# SEER: GUIDING SMALL DEEP-RESEARCH AGENTS WITH SHARED DECISION MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Deep-research agents repeatedly decide what to search, read, and verify, often using large cloud-hosted language models for action proposal. However, privacy and local control motivate running agents on users' devices, where limited memory and compute favor smaller models and complicate on-device training. Yet inference-time guidance for these agents faces three obstacles: search through real execution adds tool calls, prompted self-evaluation relies on their judgment, and model-specific guidance risks repeated adaptation costs. We introduce **Seer**, an inference-time framework providing local guidance that separates action proposal by the agent's model from learned consequence prediction and value estimation. From a shared corpus of real research trajectories, we train two models: a tool-response simulator on observed call–response pairs and a value estimator on histories labeled with discounted terminal correctness. The simulator and estimator are trained offline and reused across agents without further training. Moreover, Seer executes only the selected action and adds only real tool responses to the committed history, avoiding tool calls on discarded candidates at additional inference cost. Across multiple deep-research benchmarks, Seer substantially improves agent performance for different language models. For instance, GAIA accuracy rises from 13.6% to 41.7% for Qwen3.5-4B and from 22.3% to 50.5% for Qwen3.5-9B. Furthermore, our analyses suggest that broader candidate coverage can compensate for imperfect simulation, while prediction and scoring errors limit the benefits of wider imagined search. These findings highlight the potential of reusable models for prediction and evaluation to strengthen small research agents through additional local computation without real execution of discarded candidates.

## 1 INTRODUCTION

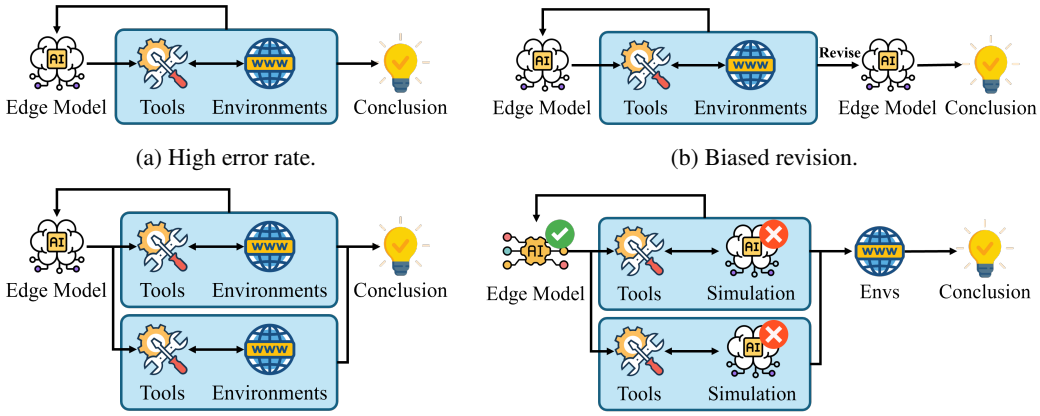
Deep-research agents use language models to decide what to search, read, and verify, integrating evidence across sources to answer complex questions (Zhang et al., 2025). In practice, OpenAI Deep Research (OpenAI, 2025) and Gemini Deep Research (Citron, 2024) offer multi-step web investigation and source-linked reports, while open efforts such as DeepResearcher (Zheng et al., 2025) and Tongyi DeepResearch (Tongyi DeepResearch Team, 2025) advance research on sustained information seeking. Meanwhile, benchmarks from GAIA's real-world reasoning and tool-use questions (Mialon et al., 2023) to BrowseComp's challenging web information-seeking tasks (Wei et al., 2025) evaluate agents' ability to gather and integrate evidence.

For users applying these capabilities to sensitive tasks, keeping an agent's reasoning and decision-making on-device offers greater local control, a motivation reflected in privacy-oriented personal assistants (Apple, 2024). However, deploying these agents on users' devices requires managing both local computational resources and external tool-use costs. Memory and compute constraints favor smaller language models (Belcak et al., 2025), while tool-call fees make repeated searches and page retrievals an additional deployment expense (Liu et al., 2026).

Under these constraints, training small models specifically for research is a natural way to improve agent performance. However, collecting training data and optimizing each model requires additional resources, making repeated adaptation costly for resource-constrained deployments. Instead, inference-time methods offer an alternative by improving agent decisions without updating the deployed model. Specifically, **output-level refinement** (Wang et al., 2023; Zeng et al., 2025;



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.




(a) High error rate. (b) Biased revision.

(c) Additional tool costs. (d) Limited generalization across agent models.

Figure 1: How can we provide *reusable* guidance that *strengthens small agents* without model-specific retraining or *real execution* of discarded candidates?

Wan et al., 2026) selects or revises completed answers but cannot redirect actions already executed. **Decision-level evaluation** (Zhou et al., 2024; Koh et al., 2024; Xia et al., 2025; Xiong et al., 2026) guides intermediate choices through search or learned critics, but evaluating branches through real execution incurs tool costs even for discarded candidates. **Model-based lookahead** (Hao et al., 2023; Gu et al., 2025; Chae et al., 2025) reduces this overhead by evaluating predicted consequences before execution. To avoid shifting adaptation costs to supporting components, such guidance should transfer across agent models without retraining. Yet approaches relying on the agent's own model for prompted prediction and scoring remain constrained by its capabilities (Vijayvargiya & Lokesh, 2025). Figure 1 summarizes these limitations. The challenge is therefore to *provide reusable guidance that strengthens small agents without model-specific retraining or real execution of discarded candidates*.

We introduce **Seer**, an inference-time framework providing local guidance in deep research. Our key idea is to *decouple action proposal from consequence prediction and value estimation to enable reusable guidance across agents*. From a shared corpus of real research trajectories collected across tasks and agents, we train two models: a *tool-response simulator* on observed call–response pairs and a *value estimator* on interaction histories labeled with discounted terminal correctness. These complementary supervision signals teach the simulator to predict what an action may reveal and the estimator to assess how the resulting history relates to eventual task success. The simulator and estimator are trained offline and reused across agents powered by different models. *Deployment requires neither updating the agent's model nor retraining the simulator or estimator.* Specifically, at inference, the agent proposes candidate actions, the simulator predicts their tool responses, and the estimator scores the resulting imagined histories to guide action selection. Seer executes only the selected action and adds real tool responses to the committed history. Candidate evaluation therefore uses additional inference computation rather than real tool calls on discarded alternatives, while the agent retains real rather than simulated responses as evidence. Our contributions are threefold.

1. We propose Seer, an inference-time framework that decouples action proposal from shared tool-response prediction and history-level value estimation. This design enables reusable guidance across frozen agents without model-specific retraining or real execution of discarded candidates.
2. We develop a shared training pipeline that collects real trajectories across agents on existing datasets through Monte Carlo branching. The resulting large-scale corpus provides complementary supervision: observed call–response pairs for the simulator and history prefixes labeled with discounted terminal correctness for the estimator.
3. Experiments with different models across extensive benchmarks demonstrate substantial gains for small agents, with benefits extending to larger-model settings. Further ablations and analyses reveal the trade-off between candidate coverage and imperfect prediction and scoring.

## 2 Related Work

**Efficient research agents.** Quantization (Lin et al., 2023), context management (Vijayvargiya & Lokesh, 2025), and tool budgeting (Liu et al., 2026) address deployment costs, while compact

research agents (Chen et al., 2026; Li et al., 2026; Venus Team et al., 2026) target research capability. Distillation (Gu et al., 2024; Agarwal et al., 2024) and reinforcement learning (Zheng et al., 2025) strengthen the agent's model but require optimization for each adapted model. Seer instead trains reusable decision models, complementing these approaches without updating the deployed agent.

**Inference-time guidance.** Output-level methods select or revise solutions (Wang et al., 2023; Madaan et al., 2023; Cobbe et al., 2021), but cannot recover tool calls already spent on discarded trajectories. Decision-level methods intervene earlier: LATS (Zhou et al., 2024) and web-agent tree search (Koh et al., 2024) use environment feedback, while AgentRM (Xia et al., 2025) learns a transferable reward model. However, evaluating branches through real execution still incurs tool costs, and prompted self-evaluation depends on the agent's own judgment.

**Model-based lookahead.** RAP (Hao et al., 2023) predicts action consequences through LLM simulation; WMA (Chae et al., 2025) and WebDreamer (Gu et al., 2025) develop learned prediction for web-agent lookahead. However, avoiding real candidate execution does not by itself ensure that both prediction and evaluation transfer across agent models without retraining. To enable this reuse, Seer separates call-level response prediction and history-level evaluation from the agent model, training the evaluator on discounted terminal correctness and reusing both checkpoints across agents. It executes only the selected action and retains real responses as evidence, trading additional local inference for comparison without tool calls on discarded candidates.

## 3 Problem Formulation

Given a request $q$, a deep-research agent $\mathcal{G}$ uses a language model to decide what to search, read, and verify, integrating evidence across sources to produce an answer or report $\hat{a}$ (Zhang et al., 2025). During a run, the agent maintains a committed interaction history $s_t$, which initially contains $q$ and the agent instructions. At round $t$, it uses this history to select an action $a_t$ that either submits a final answer without tool calls or issues one or more tool calls. To describe this choice mathematically, let $\pi_{\mathcal{G}}(\cdot \mid s_t)$ denote the agent's distribution over actions given the current history.

Let $\mathcal{E}$ denote the real tool-execution operator, which invokes the tools specified in $a_t$ with their arguments in the current external environment and returns the responses $o_t$. Action selection and the history update for a nonterminal interaction are then written as

$$a_t \sim \pi_{\mathcal{G}}(\cdot \mid s_t), \quad o_t = \mathcal{E}(a_t), \quad s_{t+1} = s_t \oplus (a_t, o_t), \tag{1}$$

where $\oplus$ denotes concatenation. Thus, the next action is selected using a history that includes the previous history, the selected action, and its real tool responses. A final-answer submission instead terminates the run with $o_t = \varnothing$.

We denote the full execution record by $\xi$ and assess its outcome using a task-dependent terminal quality score $R_{\mathrm{term}}(\xi; q) \in [0, 1]$. For the question-answering tasks considered here, this score is 1 for a correct final answer and 0 otherwise, including unanswered tasks. Gold answers are used only for evaluation and offline supervision, not for inference-time decisions. For requests $q \sim \mathcal{D}$ drawn from the task distribution and traces $\xi$ generated by $\mathcal{G}$ through real tool interactions, the objective is

$$\max_{\mathcal{G}} \ \mathbb{E}_{q\sim\mathcal{D},\xi}\big[R_{\mathrm{term}}(\xi; q)\big]. \tag{2}$$

When the agent's reasoning and decision-making run on-device, this objective is constrained by local model memory and the cost of real tool calls to external services. We denote the total number of locally deployed model parameters by $c_{\mathrm{param}}$ and the number of real tool calls by $c_{\mathrm{tool}}$, and collect them in

$$\mathbf{c}(\mathcal{G}, \xi) = \big(c_{\mathrm{param}}(\mathcal{G}),\, c_{\mathrm{tool}}(\xi)\big)^{\top}. \tag{3}$$

Given the corresponding budget vector $\mathbf{B}$, the on-device problem becomes

$$\max_{\mathcal{G}} \ \mathbb{E}_{q\sim\mathcal{D},\xi}\big[R_{\mathrm{term}}(\xi; q)\big] \quad \text{s.t.} \quad \mathbf{c}(\mathcal{G}, \xi) \preceq \mathbf{B}, \tag{4}$$

where $\preceq$ denotes componentwise inequality. The per-run count $c_{\mathrm{tool}}$ includes all real calls recorded in $\xi$, including those made for discarded candidates, with batched calls counted individually. A run stops on submission or budget exhaustion.

