V3.1	30.0	–	49.2	–	71.0	–	–	–	63.1	–
GPT-5	54.9	–	63.0	–	77.8	–	75.0	–	76.4	–
DeepSeek-V3.2	67.6	–	65.0	–	78.0	–	55.7	–	75.1	–
GLM-4.7	67.5	–	66.6	–	72.0	–	52.3	–	61.9	–
Open-source search agents ($\geq$ 30B)										
Tongyi DeepResearch	43.4	59.6	46.7	63.7	75.0	86.0	47.5	–	70.9	85.5
DeepMiner-32B-RL	33.5	–	40.0	–	62.0	–	–	–	58.7	–
E-GRPO-30B	12.9	21.0	26.4	41.2	46.7	66.0	–	–	48.5	65.1
WebSailor-V2-30B	35.3	–	44.1	–	73.7	–	–	–	74.1	–
OpenSeeker	29.5	–	48.4	–	74.0	–	–	–	–	–
OpenSeeker-V2	46.0	–	58.1	–	78.0	–	–	–	–	–
PrimeSeeker-SFT	34.3	51.5	40.8	62.0	71.6	88.0	40.3	60.0	58.3	76.7
+ Outcome Reward	32.8	50.5	**47.5**	**63.5**	74.0	84.0	35.8	56.0	56.3	69.9
+ D_q Reward	**39.2**	**53.0**	42.3	62.0	**76.0**	**90.0**	**41.0**	**64.0**	**62.1**	**84.5**

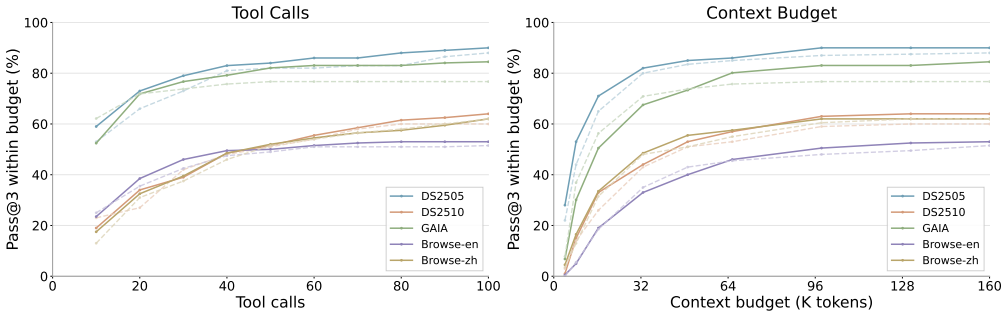


Figure 3: Pass@3 across tool call and context budgets. Solid lines denote PrimeSeeker-D_q, and pale dashed lines denote PrimeSeeker-SFT. A question is solved if correct within the cutoff.

Evaluation metrics. For each question, we sample three independent runs. Pass@1 is the average accuracy and Pass@3 is oracle solution coverage. Answers enclosed by `<answer>` tags are evaluated against the reference answer under an LLM-as-a-judge protocol.

4.2 Main Benchmark Results

Table 1 reports results on five benchmarks. The first two blocks reproduce external reports under their respective evaluation protocols. The final three rows report PrimeSeeker variants, which achieve competitive performance among open-source search agents. Relative to PrimeSeeker-SFT, PrimeSeeker-D_q improves Pass@1 on all five benchmarks and Pass@3 on four, with Pass@3 unchanged on BS-ZH. Compared with outcome-only RL, it improves both metrics on four benchmarks but underperforms on BS-ZH.

We retrospectively compute Pass@3 at tool-call and context-token cutoffs from recorded runs of PrimeSeeker-SFT and PrimeSeeker-D_q. Figure 3 shows that PrimeSeeker-D_q generally achieves higher solution coverage at matched cutoffs, with most curves flattening around 60–80 calls and 64–96K tokens. These results show that its gains in solution coverage are already evident below the maximum resource allowance.

Table 2: Performance under a shared 300-call budget on DS2505 and DS2510. Pass@1 values for OpenSeeker-v1 and MiroThinker are from their original reports; all remaining results are obtained by rerunning each system with its original inference procedure.

Model	DS2505				DS2510			
	Standard P@1	Budgeted Solve Rate	Consumption Avg. Calls	Correct Avg. Calls	Standard P@1	Budgeted Solve Rate	Consumption Avg. Calls	Correct Avg. Calls
OpenSeeker-v1	74.0	83.0	34.5	**16.2**	–	57.0	109.5	46.4
MiroThinker	**77.2**	85.0	99.9	70.4	**57.2**	58.0	144.2	72.6
PrimeSeeker-D_q	76.0	**91.0**	**21.8**	16.3	41.0	**68.0**	**68.6**	**33.5**

Table 3: Aggregate statistics of expert search trajectories.

Dataset	Samples	Avg. Tools	Avg. Tokens	Search Dup.	Visit Dup.
DeepResearch-9K	12,974	25.0	12.3k	3.1%	13.1%
OpenSeeker	11,677	47.0	73.9k	1.6%	10.3%
REDSearcher	10,001	65.1	56.4k	0.2%	14.5%
PrimeSeeker	9,221	19.5	37.8k	0.1%	8.0%

4.3 Search Efficiency under a Shared Interaction Budget

We further compare PrimeSeeker with OpenSeeker-v1 and MiroThinker (Team et al., 2026a), rerunning all three systems on 100 questions from DS2505 and DS2510 under a shared cumulative budget of 300 tool calls per question. All calls across context continuations and up to five retries count toward this budget. Evaluation stops at the first answer judged correct against the reference, or when the budget or retry limit is reached. Budgeted Solve Rate measures the fraction of questions solved. Consumption Avg. Calls averages cumulative calls over all questions; Correct Avg. Calls averages calls in successful attempts only, excluding earlier failures.

