CLUEFLOW: LEARNING LONG-HORIZON SEARCH THROUGH ATTRIBUTED RESEARCH-STATE TRANSITIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon search requires agents to acquire relevant evidence and integrate it into an evolving research state. Existing methods maintain research states to guide exploration or assign turn-level progress signals, yet grounding these signals in explicit, attributable state changes remains underexplored. We introduce ClueFlow, which learns long-horizon search from attributed research-state transitions. ClueFlow constructs Source–State–Final attribution chains that connect evidence-acquiring browser actions, Evidence and Candidate updates, and the final answer supported by those updates. During search, ClueFlow records typed state changes and potentially supporting browser actions, then applies gold-conditioned semantic validation and source tracing after execution. We use this attribution to guide both supervised fine-tuning and reinforcement learning. During SFT, attribution selects which decisions receive supervision with their original interaction contexts. During Turn-PPO, the same attribution augments outcome-based turn advantages with credit assigned along the Source–State–Final chains. Transition-level validation also recovers useful supervision from gold-consistent state changes in trajectories with incorrect final answers. With Qwen3.5-35B-A3B, ClueFlow achieves 59.1% accuracy on BrowseComp and 72.0% on BrowseComp-Plus. In controlled BrowseComp-Plus experiments with Qwen3-30B-A3B, attribution-guided SFT matches all-turn exact-context SFT using only about 30% of the training tokens, while attribution-guided PPO outperforms outcome-only PPO by 4.1 percentage points. These results demonstrate that state-transition attribution enables efficient supervision and effective credit assignment for long-horizon search agents.

1 INTRODUCTION

Long-horizon search requires agents to gather evidence, evaluate candidate answers, and revise their hypotheses across extended interactions. For a multi-constraint question, a page may reveal evidence that confirms or rules out a candidate, prompting the agent to preserve that evidence and update its research state. A correct final answer alone does not distinguish useful evidence acquisition and state updates from routine exploration. Conversely, filtering out trajectories with incorrect final answers may discard valid intermediate progress. Learning from long search trajectories therefore requires identifying which research-state changes constitute meaningful progress and which information-gathering actions support those changes.

Recent methods organize accumulated findings into explicit research states, such as evolving reports (Chen et al., 2026a), structured summaries (Zhu et al., 2026), and candidate–constraint tables (Lan et al., 2026), to preserve useful information and guide subsequent exploration. Related memory methods learn to manage retained information through outcome-based reinforcement learning (Yan et al., 2026) or explicit feedback on memory informativeness (Li et al., 2026b). Other methods select valuable steps for supervision (Chen et al., 2025b; Slinko et al., 2026) or provide fine-grained credit assignment (Wang et al., 2025; 2026a; Liu et al., 2026; Wang et al., 2026b). However, using validated research-state changes and explicit links to their supporting search actions to guide supervision and credit assignment remains underexplored. Figure 1 illustrates this distinction.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

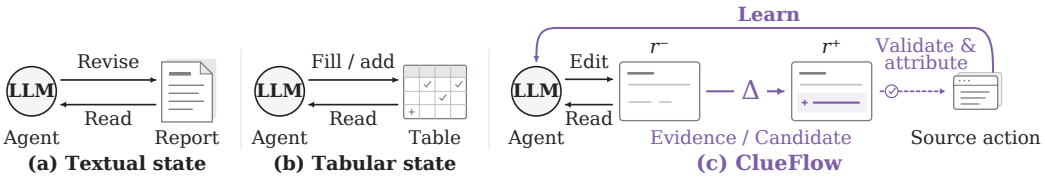


Figure 1: Learning from attributed research-state transitions. (a,b) Reports and tables guide search. (c) ClueFlow additionally learns from record-level transitions through post-execution validation and source attribution.

Our key insight is that validating research-state transitions and tracing their supporting evidence makes explicit both what progress was made and which search actions supplied the evidence for that progress. A state records the agent's current evidence and answer hypotheses; a transition makes explicit how these records change. Based on this insight, ClueFlow constructs Source–State–Final attribution chains that connect evidence-acquiring browser actions, Evidence and Candidate transitions, and the final answer supported by those transitions.

During search, ClueFlow maintains persistent Evidence and Candidate records and logs typed state changes together with potentially supporting browser actions for each Evidence update. A recorded state change does not by itself establish useful research progress. After execution, gold-conditioned semantic validation identifies Evidence and Candidate transitions that make useful progress toward the reference answer. Source tracing then links validated Evidence transitions to the browser actions that supplied their supporting information. Gold answers and validation judges are used only for post-execution attribution and are never exposed to the search policy at test time. For correct trajectories, the resulting attribution connects evidence-acquiring browser actions and validated state transitions to the final answer they support. For trajectories ending in incorrect answers, ClueFlow retains validated Source–State partial chains that capture gold-consistent intermediate progress.