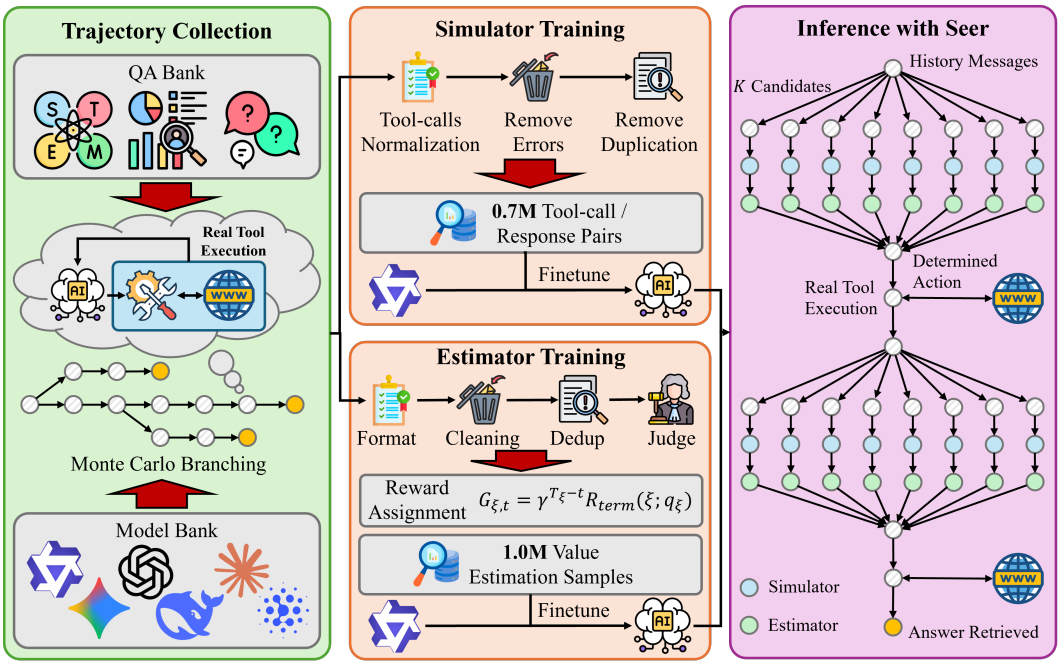



Figure 2: Seer overview. Monte Carlo branching collects real trajectories to train the simulator on tool responses and the estimator on discounted terminal correctness. At inference, these shared models guide search over imagined histories, while only selected actions are executed to obtain real evidence.

To compare candidates without increasing $c_{\text{tool}}$, let $\widehat{\mathcal{E}}$ denote a local approximation to tool responses, following planning with predicted action outcomes (Hao et al., 2023; Gu et al., 2025). For a candidate action $a^{(i)}$ at history $s_t$, the predicted response and imagined history are

$$\hat{o}^{(i)} = \widehat{\mathcal{E}}(a^{(i)}), \qquad \hat{s}^{(i)} = s_t \oplus (a^{(i)}, \hat{o}^{(i)}), \tag{5}$$

with $\hat{o}^{(i)} = \varnothing$ for final-answer candidates. These imagined histories support candidate comparison, while the committed update still uses $o_t = \mathcal{E}(a_t)$. Our focus is therefore to improve the objective in Eq. (4) for a fixed agent model using predicted outcomes for action selection, while satisfying the same resource budgets. Appendix A summarizes the notation used throughout the paper.

## 4 Our Approach

The response approximation in §3 leaves two questions: what might a candidate action reveal, and how useful is the resulting history for completing the task? Seer addresses them with a *tool-response simulator* $W_\phi$ and a *value estimator* $V_\theta$ supervised by discounted terminal correctness, while the agent's frozen model proposes actions (Figure 2). We refer to these two components jointly as the *decision models*, and individually as the simulator and estimator.

### 4.1 Decoupling Prediction from Evaluation

**Predicting tool responses.** Let $\pi$ denote the frozen agent model's action-proposal distribution and $\text{calls}(a)$ the tool invocations in action $a$. For each call $c$, containing a tool name and arguments, $W_\phi$ generates a response $\hat{o}_c$ conditioned only on that call and a tool-specific instruction. Collecting these responses in call order implements the local approximation

$$\hat{o}_c \sim W_\phi(\cdot \mid c), \qquad \widehat{\mathcal{E}}(a) = \big(\hat{o}_c\big)_{c \in \text{calls}(a)}. \tag{6}$$

This interface omits conversation history and agent-model identity, allowing reuse across agents with a common tool schema.

**Evaluating candidate histories.** Predicting a response does not establish its usefulness; $V_\theta$ therefore evaluates the resulting history using supervision from terminal task outcomes. For each proposal

$a^{(i)} \sim \pi(\cdot \mid s_t)$, we construct the imagined history as in Eq. (5) and evaluate it with $V_\theta$:

$$\hat{s}^{(i)} = s_t \oplus \big(a^{(i)}, \widehat{\mathcal{E}}(a^{(i)})\big), \qquad v^{(i)} = V_\theta(\hat{s}^{(i)}). \tag{7}$$

For final-answer candidates, $\widehat{\mathcal{E}}(a^{(i)}) = \varnothing$, so the estimator scores the history with the proposed answer appended. Search uses these scores to change the overall action-selection distribution $\pi_{\mathcal{G}}$, not the proposal distribution $\pi$. Both learned components are trained offline and reused across agents without further training.

### 4.2 Learning from Real Research Trajectories

We collect a shared corpus using agents powered by different frozen language models on training questions disjoint from the evaluation questions. Inspired by Monte Carlo search for process-supervision collection (Luo et al., 2024), we branch from intermediate histories to collect alternative continuations, all through real tool execution. The same trajectories thus supply two complementary signals: observed responses for $W_\phi$ and discounted terminal correctness for $V_\theta$.

**Learning Tool-Response Prediction**: We extract individual call–response pairs from the corpus, normalize calls, remove infrastructure failures, and deduplicate to obtain $\mathcal{D}_W = \{(c_i, o_i)\}_{i=1}^{N_W}$ with $N_W$ retained pairs. Unlike value supervision, response supervision does not depend on terminal correctness: even an unsuccessful trajectory can contain valid tool outputs.

Starting from a pretrained autoregressive language model, we optimize its trainable parameters $\phi$ using a response-only negative log-likelihood:

$$\mathcal{L}_W(\phi) = -\mathbb{E}_{(c,o)\sim\mathcal{D}_W} \sum_{\ell=1}^{|o|} \log W_\phi(o_\ell \mid c, o_{<\ell}), \tag{8}$$

where $o_\ell$ is response token $\ell$ and $o_{<\ell}$ denotes its preceding tokens. The tool-specific instruction and call are omitted from the loss; training and inference use the same input template (Appendix D).

**Learning Outcome-Supervised Value Estimation**: For value supervision, we deduplicate trajectories with submitted answers, excluding unanswered error or timeout runs. A judge compares each submission with its gold answer to assign $R_{\text{term}}(\xi; q_\xi) \in \{0, 1\}$; gold answers enter only label construction, never the estimator input. For a trajectory $\xi$ with question $q_\xi$ and terminal history index $T_\xi$, we label each retained prefix by discounted terminal correctness:

$$\mathcal{D}_V = \{(s_{\xi,t}, G_{\xi,t})\}_{\xi,t}, \qquad G_{\xi,t} = \gamma^{T_\xi - t} R_{\text{term}}(\xi; q_\xi), \tag{9}$$

where $0 < \gamma < 1$ discounts remaining interaction steps, not tokens or individual tool calls. Incorrect trajectories receive zero throughout; correct ones receive increasing targets toward completion. Consequently, $V_\theta$ ranks histories by outcome and proximity to completion rather than estimating a calibrated success probability for a specific deployment model.

We replace a pretrained language model's head with an HL-Gauss classification head (Farebrother et al., 2024), with $\theta$ denoting the resulting model's trainable parameters. Let $\beta_0 < \cdots < \beta_K$ delimit $K$ uniform bins over an interval covering the targets, with centers $z_b = (\beta_{b-1} + \beta_b)/2$. For a target $G$, Gaussian smoothing with standard deviation $\sigma$ assigns the normalized bin masses

$$y_b(G) = \frac{\Phi((\beta_b - G)/\sigma) - \Phi((\beta_{b-1} - G)/\sigma)}{\Phi((\beta_K - G)/\sigma) - \Phi((\beta_0 - G)/\sigma)}, \tag{10}$$

where $\Phi$ is the standard normal cumulative distribution function. Given predicted bin probabilities $p_\theta(b \mid s)$, we minimize soft-label cross-entropy and decode by expectation:

$$\mathcal{L}_V(\theta) = -\mathbb{E}_{(s,G)\sim\mathcal{D}_V} \sum_{b=1}^{K} y_b(G) \log p_\theta(b \mid s), \qquad V_\theta(s) = \sum_{b=1}^{K} z_b p_\theta(b \mid s). \tag{11}$$

For efficient training, each trajectory is encoded once in a causal forward pass, with scores read at its labeled prefix endpoints. This requires token-identical shared prefixes and exclusion of trajectories beyond the context limit. Training and validation are split by question, keeping all trajectories and prefixes for the same question together.

### 4.3 IMAGINED SEARCH AND REAL EXECUTION

At each real round, Seer builds a fresh tree rooted at the committed history $s_t$ and runs $N_{\text{sim}}$ simulations using predictor upper confidence bound applied to trees (PUCT) (Rosin, 2011; Silver et al., 2018). Algorithm 1 in Appendix C gives the complete round, with candidate width $k$ and depth cap $d_{\max}$. Here, a search simulation is one tree traversal and value backup, distinct from predicting an individual tool response with the simulator.

**Candidate expansion.** At an expandable history $\bar{s}$, we independently sample $m$ valid actions from $\pi(\cdot \mid \bar{s})$, with $1 \le m \le k$, and batch their tool calls through $W_\phi$. The resulting imagined histories become children with uniform edge priors $P(e) = 1/m$; final-answer children are terminal. Because $V_\theta$ is trained on real histories, simulation errors can distort its rankings, motivating the depth cap.

**Selection and backup.** For an edge $e$ with parent node $u$, selection uses

$$\text{score}(e) = Q(e) + c_{\text{puct}} P(e) \frac{\sqrt{N(u)+1}}{1+N(e)}, \tag{12}$$

where $N(u)$ and $N(e)$ are visit counts, $Q(e)$ is the mean backed-up value, and $c_{\text{puct}}$ controls exploration. Each simulation scores its endpoint with $V_\theta$ and backs up that value along the traversed path, including when the endpoint is terminal or depth-capped. Thus, terminal candidates are evaluated by the learned estimator, not by real execution or gold-answer judgments.

**Real execution.** After search, we commit the action on the most-visited root edge:

$$e^* \in \arg\max_{e \in \text{root edges}} N(e), \qquad a_t = \text{action}(e^*). \tag{13}$$

A final-answer submission terminates the run; otherwise, we execute $a_t$ and update $s_{t+1} = s_t \oplus (a_t, \mathcal{E}(a_t))$. Imagined branches are discarded, so only real responses enter the next round's root history. Accordingly, Seer's real tool-call count is

$$c_{\text{tool}}(\xi) = \sum_t \big|\text{calls}(a_t)\big|, \tag{14}$$

where final-answer actions contribute zero. Discarded candidates incur local inference but no real calls. Total calls can nevertheless differ from the unassisted agent because action choices and run length change. Meanwhile, Seer uses additional local computation and model memory for candidate comparison without relaxing the resource budgets.

## 5 EXPERIMENTS

### 5.1 EXPERIMENTAL SETUP

Table 1: Preliminary comparison of RAP@4 and Seer ($k = 4$) using average accuracy across all benchmarks (%). RAP@4 is omitted from the main experiments due to a lack of gains over Base.

| Model | Base | RAP@4 | Seer |
|---|---|---|---|
| Qwen3.5-4B | 8.3 | 8.0 | **18.8** |
| Qwen3.5-9B | 14.2 | 13.9 | **25.2** |

**Training Setup.** Frozen agents collect real trajectories by sampling questions round-robin from DeepDive (Lu et al., 2025), MiroVerse (MiroMind Team et al., 2025), Nemotron RL (NVIDIA Corporation, 2025), OpenSeeker (Du et al., 2026), and REDSearcher (Chu et al., 2026), excluding evaluation questions. This yields 0.7M call–response pairs for the simulator and 1M training samples for the estimator. All methods share two external tools: `search` returns titles, links, and snippets, while `visit` retrieves pages and extracts goal-relevant evidence. Both models are initialized from Qwen3.5-9B (Qwen Team, 2026a) and use LoRA (Hu et al., 2021) at a learning rate of $10^{-4}$, with simulator/estimator ranks of 64/32 and context limits of 16k/128k tokens. The estimator learns discounted terminal correctness ($\gamma = 0.95$) through a fully trainable HL-Gauss head (Farebrother et al., 2024) with 201 bins over $[-1, 1]$ and a Gaussian standard deviation of 0.75 bin widths. Both checkpoints are reused unchanged across agents.