As shown in Table 2, under the same interaction allowance, PrimeSeeker solves more questions on both datasets while consuming substantially fewer tool calls. This advantage is particularly pronounced on DS2510, where extended exploration becomes expensive. By focusing supervision on capability orientation rather than indirectly proxying difficulty through trajectory length, PrimeSeeker maximizes the utility of the fixed inference budget, achieving strong coverage without relying on prolonged interaction. This demonstrates that the strong deep search performance need not depend on exhaustive interaction.

4.4 Analysis of Constructed Expert Trajectories

Table 3 compares corpus size, trajectory length, tool usage, and exact within-trajectory repetition. PrimeSeeker has the lowest average tool-call count and the lowest duplicate-query and duplicate-visit rates among the compared corpora. Its average token length remains above DeepResearch-9K but well below OpenSeeker and REDSearcher. These statistics characterize the resulting demonstrations as relatively compact in tool use and low in exact retrieval repetition. Full distributional diagnostics are provided in Appendix E.

Complementing these corpus-level statistics, a controlled expert-generation analysis in Appendix D shows that reference-aligned D_q highlights consistently improve accuracy across the evaluated teacher settings. Specifically, Qwen3 makes more calls, whereas DeepSeek-V4 makes fewer. This pattern is consistent with more effective evidence use rather than uniform search compression. Guidance may encourage further evidence gathering in shorter searches, while helping focus more extended exploration. In contrast, random highlights substantially degrade accuracy and increase tool use, showing that arbitrary observation-derived cues do not reproduce the gains.

5 Related Work

Deep-search agents. Retrieval-augmented generation established external evidence as a complement to parametric knowledge, while later methods made retrieval adaptive to the model's evolving reasoning state (Lewis et al., 2020; Trivedi et al., 2023; Asai et al., 2024; Wang et al., 2025). Re-

Act formulated tool use as an iterative reasoning–action–observation process and became the basis of many web-search agents (Yao et al., 2023). Search-R1 and R1-Searcher demonstrated that this behavior can be acquired through outcome-supervised reinforcement learning, while recent Deep-Research systems combine supervised trajectories, policy optimization, and extended inference horizons (Jin et al., 2025; Song et al., 2025; Team et al., 2026b;a; Chu et al., 2026). These works mainly improve agent architectures, post-training, and inference-time search. PrimeSeeker instead studies the supervision encoded by constructed search problems, focusing on how an indirectly specified target is recovered as a retrieval anchor and enables the next information demand.

Synthetic search-task construction. Classical multi-hop QA constructs questions around linked supporting facts or composed single-hop questions (Yang et al., 2018; Trivedi et al., 2022). Recent search-agent pipelines create harder, implicit information needs through web-grounded graph expansion, relational subgraph sampling, entity obfuscation, and dispersed evidence (Li et al., 2025; 2026; Du et al., 2026b; Team et al., 2026a; Chu et al., 2026). WebSailor-V2 broadens structural and uncertainty coverage through densely connected graphs, topology-aware sampling, and information obfuscation (Li et al., 2026). OpenSeeker and related easy-to-hard pipelines iteratively expand seed questions or local web structures into more challenging search problems (Du et al., 2026b).

These approaches substantially improve the scale, diversity, and difficulty of synthetic search data. However, graph size, hop count, and realized trajectory length do not uniquely determine the retrieval capability exercised by a question. Selective clues, co-located evidence, or parametric knowledge may collapse an apparently complex problem, motivating shortcut detection and empirical difficulty filtering (Deng et al., 2026; Song et al., 2026). PrimeSeeker adopts a complementary granularity by organizing construction around the local unit $C(a_i) \Rightarrow a_i \xrightarrow{r_i} z_i$. This unit couples the recovery of a descriptively specified target with the relation demand enabled by that target, while allowing different agents to realize the capability through different search paths.

Expert supervision and process rewards. Search-agent SFT commonly relies on successful trajectories sampled from stronger teachers (Wang et al., 2025; Li et al., 2025; Chu et al., 2026). Subsequent methods improve these demonstrations during generation or through post-hoc processing. OpenSeeker summarizes earlier observations to assist teacher generation and restores the original observations before student training, whereas WebClipper prunes redundant branches and rewrites the retained reasoning (Du et al., 2026b; Wang et al., 2026). Verification-based systems also use external feedback to revise actions or final answers (Team et al., 2026a; Zhu et al., 2026).

A related line of work reuses information from task construction during policy optimization. E-GRPO rewards matches to retained ground-truth entities, while Search-P1 rewards consistency with reference plans derived from successful solutions (Zhao et al., 2026; Xia et al., 2026). PrimeSeeker instead derives the reference evidence skeleton D_q jointly with each question and uses it for expert guidance and reference rewards, with all auxiliary annotations removed before SFT.

6 Conclusion

We introduced latent anchor reasoning as a capability-oriented formulation of deep search, decomposing local retrieval into latent-anchor resolution and relation transfer. PrimeSeeker operationalizes this formulation through web-grounded anchor structures from which questions and reference evidence skeletons D_q are jointly derived. During expert generation, D_q provides strictly extractive, observation-grounded highlights without exposing unseen evidence or prescribing a canonical search path. These annotations are removed before SFT, while the same construction-time structure is later reused to audit reference-step coverage during reinforcement learning. PrimeSeeker therefore connects task construction, expert synthesis, and policy optimization through a shared representation of the retrieval capabilities required by each question.