We apply this attribution principle in both supervised fine-tuning and reinforcement learning. During supervised fine-tuning, attribution selects which decisions are supervised with their original interaction contexts, while other valid turns in successful trajectories are supervised with compactly serialized contexts. During Turn-PPO (Li et al., 2026a), the same attribution augments outcome-based turn advantages with credit assigned along the Source–State–Final chains. Both stages also use validated Source–State progress recovered from trajectories with incorrect final answers.

Our main contributions are:

- We introduce attributed research-state transitions to represent verifiable research progress and link it to the browser actions that supplied its supporting evidence, providing an explicit basis for agent learning.
- We develop ClueFlow, which constructs Source–State–Final attribution chains through semantic validation and source tracing to guide SFT context selection and Turn-PPO credit assignment. Both stages also learn from validated partial progress in trajectories with incorrect final answers.
- We evaluate ClueFlow on live-web benchmarks and BrowseComp-Plus, obtaining 59.1% accuracy on BrowseComp and 72.0% on BrowseComp-Plus with Qwen3.5-35B-A3B. On BrowseComp-Plus with Qwen3-30B-A3B, ClueFlow SFT outperforms the strongest tested ReAct SFT configuration by 6.5 percentage points. Controlled ablations show that attribution-guided SFT matches all-turn exact-context SFT using about 30% of the training tokens, while attribution-guided PPO improves accuracy by 4.1 percentage points over outcome-only PPO.

2 Related Work

2.1 Context and Memory Management for Long-Horizon Agents

Long-horizon agents operate on tasks spanning tens to hundreds of steps, where managing the ever-growing interaction history is a central challenge. Existing methods mainly control context

size by discarding past observations, selectively retaining information, or compressing past interactions (DeepSeek-AI, 2025; GLM-5-Team, 2026; Ye et al., 2025).

Recent search agents further organize accumulated findings into persistent, reusable task states that capture evidence, candidate answers, and future search plans. These approaches differ in how they represent and update research states. RE-TRAC updates the state at trajectory boundaries by recursively compressing each completed trajectory together with the previous state into a structured summary, which then guides the next rollout (Zhu et al., 2026). IterResearch adopts a finer-grained mechanism, updating an evolving report after each tool interaction (Chen et al., 2026a). Table-as-Search instead represents the task state as a candidate–constraint table and advances search by adding candidate rows and filling empty cells (Lan et al., 2026). Related work also learns how memory should be managed. Memory-R1 uses outcome-driven reinforcement learning to learn operations for adding, updating, and deleting entries in external memory (Yan et al., 2026), whereas MemPO optimizes information summarization and retention according to their usefulness for subsequent decisions (Li et al., 2026b).

Existing approaches primarily maintain research states to guide subsequent search. ClueFlow additionally validates state transitions, traces them to supporting browser actions, and uses the resulting attributed changes to guide supervision and credit assignment.

2.2 Post-Training and Credit Assignment for Search Agents

Whole-trajectory imitation and outcome-based rewards do not explicitly distinguish critical, redundant, and harmful turns. Recent work therefore seeks to identify the training value of individual turns and provide more targeted supervision.

Existing methods estimate turn value in different ways. ATLaS uses an LLM selector to identify critical steps in expert trajectories and computes the training loss only on those steps (Chen et al., 2025b). SRFT labels each step as useful, harmful, or unnecessary and masks harmful steps, thereby recovering useful supervision from failed trajectories (Slinko et al., 2026). Beyond such binary selection, StepSearch rewards evidence gain while penalizing redundant retrieval (Wang et al., 2025); IGPO and PiCA assign rewards based on changes in the probability of producing the correct answer or eventually completing the task (Wang et al., 2026a; Liu et al., 2026); and CriticSearch and CW-GRPO use LLM judgments to produce turn-level rewards or redistribute trajectory-level advantages (Zhang et al., 2026; Wang et al., 2026b). These methods convert turn-value estimates into selective losses, rewards, or advantage weights.

Rather than scoring turns directly, ClueFlow guides SFT and Turn-PPO using validated, source-attributed research-state transitions.

3 Method

3.1 Overview

A teacher agent interacts with the search environment to generate long-horizon trajectories with explicit research-state transitions. ClueFlow then validates these transitions and traces them to the browser actions that supplied the supporting evidence. The attributed teacher trajectories are used to construct SFT data and train an initial search policy. Starting from this policy, PPO generates new online rollouts and applies the same attribution procedure to assign fine-grained credit to individual turns. The following sections describe trajectory generation, attribution, SFT, and PPO in turn; Figure 2 summarizes the pipeline.

3.2 Stateful Trajectory Generation

ClueFlow uses the same stateful search interface for teacher trajectory generation, online policy rollouts during reinforcement learning, and test-time inference (Figure 2(a)).

