STAMP: Provenance-Guided Credit Assignment for Deep Search Agents

Anonymous ACL submission

Abstract

Reinforcement learning for deep-search agents has largely focused on trajectory-level scoring—outcome correctness, citation-aware rewards, and evidence coverage. Yet the actions that expose supporting documents receive no targeted credit, a gap we call the *reward-credit mismatch*. We propose STAMP, in which a reference-based verifier judges whether each cited document supports an entity or relation in a training-time evidence graph, and first-exposure attribution traces each supported citation back to the action that first surfaced it. This step credit is injected through sign-preserving advantage modulation, which redistributes advantage across steps without changing the trajectory-level reward or the relative ranking of trajectories within each group. On BrowseComp, BrowseComp-ZH, and xbench-DS, STAMP improves the GRPO baseline by **+2.0/+5.5/+3.0** points under matched SFT initialization, training data, and search tools, and composes with both outcome-only and citation-rubric base rewards. Component ablations confirm that the provenance-based credit signal and the sign-preserving advantage modulation each contribute to the gains.

1 Introduction

Deep search agents discover hidden entities, verify relations across webpages, and assemble cited evidence to support a final answer (Yao et al., 2023; OpenAI, 2025a). Reinforcement learning has become the dominant training paradigm for improving such agents, but long-horizon search exposes a distinction that trajectory-level training tends to blur: trajectory-level scoring of a rollout differs from step-level credit assignment to the actions that produced it (Zhang et al., 2026a).

Most recent progress targets the scoring side: outcome correctness, or how to score final answers, citations, and evidence coverage at the trajectory level (Jin et al., 2025; Gao et al., 2025; Li et al., 2025). Richer trajectory-level rewards—including rubric-based scoring, citation-aware rewards, entity- or relation-level evidence rewards, and multi-turn outcome supervision (Zhang et al., 2026b; Zhao et al., 2026)—have substantially improved deep-search agents, especially when paired with synthetic data and cold-start trajectories. Our analysis is consistent with this direction, but suggests that the value of a rollout is largely determined by the evidence its steps actually produce: Fig. 1a reports accuracy rising sharply from no target evidence (18.0%) to entity-only evidence (30.8%) to relation-verified evidence (69.1%). Outcome-level rewards score whether such evidence ends up cited, but do not directly reward the actions that exposed it.

The complementary side—step-level evidence, namely which search or read actions first expose target entities and verify relations—remains far less exploited. In standard outcome-supervised RL, the trajectory's correctness verdict is the only signal that survives group-relative normalization, and the resulting advantage is broadcast uniformly to every action token (Shao et al., 2025; Guo et al., 2025). This broadcast hides a strong mismatch between outcomes and evidence-producing actions: Fig. 1b shows that 83.5% of steps in correct rollouts produce no evidence at all, while 7.0% of steps in incorrect rollouts still expose useful entity- or relation-level evidence. We call this form of credit-assignment gap the **reward-credit mismatch**: the outcome channel captures whether a trajectory succeeds, but the evidence channel—which actions expose the supporting documents—remains an underused supervision signal.

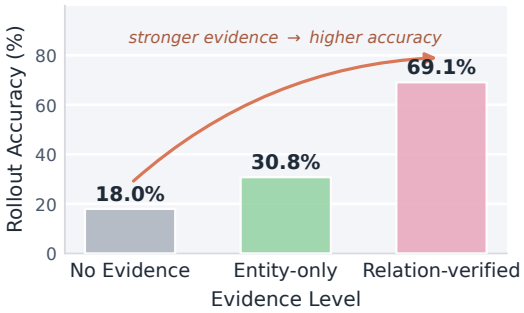
One remedy is to inject step-level reward bonuses, but in standard outcome-supervised setups—including GRPO—trajectories receive a single scalar reward; per-step bonuses are summed into that scalar and broadcast uniformly, eliminating the temporal localization they were intended

to provide. Learned process reward models could deliver local feedback in principle (Yin et al., 2025; Choudhury, 2025), but applying them to open-web search requires judging arbitrary intermediate actions in a large, evolving action space, often without an explicit reference for what evidence the action should expose.

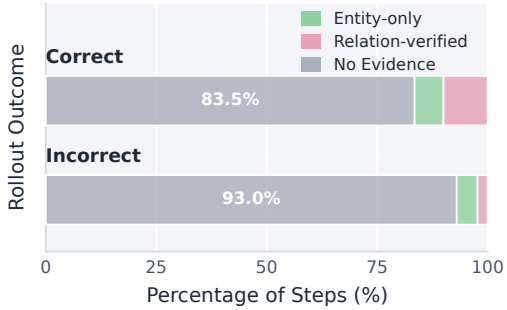
Instead, we ground credit in citation provenance recorded by the trajectory. Verification reduces to a reference-based judgment—whether a cited document supports a given entity or constraint from the training-time evidence graph—thus avoiding scoring the action itself. Existing citation-aware methods consume citations as input to a trajectory-level reward (Nakano et al., 2022; Zhang et al., 2026b); we instead use them as temporal provenance pointers, tracing each supporting document back to the action that first exposed it. This is possible in our setting because agents emit citations and our verifiable data-synthesis pipeline already constructs a training-time evidence graph for each query.

We propose **STAMP** (**S**tep-level **T**race-guided **A**dvantage **M**odulation with **P**rovenance), which addresses the reward–credit mismatch by decoupling two questions: what evidence a trajectory verifies, and which action first made that evidence available. The resulting step credit is injected through sign-preserving advantage modulation, a form of step-level advantage redistribution that leaves the trajectory-level reward and the induced trajectory ranking untouched. The evidence graph is used only at training time; at inference, the agent must rediscover entities, relations, and supporting documents through live search.

On Qwen3-30B-A3B-Thinking-2507 (Team, 2025), we evaluate STAMP in a live-web deep-search setting on BrowseComp (Wei et al., 2025), BrowseComp-ZH (Zhou et al., 2025), and xbench-DS (Xbench-Team, 2025). All controlled baselines share identical SFT initialization, training data, and search tools. On top of outcome-only GRPO, STAMP improves accuracy by **+2.0/+5.5/+3.0** points on the three benchmarks. The gains are largest for outcome-only and citation-rubric rewards and smaller for entity-only reward, suggesting that citation-provenance credit is most useful when the trajectory-level reward does not already densely supervise entity recovery. Component ablations confirm that verification quality, first-exposure attribution, bounded modulation, and negative-advantage attenuation each contribute to



(a) Accuracy by evidence level.



(b) Per-step evidence credit.