**Evaluation Setup.** We report final-answer accuracy on GAIA (Mialon et al., 2023), BrowseComp-200 (BC-200) (Wei et al., 2025), BrowseComp-ZH (BC-ZH) (Zhou et al., 2025a), SEAL-0 (Pham et al., 2025), and XBench-DeepSearch (XBench-DS) (Chen et al., 2025a), following each benchmark's evaluation protocol. The main comparison covers Qwen3.5-4B/9B/35B-A3B (Qwen Team, 2026a), Qwen3.6-35B-A3B (Qwen Team, 2026b), and Gemma-4-e2b-it/e4b-it (Gemma Team, 2026). For

Table 2: Comparisons of final-answer accuracy (%) across Qwen and Gemma agents using Base Agent, BoN@4, AV@4, and Seer ($S_k$, with candidate width $k$). Seer improves over Base Agent in 26 of 30 settings at $k = 32$ while reusing the same simulator and estimator across agents.

| | Qwen3.5-4B | | | | | Qwen3.5-9B | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Benchmark | Base | BoN | AV | $S_4$ | $S_{32}$ | Base | BoN | AV | $S_4$ | $S_{32}$ |
| GAIA | 13.6 | 21.4 | 26.2 | 29.1 | **41.7** | 22.3 | 35.9 | 35.0 | 35.9 | **50.5** |
| BC-200 | 1.0 | 4.5 | 3.5 | 5.0 | **7.0** | 2.5 | 5.5 | 6.5 | 5.5 | **9.5** |
| SEAL-0 | 4.5 | 9.9 | 8.1 | 10.8 | **16.2** | 10.8 | 23.4 | 18.9 | 22.5 | **26.1** |
| XBench-DS | 20.0 | 36.0 | 35.0 | 41.0 | **43.0** | 28.0 | 38.0 | 38.0 | 47.0 | **60.0** |
| BC-ZH | 2.4 | 7.6 | 5.2 | 8.0 | **9.7** | 7.6 | 11.4 | 13.5 | 15.2 | **16.3** |
| | Qwen3.5-35B-A3B | | | | | Qwen3.6-35B-A3B | | | | |
| Benchmark | Base | BoN | AV | $S_4$ | $S_{32}$ | Base | BoN | AV | $S_4$ | $S_{32}$ |
| GAIA | 57.3 | 53.4 | 47.6 | **62.1** | 57.3 | 64.1 | 75.7 | 73.8 | 78.6 | **79.6** |
| BC-200 | 22.0 | 24.0 | 24.0 | **24.5** | 22.0 | 20.5 | 35.0 | 35.0 | 36.0 | **37.5** |
| SEAL-0 | 27.0 | 28.8 | 33.3 | 31.5 | **34.2** | **29.7** | 20.7 | 28.8 | 24.3 | 27.0 |
| XBench-DS | 64.0 | 67.0 | 64.0 | 66.0 | **69.0** | 61.0 | **76.0** | 70.0 | 73.0 | 74.0 |
| BC-ZH | 36.3 | 37.7 | 34.9 | 36.0 | **43.3** | 27.0 | 35.3 | 36.0 | 35.3 | **38.8** |
| | Gemma-4-e2b-it | | | | | Gemma-4-e4b-it | | | | |
| Benchmark | Base | BoN | AV | $S_4$ | $S_{32}$ | Base | BoN | AV | $S_4$ | $S_{32}$ |
| GAIA | 20.4 | 23.3 | 25.2 | 30.1 | **31.1** | 21.4 | 25.2 | 30.1 | 24.3 | **36.9** |
| BC-200 | 1.5 | 1.0 | 1.0 | **2.5** | 1.0 | 0.5 | **1.5** | **1.5** | 1.0 | **1.5** |
| SEAL-0 | 1.8 | 5.4 | 4.5 | 4.5 | **6.3** | 6.3 | **9.0** | 8.1 | 5.4 | 7.2 |
| XBench-DS | 21.0 | 21.0 | 29.0 | 28.0 | **31.0** | 23.0 | 28.0 | 39.0 | 27.0 | **40.0** |
| BC-ZH | 2.1 | 1.0 | 2.8 | 1.7 | **3.8** | 1.4 | 1.0 | **4.2** | 3.8 | 3.5 |

each model, all methods share the frozen agent and tools and use the model's recommended sampling settings. *Base Agent* runs one ReAct-style trajectory (Yao et al., 2023b), whereas *BoN@4* and *AV@4* run four and select by normalized-answer majority voting (Wang et al., 2023) and prompted outcome verification (Zeng et al., 2025), respectively, without gold-answer access. We also implement *RAP@4* (Hao et al., 2023), but omit it from the main table after prompted simulation and self-evaluation yield little benefit on 4B/9B agents, as illustrated in Table 1. Each trajectory is limited to 100 committed agent rounds, 128k input tokens, and 9000 s, with a direct-answer attempt upon reaching a limit. Speculative proposals do not count toward the round limit. Seer uses depth-one imagined search with candidate width $k \in \{4, 8, 16, 32\}$, reporting $k = 4$ and $k = 32$ as low-width and scaled settings in the main comparison. These widths are not compute-matched to four full trajectories, whose real tool calls are all incurred, while Seer executes only selected actions. In tables, Base, BoN, and AV abbreviate Base Agent, BoN@4, and AV@4, respectively; $S_k$ denotes Seer with candidate width $k$, labeled Seer@$k$ in figures. In all result tables, bold and underlined values indicate the best and second-best results within each comparison group, respectively, including ties.

### 5.2 Analysis

**Main Results Across Agent Models**: Table 2 shows that Seer supports Qwen and Gemma agents, reusing the same simulator and estimator without model-specific retraining. With four candidates per decision ($S_4$), Seer outperforms both four-trajectory baselines on all five benchmarks with Qwen3.5-4B and reaches 47.0% on XBench-DeepSearch with 9B, versus 38.0% for both baselines. Expanding to $S_{32}$ raises GAIA accuracy to 41.7% and 50.5% for 4B and 9B, respectively, and surpasses both baselines on every benchmark with 9B. Gains extend to Gemma-4, with $S_{32}$ raising GAIA accuracy from 20.4% to 31.1% for e2b-it and from 21.4% to 36.9% for e4b-it. Across all six models, $S_{32}$ improves over Base Agent in 26 of 30 comparisons and reaches 79.6% on GAIA with Qwen3.6-35B-A3B without retraining either shared component. Unlike trajectory-level selection, both Seer settings evaluate alternatives without real tool calls for discarded candidates, trading additional inference computation for broader search rather than paying to execute every alternative.

**Scaling to Larger Agent Models**: Although designed for small agents, Seer also benefits substantially larger models (Table 3). Reusing the same simulator and estimator without retraining, $S_{32}$ improves

Table 3: Comparisons of final-answer accuracy (%) on larger agents. Seer transfers to 122B and 397B agents without retraining, with $S_{32}$ outperforming Base Agent in all cases.

| | Qwen3.5-122B-A10B | | | | | Qwen3.5-397B-A17B | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Benchmark | Base | BoN | AV | $S_4$ | $S_{32}$ | Base | BoN | AV | $S_4$ | $S_{32}$ |
| GAIA | 67.0 | 64.1 | 61.2 | 68.0 | **71.8** | 68.9 | 74.8 | 69.9 | **76.7** | 72.8 |
| BC-200 | 31.5 | 32.5 | 35.0 | 35.0 | **39.0** | 34.5 | 37.5 | 37.5 | 37.0 | **42.5** |
| SEAL-0 | 32.4 | 28.8 | 31.5 | 34.2 | **36.9** | 31.5 | 36.9 | 36.0 | 37.8 | **40.5** |
| XBench-DS | 66.0 | 70.0 | 61.0 | 71.0 | **72.0** | 75.0 | 73.0 | 75.0 | 79.0 | **80.0** |
| BC-ZH | 39.4 | 38.1 | 36.3 | 40.5 | **43.6** | 50.2 | 49.8 | 49.8 | **55.7** | 55.4 |

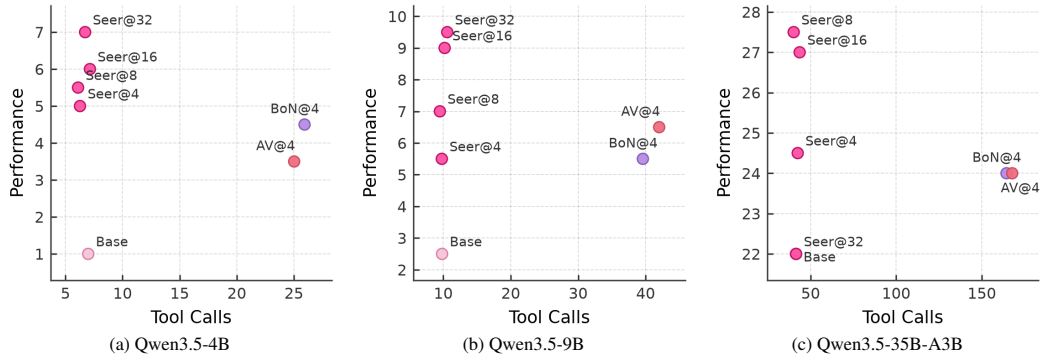
Table 4: Comparisons of cumulative component additions on GAIA across agents (accuracy %, $k = 4$). Adding the estimator improves over prompted scoring, while subsequently adding the simulator avoids real calls for discarded candidates at an accuracy cost.

| | Qwen3.5 | | | | | Qwen3.6 | Gemma-4 | |
|---|---|---|---|---|---|---|---|---|
| Variant | 4B | 9B | 35B-A3B | 122B-A10B | 397B-A17B | 35B-A3B | e2b-it | e4b-it |
| Base + search ($k = 4$) | 16.5 | 44.7 | 62.1 | 71.8 | 73.8 | 74.8 | 26.2 | 19.4 |
| + estimator | **38.8** | **45.6** | **66.0** | **72.8** | 71.8 | 75.7 | **31.1** | **29.1** |
| + simulator (Seer) | 29.1 | 35.9 | 62.1 | 68.0 | **76.7** | **78.6** | 30.1 | 24.3 |

over Base Agent on all five benchmarks for both Qwen3.5-122B-A10B and Qwen3.5-397B-A17B and outperforms both four-trajectory baselines in nine of ten comparisons. For 397B, it raises BrowseComp-200 accuracy from 34.5% to 42.5% and SEAL-0 from 31.5% to 40.5%. These gains show that Seer's shared guidance transfers across model scales rather than benefiting only small agents, without implying these larger models are suitable for on-device deployment.

**Contributions of Prediction and Evaluation**: Table 4 cumulatively builds Seer from the frozen Base Agent with four candidates per decision, all executed and ranked by prompted scoring. Replacing prompted scoring with the estimator improves accuracy by 0.9–22.3 points on seven of eight models, except Qwen3.5-397B-A17B. Retaining the estimator and adding simulator predictions lowers accuracy on six models at $k = 4$ but eliminates real calls for discarded candidates. Increasing Seer to $k = 32$ matches or exceeds real-response selection at $k = 4$ on six of eight models (Tables 2, 3, and 4), showing that broader search can compensate for prediction-related losses through additional inference rather than execution of discarded candidates.