Using 9,221 generated trajectories, PrimeSeeker achieves strong performance across five deep-search benchmarks. Its demonstrations exhibit low retrieval redundancy, reference-aligned guidance improves expert generation while random highlights substantially degrade it, and fixed-budget evaluation shows that strong solution coverage need not rely on extended search trajectories. These results support explicit local retrieval capabilities and reusable construction-time structure as a more direct basis for search-agent supervision.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, Chen Sun, Han Hou, Hui Yang, James Pan, Jianan Lou, Jiayi Mao, Jizheng Liu, Jinpeng Li, Kangyi Liu, Kenkun Liu, Rui Wang, Run Li, Tong Niu, Wenlong Zhang, Wenqi Yan, Xuanzheng Wang, Yuchen Zhang, Yi-Hsin Hung, Yuan Jiang, Zexuan Liu, Zihan Yin, Zijian Ma, and Zhiwen Mo. xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations, 2025. URL https://arxiv.org/abs/2506.13651.

Zheng Chu, Xiao Wang, Jack Hong, Huiming Fan, Yuqi Huang, Yue Yang, Guohai Xu, Chenxiao Zhao, Cheng Xiang, Shengchao Hu, Dongdong Kuang, Ming Liu, Bing Qin, and Xing Yu. Redsearcher: A scalable and cost-efficient framework for long-horizon search agents, 2026. URL https://arxiv.org/abs/2602.14234.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, et al. Deepseek-v3 technical report, 2025a. URL https://arxiv.org/abs/2412.19437.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, et al. Deepseek-v3.2: Pushing the frontier of open large language models, 2025b. URL https://arxiv.org/abs/2512.02556.

Jia Deng, Yimeng Chen, Xiaoqing Xiang, Ziyang Zeng, Shuo Tang, Wayne Xin Zhao, Feng Chang, Chuan Hao, Yuan Wei, Ran Tao, Bryan Dai, and Ji-Rong Wen. Fort-searcher: Synthesizing shortcut-resistant search tasks for training deep search agents, 2026. URL https://arxiv.org/abs/2606.12087.

Yuwen Du, Rui Ye, Shuo Tang, Keduan Huang, Xinyu Zhu, Yuzhu Cai, and Siheng Chen. Openseeker-v2: Pushing the limits of search agents with informative and high-difficulty trajectories, 2026a. URL https://arxiv.org/abs/2605.04036.

Yuwen Du, Rui Ye, Shuo Tang, Xinyu Zhu, Yijun Lu, Yuzhu Cai, and Siheng Chen. Openseeker: Democratizing frontier search agents by fully open-sourcing training data, 2026b. URL https://arxiv.org/abs/2603.15594.

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*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Jina AI. Jina ai, 2025. URL https://jina.ai/.

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 (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, et al. Websailor: Navigating super-human reasoning for web agent, 2025. URL https://arxiv.org/abs/2507.02592.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Rui Ye, Yida Zhao, Liwen Zhang, Litu Ou, Ding-Chu Zhang, Xixi Wu, Xinmiao Yu, Jialong Wu, Xinyu Wang, Zile Qiao, Zhen Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Zhi-Qin John Xu, Shuai Wang, Minhao Cheng, and Jingren Zhou.

Websailor-v2: Bridging the chasm to proprietary agents via synthetic data and scalable reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=HuP16O5SJf.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

OpenAI. GPT-5 system card, 2025a. URL https://openai.com/index/gpt-5-system-card/.

OpenAI. Openai o3 and o4-mini system card, 2025b. URL https://openai.com/index/o3-o4-mini-system-card/.

Serper. Serper: Google search api, 2025.

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, 2025. URL https://arxiv.org/abs/2503.05592.

Maojia Song, Liu Renhang, Xinyu Wang, Yong Jiang, Pengjun Xie, Fei Huang, Soujanya Poria, and Jingren Zhou. Demystifying deep search: A holistic evaluation with hint-free multi-hop questions and factorised metrics. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=x4zQDewgHr.

Qiaoyu Tang, Hao Xiang, Le Yu, Bowen Yu, Yaojie Lu, Xianpei Han, Le Sun, WenJuan Zhang, Pengbo Wang, Shixuan Liu, Zhenru Zhang, Jianhong Tu, Hongyu Lin, and Junyang Lin. Beyond turn limits: Training deep search agents with dynamic context window, 2025. URL https://arxiv.org/abs/2510.08276.

MiroMind Team, S. Bai, L. Bing, L. Lei, R. Li, X. Li, X. Lin, E. Min, L. Su, B. Wang, L. Wang, L. Wang, S. Wang, X. Wang, Y. Zhang, Z. Zhang, G. Chen, L. Chen, Z. Cheng, Y. Deng, Z. Huang, D. Ng, J. Ni, Q. Ren, X. Tang, B. L. Wang, H. Wang, N. Wang, C. Wei, Q. Wu, J. Xia, Y. Xiao, H. Xu, X. Xu, C. Xue, Z. Yang, Z. Yang, F. Ye, H. Ye, J. Yu, C. Zhang, W. Zhang, H. Zhao, and P. Zhu. Mirothinker-1.7 h1: Towards heavy-duty research agents via verification, 2026a. URL https://arxiv.org/abs/2603.15726.

Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, et al. Tongyi deepresearch technical report, 2026b. URL https://arxiv.org/abs/2510.24701.