Research state. ClueFlow maintains a persistent research state for each search trajectory. We denote the state available at the beginning of turn t by

$$S_t = (E_t, C_t), \qquad E_0 = C_0 = \varnothing, \tag{1}$$

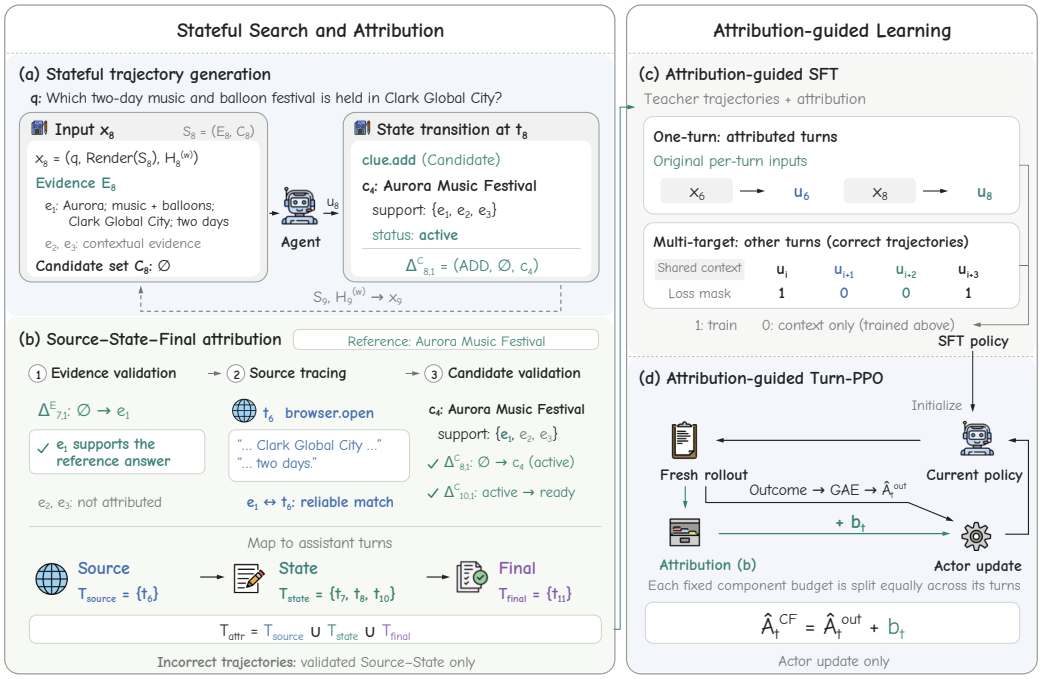


Figure 2: **Overview of ClueFlow.** **(a)** Stateful trajectory generation maintains Evidence and Candidate records across search turns. **(b)** Post-execution validation against the reference answer and source tracing identify attributed Source, State, and Final turns. Validated Source–State progress can also be retained from incorrect trajectories. **(c)** Attribution-guided SFT uses one-turn examples with original per-turn inputs for attributed turns and multi-target chunks for other supervised turns from correct trajectories. **(d)** Attribution-guided Turn-PPO adds attribution bonuses to outcome advantages for actor updates. Here, $\Delta_{t,k}^{E/C}$ denotes the k-th Evidence/Candidate state transition at assistant turn t.

where E_t is the set of Evidence records containing information extracted from browser observations, and C_t is the set of Candidate records that maintain answer hypotheses, their supporting Evidence, and their verification progress. Each Candidate has a status of `active`, `ready`, or `rejected`, indicating that it remains under investigation, is eligible for submission, or has been ruled out, respectively. Candidate records also track satisfied and unresolved constraints, contradictions, and the next search step. The state fields are summarized in Appendix A.

Trajectory generation. Given a question q, the agent generates a search trajectory by repeatedly producing an assistant response and executing its requested tools. It can call `browser.search`, `browser.open`, and `browser.find` to search the web, read a page, and locate relevant passages. It can also use `clue.add` and `clue.update` to create or revise Evidence and Candidate records, and `clue.discard` to remove obsolete Evidence. The tool interfaces are summarized in Appendix A.

At turn t, we construct

$$x_t = \left(q, \mathrm{Render}(S_t), H_t^{(w)}\right), \qquad u_t \sim \pi_\theta(\cdot \mid x_t), \tag{2}$$

where x_t is the model input, $\mathrm{Render}(S_t)$ converts the structured state into text, and $H_t^{(w)}$ contains up to w most recent turns preceding turn t. The agent model π_θ produces the response u_t, which may contain reasoning and multiple tool calls. Fixed system instructions and tool schemas are omitted from the notation.