Figure 1: Evidence in deep-search rollouts. (a) Accuracy by the highest evidence level reached: none, entity-only, or relation-verified. (b) Per-step evidence credit by trajectory correctness, showing that evidence-producing steps are sparse and occur in both correct and incorrect rollouts.

the final performance.

Our contributions are as follows: We empirically characterize the reward-credit mismatch in deep-search RL: 83.5% of steps in correct rollouts produce no target evidence, while 7.0% of steps in incorrect rollouts still expose useful entity- or relation-level evidence. We propose STAMP, which turns citation provenance into verifiable step credit via a reference-based verifier and first-exposure attribution, then injects this credit through sign-preserving advantage modulation while leaving the trajectory-level reward and ranking untouched. STAMP improves BrowseComp, BrowseComp-ZH, and xbench-DS by **+2.0/+5.5/+3.0** points over the GRPO baseline under matched SFT initialization, training data, and tools, and composes with outcome-only and citation-rubric rewards.

2 Methodology

STAMP (**S**tep-level **T**race-guided **A**dvantage **M**odulation with **P**rovenance) addresses the reward–credit mismatch in outcome-only training

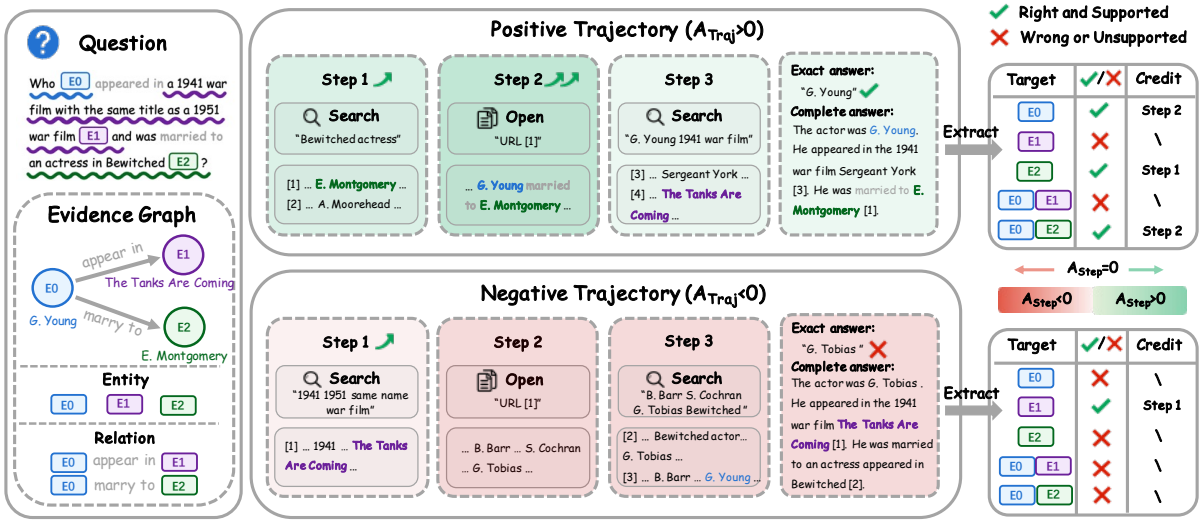


Figure 2: Overview of STAMP. Citation provenance traces verified evidence back to its earliest exposure step, converting document-level support into step-level credit. Sign-preserving advantage modulation then amplifies credited steps in positive trajectories ($A_{\text{traj}} > 0$) and shields them from uniform penalization in negative ones ($A_{\text{traj}} < 0$), while uncredited steps retain the original trajectory advantage.

by decoupling two questions: *what* evidence a trajectory verifies, and *which* action first made it available. Given a reference-anchored verifier that judges cited documents against the training-time evidence graph (§2.2), STAMP (i) uses **citation provenance** to map each supporting document to its earliest exposure step, yielding bounded step-level credit (§2.3), and (ii) injects this credit through **sign-preserving advantage modulation** that leaves the trajectory-level reward $R(\cdot)$ untouched (§2.4). The design is orthogonal to the choice of trajectory-level reward.

2.1 Preliminaries: Deep Search with Evidence Traces

Task Formulation We formulate deep search as an episodic decision process. At step t, the agent observes history $h_t = (q, a_1, o_1, \ldots, a_{t-1}, o_{t-1})$ and issues an action a_t such as a search query, page-reading request, or final cited response; the environment returns observation o_t. An episode ends with a final answer r and cited URLs, yielding trajectory $\tau = (a_1, o_1, \ldots, a_T, r)$. Outcome-supervised training assigns a terminal reward $R(\tau)$ from final-answer correctness, leaving intermediate search actions with undifferentiated credit.

Training-Time Evidence Graph Each training query q is paired with a training-time evidence graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Nodes $\mathcal{V}$ are task-relevant entities, with target entities $\mathcal{V}_{\text{tar}} \subseteq \mathcal{V}$ denoting *hidden* entities that are not revealed in q and must be recovered through search; entities mentioned in q are excluded from $\mathcal{V}_{\text{tar}}$ by construction and never enter the entity-credit budget. Edges $(e_i, e_j) \in \mathcal{E}$ are single-hop factual relations instantiated as atomic, single-document-verifiable natural-language constraints c_{ij}; $\mathcal{E}_{\text{tar}} \subseteq \mathcal{E}$ denotes the target constraints used for verification. The graph is only a training-time scaffold for reward computation and credit assignment, not an exhaustive set of valid evidence paths; at inference time, the agent receives only the query. Training data synthesis details are provided in Appendix A.

Standard GRPO GRPO (Shao et al., 2024) samples a group of G rollouts $\{\tau_1, \ldots, \tau_G\}$ from π_θ given q, scores each with $R_i = R(\tau_i)$, and produces a trajectory-level advantage

$$A_i = \frac{R_i - \mu}{\sigma + \epsilon}, \tag{1}$$

where μ, σ are the group mean and standard deviation of $\{R_i\}$ and ϵ guards zero-variance groups; A_i is then broadcast uniformly to every action token of τ_i.

2.2 Evidence Verification

For each target edge $(e_i, e_j) \in \mathcal{E}_{\text{tar}}$ with constraint c_{ij}, define a verification unit $u_k = (e_i, e_j, c_{ij})$ and let $\mathcal{U}_{\text{tar}}$ be the set of such units. Let $\mathcal{J}$ be a binary verifier over cited document contents that judges support for a *specific* entity or constraint—not general topical relevance—against a reference drawn from the evidence graph. It is invoked post-hoc

rather than inside the policy update. We instantiate $\mathcal{J}$ as an LLM-based extractor-matcher; verifier prompts and validation accuracy are reported in Appendix F. After URL canonicalization, $\mathcal{A}_t$ denotes the documents whose URL first becomes available at step t (via a query result or read action), and $\mathcal{C}(\tau)$ denotes the documents cited in r whose URL matches one observed in τ; unresolved citations are discarded.