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. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

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 (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.557. URL https://aclanthology.org/2023.acl-long.557/.

Junjie Wang, Zequn Xie, Dan Yang, Jie Feng, Yue Shen, Duolin Sun, Meixiu Long, Yihan Jiao, Zhehao Tan, Jian Wang, Peng Wei, and Jinjie Gu. WebClipper: Efficient evolution of web agents with graph-based trajectory pruning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21646–21666, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.988. URL https://aclanthology.org/2026.acl-long.988/.

Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. Chain-of-retrieval augmented generation. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 59888–59915. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/566aad4fac1acf17fd1ae8c3aef75326-Paper-Conference.pdf.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents, 2025. URL https://arxiv.org/abs/2504.12516.

Tianle Xia, Ming Xu, Lingxiang Hu, Yiding Sun, Wenwei Li, Linfang Shang, Liqun Liu, Peng Shu, Huan Yu, and Jie Jiang. Search-p1: Path-centric reward shaping for stable and efficient agentic RAG training. In Yunyao Li, Georg Rehm, and Mei Tu (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 517–534, San Diego, California, USA, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-394-4. doi: 10.18653/v1/2026.acl-industry.36. URL https://aclanthology.org/2026.acl-industry.36/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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 (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

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. URL https://openreview.net/forum?id=WE_vluYUL-X.

Z.ai. GLM-4.7: Advancing the coding capability, 2025. URL https://z.ai/blog/glm-4.7.

Yida Zhao, Kuan Li, Xixi Wu, Liwen Zhang, Ding-Chu Zhang, Baixuan Li, Maojia Song, Zhuo Chen, Chenxi Wang, Xinyu Wang, Kewei Tu, Pengjun Xie, Fei Huang, Jingren Zhou, and Yong Jiang. Repurposing synthetic data for fine-grained search agent supervision. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=CByVWPpb8T.

Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, Yuxin Gu, Sixin Hong, Jing Ren, Jian Chen, Chao Liu, and Yining Hua. Browsecomp-zh: Benchmarking web browsing ability of large language models in chinese, 2025. URL https://arxiv.org/abs/2504.19314.

Bin Zhu, Qianghuai Jia, Tian Lan, Junyang Ren, Feng Gu, Feihu Jiang, Longyue Wang, Zhao Xu, and Weihua Luo. Marco deepresearch: Unlocking efficient deep research agents via verification-centric design, 2026. URL https://arxiv.org/abs/2603.28376.

A Appendix Overview

This appendix provides implementation details, controlled analyses, qualitative examples, and reproducibility information omitted from the main paper. Section B contains the AI-use statement, ethics statement, and reproducibility statement. Section C details the training, inference, tool environment, and auxiliary-model configurations. Section D presents a controlled study of unguided, random-highlight, and D_q-guided expert generation. Section E analyzes the trajectory-length, tool-use, and retrieval-redundancy distributions of the constructed expert trajectories. Section F provides case studies illustrating how web-grounded anchor structures give rise to aligned questions q and reference evidence skeletons D_q. Section G provides the prompt templates for observation-grounded evidence extraction and reference-step coverage auditing. Section H presents a representative test-time rollout from the trained PrimeSeeker policy. Section I summarizes the released artifacts and discusses limitations.

B Statements

B.1 AI use statement

Large language models were used as components of the research pipeline for web-grounded question–answer construction and validation, reference evidence skeleton generation, expert trajectory synthesis, evidence-highlight extraction, trajectory consistency checking, reference-step coverage auditing for reinforcement-learning rewards, and answer evaluation. These uses are described in the method and experimental sections and the appendices.

We also used generative AI tools for writing polishing and grammar checking, code generation and completion, prompt formatting modification, and translation. We used generative AI tools for the final automated testing pipeline of fixed scripts in our experiments. Additionally, we used these tools for quickly listing parameter matrices for automated running scripts during experiment execution.

We have reviewed and revised the AI-assisted manuscript content and take full responsibility for the research design, implementation, reported results, and final manuscript, including content and artifacts produced with the assistance of generative AI.

B.2 Ethics statement

We declare that this work does not raise any ethical concerns. Specifically, this work does not involve human subjects, dataset releases with privacy concerns, bias/fairness/discrimination issues, potentially harmful insights or applications, conflicts of interest or sponsorship, legal compliance issues, or research integrity concerns. The work is purely algorithmic and experimental in nature, focusing on the development and evaluation of a novel model/algorithm. All datasets used are either publicly available or properly anonymized.

B.3 Reproducibility statement

To ensure reproducibility, we have made the following efforts: (1) For the evaluation procedure, anonymous source code is provided as supplementary material. (2) The novel model and data used for fine-tuning and reinforcement learning are publicly available on HuggingFace, and a complete description of the data processing steps is provided in the paper. (3) Hyperparameters and experimental settings are clearly listed in the main paper and appendix. (4) Detailed instructions for running the experiments are included in the README file of the supplementary code package. We believe these materials are sufficient for researchers to reproduce the results presented in this paper.

C Training and Deployment Details

Supervised fine-tuning. We initialize PrimeSeeker from `Qwen3-30B-A3B-Thinking-2507`. The training corpus contains 9,221 PrimeSeeker expert trajectories and 902 correct Chinese trajectories from OpenSeeker (Du et al., 2026b), yielding 9,589 training instances and 534 validation instances. Tool observations are retained as context but masked from the language-modeling loss.

Table 4: Comparison of random and D_q-guided highlights against unguided expert generation.

