VISTA: LEARNING WIDE SEARCH AGENTS WITH VERIFIABLE INTERMEDIATE STATES AND TURN-LEVEL CREDIT ASSIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Wide search requires agents to explore many target items, track which ones they have covered, and combine their findings. Yet existing systems largely rely on designed multi-agent systems. Training capable open-source wide-search agents is challenging because reinforcement learning with verifiable rewards (RLVR) typically assigns credit to an entire trajectory, making it difficult to identify the few decisions that matter most in long interactions. We propose **VISTA** (**V**erifiable **I**ntermediate **S**tates for **T**urn-level **A**ttribution), a reinforcement learning framework that transforms sparse outcome supervision into fine-grained turn-level credit. Our key insight is to make otherwise latent search progress explicit and verifiable through intermediate table states that track the agent's accumulated discoveries. VISTA uses changes across consecutive states to attribute credit to the responsible turns, while preserving the terminal outcome as the dominant learning signal. Across six benchmarks and seven models, ranging from 4B to 27B, VISTA consistently outperforms outcome-only GRPO and step-wise RL baselines. On WideSearch, VISTA improves Item F1 by 13.84% relative over outcome-only GRPO on Qwen3-4B, with consistent gains across other model families and scales, and further generalizes to standard multi-hop QA benchmarks. Extensive ablations show that the gains come not merely from adding process supervision, but from making intermediate progress verifiable and assigning credit to the responsible turns. These results highlight that where and how credit is assigned is critical for training long-horizon search agents.

1 INTRODUCTION

Large Language Models (LLMs) are transforming web search from simple information retrieval (Gao et al., 2023) into autonomous, agentic information seeking (Luo et al., 2026; Gao et al., 2025; Wang et al., 2025; Liu et al., 2026). Modern search agents can iteratively reason, retrieve evidence, and synthesize information over long interaction horizons, enabling increasingly complex tasks such as deep research (Mialon et al., 2024; Team et al., 2025b; OpenAI, 2025). Accordingly, substantial recent effort has focused on improving agents for deep search, where success requires multi-step reasoning, evidence refinement, and long chains of information seeking (Jin et al., 2025; Wang et al., 2025; Xie et al., 2026). Many real-world information-seeking tasks require not only searching deeply, but also searching widely (Lan et al., 2025; Zhu et al., 2026). Wide-search tasks ask agents to discover, verify, and organize a large number of target items collected across the Web. Recent benchmarks such as WideSearch (Wong et al., 2026) expose substantial weaknesses of current agents in this setting (Lee et al., 2026), including incomplete planning, insufficient coverage, and ineffective use of retrieved evidence. Recent systems improve wide-search performance through structured planning (Huang et al., 2026), parallel execution (Lan et al., 2026), and multi-agent decomposition (Zhang et al., 2026b). However, many of the strongest systems rely on frontier proprietary models or inference-time orchestration, leaving an important question largely open: *how can we train open models to acquire effective wide-search behavior?*

This problem is challenging because the learning signal required by wide search differs from that of conventional deep search. Deep-search agents typically refine a small set of hypotheses or clues through iterative reasoning (Li et al., 2025; Xie et al., 2026). In contrast, wide-search agents must explore many targets while continuously tracking what they have found, what remains missing, and how to synthesize the collected evidence. Recent reinforcement learning (RL) methods have substantially improved search and research agents (Wang et al., 2025; Xie et al., 2026; Tao et al., 2026), while initial efforts have also explored RL for broad information seeking (Tao et al., 2025; Xu et al., 2026). However, these methods predominantly optimize trajectory-level outcomes for wide search task. For a long trajectory consisting of many search and reasoning turns, a single terminal reward provides little information about which decisions expanded useful coverage and which were redundant or harmful. As the interaction horizon grows, this coarse credit assignment makes it increasingly difficult to learn the behaviors that wide search requires.

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.

To address these challenges, we propose VISTA, an RL framework that enables fine-grained credit assignment for long-horizon wide-search agents through verifiable intermediate states and terminal-dominant turn-level attribution. A key insight is that, although progress along a long search trajectory is otherwise latent, wide-search tasks naturally admit a structured representation of partial progress. Specifically, VISTA maintains an intermediate answer table that records the agent's findings throughout the trajectory. These tables serve as task-grounded checkpoints: changes across consecutive states reveal whether a search turn discovers new information, corrects existing entries, or makes little useful progress. Building on these intermediate states, VISTA performs turn-level credit assignment based on the progress made at each turn. Importantly, we do not directly optimize intermediate-state rewards. Doing so can over-reward locally useful search behavior while weakening the final synthesis objective. Instead, VISTA uses intermediate progress to redistribute the credit across the responsible turns, while a terminal-dominant constraint ensures that the final outcome remains the primary learning signal guided by the diagnosis study in Appendix B.2. Together, these designs allow the agent to learn not only whether a trajectory succeeds, but also which search decisions contribute to that success.

We evaluate VISTA across three wide-search benchmarks, three model families, and four model scales. VISTA consistently outperforms outcome-only GRPO and strong search-agent baselines, improving both item discovery and structured answer quality. The gains also extend beyond wide search: models trained on wide-search data transfer to multi-hop QA, and applying VISTA to deep-search training data shows promising gains on the corresponding multi-hop QA tasks. Ablations further show that these gains depend on an explicit intermediate state and credit assignment to the responsible turns, while preserving the dominance of the terminal objective.

Our contributions are summarized as follows:

- We formulate **explicit and verifiable intermediate states** as task-grounded checkpoints that expose otherwise latent progress in long-horizon wide search.
- We introduce a **terminal-dominant turn-level credit attribution** method that attributes trajectory-level learning signals according to intermediate state changes while preserving the terminal task objective.
- We develop VISTA, a trainable framework for wide-search agents that improves both information discovery and final synthesis, consistently outperforming strong baselines across multiple benchmarks, model families, and scales.
- Through extensive ablations and training analysis, we show how intermediate-state design and credit assignment jointly determine the search behaviors learned by the policy.

2 Related Work

Wide Search Agents and Systems. LLM-based information-seeking agents interleave reasoning, retrieval, and evidence synthesis to acquire knowledge beyond their parametric memory (Jin et al., 2025; Song et al., 2025; Li et al., 2025; Wu et al., 2026). While most prior work focuses on deep search over a set of evidence chains or anchor entities (Wang et al., 2025; Luo et al., 2026), wide search requires collecting many atomic facts across entities and organizing them into a complete structured output (Wong et al., 2026; Lan et al., 2025; Zhu et al., 2026). Training-based approaches improve this capability through entity-rich data (Tao et al., 2025; Du et al., 2026) or multi-agent reinforcement learning (Xu et al., 2026), whereas training-free systems rely on manually designed decomposition, parallel execution, and shared-state management (Lan et al., 2026; Lee et al., 2026; Huang et al., 2026; Zhang et al., 2026b; Hu et al., 2026). In contrast, VISTA trains a single long-horizon policy by using verified scores in its evolving answer table to attribute credit to individual search turns.

Fine-Grained Credit Assignment for Agentic RL. Outcome-supervised RL commonly assigns the same trajectory-level advantage to all policy-generated tokens, making it difficult to distinguish useful actions from redundant or harmful ones in long-horizon interactions (Feng et al., 2026). Recent methods derive finer-grained credit from policy's own intrinsic beliefs (Auzina et al., 2026), teacher- or reference-model likelihoods (Xie et al., 2026; Tao et al., 2026), LLM-based critics (Zhang et al., 2026a; Wang et al., 2026b; Liu et al., 2026), privileged self-distillation (Wang et al., 2026c;d), or environmental feedback (Huo et al., 2026). However, model-inferred credit does not necessarily measure objective task progress: counterfactual-replay analysis finds limited causal fidelity for commonly used LLM-judge, confidence-based, and likelihood-based signals (Zhang, 2026). CW-GRPO (Wang et al., 2026b) uses an LLM judge to estimate each search round's contribution and reweights its trajectory-level GRPO advantage. In wide search, VISTA instead scores the evolving answer table and assigns its progress signal to the responsible turn, while keeping the final outcome dominant.

Explicit State and Verifiable Process Supervision. Inference-time methods also track intermediate progress: Reflexion stores feedback and self-reflections across attempts (Shinn et al., 2023), while LATS evaluates candidate states

to guide exploration (Zhou et al., 2023). Harness-1 (Jiang et al., 2026) maintains an explicit environment-side search state, but does not use state changes for turn-level training credit. For training-time supervision, VPR (Yuan et al., 2026) and related work (Pronesti et al., 2026) use task-specific oracles to verify intermediate steps. Such oracles are harder to construct for open-ended web search, where retrieved evidence and partial answers may not admit simple symbolic checks. VISTA instead externalizes the partial answer as a table and uses an LLM-based task verifier to score successive states, yielding turn-level credit grounded in answer progress.

3 PRELIMINARIES

Multi-Turn Wide Search. We consider a wide-search task specified by a user query $x \sim \mathcal{D}$. Unlike conventional search augmented question answering, which typically seeks a single answer, wide search requires an agent to discover a potentially large set of target entities and their associated attributes from external sources. The desired output is a structured table $\mathcal{T}^* = \{(e_j, v_{j,1}, \ldots, v_{j,C})\}_{j=1}^{N}$, where e_j denotes a target entity and $v_{j,c}$ denotes the value of its c-th requested attribute. Following Jin et al. (2025), we treat the search engine as part of the environment. A rollout alternates between policy generated actions and environment observations:

$$\tau = [x]\,[a_1][o_1][a_2][o_2]\cdots[a_T][o_T] \tag{1}$$

where a_k is the model-generated action at interaction turn t and o_t is the corresponding observation returned by the search environment. An action may contain planning, a search query, or the final answer, while an observation contains retrieved Web content or other tool feedback. At each turn, the policy conditions on the entire context observed so far: $a_t \sim \pi_\theta\left(\cdot \mid x, a_{<t}, o_{<t}\right)$ The search environment $\mathcal{R}$ executes a search action and produces an observation $o_t \sim \mathcal{R}(\cdot \mid a_t)$. The interaction continues until the model synthesizes a final answer or reaches the maximum turn budget. For a tokenized rollout $y = (y_1, \ldots, y_L)$, let $\mathcal{P} \subseteq \{1, \ldots, L\}$ denote policy generated token positions and $\mathcal{O} \subseteq \{1, \ldots, L\}$ denote environment-observation positions. The retrieved environment tokens affect subsequent policy decisions but receive no direct policy-gradient update.

After the interaction ends, we parse the final answer into a table $\mathcal{T}(\tau)$ and compute the outcome reward against the reference table: $R(\tau) = \mathcal{E}\left(\mathcal{T}(\tau), \mathcal{T}^*\right)$, where $\mathcal{E}$ is the task verifier. The tabular formulation is one specific instantiation of a broader verifiable intermediate state concept.

GRPO for Multi-Turn Search Agents. We optimize the search policy with Group Relative Policy Optimization (GRPO). For each task x, GRPO samples a group of G trajectories $\{\tau_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x; \mathcal{R})$. Let $R_i = R(\tau_i)$ be the verifier reward of rollout i. GRPO avoids a separately learned value function by constructing a group-relative advantage from rewards within the same prompt group: $\widehat{A}_i^{\text{traj}} = \frac{R_i - \mu_g}{\sigma_g + \epsilon}$, where $\mu_g = \frac{1}{G}\sum_{i=1}^{G} R_i$, $\sigma_g = \sqrt{\frac{1}{G}\sum_{i=1}^{G}(R_i - \mu_g)^2}$. For token $y_{i,k}$, define the importance ratio $\rho_{i,k}(\theta) = \frac{\pi_\theta(y_{i,k} \mid y_{i,<k}, x; \mathcal{R})}{\pi_{\theta_{\text{old}}}(y_{i,k} \mid y_{i,<k}, x; \mathcal{R})}$. Let $\mathcal{P}_i$ denote all policy-generate tokens in the rollout i and ϵ_c denote the clipping hyperparameter. The clipped objective is[1]:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathcal{P}_i|}\sum_{k\in\mathcal{P}_i}\min\left(\rho_{i,k}(\theta)\widehat{A}_i^{\text{traj}}, \text{clip}\left(\rho_{i,k}(\theta), 1-\epsilon_c, 1+\epsilon_c\right)\widehat{A}_i^{\text{traj}}\right)\right]. \tag{2}$$

Following Shao et al. (2024), every policy action within the same trajectory receives the same outcome-derived advantage. However, it becomes increasingly coarse for long-horizon wide search. Our method addresses this gap by introducing an explicit, task-verifiable intermediate state after each turn and using progress in this state to redistribute the advantage across interaction turns, while preserving terminal outcome dominance.

4 METHOD

Wide-search agents interact with search environments over long horizons, yet standard outcome-based RL assigns the same trajectory-level credit to every decision within a rollout. This creates a fundamental attribution problem: a successful trajectory may contain both productive and unproductive action turns, while a single terminal reward cannot distinguish between them. Thus, we introduce **Verifiable Intermediate States for Turn-level Attribution (VISTA)** (Figure 1). The key idea is to expose the agent's evolving answer as an explicit intermediate state, evaluate how this state evolves after each interaction turn, and use the resulting state improvement to redistribute credit across the trajectory. To prevent imperfect intermediate-state evaluation from overwhelming the verifiable task outcome, we introduce a terminal-dominant projection that explicitly bounds all non-terminal advantages by the terminal advantage.

[1]The standard GRPO objective may include a KL penalty to a reference policy. We omit it from Eq. (2) because its coefficient is set to zero in all our experiments (Appendix C.2).

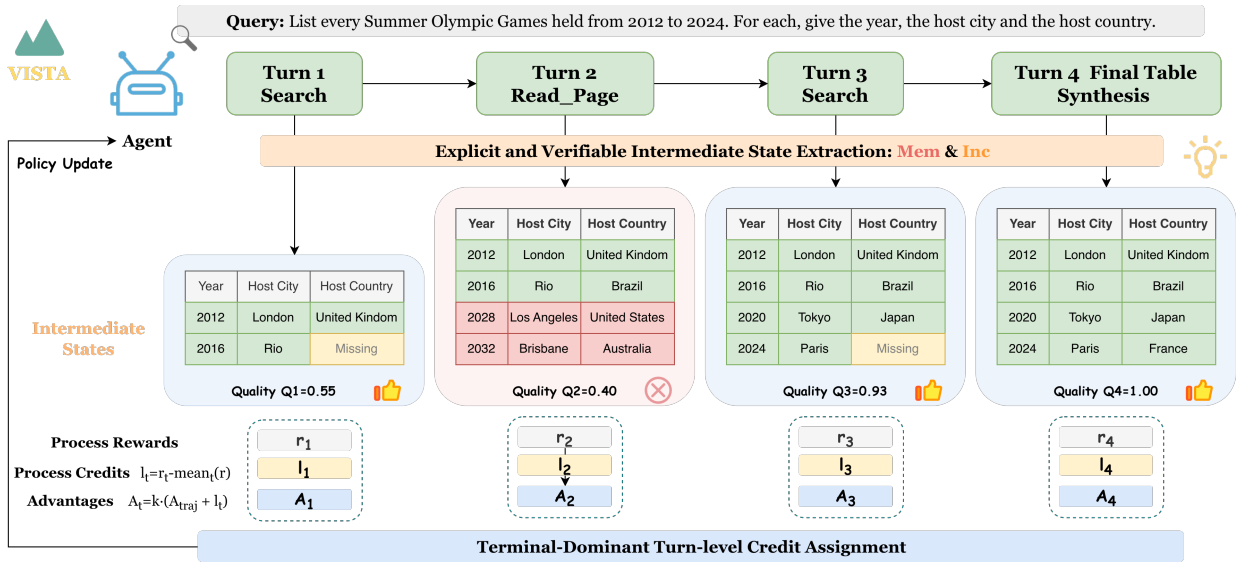

Figure 1: **Overview of VISTA.** At each turn, VISTA extracts an answer table from the agent's current context as an explicit intermediate state and scores its quality against the ground truth. Changes in state quality provide turn-level credit, which is combined with the dominant terminal reward to compute turn-level advantages.

4.1 Explicit Intermediate Table States

Let $h_{i,t} = (x, a_{i,1}, o_{i,1}, \ldots, a_{i,t-1}, o_{i,t-1})$ denote the interaction history before turn t. The policy samples an action $a_{i,t} \sim \pi_\theta(\cdot \mid h_{i,t})$, and receives an environment observation $o_{i,t}$. The main difficulty is that $h_{i,t}$ is an unstructured language context whose task progress cannot be directly measured. VISTA therefore extracts, after every turn, an explicit table state $\mathcal{T}_{i,t} = \{(\hat{e}_j, \hat{v}_{j,1}, \ldots, \hat{v}_{j,C})\}_{j=1}^{N_{i,t}}$, representing the agent's current estimate of the desired answer, producing a sequence $\mathcal{T}_{i,0}, \ldots, \mathcal{T}_{i,T_i}$ for a trajectory τ_i with T_i turns. Extraction is always performed by a dedicated extraction call $\pi_\theta^{\text{table}}$ that runs on a copy of the current context and whose output tokens are parsed and then discarded, so the isolation of the acting rollout is preserved by construction. We instantiate the construction using two mechanisms that differ in where the extracted table lives after it is parsed, which we refer to as **VISTA-Inc** and **VISTA-Mem**; VISTA-Mem is our main method.

INC: Harness-side incremental table state. INC keeps the table strictly outside the agent's interaction context. After turn t, the harness runs the extraction call on a fork of $h_{i,t+1}$ whose prompt additionally carries the previous table $\mathcal{T}_{i,t-1}^{\text{INC}}$ together with an "amend this table" instruction:

$$\mathcal{T}_{i,t}^{\text{INC}} \sim \pi_\theta^{\text{table}}\left(\cdot \mid h_{i,t+1}, \mathcal{T}_{i,t-1}^{\text{INC}}\right). \tag{3}$$

The extracted tokens are parsed into $\mathcal{T}_{i,t}^{\text{INC}}$. The table is never observed by the policy while it plans the next tool call, and it can shape behavior only indirectly, through the turn-level reward signal built from $\{\mathcal{T}_{i,t}^{\text{INC}}\}$.

MEM: In-context visible working memory. MEM performs the same extraction step, but additionally injects the extracted table back into the acting rollout as a masked user-role memory $\mathcal{M}_{i,t} \equiv \mathcal{T}_{i,t}^{\text{MEM}}$. The interaction history the policy conditions on at turn $t{+}1$ is augmented with the previous memory: $h_{i,t+1}^{\text{MEM}} = \left(x, a_{i,1}, o_{i,1}, \ldots, a_{i,t}, o_{i,t}, \mathcal{M}_{i,t-1}\right)$, and the update proceeds as:

$$\widehat{\mathcal{M}}_{i,t} \sim \pi_\theta^{\text{update}}\left(\cdot \mid h_{i,t+1}^{\text{MEM}}\right), \qquad \mathcal{M}_{i,t} = \text{Update}\left(\mathcal{M}_{i,t-1}, \widehat{\mathcal{M}}_{i,t}\right). \tag{4}$$

The memory tokens carry a zero response mask (they are not the policy's generation and are excluded from the training loss), but every subsequent action token is conditioned on them. The table thus doubles as a running working memory the policy consults while planning further searches.

INC vs. MEM. INC and MEM use the same table extraction call, $\pi_\theta^{\text{table}}$, but differ in whether the acting policy can see the extracted table. INC stores $\mathcal{T}_{i,t}^{\text{INC}}$ only in the harness, so the next action is sampled from $\pi_\theta(\cdot \mid h_{i,t+1})$. MEM also appends the parsed table to the prompt as a masked user-role memory, making it visible to the policy; the next action is

sampled from $\pi_\theta(\cdot \mid h^{\mathrm{MEM}}_{i,t+1})$, where $h^{\mathrm{MEM}}_{i,t+1}$ includes $\mathcal{M}_{i,t}$. For the next extraction call, INC passes the previous table through an explicit carry instruction, while MEM includes it through the inherited context. Both variants use the same update, but expose the acting policy to different information: In INC, the acting policy sees the interaction history but not the extracted table; in MEM, it also sees the table as an appended memory when choosing its next action. Thus, INC uses the tables to shape the policy through training, while MEM also lets the policy use them during the rollout.

4.2 Terminal-Dominant Turn-Level Credit Assignment

Intermediate states provide process rewards, but they are inherently less reliable than the final verifiable task outcome. Directly optimizing process rewards can cause the policy to favor locally rewarded behaviors at the expense of final task completion (Appendix B.2). We address this issue with **terminal-dominant turn-level credit assignment**.