The environment executes the tool calls in u_t: Browser results are appended to the interaction history, while a state transition occurs whenever a successful Clue operation modifies an Evidence or

Candidate record. A single turn may contain multiple state-changing operations. We record the k-th transition in turn t by its operation and the affected record's contents before and after the change:

$$\Delta^{v}_{t,k} = \left(\omega_{t,k}, r^{-}_{t,k}, r^{+}_{t,k}\right), \tag{3}$$

where k indexes the state-changing operations within the turn, $v \in \{E, C\}$ indicates whether the transition modifies an Evidence or Candidate record, $\omega_{t,k} \in \{\text{ADD}, \text{UPDATE}, \text{DISCARD}\}$ denotes the operation, and $r^{-}_{t,k}$ and $r^{+}_{t,k}$ are the record contents before and after the transition. All valid transitions in a turn jointly update S_t to S_{t+1}. All tool calls in u_t are generated from the same pre-turn input x_t. The browser observations returned at turn t can therefore support state updates only from turn $t+1$ onward. Search terminates when the agent submits an answer or exhausts its Browser budget. The resulting trajectory records the model responses, tool interactions, and state transitions.

3.3 SOURCE–STATE–FINAL ATTRIBUTION

Not every state transition in Equation 3 reflects useful progress. ClueFlow therefore processes each completed trajectory in three steps: validating Δ^E, tracing its browser source, and validating Δ^C. Together, these components form Source–State–Final attribution chains (Figure 2(b)).

Evidence validation. An LLM judge evaluates the final content of each Evidence record against the reference answer. We retain Evidence that supports the reference answer or rules out an incorrect candidate. Superseded transitions and discarded Evidence records are excluded. For each retained Evidence record, repeated writes of the same final content are represented by a single effective transition. For both correct and incorrect trajectories, only judge-approved Evidence listed as support by the Candidate associated with the submitted answer is eligible for positive attribution.

Source tracing. During trajectory generation, ClueFlow records recent browser actions available before each Δ^E as candidate sources. A deterministic matcher compares the post-transition Evidence content with the observations returned by these actions using document identifiers, URLs, page titles, and textual overlap. The best-matching browser action is retained only when the match is reliable (Appendix B.3); otherwise, Δ^E is excluded from positive attribution.

Candidate validation. An LLM judge also evaluates Candidate records against the reference answer. For the Candidate associated with the submitted answer, we retain Δ^C only if the Candidate passes semantic validation and cites validated Evidence. The transition must introduce or revise the submitted answer, add new validated support, or change the Candidate status from `active` to `ready`. Transitions that do not change the answer, its validated support, or its readiness are excluded.

Attribution chains and turn mapping. For a correct trajectory with a valid Source–State chain, a complete attribution chain connects the matched browser action (Source), the corresponding Δ^E and Δ^C (State), and the answer-submission turn (Final). If no Δ^C qualifies, the State part contains only Δ^E. For an incorrect trajectory, the Final link is omitted. We refer to reusing validated Source–State progress from incorrect trajectories as failure recovery.

ClueFlow then maps each action and transition to its enclosing assistant turn. Let T_{source} contain the turns that execute matched browser actions, T_{state} contain the turns that perform validated Δ^E or Δ^C, and T_{final} contain the final submission turn for a correct trajectory and be empty otherwise. We define the attributed turns as

$$T_{\text{attr}} = T_{\text{source}} \cup T_{\text{state}} \cup T_{\text{final}}. \tag{4}$$

For an unfinished trajectory, $T_{\text{attr}} = \varnothing$. Reference answers and judge outputs are used only after execution and are never exposed to the search policy. Appendix E provides qualitative examples of Source–State–Final attribution, including validated partial progress recovered from incorrect trajectories.

3.4 ATTRIBUTION-GUIDED SFT

Because state updates and history truncation change the model's visible context across turns, standard full-trajectory SFT cannot faithfully reproduce the inputs under which the responses were generated.

During SFT, our goal is to emphasize turns $t \in T_{\text{attr}}$ by training each response u_t with the original input x_t. A direct approach constructs a separate one-turn example (x_t, u_t) for every turn. A long trajectory would then produce many examples, and neighboring examples would repeatedly contain largely the same interaction history, substantially increasing training cost. Alternatively, several consecutive turns can be grouped into one multi-target chunk. This reduces repeated context but does not preserve the original x_t for every turn. ClueFlow uses one-turn examples for supervised attributed turns and multi-target chunks for other supervised turns in correct trajectories (Figure 2(c)).