Expert	K	Unguided		Random Highlight				D_q-guided			
		Acc.	Calls	Acc.	ΔAcc.	Calls	ΔCalls	Acc.	ΔAcc.	Calls	ΔCalls
Qwen3	2	52.0	6.45	28.0	-24.0	70.52	+64.07	54.0	+2.0	9.09	+2.64
Qwen3	3	57.0	7.01	34.0	-23.0	72.10	+65.09	61.0	+4.0	9.87	+2.86
DeepSeek-V4	2	51.0	24.67	24.0	-27.0	50.12	+25.45	56.0	+5.0	21.99	-2.68
DeepSeek-V4	3	59.0	28.68	24.0	-35.0	57.18	+28.50	66.0	+7.0	24.59	-4.09

We train for two epochs with a global batch size of 64, a maximum sequence length of 196,608 tokens, and a peak learning rate of 1×10^{-5} followed by cosine decay. The resulting checkpoint initializes both reinforcement-learning variants.

Reinforcement learning. PrimeSeeker-Outcome and PrimeSeeker-D_q are initialized from the same SFT checkpoint and trained on the same set of question–answer pairs generated using the QA construction procedure described in Section 3.1. Both variants are trained with GRPO for 64 steps. Each step uses a batch of 16 questions, with four rollouts sampled per question to form a GRPO group, yielding 64 rollouts per step. We use an actor learning rate of 2×10^{-6}, a PPO clipping ratio of 0.2, and a KL coefficient of 0.001. PrimeSeeker-Outcome uses final-answer correctness alone, whereas PrimeSeeker-D_q additionally uses the reference-step coverage reward defined in Section 3.3. Invalid formats, repeated generations, context failures, and tool-budget violations receive a failure reward. For all PrimeSeeker-D_q experiments, we set $\alpha = 0.10$, $\beta = 0.20$, and $C = 0.50$.

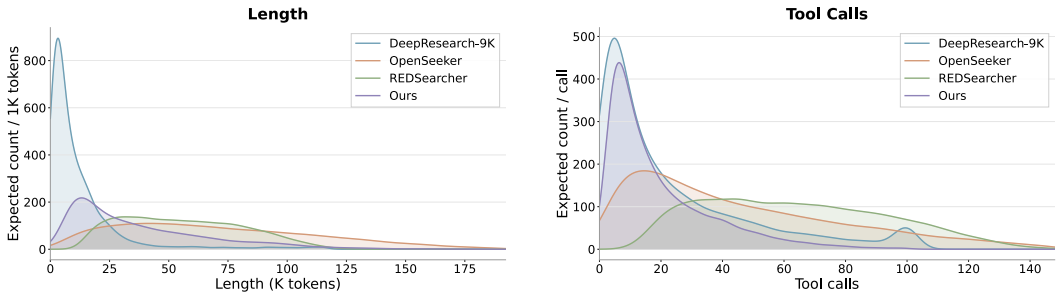
Rollout and inference. RL rollouts are sampled with temperature 1.0 and top-$p = 0.95$. For benchmark evaluation, we use temperature 0.8, top-$p = 0.95$, and a repetition penalty of 1.05. Unless otherwise specified, each trajectory is limited to 100 tool calls.

Construction-scale evaluation. A full-scale downstream comparison of alternative QA construction rules would require a new aligned corpus of questions and reference skeletons, fresh expert trajectories, and training the corresponding student policies. Unlike disabling a component on a fixed corpus, this changes the supervision consumed by downstream stages and entails substantial generation and retraining costs at the same scale of long-horizon trajectories. Our evaluation therefore combines complete-agent results with controlled comparisons of expert guidance and reference-step rewards.

Tool environment and auxiliary models. Following OpenSeeker (Du et al., 2026b), all experiments use the same live-web interaction environment and evaluation protocol, with Serper (Serper, 2025) for web search and Jina Reader (Jina AI, 2025) for page fetching. The construction operators, closed-book filter, $\mathrm{Extract}_D$, $\mathrm{Extract}_k$, trajectory verifier, reference-step auditor, and answer evaluator are served by the same `Qwen3-30B-A3B-Instruct-2507` deployment with a 230K-token context window. The complete tool and evaluation implementation is released at `https://anonymous.4open.science/r/PrimeSeeker-445D/`.

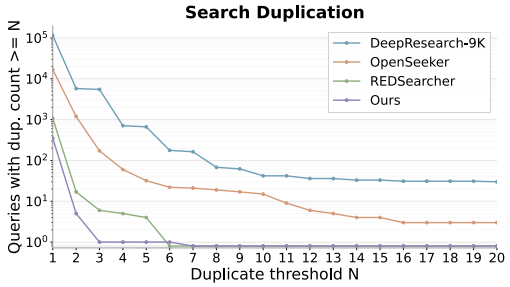
D Effect of Observation-grounded D_q Guidance

We compare unguided, D_q-guided, and random-highlight generation using Qwen3 and DeepSeek-V4 on questions with $K \in \{2, 3\}$ independently sourced specifications per anchor. The random control preserves the key-field format but samples two to five content words from the current observation independently of q and D_q; all other settings are identical. As shown in Table 4, D_q-guided highlights improve accuracy by 2%–7%, whereas random highlights reduce it by 23%–35% and add 25–65 tool calls. This contrast shows that the benefit arises from reference alignment rather than the additional key field itself.