Trajectory-level Reward. For each rollout i, we compute a terminal trajectory reward

$$R_i = \lambda_{\text{out}} R_i^{\text{out}} + \lambda_{\text{proc}} R_i^{\text{proc}} + \lambda_{\text{fmt}} R_i^{\text{fmt}}, \tag{5}$$

where R_i^{out} measures the correctness of the final answer, R_i^{proc} evaluates the quality of the search trajectory, and R_i^{fmt} rewards compliance with the required output format. R_i is computed after the complete rollout and provides trajectory-level supervision. Following GRPO, we first compute the conventional group-relative advantage $\widehat{A}_i^{\text{traj}} = \frac{R_i - \mu_g}{\sigma_g + \epsilon}$, where μ_g and σ_g are the mean and standard deviation of trajectory rewards within the group. The resulting A_i^{traj} captures the relative quality of the complete trajectory but does not distinguish the contributions of individual turns.

Process-level Credit. Given an intermediate state $\mathcal{T}_{i,t}$, an LLM evaluator assigns a quality score $Q_{i,t} = \mathcal{E}(\mathcal{T}_{i,t}, \mathcal{T}^*)$. We distribute the process signal uniformly across turns as $r_{i,t} = Q_{i,t}/T_i$, whose trajectory sum defines the process component of the trajectory reward, $R_i^{\text{proc}} = \hat{Q}_i = \frac{1}{T_i}\sum_{t=1}^{T_i} Q_{i,t}$, weighted by λ_{proc} in Eq. (5). Absolute process rewards may vary substantially across trajectories. We center the process rewards within each trajectory and define the relative process credit $\ell_{i,t}$:

$$\ell_{i,t} = r_{i,t} - \frac{1}{T_i}\sum_{t'=1}^{T_i} r_{i,t'}. \tag{6}$$

By construction $\sum_{t=1}^{T_i} \ell_{i,t} = 0$, the process signal redistributes credit within a trajectory rather than uniformly increasing or decreasing the trajectory's overall learning signal. We combine global and local credit as:

$$\widetilde{A}_{i,t} = \widehat{A}_i^{\text{traj}} + \ell_{i,t}. \tag{7}$$

Centering the process credit gives $\sum_{t=1}^{T_i} \ell_{i,t} = 0$. The preliminary advantages redistribute credit across turns while preserving trajectory advantage under an unweighted turn average: $\frac{1}{T_i}\sum_{t=1}^{T_i} \widetilde{A}_{i,t} = A_i^{\text{traj}}$. This equality applies before terminal-dominant projection and does not imply zero-mean process credit in the token-weighted policy objective.

Terminal-Dominant Projection. The trajectory advantage A_i^{traj} reflects the final outcome, whereas the process signal $\ell_{i,t}$ identifies turns that improve the intermediate answer state. Our preliminary study (Appendix B) shows that emphasizing local progress can weaken final-answer quality for wide search task. We therefore leave the terminal-turn advantage $\widetilde{A}_{i,T_i}$ unchanged and scale the preceding advantages by a shared coefficient:

$$A_{i,t} = \begin{cases} \kappa_i \widetilde{A}_{i,t}, & t < T_i, \\ \widetilde{A}_{i,T_i}, & t = T_i, \end{cases} \qquad \kappa_i = \begin{cases} 1, & T_i = 1, \\ \min\left(1, \dfrac{(1-\delta)|\widetilde{A}_{i,T_i}|}{\max_{t<T_i} |\widetilde{A}_{i,t}| + \epsilon}\right), & T_i > 1. \end{cases} \tag{8}$$

Here, $\delta \in (0,1)$ sets the dominance margin and $\epsilon > 0$ ensures numerical stability. For a single-turn trajectory, no scaling is needed. For longer trajectories, the projection preserves the relative credit assigned to earlier turns and bounds the magnitude of each non-terminal advantage by $(1-\delta)$ times that of the terminal-turn advantage.

Proposition 1 (Terminal-Dominance Guarantee)**.** *For trajectory-specific projection coefficient κ_i and advantages defined by Eq. (8), every non-terminal turn satisfies: $|A_{i,t}| \le (1-\delta)|A_{i,T_i}|, \quad \forall t < T_i$. Moreover, the terminal advantage is invariant under the projection: $A_{i,T_i} = \widetilde{A}_{i,T_i}$. The proof is given in Appendix A.*

4.3 Policy Optimization

VISTA preserves the token-level clipped policy objective of GRPO (Shao et al., 2024) while replacing trajectory-level credit with turn-level advantages. For rollout i, let $\mathcal{P}_{i,t}$ denote the set of policy-generated token positions in turn t,

and let $\mathcal{P}_i = \bigcup_{t=1}^{T_i} \mathcal{P}_{i,t}$ collect all policy-generated tokens in the rollout. Tokens returned by the search environment are excluded from $\mathcal{P}_i$ and therefore masked from policy optimization. For a policy-generated token m, we define the importance ratio as $\rho_{i,m}(\theta) = \frac{\pi_\theta(y_{i,m}|y_{i,<m},x)}{\pi_{\theta_{\text{old}}}(y_{i,m}|y_{i,<m},x)}$. Each policy-generated token in interaction turn t is assigned the terminal-dominant projected turn-level advantage $A_{i,t}$ from Eq. (8). We omit the KL regularization term following recent agentic RL practice (Wang et al., 2026b; Yu et al., 2026) and aggregate the loss over all valid policy tokens in the batch so that each policy decision receives equal weight regardless of the containing rollout's length. The VISTA objective is:

$$\mathcal{J}_{\text{VISTA}}(\theta) = \mathbb{E}\left[\frac{1}{\sum_{i=1}^{G}|\mathcal{P}_i|}\sum_{i=1}^{G}\sum_{t=1}^{T_i}\sum_{m\in\mathcal{P}_{i,t}} \min\Big(\rho_{i,m}(\theta)\,A_{i,t},\ \text{clip}\big(\rho_{i,m}(\theta),\,1-\epsilon_c,\,1+\epsilon_c\big)\,A_{i,t}\Big)\right]. \tag{9}$$

VISTA changes the **granularity of credit assignment**: standard GRPO broadcasts a single trajectory-level advantage $\widehat{A}_i^{\text{traj}}$ to all policy-generated tokens in rollout i, whereas VISTA broadcasts the turn-specific advantage $A_{i,t}$ only to policy-generated tokens within interaction turn t. This preserves the standard clipped surrogate optimization while allowing intermediate search decisions to receive distinct credit.

5 Experiments

In this section, we compare VISTA with baselines on three wide-search benchmarks across multiple model families and scales. We also evaluate its generalization to multi-hop QA benchmarks. Ablation studies examine the contributions of turn-level credit assignment and explicit intermediate states, followed by an analysis of training dynamics. Additional experimental details and results are provided in Appendix C and Appendix D.

Experimental Setup. We use Qwen3-4B, Qwen3-8B, Qwen3-14B (Yang et al., 2025), Qwen3.8-27B (Qwen Team, 2026), Llama3.1-8B-Instruct (Grattafiori et al., 2024), and Gemma4-E4B-it (Team, 2026) as backbone models for a ReAct-style single agent. We train on the WideSeekR1 training set (Xu et al., 2026) and use the Perplexity Search API (Perplexity AI, 2026) as the web search backend. Further training details are provided in Appendix C.2.

Benchmarks. We evaluate our method on three wide-search benchmarks: WideSearch (Wong et al., 2026), Deep-WideSearch (Lan et al., 2025), and GISA (Zhu et al., 2026). To assess generalization beyond wide search, we also evaluate on three multi-hop QA benchmarks: HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MusiQue (Trivedi et al., 2022). Following Gao et al. (2025), we randomly sample 1,000 instances from the validation set of each multi-hop QA benchmark.

Baselines. We compare VISTA with two groups of training baselines. Outcome-supervised methods include Search-R1 (Jin et al., 2025), WideSeek R1 (Xu et al., 2026), and WebLeaper (Tao et al., 2025) (denoting as the GRPO baseline using the granular F-score reward). Process-supervised methods include Step Search (Wang et al., 2025), TIPS (Xie et al., 2026), and CW-GRPO (Wang et al., 2026b). We also report results for reference models evaluated without additional training: the API-based GPT5.4-mini (OpenAI, 2026), and the open-weight SearchR1-7B (Jin et al., 2025) and ASearcher-Web-14B (Gao et al., 2025). Implementation details are provided in Appendix C.3.

Evaluation Metrics. For WideSearch, we follow Wong et al. (2026) and report item F1, row F1, and success rate. To distinguish finding the correct rows from filling in their cells, we additionally report discovery F1 (Disc F1), which matches rows by primary key regardless of cell values, and completion rate (Comp Rate), the fraction of correct cells among all cells in the matched rows. We also report soft success rate at threshold τ, defined as the fraction of trials with item F1 $\geq \tau$. For the other benchmarks, we use their respective evaluation metrics. Metric definitions and evaluation details are provided in Appendix C.1.

5.1 Main Results

VISTA-Mem improves wide-search performance across model scales and families. As shown in Table 1, VISTA-Mem achieves the best results on most WideSearch metrics across the evaluated backbones, with particularly strong gains in item F1 and discovery F1. With Qwen3.8-27B, it also surpasses GPT5.4-mini on the most reported metrics. Even with Qwen3-4B, it outperforms the open-weight SearchR1-7B and ASearcher-Web-14B reference models. Compared with the corresponding base single agent, VISTA-Mem raises item F1 from 31.08 to 36.61 with Qwen3-4B, from 32.67 to 41.71 with Qwen3-14B, and from 43.36 to 65.25 with Qwen3.8-27B. These are gains of 5.53, 9.04, and 21.89 points, respectively, showing larger gains for the larger Qwen backbones evaluated here. The method also improves over the baseline on the evaluated Llama backbone, while its performance on Gemma4-E4B is comparable to outcome-only GRPO. Additional results are provided in Appendix D.1.

Table 1: **Main results on WideSearch Benchmark.** Avg@4 and Max@4 are the per-instance mean and max over 4 trials, averaged over instances; Pass@4 is the per-instance max of the exact-table success rate. Disc F1 scores row identity only and Comp Rate is the fraction of cells correct within the rows a method found; both are Avg@4. For Success Rate: τ=.5 and τ=.95 count a trial as successful when its Item F1 reaches τ. Best value per column in **bold**.

Method	Item F1 (%)		Row F1 (%)		Row Discovery (%)		Success Rate (%)			
	Avg@4	Max@4	Avg@4	Max@4	Disc F1	Comp Rate	Avg@4	Pass@4	τ=.95	τ=.5
Reference Models (No Additional Training)										
GPT5.4-mini	58.68	71.18	32.50	49.26	73.62	79.70	2.75	6.50	9.50	56.32
Search-R1-7B	28.29	36.81	6.10	10.41	47.13	59.41	0.00	0.00	0.12	18.75
ASearcher-Web-14B	12.22	30.85	3.20	8.99	44.87	22.65	0.00	0.00	0.50	9.62
Qwen3-4B	31.08	41.74	7.22	12.84	51.59	55.47	0.00	0.00	0.50	25.12
Search-R1	30.56	44.41	8.69	16.09	46.87	52.62	0.00	0.00	0.62	26.88
WideSeek-R1-4B	32.12	42.69	8.01	13.64	56.36	55.84	0.00	0.00	0.62	27.62
GRPO - WebLeaper	29.85	45.01	8.43	15.73	45.73	52.36	0.25	1.00	0.88	25.88
GRPO - WebLeaper+Fmt	32.16	46.17	9.15	16.20	50.35	53.37	0.25	0.50	1.00	30.12
StepSearch	33.37	42.71	8.39	14.89	54.29	60.59	0.25	1.00	0.88	26.38
TIPS	32.95	41.66	8.72	13.78	51.54	55.91	0.00	0.00	0.53	27.85
CW-GRPO	32.32	42.60	6.59	11.55	55.44	53.66	0.25	0.50	0.50	25.62
VISTA-Inc	35.57	46.30	9.10	15.56	57.09	59.08	0.12	0.50	**1.00**	31.88
VISTA-Mem	**36.61**	**46.68**	**9.54**	**16.30**	**58.10**	**60.69**	0.25	0.50	0.75	**34.75**
Llama-3.1-8B-Instruct	27.29	40.80	7.46	13.72	43.01	46.33	0.00	0.00	1.12	24.88
GRPO - WebLeaper	30.89	39.55	7.14	11.09	51.37	53.15	0.00	0.00	0.62	25.37
GRPO - WebLeaper+Fmt	31.73	40.30	**9.58**	13.34	49.84	56.62	0.12	0.50	**1.38**	27.75
StepSearch	30.59	37.52	7.42	11.52	51.51	56.61	0.00	0.00	0.62	22.12
TIPS	32.82	45.34	8.30	14.57	52.56	52.36	0.12	0.50	0.38	29.38
CW-GRPO	28.15	38.74	5.49	9.60	49.37	47.27	0.00	0.00	1.00	22.38
VISTA-Inc	28.93	37.54	6.92	11.29	48.75	48.93	0.00	0.00	0.88	24.50
VISTA-Mem	**33.79**	**42.36**	8.52	**14.00**	**55.25**	**57.14**	0.12	0.50	1.25	**30.25**
Qwen3-8B	34.23	44.98	8.80	16.71	55.50	16.07	0.25	1.00	0.38	28.88
GRPO - Webleaper+Fmt	20.76	25.66	1.84	3.66	43.79	45.24	0.00	0.00	0.00	12.38
VISTA-Mem	**39.23**	**48.04**	**10.12**	**16.66**	**60.36**	**64.78**	0.00	0.00	**0.62**	**37.00**
Qwen3-14B	32.67	44.04	9.24	15.50	56.13	18.80	0.00	0.00	0.62	28.88
GRPO - Webleaper+Fmt	41.43	50.63	12.54	19.89	62.99	65.19	0.00	0.00	1.12	38.88
VISTA-Mem	**41.71**	**52.28**	**13.14**	**21.37**	**63.62**	**65.94**	0.00	0.00	**2.00**	**41.00**
Qwen3.8-27B	43.36	62.50	30.65	46.04	48.69	50.71	2.75	7.00	13.00	47.62
GRPO - Webleaper+Fmt	63.62	73.28	34.64	46.92	77.41	77.91	2.88	7.00	**15.38**	69.12
VISTA-Mem	**65.25**	**74.08**	**37.73**	**49.28**	**78.25**	**78.72**	3.25	6.00	14.00	**71.75**
Gemma-4-E4B-it	37.99	47.31	13.05	20.98	56.42	65.31	0.50	2.00	1.50	35.00
GRPO - Webleaper+Fmt	**45.75**	**56.26**	16.53	**26.48**	64.96	66.36	0.12	0.50	2.12	45.38
VISTA-Mem	45.30	54.70	**16.75**	25.96	**64.96**	**69.42**	0.62	2.00	**2.12**	**46.00**

Explicit memory strengthens turn-level supervision. Both VISTA-Mem and VISTA-Inc outperform the training baselines overall, while VISTA-Mem generally performs better than VISTA-Inc. This result is consistent with the benefit of making the intermediate table available to the agent in subsequent turns: the agent can condition its next actions on the state that receives turn-level credit during training.

VISTA transfers to benchmarks that combine deep and wide search. DeepWideSearch and GISA add a deep-search component to wide search: finding a target item can require several search steps before the agent can add it to the answer table. Although VISTA is trained only on wide-search data, it improves performance on both benchmarks without further training (Figure 2). On DeepWideSearch (Table 9), the best variant improves Item F1 (Avg@4) over the base single agent by 43%, 36%, and 20% relative for the 4B, 8B, and 14B backbones, respectively. On GISA (Table 10), overall EM improves by 31 to 45% relative. The gains across all six model - benchmark pairs suggest that VISTA's learned behavior also helps when finding items requires deeper search. We further test transfer to multi-hop QA tasks in Section 5.4.

5.2 Ablation Study

We ablate the explicit intermediate state and terminal-dominant turn-level attribution on WideSearch-EN to examine their individual and combined effects. As shown in Table 2, adding the memory state without turn-level attribution

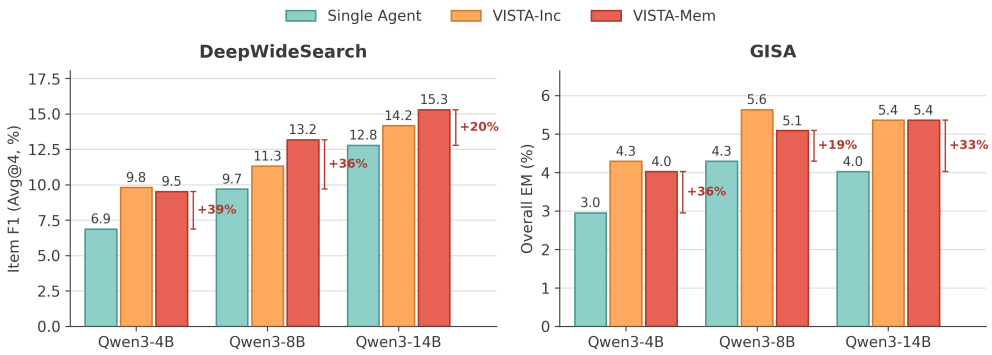

Figure 2: Relative improvements over the base single agent on DeepWideSearch and GISA. Full results are provided in Appendix D.2.

Table 2: **Component ablation of VISTA on WideSearch-EN.** I.S. denotes the intermediate state and TA denotes terminal-dominant turn-level attribution. When I.S. is disabled but T.A. is enabled, turn-level credit is based on overlap between tool responses and reference entity–attribute pairs, without constructing an intermediate table. All variants use the same outcome, process, and format reward components.

Components		**Item F1**		**Row F1**		**Disc F1**	
I.S.	**TA**	Avg@4	Max@4	Avg@4	Max@4	Avg@4	Max@4
–	✗	37.01	48.86	8.46	15.20	59.12	72.88
Mem	✗	35.00	48.92	9.60	17.37	54.03	71.43
–	✔	37.34	**51.06**	9.43	16.45	57.18	**74.64**
Mem	✔	**40.17**	50.46	**9.66**	**17.85**	**62.93**	74.32

does not consistently help. Compared with trajectory-level training without the state, Row F1 increases from 8.46 to 9.60, but Item F1 drops from 37.01 to 35.00 and Discovery F1 from 59.12 to 54.03. Thus, making intermediate progress explicit is not sufficient when its score is folded into a single trajectory-level learning signal. Turn-level attribution alone has a similarly limited effect. Without the memory state, it changes Item F1 only from 37.01 to 37.34, while Discovery F1 decreases from 59.12 to 57.18. The strongest results instead come from combining the two components: VISTA-Mem reaches 40.17 Item F1, 9.66 Row F1, and 62.93 Discovery F1. Compared with the memory-state variant using trajectory-level credit, this corresponds to relative gains of 14.77% in Item F1 and 16.47% in Discovery F1. The comparison shows that the state becomes more useful when its progress signal is assigned to the turns responsible for producing it.

We also compare turn-level credit placements while holding the reward and memory mechanism fixed (Table 12). Reward-to-go passes terminal reward back to earlier turns, whereas terminal-preserved attribution can assign a larger advantage to an earlier turn than to the answer turn. Both yield smaller gains than terminal-dominant attribution, which uses intermediate progress to distinguish earlier turns while keeping the terminal objective dominant. Overall, the component ablation supports the joint design of an explicit intermediate state and terminal-dominant credit assignment. Full results, including alternative reward shaping and advantage estimation strategies, are provided in Appendix D.3.

5.3 RL Training Dynamics Analysis

We examine how performance and search behavior change during training. VISTA improves held-out Item F1 while outcome-only GRPO declines after its peak (Figure 3(a)). For both VISTA variants, the mean magnitude of terminal advantages remains larger than that of process advantages throughout training, while process advantages retain a substantial magnitude (Figure 3(b)). The methods also show different search behavior. GRPO makes progressively more searches per trajectory, whereas the search counts of both VISTA variants remain comparatively stable (Figure 3(c)). GRPO's duplicate-query rate rises sharply later in training, while both VISTA variants keep it low (Figure 3(d)). These observations suggest that VISTA's gains do not depend on increasing search volume and are accompanied by less repeated search. We provide the extensive training dynamics analysis in Appendix D.4.

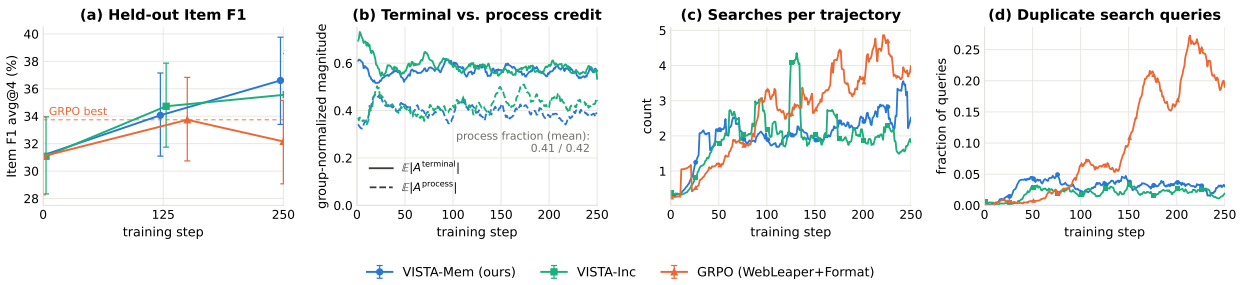

Figure 3: **RL training dynamics on WideSearch.** (a) Held-out Item F1 over training steps. (b) Group-normalized terminal and process credit magnitudes for VISTA-Mem and VISTA-Inc; terminal credit remains larger throughout training. (c) Mean searches per trajectory and (d) fraction of duplicate search queries. Compared with GRPO, both VISTA variants keep duplicate-query rates low as training proceeds, while GRPO increasingly repeats searches.