**Accuracy versus Real Tool Use**: Figure 3 shows that Seer improves the accuracy–tool-cost trade-off on BrowseComp-200 across all agents. Its tool-call costs remain close to those of Base Agent across the tested widths, whereas BoN@4 and AV@4 incur substantially more real calls by executing four complete trajectories. These results illustrate the benefit of evaluating alternatives through simulation and executing only selected actions: additional inference computation enables more effective search without paying the real tool costs of discarded candidates.




(a) Qwen3.5-4B (b) Qwen3.5-9B (c) Qwen3.5-35B-A3B

Figure 3: Comparisons of final-answer accuracy and real tool-call counts on BrowseComp-200 across agents. The vertical axis (Performance) reports accuracy (%). Seer achieves the highest accuracy while keeping tool-call counts close to those of Base Agent.

Table 5: Comparisons of Seer accuracy (%) across candidate widths $k$ on all five benchmarks, with the best and second-best results within each model and benchmark shown in bold and underlined, respectively, including ties. Larger $k$ does not consistently improve performance, highlighting the need to select a suitable width for each agent and task.

| | Qwen3.5-4B | | | | Qwen3.5-9B | | | | Qwen3.5-35B-A3B | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Benchmark | $k$=4 | $k$=8 | $k$=16 | $k$=32 | $k$=4 | $k$=8 | $k$=16 | $k$=32 | $k$=4 | $k$=8 | $k$=16 | $k$=32 |
| GAIA | 29.1 | 35.9 | 39.8 | **41.7** | 35.9 | 38.8 | 42.7 | **50.5** | 62.1 | **64.1** | 59.2 | 57.3 |
| BC-200 | 5.0 | 5.5 | 6.0 | **7.0** | 5.5 | 7.0 | 9.0 | **9.5** | 24.5 | **27.5** | 27.0 | 22.0 |
| SEAL-0 | 10.8 | 10.8 | 15.3 | **16.2** | 22.5 | 24.3 | 23.4 | **26.1** | 31.5 | 29.7 | 31.5 | **34.2** |
| XBench-DS | 41.0 | 39.0 | **43.0** | **43.0** | 47.0 | 50.0 | 51.0 | **60.0** | 66.0 | 69.0 | **71.0** | 69.0 |
| BC-ZH | 8.0 | 9.3 | 9.0 | **9.7** | 15.2 | 15.9 | 15.9 | **16.3** | 36.0 | 41.5 | 41.0 | **43.3** |

| | Qwen3.5-122B-A10B | | | | Qwen3.5-397B-A17B | | | | Qwen3.6-35B-A3B | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Benchmark | $k$=4 | $k$=8 | $k$=16 | $k$=32 | $k$=4 | $k$=8 | $k$=16 | $k$=32 | $k$=4 | $k$=8 | $k$=16 | $k$=32 |
| GAIA | 68.0 | 68.9 | **72.8** | 71.8 | **76.7** | **76.7** | 71.8 | 72.8 | 78.6 | 79.6 | **80.6** | 79.6 |
| BC-200 | 35.0 | 34.0 | 36.5 | **39.0** | 37.0 | 38.0 | 42.0 | **42.5** | 36.0 | **41.5** | 38.0 | 37.5 |
| SEAL-0 | 34.2 | 35.1 | 36.0 | **36.9** | 37.8 | 36.0 | 37.8 | **40.5** | 24.3 | **28.8** | **28.8** | 27.0 |
| XBench-DS | 71.0 | 68.0 | 71.0 | **72.0** | 79.0 | 77.0 | 79.0 | **80.0** | 73.0 | **74.0** | **74.0** | **74.0** |
| BC-ZH | 40.5 | 39.8 | 41.9 | **43.6** | 55.7 | 55.4 | **57.1** | 55.4 | 35.3 | 39.8 | **41.5** | 38.8 |

| | Gemma-4-e2b-it | | | | Gemma-4-e4b-it | | | |
|---|---|---|---|---|---|---|---|---|
| Benchmark | $k$=4 | $k$=8 | $k$=16 | $k$=32 | $k$=4 | $k$=8 | $k$=16 | $k$=32 |
| GAIA | 30.1 | 27.2 | 28.2 | **31.1** | 24.3 | 30.1 | 34.0 | **36.9** |
| BC-200 | **2.5** | 0.5 | 1.0 | 1.0 | 1.0 | **2.5** | 1.5 | 1.5 |
| SEAL-0 | 4.5 | **9.0** | 4.5 | 6.3 | 5.4 | **7.2** | 5.4 | **7.2** |
| XBench-DS | 28.0 | 30.0 | 28.0 | **31.0** | 27.0 | 26.0 | 32.0 | **40.0** |
| BC-ZH | 1.7 | 2.4 | 3.5 | **3.8** | 3.8 | 2.8 | **4.5** | 3.5 |

**Scaling Candidate Width**: Across all agents and benchmarks (Table 5), $k = 32$ falls below the best tested width in 14 of 40 settings, although it achieves the highest accuracy, including ties, on all five benchmarks for Qwen3.5-4B/9B. Optimal widths depend on both agent and task: Qwen3.6-35B-A3B peaks at $k = 16$ on GAIA and $k = 8$ on BrowseComp-200, whereas Gemma-4-e2b-it favors $k = 32$ and $k = 4$, respectively. Table 6 further probes ranking reliability by comparing the estimator's action choice with that of a GPT-5.5 oracle given the ground-truth answer for this diagnostic. Agreement generally decreases with $k$, though not monotonically on BC-200, consistent with greater sensitivity to estimator noise when selecting among more candidates. Together, these results suggest a trade-off between candidate coverage under simulator noise and ranking reliability under estimator noise. Wider search offers more opportunities to find useful actions despite imperfect predictions, without reducing per-call simulator noise, but also increases the risk of selecting overvalued branches, potentially compounded by simulator errors. This trade-off motivates specific width selection rather than simply maximizing $k$.

Table 6: Estimator–oracle agreement (%): how often both select the same candidate action. Agents are Qwen3.5 variants.

| Model | Dataset | $k$=4 | $k$=8 | $k$=16 | $k$=32 |
|---|---|---|---|---|---|
| 397B | GAIA | 25.0 | 21.1 | 11.8 | 10.0 |
| 122B | BC-200 | 40.9 | 21.4 | 17.6 | 22.5 |
| 397B | XBench-DS | 30.8 | 7.9 | 0.0 | 0.0 |

## 6 Conclusion

We presented Seer, an inference-time framework for local guidance in deep research that separates agent proposals from learned tool-response prediction and value estimation. Trained with complementary supervision from a shared trajectory corpus, its simulator and estimator guide different agent models without retraining, using imagined search but only real responses as evidence. Experiments show substantial small-agent gains and transfer across model families and scales, with tool-call costs close to those of Base Agent without search guidance in the measured comparisons. We believe Seer suggests a path toward capable, locally controlled research agents through reusable guidance rather than repeated model-specific adaptation or costly external exploration.

## AI USE STATEMENT

AI assistance was used for language editing, consistency checks, and manuscript presentation. The authors retain full responsibility for reviewing the final manuscript and for all technical claims and conclusions.

## ETHICS STATEMENT

This work uses only publicly available benchmarks and training question sets and involves no human subjects. The method avoids real tool calls on discarded candidate actions by evaluating them through local simulation. Local reasoning does not make external tool use private: selected search queries and page requests are still sent to external services. We are not aware of any conflicts of interest related to this work.

## REPRODUCIBILITY STATEMENT

We describe the training sources, agent models, benchmarks, baselines, and evaluation settings in §5.1. §4.2 and Appendices D and E detail data preparation, supervision, data splits, and decision-model configurations, including the simulator's input templates and the estimator's causal prefix training. Algorithm 1 in Appendix C specifies imagined search and real-action commitment, with resource accounting defined in §3. Both decision-model checkpoints are reused unchanged across agents. We will release the code upon acceptance.

## REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *ICLR*, 2024.

Apple. Introducing Apple's on-device and server foundation models. `https://machinelearning.apple.com/research/introducing-apple-foundation-models`, 2024.

Lei Bai, Zongsheng Cao, Yang Chen, Zhiyao Cui, Shangheng Du, Yue Fan, Shiyang Feng, Zijie Guo, Haonan He, Liang He, et al. Scaling the horizon, not the parameters: Reaching trillion-parameter performance with a 35b agent. *arXiv preprint arXiv:2606.30616*, 2026.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic AI. *arXiv:2506.02153*, 2025.

Hyungjoo Chae, Namyoung Kim, Kai Tzu-iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *ICLR*, 2025.

Haotian Chen, Xin Cong, Shengda Fan, Yuyang Fu, Ziqin Gong, Yaxi Lu, Yishan Li, Boye Niu, Chengjun Pan, Zijun Song, Huadong Wang, Yesai Wu, Yueying Wu, Zihao Xie, Yukun Yan, Zhong Zhang, Yankai Lin, Zhiyuan Liu, and Maosong Sun. AgentCPM-Explore: Realizing long-horizon deep exploration for edge-scale agents. *arXiv:2602.06485*, 2026.

Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, et al. xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations. *arXiv:2506.13651*, 2025a.

Zhaorun Chen, Zhuokai Zhao, Kai Zhang, Bo Liu, Qi Qi, Yifan Wu, Tarun Kalluri, Sara Cao, Yuanhao Xiong, Haibo Tong, Huaxiu Yao, Hengduo Li, Jiacheng Zhu, Xian Li, Dawn Song, Bo Li, Jason Weston, and Dat Huynh. Scaling agent learning via experience synthesis. *arXiv:2511.03773*, 2025b.

Sanjiban Choudhury. Process reward models for LLM agents: Practical framework and directions. *arXiv:2502.10325*, 2025.

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. *arXiv:2602.14234*, 2026.

Dave Citron. Try deep research and our new experimental model in Gemini, your AI assistant. Google. `https://blog.google/products/gemini/google-gemini-deep-research/`, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv:2110.14168*, 2021.

Mingkai Deng, Jinyu Hou, Zhiting Hu, and Eric Xing. General agentic planning through simulative reasoning with world models. *arXiv:2507.23773v3*, 2026.

Hang Ding, Peidong Liu, Junqiao Wang, Ziwei Ji, Meng Cao, Rongzhao Zhang, Lynn Ai, Eric Yang, Tianyu Shi, and Lei Yu. DynaWeb: Model-based reinforcement learning of web agents. *arXiv:2601.22149*, 2026.

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. *arXiv:2603.15594*, 2026.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taiga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop regressing: Training value functions via classification for scalable deep RL. In *ICML*, 2024.

Shiyang Feng, Runmin Ma, Xiangchao Yan, Yue Fan, Yusong Hu, Songtao Huang, Shuaiyu Zhang, Zongsheng Cao, Tianshuo Peng, Jiakang Yuan, et al. InternAgent-1.5: A unified agentic framework for long-horizon autonomous scientific discovery. *arXiv preprint arXiv:2602.08990*, 2026.

Giridhar Ganapavarapu and Dhaval Patel. MCP-Cosmos: World model-augmented agents for complex task execution in MCP environments. *arXiv:2605.09131*, 2026.

Gemma Team. Gemma 4 technical report. *arXiv:2607.02770*, 2026.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your LLM secretly a world model of the internet? model-based planning for web agents. *Transactions on Machine Learning Research*, 2025.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *ICLR*, 2024.

David Ha and Jürgen Schmidhuber. World models. *arXiv:1803.10122*, 2018.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv:2301.04104*, 2023.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv:2305.14992*, 2023.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv:2106.09685*, 2021.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *NeurIPS*, 2019.

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:2503.09516*, 2025.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents. *arXiv:2407.01476*, 2024.

Nicholas Lee, Lutfi Eren Erdogan, Chris Joseph John, Surya Krishnapillai, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. Agentic test-time scaling for WebAgents. *arXiv:2602.12276*, 2026.

Yishan Li, Wentong Chen, Yukun Yan, Mingwei Li, Sen Mei, Xiaorong Wang, Kunpeng Liu, Xin Cong, Shuo Wang, Zhong Zhang, Yaxi Lu, Zhenghao Liu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. AgentCPM-Report: Interleaving drafting and deepening for open-ended deep research. *arXiv:2602.06540*, 2026.