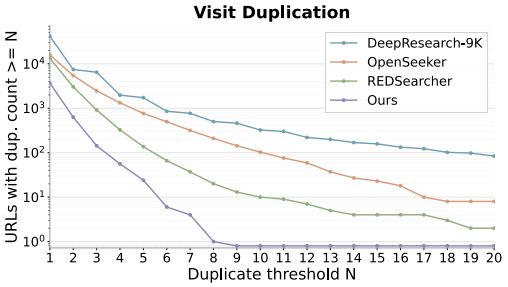


(a) Trajectory Length

(b) Tool Calls



(c) Duplicate Searches



(d) Duplicate URLs

Figure 4: Distributional diagnostics of expert trajectories. (a) Trajectory length distribution, shown as KDE-smoothed expected sample count per 1K-token window. (b) Tool-call distribution, shown as KDE-smoothed expected sample count per one-tool-call window. (c) CCDF of duplicated search-query counts within trajectories. (d) CCDF of duplicated visited-URL counts under the same criterion. PrimeSeeker trajectories occupy moderate tool-use ranges and exhibit lighter duplicate-retrieval tails, indicating more compact and less redundant expert supervision.

E Trajectory Distribution Diagnostics

Distributional analysis. To complement the aggregate statistics, Figure 4 visualizes the complete trajectory distributions. Figures 4a and 4b show KDE-smoothed expected sample counts over trajectory length and tool-call count, respectively. For trajectory length, the vertical axis represents the expected number of samples within a nearby 1K-token window after smoothing. For tool calls, it represents the expected number of trajectories around each tool-call count.

DeepResearch-9K is concentrated in very short trajectories, whereas OpenSeeker has the heaviest long-context tail. REDSearcher is concentrated in the mid-to-long range. PrimeSeeker occupies the short-to-middle range and has a substantially lighter long tail than OpenSeeker and REDSearcher. The tool-call distribution exhibits a similar pattern: DeepResearch-9K and PrimeSeeker concentrate on lower tool-call ranges, OpenSeeker has a broader middle-range distribution, and REDSearcher is shifted toward heavier tool usage. The small concentration of DeepResearch-9K trajectories near its maximum tool-call region may reflect a tool-budget boundary.

Figures 4c and 4d show complementary cumulative distributions of duplicate searches and duplicate page visits. The horizontal axis gives the duplicate count within a trajectory, while the vertical axis reports the number of trajectories whose duplicate count is at least that value. Lower curves therefore indicate lighter redundancy tails. PrimeSeeker exhibits consistently lighter duplicate-search and duplicate-visit tails, providing distribution-level evidence that its compactness is associated with less low-information retrieval rather than premature termination alone.

Together, these statistics show that PrimeSeeker does not rely on globally inflated trajectories and that its demonstrations contain substantially less repeated retrieval. Their downstream utility is evaluated through the trained policy results in Section 4.2.

F QA CONSTRUCTION CASE STUDY

We present two construction examples showing the aligned outputs q and D_q derived from a shared web-grounded anchor structure. Blue text denotes descriptive specifications, orange text denotes latent or recovered anchors, and green text denotes the target answer. Each specification is extracted from an independently retrieved source page; the complete provenance records are included in the released data.

Case 1: Isle of Skye

> **Question** q**.** In a 17th-century play, a character forces two others to perform a temporary marriage ceremony. This ceremony has a historical counterpart in a European country, where on a specific island, such a ceremony was recorded in the late 1600s. What is the name of that island?

Reference evidence skeleton D_q

- **R1** **Search cue:** CHAIN d_1
  Verification: *The Witch* is the 17th-century play described in the question.
- **R2** **Search cue:** CHAIN d_2
  Verification: Antonio forces Francisca and Aberzanes to perform a handfast ceremony.
- **R3** **Search cue:** CHAIN d_3
  Verification: Handfasting was associated with temporary or probationary marriage in later Scottish tradition.
- **R4** **Search cue:** ANSWER d_4
  Verification: A late-17th-century account records this custom on the Isle of Skye.

Latent-anchor sequence

17th-century play $\longrightarrow$ ***The Witch*** $\longrightarrow$ **temporary marriage ceremony** $\longrightarrow$ **handfasting** $\longrightarrow$ **Scottish tradition** $\longrightarrow$ **Isle of Skye**

Case 2: Castruccio Castracani

> **Question** q**.** A 13th-century treatise on human equality is linked to a historic mass emancipation of serfs in a major Italian city-state. Decades later, that city supported Florence against a Ghibelline ruler from a noble family of Lucca. This ruler had been a leading figure in the 1315 victory at Montecatini. Who was he?

Reference evidence skeleton D_q

- **R1** **Search cue:** CHAIN d_1
  Verification: *Liber Paradisus* is the 13th-century treatise associated with human equality.
- **R2** **Search cue:** CHAIN d_2
  Verification: The document records the collective emancipation of serfs by Bologna.
- **R3** **Search cue:** CHAIN d_3
  Verification: The city-state referred to in the question is Bologna.
- **R4** **Search cue:** CHAIN d_4
  Verification: Bologna supported Florence against a Ghibelline ruler from the Antelminelli/Castracani family of Lucca.
- **R5** **Search cue:** ANSWER d_5
  Verification: Castruccio Castracani was a leading figure in the 1315 Ghibelline victory at Montecatini.

Latent-anchor sequence

Case 2: Castruccio Castracani (continued)

13th-century treatise on human equality ⟶ *Liber Paradisus* ⟶ Bolognese mass emancipation ⟶ Bologna ⟶ Ghibelline ruler from Lucca ⟶ Castruccio Castracani

G Prompt Templates

We provide the prompts for the operators that instantiate and use the reference evidence skeleton. All operators use the auxiliary model specified in Section C. The exact upstream prompts used to expand anchor structures and verbalize questions are not included. Instead, we specify their input–output contracts and release the resulting questions, reference evidence skeletons, trajectories, and source provenance.