Table 3: **Zero-shot transfer from wide-search training to three multi-hop QA benchmarks.** All methods use Qwen3-4B and receive no additional training on these benchmarks. EM and F1 are reported in percent; the best result in each column is shown in **bold**.

	HotpotQA		2WikiMultiHopQA		MuSiQue		Avg	
Method	EM	F1	EM	F1	EM	F1	EM	F1
Qwen3-4B (Base)	32.9	43.2	37.1	40.7	8.7	16.2	26.2	33.4
WideSeekR1-4B	36.0	46.4	41.8	45.5	10.5	17.8	29.4	36.5
VISTA-Inc	38.5	50.1	**48.7**	**53.6**	13.0	21.6	33.4	41.8
VISTA-Mem	**40.1**	**51.4**	47.7	52.7	**14.6**	**22.5**	**34.1**	**42.2**

5.4 Transfer to Multi-Hop QA Tasks

We next test whether VISTA transfers beyond wide search. We evaluate the Qwen3-4B models trained on wide search task directly on HotpotQA, 2WikiMultiHopQA, and MuSiQue, without further training on these datasets. Following Gao et al. (2025), we report exact match (EM) and F1. As shown in Table 3, both VISTA variants improve over the base model and WideSeekR1-4B on all three benchmarks. VISTA-Mem raises average F1 from 33.4 to 42.2 over the base model, an 8.8 point gain, and from 36.5 to 42.2 over WideSeekR1-4B, a 5.7 point gain. VISTA-Inc performs best on 2WikiMultiHopQA, while VISTA-Mem achieves the highest average EM and F1. These results show that training on wide search also improves performance on multi-hop QA, where questions require finding and connecting evidence across multiple steps.

We also train VISTA-Mem directly on the multi-hop QA training datasets. It reaches an average F1 of 59.5, improving by 26.1 points (78.1% relative) over the untrained Qwen3-4B base model. This shows that VISTA can be applied to multi-hop QA training. Full results and training details are provided in Appendix D.7.

Further analysis of the training harness and WideSearch results appears in Appendix D.1. Training efficiency and inference cost are discussed in Appendix D.5, and hyperparameter sensitivity in Appendix D.6.

6 Conclusion

In this work, we study how to train open-source agents for long-horizon wide search, where sparse trajectory-level rewards provide limited guidance about which decisions contribute to the final outcome. We introduce VISTA, which makes intermediate search progress explicit through verifiable table states and uses their scores for turn-level credit assignment while preserving the terminal outcome as the dominant learning signal. Across multiple benchmarks, model families, and scales, VISTA consistently improves over outcome-only and step-wise RL baselines and generalizes to standard multi-hop QA tasks. Our analysis further shows that effective process supervision depends not simply on adding intermediate rewards, but on assigning credit to the responsible decisions. We hope these findings provide a step toward more effective credit assignment for training long-horizon broad information-seeking agents.

AI USE STATEMENT

LLMs were used in this work primarily for language polishing, grammar correction, and improving writing clarity during the preparation of the manuscript. We also used LLMs to help identify potentially relevant literature. The authors reviewed and take responsibility for the final text.

ETHICS STATEMENT

VISTA trains search agents to gather and organize information from web search results. Such agents may encounter inaccurate, biased, or sensitive content, and their outputs may reproduce these issues. Our experiments use existing research benchmarks and evaluate answer quality rather than the suitability of generated content for consequential decisions. We encourage users to verify retrieved information and consider privacy and potential misuse when deploying search agents beyond the benchmark setting.

REPRODUCIBILITY STATEMENT

We describe VISTA's intermediate states, reward construction, and turn-level credit assignment in Section 4. The experimental setup, including datasets, model backbones, baselines, evaluation metrics, and detailed training settings, is provided in Section 5 and the Appendix C. The appendix also reports additional results, ablations, and hyperparameter analyses. These details are intended to support independent reproduction of our results.

REFERENCES

Ilze Amanda Auzina, Joschka Strüber, Sergio Hernández-Gutiérrez, Shashwat Goel, Ameya Prabhu, and Matthias Bethge. Intrinsic credit assignment for long horizon interaction. *arXiv preprint arXiv:2602.12342*, 2026.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 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 preprint arXiv:2603.15594*, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026.

Chengguang Gan, Zhixi Cai, Yunhao Liang, Hanjun Wei, Shiwen Ni, and Qinghao Zhang. A learning-rate-gated failure of grpo in a small language and vision-language model web agent: A controlled null and its mechanism. *arXiv preprint arXiv:2607.12640*, 2026.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous rl. *arXiv preprint arXiv:2508.07976*, 2025.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

Yuyang Hu, Hongjin Qian, Shuting Wang, Jiongnan Liu, Ziliang Zhao, Jiejun Tan, Zheng Liu, and Zhicheng Dou. Sam: State-adaptive memory for long-horizon reasoning agent. *arXiv preprint arXiv:2605.24468*, 2026.

Yuxuan Huang, Yihang Chen, Zhiyuan He, Yuxiang Chen, Ka Yiu Lee, Huichi Zhou, Weilin Luo, Meng Fang, and Jun Wang. Web2bigtable: A bi-level multi-agent llm system for internet-scale information search and extraction. *arXiv preprint arXiv:2604.27221*, 2026.

Yifu Huo, Shunjie Xing, Chenglong Wang, Peinan Feng, Qiaozhi He, Yan Ding, Anxiang Ma, Yuxin Gao, Tongran Liu, Tong Xiao, et al. Learning from environmental feedback: Credit assignment across multiple timescales for agentic reinforcement learning. *arXiv preprint arXiv:2608.08255*, 2026.

Pengcheng Jiang, Zhiyi Shi, Kelly Hong, Xueqiang Xu, Jiashuo Sun, Jimeng Sun, Hammad Bashir, and Jiawei Han. Harness-1: Reinforcement learning for search agents with state-externalizing harnesses. *arXiv preprint arXiv:2606.02373*, 2026.

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

Tian Lan, Bin Zhu, Qianghuai Jia, Junyang Ren, Haijun Li, Longyue Wang, Zhao Xu, Weihua Luo, and Kaifu Zhang. Deepwidesearch: Benchmarking depth and width in agentic information seeking. *arXiv preprint arXiv:2510.20168*, 2025.

Tian Lan, Felix Henry, Bin Zhu, Qianghuai Jia, Junyang Ren, Qihang Pu, Haijun Li, Longyue Wang, Zhao Xu, and Weihua Luo. Table-as-search: Agentic information seeking is table completion. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 15088–15107, 2026.

Ka Yiu Lee, Yuxuan Huang, Zhiyuan He, Huichi Zhou, Weilin Luo, Kun Shao, Meng Fang, and Jun Wang. Infoseeker: A scalable hierarchical parallel agent framework for web information seeking. *arXiv preprint arXiv:2604.02971*, 2026.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, 2025.

Zheng Liu, Longxiang Zhang, Xintong Wang, Zhiang Xu, Shaoxiong Zhan, Xin Shan, Wen Huang, Tao Dai, Shu-Tao Xia, Chengfu Huo, et al. Arbor: Online process rewards via a reusable rubric buffer for search agents. *arXiv preprint arXiv:2606.03239*, 2026.

Kun Luo, Hongjin Qian, Zheng Liu, Ziyi Xia, Shitao Xiao, Zhao Cao, Siqi Bao, Jun Zhao, and Kang Liu. Reinforcing agentic search via reward density optimization. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10261–10283, 2026.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pp. 9025–9049, 2024.

OpenAI. Deep research system card. https://openai.com/index/deep-research-system-card/, February 2025. Accessed: 2026-09-08.

OpenAI. Introducing GPT-5.4 mini and nano. https://openai.com/index/introducing-gpt-5-4-mini-and-nano/, March 2026. Accessed: 2026-09-25.

Perplexity AI. Perplexity search api. https://docs.perplexity.ai/, 2026. Accessed: 2026-09-03.

Massimiliano Pronesti, Anja Belz, and Yufang Hou. Beyond outcome verification: Verifiable process reward models for structured reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 32187–32202, 2026.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

Leitian Tao, Baolin Peng, Wenlin Yao, Tao Ge, Hao Cheng, Mike Hang Wang, Jianfeng Gao, and Sharon Li. Trace: Turn-level reward assignment via credit estimation for long-horizon agents. *arXiv preprint arXiv:2607.13988*, 2026.

Zhengwei Tao, Haiyang Shen, Baixuan Li, Wenbiao Yin, Jialong Wu, Kuan Li, Zhongwang Zhang, Huifeng Yin, Rui Ye, Liwen Zhang, et al. Webleaper: Empowering efficiency and efficacy in webagent via enabling info-rich seeking. *arXiv preprint arXiv:2510.24697*, 2025.

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025a. URL https://arxiv.org/abs/2412.15115.

Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, et al. Tongyi deepresearch technical report. *arXiv preprint arXiv:2510.24701*, 2025b.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Daoyu Wang, Qingchuan Li, Mingyue Cheng, Jie Ouyang, Shuo Yu, Qi Liu, and Enhong Chen. Steppo: Step-aligned policy optimization for agentic reinforcement learning. *arXiv preprint arXiv:2604.18401*, 2026a.

Junzhe Wang, Zhiheng Xi, Yajie Yang, Hao Luo, Shihan Dou, Tao Gui, and Qi Zhang. Enhancing llm-based search agents via contribution weighted group relative policy optimization. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31704–31718, 2026b.

Zechuan Wang, Siyuan Lu, Hongxuan Zhang, Linjian Mo, Chenyi Zhuang, and Leilei Gan. Teach the magnitude, not the direction: Verifier-bounded credit assignment for multi-turn multi-step llm agents. *arXiv preprint arXiv:2608.13179*, 2026c.

Zi-Han Wang, Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Jie Wu, Zhengzhou Cai, Yueqing Sun, Ziang Ye, Linji Hao, Qi Gu, et al. Agentopsd: Recursive self-distillation for agentic reinforcement learning. *arXiv preprint arXiv:2608.05987*, 2026d.

Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. *arXiv preprint arXiv:2505.15107*, 2025.

Ryan Wong, Jiawei Wang, Li Chen, Yan Gao, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, Wenhao Huang, Yang Wang, et al. Widesearch: Benchmarking agentic broad info-seeking. In *International Conference on Learning Representations*, volume 2026, pp. 10012–10086, 2026.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhenglin Wang, Zhengwei Tao, Ding-Chu Zhang, Zekun Xi, Robert Tang, et al. Webdancer: Towards autonomous information seeking agency. *Advances in Neural Information Processing Systems*, 38:120957–120985, 2026.

Yutao Xie, Nathaniel Thomas, Nick Hansen, Yang Fu, Li Li, and Xiaolong Wang. Tips: Turn-level information-potential reward shaping for search-augmented llms. In *International Conference on Learning Representations*, volume 2026, pp. 156549–156584, 2026.

Zelai Xu, Zhexuan Xu, Ruize Zhang, Chunyang Zhu, Shi Yu, Weilin Liu, Quanlu Zhang, Wenbo Ding, Chao Yu, and Yu Wang. Wideseek-r1: Exploring width scaling for broad information seeking via multi-agent reinforcement learning. *arXiv preprint arXiv:2602.04634*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Huining Yuan, Zelai Xu, Huaijie Wang, Xiangmin Yi, Jiaxuan Gao, Xiao-Ping Zhang, Yu Wang, Chao Yu, and Yi Wu. Verifiable process rewards for agentic reasoning. *arXiv preprint arXiv:2605.10325*, 2026.

Haiyue Zhang. Credit without ground truth: Auditing step-level credit assignment in llm agents against executed replay. *arXiv preprint arXiv:2608.19760*, 2026.

Yaocheng Zhang, Haohuan Huang, Zijun Song, Zijie Zhao, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. Criticsearch: Fine-grained credit assignment for search agents via a retrospective critic. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12272–12290, 2026a.

Yuyao Zhang, Junjie Gao, Zhengxian Wu, Jiaming Fan, Jin Zhang, Shihan Ma, Yao Yao, Weiran Qi, Chuyan Jin, Guiyu Ma, et al. Searchos-v1: Towards robust open-domain information-seeking agent collaboration. *arXiv preprint arXiv:2607.15257*, 2026b.

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. *arXiv preprint arXiv:2310.04406*, 2023.

Yutao Zhu, Xingshuo Zhang, Maosen Zhang, Jiajie Jin, Liancheng Zhang, Xiaoshuai Song, Kangzhi Zhao, Wencong Zeng, Ruiming Tang, Han Li, et al. Gisa: A benchmark for general information-seeking assistant. *arXiv preprint arXiv:2602.08543*, 2026.

APPENDIX CONTENTS

The Appendix is organized as follows.

- Appendix A: **Proof for The Proposition 1**
- Appendix B: **Preliminary Study**
  - **Section § B.1** *Intermediate State Motivation*: Why Intermediate States Are a Viable Learning Signal
  - **Section § B.2** *Terminal Dominance Turn-level Attribution Motivation*: Why Turn-Level Credit Need the Terminal Dominance
- Appendix C: **Detailed Experimental Settings**
  - **Section § C.1** Benchmarks & Evaluation Details
  - **Section § C.2** Implementation Details
  - **Section § C.3** Baselines
  - **Section § C.4** VISTA-Mem vs. VISTA-Inc
- Appendix D: **More Experimental Results**
  - **Section § D.1** WideSearch Benchmark
  - **Section § D.2** DeepWideSearch & GISA Benchmark
  - **Section § D.3** Ablation Study
    - The Reward Shaping and Credit Assignment
    - Removing LLM Judge from the Reward
    - Ablation Study on Quality Score
    - Ablation Study on Training Data
  - **Section § D.4** Training Dynamics
  - **Section § D.5** Efficiency Analysis
  - **Section § D.6** Hyperparameter Sensitivity
    - VISTA Reward Weights
    - GRPO Rollouts
    - Learning Rate
    - Terminal Dominance Cap
  - **Section § D.7** Beyond Wide Search
- Appendix E: Notation Overview

A PROOF FOR THE PROPOSITION 1

Proposition 1 (Terminal-Dominance Guarantee). *For trajectory-specific projection coefficient κ_i and advantages defined by Eq. (8), every non-terminal turn satisfies*

$$|A_{i,t}| \le (1-\delta)|A_{i,T_i}|, \qquad \forall t < T_i.$$

Moreover, the terminal advantage is invariant under the projection: $A_{i,T_i} = \widetilde{A}_{i,T_i}$.

Proof. Let

$$M_i = \max_{t<T_i} |\widetilde{A}_{i,t}|.$$

If

$$M_i \le (1-\delta)|\widetilde{A}_{i,T_i}|,$$

then $\kappa_i = 1$, and the desired inequality already holds.

Otherwise,

$$\kappa_i = \frac{(1-\delta)|\widetilde{A}_{i,T_i}|}{M_i + \epsilon} < 1.$$

For every $t < T_i$,

$$\begin{aligned} |A_{i,t}| &= \kappa_i |\widetilde{A}_{i,t}| \\ &\leq \kappa_i M_i \\ &\leq (1-\delta)|\widetilde{A}_{i,T_i}| \\ &= (1-\delta)|A_{i,T_i}|. \end{aligned}$$

The second claim follows directly from $A_{i,T_i} = \widetilde{A}_{i,T_i}$ by definition. □

B Preliminary Study

In this section, we conduct two empirical studies to motivate the key design choices of VISTA. First, we show that base policies can already externalize meaningful intermediate table states, that harness-level elicitation makes these states reliably observable, and that their quality evolves with search progress. Second, we show that process signals must be incorporated carefully: rewarding retrieval or evidence utilization alone can optimize intermediate behavior while weakening the terminal outcome. These findings motivate VISTA's two central principles: **verifiable intermediate states** for exposing process progress and **terminal-dominant attribution** for assigning turn-level credit while preserving the final objective.

B.1 Why Intermediate States Are a Viable Learning Signal

VISTA relies on a simple premise: during wide search, the policy can externalize its current evidence into a structured intermediate table, whose evolution can then provide verifiable turn-level supervision. However, three empirical preconditions must hold before using such states for RL:

- The base policy must already be able to produce a meaningful table from partial search evidence; otherwise, there is no useful signal from which RL can bootstrap;
- This state must be extracted reliably throughout the trajectory rather than depending on an additional tool-routing decision;
- The intermediate table should evolve with newly acquired evidence and expose information that is not already captured by the terminal answer.

We examine these conditions below.

Base policies can already externalize partial search state. A central design choice in VISTA is to request a best-effort intermediate table state after each tool observation. This does not require the policy to have already solved the task: the table is intended to summarize only the evidence accumulated so far. If an untrained policy rarely follows this instruction, or produces tables unrelated to the observed evidence, the resulting state would provide little useful supervision at the beginning of RL training. We therefore probe this capability before training. We freeze 100 mid-rollout states sampled from real WideSearch trajectories, each containing at least two ground-truth row keys in the visible tool observation, thereby excluding cases in which there is simply nothing relevant to record. For each state, we issue the exact VISTA state instruction to five base models and sample four responses per model. As shown in Table 4, structured state generation is already a strong zero-shot capability. All evaluated Qwen3 models and Llama-3.1-8B produce a parseable Markdown table in 100% of responses, while Gemma-4-E4B does so in 92.7%. More importantly, these tables are not merely syntactically valid. Across models, 95 to 99% of emitted columns conform to the requested schema, 75 to 81% of rows are grounded in the immediately preceding tool observation, and 70 to 80% of visible gold keys are transferred into the table. Even before RL, the resulting states achieve 24 to 32% cell-level F1 against the full ground truth under the same deterministic scorer later used. The relatively small variation across model scales further suggests that externalizing partial evidence into a table is not a capability unique to a particular backbone. These results establish the first precondition for VISTA : the intermediate state does not need to be learned from scratch. **The base policy already possesses sufficient instruction-following and evidence-extraction ability to instantiate a partially correct state, giving RL a non-trivial signal from the beginning of training.**

Explicit elicitation (Harness-level Extraction) avoids a tool-routing bottleneck. Being able to construct a useful table does not mean that a policy will reliably choose to maintain one during search. This distinction is important for VISTA : if state maintenance itself depends on a learned tool-routing decision, then the intermediate state may be absent precisely at the beginning of training, when it is needed to provide supervision. We therefore isolate state

Table 4: **Zero-shot feasibility of intermediate state construction.** Untrained base policies can already externalize partial search evidence into structured and grounded tables, providing a non-trivial starting signal for VISTA. Table: response contains a parseable markdown table; Schema: share of emitted columns that match the requested columns; Grounded: share of emitted rows whose key values appear in the shown tool result; Key recall: share of the visible gold keys that reach the table; Cell F1: cell-level F1 against the full ground truth under VISTA's deterministic scorer (an exact-substring matcher, so a lower bound).

Base model	Table (%)	Schema (%)	Grounded (%)	Key recall (%)	Cell F1 (%)
Qwen3-4B	100.0	95.2	74.6	77.1	29.0
Qwen3-8B	100.0	98.6	75.5	78.3	29.7
Qwen3-14B	100.0	99.0	75.0	76.2	31.8
Llama-3.1-8B-Instruct	100.0	99.1	78.7	79.9	24.4
Gemma-4-E4B-It	92.7	95.1	80.8	70.3	24.9

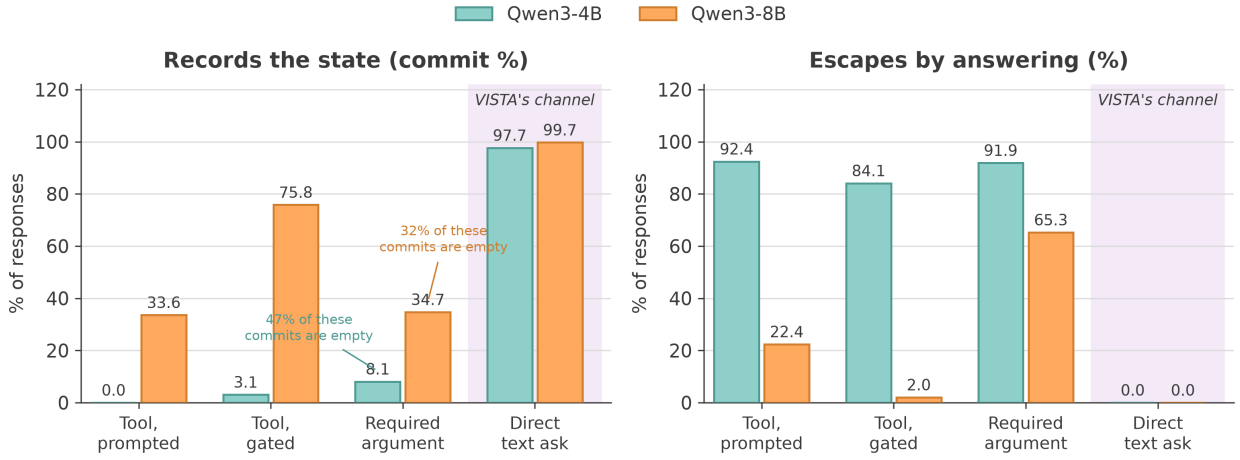

Figure 4: **State construction is easy, state routing is the bottleneck.** Qwen3-4B and Qwen3-8B are evaluated on the same 100 frozen mid-rollout states (4 samples each), while varying only how state recording is elicited: a prompted `update_table` tool, a gated tool, a required retrieval argument, or VISTA 's harness-level direct elicitation. **Left:** tool-based interfaces yield low state commit rates, while harness-level elicitation reaches 97.7% and 99.7%; forcing state through a required argument additionally produces many empty commits. **Right:** when tool-based recording is skipped, models frequently proceed directly to the final answer. These results identify policy-side routing, rather than table construction, as the primary bottleneck and motivate VISTA's harness-level state elicitation.

construction from state routing by presenting the same frozen search states under different recording interfaces, ranging from an optional `update_table` tool to a harness-level direct text request.