For query-mentioned entities (excluded from $\mathcal{V}_{\text{tar}}$ by construction), we set $\text{cov}(e) = 1$ so units anchored on a revealed endpoint remain groundable; such units generate no entity credit, and the relation gate $\text{verified}(u_k) = 1$ (Eq. 3) still requires the hidden endpoint and the constraint to be supported by cited documents. For target entities and grounded verification units, the verifier induces supporting document sets

$$\begin{aligned} \mathcal{D}(e) &= \{d \in \mathcal{C}(\tau) : \text{sup}(e, d) = 1\}, \\ \mathcal{D}(u_k) &= \{d \in \mathcal{C}(\tau) : \text{sup}(c_{ij}, d) = 1\}. \end{aligned} \tag{2}$$

The corresponding indicators are

$$\begin{aligned} \text{cov}(e) &= \mathbf{1}[\mathcal{D}(e) \neq \emptyset], \\ \text{cov}(c_{ij}) &= \mathbf{1}[\mathcal{D}(u_k) \neq \emptyset], \\ \text{verified}(u_k) &= \text{cov}(e_i)\,\text{cov}(e_j)\,\text{cov}(c_{ij}). \end{aligned} \tag{3}$$

2.3 Credit from Citation Provenance

Having defined *what* evidence the trajectory verified (§2.2), we now ask *which* action receives credit for it. We attribute credit only to actions that first expose evidence the trajectory later verifies, leaving all other intermediate actions unjudged.

Evidence-to-Document Credit Each grounded hidden target entity ($e \in \mathcal{V}_{\text{tar}}$ and $\text{cov}(e) = 1$) generates entity credit, and each verified unit ($\text{verified}(u_k) = 1$) generates relation credit, even when one endpoint entity is already revealed in the query. Entity and relation units use the same per-unit weight δ; the difference between the two channels lies in their trigger conditions rather than in per-unit weighting. By Eqs. 2–3, $\mathcal{D}(e)$ and $\mathcal{D}(u_k)$ are nonempty whenever the corresponding indicator equals 1, so the divisions below are well-defined. Document-level credit is distributed uniformly across each supporting document set:

$$\text{credit}_{\text{ent}}(d, e) = \frac{\delta}{|\mathcal{D}(e)|}, \quad d \in \mathcal{D}(e), \tag{4}$$

$$\text{credit}_{\text{rel}}(d, u_k) = \frac{\delta}{|\mathcal{D}(u_k)|}, \quad d \in \mathcal{D}(u_k), \tag{5}$$

when $e \in \mathcal{V}_{\text{tar}}$ with $\text{cov}(e) = 1$ and when $\text{verified}(u_k) = 1$, respectively, and 0 otherwise. Uniform within-set distribution avoids introducing a learned weighting over documents, keeping attribution reproducible given verifier outputs. A document supporting both an entity and a verified relation receives credit from both channels, since the two channels measure complementary contributions (identification vs. relation establishment). Credit attribution applies to all trajectories regardless of outcome; low-reward trajectories may still contain steps that retrieved genuinely useful evidence, and §2.4 ensures such steps are not uniformly penalized when the group-relative advantage is negative.

Provenance Map to Search Steps To make temporal localization reproducible from the trajectory log, we attribute each supporting document d to the earliest action that makes it available to the agent. Recalling the document set $\mathcal{A}_t$ defined in §2.2, we define the exposure step of document d as:

$$\text{step}(d) = \min\big\{t : d \in \mathcal{A}_t\big\}. \tag{6}$$

All credit assigned to d is transferred to this exposure step; later actions that re-encounter the same document receive no extra credit. Earliest-exposure is a deliberate inductive bias rather than a default choice: it credits the action that first *discovers* a supporting document, whereas a read-time alternative $\text{step}(d) = \min\{t : a_t \text{ reads } d\}$ would credit the action that *consumes* it. In deep search, query formulation is the dominant bottleneck—many queries fail to surface any supporting document at all—so we route credit to discovery actions; learned alternatives (LLM-judged attribution, information-gain estimation) trade off this log-reproducibility for a separate attribution model and are left to future work.

Bounded Step Credit The total credit for step t accumulates entity and relation contributions from all documents first exposed at that step:

$$\begin{aligned} E_t &= \sum_{\substack{e \in \mathcal{V}_{\text{tar}} \\ \text{cov}(e)=1}} \sum_{\substack{d \in \mathcal{D}(e) \\ \text{step}(d)=t}} \text{credit}_{\text{ent}}(d, e), \\ U_t &= \sum_{\substack{u_k \in \mathcal{U}_{\text{tar}} \\ \text{verified}(u_k)=1}} \sum_{\substack{d \in \mathcal{D}(u_k) \\ \text{step}(d)=t}} \text{credit}_{\text{rel}}(d, u_k), \\ \text{credit}_t &= \min\big(C,\ E_t + U_t\big). \end{aligned} \tag{7}$$

The cap $C \in [0, 1)$ prevents any single step from dominating the gradient signal; together with $\text{credit}_t \in [0, C]$ it ensures sign preservation in the modulation operator of §2.4. The cap is in practice active only when many evidence units concentrate on a single step, which can happen when a broad query coincidentally surfaces many relevant URLs at once. Define the (uncapped) trajectory evidence budget as $B(\tau) := \sum_t (E_t + U_t)$. Each grounded hidden target entity ($\text{cov}(e) = 1$) contributes δ in total across the trajectory, and each verified unit ($\text{verified}(u_k) = 1$) contributes δ; ungrounded entities and unverified units contribute 0. Combined with the monotone, dominated capped accumulator $\text{credit}_t = \min(C, E_t + U_t)$, and writing $n_e(\tau) := |\{e \in \mathcal{V}_{\text{tar}} : \text{cov}(e) = 1\}|$ and $n_u(\tau) := |\{u_k \in \mathcal{U}_{\text{tar}} : \text{verified}(u_k) = 1\}|$ for the grounded-entity and verified-unit counts, we obtain

$$\begin{aligned} \sum_t \text{credit}_t \;\le\; B(\tau) \;=\; \big(n_e(\tau) + n_u(\tau)\big)\,\delta \\ \le\; (|\mathcal{V}_{\text{tar}}| + |\mathcal{U}_{\text{tar}}|)\,\delta. \end{aligned} \tag{8}$$

The bound depends only on the size of the target evidence graph, not on the trajectory length T; trajectories that take more steps distribute the same credit budget more sparsely rather than accumulating more total modulation.