Attribution and trajectory outcome determine each response turn's construction mode, while response-level masking determines which responses contribute loss:

$$g_t = \begin{cases} \text{ONE-TURN}, & t \in T_{\text{attr}}, \\ \text{CHUNK}, & t \notin T_{\text{attr}} \text{ and the trajectory is correct}, \\ \text{EXCLUDE}, & \text{otherwise}. \end{cases} \tag{5}$$

Supervised attributed turns use independent one-turn examples (x_t, u_t). The remaining turns from correct trajectories are grouped into consecutive multi-target chunks, where one shared context supervises multiple assistant responses. If an attributed response also appears as history within a chunk, its loss is masked there; when retained for supervision, it is trained in its one-turn example instead.

Incorrect trajectories can contribute validated Source and State turns in $T_{\text{source}} \cup T_{\text{state}}$, each with its original context, subject to response-level masking (Appendix C.1). Unfinished trajectories are excluded from training.

Let $z_n = (z_{n,1}, \ldots, z_{n,L_n})$ be a serialized SFT example and $m_{n,h} \in \{0, 1\}$ its token-level loss mask. We optimize

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{N} \sum_n \sum_{h=1}^{L_n} m_{n,h} \log \pi_\theta(z_{n,h} \mid z_{n,<h}), \qquad N = \sum_n \sum_{h=1}^{L_n} m_{n,h}. \tag{6}$$

All unmasked target tokens receive equal weight. Attribution guides example construction and, together with trajectory outcome and post-execution validation, determines response-level masking; it does not directly reweight the training loss. Chunk construction and response-level masking are detailed in Appendix C.1.

3.5 ATTRIBUTION-GUIDED TURN-PPO

Starting from the SFT policy, we sample new online search trajectories. Following Turn-PPO (Li et al., 2026a), each complete assistant response is treated as one action. The terminal outcome reward is 1 for a correct answer, 0 for an incorrect answer, and -0.1 for an unfinished trajectory; all intermediate rewards are zero. Turn-level GAE then produces an outcome advantage $\widehat{A}_t^{\text{out}}$ for each turn. We use these raw advantages without normalization or whitening.

ClueFlow augments this outcome advantage for turns in T_{source}, T_{state}, and T_{final} (Figure 2(d)). Within each trajectory, it assigns a fixed advantage budget to each chain component and divides it equally among the corresponding turns. For each chain component $j \in \{\text{source}, \text{state}, \text{final}\}$, let α_j denote its advantage budget. Attribution bonuses are applied only when a valid Source–State chain is recovered from a completed trajectory. The bonus assigned to turn t is

$$\begin{aligned} b_t &= \begin{cases} \displaystyle\sum_j \frac{\alpha_j \mathbb{1}[t \in T_j]}{\max(1, |T_j|)}, & \text{if a valid chain is recovered}, \\ 0, & \text{otherwise}, \end{cases} \\ \widehat{A}_t^{\text{CF}} &= \widehat{A}_t^{\text{out}} + b_t. \end{aligned} \tag{7}$$

We use $(\alpha_{\text{source}}, \alpha_{\text{state}}, \alpha_{\text{final}}) = (0.03, 0.03, 0.04)$. Turns are deduplicated within each set, while a turn appearing in multiple parts of the chain receives the sum of their bonuses. When a valid Source–State chain is recovered, the bonuses sum to 0.10 for a correct trajectory and 0.06 for an incorrect trajectory. Incorrect trajectories receive no Final bonus, and unfinished trajectories receive no attribution bonus. If no valid Source–State chain is recovered, $b_t = 0$ for all turns and the update reduces to outcome-only Turn-PPO.

The augmented advantage is used only in the actor update. It is not added to the environment reward and therefore does not change the return or critic target. For a response containing L_t tokens, we define one length-normalized importance ratio for the complete turn:

$$\rho_t(\theta) = \exp\left(\frac{1}{L_t}\sum_{h=1}^{L_t}\log\frac{\pi_\theta(u_{t,h} \mid x_t, u_{t,<h})}{\pi_{\text{old}}(u_{t,h} \mid x_t, u_{t,<h})}\right). \tag{8}$$

The actor objective averages the clipped turn-level losses within each trajectory and then across trajectories:

$$\mathcal{L}_{\text{actor}} = -\mathbb{E}_{\text{trajectories}}\left[\frac{1}{T}\sum_{t=1}^{T}\min\Big(\rho_t(\theta)\widehat{A}_t^{\text{CF}}, \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_t^{\text{CF}}\Big)\right], \tag{9}$$

where T is the number of valid assistant turns in a trajectory. This design isolates the effect of attribution to actor-side credit assignment.

4 Experiments

We first evaluate ClueFlow's end-to-end performance across two backbones on live-web benchmarks and BrowseComp-Plus. We then isolate the contributions of attribution-guided SFT and PPO through controlled ablations. Finally, we evaluate inference-time state management using a fixed GLM-5.1 backbone without ClueFlow-specific post-training.