Figure 4 reveals a clear interface-dependent gap. When state recording requires a separate tool-routing decision, the policy frequently bypasses the update and proceeds directly to answering; for Qwen3-4B, only 0 to 8% of responses commit state across the tested tool-based interfaces. Simply forcing the update into the retrieval API does not solve the problem: making table rows a required argument instead produces empty commits in up to 47% of cases.In contrast, when the harness explicitly requests the same extraction after the tool observation, state emission rises to 97.7 to 99.7%. Crucially, the underlying evidence and extraction task are held fixed across these conditions: only the mechanism by which the policy is asked to expose the state changes. This contrast identifies routing, rather than state construction, as the practical bottleneck. VISTA therefore deliberately removes state elicitation from the policy's action space: after each tool observation, the harness requests the intermediate table directly. The policy remains responsible for what evidence to search for and what information to record, while the harness determines only when the current state is externalized. **This separation makes the intermediate state consistently observable from the start of training, without requiring an additional state-maintenance behavior to first emerge through supervised warm-up or RL.**

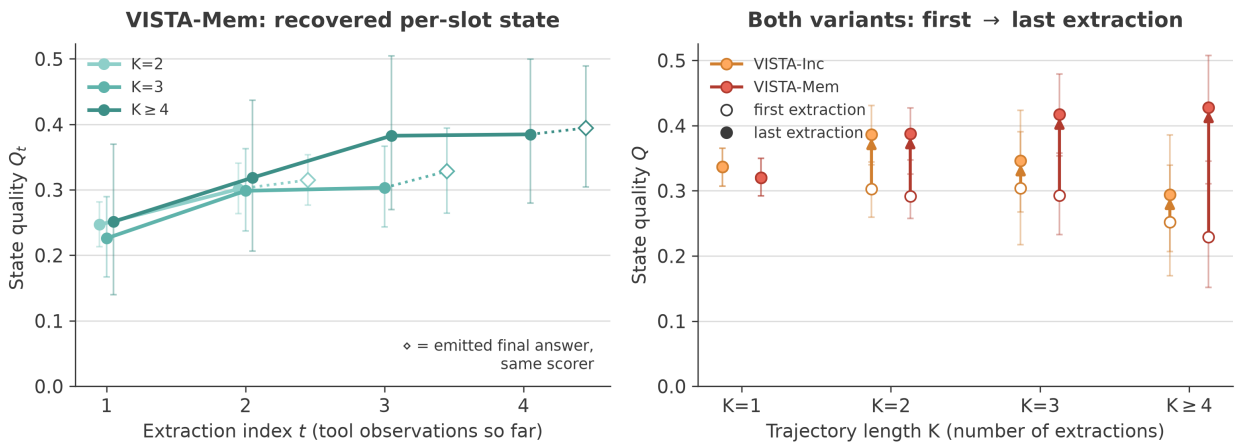

Figure 5: **Intermediate states expose measurable search progress.** Training rollouts from VISTA-Inc and VISTA-Mem (n = 1,088 per variant). **Left:** VISTA-Mem state quality increases with successive extractions, with the last state approaching final-answer quality **Right:** both variants improve from the first to the last extraction, with larger gains on longer trajectories. Together, the results show that intermediate states form partial, evolving representations of search progress, providing a measurable signal for turn-level attribution.

Intermediate states expose a genuine process signal. A reliably elicited state is useful for turn-level credit assignment only if its quality changes meaningfully as search progresses. If the intermediate table were already complete after the first observation, or remained unchanged throughout the trajectory, repeatedly scoring it would provide little information about which interactions improved the search process. Figure 5 shows that this is not the case. For VISTA-Mem , state quality increases progressively with additional tool observations. The first extracted state is substantially incomplete, while subsequent states improve as more evidence is accumulated and the last state approaches the quality of the emitted final answer. The same first-to-last improvement is observed for both VISTA-Inc and VISTA-Mem, with larger gains on trajectories containing more extraction steps. This evolution provides precisely the process signal required by VISTA: successive interactions induce measurable changes in a verifiable state, allowing the contribution of turn t to be characterized through changes. Thus, **the intermediate table is neither a static auxiliary output nor an early copy of the final answer, but an evolving representation of accumulated search progress.** The state further provides supervision where terminal rewards become uninformative. Among trajectories with no parseable final answer, for which the outcome reward is zero, 75% (27/36) nevertheless retain a valid and scoreable intermediate table. Intermediate states can therefore preserve verifiable evidence of partial progress even when terminal answer generation fails.

Takeaway. These preliminary studies motivate intermediate table states as a practical interface for process supervision. Base policies can already construct partially correct states without training; harness-side elicitation makes this capability reliable without introducing an additional tool-routing problem; and the resulting states evolve with search progress while retaining useful information even when terminal supervision fails. VISTA builds on these properties to transform changes in verifiable state quality into turn-level credit for Agentic RL.

B.2 Why Turn-Level Credit Need the Terminal Dominance

Having established that intermediate tables provide a measurable process signal, a second question remains: **how should this signal be used to assign credit across turns?** A natural approach is to reward each search turn whenever it retrieves useful ground-truth information. Our preliminary experiments reveal a critical failure mode of this strategy: process credit can successfully improve the behavior it directly rewards while simultaneously weakening the terminal behavior that determines task success. This observation motivates the second principle behind VISTA : turn-level credit must remain anchored to the terminal objective.

Rewarding acquisition improves search, but hurts the final outcome. We first construct turn-level rewards by string matching ground-truth facts against retrieved tool observations. Note that these experiments are conducted on a subset of the WideSearch benchmark and serve only as a preliminary diagnostic study. All reward variants considered here ultimately produce a single trajectory-level scalar reward: although some incorporate process-related signals such

Table 5: **Retrieval-centric process rewards improve acquisition but can weaken terminal behavior.** All variants use the same training recipe and data and differ only in reward design; results are measured on 50×4 held-out trajectories per variant. Utilization is the fraction of retrieved ground-truth facts that appear in the final table; Waste= 1 − Utilization; and Duplicate measures retrievals whose facts are already present in context. String-match rewards substantially increase search but also increase redundancy and waste while reducing final-table emission and F1. Utilization-aware reward largely repairs these behaviors but does not improve F1 over outcome-only training, motivating process credit that remains anchored to the terminal objective.

Reward	Utilization (%)	Waste (%)	Searches/traj.	Duplicate (%)	Table Produced (%)	Item F1	Disc. F1
Outcome only (baseline)	63.3	36.7	1.04	0.04	97.5	37.01	57.77
String-match (entity)	49.0	51.0	5.85	50.7	82.0	32.46	49.33
String-match (entity–attribute)	47.2	52.8	4.15	36.8	74.5	33.45	52.35
Utilization-aware	61.1	38.9	2.35	19.4	96.5	36.35	58.42

as retrieval or evidence utilization, none uses VISTA's explicit intermediate table state or performs turn-level process credit assignment. As shown in Table 5, the policy responds strongly: searches per trajectory increase from 1.04 under outcome-only GRPO to around 5. However, much of this additional search is redundant or unused. Duplicate retrieval rises from 0.04% to 36.8 to 50.7%, while the fraction of retrieved facts that never reach the final table increases from 36.7% to around 51%. At the same time, final-table emission falls from 97.5% to 74.5 to 82.0%, accompanied by a substantial decrease in final F1. In effect, retrieval-based rewards act as acquisition counters: they encourage collecting relevant evidence without ensuring that it is converted into the scored output.

Fixing evidence utilization is not sufficient. The increased waste suggests that poor evidence utilization might explain the performance degradation. We therefore introduce a utilization-aware reward that explicitly favors retrieved facts that are subsequently incorporated into the answer. The utilization-aware arm replaces the string-match process reward with one that pays a search turn only for retrieved facts that actually end up used in the final table, penalizing wasted and duplicate retrieval instead of rewarding retrieval volume. This intervention largely repairs the targeted behavior: relative to the retrieval-based string-match reward, waste decreases by 12.1 percentage points, duplicate retrieval decreases by 31.3%, and final-table emission recovers from 82.0% to 96.5%. Yet final performance remains statistically indistinguishable from the outcome-only baseline. This result suggests that evidence acquisition and utilization are not sufficient objectives on their own. The terminal step remains a critical bottleneck: regardless of how much useful evidence has been collected, the trajectory receives little or no task reward if that evidence is not successfully synthesized and emitted as the final table. Process supervision must therefore improve intermediate behavior without diluting optimization pressure on this terminal objective.

Motivation for terminal-dominant attribution. This diagnosis motivates a simple principle: **process credit should refine the terminal objective, not compete with it**. Rather than rewarding retrieval or utilization as separate objectives, VISTA measures progress in the same task-relevant representation used for the final outcome. For an intermediate state $\mathcal{T}_t$ with quality Q_t, earlier turns receive credit for verifiable progress in this state, while the terminal outcome remains the primary anchor of the trajectory objective. We refer to this principle as **terminal-dominant attribution**: **intermediate turns receive credit for verifiable progress, while terminal credit is preserved to protect final-answer production.** section 4 formalizes the resulting estimator. Importantly, this design is not only motivated by the diagnostics above: the advantage-estimation ablation in Table 12 directly shows that generic turn-level attribution, which redistributes more credit toward earlier turns, underperforms our terminal-dominant estimator.

C Detailed Experimental Settings

C.1 Benchmarks and Evaluation Details

WideSearch (Wong et al., 2026). WideSearch is a structured information synthesis benchmark that requires agents to populate complete tables in response to natural-language queries. Unlike conventional search QA tasks that typically target a small number of answers, WideSearch requires exhaustive entity discovery, attribute verification, and schema-compliant synthesis across dozens of heterogeneous sources. The benchmark consists of 200 tasks, including 100 English and 100 Chinese instances. The official WideSearch evaluation considers three levels of task performance: (1) **Success Rate (SR)**, a strict binary metric indicating whether the generated table exactly matches the reference; (2) **Row F1**, which evaluates entity-level correctness by penalizing missing and spurious rows; and (3) **Item F1**, which measures fine-grained attribute correctness at the cell level after row matching.

We evaluate all methods on the same WideSearch tasks following the official evaluation protocol. For each task, we independently sample $K = 4$ trajectories. We report Avg@4, which averages performance across the four trajectories, and Max@4, which selects the best-performing trajectory for each task before averaging across tasks. We additionally report Pass@4 following the WideSearch evaluation setting. Unless otherwise specified, Item F1 is our primary evaluation metric.

To better diagnose where performance gains arise, we introduce two additional metrics that decompose the ability to **discover the correct rows** from the ability to **correctly populate them**. **Discovery F1 (Disc F1)** evaluates row identity based solely on primary-key matching, irrespective of the correctness of non-key cell values. **Completion Rate (Comp Rate)** measures, among the correctly discovered rows, the fraction of attribute cells that are correctly filled. Together, Disc F1 and Comp Rate separate entity discovery from attribute completion, exposing two distinct capabilities that are conflated by Item F1. We also report **Soft Success Rate (Soft SR)** as a relaxed counterpart to exact Success Rate. Given a threshold τ, $SR(\tau)$ measures the fraction of trajectories whose Item F1 is at least τ. Under this definition, exact Success Rate corresponds to the limiting case $\tau = 1$. We report Soft Success at both $\tau = 0.95$ and $\tau = 0.5$ to capture different levels of task completion. The former measures near-complete solutions, while the latter measures whether the agent can recover a substantial portion of the target information, providing a less brittle view of progress on challenging wide-search tasks.

DeepWideSearch (Lan et al., 2025). DeepWideSearch is explicitly designed to evaluate agents to integrate depth and width in information seeking. The benchmark comprises 220 meticulously curated questions spanning 15 diverse domains, featuring both Chinese and English queries with human-verified ground truths. The official DeepWideSearch evaluation considers both the depth and width metrics with the Avg@4, Max@4 and Pass@4. For the width part, it builds upon the evaluation framework of Wong et al. (2026), using the Success Rate, Row-level F1 and Item F1. For the depth part, it introduces the Column F1, which is computed as the F1 score over the unique columns in the table. These unique columns correspond to the core attributes of entities (i.e., rows) that uniquely identify them. They also introduce the Core Entity Accuracy (CE Acc.), serving as an additional indicator of deep reasoning capability. We use the original evaluation protocols faithfully.

GISA (Zhu et al., 2026) is a benchmark for General Information-Seeking Assistants containing carefully crafted queries along with their answers and human-annotated search trajectories. It integrates both exploring deep reasoning paths and aggregating information across diverse sources, evaluating an agent's ability to perform vertical investigation and horizontal summarization within complex, long-horizon tasks. The benchmark has 373 human-crafted queries that reflect authentic information-seeking scenarios. Since the benchmark contains various types of answers, they employ specific metrics for each type (item, set, list and table). They apply exact match (EM) as the strictest evaluation metric across all answer types. This metric assigns a score of 1 only if the generated result is completely identical to the ground truth; otherwise, the score is 0.

We evaluate on all 373 GISA queries following the official release. We reuse the authors' agent prompt and their evaluation script unmodified. We keep their ReAct scaffold with native function calling, a budget of 30 tool invocations per query, 8,192 output tokens per step, temperature 0.7, thinking mode enabled, and a single run per query, which is the protocol used in both their code and their reported results. Our implementation differs in the tool layer: instead of the official Serper search and Jina browse tool, we use the same Perplexity search and web-reading tools as in our other experiments, which return raw paginated page content directly to the agent. Because retrieval backends and page representations differ, our absolute scores are not directly comparable to the official leaderboard; they are comparable across the systems compared here, which share an identical tool layer and a shared retrieval cache.

Standard Multi-hop Search QA Benchmark: HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022) To test whether the process supervision we develop for wide search transfers beyond it, we evaluate on three standard multi-hop question answering benchmarks, all built over Wikipedia and all requiring the agent to chain evidence across several documents rather than read a single one. HotpotQA consists of crowd-written two-hop questions, split between bridge questions, which require following an entity from one passage into another, and comparison questions, which contrast two entities. 2WikiMultiHopQA extends this to two- to four-hop questions built from Wikipedia together with Wikidata relation triples, which makes its reasoning chains explicit and its questions more structurally varied. MuSiQue composes single-hop questions into two- to four-hop ones under filters that remove items solvable by shortcuts or disconnected reasoning, and is the hardest of the three. We hold it out entirely, using it only as an out-of-distribution test. Note that we evaluate our approach on 1000 randomly sampled instances from the validation sets of these three datasets following the setup in Gao et al. (2025).

C.2 Implementation Details

General Training Hyperparameters. All RL runs share a single recipe, unless otherwise specified. Training is conducted using the VeRL framework (Sheng et al., 2024). For GRPO, We sample $G = 8$ rollouts per prompt with a batch of 8 prompts per step (64 trajectories). Advantages are normalized by the mean and standard deviation within each prompt-level rollout group. We perform a single PPO epoch per batch with clipping range $\epsilon = 0.2$ and token-mean loss aggregation. The rain batch is 8 prompts and the PPO mini-batch is 8 (micro-batch 1 per GPU). Following common practice for search agents we disable the KL penalty entirely (use_kl_loss=false, $\beta = 0$), so the only training signal is the reward. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay 0.01. The learning rate is 1×10^{-6} with a linear warmup over the first 5% of steps followed by a constant schedule. Gradients are clipped to a maximum norm of 1.0. Training uses bfloat16 precision with FSDP. We train 2 epochs for each run, using colocated rollout and training (hybrid engine). Training rollouts are sampled with both temperature and top_p set to be 1.0 with vLLM. Rollout tensor parallelism is 1 for models up to 8B and 2 at 14B and above, with all other hyperparameters held fixed across scales. Each trajectory allows up to 8 assistant turns within a 20,480 token context window (4,096 prompt / 16,384 response tokens); tool responses are truncated to 2,048 tokens.

Reward Hyperparameters. The trajectory-level reward is $R_i = \lambda_{\text{out}} R_i^{\text{out}} + \lambda_{\text{proc}} R_i^{\text{proc}} + \lambda_{\text{fmt}} R_i^{\text{fmt}}$, where R_i^{out} is the embedding-based table F-score (here we use the WebLeaper reward (Tao et al., 2025)) of the final answer (BGE-M3 embeddings (Chen et al., 2024), cell-similarity threshold 0.75, recall-weighted with $\omega = 1.5$) and $R_i^{\text{fmt}} = \mathbb{1}[\text{a parseable table is emitted}]$ with $\lambda_{\text{fmt}} = 0.1$. The outcome-only baseline uses $(\lambda_{\text{out}}, \lambda_{\text{proc}}) = (1, 0)$; process-shaped variants use $(\lambda_{\text{out}}, \lambda_{\text{proc}}) = (0.7, 0.3)$.

VISTA Implementation Details. Intermediate table states are scored at every interaction turn, yielding a quality curve Q_t. Each turn receives $\tilde{A}_{i,t} = A_i + \ell_{i,t}$, where A_i is the group-normalized trajectory advantage and $\ell_{i,t}$ is the turn's local credit derived from Q_t. Non-terminal turns are rescaled by $\kappa = \min\big(1,\ (1-\delta)\,|\tilde{A}_{i,T}| \,/\, \max_{t<T} |\tilde{A}_{i,t}|\big)$ with $\delta = 0.1$, while the terminal turn keeps $\tilde{A}_{i,T}$ exactly, so intermediate credit can never dominate the final-answer signal.

Judge Model Details. The per-step LLM judge used by the process reward is DeepSeek-V4-Flash (DeepSeek-AI, 2026) served locally with SGLang (tensor parallel 8, W4A16 weights, 131,072 token context) behind an OpenAI-compatible /chat/completions endpoint.

Agent Details. The agent's search tool is the Perplexity Search API (Perplexity AI, 2026) (at most 20 results per call, 1,024 tokens per page) paired with a self-contained HTML/PDF-to-Markdown reader.

Training Data. Our RL training corpus is drawn from the width_20k subset of the public WideSeek-R1 synthetic training set (Xu et al., 2026). We first filter out instances whose target tables contain fewer than two rows or two columns. We then define a difficulty score based on search accessibility. For each query, we issue a single search and measure the fraction of ground-truth entities in the key column that appear in the returned snippets. We rank the instances according to this score and construct our filtered training set from the resulting ranking. For the multi-hop QA transfer experiments, we build a self-contained search environment using three standard benchmarks. Specifically, 2,000 HotpotQA (Yang et al., 2018) and 2,000 2WikiMultiHopQA (Ho et al., 2020) questions form the 4,000-question training set, while MuSiQue (Trivedi et al., 2022) contributes no training data and is held out entirely as an out-of-distribution test set. Answers, gold supporting sentences, and gold titles are retained only on the verifier side and never included in the model prompts.

Geneal Evaluation Details. At evaluation time we merge the FSDP checkpoint to HuggingFace format, serve it with SGLang at a 32,768 token context, and decode with temperature 0.6, top-p 0.95, top-k 20, and a 16,384 token cap, reporting the mean over 4 trials per instance with GPT-4.1 as the scoring judge following Wong et al. (2026). Qwen3.8-27B is served with vLLM at a 65,536 token context with temperature 1.0 and thinking enabled at low reasoning effort, so its numbers are not directly comparable to other model's results.

Reproduction Details. We train all models on 8 $\times$ H100 GPUs. We will release all the training and eval code and the checkpoints sooner.

C.3 BASELINES

Single Agent (training-free) (Wong et al., 2026). The untrained backbone driven by the benchmark's own single-agent scaffold: one model plans, searches, reads pages and writes the final table. This is the reference point that tells us how much of any gain comes from RL rather than from the harness. Released system prompt and tool descriptions unchanged, same tools as every trained model.