2.4 Sign-Preserving Advantage Modulation

Once provenance credit is localized to action steps, we use it to redistribute optimization pressure within the trajectory-level signal produced by GRPO, without introducing a separate local reward. We let m index generated action tokens and write $t(m)$ for the action step containing token m, and abbreviate the per-step credit of trajectory τ_i as $\text{credit}_t^{(i)}$ (the quantity defined in Eq. 7).

Sign-Preserving Modulation STAMP keeps $R(\cdot)$ and the group statistic in Eq. 1 untouched and modulates the broadcast at the action-step level. A trajectory-summed credit bonus $R'_i = R_i + \sum_{t'} \text{credit}_{t'}^{(i)}$ would be rebroadcast uniformly by GRPO's group normalization and erase temporal localization, whereas modulation acts on A_i at the step level, preserving per-step granularity and the trajectory ranking induced by $R(\cdot)$. Concretely, we define the *sign-preserving modulation operator*

$$\mathcal{M}_c(A) := A\,\big(1 + \text{sign}(A)\,c\big), \tag{9}$$

with $c \in [0, C]$ and cap $C \in [0, 1)$, and apply it at the step level using the citation-provenance credit $\text{credit}_t^{(i)}$, which depends only on citation provenance and verifier outputs, not on A_i or R_i. This gives $\widetilde{A}_{i,t} = \mathcal{M}_{\text{credit}_t^{(i)}}(A_i)$, and all action tokens m in the same step share the modifier $A^*_{i,m} = \widetilde{A}_{i,t(m)}$. We adopt the multiplicative form over two alternatives sharing the same amplify-positive / attenuate-negative semantics:

- *Multiplicative* $A(1{+}\text{sign}(A)c)$: the multiplier $1 \pm c \in [1{-}C,\, 1{+}C]$ is bounded away from zero for any $C < 1$, so sign is preserved unconditionally; modulation scales with $|A|$ and vanishes as the group becomes homogeneous.
- *Additive* $A{+}\text{sign}(A)\,c\,A_{\text{ref}}$: sign preservation requires the trajectory-dependent condition $|A| > c\,A_{\text{ref}}$ and may fail on near-consensus rollouts; this is equivalent to step-level reward shaping injected at the advantage layer.
- *Power* $\text{sign}(A)|A|^{1-c}$: amplifies when $|A| < 1$ but attenuates when $|A| > 1$, contradicting the intended semantics in roughly half of the cases.

Among these three forms, the multiplicative form uniquely combines unconditional sign preservation with $|A|$-scaled modulation: the operator satisfies $\text{sign}(\mathcal{M}_c(A)) = \text{sign}(A)$ and $|\mathcal{M}_c(A)| \in [(1{-}C)|A|, (1{+}C)|A|]$ for all $c \in [0, C]$, so credit only rescales optimization pressure and cannot reverse the direction set by the trajectory-level signal. For positive-advantage trajectories, evidence-producing steps receive amplified reinforcement; for negative-advantage trajectories, those steps are attenuated by $1 - c \in [1{-}C,\, 1]$ rather than converted into positive supervision, so a failed trajectory remains penalized but its evidence-producing search actions are not penalized as if they were uninformative. When $A_i = 0$ the modulation vanishes by construction, so groups whose rollouts share the same reward provide no advantage-layer signal.

Gradient and Objective With $g_{i,m}(\theta) = \nabla_\theta \log \pi_\theta(a_{i,m} \mid h_{i,m})$, the modulated update reads

$$A^*_{i,m} g_{i,m}(\theta) = \lambda_{i,m}\, A_i\, g_{i,m}(\theta), \tag{10}$$

$$\lambda_{i,m} = 1 + \text{sign}(A_i)\,\text{credit}_{t(m)}^{(i)}, \tag{11}$$

with $\lambda_{i,m} \in [1{-}C,\, 1{+}C]$, so every token gradient is rescaled but never sign-flipped. Thus, STAMP

reweights per-token learning pressure within a trajectory rather than redefining the learning target. Combined with Eq. 8, the trajectory-level deviation it introduces is bounded by the evidence-graph size rather than by trajectory length. The resulting clipped GRPO objective replaces the trajectory-level advantage broadcast to action tokens with $A^*_{i,m}$:

$$\mathcal{L}(\theta) = -\mathbb{E}\left[\sum_{i,m} \mathbb{I}_{i,m} \min\big(\rho_{i,m} A^*_{i,m},\ \bar{\rho}_{i,m} A^*_{i,m}\big)\right], \tag{12}$$

where $\bar{\rho}_{i,m} = \text{clip}(\rho_{i,m}, 1-\varepsilon, 1+\varepsilon)$, $\rho_{i,m}$ is the generated-token importance ratio, and $\mathbb{I}_{i,m}$ masks non-action tokens. The PPO/GRPO clipping is preserved unchanged; STAMP modulates only the advantage signal. Training proceeds in four post-rollout steps: verify cited documents against the evidence graph, attribute supporting documents to exposure steps, compute step credits, and apply sign-preserving modulation during the GRPO update. All verification and attribution are performed after rollout completion, so STAMP adds no cost during rollout generation; the overhead is limited to parallel post-hoc verification.

3 Experiments

3.1 Setup

Tools and Environment The agent interacts with the live web using two tools. The `search` tool queries the Serper API (Serper, 2026) and returns ranked snippets with their corresponding URLs, while the `open` tool uses the Jina API (Jina.ai, 2026) to extract page content from a specified URL.

Training Setup The base model is Qwen3-30B-A3B-Thinking-2507 (Team, 2025). We synthesize bilingual training data from Wikipedia (Appendix A): 3000 queries for SFT (2000 EN, 1000 ZH) and 2,600 for RL (2,100 EN, 500 ZH). Our training process includes cold-start SFT and subsequent RL. For cold-start SFT, we leverage Seed Pro 2.0 (Seed, 2026) to generate 1,500 high-quality search trajectories through reject sampling on the SFT query set, then fine-tune the base model for 1 epoch with a batch size of 32, a learning rate of 2e-5, and a maximum context length of 64K. For RL, starting from the SFT checkpoint, we train with GRPO on the 2,600 synthesized queries. The training configuration includes a rollout size of 32, 8 samples per prompt, a global batch size of 256, a temperature of 1.0, a learning rate of 1e-6, and a maximum context length of 64K tokens. Training is run for 2 epochs. GPT-5.1 (OpenAI, 2025b) serves as the judge LLM for both outcome reward scoring and evidence verification. Full hyperparameters are listed in Appendix B.