4.1 Experimental Setup

Benchmarks. We evaluate ClueFlow on three live-web benchmarks: BrowseComp (Wei et al., 2025), BrowseComp-ZH (Zhou et al., 2025), xBench-DeepSearch-2510 (Chen et al., 2025a). We use BrowseComp-Plus (BCP) (Chen et al., 2026b) for fixed-corpus evaluation and controlled ablations of supervision construction and credit assignment, keeping the corpus and retriever fixed across all ablation settings.

Baselines. We compare our trained agents with representative deep-research systems, including IterResearch (Chen et al., 2026a), RE-TRAC (Zhu et al., 2026), Tongyi DeepResearch (Tongyi DeepResearch Team, 2025), REDSearcher (Chu et al., 2026), QUEST (Xie et al., 2026), and MiroThinker-1.7-mini (MiroMind Team et al., 2026), using full-benchmark results from published reports. We also evaluate each base model with ReAct without task-specific post-training. For inference-time comparisons, we fix GLM-5.1 (GLM-5-Team, 2026) as the backbone and compare ClueFlow with ReAct (Yao et al., 2023), Keep-k (GLM-5-Team, 2026), Discard-all (DeepSeek-AI, 2025), and HCM (GLM-5-Team, 2026) under the same browsing environment and budget.

Implementation Details. For each backbone—Qwen3-30B-A3B-Thinking-2507 (Yang et al., 2025) and Qwen3.5-35B-A3B (Qwen Team, 2026)—we train separate policies for live-web search on REDSearcher QA (Chu et al., 2026) and fixed-corpus search on OpenResearcher QA (Li et al., 2026c). For OpenResearcher QA and BrowseComp-Plus (Chen et al., 2026b), we use their respective officially released corpora and precomputed embeddings, with Qwen3-Embedding-8B (Zhang et al., 2025) as the dense retriever. GLM-5.1 (GLM-5-Team, 2026) generates SFT trajectories, and GPT-5.5 (OpenAI, 2026) validates Evidence and Candidate records for attribution. PPO starts from the corresponding SFT checkpoint. Our evaluations allow up to 800 browsing calls per question. Appendix C provides data statistics, training configurations, and evaluation protocols.

4.2 End-to-End Search Performance

Full-benchmark comparison. Table 1 summarizes end-to-end performance. External results use different training data, tool environments, and inference budgets.

Table 1: Comparison with representative deep-research systems on three live-web benchmarks and BrowseComp-Plus (BCP). Scores are accuracy (%).

		Live-web			Fixed-corpus
Method	Post-training	BC	BC-ZH	xBench-2510	BCP[†]
Representative deep-research systems					
IterResearch	RFT + RL	37.3	45.2	–	–
Tongyi DeepResearch	MT + SFT + RL	43.4	46.7	55.0	44.5
RE-TRAC	SFT	53.0	57.3	–	–
REDSearcher	MT + SFT + RL	57.4	58.2	–	–
QUEST-35B	MT + SFT + RL	64.6	–	–	69.5
MiroThinker-1.7-mini	MT + SFT + PO + RL	67.9	72.3	57.2	–
Our evaluations: Qwen3-30B-A3B					
ReAct	None	2.1	8.7	7.0	16.3
ClueFlow	SFT	49.1	54.0	53.5	48.3
ClueFlow	SFT + PPO	51.6	55.7	54.2	59.8
Our evaluations: Qwen3.5-35B-A3B					
ReAct	None	37.5	47.1	42.0	60.4
ClueFlow	SFT	57.9	58.5	54.8	68.8
ClueFlow	SFT + PPO	59.1	59.2	55.3	72.0

[†] For our post-trained models, BCP and live-web scores use separate checkpoints trained on OpenResearcher QA and REDSearcher QA, respectively. The Tongyi DeepResearch BCP score is taken from OpenResearcher (Li et al., 2026c). REDSearcher and MiroThinker results use their reported context-management settings; QUEST-35B BC/BCP scores use Discard-all. "None" indicates no task-specific post-training; "–" indicates an unavailable score. MT, RFT, and PO denote mid-training, rejection-sampling fine-tuning, and preference optimization, respectively.

Live-web performance. Both backbones improve substantially over ReAct after SFT, with further gains from PPO across all three live-web benchmarks. ClueFlow achieves 51.6% and 59.1% accuracy on BrowseComp with Qwen3 and Qwen3.5, respectively.

Fixed-corpus performance. On BrowseComp-Plus, we compare Qwen3-30B-A3B SFT policies using the same teacher and matched numbers of successful trajectories (Appendix D.1, Tables 6–7). ClueFlow SFT outperforms ReAct SFT with Discard-all, the strongest tested ReAct SFT configuration, by 6.50 percentage points (48.31% vs. 41.81%). PPO further raises ClueFlow accuracy to 59.8% for Qwen3 and 72.0% for Qwen3.5.