Multi Agent (training-free) (Wong et al., 2026). The WideSearch benchmark's multi-agent scaffold, in which a lead agent spawns sub-agents that each work on part of the table in parallel and returns their results for aggregation. It tests whether the width of the task is better handled by splitting it across agents than by one long trajectory. Released multi-agent prompt and create_sub_agents tool, up to four sub-agents concurrently, same decoding settings.

Search-R1 (Jin et al., 2025). Trains a search agent with RL using outcome supervision only: multi-turn rollouts interleaving generation and retrieval, loss masking on retrieved tokens, and a rule-based reward that is exactly final-answer correctness, without format term. The first two components are already standard in our framework, so reproducing it amounts to supplying an outcome-only reward. We use exact string match for the item-level correctness. No embedding model, no format bonus.

$$R_i = \text{ExactMatch}(\widehat{\mathcal{T}}, \mathcal{T}^*)$$

WebLeaper (Tao et al., 2025). WebLeaper uses a granular F-score as the outcome reward. For trajectory i, let $\mathcal{E}_i$ denote the set of entities in the final predicted table and $\mathcal{O}^*$ denote the ground-truth entity set. Rather than relying on exact string matching, WebLeaper defines an entity-level matching function $s(e, e') \in [0, 1]$ that measures the semantic similarity between a predicted entity $e \in \mathcal{O}_i$ and a ground-truth entity $e' \in \mathcal{O}^*$. The corresponding soft precision and recall are

$$\mathcal{P}_i = \frac{1}{|\mathcal{O}_i|} \sum_{e \in \mathcal{O}_i} \max_{e' \in \mathcal{O}^*} s(e, e'), \qquad \mathcal{R}_{i,c} = \frac{1}{|\mathcal{O}^*|} \sum_{e' \in \mathcal{O}^*} \max_{e \in \mathcal{O}_i} s(e, e').$$

These quantities are combined into a weighted F-score reward:

$$R_i = (1 + \omega^2) \frac{\mathcal{P}_i \cdot \mathcal{R}_{i,c}}{\omega^2 \mathcal{P}_i + \mathcal{R}_{i,c}}$$

where ω controls the relative emphasis on recall and precision. This reward provides a denser learning signal than exact-match accuracy by assigning partial credit to semantically matched entities while penalizing irrelevant predictions. Importantly, despite being granular at the entity level, R_i is computed from the final output and remains a single trajectory-level reward. Consequently, under standard GRPO, the same trajectory-level advantage derived from R_i is assigned to all policy-generated actions in trajectory i, without distinguishing which intermediate turns contributed to the final outcome. In our implementation, we set the threshold for the sematic similarity function to be 0.75, and the $\omega = 1.5$ since the failure mode for the WideSearch task is more about missing rows.

GRPO – WebLeaper + Fmt. Same reward as the WebLeaper (Tao et al., 2025) plus a bonus for a well-formed response.

$$R_i = \lambda_{\text{out}} (1 + \omega^2) \frac{\mathcal{P}_i \cdot \mathcal{R}_{i,c}}{\omega^2 \mathcal{P}_i + \mathcal{R}_{i,c}} + \lambda_{\text{fmt}} R_i^{fmt}$$

StepSearch (Wang et al., 2025). StepSearch introduces process supervision for multi-turn search by assigning a local reward to each retrieval round. For trajectory i, the reward at search turn t is defined as:

$$R_{i,t}^{\text{step}} = G_{i,t} - P_{i,t},$$

where $G_{i,t}$ measures the marginal information gain of the newly retrieved documents with respect to a set of ground-truth supporting documents $\mathcal{D}^* = \{d_1^*, \ldots, d_n^*\}$, and $P_{i,t}$ penalizes documents that have already been retrieved in previous search rounds. Specifically, if $c_{i,t,j}$ denotes the best similarity between the documents retrieved at turn t and the j-th ground-truth document, and $m_{i,t-1,j}$ is the best similarity observed before turn t, the information gain is:

$$G_{i,t} = \frac{1}{|\mathcal{D}^*|} \sum_{j=1}^{|\mathcal{D}^*|} \max\left(c_{i,t,j} - m_{i,t-1,j},\, 0\right).$$

In addition to the step-wise signal, StepSearch assigns a terminal reward R_i^{term} based on final-answer correctness and search-keyword matching. Following our unified GRPO implementation, we combine the terminal and process signals into the trajectory reward:

$$R_i = R_i^{\text{out}} + \lambda_{\text{proc}} \sum_{t \in \mathcal{S}_i} R_{i,t}^{\text{proc}},$$

where $\mathcal{S}_i$ denotes the set of search turns in trajectory i and λ_{step} controls the contribution of step-wise supervision. The corresponding group-relative advantage is:

$$\widehat{A}_i^{\text{StepSearch}} = \frac{R_i - \mu_g}{\sigma_g + \epsilon}, \qquad \mu_g = \frac{1}{G}\sum_{j=1}^{G} R_j, \qquad \sigma_g = \sqrt{\frac{1}{G}\sum_{j=1}^{G}(R_j - \mu_g)^2}$$

As in standard GRPO, $\widehat{A}_i^{\text{StepSearch}}$ is shared across all policy-generated tokens in the trajectory. The original StepSearch places $R_{i,t}^{\text{proc}}$ at the end of each search round and the global reward at the terminal token, and uses GAE with a learned value function to propagate these signals. Our implementation replaces this PPO-based optimization with GRPO for a controlled comparison under the same training framework. Consequently, while StepSearch provides fine-grained supervision when constructing R_i, it does not explicitly attribute the resulting group-relative advantage to individual turns. We faithfully follow the original code and set the $\lambda_{\text{proc}} = 0.6$.

Importantly, StepSearch is designed for **deep search**, where a task decomposes into a chain of hops and each hop comes with an annotated supporting passage, so process reward can be defined as the marginal utility of a retrieval round against a designated target. Wide search provides no such targets: the answer is a wide table of many independent entities, there are no gold passages, and the corpus is the live web. StepSearch therefore requires substituting a surrogate for the annotated evidence it assumes, we serialize each ground-truth row as its golden document, so that its per-round gain, while computed exactly as specified, now measures word-level resemblance between retrieved text and the answer row itself rather than the recovery of a designated supporting passage. Our method needs no such surrogate: it scores the policy's own intermediate table at every turn against the ground truth, a signal that is directly verifiable, and propagates it backwards through credit to attribute improvement to the turns that produced it, providing process supervision tailored to entity-intensive wide-search trajectories.

TIPS ((Xie et al., 2026)). TIPS is Turn-Level Information Potential Reward Shaping, which assigns dense rewards to each tool-call segment based on how much it increases a teacher model's log-lielihood of the correct answer. Let $\mathcal{Y}_i = \{y_i^{(1)}, \ldots, y_i^{(M)}\}$ denote the set of acceptable answers for task i, and let $S_{i,t}$ denote the interaction context after turn t. TIPS defines the answer potential as:

$$\Phi(S_{i,t}) = \log \sum_{m=1}^{M} p_{\text{teach}}\big(y_i^{(m)} \mid S_{i,t}\big)$$

and assigns turn t the information-shaping reward

$$R_{i,t}^{\text{proc}} = \alpha\,[\Phi(S_{i,t}) - \Phi(S_{i,t-1})]$$

where $\alpha > 0$ controls the strength of reward shaping. The resulting per-token reward is:

$$r_{i,k}^{\text{TIPS}} = r_{i,k}^{\text{out}} + \sum_{t=1}^{T_i} \mathbb{1}[k = b_{i,t}] R_{i,t}^{\text{proc}}$$

where $b_{i,t}$ is the token boundary of turn t, and $r_{i,k}^{\text{out}}$ is zero except at the terminal position, where it equals the final-answer correctness reward R_i^{out}. Equivalently, the trajectory return can be written as

$$R_i^{\text{TIPS}} = R_i^{\text{out}} + \sum_{t=1}^{T_i} R_{i,t}^{\text{proc}}$$

Unlike TIPS, which estimates progress implicitly through a teacher model's likelihood of a short gold answer, VISTA measures progress directly from verifiable changes in an explicit intermediate table state. Moreover, TIPS adds potential differences as shaping rewards, whereas VISTA uses table-state progress to construct turn-specific advantages and explicitly caps non-terminal credit to preserve terminal dominance.

CW-GRPO (Wang et al., 2026b). CW-GRPO uses an LLM judge to identify informative search rounds and reallocates the trajectory-level GRPO advantage accordingly. In the original method, each trajectory τ_i receives a binary outcome reward based on the exact match of its final answer: $R_i = \mathbb{1}\left[\hat{y}_i \in \mathcal{Y}_i\right]$, where $\hat{y}_i$ is the predicted answer and $\mathcal{Y}_i$ is the set of acceptable answers. The corresponding outcome-level advantage is:

$$\widehat{A}_i^{\text{out}} = \frac{R_i - \text{mean}_{j=1}^{G} R_j}{\text{std}_{j=1}^{G} R_j + \epsilon}.$$

For each non-terminal search round $t < T_i$, an LLM judge evaluates retrieval utility $u_{i,t} \in \{0,1\}$ and reasoning correctness $v_{i,t} \in \{0,1\}$, producing the contribution signal: $p_{i,t} = u_{i,t} v_{i,t}$. For successful trajectories, the normalized contribution weight is

$$c_{i,t} = \frac{\exp(\alpha p_{i,t})}{\sum_{t'=1}^{T_i-1} \exp(\alpha p_{i,t'})}$$

while failed trajectories use the uniform weight $c_{i,t} = 1/(T_i - 1)$. CW-GRPO then defines the round-level advantage as

$$\widehat{A}_{i,t}^{\text{CW}} = \begin{cases} (T_i - 1) c_{i,t} \widehat{A}_i^{\text{out}}, & t < T_i, \\ \widehat{A}_i^{\text{out}}, & t = T_i. \end{cases}$$

Unlike CW-GRPO, which infers round contributions using an external LLM judge, VISTA measures progress through verifiable changes in the explicit intermediate table state. Moreover, VISTA attributes both successful and failed trajectories using state progress and explicitly bounds non-terminal advantages to preserve terminal dominance.

Released Open Models. We include the `SearchR1-7B` (Jin et al., 2025) and `ASearcher-Web-14B` (Gao et al., 2025) checkpoints as our zero-shot baselines. We compare directly against these released checkpoints. `SearchR1-7B`[2] is based on a Qwen2.5-7B (Team et al., 2025a) backbone optimized with PPO under an exact-match reward. ASearcher (Gao et al., 2025) is an open-source framework for large-scale online reinforcement learning of LLM search agents, released together with model weights, training recipes, and data-construction pipelines. We evaluate `ASearcher-Web-14B`[3], the web-search variant of that release. It is initialized from Qwen2.5-14B and RL-trained to interleave reasoning with live web search and page browsing. We include these models as external, openly released search agents at the same parameter scale as our own model, so that they provide same-scale reference points for the single-agent setting. We run these openly released models as single agents under exactly the same harness as our own model, changing only the weights: the same agent loop, the same search and page-reading tools, the same evaluator judge, and the same decoding parameters. We also include the released `WideSeek-R1-4B`[4] model by Xu et al. (2026).

C.4 VISTA-MEM VS. VISTA-INC

VISTA-Inc and VISTA-Mem use the same table extraction call $\pi_\theta^{\text{table}}$, turn-level table scoring, and terminal-dominant GRPO update. They differ in how the extracted table is stored and used:

- Table storage. VISTA-Inc stores $\mathcal{T}_{i,t}^{\text{INC}}$ only in the harness and does not change the acting rollout's prompt. VISTA-Mem also appends the parsed table to prompt as a masked user-role memory $\mathcal{M}_{i,t}$. The table is therefore stored in both the harness and the rollout context.
- Information available to the acting policy. In VISTA-Inc, the next action is sampled from $\pi_\theta(\cdot \mid h_{i,t+1})$, where the history does not include the extracted table. In VISTA-Mem, it is sampled from $\pi_\theta(\cdot \mid h_{i,t+1}^{\text{MEM}})$, where $h_{i,t+1}^{\text{MEM}}$ includes $\mathcal{M}_{i,t}$. Thus, the memory can inform later actions during the rollout.
- Use of the table during training. Both variants score the extracted tables and use those scores to assign turn-level credit through the same GRPO update. In VISTA-Inc, this training signal is the only way the extracted table affects the acting policy. In VISTA-Mem, the table also affects later actions because it is visible in the rollout context.
- Previous table in the next extraction call. VISTA-Inc explicitly adds $\mathcal{T}_{i,t-1}^{\text{INC}}$ to the next extraction prompt through an amend instruction. VISTA-Mem copies the current rollout context, which already contains earlier memories, and therefore needs no separate carry instruction.

[2]Please refer to PeterJinGo/SearchR1-nq_hotpotqa_train-qwen2.5-7b-it-em-ppo
[3]Please refer to inclusionAI/ASearcher-Web-14B
[4]Please refer to RLinf/WideSeek-R1-4b

- Context growth. VISTA-Inc adds no table tokens to the acting rollout. VISTA-Mem appends one memo after each extraction turn, so its context grows with the number of extraction turns. Earlier memos are kept because removing their tokens would break the alignment of response mask and the token log-probabilities used to compute importance ratios.

Table 19 shows VISTA-Mem reaching every crossed quality bar in fewer steps and fewer wall-clock hours than VISTA-Inc, because its in-context memory hands the policy a pre-digested working state at each turn, whereas VISTA-Inc can only benefit from the extracted table indirectly, through gradient updates on a state it must still re-derive from raw observation history.

D More Experimental Results

D.1 WideSearch Benchmark

VISTA-Mem vs. VISTA-Inc . VISTA-Mem outperforms VISTA-Inc in all tested backbones, including Qwen3-4B / 8B and Llama-3.1-8B-Instruct, except Qwen3-14B. Both variants use the same table extraction and turn-level credit assignment. The key difference is whether the agent can see the extracted table when choosing its next action. VISTA-Mem adds the latest table to the rollout as a masked user-role memo. The agent can then use it to track what it has found, decide what to search for next, and prepare its final answer. In VISTA-Inc , the table remains in the training harness and is not visible to the acting agent; it affects the policy only through the turn-level reward during training. This difference is also reflected in training: Table 19 shows that VISTA-Mem reaches each reported quality threshold in fewer steps and less wall-clock time. Together, these results suggest that turn-level credit works better when the agent can also use the intermediate state while acting.

Single-Agent vs. Multi-Agent Execution. VISTA-Mem consistently improves the base models under the single-agent setting, but directly transferring the learned policy to a multi-agent scaffold can hurt. Importantly, this is not a limitation of multi-agent execution itself: the base models remain effective under the same scaffold and can even benefit from it. Instead, our analysis points to a mismatch between the RL-trained interaction policy and an inference interface that it never encountered during training. The main failure mode is context exhaustion. Smaller VISTA-Mem checkpoints interact much more aggressively with the multi-agent scaffold, issuing substantially more searches and webpage reads. Their trajectories therefore accumulate long tool observations and can exceed the context limit before producing the final table, causing the entire trial to receive zero reward. This effect decreases rapidly with model scale: the unanswered rate drops from 23% at 8B to 16% at 14B, and only 1% for the largest tested backbone. Page-reading volume follows the same trend. These results suggest that larger models better preserve general instruction-following and termination behavior when the learned policy is transferred to an unseen agent scaffold. Notably, the degradation does not indicate that VISTA-Mem weakens search or table construction. Among trajectories that successfully produce an answer, VISTA-Mem can still outperform the corresponding base model. Rather, the main bottleneck is whether the policy terminates before exhausting the context budget. Overall, these results highlight an important distinction between **policy improvement within the training scaffold** and **policy robustness under scaffold transfer**: VISTA-Mem provides strong gains in its native single-agent setting, while reliable transfer to substantially different multi-agent interfaces remains sensitive to model scale and families.

Trends across Models and Languages. We further verify whether the gains of VISTA-Mem hold across model families and languages. As shown in Table 8, VISTA-Mem consistently improves Item F1 over the corresponding base model on both EN and ZH for all three backbones. The gains are $+5.34/+5.71$ for Qwen3-4B, $+8.55/+4.45$ for Llama-3.1-8B-Instruct, and $+7.89/+6.74$ for Gemma-4-E4B-it on EN/ZH, respectively. Similar improvements are observed in Row F1, Disc F1, and Soft Success Rate. While ZH remains more challenging, especially for Llama-3.1-8B-Instruct, the improvement from VISTA-Mem is preserved across both languages. These results show that the effectiveness of our method is not specific to one language or model family.

D.2 DeepWideSearch & GISA Benchmarks

DeepWideSearch Results. We evaluate our trained checkpoints zero-shot, with no training on deep search related data. As Table 9 shows, the gains cover both wide and deep parts. On the wide part, VISTA-Mem improves Item F1 (Avg4) over the single-agent base at every scale (6.87% to 9.52% at Qwen3-4B, 9.69% to 13.18% at Qwen3-8B, 12.79% to 15.30% at Qwen3-14B), with matching gains in Row F1. On the deep part, CE Acc. (Avg@4) improves by 6.6 points at Qwen3-4B and 5.6 at Qwen3-8B, and Pass@4 improves at all three scales. The only flat cell is the 14B

Table 6: Main results on WideSearch Benchmark. Avg@4 and Max@4 are the per-instance mean and max over 4 trials, averaged over instances; Pass@4 is the per-instance max of the exact-table success rate. Disc F1 scores row identity only and Comp Rate is the fraction of cells correct within the rows a method found; both are Avg@4. For Success Rate: τ=.5 and τ=.95 count a trial as successful when its Item F1 reaches τ. Best value per column in **bold**.

Method	Item F1 (%)		Row F1 (%)		Row Discovery (%)		Success Rate (%)			
	Avg@4	Max@4	Avg@4	Max@4	Disc F1	Comp Rate	Avg@4	Pass@4	τ=.95	τ=.5
Search-R1-7B	28.29	36.81	6.10	10.41	47.13	59.41	0.00	0.00	0.12	18.75
ASearcher-Web-14B	12.22	30.85	3.20	8.99	44.87	22.65	0.00	0.00	0.50	9.62
Qwen3-4B										
Single Agent	31.08	41.74	7.22	12.84	51.59	55.47	0.00	0.00	0.50	25.12
Search-R1	30.56	44.41	8.69	16.09	46.87	52.62	0.00	0.00	0.62	26.88
SingleSeek-R1-4B	32.12	42.69	8.01	13.64	56.36	55.84	0.00	0.00	0.62	27.62
GRPO - WebLeaper	29.85	45.01	8.43	15.73	45.73	52.36	0.25	1.00	0.88	25.88
GRPO - WebLeaper+Fmt	32.16	46.17	9.15	16.20	50.35	53.37	0.25	0.50	1.00	30.12
StepSearch	33.37	42.71	8.39	14.89	54.29	60.59	0.25	1.00	0.88	26.38
TIPS	32.95	41.66	8.72	13.78	51.54	55.91	0.00	0.00	0.53	27.85
CW-GRPO	32.32	42.60	6.59	11.55	55.44	53.66	0.25	0.50	0.50	25.62
VISTA-Inc	35.57	46.30	9.10	15.56	57.09	59.08	0.12	0.50	**1.00**	31.88
VISTA-Mem	**36.61**	**46.68**	**9.54**	**16.30**	**58.10**	**60.69**	0.25	0.50	0.75	**34.75**
Llama-3.1-8B-Instruct										
Single Agent	27.29	40.80	7.46	13.72	43.01	46.33	0.00	0.00	1.12	24.88
GRPO - WebLeaper	30.89	39.55	7.14	11.09	51.37	53.15	0.00	0.00	0.62	25.37
GRPO - WebLeaper+Fmt	31.73	40.30	**9.58**	13.34	49.84	56.62	0.12	0.50	**1.38**	27.75
StepSearch	30.59	37.52	7.42	11.52	51.51	56.61	0.00	0.00	0.62	22.12
TIPS	32.82	45.34	8.30	14.57	52.56	52.36	0.12	0.50	0.38	29.38
CW-GRPO	28.15	38.74	5.49	9.60	49.37	47.27	0.00	0.00	1.00	22.38
VISTA-Inc	28.93	37.54	6.92	11.29	48.75	48.93	0.00	0.00	0.88	24.50
VISTA-Mem	**33.79**	**42.36**	8.52	**14.00**	**55.25**	**57.14**	0.12	0.50	1.12	**30.25**
Qwen3-8B										
Single Agent	34.23	44.98	8.80	16.71	55.50	16.07	0.25	1.00	0.38	28.88
Multi Agent	32.89	48.06	8.89	17.29	51.20	13.96	0.62	1.00	0.88	29.00
GRPO - Webleaper+Fmt	20.76	25.66	1.84	3.66	43.79	45.24	0.00	0.00	0.00	12.38
VISTA-Inc	37.93	46.25	9.70	15.52	59.65	63.01	0.12	0.50	**0.75**	33.50
VISTA-Mem	**39.23**	**48.04**	**10.12**	**16.66**	**60.36**	**64.78**	0.00	0.00	0.62	**37.00**
Qwen3-14B										
Single Agent	32.67	44.04	9.24	15.50	56.13	18.80	0.00	0.00	0.62	28.88
Multi Agent	34.18	48.37	10.23	19.59	57.48	19.80	0.38	1.50	0.88	30.75
GRPO - Webleaper+Fmt	41.43	50.63	12.54	19.89	62.99	65.19	0.00	0.00	1.12	38.88
VISTA-Inc	**42.24**	52.02	12.67	19.64	**63.79**	**65.98**	0.00	0.00	1.00	40.38
VISTA-Mem	41.71	**52.28**	**13.14**	**21.37**	63.62	65.94	0.00	0.00	**2.00**	**41.00**
Qwen3.8-27B										
Single Agent	43.36	62.50	30.65	46.04	48.69	50.71	2.88	7.00	13.00	47.62
GRPO - Webleaper+Fmt	63.62	73.28	34.64	46.92	77.41	77.91	2.88	7.00	**15.38**	69.12
VISTA-Mem	**65.25**	**74.08**	**37.73**	**49.28**	**78.25**	**78.72**	3.25	6.00	14.00	**71.75**
Gemma-4-E4B-it										
Single Agent	37.99	47.31	13.05	20.98	56.42	65.31	0.50	2.00	1.50	35.00
GRPO - Webleaper	44.53	54.43	16.09	25.47	63.92	68.81	0.50	1.00	1.88	44.25
GRPO - Webleaper+Fmt	**45.75**	**56.26**	16.53	**26.48**	64.96	66.36	0.12	0.50	2.12	45.38
StepSearch	35.67	44.35	9.83	16.17	57.35	59.84	0.38	1.00	1.00	28.50
VISTA-Mem	45.30	54.70	**16.75**	25.96	**64.96**	**69.42**	0.62	2.00	**2.12**	**46.00**

average (52.50% vs. 52.05%). Success Rate stays near zero for every method, since it requires the entire table to be exactly correct, which reflects the difficulty of the benchmark.