Baselines All controlled baselines share identical SFT initialization, training data, hyperparameters, and search engine; they differ only in reward design: GRPO uses a binary outcome reward (0/1 correctness); E-GRPO uses entity hit rate against the evidence graph as reward (Zhao et al., 2026); C-GRPO combines a rubric-based citation score with outcome correctness (Zhang et al., 2026b). Reward details are in Appendix B. For broader context, Table 1 also lists proprietary and open-source agents as reference points.

Benchmarks We evaluate on BrowseComp (Wei et al., 2025), BrowseComp-ZH (Zhou et al., 2025), and xbench-DS (Xbench-Team, 2025). Together, these benchmarks assess core capabilities for deep search: long-horizon information-seeking, multi-step web navigation, complex reasoning, and cross-lingual synthesis. All models use the same inference environment (API, tool interface, 128K context window) with no turn or time limit.

3.2 Main Results

Table 1 presents results across all three benchmarks. All controlled methods start from the same SFT checkpoint; RL training with GRPO variants raises performance substantially across all reward designs. When STAMP is applied on top of the GRPO baselines, it yields consistent gains: +2.6 overall on GRPO, +1.2 on E-GRPO, and +2.4 on C-GRPO. These gains come from injecting citation-traced step credit through sign-preserving advantage modulation—no additional reward signal is introduced—indicating that provenance-based credit assignment captures complementary optimization information beyond the trajectory-level reward.

The improvement holds across three base rewards with fundamentally different designs (outcome-only, entity-aware, and rubric-based), confirming that STAMP's credit assignment is orthogonal to reward design and provides additive gains under the major reward formulations explored in recent deep-search agents.

Model	BrowseComp	BrowseComp-ZH	xbench-DS	Overall
Advanced Agents with Proprietary Data				
Tongyi-DeepResearch (Team et al., 2026)	43.4	46.7	75.0	45.9
RedSearcher (Chu et al., 2026)	42.1	49.8	-	-
Gemini 3.1 Pro (DeepMind, 2026)	53.6	75.9	86.0	59.5
Claude Sonnet 4.6 (Anthropic, 2026)	29.5	42.4	-	-
Seed 2.0 Pro (Seed, 2026)	77.3	82.4	88.0	78.8
Agents with Open-source Data				
Asearcher-Web-32B (Gao et al., 2025)	5.2	15.6	42.1	9.2
WebSailor-32B (Li et al., 2025)	10.5	25.5	53.3	15.7
WebExplorer-8B (Liu et al., 2025)	15.2	32.0	53.7	20.5
OpenSeeker-30B (Du et al., 2026)	29.5	48.4	74.0	35.5
Controlled Comparison†				
Qwen-30B-SFT	11.1	20.4	57.0	15.5
+ GRPO	18.9	30.8	70.0	24.1
+ GRPO + STAMP	**20.9** (+2.0)	**36.3** (+5.5)	**73.0** (+3.0)	**26.7** (+2.6)
+ E-GRPO	15.9	27.7	65.0	20.9
+ E-GRPO + STAMP	**17.0** (+1.1)	**29.4** (+1.7)	**65.0** (+0.0)	**22.1** (+1.2)
+ C-GRPO	18.3	31.1	69.0	23.6
+ C-GRPO + STAMP	**20.5** (+2.2)	**34.6** (+3.5)	**71.0** (+2.0)	**26.0** (+2.4)

Table 1: Main results on three deep search benchmarks. †Controlled baselines share identical SFT initialization, training data, and search engine—differences are exclusively in reward design and credit attribution.

3.3 Ablation Studies

We conduct ablations on the outcome-only GRPO baseline to isolate the contribution of individual design choices in STAMP.

Evidence Verification Method Table 2 compares two approaches for determining whether a cited document supports target evidence. String matching checks whether entity names and relation keywords appear literally in the document text; it captures surface-level co-occurrence but cannot handle paraphrases or verify semantic constraints. Our method restricts verification to cited documents and uses an LLM verifier to judge entity and relation support, yielding substantially higher credit quality.

Verification Method	BC	BC-ZH	xbench
String matching	15.6	26.6	65.0
URL + Judge (ours)	**20.9**	**36.3**	**73.0**

Table 2: Ablation on evidence verification method.

Advantage Modulation Variants Table 3 ablates three design dimensions. "All-step" distributes credit to every step that encounters a supporting URL rather than only the earliest, diluting the credit signal across redundant steps. "Unbounded" removes the per-step cap C, allowing a single evidence-rich step to dominate the gradient (sensitivity to δ and C is examined in Appendix D). "Pos. only" applies modulation exclusively to positive-advantage trajectories, removing the attenuation effect on negative ones. Our full design (earliest attribution, bounded cap, both signs) achieves the best result.

Attribution	Modulation	Scope	BC	BC-ZH	xbench
All-step	Bounded	Both	19.2	31.1	71.0
Earliest	Unbounded	Both	16.3	28.7	67.0
Earliest	Bounded	Pos. only	20.1	34.0	71.0
Earliest	**Bounded**	**Both**	**20.9**	**36.3**	**73.0**

Table 3: Design variant ablation. Each row changes one dimension from the full design (bottom).

3.4 Analysis

Training Dynamics Figure 3 tracks outcome reward, relation verification rate, and entity grounding rate over training. STAMP achieves higher outcome reward and converges faster than the GRPO baseline, indicating that localized credit accelerates policy improvement. The separation is most pronounced on relation rate (Fig. 3b): STAMP sustains a clear advantage throughout training, while the baseline plateaus at a lower level. Entity rate

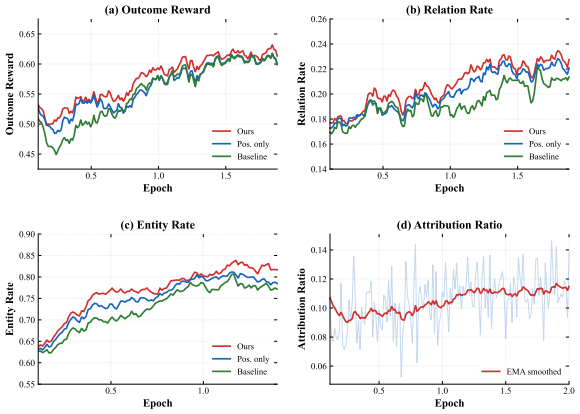


Figure 3: Training dynamics and credit sparsity. (a) Outcome reward; (b) relation verification rate; (c) entity grounding rate; (d) fraction of steps receiving non-zero credit ($\sim$10–12%), confirming that provenance-based attribution concentrates on a sparse subset of actions.

(Fig. 3c) saturates quickly with smaller separation—consistent with entity discovery being relatively easy, whereas relation verification requires targeted multi-hop evidence gathering that benefits most from step-level credit.