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. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, Pieter Abbeel, Dan Klein, and Anca Dragan. Learning to model the world with language. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 29992–30017, 2024.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for LLM compression and acceleration. *arXiv:2306.00978*, 2023.

Tengxiao Liu, Zifeng Wang, Jin Miao, I-Hung Hsu, Jun Yan, Jiefeng Chen, Rujun Han, Fangyuan Xu, Yanfei Chen, Ke Jiang, Samira Daruki, Yi Liang, William Yang Wang, Tomas Pfister, and Chen-Yu Lee. Budget-aware tool-use enables effective agent scaling, 2026. URL https://arxiv.org/abs/2511.17006.

Rui Lu, Zhenyu Hou, Zihan Wang, Hanchen Zhang, Xiao Liu, Yujiang Li, Shi Feng, Jie Tang, and Yuxiao Dong. DeepDive: Advancing deep search agents with knowledge graphs and multi-turn RL. *arXiv:2509.10446*, 2025.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. *arXiv:2406.06592*, 2024.

Jindi Lv, Aoyu Li, Yuhao Zhou, Zheng Zhu, Xiaofeng Wang, Qing Ye, Yueqi Duan, Wentao Feng, and Jiancheng Lv. Spatial-aware reduction framework: Towards efficient and faithful visual state space models. *arXiv preprint arXiv:2606.19932*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. *arXiv:2303.17651*, 2023.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. *arXiv:2311.12983*, 2023.

MiroMind Team, Song Bai, Lidong Bing, Carson Chen, Guanzheng Chen, Yuntao Chen, Zhe Chen, Ziyi Chen, Jifeng Dai, Xuan Dong, et al. MiroThinker: Pushing the performance boundaries of open-source research agents via model, context, and interactive scaling. *arXiv:2511.11793*, 2025.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv:2112.09332*, 2021.

NVIDIA Corporation. Nemotron-RL-knowledge-web_search-mcqa. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/nvidia/Nemotron-RL-knowledge-web_search-mcqa.

OpenAI. Introducing deep research. https://openai.com/index/introducing-deep-research/, 2025.

Thinh Pham, Nguyen Nguyen, Pratibha Zunjare, Weiyuan Chen, Yu-Min Tseng, and Tu Vu. SealQA: Raising the bar for reasoning in search-augmented language models. *arXiv:2506.01062*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Christopher D. Rosin. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3):203–230, 2011.

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

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588: 604–609, 2020. doi: 10.1038/s41586-020-03051-4.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv:2303.11366*, 2023.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419): 1140–1144, 2018. doi: 10.1126/science.aar6404.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991. doi: 10.1145/122344.122377.

Tongyi DeepResearch Team. Tongyi DeepResearch technical report. *arXiv:2510.24701*, 2025.

Venus Team, Sunhao Dai, Yong Deng, Jinzhen Lin, Yusheng Song, Guoqing Wang, Xiaofeng Wu, Yuqi Zhou, Shuo Yang, Zhenzhe Ying, Zhanwei Zhang, Changhua Meng, and Weiqiang Wang. DR-Venus: Towards frontier edge-scale deep research agents with only 10K open data. *arXiv:2604.19859*, 2026.

Sanidhya Vijayvargiya and Rahul Lokesh. Efficient on-device agents via adaptive context management. *arXiv:2511.03728*, 2025.

Yuxuan Wan, Tianqing Fang, Zaitang Li, Yintong Huo, Wenxuan Wang, Haitao Mi, Dong Yu, and Michael R. Lyu. Inference-time scaling of verification: Self-evolving deep research agents via test-time rubric-guided verification. *arXiv:2601.15808*, 2026.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023.

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. *arXiv:2504.12516*, 2025.

Yu Xia, Jingru Fan, Weize Chen, Siyu Yan, Xin Cong, Zhong Zhang, Yaxi Lu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. AgentRM: Enhancing agent generalization with reward modeling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19277–19290, 2025. doi: 10.18653/v1/2025.acl-long.945.

Guangzhi Xiong, Qiao Jin, Xiao Wang, Yin Fang, Haolin Liu, Yifan Yang, Fangyuan Chen, Zhixing Song, Dengyu Wang, Minjia Zhang, Zhiyong Lu, and Aidong Zhang. Supervising the search process produces reliable and generalizable information-seeking agents. *arXiv:2502.13957v3*, 2026.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv:2305.10601*, 2023a.

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*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.

Weihao Zeng, Keqing He, Chuqiao Kuang, Xiaoguang Li, and Junxian He. Pushing test-time scaling limits of deep search with asymmetric verification. *arXiv:2510.06135*, 2025.

Wenlin Zhang, Xiaopeng Li, Yingyi Zhang, Pengyue Jia, Yichao Wang, Huifeng Guo, Yong Liu, and Xiangyu Zhao. Deep research: A survey of autonomous research agents. *arXiv:2508.12752*, 2025.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv:2504.03160*, 2025.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *ICML*, 2024.

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. *arXiv:2504.19314*, 2025a.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. *arXiv:2307.13854*, 2023.

Yuhao Zhou, Yuxin Tian, Jindi Lv, Mingjia Shi, Yuanxi Li, Qing Ye, Shuhao Zhang, and Jiancheng Lv. Ferret: An efficient online continual learning framework under varying memory constraints. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4850–4861. IEEE, 2025b.

Yuhao Zhou, Jindi Lv, Yuxin Tian, Dan Si, Qing Ye, and Jiancheng Lv. Deploying models to non-participating clients in federated learning without fine-tuning: A hypernetwork-based approach. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 63600–63621, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/677f55fb1c676ba02f8e38ccbe893a71-Paper-Conference.pdf.

Yuhao Zhou, Yunpeng Zhu, Yang Zhou, Jindi Lyu, Jian Lan, Zhangyuan Wang, Dan Si, Thomas Seidl, Qing Ye, and Jiancheng Lyu. ForgeVLA: Federated vision-language-action learning without language annotations. *arXiv preprint arXiv:2605.07474*, 2026b.

Yuxin Zuo, Zikai Xiao, Li Sheng, Fei Huang, Jianhong Tu, Yuxuan Liu, Tianyi Tang, Xiaomeng Hu, Yang Su, Qingfeng Lan, Yantao Liu, Qin Zhu, Yinger Zhang, Bowen Yu, Haiquan Zhao, Haiyang Xu, Jianxin Yang, Jiayang Cheng, Junyang Wang, Lianghao Deng, Mingfeng Xue, Tianyi Bai, Yang Fan, Yubo Ma, Yucheng Li, Zeyu Cui, Zhihai Wang, Zhihui Xie, Zhuorui Ye, An Yang, Dayiheng Liu, Jingren Zhou, and Ning Ding. Qwen-AgentWorld: Language world models for general agents. *arXiv:2606.24597*, 2026.

# APPENDIX CONTENTS


| Section | | Page |
|---|---|---|
| A | Notation | 15 |
| B | Extended Background and Related Work | 16 |
| C | Seer Search Algorithm | 18 |
| D | Tool-Response Simulator Implementation Details | 19 |
| E | Value Estimator Implementation Details | 20 |
| F | Simulator Case Studies: Predicted and Real Tool Responses | 21 |
| G | Case Studies: Action Selection and Real Tool Use | 23 |
| H | Limitations and Future Work | 26 |


## A NOTATION

Table 7 summarizes the notation used in this paper. Note that we distinguish the frozen model's proposal distribution $\pi$ from the agent's search-guided action distribution $\pi_{\mathcal{G}}$, and real observations $o_t$ from predicted observations $\hat{o}^{(i)}$. Lowercase $k$ denotes candidate width, whereas uppercase $K$ denotes the number of value bins.

Table 7: Notation used in the paper.

| Symbol | Meaning |
|---|---|
| **Research interactions and resource constraints** | |
| $q, \mathcal{D}$ | Research request and task distribution, with $q \sim \mathcal{D}$. |
| $\mathcal{G}, \pi_{\mathcal{G}}$ | Research agent and its action distribution, including search guidance. |
| $\pi$ | Action-proposal distribution of the frozen agent model. |
| $t, s_t$ | Real-round index and committed history, initially containing $q$ and agent instructions. |
| $a_t, \hat{a}$ | Selected action (tool use or final submission) and submitted answer, respectively. |
| $o_t, \varnothing$ | Real tool responses and the empty response for final submission. |
| $\mathcal{E}, \oplus$ | Real execution and concatenation in $s_{t+1} = s_t \oplus (a_t, \mathcal{E}(a_t))$. |
| $\mathrm{calls}(a), c$ | Ordered calls in action $a$ and one call (tool name and arguments). |
| $\xi, R_{\mathrm{term}}(\xi; q)$ | Execution record and terminal quality score; binary correctness in the evaluated tasks, with zero for unanswered tasks. |
| $\mathbf{c}(\mathcal{G}, \xi)$ | Resource vector $(c_{\mathrm{param}}(\mathcal{G}), c_{\mathrm{tool}}(\xi))^\top$. |
| $c_{\mathrm{param}}(\mathcal{G})$ | Total number of locally deployed model parameters, including those of the agent model and both decision models in Seer. |
| $c_{\mathrm{tool}}(\xi)$ | All real calls in $\xi$, including those for discarded candidates if executed; batched calls count individually. Seer executes only selected actions. |
| $\mathbf{B}, \preceq$ | Resource-budget vector and componentwise inequality. |
| **Tool-response prediction** | |
| $W_\phi, \phi$ | Tool-response simulator and its trainable parameters. |
| $\widehat{\mathcal{E}}$ | Local response approximation, collecting predictions in call order. |
| $\hat{o}_c$ | Predicted response to an individual call, sampled from $W_\phi(\cdot \mid c)$. |
| $\mathcal{D}_W, N_W$ | Dataset of retained call–response pairs and its size. |
| $c_i, o_i$ | Tool call and observed response in simulator training pair $i$. |
| $\ell, o_\ell, o_{<\ell}, \lvert o \rvert$ | Response-token index, current token, preceding tokens, and response length. |
| $\mathcal{L}_W(\phi)$ | Response-only negative log-likelihood used to train the simulator. |
| **Outcome-supervised value estimation** | |
| $V_\theta, \theta$ | Value estimator over histories and its trainable parameters, including its head. |
| $\mathcal{D}_V$ | Dataset of history prefixes and discounted terminal correctness targets. |
| $q_\xi, s_{\xi,t}, T_\xi$ | Question, prefix at position $t$, and terminal history index of $\xi$. |

Continued on next page

Table 7 (continued)

| Symbol | Meaning |
|---|---|
| $\gamma, G_{\xi,t}$ | Discount factor and target $\gamma^{T_\xi - t} R_{\text{term}}(\xi; q_\xi)$, discounted by remaining interaction steps, not tokens or individual tool calls. |
| $s, G$ | Generic history and scalar target in a value-training pair. |
| $K, b$ | Number of uniform value bins and bin index $b \in \{1, \ldots, K\}$. |
| $\beta_0, \ldots, \beta_K$ | Ordered bin boundaries delimiting the value-head support. |
| $z_b$ | Bin center $(\beta_{b-1} + \beta_b)/2$. |
| $\sigma, \Phi$ | Smoothing standard deviation and standard normal cumulative distribution function. |
| $y_b(G), p_\theta(b \mid s)$ | Normalized Gaussian target mass and predicted probability for bin $b$. |
| $\mathcal{L}_V(\theta)$ | Soft-label cross-entropy; scores are decoded as $V_\theta(s) = \sum_b z_b p_\theta(b \mid s)$. |
| **Imagined search and commitment** | |
| $\bar{s}, \hat{s}^{(i)}$ | Selected-node history and imagined history for candidate $i$. |
| $a^{(i)}, \hat{o}^{(i)}, v^{(i)}$ | Candidate action, predicted responses, and estimated history value. |
| $k, m$ | Candidate width (maximum proposals per expansion) and actual number of valid proposals, with $1 \le m \le k$. |
| $N_{\text{sim}}, n, d_{\max}$ | Search simulations per real round, search-simulation index, and depth cap. |
| $u, e, e^{(i)}$ | Parent node, outgoing edge, and edge corresponding to candidate $i$. |
| $P(e), c_{\text{puct}}$ | Edge prior ($1/m$ for new siblings) and PUCT exploration coefficient. |
| $N(u), N(e), Q(e)$ | Node visits, edge visits, and mean value backed up through an edge. |
| $\text{score}(e), v$ | PUCT selection score and simulation-endpoint value. |
| $e^*, \text{action}(e^*)$ | Most-visited root edge selected for commitment and its action. |

# B Extended Background and Related Work

We begin with the research-agent setting and its resource constraints, then expand the three directions reviewed in §2. Efficient research agents address the cost and capability of the deployed model, while inference-time guidance improves how its proposals are selected. Within inference-time guidance, model-based lookahead addresses a further question: how to evaluate candidate consequences without executing every alternative. These directions are complementary rather than mutually exclusive.