G.1 Key Observation Extraction

External validation of extracted highlights. The `key_observation` entries returned by the prompt below are treated as candidate highlights. Before they are used for trajectory generation, the surrounding pipeline checks each candidate by direct string matching against the current raw observation. Candidates not found verbatim are discarded, as are candidates containing a direct string match to the gold answer. Only the surviving spans are used as auxiliary highlights; if none survive, the original observation is passed to the expert without additional guidance. This post-processing filters the highlights without modifying the retrieved observation.

Prompt: Key Observation Extraction

[SYSTEM]

Role

You are an information filter for tool–augmented question answering.
The reference steps describe the evidence needed to solve the question.
Your task is to extract matching evidence from the current tool observation
and conservatively identify repeated search keywords or visit goals.

Procedure

1. Align the observation with the reference steps

– The raw observation must explicitly support the same specific fact described
by a reference step, such as an entity, name, date, event, or relation.
Broad topical similarity is insufficient.
– If no reference step is matched, set `matched_reference_index` and
`matched_reference_step` to null.

2. Extract the key observation

– Copy every sentence from the raw observation that provides direct evidence
for the matched reference step.
– Copy the evidence verbatim. Do not infer, summarize, rewrite, or repeat it.
– `matched_reference_step` must be copied verbatim from the reference steps.
– `key_observation` must contain verbatim sentences from the raw observation.
– The matched reference step and key observation must come from different
sources and must not be identical.
– If the evidence is uncertain, unsupported, or not a verbatim substring of
the raw observation, set `key_observation` to null.

Prompt: Key Observation Extraction (continued)

Hard Rules

– Return strict JSON only.
– Do not include markdown, explanations, or reasoning outside the JSON.
– Use null for any unavailable field.

Output Format

Return JSON with exactly the following keys:

{
matched_reference_index: 0,
matched_reference_step: <verbatim reference step or null>,
key_observation: [<verbatim sentence from raw_observation>],
matched_reason: <brief explanation of the evidence match or null>
}

[USER]

Question:
{question}

Reference steps:
{reference_steps}

Current search keywords or visit goal:
{search_keywords_or_visit_goal}

Raw observation:
{raw_observation}

Previous tool keywords:
{history_tool_keywords}

G.2 Reference-Step Coverage Auditing

Prompt: Reference-Step Coverage Auditing

[SYSTEM]

Role

You are an evidence auditor for a tool−using search agent. You may inspect hidden reference milestones, but you must evaluate coverage using only evidence available in the public trajectory. Hidden anchors and unretrieved facts must never be attributed to the agent.

Evidence Types

– `descriptor`: evidence supporting a descriptive clue for a hidden anchor
– `anchor`: evidence identifying or substantially narrowing a hidden anchor
– `transition`: evidence supporting a verifiable relation between anchors

Prompt: Reference-Step Coverage Auditing (continued)

– `answer`: evidence directly identifying or verifying the final answer

Evaluation Procedure

1. Audit each reference step

For every numbered reference step, determine:

– its evidence type;
– whether it is covered;
– the public trajectory event that supports it;
– the minimal supporting quote or paraphrase; and
– whether the evidence contributes to the reasoning chain or directly supports the answer.

2. Collect effective evidence

Record each effective evidence item separately and identify:

– the trajectory step where it appears;
– the reference steps it supports;
– its evidence role and evidence type;
– its importance for the search decision; and
– the next action it enables.

Use the following decision levels:

– `weak`: topical evidence that does not materially reduce uncertainty
– `useful`: evidence that enables a better query or anchor transition
– `decisive`: evidence that identifies a required anchor or directly verifies the answer

Use the following action implications:

– `continue_search`
– `transition_query`
– `answer_ready`
– `no_progress`

Set `answer_ready` only when the public evidence is sufficient to answer the question.

3. Assess trajectory reliability

Indicate whether the audit relies on an unsupported model guess, whether the trajectory contains repeated or irrelevant behavior, and the overall confidence of the audit.

Hard Rules

– Evaluate only the public trajectory.
– Return compact, strict JSON only.
– Do not include markdown, explanations, or reasoning outside the JSON.
– Include one item in `reference_step_audit` for every numbered reference step.

Prompt: Reference-Step Coverage Auditing (continued)

Output Format

Return JSON with exactly the following structure:

```
{
 reference_step_audit: [
   {
    ref_step: 1,
    milestone_type: descriptor | anchor | transition | answer | unknown,
    covered: true,
    event_type: tool_response | model_message | null,
    public_evidence: <minimal quote or paraphrase from the public trajectory or null>,
    evidence_role: chain | answer | none
   }
 ],
 all_effective_evidence: [
   {
    step: 1,
    event_type: tool_response | model_message,
    ref_steps: [1],
    evidence: <minimal public evidence>,
    evidence_role: chain | answer | none,
    milestone_type: descriptor | anchor | transition | answer | unknown,
    decision_level: weak | useful | decisive,
    next_action_implication: continue_search | transition_query | answer_ready | no_progress,
    why_effective: <brief justification>
   }
 ],
 used_model_guess_as_evidence: false,
 off_chain_or_redundant: false,
 confidence: 0.0
}
```

[USER]

Question:
{question}

Ground−truth answer:
{ground_truth}

Reference steps:
{reference_step}

Predicted final answer:
{predicted_answer}

Audit policy:
{audit_policy}

Step−labeled public trajectory:
{trajectory}

H TEST-TIME ROLLOUT CASE STUDY

We present a representative test-time rollout produced by the trained PrimeSeeker policy preserving the original order. The conversations are reproduced in their original order without rewriting or reordering intermediate reasoning, tool calls, and observations.