Credit Sparsity Figure 3d shows the fraction of steps receiving non-zero credit over training. Throughout the process, only $\sim$10–12% of steps are credited, confirming that provenance-based attribution concentrates optimization pressure on a small subset of actions rather than spreading it uniformly. The slight upward trend ($\sim$9% at epoch 0.3 to $\sim$11.5% at epoch 2.0) suggests a positive feedback loop: as the policy improves, it issues more evidence-producing queries, which generate more credit signal that further reinforces targeted evidence gathering.

4 Related Work

RL and Reward Design for Deep Search Agents Reinforcement learning has become central to training deep search agents that interact with live web environments (Jin et al., 2025). Recent work improves both the data and infrastructure for agentic RL (Lu et al., 2025; Tao et al., 2025) and the reward signals used to supervise multi-turn search behavior (Zhang et al., 2025a). Beyond binary outcome rewards, citation-aware rubrics, entity-level rewards, and decomposed evidence constraints provide denser trajectory-level supervision. CaRR (Zhang et al., 2026b) is closest to our setting: it decomposes questions into single-hop constraints and verifies cited evidence with an LLM-based rubric. E-GRPO (Zhao et al., 2026) similarly rewards entity match rate to distinguish near-miss rollouts from complete failures. These methods refine *what* a trajectory should be rewarded for, but the resulting scalar reward is still broadcast uniformly across actions; they do not identify which search or read step first exposed the verified evidence.

Step-Level Credit Assignment Temporal credit assignment is a foundational challenge in reinforcement learning (Pignatelli et al., 2024). Process reward models provide local supervision in mathematical reasoning (Zhang et al., 2025b), but open-web search lacks canonical labels for arbitrary intermediate actions and evolves over time. Hindsight relabeling and token-level reward modeling offer alternative ways to densify sparse feedback (Andrychowicz et al., 2018), yet they either operate over goal relabeling or require learned local reward models. STAMP instead derives step-level credit from citation provenance already present in the trajectory: a reference-based verifier determines which cited documents support evidence units, and first-exposure attribution maps those supported documents back to the actions that surfaced them. This provides localized optimization pressure without changing the trajectory-level reward or requiring additional human step annotations.

5 Conclusion

We presented STAMP, a citation-provenance approach for assigning step-level credit in deep-search RL. The central idea is to separate trajectory-level scoring from step-level credit assignment: a reference-based verifier identifies which cited documents support target entities and constraints, citation provenance maps those supporting documents to their first-exposure actions, and sign-preserving advantage modulation injects the resulting credit without changing the trajectory-level reward or its induced ranking.

Across BrowseComp, BrowseComp-ZH, and xbench-DS, STAMP consistently improves controlled GRPO baselines under matched training and search conditions, with gains of **+2.0/+5.5/+3.0** points over the controlled baseline on each benchmark. Ablations further show that verification quality, first-exposure attribution, bounded modulation, and negative-advantage attenuation are all important to the final performance. These results indicate that citation provenance carries optimization signal not captured by trajectory-level rewards alone.

6 Limitations

All experiments are conducted on Qwen3-30B-A3B-Thinking. While this scale is representative of current open-source deep-search agents, we have not validated STAMP on substantially larger backbones (e.g., 70B+ dense or MoE models). Whether provenance-guided credit remains equally beneficial as model capacity grows is an open question.

The open-source community has released various search-augmented SFT checkpoints with differing training recipes and tool interfaces. To ensure fully controlled comparison—identical SFT data, tool definitions, search environment, and inference pipeline—we initialize exclusively from our own SFT checkpoint rather than adopting external ones. Extending STAMP to diverse community checkpoints is a natural next step.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2018. Hindsight experience replay. *Preprint*, arXiv:1707.01495.

Anthropic. 2026. Introducing claude sonnet 4.6.

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

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

Google DeepMind. 2026. Gemini 3.1 pro model card.

Yuwen Du, Rui Ye, Shuo Tang, Xinyu Zhu, Yijun Lu, Yuzhu Cai, and Siheng Chen. 2026. Openseeker: Democratizing frontier search agents by fully open-sourcing training data. *Preprint*, arXiv:2603.15594.

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

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. 2025. Segment policy optimization: Effective segment-level credit assignment in rl for large language models. *Preprint*, arXiv:2505.23564.

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

Jina.ai. 2026. Jina.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025. Websailor: Navigating super-human reasoning for web agent. *ArXiv*, abs/2507.02592.

Junteng Liu, Yunji Li, Chi Zhang, Jingyang Li, Aili Chen, Ke Ji, Weiyu Cheng, Zijia Wu, Chengyu Du, Qidi Xu, Jiayuan Song, Zhengmao Zhu, Wenhu Chen, Pengyu Zhao, and Junxian He. 2025. Webexplorer: Explore and evolve for training long-horizon web agents. *Preprint*, arXiv:2509.06501.

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

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback. *Preprint*, arXiv:2112.09332.

OpenAI. 2025a. Deep research system card.

OpenAI. 2025b. Introducing gpt-5.1 for developers.

Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, Olivier Pietquin, and Laura Toni. 2024. A survey of temporal credit assignment in deep reinforcement learning. *Preprint*, arXiv:2312.01072.

Bytedance Seed. 2026. Seed2.0 model card: Towards intelligence frontier for real-world complexity.

Serper. 2026. Serper: Google search api.

Zhihong Shao, Yuxiang Luo, Chengda Lu, Z. Z. Ren, Jiewen Hu, Tian Ye, Zhibin Gou, Shirong Ma, and Xiaokang Zhang. 2025. Deepseekmath-v2: Towards self-verifiable mathematical reasoning. *Preprint*, arXiv:2511.22570.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

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

Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025. Webshaper: Agentically data synthesizing via information-seeking formalization. *Preprint*, arXiv:2507.15061.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, Kuan Li, Liangcai Su, Litu Ou, Liwen Zhang, Pengjun Xie, Rui Ye, Wenbiao Yin, Xinmiao Yu, Xinyu Wang, and 38 others. 2026. Tongyi deepresearch technical report. *Preprint*, arXiv:2510.24701.

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

Xbench-Team. 2025. xbench-deepsearch.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Zhangyue Yin, Qiushi Sun, Zhiyuan Zeng, Qinyuan Cheng, Xipeng Qiu, and Xuanjing Huang. 2025. Dynamic and generalizable process reward modeling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4203–4233, Vienna, Austria. Association for Computational Linguistics.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Francisco Piedrahita-Velez, Yue Liao, Hongru Wang, and 6 others. 2026a. The landscape of agentic reinforcement learning for llms: A survey. *Preprint*, arXiv:2509.02547.