## B.1 Research Agents and Tool Use

Research agents combine a language model's parametric knowledge with information acquired through tools. WebGPT (Nakano et al., 2021) combines browser-assisted question answering with human-feedback training, while ReAct (Yao et al., 2023b) interleaves reasoning and actions so that subsequent decisions incorporate environmental observations. Toolformer (Schick et al., 2023) learns when and how to invoke APIs through self-supervised examples. Together, these approaches establish the interaction pattern underlying research agents: selecting tools, observing their outputs, and using the resulting information to guide subsequent reasoning. InternAgent-1.5 (Feng et al., 2026) extends this pattern to scientific discovery by coordinating generation, verification, and evolution across computational and empirical tasks.

Within this setting, the task determines which observations are useful. WebArena (Zhou et al., 2023) evaluates functional task completion in interactive websites, whereas the benchmarks used in this study assess answers assembled through search and page retrieval (§5.1). Research agents must therefore decide which query or goal-conditioned page visit is likely to reveal useful information. This creates two deployment concerns: the local resources needed to run the model and the external tool calls needed to gather evidence. The following subsections consider improving the deployed model first, then guiding its decisions without updating it.

## B.2 Efficient Research Agents

One approach is to reduce the cost of the deployed model or improve its capabilities through training. AWQ (Lin et al., 2023) reduces weight storage and inference costs through activation-aware quantization. MiniLLM (Gu et al., 2024) distills larger language models into smaller ones, while generalized knowledge distillation (Agarwal et al., 2024) uses teacher feedback on student-

generated sequences to address the mismatch between training and inference. These methods make model compression and capability transfer central to efficient deployment. Beyond language models, STORM (Lv et al., 2026) reduces visual tokens while preserving spatial structure in visual state space models, whereas Ferret (Zhou et al., 2025b) adapts model partitioning and pipeline execution to changing memory budgets during online continual learning. Both illustrate how efficiency also depends on the representation and execution of computation, complementing methods that improve an agent's decisions.

Reinforcement learning further targets the agent's tool-use behavior. Search-R1 (Jin et al., 2025) trains models to interleave reasoning with retrieval using outcome-based rewards, and DeepResearcher (Zheng et al., 2025) optimizes research agents through real web interactions. AgentCPM-Explore (Chen et al., 2026) extends this direction to long-horizon research with an edge-scale model. Alongside reinforcement learning, Agents-A1 (Bai et al., 2026) combines supervised fine-tuning with distillation from domain-specific teachers to strengthen long-horizon capabilities in a 35B model. Although such training can strengthen the proposal model, adapting another backbone requires additional optimization. This motivates the complementary approach considered next: guiding proposals at inference time while keeping the deployed agent frozen. Such guidance does not require training-free components, but reusing them across agents can avoid repeated model-specific adaptation.

Reuse across deployments is also studied in federated learning, although the objects being transferred differ. HyperFedZero (Zhou et al., 2026a) generates models for previously non-participating clients without client-side fine-tuning, while ForgeVLA (Zhou et al., 2026b) learns vision-language-action models from distributed robot experience without centralizing raw data or requiring manual language annotations. These approaches address model deployment and collaborative training across clients, whereas Seer reuses prediction and evaluation models to guide frozen agents at inference time.

### B.3 Inference-Time Guidance

Inference-time guidance can act on completed outputs or intervene at intermediate decisions. This distinction matters for research agents because generating a candidate answer may already require substantial tool use.

**Output-level refinement.** Learned verifiers (Cobbe et al., 2021) score candidate solutions, self-consistency (Wang et al., 2023) aggregates independently sampled reasoning paths, and Self-Refine (Madaan et al., 2023) alternates generation with feedback and revision. For deep research, asymmetric verification (Zeng et al., 2025) allocates a relatively modest verification budget to selecting among expensive search trajectories, while DeepVerifier (Wan et al., 2026) uses test-time rubric-guided verification in an iterative refinement loop. These methods can select an existing answer or inform a subsequent revision, but cannot recover tool calls already spent producing the candidates. For example, the BoN@4 and AV@4 baselines used in our experiments execute all four trajectories before selecting an answer (§5.1). This cost motivates guidance before the next real action, rather than only after a trajectory has been completed.

**Decision-level evaluation.** Tree of Thoughts (Yao et al., 2023a) explores intermediate reasoning states through search and self-evaluation, while Reflexion (Shinn et al., 2023) uses verbal feedback from earlier attempts to guide later behavior. For interactive agents, LATS (Zhou et al., 2024) combines tree search, value estimation, and reflection with environmental feedback, and Koh et al. (2024) perform search in the actual web environment.

**Learning reusable evaluators.** Search determines which alternatives to explore, while evaluation determines which appear promising. Rather than relying only on prompted self-evaluation, a separately trained evaluator can provide this ranking signal. Step-level process supervision (Lightman et al., 2024) trains evaluators from intermediate feedback, and OmegaPRM (Luo et al., 2024) automates process-supervision collection through Monte Carlo search for mathematical reasoning. AgentPRM (Choudhury, 2025) uses Monte Carlo reward targets in an actor–critic framework to train agent policies, rather than restricting guidance to inference time. AgentRM (Xia et al., 2025) learns a reward model for test-time guidance that transfers across tasks and policy models, while RAG-Gym (Xiong et al., 2026) studies transferable search critics for information seeking.

These methods separate the model that proposes actions from the component that evaluates them. However, changing the evaluator does not by itself remove the cost of obtaining candidate observations. Branches explored through real interaction still incur tool calls even when discarded. This remaining cost motivates model-based lookahead, which supplies predicted observations for evaluation before execution.

### B.4 Model-Based Lookahead

Model-based lookahead addresses how candidate consequences are obtained. Instead of executing every alternative, an agent can use a learned or prompted world model to predict aspects of the resulting observation or state. The distinction between using such predictions for policy learning and for inference-time planning clarifies Seer's relationship to this literature.

**Foundations of world modeling.** Dyna (Sutton, 1991) integrates real experience with model-based planning, while Ha & Schmidhuber (2018) learn compact representations of environment dynamics and train a controller within a learned model. MuZero (Schrittwieser et al., 2020) learns latent dynamics together with planning-relevant reward, policy, and value predictions. DreamerV3 (Hafner et al., 2023) learns behavior from imagined trajectories, and Model-Based Policy Optimization (Janner et al., 2019) uses short model-generated rollouts branched from real data to limit the effects of model bias. These approaches establish the use of internal simulation, with different choices of predictive representation and learning objective.

**World models for agent training.** Language-based and synthetic environments extend this idea to learning interactive policies. Dynalang (Lin et al., 2024) learns a multimodal world model that predicts future text and image representations and trains a policy from imagined rollouts. DreamGym (Chen et al., 2025b) generates transitions and feedback for agent reinforcement learning, and DynaWeb (Ding et al., 2026) combines rollouts in a learned web environment with real expert trajectories. Qwen-AgentWorld (Zuo et al., 2026) learns language world models across multiple agent environments and studies their use both as simulators for reinforcement learning and as a foundation for agent training. In these settings, predicted experience supports learning the agent itself.

**Language world models for inference-time planning.** Predictions can also guide a fixed agent without updating its policy. RAP (Hao et al., 2023) repurposes an LLM as both a reasoning agent and a world model, combining predicted transitions with Monte Carlo tree search and task-dependent rewards. For web navigation, WMA (Chae et al., 2025) learns transition-focused natural-language descriptions of changes between observations rather than reproducing full HTML states. Web-Dreamer (Gu et al., 2025) combines simulated action outcomes with value-based selection and trains specialized world models using synthesized interaction data. SiRA (Deng et al., 2026) uses natural-language belief states for goal-oriented simulative planning, while MCP-Cosmos (Ganapavarapu & Patel, 2026) integrates world models with MCP tools to support planning before execution. These methods bring language-based prediction into candidate comparison rather than requiring real interaction for every alternative.

**Relation to Seer.** Seer combines model-based lookahead with a reusable evaluator for frozen research agents. Its simulator can be viewed as a specialized language world model that predicts tool responses from individual calls, rather than modeling the full external environment. The estimator separately scores the resulting histories using discounted terminal correctness from real continuations. These targets reflect observed outcomes and remaining interaction steps, not direct judgments of each action's correctness. Both components are trained offline on real trajectories and reused across agents, rather than updating the agent through imagined experience. This combines call-level prediction with history-level evaluation while keeping action proposal separate. Only selected actions are executed, and only their real responses enter the next committed history. Seer therefore complements model compression and policy training while moving candidate comparison before real tool execution.

## C Seer Search Algorithm

Algorithm 1 describes one round of imagined search followed by real execution, as introduced in §4.3. Search begins at the committed history and explores branches with separate imagined continuations, up to a depth cap $d_{\max}$. During each simulation, selection follows Eq. (12), and the estimator scores the endpoint to assess its potential for task completion. This value is then backed up along the

**Algorithm 1** Seer: imagined search and real execution (one round)

**Require:** nonterminal committed history $s_t$; frozen proposal distribution $\pi$; models $W_\phi, V_\theta$; $N_{\text{sim}}, k, d_{\max} \geq 1$; $c_{\text{puct}} \geq 0$

Initialize root $s_t$ at depth zero; new nodes have $N = 0$ and new edges have $N = Q = 0$
**for** $n = 1$ **to** $N_{\text{sim}}$ **do**
  Select a path to a terminal, unexpanded, or depth-capped node with history $\bar{s}$ using Eq. (12)
  **if** the node is nonterminal, unexpanded, and below $d_{\max}$ **then**
    Independently sample $m$ valid actions $\{a^{(i)}\}_{i=1}^{m}$ from $\pi(\cdot \mid \bar{s})$, with $1 \leq m \leq k$
    Batch their tool calls through $W_\phi$ to obtain $\hat{o}^{(i)}$ in call order; use $\varnothing$ for final-answer candidates
    Add children $\hat{s}^{(i)} = \bar{s} \oplus (a^{(i)}, \hat{o}^{(i)})$ with edge priors $P(e^{(i)}) = 1/m$; mark final-answer children terminal
    Extend the path through the first new edge
  **end if**
  $v \leftarrow V_\theta$(history at path endpoint), including terminal histories
  For each path edge: $N(e) \leftarrow N(e) + 1$, $Q(e) \leftarrow Q(e) + (v - Q(e))/N(e)$
  Increment visit counts of all path nodes, including the root
**end for**
Select $a_t$ by Eq. (13)
**if** $a_t$ is a final-answer submission **then**
  Return the answer and terminate
**else**
  Execute only $a_t$'s tool calls to obtain $o_t = \mathcal{E}(a_t)$
  Discard the imagined tree; return the real history $s_{t+1} = s_t \oplus (a_t, o_t)$ for the next round
**end if**

traversed path, updating the mean value of each edge. After search, the action on the most visited edge from the root is selected for real execution according to Eq. (13).

Note that no gold-answer judgment is queried during search, including for terminal candidates. After a tool-using action is selected, its actual responses replace the predicted continuation rather than being appended to it. Thus, each subsequent round starts from a history containing only committed actions and real tool observations.

# D Tool-Response Simulator Implementation Details

The simulator $W_\phi$ approximates the information returned by the shared tools, rather than the full state of the web. Its modeling unit is an individual call $c$: a tool name together with its arguments. As in Eq. (6), predictions for a multi-call action are assembled in the original call order. This factorization makes response prediction independent of the agent architecture while preserving the format expected by the value estimator.