GISA Results. GISA (Zhu et al., 2026) contains 373 human-crafted information-seeking queries over ten domains, each answered in one of four structured formats: a single *item*, an unordered *set*, an ordered *list*, or a *table*. The tasks are designed to mix depth and width: an agent must first gather a broad candidate pool and then verify each candidate

Table 7: Main results on WideSearch Benchmark. Avg@4 and Max@4 are the per-instance mean and max over 4 trials, averaged over instances; Pass@4 is the per-instance max of the exact-table success rate. Disc F1 scores row identity only and Comp Rate is the fraction of cells correct within the rows a method found; both are Avg@4. For Success Rate: τ=.5 and τ=.95 count a trial as successful when its Item F1 reaches τ. Best value per column in **bold**.

Method	**Item F1 (%)**		**Row F1 (%)**		**Row Discovery (%)**		**Success Rate (%)**			
	Avg@4	Max@4	Avg@4	Max@4	Disc F1	Comp Rate	Avg@4	Pass@4	τ=.95	τ=.5
Qwen3-8B										
Single Agent (Base)	34.23	44.98	8.80	16.71	55.50	16.07	0.25	1.00	0.38	28.88
Single Agent (VISTA)	39.23	48.04	10.12	16.66	60.36	64.78	0.00	0.00	0.62	37.00
Multi Agent (Base)	32.89	48.06	8.89	17.29	51.20	13.96	0.62	1.00	0.88	29.00
Multi Agent (VISTA)	31.72	48.35	9.20	17.81	48.34	14.71	0.38	1.50	1.50	31.25
Qwen3-14B										
Single Agent (Base)	32.67	44.04	9.24	15.50	56.13	18.80	0.00	0.00	0.62	28.88
Single Agent	41.71	52.28	13.14	21.37	63.62	65.94	0.00	0.00	2.00	41.00
Multi Agent (Base)	34.18	48.37	10.23	19.59	57.48	19.80	0.38	1.50	0.88	30.75
Multo Agent (VISTA)	36.59	51.30	10.93	19.90	54.48	17.45	0.25	0.50	1.38	35.50
Gemma-4-E4B-it										
Single Agent (Base)	37.99	47.31	13.05	20.98	56.42	65.31	0.50	2.00	1.50	35.00
Single Agent (VISTA)	45.30	54.70	16.75	25.96	64.96	69.42	0.62	2.00	2.12	46.00
Multi Agent (Base)	40.64	52.15	14.65	24.36	59.63	67.08	0.38	0.50	1.25	37.88
Multi Agent (VISTA)	43.23	54.04	13.41	22.53	64.73	65.49	0.25	0.50	1.62	42.38

Table 8: **Performance across English and Chinese splits of WideSearch Benchmark.** We compare the single-agent base model with VISTA-Mem across three model families. Δ denotes the improvement of VISTA-Mem over the corresponding base model.

Method	**Item F1** (%)			**Row F1** (%)			**Disc F1** (%)			**Soft SR** τ=.5 (%)		
	EN	ZH	Δ	EN	ZH	Δ	EN	ZH	Δ	EN	ZH	Δ
Qwen3-4B												
Single Agent (base)	34.83	27.33	+7.51	7.34	7.10	+0.24	57.03	46.15	+10.89	29.00	21.25	+7.75
VISTA-Mem	40.17	33.04	+7.13	9.57	9.52	+0.05	62.22	53.98	+8.25	40.25	29.25	+11.00
Δ (VISTA-Mem − base)	+5.34	+5.71	-0.37	+2.23	+2.42	-0.19	+5.19	+7.83	-2.64	+11.25	+8.00	+3.25
Llama-3.1-8B-Instruct												
Single Agent (base)	36.08	18.49	+17.60	9.89	5.03	+4.86	53.55	32.46	+21.09	37.00	12.75	+24.25
VISTA-Mem †	44.64	22.94	+21.69	11.95	5.08	+6.86	68.07	42.43	+25.64	44.50	16.00	+28.50
Δ (VISTA-Mem − base)	+8.55	+4.45	+4.10	+2.05	+0.05	+2.00	+14.52	+9.97	+4.55	+7.50	+3.25	+4.25
Gemma-4-E4B-it												
Single Agent (base)	43.64	32.34	+11.30	14.45	11.64	+2.81	63.94	48.89	+15.05	44.00	26.00	+18.00
VISTA-Mem †§	51.53	39.08	+12.45	19.72	13.77	+5.95	71.20	58.73	+12.46	57.00	35.00	+22.00
Δ (VISTA-Mem − base)	+7.89	+6.74	+1.15	+5.27	+2.13	+3.14	+7.25	+9.84	-2.59	+13.00	+9.00	+4.00

through deeper search. The overall score is the weighted average of exact-match (EM) scores over all four types, and it is a hard test: the best model reported in the original paper reaches only 19.30%. Evaluating our checkpoints zero-shot (Table 10), both variants improve Overall EM at every scale. The gains cover both ends of the depth, width spectrum. On the deep end, Item EM roughly doubles at every scale. Set and List scores also improve at Qwen3-8B and Qwen3-14B, though by smaller margins. On the wide end, Table Item-F1 improves 65.52% at Qwen3-8B and 21.77% at Qwen3-14B, with Table Row-F1 improving as well. **The results on the two benchmarks show that training on wide-search related data alone teaches behavior that transfers to tasks whose difficulty comes from the deep component as much as the wide one.**

Table 9: Main results on DeepWideSearch benchmark. CE Acc denotes the core entity accuracy and Column F1 computed as the F1 score over the unique columns (core attributes of entities) in the table.

Method	**Item F1 Score (%)**		**Row F1 Score (%)**		**Success Rate (%)**		**Column F1 (%)**		**CE Acc. (%)**	
Metric	Avg@4	Max@4	Avg@4	Max@4	Avg@4	Pass@4	Avg@4	Max@4	Avg@4	Pass@4
Qwen3-4B										
Single Agent	6.87	13.50	0.92	2.67	0.00	0.00	13.55	24.39	36.25	58.18
SingleSeek-R1-4B	7.69	12.74	1.03	1.99	0.00	0.00	14.93	22.98	37.95	54.09
GRPO - Webleaper+Fmt	3.47	9.21	1.35	3.65	0.11	0.45	5.05	13.74	**45.68**	62.73
VISTA-Inc	**9.81**	16.51	1.91	3.89	0.0	0.0	**17.09**	27.04	39.89	58.18
VISTA-Mem	9.52	**18.13**	**2.03**	**4.84**	**0.11**	**0.45**	15.58	**28.56**	42.84	**63.18**
Qwen3-8B										
Single Agent	9.69	17.19	2.00	5.05	0.00	0.00	17.11	27.86	39.77	60.00
VISTA-Inc	11.31	16.96	2.16	3.88	0.00	0.00	19.83	28.26	42.84	59.55
VISTA-Mem	**13.18**	**21.43**	**3.33**	**6.60**	0.00	0.00	**21.18**	**33.25**	**45.34**	**64.09**
Qwen3-14B										
Single Agent	12.79	19.67	3.04	5.96	0.00	0.00	21.46	31.39	**52.50**	67.73
VISTA-Inc	14.18	20.97	3.35	6.13	0.00	0.00	22.73	32.56	46.36	61.36
VISTA-Mem	**15.30**	**23.79**	**3.91**	**7.22**	**0.11**	**0.45**	**23.99**	**36.24**	52.05	**69.55**

Table 10: Main results on GISA benchmark. The overall score is calculated by the weighted average of all EM scores.

Method	**Overall EM**	**Item**	**Set**		**List**			**Table**		
		EM	EM	F1	EM	F1	Order	EM	Row-F1	Item-F1
Qwen3-4B (Single Agent)	2.95	13.64	8.00	45.39	4.17	35.00	30.03	0.79	15.90	29.56
+ VISTA-Inc	4.29	27.27	10.00	45.53	4.17	31.64	29.33	1.19	16.53	33.13
+ VISTA-Mem	4.02	22.73	8.00	42.21	6.25	36.24	31.25	1.19	15.02	29.83
Qwen3-8B (Single Agent)	4.29	9.09	10.00	34.41	8.33	25.97	23.43	1.98	13.78	24.97
+ VISTA-Inc	5.63	22.73	14.00	37.37	8.33	28.70	26.95	1.98	16.26	30.36
+ VISTA-Mem	5.09	18.18	12.00	45.10	6.25	35.97	33.32	2.37	21.08	41.33
Qwen3-14B (Single Agent)	4.02	18.18	10.00	40.48	4.17	37.63	33.39	1.58	21.29	37.99
+ VISTA-Inc	5.36	31.82	8.00	43.80	2.08	36.21	29.57	3.16	23.38	46.26
+ VISTA-Mem	5.36	27.27	12.00	48.18	6.25	38.38	34.44	1.98	23.50	41.84

D.3 Ablation Study

D.3.1 Ablation Study on the Reward Shaping and Credit Assignment

To further understand the effectiveness of our proposed method, we ablate VISTA's two components: the explicit intermediate state and turn-level credit assignment. All results are reported on WideSearch-EN following Lee et al. (2026).

Ablation Study on the intermediate state and the turn-level attribution. (1) Adding the intermediate state without turn-level attribution hurts: intermediate table quality is only weakly correlated with the final outcome, so when its reward is folded into the single trajectory-level scalar it perturbs the group-relative ranking that the outcome reward defines, and every token in the trajectory inherits the resulting mis-attributed advantage. The incremental variant is hit hardest because its carried state propagates early errors into all later Q_t. (2) Terminal-dominant turn-level attribution resolves this without changing the terminal reward itself: the same Q_t values, placed on the turns that produced them and capped below the terminal advantage, turn the intermediate state from a confound on the outcome signal into usable credit. When holding the reward fixed, moving it from trajectory to turn level achieves substantial improvements, achieving relative gains of 16.47% for Discovery F1, 14.77% for item F1.

Different Advantage Estimation Methods. Turn-level credit can be distributed over a multi-turn trajectory in several ways, and the choice matters more than the shaping signal it carries. Let A_{traj} be the GRPO group-normalized trajectory advantage, r_t the reward assigned to turn t, and T the terminal turn that writes the final table. We com-

Table 11: **Ablation Studies on the components of VISTA.** I.S. denotes the intermediate state, TA denotes the terminal dominant Turn-level Attribution. The best ones are in Bold.

Trajectory Reward Component			VISTA		Item F1 (%)		Row F1 (%)		Disc F1 (%)	
R_i^{out}	R_i^{proc}	R_i^{fmt}	I.S.	TA	Avg@4	Max@4	Avg@4	Max@4	Avg@4	Max@4
Trejectory-level Reward										
✔	✗	✗	✗	✗	29.85	46.73	8.07	15.21	44.68	69.81
✔	✗	✔	✗	✗	33.79	48.85	9.10	15.95	52.02	71.80
✔	✔	✗	✗	✗	36.35	46.70	8.42	14.91	58.42	71.10
✔	✔	✔	✗	✗	37.01	48.86	8.46	15.20	59.12	72.88
VISTA : Intermediate State										
✔	✔	✔	Inc	✗	33.60	48.85	9.68	17.73	51.58	70.75
✔	✔	✔	Mem	✗	35.00	48.92	9.60	17.37	54.03	71.43
VISTA : Turn-level Attribution										
✔	✔	✔	✗	✔	37.34	**51.06**	9.43	16.45	57.18	**74.64**
✔	✔	✔	Inc	✔	39.81	50.38	9.57	16.77	62.22	74.00
✔	✔	✔	Mem	✔	**40.17**	50.46	**9.66**	**17.85**	**62.93**	74.32

Table 12: **Ablation studies on advantage estimation methods.** All methods use the same intermediate reward and external memory mechanism; only the advantage assignment differs.

	Item F1 (%)		Row F1 (%)		Disc F1 (%)	
Credit Placement	Avg@4	Max@4	Avg@4	Max@4	Avg@4	Max@4
Trajectory-level with Memory	35.00	48.92	9.60	17.37	54.03	71.43
Reward-to-go	35.90	47.92	8.06	14.84	56.87	71.79
Terminal Preserved	36.96	47.26	8.43	14.13	59.90	71.69
Terminal Dominant (Ours)	**40.17**	**50.46**	**9.66**	**17.85**	**62.93**	**74.32**

pare three placements, all sharing the same reward and the same group statistics (trajectory totals), so that only the placement varies.

- **Reward-to-go** (the standard choice for turn-level credit assignment (Feng et al., 2026; Wang et al., 2026a)) sets $G_t = \sum_{t' \geq t} \gamma^{t'-t} r_{t'}$ and $A_t = (G_t - \mu_g)/\sigma_g$.
- **Terminal-preserved** sets $A_t = A_{\text{traj}} + r_t$ for $t < T$ and $A_T = A_{\text{traj}}$, protecting the terminal turn from dilution but not from being overtaken.
- **Terminal-dominant (ours)** defines the turn rewards, $l_t = r_t - \bar{r}$, forms $\tilde{A}_t = A_{\text{traj}} + l_t$ for all turns including T, and applies a single per-trajectory projection $\kappa = \min\big(1,\ (1-\delta)\,|\tilde{A}_T| / \max_{t<T} |\tilde{A}_t|\big)$, giving $A_t = \kappa \tilde{A}_t$ for $t < T$ and $A_T = \tilde{A}_T$.

Our diagnostics identify answer production as one main bottleneck (Appendix B.2). Credit assignment should therefore preserve a strong learning signal at the turn that produces the final answer. Standard placements do not ensure this. Reward-to-go assigns the terminal reward to earlier turns as well, so those turns receive credit for the final answer even when their local contribution is unclear. In contrast, terminal-dominant credit uses intermediate rewards to distinguish earlier turns while preserving the strongest assigned advantage for the answer turn. Terminal-preserved keeps the terminal advantage unchanged, but adding r_t to earlier turns can give a non-terminal turn a larger assigned advantage than the answer turn. VISTA's κ projection instead ensures that the terminal turn has the largest assigned advantage magnitude in every trajectory. It preserves the signs and relative ordering of the process signals on earlier turns, and reduces exactly to trajectory-level GRPO when no local signal is available. As shown in Table 12, terminal-dominant credit outperforms both reward-to-go and terminal-preserved placement. Because these comparisons use the same shaping reward and external memory mechanism, the improvement comes from where credit is assigned, rather than from a change in the reward.

Table 13: **Ablation of the scorer used to compute the intermediate quality signal Q_t.** We compare the LLM judge with a deterministic normalized string-matching scorer under otherwise identical training settings. Results are reported on the 100 held-out English WideSearch instances.

			Item F1 (%)		Row F1 (%)		Disc F1 (%)	
Method	**Q_t scorer**	**Steps**	Avg@4	Max@4	Avg@4	Max@4	Avg@4	Max@4
GRPO - WebLeaper + Format	Null	250	33.79	48.85	9.10	15.95	52.02	71.80
GRPO - StepSearch	Null	250	36.69	45.68	8.61	15.02	59.10	69.33
VISTA-Inc	Deterministic	250	37.94	47.57	9.11	15.38	60.13	71.84
VISTA-Inc	LLM judge	250	39.81	50.38	9.57	16.77	62.22	74.00
VISTA-Mem	Deterministic	250	38.74	48.79	8.93	15.22	61.99	73.61
VISTA-Mem	LLM judge	250	40.17	50.46	9.66	17.85	62.93	74.32

D.3.2 Removing the LLM Judge from the Reward

VISTA uses an LLM judge as part of its reward computation: at every turn, the judge scores the intermediate table to produce the quality signal Q_t. We conduct an ablation study to determine whether VISTA's gains primarily stem from this LLM-based judge, while also quantifying the additional cost and external dependency it introduces during training. To isolate the role of the judge, we replace it with a deterministic scorer while keeping all other components unchanged. The deterministic scorer measures how many ground-truth values appear in the intermediate table using normalized string matching. Thus, the reward uses the same ground-truth information but requires no LLM calls during training. All other settings are held fixed, making the computation of Q_t the only variable (Table 13).

The judge can be removed with little loss in quality. On the English WideSearch instances, the judge-free Memory variant achieves 38.74 Item F1 and 61.99 Discovery F1 (Avg@4), compared with 40.17 and 62.93, respectively, for its LLM-judged counterpart. The judge-free variant also retains VISTA's full advantage over the outcome-only GRPO baseline ($+5.0$ Item F1 and $+10.0$ Discovery F1, both significant). These results suggest that the primary source of VISTA's gains is the turn-level credit assignment rather than the particular verifier used to compute Q_t, although the LLM judge still provides a modest improvement in reward quality.

The LLM judge is relatively inexpensive but not free. It introduces 72 to 81 external API calls per training step, making training dependent on a live external service. Replacing it with the deterministic string matcher reduces this overhead to zero external calls. The judge is never used at inference time, so evaluation and inference costs remain unaffected.

D.3.3 Ablation Study on the Quality Score

We further study the choice of intermediate quality score Q_t used to measure progress in the table state. Table 14 compares Item F1 and Row F1 as the underlying quality signal. **Item F1 consistently provides the strongest supervision**, improving over the Single Agent baseline from 31.08 to 36.61 Item F1 and from 7.22 to 9.54 Row F1. Notably, using Item F1 as Q_t also outperforms directly optimizing Row F1 as the intermediate signal (9.54 vs. 8.37 Row F1). We attribute this difference to the granularity of the intermediate supervision. Item F1 measures progress at the cell level, providing dense credit whenever the agent correctly fills or updates individual entries. In contrast, Row F1 requires multiple cells within a row to be jointly correct and therefore provides a coarser signal for partially completed intermediate states. This makes Item F1 better suited for tracking incremental progress during long-horizon search, while still transferring to row-level and discovery-level performance.

D.3.4 Ablation Study on Training Data

Hardness-based selection. We construct the training set by first removing examples with empty ground-truth tables or with at most two rows and two columns. For each remaining query, we issue a single search and measure the fraction of ground-truth key-column entities recoverable from the returned snippet. We call this measure single-search key coverage. We retain examples with $\mathrm{key_cov} \leq 0.5$ and sample 1,000 of them for our default training set.

Training data scale. Table 15 compares hardness-selected sets of 1,000 and 5,000 examples using Qwen3-4B on the full WideSearch benchmark. The hard-1k model improves from 34.07 to 36.61 Item F1 and from 8.56 to 9.54 Row

Table 14: **Ablation of the intermediate quality score Q_t on WideSearch.** Item F1 provides a denser cell-level progress signal than Row F1 and yields the best overall performance, including on the downstream Row F1 metric.