Jiajie Zhang, Xin Lv, Ling Feng, Lei Hou, and Juanzi Li. 2026b. Chaining the evidence: Robust reinforcement learning for deep search agents with citation-aware rubric rewards. *Preprint*, arXiv:2601.06021.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, Ruiming Tang, and Xiangyu Zhao. 2025a. Process vs. outcome reward: Which is better for agentic rag reinforcement learning. *Preprint*, arXiv:2505.14069.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025b. The lessons of developing process reward models in mathematical reasoning. *Preprint*, arXiv:2501.07301.

Yida Zhao, Kuan Li, Xixi Wu, Liwen Zhang, Dingchu Zhang, Baixuan Li, Maojia Song, Zhuo Chen, Chenxi Wang, Xinyu Wang, Kewei Tu, Pengjun Xie, Jingren Zhou, and Yong Jiang. 2026. Repurposing synthetic data for fine-grained search agent supervision. *Preprint*, arXiv:2510.24694.

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

A Data Synthesis Details

A.1 Tree Structure Examples

Easy ($n_{\text{hidden}} = 1$, 2 leaves, 1 inference step) Two leaf nodes converge to uniquely identify the answer:

```
Leaf_A (attended) --> D (answer)
Leaf_B (graduated) --> D (answer)
```

Agent must find what both Leaf_A and Leaf_B share.

Medium ($n_{\text{hidden}} = 2$, 3–5 leaves, 2 inference steps)

```
Leaf_A (connects) --> Hidden_P
Leaf_B (hosted)   --> Hidden_P
Hidden_P (located_in) --> D
Leaf_C (adjacent_to)  --> D
```

Step 1: Identify Hidden_P from Leaf_A and Leaf_B. Step 2: Combine Hidden_P with Leaf_C to determine D.

Hard ($n_{\text{hidden}} \geq 4$, 5+ leaves, 3+ inference steps) Multiple layers of hidden nodes requiring recursive inference across 3+ steps.

A.2 Relation Refinement

Raw relation constraints are refined to ensure each constraint references exactly two entity placeholders. Constraints spanning 3+ entities are split into atomic binary relations; constraints referencing only one entity are merged with related constraints. This normalization ensures consistent granularity across questions of different complexity.

A.3 Edge Verification Details

The dual-path judge evaluates three criteria for each edge: (1) whether the stated relation factually holds; (2) whether the relation is current (not outdated); (3) whether evidence from both paths is consistent. This catches common failure modes in raw KG edges: outdated sports affiliations, retracted mergers reported as completed, and editorial "See Also" links misinterpreted as semantic relations.

B Training Configuration

B.1 Model Architecture

We use Qwen3-30B-A3B-Thinking-2507 as the base model: 30B total parameters with 3B active per token via Mixture-of-Experts routing. Supports up to 128K context; we use 64K maximum context length during both SFT and RL training. Both stages are implemented on the VeRL (Sheng et al., 2024) training framework.

B.2 SFT Stage

We use Seed Pro 2.0 to generate 1,500 high-quality search trajectories on the SFT query set, then fine-tune the base model for 1 epoch with batch size 32, learning rate 2e-5, and maximum context length 64K. Training is conducted on 16×H800 GPUs. All Training baselines share this SFT initialization.

B.3 RL Stage

Starting from the SFT checkpoint, we train with GRPO on 2,600 synthesized queries using 16×H800 GPUs. GPT-5.1 serves as the verification judge for evidence reward computation. Table 4 lists the complete hyperparameters.

B.4 Training Data Composition

The 2.6K RL training questions are distributed across difficulty levels: easy (n_{hidden}=1): 40%, medium (n_{hidden}=2–3): 45%, hard (n_{hidden}≥4): 15%. This distribution reflects the natural yield from CHP mining after verification filtering.

C STAMP Training Algorithm

Algorithm 1 summarizes the complete STAMP training loop. Each iteration rolls out a group of trajectories, verifies cited URLs against the evidence graph, attributes credit to first-exposure steps, and applies sign-preserving modulation before the policy update.

Hyperparameter	Value
Training questions	2.6K (2.1K EN + 0.5K ZH)
Rollouts per prompt	8
Rollout batch size	32
Global batch size	256
Sampling temperature	1.0
Maximum context length	64K tokens
Training epochs	2
Learning rate	1×10^{-6}
KL coefficient (β)	0.01
Clip range (ε)	0.2
Credit quantum δ	0.03
Per-step credit cap C	0.10
Max cited URLs for verification	15
Verification model	GPT-5.1

Table 4: Complete RL training hyperparameters.

Algorithm 1 STAMP training: reward-decoupled citation-traced credit with sign-preserving modulation

Require: Policy π_θ, training queries $\{(q, \mathcal{G}, \mathcal{V}_{\text{tar}}, \mathcal{U}_{\text{tar}})\}$, base reward $R(\cdot)$, group size G, credit quantum δ, cap C

for each training iteration **do**
  Sample query $(q, \mathcal{G}, \mathcal{V}_{\text{tar}}, \mathcal{U}_{\text{tar}})$ with training-time evidence graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
  Roll out G trajectories $\{\tau_i\}_{i=1}^{G}$ from π_θ given q
  for each τ_i **do**
    Verify: run verification (§2.2) against $\mathcal{G}$ using cited URLs in τ_i, producing $\{\mathcal{D}(e)\}, \{\mathcal{D}(u_k)\}, \{\text{sup}(e)\}, \{\text{verified}(u_k)\}$
    Compute base reward $R_i = R(\tau_i)$
    Attribute: distribute δ to supporting documents via entity/relation channels, trace to steps via $\text{step}(d)$ (Eq. 6)
    Accumulate $\text{credit}_s^{(i)}$ per action step with cap C (Eq. 7)
  end for
  Compute group advantages $A_i = (R_i - \mu)/(\sigma + \epsilon)$ (Eq. 1)
  Compute step-modulated $\widetilde{A}_{i,s}$ via $\mathcal{M}_{\text{credit}_s^{(i)}}(A_i)$ (Eq. 9) and broadcast to action tokens as $A^*_{i,m}$
  Update θ by minimizing $\mathcal{L}(\theta)$ (Eq. 12)
end for

D Supplementary Experiments

Table 5 examines sensitivity to the credit quantum δ and per-step cap C. Performance peaks at δ=0.03 and C=0.10; smaller values underweight the credit signal, while larger values concentrate too much advantage on a few steps.

δ	BC	BC-ZH	xbench
0.01	17.8	29.4	69.0
0.03 (ours)	**20.9**	**36.3**	**73.0**
0.05	19.7	33.2	70.0
C	**BC**	**BC-ZH**	**xbench**
0.05	17.9	29.1	71.0
0.10 (ours)	**20.9**	**36.3**	**73.0**
0.15	16.3	27.7	65.0

Table 5: Sensitivity to credit quantum δ and per-step cap C.