## D.1 Supervision from Observed Tool Responses

The real trajectories collected in §4.2 provide call–response supervision for both `search` and `visit`. Search outputs contain titles, links, and snippets; visit outputs contain page content extracted for the specified retrieval goal. We normalize equivalent call representations, remove infrastructure exceptions, and deduplicate repeated call–response pairs. Empty results and inaccessible pages are represented consistently, while substantive returned content is retained. These steps reduce variation caused by the collection infrastructure without filtering examples according to whether the final answer was correct. An unsuccessful research trajectory can still supply a valid observation of what a tool returns.

The resulting simulator corpus contains 0.7M pairs, as reported in §5.1. Validation is separated by normalized call identity so that repeated observations of the same call do not appear on both sides of the split. This differs from the estimator's question-disjoint split: the simulator learns a call-level conditional distribution, whereas the estimator learns to assess question-dependent histories.

### D.2 Tool-Conditioned Prediction

The simulator uses the same tool-specific input templates during training and inference. The boxed templates below make the conditioning information explicit. The query, URL, and goal are supplied by the proposed tool call; the enclosing conversation and the identity of the proposing model are not additional inputs. For a page visit, conditioning on the goal lets the simulator predict a focused extraction rather than an undirected reconstruction of the page.

**Search simulator prompt**

```
Act as a web search engine simulator.  Return the top search results
for the query below.

Query:  {query}
```

**Page-visit simulator prompt**

```
Act as a webpage extraction tool simulator.  Extract content from the
URL that is relevant to the goal.

URL: {url}
Goal:  {goal}
```

Training minimizes the response-only negative log-likelihood in §4.2. The instruction and call arguments supply context but do not contribute target tokens to the loss. This objective learns the conditional form and content of tool responses, rather than training the simulator to choose actions or judge whether a research question has been solved.

### D.3 Configuration and Use in Search

Following §5.1, we initialize the simulator from Qwen3.5-9B and train it with LoRA rank 64, a learning rate of $10^{-4}$, and a context limit of 16k tokens. Each training example must retain response tokens within this limit, so examples that leave no tokens for response supervision are excluded. After training, the same checkpoint is reused across agent models without further updates.

At inference, candidate calls are grouped for simulation, and each predicted response is attached only to its corresponding imagined branch. Proposals that submit a final answer contain no tool calls and therefore require no simulator output. These predictions support candidate comparison but do not serve as evidence. Once an action is selected, its tool calls are executed in the real environment, and only the resulting observations enter the next root history. This procedure allows comparison before execution while keeping subsequent reasoning grounded in real tool responses.

## E Value Estimator Implementation Details

The estimator $V_\theta$ complements call-level prediction by assessing a history in the context of its research question. Its supervision associates intermediate histories with observed task outcomes and their distance from completion. It provides an outcome-supervised scalar ranking signal for candidate histories, rather than predicting tool responses or generating textual critiques.

### E.1 History-Level Targets

We retain deduplicated trajectories with submitted answers and exclude unanswered runs that end in errors or timeouts, as in §4.2. A judge compares each submitted answer with its gold answer to assign binary terminal correctness. The gold answer and judge output are used only to construct labels; neither is appended to the estimator's input history. For a retained trajectory $\xi$, each labeled prefix receives

$$G_{\xi,t} = \gamma^{T_\xi - t} R_{\text{term}}(\xi; q_\xi), \qquad \gamma = 0.95. \tag{15}$$

The distance $T_\xi - t$ counts remaining interaction steps, not tokens or individual calls within an action. Correct trajectories have positive targets that approach one at completion, whereas incorrect submitted

answers induce zero targets throughout their trajectories. Excluding unfinished runs from this training set does not change the evaluation convention that unanswered questions receive a terminal quality score of zero.

Discounting distinguishes prefixes associated with successful completion at different remaining distances without requiring manual step-by-step annotation. However, each target describes an observed continuation from the shared collection corpus; it does not assert that the immediately preceding action was individually correct. Since the corpus contains continuations from different frozen models, the learned score reflects this supervision rather than representing a calibrated success probability for a particular deployment policy. These semantics are preserved when the estimator ranks imagined histories at inference.

### E.2 Gaussian-Smoothed Value Classification

We replace the language-modeling head with a fully trainable HL-Gauss classification head (Farebrother et al., 2024). As specified in §5.1, the head has $K = 201$ uniform bins over $[-1, 1]$. Their width is $\beta_b - \beta_{b-1} = 2/201$, and the smoothing standard deviation is $\sigma = 0.75(2/201)$. Although the supervision targets lie in $[0, 1]$, the classification support is the wider interval $[-1, 1]$; the support should not be confused with the range of observed labels.

For each target, Eq. (10) integrates a Gaussian over the bins and normalizes its mass over the bounded support. This gives neighboring bins similar supervision for nearby target values, rather than treating two targets on opposite sides of a bin boundary as unrelated classes. Training minimizes soft-label cross-entropy, and inference returns the expectation of bin centers under the predicted distribution, as in §4.2. The scalar expectation is the value used for search and backup; no text-generation prompt or gold-answer comparison is required to obtain it.

### E.3 Causal Prefix Training and Shared Deployment

The estimator corpus contains 1M training samples derived from the real trajectories (§5.1). We split training and validation by question, keeping every trajectory and labeled prefix for the same question together. This prevents closely related continuations from leaking across the split. Each trajectory is processed in one causal forward pass, with supervision read at the endpoints of its labeled prefixes. An endpoint representation can attend only to its own history, so later actions and the final submission are unavailable when scoring an earlier prefix.

Shared-prefix training requires each shorter history to have exactly the same tokens as the corresponding prefix of the full trajectory. We exclude trajectories beyond the context limit rather than changing the evidence available to earlier prefixes through truncation. These conditions preserve the intended history–target pairing while avoiding repeated encoding of shared context.

The estimator is initialized from Qwen3.5-9B and uses LoRA rank 32, a learning rate of $10^{-4}$, and a 128k-token context limit, together with the fully trainable classification head. Its checkpoint is reused unchanged across agents. At deployment, it scores the history formed by appending a candidate action and its predicted responses; final-answer candidates are scored with an empty response. The experiments use depth-one imagined search, so each candidate extends a real root by only one imagined action before evaluation.

## F Simulator Case Studies: Predicted and Real Tool Responses

We compare predictions from the tool-response simulator $W_\phi$ with the corresponding real tool responses. Unlike the trajectory comparisons in Appendix G, these cases examine the content supplied to the value estimator, not final-answer accuracy or realized tool savings. The shared interface has two tools: `search` returns result lists, and `visit` retrieves pages and extracts goal-relevant content (§5.1). Page reading is therefore covered by `visit`, not a separate `read` tool, for which no examples are available here.

The boxes distinguish call arguments, predicted responses, and real responses. Queries and outputs are condensed where indicated, and descriptive summaries are not verbatim tool logs. Any research context is provided for the reader: $W_\phi$ receives only the call and its tool-specific instruction, not the

enclosing conversation (Appendix D). We examine agreement and discrepancies in response content rather than treating textual similarity as a measure of factual correctness. These comparisons provide a qualitative analysis rather than an estimate of average prediction accuracy.

### F.1 Predicting Search Results

**Simulator Case A: A specific historical search result**

**Call.** `search`, with query: `hatpin "If Men Use Their Bare Fists to Fight Life's Battles" 1919 advertisement`. The investigation seeks the wording of an advertisement for the 1919 film *Bare Fists*.

**Predicted response.** The result list contains *The Hat-Pin: Fashionably Dangerous*, with URL `https://femmefashionforward.wordpress.com/2020/09/28/the-hat-pin-fashionably-dangerous/`. Its snippet connects the film advertisement to the phrase "If Men Use Their Bare Fists to Fight Life's Battles".

**Real response.** The returned hit has the same title, URL, and answer-relevant snippet linking the phrase to the 1919 advertisement.

**Interpretation.** The match preserves a concrete retrieval target and the relation among the film, date, and wording, rather than merely predicting a historical topic. This is the information an imagined continuation can expose to $V_\theta$ when comparing candidate actions. It does not itself verify the phrase in the committed history: a selected search must still be executed, and its real response supplies the evidence.

**Simulator Case B: Preserving a useful result for a noisy query**

**Call (abbreviated).** `search`, with a long query combining `"Different Tempo" rooster hamster animation distinct song versions composer` and additional terms. The research goal is to connect animated song performances to their musical source. The displayed query preserves the main terms, not the full original argument.

**Predicted response (summary).** A relevant hit links the hamster performance to *The Hampsterdance Song* and the melody *Whistle-Stop* from Disney's 1973 *Robin Hood*.

**Real response (summary).** The corresponding hit supports the same musical connection rather than redirecting the investigation to a different dominant topic.

**Interpretation.** The supplied response pair preserves a useful association despite distracting query terms. Unlike Case A, the available excerpt does not reproduce an exact result title or URL, so the visible comparison supports agreement on the research direction rather than an independently checkable, character-level match. It does not show that the simulator corrected the query or recovered the final composer answer.

**Simulator Case C: Matching a timeout rather than inventing content**

**Call (summary).** `search`, with an oversized query about the 2024 price range of Shanghai Gold Exchange Au(T+D), expanded with financial and unrelated technical terms.

**Predicted response (summary).** A Google search timeout for the expanded query, rather than a list of market results.

**Real response (summary).** The actual request also returns a Google search timeout.

**Interpretation.** Both responses indicate an absence of retrieved evidence, not a fact about gold prices. Such a prediction could inform candidate-history evaluation, but this pair neither establishes that query length caused the timeout nor demonstrates reliable prediction of transient tool failures. The training pipeline excludes infrastructure failures (§4.2), so this diagnostic example should not be read as evidence that timeout messages were retained as training targets. No downstream reformulation or avoided call is established by the response pair alone.

### F.2 Predicting Goal-Conditioned Page Extractions

**Simulator Case D: Preserving the subject of a biographical fact**

**Call.** `visit`, with URL `https://www.dailymail.com/news/article-9925789/Elon-Musks-secret-sister-tells-Errol-Musk-turned-asked-DNA-test.html`. **Goal:** Find how old Riette Nel's mother was when she gave birth to her.

**Predicted response (condensed).** Riette Nel was born as Mariette van Deventer in Ladysmith, South Africa, on August 8, 1975. Her birth certificate did not name a father, and her mother was 21 at her birth. The predicted summary answers the goal with *21 years old*.

**Real response (condensed).** The extraction reports the same birth name, date, place, and birth-certificate detail, and states that her mother was 21.

**Interpretation.** The response pair agrees on both the requested age and the context identifying whose mother and birth are being discussed. This illustrates goal-conditioned prediction of an answer-relevant passage, not reconstruction of the entire article or independent validation of every claim it contains. The predicted extraction remains part of an imagined history until the selected call obtains a real response.

**Simulator Case E: Retaining an exact scientific award identifier**

**Call.** `visit`, with URL `https://iopscience.iop.org/article/10.3847/2041-8213/acd54b`. **Goal:** Find the NASA award number supporting work by R. G. Arendt.

**Predicted response (condensed).** Work by R.G.A. was supported by NASA under award number `80GSFC21M0002`. The surrounding acknowledgement also identifies the South African Radio Astronomy Observatory as the operator of MeerKAT and a facility of the National Research Foundation. The summary repeats `80GSFC21M0002` as the requested award number.

**Real response (condensed).** The extraction contains the identical award string and the same surrounding acknowledgement context.

**Interpretation.** Here, usefulness depends on preserving an exact alphanumeric identifier and its association with the author, not merely recognizing that the paper acknowledges NASA support. The agreement shows that the predicted extraction retains the requested field despite nearby institutional information.

**Simulator Case F: Textual agreement does not guarantee layout fidelity**

**Call.** `visit`, with URL `https://lindas-literary-home.com/my-literary-home/essays-poems-with-commentaries/audre-lordes-father-son-and-holy-ghost/`. **Goal:** Retrieve the text of Audre Lorde's *Father Son and Holy Ghost*, including indentation, to identify which stanza has indented lines.