Case: Identifying a television episode from indirect evidence

Dataset: BrowseComp-EN **Model:** PrimeSeeker-D_q
Tool calls: 6 **Result:** Correct
Reference answer: *3rd Rock from the Sun; Season 1; Episode 2; Post-Nasal Dick*

question

```
question:"Provide the name of the TV series along with the season,
episode number, and episode title that meet the following criteria:
The episode aired in January between 1994 and 1997, inclusive.  One of
 the genres of the TV series is comedy.  In the episode, one character
 accompanies another, who is considered their love interest in the
series, to a wedding.  A few characters fall sick for the first time
and become convinced they are dying.  One filming mistake in the
episode involves a character placing their feet in a pot filled with
liquid, but by the end of the scene, their socks appear dry."
```

assistant

```
assistant:"
<think>
We need to identify a TV series episode that meets several specific
criteria.
...
</think>
<tool_call>
[
  {
    "function": {
      "name": "search",
      "arguments": {
        "query": [
          "TV episode aired January 1994 to 1997 comedy wedding"
        ]
      }
    }
  }
]
</tool_call>"
```

user

```
user:"Observation from search:
### A Google search for 'TV episode aired January 1994 to 1997 comedy
wedding' found 10 results:
..."
```

assistant

```
assistant:"
<think>
```

```
The search results did not directly identify the episode. Let me
refine the
query using the filming mistake.
</think>
<tool_call>
[
  {
    "function": {
      "name": "search",
      "arguments": {
        "query": [
          "\"feet in a pot\" \"dry socks\" TV episode mistake"
        ]
      }
    }
  }
]
</tool_call>"
```

user

```
user:"Observation from search:
### A Google search for '\"feet in a pot\" \"dry socks\" TV episode
mistake' found 0 results:

"
```

assistant

```
assistant:"
<think>
No direct results. Let me try searching for the filming mistake in a
different way, perhaps by looking for continuity errors in television
episodes.
</think>
<tool_call>
[
  {
    "function": {
      "name": "search",
      "arguments": {
        "query": [
          "continuity error TV episode feet in pot dry socks"
        ]
      }
    }
  }
]
</tool_call>"
```

user

```
user:"Observation from search:
### A Google search for 'continuity error TV episode feet in pot dry
socks' found 10 results:

1. [3rd Rock from the Sun (1996) revealing mistakes in Post Nasal Dick
](https://www.moviemistakes.com/tv4916/episode23240/revealing)
```

```
Harry put his feet in a pot of warm chicken soup, yet at the end of
that scene when they hug each other Harry lifts his feet up and you
can see his socks are ...

2. ..."
```

assistant

```
assistant:"
<think>
I found a likely match from 3rd Rock from the Sun. The filming mistake
 matches
the clue, but I still need to verify the episode title, season,
episode number,
air date, and plot details.
</think>
<tool_call>
[
  {
    "function": {
      "name": "visit",
      "arguments": {
        "url": "https://www.moviemistakes.com/tv4916/episode23240/
revealing",
        "goal": "Find details about the episode including the title,
season, episode number, air date, and plot summary"
      }
    }
  }
]
</tool_call>"
```

user

```
user:"Observation from visit:
The useful information in https://www.moviemistakes.com/tv4916/
episode23240/revealing for user goal Find details about the episode
including the title, season, episode number, air date, and plot
summary as follows:

Evidence in page:
The episode in question is titled 'Post Nasal Dick,' which is
explicitly mentioned in the URL and heading: 'Post Nasal Dick - S1-E2
'. It is part of Season 1, Episode 2 of '3rd Rock from the Sun.' The
episode features a revealing mistake involving Harry, who put his feet
 in a pot of warm chicken soup, but his socks later appear dry.
"
```

assistant

```
assistant:"
<think>
I now have all the confirmation I need.
...
</think>
<answer>
TV Series: 3rd Rock from the Sun
Season: 1
Episode Number: 2
Episode Title: "Post-Nasal Dick"
```

```
</answer>"
```

I RELEASED ARTIFACTS AND LIMITATIONS

We release the evaluation code, tool-use implementation, the 9,221 PrimeSeeker SFT questions and trajectories, and the construction-verified RL questions with their reference evidence skeletons. The release also contains source provenance, the $\mathrm{Extract}_D$ and $\mathrm{Extract}_k$ prompts, the reference-step auditor, and the evaluation configuration. The exact upstream prompts used to expand anchor structures and verbalize questions are not included. Models and data are available at `https://anonymous.4open.science/r/PrimeSeeker-445D`.

Several additional limitations should be considered.

- **Web non-stationarity.** Search results and webpage contents may change over time. Reproducing an individual trajectory may therefore yield observations different from those collected during data construction or evaluation.
- **Evaluator dependence.** Benchmark accuracy depends partly on the behavior of the answer evaluator. We release the evaluator prompt and code to make this source of variation explicit.
- **Evidence-skeleton coverage.** D_q records a compact answer-supporting structure, but it is not intended to enumerate every valid search path or every supporting webpage.
- **Open-ended candidate exploration.** Latent anchor resolution provides a local formulation of candidate-space reduction, but difficult tasks with sparse clues and extremely large candidate spaces may still require broader exploration.
- **Model dependence.** Closed-book filtering, trajectory generation, and evidence guidance depend on the capabilities and search styles of the selected expert models.