		Item F1 (%)		Row F1 (%)		Disc F1 (%)	
Method	Q_t	Avg@4	Max@4	Avg@4	Max@4	Avg@4	Max@4
Single Agent	-	31.08	41.47	7.22	12.84	51.59	55.47
VISTA-Mem	Row F1	35.47	45.67	8.37	14.55	59.23	-
VISTA-Mem	Item F1	36.61	46.68	9.54	16.30	58.10	60.69
VISTA-Mem	Row F1 + Item F1	34.61	44.97	7.38	13.51	58.57	-

Table 15: **Ablation of training-data scale and selection on WideSearch.** Increasing the hardness-selected training corpus from 1k to 5k examples does not improve performance, while replacing the hard 1k examples with randomly sampled data consistently degrades performance. This supports our default choice of 1k hardness-selected examples. Results use Qwen3-4B on the full WideSearch benchmark.

Method	**Item F1 (%)**		**Row F1 (%)**		**Row Discovery (%)**		**Success Rate (%)**			
	Avg@4	Max@4	Avg@4	Max@4	Disc F1	Comp Rate	Avg@4	Pass@4	τ=.95	τ=.5
Single Agent	31.08	41.74	7.22	12.84	51.59	55.47	0.00	0.00	0.50	25.12
VISTA-Mem (1k Samples, 1 epoch) @125	34.07	44.91	8.56	15.28	54.83	58.38	0.00	0.00	0.38	29.88
VISTA-Mem (1k Sample, 2 epochs) @250	36.61	46.68	9.54	16.30	58.10	60.69	0.25	0.50	0.75	34.75
VISTA-Mem (1k Random Samples, 2 epochs) @250	34.00	44.97	8.26	14.88	55.45	57.40	0.12	0.50	0.75	30.75
VISTA-Mem (5k Samples, 1 epoch) @375	35.66	44.97	7.11	12.39	60.62	57.96	0.12	0.50	0.50	29.38
VISTA-Mem (5k Samples, 2 epochs) @750	31.14	46.98	8.51	15.63	48.89	50.16	0.12	0.50	0.88	27.25

F1 between 125 and 250 steps. At 250 steps, it also reaches 58.10 Discovery F1 and 34.75 success rate at $\tau = 0.5$. Scaling the selected set to 5,000 examples does not improve these joint outcomes under our training recipe. The reported 5k checkpoint at 375 steps reach 35.66/7.11 Item/Row F1; It does not match the hard-1k model at 250 steps on both metrics. Further training to 750 steps lowers Item F1 to 31.14 and Discovery F1 to 48.89. Although individual metrics occasionally improve, for example, the 750-step model reaches 46.98 Max@4 Item F1, the larger corpus does not produce a consistent gain across answer quality, row discovery, and success rate.

Effect of data selection. At the same 1,000-example size and 250-step budget, replacing the hardness-selected data with uniformly sampled examples reduces Item F1 from 36.61 to 34.00, Row F1 from 9.54 to 8.26, Discovery F1 from 58.10 to 55.45, and success rate at $\tau = 0.5$ from 34.75 to 30.75. The random subset has a median single-search key coverage of 0.74, close to the original pool's 0.76, whereas the selected examples have coverage at most 0.5. This comparison suggests that concentrating training on queries requiring broader search is more useful than uniformly sampling the same number of examples.

Overall, the hard-1k configuration provides the strongest balance of metrics in this ablation. Increasing the corpus to 5k examples does not yield a consistent improvement under the tested training recipe, while random sampling at the same size degrades performance.

D.4 Training Dynamics

Training Reward. Figure 6 shows the training reward (R_i) over the first 250 steps for VISTA-Inc and VISTA-Mem. Across all model backbones, training is stable: rewards consistently increase without collapse or divergence, with overall gains ranging from $+0.11$ to $+0.36$. Most improvement occurs early, with 40 to 70% of the total reward gain achieved within the first 50 steps; after roughly 80 steps, the curves largely plateau with only gradual improvements. Model scale primarily determines the final reward level: Llama-3.1-8B-Instruct reaches the highest reward (~ 0.85), while the Qwen3 models generally improve with scale. In contrast, VISTA-Inc and VISTA-Mem exhibit similar optimization behavior, finishing within 0.03 reward on shared backbones. Qwen3.8-27B with VISTA-Mem is a notable exception in convergence speed, starting from a lower reward but rapidly reaching 0.82 by step 100. **These results show that both VISTA variants optimize stably across model families and scales, with most reward improvement occurring early in training.**

Policy Entropy. Figure 7 shows the policy entropy during training, normalized by each run's average entropy over the first five steps. For most backbones, entropy remains relatively stable throughout training: the Qwen3 and Gemma models exhibit only moderate increases or decreases, with no evidence of systematic policy collapse. Qwen3.8-27B

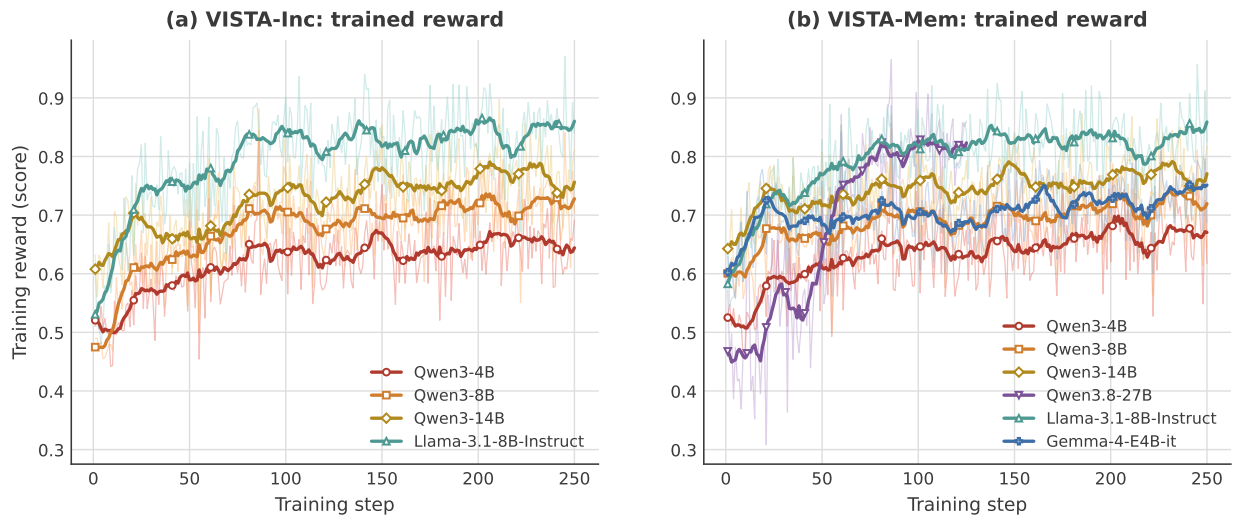

Figure 6: **Training reward dynamics of VISTA.** Training rewards (0.7outcome + 0.3state + 0.1format) over 250 RL steps for **(a)** VISTA-Inc and **(b)** VISTA-Mem across different model backbones. Both variants exhibit stable optimization, with most reward gains occurring early in training. Thin lines denote per-step rewards and thick lines denote centered rolling averages.

shows a temporary increase in entropy during its early stage of rapid reward improvement, before returning closer to its initial level. The main exception is Llama-3.1-8B-Instruct, whose entropy decreases sharply under both VISTA-Mem and VISTA-Inc, with most of the reduction occurring within the first 100 steps. Since this behavior consistently appears for the same backbone across both VISTA variants, while other backbones remain stable, it suggests that the entropy collapse is primarily backbone-dependent rather than specific to the VISTA state design. Interestingly, Llama still achieves the highest training reward despite this collapse, although its validation performance largely plateaus afterward. In summary, VISTA maintains stable policy entropy across most tested backbones, while optimization dynamics can differ substantially across model families.

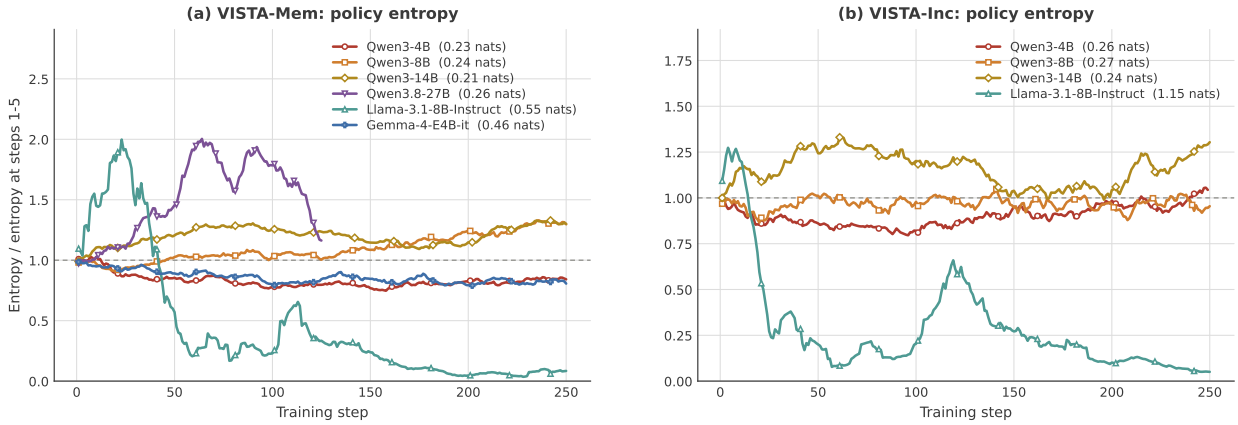

Figure 7: **Policy entropy during RL training.** Normalized policy entropy over 250 training steps for **(a)** VISTA-Mem and **(b)** VISTA-Inc across different model backbones. Entropy is normalized by each run's mean over the first five steps, with absolute initial entropy reported in the legend. Most backbones maintain relatively stable entropy, while Llama-3.1-8B-Instruct exhibits a substantial decrease under both variants.

Validation during Training. Figure 8 tracks validation performance every 25 training steps on `wideseek_hard_val150`, using the soft-F1 mean@2, which is an in-corpus diagnostic rather than the full WideSearch evaluation used for the main results. Across VISTA runs, validation performance generally improves together with the training reward,

indicating that the optimization gains transfer to unseen instances from the same data distribution. Most improvements occur early: the curves typically approach their final performance within the first 75–175 steps and remain relatively stable afterward. Performance is primarily determined by the model backbone, with Llama-3.1-8B-Instruct achieving the strongest final results and larger Qwen3 models generally outperforming smaller ones. VISTA-Mem and VISTA-Inc reach similar final performance on their shared backbones, suggesting that both intermediate-state designs support comparable optimization.

For Qwen3.8-27B, VISTA-Mem improves substantially earlier: at step 50, its validation soft-F1 reaches 0.615, compared with 0.236 for GRPO. GRPO subsequently catches up, reducing the gap to 0.847 versus 0.794 at step 125. This comparison suggests that VISTA's shaped turn-level supervision primarily improves early-stage learning efficiency rather than producing a fundamentally different optimization ceiling.

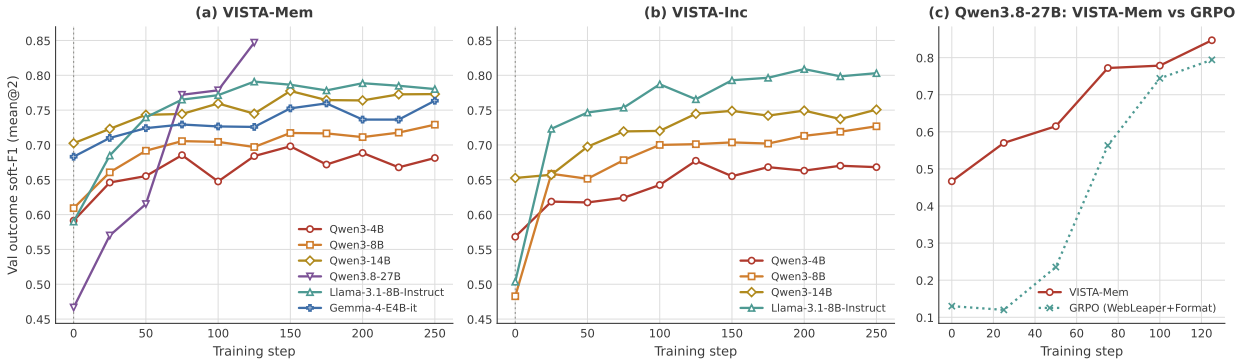

Figure 8: **Validation dynamics during RL training.** Soft-F1 mean@2 on validation dataset (50 instances with two rollouts each), evaluated every 25 training steps. **(a–b)** VISTA-Mem and VISTA-Inc across model backbones. **(c)** VISTA-Mem compared with GRPO (WebLeaper+Format) on Qwen3.8-27B. VISTA improves rapidly during early training, while performance generally stabilizes at later steps.

Table-Discovery Dynamics. Figure 9 examines how table construction changes during training from three complementary perspectives: table emission, content recall, and intermediate-state quality. Soft recall is the recall half of the WebLeaper outcome score, which is an embedding soft-F1 of the final table against the ground-truth facts (bge-m3, cosine threshold 0.75), so a cell counts as found if it matches in meaning, not only in exact wording. It measures how much of the required entities the answer contains.

First, table emission is nearly saturated from the beginning for most backbones, remaining above 97% throughout training. The main exception is Qwen3.8-27B, where both VISTA-Mem and GRPO initially struggle to produce valid tables and gradually recover during training. For most models, the performance gains cannot be explained simply by learning the required output format.

Second, soft recall generally improves with training, but VISTA does not consistently outperform GRPO on this metric. GRPO achieves comparable or higher final recall on Qwen3-4B and Qwen3-14B, while VISTA is stronger at the matched checkpoint on Qwen3-8B. This suggests that VISTA's benefit is not primarily driven by retrieving or including more ground-truth semantic like entities.

The clearest change appears in the quality of the intermediate table state. The judged state quality increases substantially across all tested backbones, from 0.07 to 0.38 on Qwen3-4B, 0.23 to 0.42 on Qwen3-8B, and 0.31 to 0.51 on Qwen3-14B. VISTA-Mem is also consistently comparable to or better than VISTA-Inc. Together, these results suggest that VISTA primarily improves how discovered information is maintained and organized in the intermediate state, rather than simply teaching the policy to emit tables or increasing raw content recall.

Search Behavior. Figure 10 examines how RL training changes search behavior by tracking the number of search calls and duplicate-query rate. Search volume itself does not exhibit a consistent trend: larger backbones substantially reduce their number of searches during training, while Qwen3-4B increases it. This suggests that the number of searches alone is not a reliable indicator of a better search policy.

A clearer separation appears in search redundancy. Across the VISTA runs for which query-level traces are available, duplicate-query rates remain consistently low, ending below 0.03. In contrast, the observed GRPO runs exhibit two

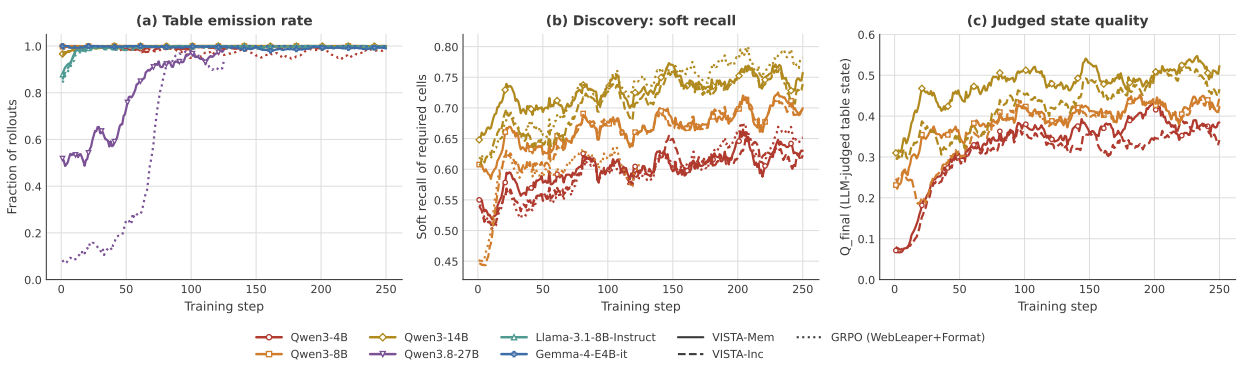

Figure 9: **Table-discovery dynamics during RL training.** We track **(a)** valid table emission rate, **(b)** soft recall of discovered ground-truth entities, and **(c)** judged quality of the final intermediate table state. Table emission is largely saturated from the beginning and content recall improves moderately, while VISTA substantially improves the quality of its maintained table state during training.

distinct failure modes. On Qwen3-8B, GRPO nearly stops using search, decreasing from 3.3 calls per trajectory to almost zero. On Qwen3-4B and Llama-3.1-8B, GRPO instead develops increasingly repetitive search behavior; most notably, Llama-3.1-8B reaches over 20 searches per trajectory and a high fraction of trajectories contain repeated queries. Neither behavior is observed in the corresponding VISTA runs.

Importantly, these behavioral differences are not apparent from reward or validation curves alone: for example, Llama-3.1-8B with GRPO maintains competitive reward and validation performance despite highly repetitive search. These results suggest that VISTA's turn-level supervision helps maintain a more controlled and less redundant search policy, rather than simply encouraging more tool use.

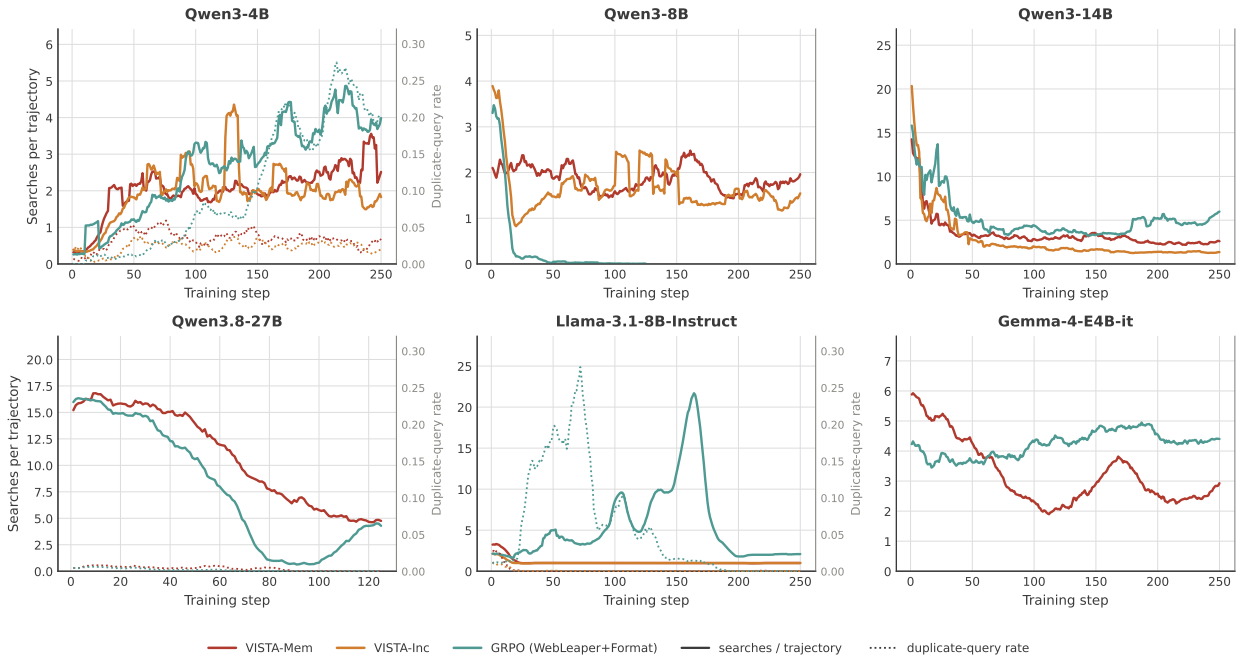

Figure 10: **Search behavior during RL training.** For each model backbone, we track the average number of search calls per trajectory (solid lines, left axis) and the fraction of duplicate queries (dotted lines, right axis). Search volume changes substantially across backbones without a consistent direction, while VISTA maintains low query redundancy. The observed GRPO runs exhibit two failure modes: near-zero search on Qwen3-8B and increasingly repetitive search on Qwen3-4B and Llama-3.1-8B.

Table 16: Inference cost per trajectory on the WideSearch benchmark using Qwen3-4B backbone. Turns is the number of assistant steps; Decode is generated tokens summed over turns; Obs. is tool-result tokens fed back; Context is the context length at the final turn, including prompt, decode, and observation tokens. All are means over trajectories.

Method	**Turns**	**Tool calls**		**Decode**	**Obs.**	**Context**
		Search	Read	(Tok)	(Tok)	(Tok)
Single Agent	1.9	1.8	0.0	5,350	7,053	12,611
Search-R1	7.4	5.2	3.4	15,458	31,339	47,004
StepSearch	2.6	1.4	0.7	4,574	10,143	14,925
GRPO – WebLeaper	2.0	2.1	0.1	6,339	8,678	15,224
GRPO – WebLeaper + Format	6.9	4.6	2.7	15,518	29,374	45,100
VISTA-Inc	3.0	2.1	0.7	8,074	12,427	20,708
VISTA-Mem	3.6	2.9	1.4	5,840	18,538	24,585

Table 17: Inference cost per trajectory on the multi-hop QA benchmarks using Qwen3-4B backbone. Turns is the number of assistant steps; Decode is generated tokens summed over turns; Obs. is tool-result tokens fed back; Context is the context length at the final turn, including prompt, decode, and observation tokens. All are means over trajectories.