**Predicted response (summary).** The extraction reproduces the poem's line sequence and includes leading spaces before lines such as "Not that his judgment eyes" and "have been forgotten". Its summary describes the text as preserving the poem's structure and line breaks.

**Real response (summary).** The extraction contains the same line sequence but normalizes away the displayed leading spaces, while describing the stanza and line structure as preserved.

**Interpretation.** The pair agrees closely on textual content but differs on the field required by this particular goal: indentation. A statement that structure is preserved cannot substitute for an actual layout check. Neither high textual similarity nor leading spaces in a prediction establish the original page's typography. Layout-sensitive questions therefore require verification against a real source representation that retains formatting, rather than relying on the imagined extraction or a whitespace-normalized response alone.

## G Case Studies: Action Selection and Real Tool Use

We examine six cases of Seer's search guidance and the resulting committed research trajectories. The first four contrast a correct Seer answer with unsuccessful Base Agent, BoN@4, and AV@4 runs; the final two compare successful runs with different real tool costs. All Seer runs use candidate width $k = 32$ and depth cap $d_{\max} = 1$. Questions and trajectory excerpts are condensed for

readability, rather than reproduced as verbatim tool logs. These qualitative comparisons complement the aggregate results in §5.2.

Table 8 summarizes the outcomes and resource use. We count $c_{\text{tool}}(\xi)$ as in §3: every real `search` or `visit` call counts, including calls in discarded BoN@4 and AV@4 trajectories, and calls batched in one action count individually. Final-answer submissions contribute no tool calls. For Seer, $W_\phi$ predicts candidate tool responses and $V_\theta$ scores the resulting imagined histories before search selects an action for real execution. These local computations are not real tool calls, but they are not free computation either.

Table 8: Case outcomes and real tool-call counts. C denotes a correct answer; I denotes an incorrect or unanswered result; — indicates a comparison not included in the case report. Base, BoN, AV, and $S_{32}$ follow the notation in §5.1.

| Case | Base | BoN | AV | $S_{32}$ |
|---|---|---|---|---|
| 1. OpenCV contributor | 11 (I) | 39 (I) | 50 (I) | **3 (C)** |
| 2. Hiking location | 54 (I) | 178 (I) | 124 (I) | **17 (C)** |
| 3. UN parliamentary seats | 40 (I) | 135 (I) | 104 (I) | **16 (C)** |
| 4. NBA final-game margin | 14 (I) | 62 (I) | 47 (I) | **8 (C)** |
| 5. Encoder-layer comparison | 11 (C) | — | — | **2 (C)** |
| 6. Elevator-accident article | — | 21 (C) | — | **2 (C)** |

### G.1 Correct Answers Through More Focused Trajectories

**Case 1: Turning a retrieved source into a targeted identity check**

**Setting.** GAIA; Qwen3.5-9B.

**Question.** Which contributor to the OpenCV version that added Mask-RCNN support shares a Latin-transliterated name with a former Chinese head of government? **Labeled answer:** Li Peng.

**Seer's committed trajectory: 3 calls (2 searches, 1 visit).** A search for OpenCV's addition of Mask-RCNN support returns the ChangeLog. A visit identifies the relevant version, 4.0.0, and a contributor list containing Li Peng. One targeted search for Li Peng as a former Chinese premier checks the identity match, after which Seer submits *Li Peng*.

**Comparison.** Base Agent reaches the same ChangeLog, but subsequent queries accumulate contributor names and loosely related keywords rather than checking a specific candidate. It uses 9 searches and 2 visits without a correct submission. BoN@4 and AV@4 also fail, using 39 and 50 real calls, respectively, with similar query deterioration in their trajectories.

**Interpretation.** The useful source is available to both Seer and Base Agent; the difference appears in what they do next. Seer's trajectory preserves the link between the release, its contributors, and the identity clue. This illustrates the role of next-action guidance after retrieval, rather than an advantage attributable only to finding a better initial source. The excerpts do not reveal which unexecuted candidates received lower estimator scores.

**Case 2: Revising a location hypothesis as constraints accumulate**

**Setting.** BrowseComp-200; Qwen3.5-397B-A17B.

**Question.** Identify a scenic hiking location formed during the last ice age, with a body-part-related name, a fall from its viewpoint between 2000 and 2021, and a 2014 search-and-rescue count related to the Grand Canyon's 2022 count by a factor of 84.5. **Labeled answer:** Trolltunga.

**Seer's committed trajectory: 17 searches.** Early searches consider directions including Angels Landing, the Grand Canyon, and Finger Lakes without establishing a joint match. A subsequent search connects Trolltunga with the tongue-related name and glacial formation clues. Further searches check a 2015 fatal fall and the rescue-statistics clue, leading to the answer *Trolltunga*.

**Comparison.** Base Agent spends 54 calls exploring partially matching locations and repeatedly searching the rescue-count ratio, then submits *Half Dome*. BoN@4 and AV@4 use 178 and 124 calls, respectively, without reaching a correct answer.

**Interpretation.** Seer's visible trajectory changes direction when another location better fits multiple constraints, whereas Base Agent ultimately settles on a partial match. This behavior is consistent with comparing alternative next actions before commitment. It does not imply that Seer retains a persistent tree of location hypotheses: at depth one, each real round starts a new imagined search from the committed history. The example shows that a large frozen agent can also benefit from guidance over its proposals.

**Case 3: Refining a data lookup rather than rejecting its premise**

**Setting.** SEAL-0; Qwen3.5-397B-A17B.

**Question.** According to United Nations data from 2015, what percentage of national parliamentary seats in Northern America were held by women? **Labeled answer:** 21.8%.

**Seer's committed trajectory: 16 calls (9 searches, 7 visits).** Initial searches locate UN, IPU, and UNSD material. Seer moves from broader regional presentations, including *The World's Women 2015*, to a UNdata table containing the Northern America row. A secondary data presentation supplies a cross-check of the 2015 value, and Seer submits *21.8%*.

**Comparison.** Base Agent encounters the same source ecosystem but questions whether Northern America is reported separately, eventually giving an approximate 22–23% range after 40 calls. BoN@4 uses 135 calls and returns approximately 27–28% with a premise objection; AV@4 uses 104 calls without supplying the exact labeled figure. Their searches repeatedly return to broader regional groupings rather than resolving the requested row.

**Interpretation.** Failing to locate a row in an initial document is not evidence that the requested statistic is unavailable. Seer's committed actions continue refining the lookup within the relevant data sources instead of substituting an approximate answer or a premise objection. The contrast concerns action selection under incomplete retrieval, not privileged access to the labeled answer during search.

**Case 4: Preserving the two-hop structure of a sports lookup**

**Setting.** XBench-DeepSearch; Qwen3.5-9B.

**Question.** For the team of the player ranked second in total fouls in the 2023–24 NBA regular season, what was the score difference in its final game that season, including the play-in and playoffs? **Labeled answer:** 7 points.

**Seer's committed trajectory: 8 calls (6 searches, 2 visits).** Seer searches the regular-season total-foul leaderboard and visits Basketball-Reference to identify Domantas Sabonis and the Sacramento Kings. It then searches the Kings' final game and visits a game or season page to resolve the relevant score difference. The submitted answer is *7 points*.

**Comparison.** Base Agent initially finds the relevant basketball source, but later queries mix unrelated names and malformed keyword sequences; timeouts also occur. It ends without an answer after 14 calls. BoN@4 spends 62 calls and recovers part of the chain without a correct final difference, while AV@4 spends 47 calls and submits $-10$.

**Interpretation.** Seer's trajectory keeps the player-to-team and team-to-game links explicit rather than repeatedly reopening inconsistent combinations. Together with Case 1, this provides a small-agent example of more focused committed queries. Because the unsuccessful run also contains tool failures, the difference cannot be attributed exclusively to query drift or to any single decision-model component.

### G.2 Fewer Real Calls at the Same Answer Quality

**Case 5: Submitting once the comparison is supported**

**Setting.** GAIA; Qwen3.5-4B.

**Question.** How many more encoder blocks or layers does BERT-base have than the encoder in *Attention Is All You Need*? **Labeled answer:** 6.

**Seer's committed trajectory: 2 searches, 3 rounds.** One search establishes that BERT-base has 12 encoder layers, and another establishes that the original Transformer encoder has 6. Seer then submits $12 - 6 = 6$; the submission is a committed round but not a tool call.

**Comparison.** Base Agent also obtains the two layer counts early, but continues checking arXiv, Wikipedia, Hugging Face, and course material before returning the same answer. It uses 5 searches and 6 visits over 12 rounds, for 11 real calls in total. Thus, its real-call count is $5.5\times$ Seer's despite identical final correctness.

**Interpretation.** The visible difference is the timing of final submission, not the arithmetic. In Seer, a final-answer action is evaluated by $V_\theta$ with an empty tool response and competes with further tool-using actions (§4.1). This supplies a mechanism for selecting submission once the history supports an answer, without a separate stopping threshold. The example is consistent with this mechanism, but does not by itself establish that the estimator is calibrated to evidence sufficiency.

**Case 6: Reaching the same evidence without executing four trajectories**

**Setting.** BrowseComp-200; Qwen3.5-397B-A17B.

**Question.** A machine used beneath the Colosseum later developed into steam-powered and electric passenger versions. In a 2010s article about accidents involving this machine, who is the sibling of a person catastrophically injured on Thanksgiving? **Labeled answer:** Josh / Josh Nelson.

**Seer's committed trajectory: 2 calls (1 search, 1 visit).** A search combining the historical-machine and Thanksgiving-injury clues returns a CBS News article about in-home elevator accidents. A visit identifies the injured person as Jordan Nelson and the sibling as Josh Nelson. Seer submits *Josh*.

**Comparison.** BoN@4 also selects the correct answer through the same essential evidence chain, but its four real trajectories incur 11 searches and 10 visits in aggregate. Its 21 real calls are $10.5\times$ Seer's count.

**Interpretation.** Here the successful evidence chain is short for both methods; the accounting difference includes the real execution of alternative BoN@4 trajectories. Seer instead uses the simulator and estimator to guide depth-one imagined search, executing only the selected action at each round. Candidate width $k = 32$ is a maximum number of next-action proposals per expansion, not 32 complete simulated research trajectories. The comparison establishes lower real tool use in this case, not matched local computation or a $10.5\times$ latency speedup.

## H Limitations and Future Work

Seer uses the same simulator and estimator to guide different frozen agents through shared search and page retrieval tools. This design provides a basis for extending the supported tasks and improving the use of local computation, although both directions require further evaluation.

**From question answering to research reports.** We focus on question answering because terminal correctness provides a clear training signal and a consistent basis for comparing agents. However, correctness alone does not capture the quality of a full research report, where coverage, citation support, and synthesis also matter. Extending Seer to such tasks would therefore require richer supervision for the estimator, potentially using rubrics that reflect the user's requirements. This would preserve the separation between proposal, prediction, and evaluation while broadening what the estimator learns to assess.

**Adaptive allocation of local computation.** We use fixed candidate widths and a search depth of one to compare the effect of candidate coverage under a common search procedure. This controlled setting leaves open how much search each decision needs. Future work could vary the candidate width and simulation effort according to decision uncertainty and the remaining budget, building on adaptive agent scaling (Lee et al., 2026; Liu et al., 2026). The aim would be to spend more computation where additional candidates are useful and less where they add little. Such a policy should choose its budget before observing the outcome and be evaluated under the same resource constraints as fixed search settings.

**Efficient deployment on local devices.** By accounting for model parameters and real tool calls separately, we study the cost of guidance without tying the comparison to a particular device. These measures do not, however, establish its peak memory use, energy consumption, or latency on actual hardware. Quantizing the agent and both decision models, sharing representations between prediction and evaluation, and tailoring batching to the device could reduce the added local cost. Evaluating these changes on target devices would show whether the savings preserve decision quality and translate into practical deployment benefits.

Progress along these directions could make reusable guidance practical for a wider range of research tasks and devices, accelerating broader access to research assistance while preserving users' local control over their agents' reasoning and decisions.