4.3 Attribution Ablations

We conduct attribution ablations on BrowseComp-Plus with Qwen3-30B-A3B trained on OpenResearcher QA (Table 2). All PPO variants start from the same ClueFlow SFT checkpoint and share rollout and optimization settings.

SFT supervision construction. Using only correct teacher trajectories, attribution-guided selection of turns for original-context supervision outperforms random selection by 3.26 percentage points (46.27% vs. 43.01%). Random selection performs similarly to chunk-only supervision (43.01% vs. 42.89%), suggesting that the choice of turns matters in this setting. Attribution-guided supervision also achieves comparable accuracy to all-turn exact-context SFT (46.27% vs. 46.14%) while using approximately 30% of its training tokens.

Recovering validated Source and State turns from incorrect trajectories further improves accuracy from 46.27% to 48.31%, a gain of 2.04 percentage points with only a 7% increase in training tokens.

Table 2: Controlled ablations of attribution-guided SFT supervision construction and PPO credit assignment on BrowseComp-Plus.

(a) SFT supervision construction

SFT variant	Random turns	Attributed turns	All turns	Acc. (%) ↑	Rel. tokens ↓
Chunk only	×	×	×	42.89	0.69×
Chunk + random turns	✓	×	×	43.01	1.06×
Chunk + attributed turns	×	✓	×	46.27	1.00×
+ Failure recovery	×	✓	×	**48.31**	1.07×
All-turn exact-context	×	×	✓	46.14	3.33×

(b) PPO credit assignment

Variant	Source	State	Final	Acc. (%) ↑	State conv. (%)	S–S Cov. (%)
Outcome-only	×	×	×	55.66	39.45	55.54
Source-only attribution	✓	×	×	54.34	41.86	48.43
Full-chain attribution	✓	✓	✓	58.67	44.25	57.71
+ Failure recovery	✓	✓	✓	**59.76**	44.26	59.04

Relative tokens count all non-padding serialized tokens, including masked context, and are normalized by Chunk + attributed turns; they are distinct from the unmasked target-token count N in Eq. (6). Source-only distributes the full 0.10 attribution budget across attributed Source turns. State conv. is the fraction of raw State-update turns retained after validation; S–S Cov. is the fraction of trajectories with at least one valid Source–State chain.

PPO credit assignment. Outcome-only PPO reaches 55.66%. With the same trajectory-level attribution budget of 0.10, full-chain attribution achieves 58.67%, compared with 54.34% for Source-only attribution. Extending attribution to validated Source and State turns from incorrect trajectories further improves accuracy to 59.76%, a gain of 1.09 percentage points over full-chain attribution on correct trajectories alone and 4.10 percentage points over outcome-only PPO.

State-level diagnostics are consistent with the intended mechanism: full-chain attribution raises State conversion from 39.45% to 44.25% while reducing raw State updates and slightly increasing validated ones (Table 8(b)). Failure recovery preserves conversion at 44.26% and raises Source–State coverage from 57.71% to 59.04%, indicating more selective updates and broader validated-progress coverage.

4.4 Inference-Time State Management

Table 3: Inference-time comparison on BrowseComp with GLM-5.1.

Method	Acc. (%)
ReAct	60.03
Keep-k	66.59
Discard-all	63.19
HCM	72.91
ClueFlow	**73.54**

All methods use the same browsing environment and an 800-call budget, without ClueFlow-specific post-training. ClueFlow obtains 73.54% accuracy, closely matching HCM (72.91%) and exceeding Discard-all (63.19%). This supports the inference-time utility of its research-state representation, complementing its role as a learning interface. Context-management settings are detailed in Appendix C.3.

5 Conclusion

ClueFlow uses attributed research-state transitions as a shared interface for supervision and credit assignment in long-horizon search. Gold-conditioned validation and source tracing identify useful browser actions and state updates for exact-context SFT and actor-side PPO corrections, including partial progress recovered from incorrect trajectories. Experiments on two backbones demonstrate its applicability to live-web search, while controlled BrowseComp-Plus ablations establish the benefits of attribution-guided context selection, full-chain credit assignment, and failure recovery. These findings support treating validated research-state transitions as explicit units for agent learning.

Reproducibility Statement

We provide the methodological and experimental details needed to reproduce ClueFlow throughout the paper and appendices. Sections 3.2–3.5 define the research-state representation, Source–State–Final attribution procedure, attribution-guided SFT construction, and Turn-PPO credit assignment. Appendix A specifies the runtime state and tool interfaces, context construction, provenance tracking, and submission rules, while Appendix B details semantic validation, effective-transition recovery, and deterministic source tracing. Appendix C reports the training data statistics, optimization hyperparameters, retrieval configurations, browsing budgets, and evaluation and scoring protocols. Additional baseline results and rollout diagnostics are provided in Appendix D, and Appendix E presents qualitative trajectory traces. For controlled attribution ablations, we use BrowseComp-Plus with its released fixed corpus and precomputed embeddings; live-web evaluations use Serper and Jina Reader and may therefore be affected by changes in web content and retrieval results over time.