E Human Verification of LLM Judge

To assess the reliability of LLM as the verification judge for evidence reward computation, we conducted a manual review of its judgments across 200 randomly sampled training trajectories from the SFT checkpoint. Using human assessments as the gold standard, the judge LLM achieved an accuracy of 96.04% for entity identification (whether a cited document supports the presence of a target entity) and 97.31% for relation verification (whether a cited document confirms the stated relationship between entities). The majority of disagreements involved borderline cases where evidence was implicit rather than explicitly stated. These results indicate that the LLM judge provides sufficiently reliable verification signals for credit attribution during training.

F Prompts and Formats

F.1 Tool Descriptions

The agent is equipped with two tools for interacting with the live web. Their specifications are provided verbatim in the system prompt:

F.2 Trajectory Format

Each training trajectory follows the multi-turn format below. The agent alternates between reasoning (within `<think>` tags), tool invocations, and environment observations until producing a final cited response.

F.3 Prompt for Outcome Rewards

The outcome reward is computed by prompting GPT-5.1 to judge whether the agent's final answer matches the ground-truth answer. The judge extracts the answer from the response, compares it against the reference, and returns a binary correctness label (Figure 6).

F.4 Prompt for Entity Verification

Entity verification (Stage 1 of evidence reward) checks whether each target entity is correctly identified in the response and supported by cited webpage evidence. The verifier receives the question, relation constraints with entity placeholders, ground-truth values, the agent's response, and the cited webpage contents (Figure 7).

F.5 Prompt for Relation Verification

Relation verification (Stage 2 of evidence reward) determines whether each relation constraint between identified entities is supported by the cited webpage contents. Unlike entity verification which checks identity, this stage evaluates whether the stated factual relationship can be confirmed from the evidence (Figure 8).

Tool Definitions

```
[
  {
    "name": "search",
    "description": "Search the web for information. Returns top results with url and snippet.",
    "parameters": {
      "type": "object",
      "properties": {
        "query": {"type": "string", "description": "The search query."},
        "num": {"type": "integer", "description": "Number of results (default 10)."}
      },
      "required": ["query"]
    }
  },
  {
    "name": "open",
    "description": "Open a page by URL and return the complete content.",
    "parameters": {
      "type": "object",
      "properties": {
        "id": {"type": "string", "description": "The absolute URL from prior search results."}
      },
      "required": ["id"]
    }
  }
]
```

Figure 4: Tool definitions provided to the agent in the system prompt.

Trajectory Format

```
<|im_start|>system
You are a helpful assistant. ...
<tools> [tool definitions] </tools>
<|im_end|>
<|im_start|>user
[question + instruction]
<|im_end|>
<|im_start|>assistant
<think> reasoning process </think>
<tool_call>
{"name": "search", "arguments": {"query": "..."}}
</tool_call>
<|im_end|>
<|im_start|>user
<tool_response> search results </tool_response>
<|im_end|>
<|im_start|>assistant
<think> reasoning process </think>
<tool_call>
{"name": "open", "arguments": {"id": "https://..."}}
</tool_call>
<|im_end|>
<|im_start|>user
<tool_response> page content </tool_response>
<|im_end|>
... (more search/open iterations) ...
<|im_start|>assistant
<think> final reasoning </think>
## Complete Answer
[explanation with inline citations, e.g. [1], [2]]
## Exact Answer
[final answer]
## References
[1] https://...
[2] https://...
<|im_end|>
```

Figure 5: Multi-turn trajectory format used during training and inference.

Prompt for Outcome Rewards

Judge whether the following **[response]** to **[question]** is correct or not based on the precise and unambiguous **[correct_answer]** below.

[question]: {question}
[response]: {response}

Your judgement must be in the format and criteria specified below:

extracted_final_answer: The final exact answer extracted from the **[response]**. Put the extracted answer as 'None' if there is no exact, final answer to extract from the response.

[correct_answer]: {correct_answer}

reasoning: Explain why the **extracted_final_answer** is correct or incorrect based on **[correct_answer]**, focusing only on whether there are meaningful differences between **[correct_answer]** and the **extracted_final_answer**. Do not comment on any background to the problem. Do not attempt to solve the problem. Focus only on whether the answers match.

correct: Answer 'yes' if the **extracted_final_answer** matches the **[correct_answer]** given above, or is within a small margin of error for numerical problems. Answer 'no' otherwise, including cases of inconsistency, ambiguity, non-equivalency, or incorrectness.

Figure 6: Prompt for outcome reward judgment.

Prompt for Entity Verification

You will receive:
1. A complex multi-hop question.
2. A set of relation constraints with entity placeholders (`<E0>`, `<E1>`, ...).
3. The correct (ground truth) values for each entity.
4. An AI assistant's response to the question.
5. The contents of webpages cited by the assistant (evidence).

Your task (for each entity):
1. Extract the identity of each entity from the assistant's response. For `<E0>`, prioritize the `## Exact Answer` section. For other entities, look in the `## Explanation with Citations` section.
2. Judge whether each extracted entity value matches its ground truth value. Consider equivalent names, minor wording differences, abbreviations, or alternate transliterations as correct.
3. Judge whether each entity's identity is explicitly mentioned or supported by the provided webpage evidence. If supported, indicate which URL contains the evidence.

[Question]: {question}
[Relation Constraints]: {constraints}
[Ground Truth Entity Values]: {ground_truth}
[Assistant's Response]: {response}
[Webpage Contents (cited evidence)]: {context}

Output format:

```
{
  "E0": {"identified_value": "...", "correct": true/false,
         "supported": true/false, "source_url": "URL N"},
  "E1": {...},
  ...
}
```

Figure 7: Prompt for entity verification.

Prompt for Relation Verification

You will receive:
1. The contents of several webpages (evidence collected by the AI assistant).
2. Several relation constraints: S1, S2, ..., Sn — factual statements describing relationships between identified entities.

Your task: For each constraint:
- Find the exact evidence from the webpage contents that supports or contradicts it.
- A constraint is "supported" only if the webpage content contains sufficient information to verify the stated relationship or fact.
- Conclude with a judgment: **Fully Supported: yes** or **Fully Supported: no**.

[Webpage Contents]: {context}
[Relation Constraints]: {statements}

Output format:

```
{
  "S1": {"supported": true/false, "source_url": "URL N"},
  "S2": {"supported": true/false, "source_url": "URL N"},
  ...
}
```

Figure 8: Prompt for relation verification.