Method	**Turns**	**Tool calls**		**Decode**	**Obs.**	**Context**
		search	read	(tok)	(tok)	(tok)
GRPO	3.0	1.63	0.43	3,076	758	3,854
VISTA	2.5	1.41	0.23	1,736	633	2,389

D.5 Efficiency Analysis

VISTA often learns faster and uses inference budget more productively.

Inference Efficiency We measured exact rollouts using the Qwen3-4B tokenizer on 800 WideSearch Benchmark trajectories and produced the inference efficiency results as shown in Table 16. VISTA-MEM is the best method in Table 1 and the cheapest run vs the GRPO + Format baseline, it's 2.66 $\times$ fewer decode tokens, 1.93 $\times$ fewer turns, 1.83 $\times$ smaller context. The quality win is not bought with test-time compute. Where it does spend more is observation tokens (18.5K vs 12.4K for INC): the memory variant trades generated tokens for retrieved ones, which is the favorable direction since obs is prefill and generation is decode.

Training Efficiency At equal steps VISTA costs more to train (14–16 h vs 11.6 h, the judge call), and pays it back on quality and on inference cost. Table 18 shows VISTA costs more to train per step (its per-turn judge call makes a step 20% more expensive). But it achieves higher held-out quality and improves how agents spend their inference budget. VISTA-Mem reaches the best quality on the board with 2.7$\times$ fewer decoded tokens and 1.9$\times$ fewer turns per trajectory than the GRPO baseline.

VISTA accelerates early-stage policy learning. Table 19 shows that VISTA, particularly VISTA-Mem , reaches intermediate quality levels substantially faster than standard GRPO. At the 0.630 quality threshold, VISTA-Mem requires only 86 steps compared with 163 for GRPO, corresponding to a 1.90$\times$ reduction in training steps and a 1.57$\times$ wall-clock speedup despite the additional cost of intermediate-state evaluation. VISTA-Inc provides a smaller improvement, reaching the same threshold in 104 steps, but its higher per-step overhead limits the corresponding wall-clock gain. The acceleration is most pronounced at intermediate quality levels and diminishes at higher thresholds: neither VISTA variant improves over GRPO at 0.648, and neither reaches GRPO's final 0.676 training score within its training budget. Together, these results suggest that **verifiable intermediate-state credit primarily improves learning efficiency, enabling the policy to acquire useful search behavior earlier, rather than simply maximizing the terminal training reward.**

Table 18: Training Cost per trajectory on the WideSearch benchmark using Qwen3-4B backbone. VISTA-Mem makes a step 20% more expensive and pays that back twice: higher held-out quality and roughly a third of the generation at inference. Pink marks the best cell in each row.

Qwen3-4B, 250 GRPO steps		**GRPO**	**VISTA-Inc**	**VISTA-Mem**
Training Cost	Steps	250	250	250
	Wall-clock	11.6 h	16.2 h	14.0 h
Held-out Quality	Item F1	32.16	35.57	**36.61**
	Disc F1	50.35	57.09	**58.10**
Inference Cost	Turns	6.9	3.0	3.6
	Decode (Tok)	15,518	8,074	**5,840**
	Context (Tok)	45,100	20,708	24,585

Table 19: **Training efficiency of GRPO and VISTA variants.** Steps and wall-clock hours denote the first time each method reaches a given training-quality threshold.

Quality Bar	**GRPO**		**VISTA-INC**		**VISTA-MEM**	
	Steps	Hours	Steps	Hours	Steps	Hours
0.610	86	4.0	86	5.6	60	3.4
0.619	93	4.3	86	5.6	74	4.1
0.630	163	7.5	104	6.8	86	4.8
0.648	197	9.1	211	13.7	203	11.3
Best Speedup	—		1.57× / 1.12×		**1.90× / 1.57×**	

D.6 Hyperparameter Sensitivity

VISTA is not sensitive to precise Reward Weight Tuning We vary the relative weighting of the outcome and per-turn process terms while keeping their total weight fixed, with $\lambda_{\text{out}}/\lambda_{\text{proc}} \in 0.9/0.1, , 0.7/0.3, , 0.5/0.5$ and a fixed format weight of 0.1. As shown in Table 20, performance remains remarkably stable across this range. Increasing the process weight from the default $0.7/0.3$ to $0.5/0.5$ produces no statistically significant change on any metric, while reducing it to $0.9/0.1$ degrades several metrics, suggesting that an overly weak process signal might be insufficient to provide effective turn-level credit. These results show that the benefit of VISTA is robust to the exact mixing coefficient once the process term is assigned sufficient weight. This reduces sensitivity to hyperparameter tuning while also confirming that the intermediate process signal makes a substantive contribution to learning.

Table 20: **Hyperparameter sensitivity to reward weights for VISTA .** We vary the relative weights of the outcome and process rewards while training all variants for one epoch. We additionally report the default reward configuration trained for two epochs as a reference for the effect of longer training.

Reward Weights			**Training**	**Item F1 (%)**		**Row F1 (%)**		**Disc F1 (%)**	
λ_{out}	λ_{proc}	λ_{fmt}	Epoches	Avg@4	Max@4	Avg@4	Max@4	Avg@4	Max@4
VISTA-Mem									
0.7	0.3	0.1	2	40.17	50.46	9.66	17.85	62.93	74.32
0.5	0.5	0.1	1	34.82	45.03	8.23	14.71	57.24	70.17
0.7	0.3	0.1	1	35.24	45.28	8.66	14.53	57.47	70.28
0.9	0.1	0.1	1	33.88	44.66	7.64	13.61	56.92	70.64
Trejectory-level Reward (Reference)									
1.0	0.0	0.0	2	29.85	46.73	8.07	15.21	44.68	69.81
1.0	0.0	0.1	2	33.79	48.85	9.10	15.95	52.02	71.80
0.7	0.3	0.0	2	36.35	46.70	8.42	14.91	58.42	71.10
0.7	0.3	0.1	2	37.01	48.86	8.46	15.20	59.12	72.88

Table 21: **Hyperparameter sensitivity to GRPO Rollouts, Learning Rate, and Terminal Dominance Cap for VISTA .** We vary the relative weights of the outcome and process rewards while training all variants for one epoch. We additionally report the default reward configuration trained for two epochs as a reference for the effect of longer training.

	Training	**Item F1 (%)**		**Row F1 (%)**		**Disc F1 (%)**	
Default	2	40.17	50.46	9.66	17.85	62.93	74.32
Rollouts	Epochs	Avg@4	Max@4	Avg@4	Max@4	Avg@4	Max@4
8*	1	35.24	45.28	8.66	14.53	57.47	70.28
16	1	34.76	44.82	7.18	12.49	57.76	70.41
Learning Rate	Epochs	Avg@4	Max@4	Avg@4	Max@4	Avg@4	Max@4
5e-7	1	33.27	43.78	7.69	13.00	55.62	69.79
1e-6*	1	35.24	45.28	8.66	14.53	57.47	70.28
3e-6	1	27.73	45.23	7.57	16.08	42.93	67.12
TD Cap δ	Epochs	Avg@4	Max@4	Avg@4	Max@4	Avg@4	Max@4
0.0	1	33.73	44.19	7.22	13.04	56.30	69.73
0.1*	1	35.24	45.28	8.66	14.53	57.47	70.28
0.5	1	34.25	44.49	7.64	12.57	57.64	70.51

Rollout Group Size. Increasing the number of rollouts per prompt from 8 to 16 provides more samples for group-relative advantage estimation, but does not improve downstream performance (Table 21). Specifically, Item F1 changes from 35.24% to 34.76%, while Row F1 and Discovery F1 change from 8.66%/57.47% to 7.18%/57.76% respectively. The absence of consistent gains suggests that $n = 8$ already provides sufficient within-group comparisons for effective credit estimation. We therefore use $n = 8$ by default, which achieves comparable performance at substantially lower rollout cost.

Learning Rate Sensitivity As shown in Table 21, halving the learning rate to 5×10^{-7} leads to a modest drop in Item F1 (-1.96 points) and Discovery F1 (-1.85 points) under the same 125-step budget. In contrast, increasing it to 3×10^{-6} destabilizes training: the answer-emission rate on the English split drops from 96% to 58%, resulting in substantial decreases in Item F1 (-7.51 points) and Discovery F1 (-14.54 points). Among examples where an answer is successfully produced, table quality remains comparable to the default, suggesting that the degradation mainly comes from failure to complete the final answer rather than poorer intermediate search. Such instability under overly large learning rates has also been observed in prior GRPO training (Gan et al., 2026). VISTA is robust around the default learning rate (1×10^{-6}) (Tao et al., 2026) and does not require fine-grained learning-rate tuning, while retaining the usual optimization stability boundary of the underlying RL algorithm.

Terminal Dominance Cap Sensitivity. VISTA constrains the advantage of each non-terminal turn to be at most $(1 - \delta)$ times the terminal advantage, with δ=0.1 by default (i.e., a cap of 0.9). As shown in Table 21, the default setting achieves the best overall performance. A tighter cap (δ=0.5) reduces Item F1 and Row F1 by 0.99 and 1.02 points, respectively, while removing terminal dominance entirely (δ=0) leads to larger drops of 1.51 Item F1 and 1.44 Row F1. These results reveal an important trade-off in turn-level credit assignment. Constraining intermediate credit too aggressively can weaken the process signal available for shaping intermediate search behavior, whereas removing the constraint allows intermediate credit to become comparable to the terminal signal, weakening the intended terminal dominance. The best performance at the intermediate setting (δ=0.1) supports the central design principle of VISTA : process credit should be strong enough to guide intermediate decisions, while remaining subordinate to the terminal objective. The results show that **VISTA is reasonably robust to the precise cap value while benefiting from explicitly maintaining terminal dominance.**

D.7 Beyond Wide Search

Zero-shot transfer from WideSearch. We first evaluate checkpoints trained only on wide search, without further training on HotpotQA, 2WikiMultiHopQA, or MuSiQue. Following the evaluation size used by Gao et al. (2025), we randomly sample 1,000 validation questions from each benchmark and report exact match (EM) and token-level F1. As shown in Table 22, both Qwen3-4B VISTA variants improve over the untrained backbone on all three benchmarks. VISTA-Mem raises average F1 from 33.4 to 42.2, an 8.8 point gain, while VISTA-Inc reaches 41.8. Transfer with Llama-3.1-8B-Instruct is less consistent: both variants improve substantially on 2WikiMultiHopQA, but do not improve on HotpotQA or MuSiQue. Thus, zero-shot transfer depends on the backbone and benchmark.

Table 22: VISTA on multi-hop QA. The zero-shot models are trained only on WideSearch; the Search-QA-trained model uses an evidence state. EM and F1 are reported in percent.

Method	**HotpotQA**		**2WikiMultiHopQA**		**MuSiQue**		**Avg**	
	EM	F1	EM	F1	EM	F1	EM	F1
Qwen3-4B: zero-shot transfer from WideSearch								
Single Agent	32.9	43.2	37.1	40.7	8.7	16.2	26.2	33.4
VISTA-Inc	38.5	50.1	**48.7**	**53.6**	13.0	21.6	33.4	41.8
VISTA-Mem	**40.1**	**51.4**	47.7	52.7	**14.6**	**22.5**	**34.1**	**42.2**
Llama-3.1-8B-Instruct: zero-shot transfer from WideSearch								
Single Agent	**31.4**	**41.4**	15.6	18.9	**11.7**	**17.3**	19.6	25.8
VISTA-Inc	31.3	41.3	**29.0**	**33.7**	7.2	15.3	**22.5**	**30.1**
VISTA-Mem	30.2	40.4	26.7	31.1	7.1	15.1	21.3	28.9
Qwen3-4B: training with multi-hop QA data and an evidence state								
VISTA-Mem	51.9	66.1	67.9	75.4	27.5	37.2	49.1	59.5

Intermediate state for multi-hop QA. VISTA requires an intermediate state whose progress can be verified during training, but this state need not be a table. For multi-hop QA, we replace the partial answer table with an explicit evidence state that records the supporting facts found so far and a candidate answer. We score progress in this state against the task's available annotations and assign turn-level credit using the same terminal-dominant mechanism as in wide search. The evidence-state verifier is used during training; the agent does not need it at evaluation time.

Training VISTA on multi-hop QA. We also test whether the method can be applied directly to a task without answer tables. We train VISTA-Mem with the evidence-state interface on 2,000 HotpotQA and 2,000 2WikiMultiHopQA questions, using a pooled BM25 retrieval corpus. The resulting model reaches 59.5 average F1 across the three benchmarks, compared with 33.4 for the untrained Qwen3-4B backbone. MuSiQue tests transfer beyond the training datasets: no MuSiQue questions, paragraphs, or annotations are used for training. These results show how **VISTA's intermediate-state and turn-level credit mechanism can be instantiated for multi-hop QA**, while the zero-shot results above measure transfer of the original WideSearch-trained checkpoints.

Extending VISTA beyond wide search. Wide search naturally produces an answer table: each row represents a target item, and filled cells show what the agent has found. This makes table quality a useful measure of progress at each turn. VISTA can also be applied to deep-search tasks if their training examples are represented as tables with meaningful intermediate scores. However, building such tables is less direct for deep search. Multiple search and reasoning paths may lead to the same answer, and a useful step may not immediately fill a cell. Defining checkpoints for every reasoning path is impractical, while scoring only selected checkpoints may introduce noisy or misleading process rewards. Our direct-training results show that this extension is feasible but not perfect; designing table states and scores that capture diverse reasoning paths remains an important direction for future work.

E Notation Overview

Table 23: Summary of notation used throughout the paper.

Notation	Type / Domain	Definition	Description
		Task and Wide-Search Trajectory	
x	Query	$x \sim \mathcal{D}$	Input wide-search query sampled from the task distribution $\mathcal{D}$.
$\mathcal{D}$	Distribution	–	Distribution of wide-search tasks.
i	Index	$i \in \{1, \dots, G\}$	Index of a rollout sampled for the same query.
G	Integer	–	Number of rollouts sampled per query.
t	Turn index	$t \in \{1, \dots, T_i\}$	Index of an interaction/search turn in rollout i.
T_i	Integer	–	Number of interaction turns in rollout i.
$a_{i,t}$	Policy action	$a_{i,t} \sim \pi_\theta(\cdot \mid h_{i,t})$	Policy-generated action at turn t, including reasoning, search queries, or answer-state updates.
$o_{i,t}$	Observation	–	Environment observation returned after action $a_{i,t}$, e.g., retrieved search evidence.
$h_{i,t}$	Interaction history	$(x, a_{i,1}, o_{i,1}, \dots, a_{i,t-1}, o_{i,t-1})$	History available to the policy before interaction turn t.
τ_i	Trajectory	$(x, a_{i,1}, o_{i,1}, \dots, a_{i,T_i}, o_{i,T_i})$	Complete multi-turn search trajectory of rollout i.
π_θ	Policy	–	Search-agent policy parameterized by θ.
		Structured Answer and Intermediate Table States	
$\mathcal{C}$	Attribute set	$\{c_1, \dots, c_C\}$	Set of attributes requested by wide-search task.
c	Attribute index	$c \in \{1, \dots, C\}$	Index of a requested table attribute.
C	Integer	$\lvert\mathcal{C}\rvert$	Number of requested attributes in answer table.
e_j	Entity	–	Target entity corresponding to the j-th table row.
j	Row/entity index	$j \in \{1, \dots, N\}$	Index of a target entity or table row.
N	Integer	–	Number of target entities in the GT answer table.
$\mathcal{T}^*$	Ground-truth table	$\mathcal{T}^* = \{(e_j, v_{j,1}, \dots, v_{j,c})\}_{j=1}^{N}$	Ground-truth structured answer; $\mathcal{T}^*(e_j, c_c)$ denotes the value of attribute c_c for entity e_j.
$\mathcal{T}_{i,t}$	Intermediate table	–	Explicit answer table state after turn t of rollout i.
$\mathcal{T}_{i,0}$	Initial table state	–	Initial table state before search begins.
$\mathcal{T}_{i,t}^{\mathrm{INC}}$	Incremental table state	–	Intermediate table state under VISTA-INC, incrementally updated from the previous table within the interaction context.
$\mathcal{T}_{i,t}^{\mathrm{MEM}}$	Memory table state	$\mathcal{M}_{i,t}$	Intermediate table state under VISTA-MEM, as persistent structured external memory.
$\pi_\theta^{\mathrm{table}}$	Table-update Policy	–	Policy operation responsible for producing or updating the explicit intermediate table state.
		Verifiable Intermediate-State Credit	
$\mathcal{E}$	Evaluator	–	Evaluator to assign the table state a score.
$Q_{i,t}$	Scalar	$\mathcal{E}(\mathcal{T}_{i,t}, \mathcal{T}^*)$	Verifiable quality score of the intermediate table state after turn t.
$r_{i,t}$	Turn-level signal	$Q_{i,t} - Q_{i,t-1}$	Intermediate-state progress signal measuring the change in table quality caused by turn t.
$\bar{r}_i$	Scalar	$\frac{1}{T_i}\sum_{t=1}^{T_i} r_{i,t}$	Mean intermediate-state progress over rollout i, used to center turn-level credit.
$\ell_{i,t}$	Local credit	$r_{i,t} - \bar{r}_i$	Centered local progress credit assigned to turn t.

Continued in next page...

Table 24: Summary of notation used throughout the paper (Continued).

Notation	Type / Domain	Definition	Description
		Trajectory-Level Reward and Advantage	
R_i	Scalar	$\lambda_{\text{out}} R_i^{\text{out}} + \lambda_{\text{proc}} R_i^{\text{proc}} + \lambda_{\text{fmt}} R_i^{\text{fmt}}$	Terminal trajectory reward computed after completion of rollout i.
R_i^{out}	Scalar	–	Terminal outcome reward measuring correctness of the final answer.
R_i^{proc}	Scalar	–	Trajectory-level process reward used in the terminal reward.
R_i^{fmt}	Scalar	–	Reward for compliance with the required answer/output format.
$\lambda_{\text{out}}, \lambda_{\text{proc}}, \lambda_{\text{fmt}}$	Scalars	–	Weights of the outcome, process, and format components of the terminal reward.
μ_g	Scalar	$\frac{1}{G}\sum_{i=1}^{G} R_i$	Mean terminal reward within the rollout group for the same query.
σ_g	Scalar	–	Standard deviation of terminal rewards within the rollout group.
A_i^{traj}	Scalar	$\frac{R_i - \mu_g}{\sigma_g + \epsilon}$	Group-normalized trajectory-level advantage of rollout i.
		Turn-Level Attribution and Terminal-Dominant Projection	
$\widetilde{A}_{i,t}$	Scalar	$A_i^{\text{traj}} + \ell_{i,t}$	Raw turn-level advantage obtained by combining global trajectory quality with local state progress.
δ	Hyperparameter	$\delta \in [0, 1)$	Margin controlling the strength of terminal dominance.
κ_i	Projection coefficient	$\min\left(1, \frac{(1-\delta)\lvert\widetilde{A}_{i,T_i}\rvert}{\max_{t<T_i}\lvert\widetilde{A}_{i,t}\rvert + \epsilon}\right)$	Rollout-specific coefficient used to scale non-terminal advantages.
$A_{i,t}$	Scalar	$\kappa_i \widetilde{A}_{i,t}$ if $t < T_i$; $\widetilde{A}_{i,T_i}$ otherwise	Final VISTA turn-level advantage after terminal-dominant projection.
ϵ	Small constant	$\epsilon > 0$	Numerical-stability constant used in normalization and projection.
		Token-Level Policy Optimization	
k	Token index	–	Token position within a policy-generated rollout; distinguished from interaction-turn index t.
$\mathcal{P}_{i,t}$	Token-index set	–	Set of policy-generated token positions belonging to interaction turn t of rollout i; environment-observation tokens are excluded.
$\mathcal{P}_i$	Token-index set	$\mathcal{P}_i = \bigcup_{t=1}^{T_i} \mathcal{P}_{i,t}$	Set of all policy-generated token positions in rollout i.
$\rho_{i,k}(\theta)$	Probability ratio	$\frac{\pi_\theta(y_{i,k} \mid y_{i,<k}, x)}{\pi_{\theta_{\text{old}}}(y_{i,k} \mid y_{i,<k}, x)}$	Importance ratio between the current and rollout policies at policy-generated token k.
ϵ_c	Clipping coefficient	–	GRPO clipping hyperparameter for the policy probability ratio.
θ_{old}	Parameters	–	Parameters of the behavior policy used to collect the current rollouts.