AI Usage Statement

Generative AI tools were used both as components of the research pipeline and as assistance during the research and writing process. As part of the research pipeline, GLM-5.1 was used to generate synthetic teacher trajectories, and GPT-5.5 (OpenAI, 2026) was used for post-execution Evidence and Candidate validation and final-answer evaluation, as described in the paper.

Generative AI tools were also used to support methodological and experimental discussions, interpretation of experimental results, literature search, manuscript drafting and editing, and LaTeX formatting. The authors reviewed AI-assisted research suggestions and manuscript content against the implemented methods, experimental records, and relevant literature. All methodological choices, experimental decisions, interpretations, and final manuscript content were determined by the authors. The authors take full responsibility for the accuracy, integrity, and content of the paper and its associated artifacts.

References

Guoxin Chen, Zile Qiao, Xuanzhong Chen, Donglei Yu, Haotian Xu, Wayne Xin Zhao, Ruihua Song, Wenbiao Yin, Huifeng Yin, Liwen Zhang, Kuan Li, Minpeng Liao, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. IterResearch: Rethinking long-horizon agents with interaction scaling. In *International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=qQ5MZ5Mx7p.

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

Zhixun Chen, Ming Li, Yuxuan Huang, Yali Du, Meng Fang, and Tianyi Zhou. ATLaS: Agent tuning via learning critical steps. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25334–25349, Vienna, Austria, 2025b. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.1299. URL https://aclanthology.org/2025.findings-acl.1299/.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Sahel Sharifymoghaddam, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Hosna Oyarhoseini, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A fair and disentangled evaluation benchmark for deep search agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22349–22370, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1023. URL https://aclanthology.org/2026.acl-long.1023/.

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

DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models, 2025. arXiv:2512.02556.

GLM-5-Team. GLM-5: From vibe coding to agentic engineering, 2026. arXiv:2602.15763.

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, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.743. arXiv:2602.06724.

Junbo Li, Peng Zhou, Rui Meng, Meet P. Vadera, Lihong Li, and Yang Li. Turn-PPO: Turn-level advantage estimation with PPO for improved multi-turn RL in agentic LLMs. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 6227–6243, Rabat, Morocco, 2026a. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-eacl.328. URL https://aclanthology.org/2026.findings-eacl.328/.

Ruoran Li, Xinghua Zhang, Haiyang Yu, Shitong Duan, Xiang Li, Wenxin Xiang, Chonghua Liao, Xudong Guo, Yongbin Li, and Jinli Suo. MemPO: Self-memory policy optimization for long-horizon agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 23286–23301, San Diego, California, United States, 2026b. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1166. URL https://aclanthology.org/2026.findings-acl.1166/.

Zhuofeng Li, Dongfu Jiang, Xueguang Ma, Haoxiang Zhang, Ping Nie, Yuyu Zhang, Kai Zou, Jianwen Xie, Yu Zhang, and Wenhu Chen. OpenResearcher: A fully open pipeline for long-horizon deep research trajectory synthesis, 2026c. arXiv:2603.20278.

Dongyi Liu, Yifan Niu, Qinwen Wang, Han Xiao, and Jia Li. PiCA: Pivot-based credit assignment for search agentic reinforcement learning, 2026. arXiv:2605.09287.

MiroMind Team, S. Bai, L. Bing, L. Lei, R. Li, X. Li, X. Lin, E. Min, L. Su, B. Wang, L. Wang, L. Wang, S. Wang, X. Wang, Y. Zhang, Z. Zhang, G. Chen, L. Chen, Z. Cheng, Y. Deng, Z. Huang, D. Ng, J. Ni, Q. Ren, X. Tang, B. L. Wang, H. Wang, N. Wang, C. Wei, Q. Wu, J. Xia, Y. Xiao, H. Xu, X. Xu, C. Xue, Z. Yang, Z. Yang, F. Ye, H. Ye, J. Yu, C. Zhang, W. Zhang, H. Zhao, and P. Zhu. MiroThinker-1.7 & H1: Towards heavy-duty research agents via verification, 2026. arXiv:2603.15726.

OpenAI. Introducing GPT-5.5, April 2026. URL https://openai.com/index/introducing-gpt-5-5/.

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

Igor Slinko, Ilia Zavidnyi, Egor Bogomolov, and Yaroslav Zharov. Step rejection fine-tuning: A practical distillation recipe, 2026. arXiv:2605.10674.

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

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information gain-based policy optimization: A simple and effective approach for multi-turn search agents, 2026a. arXiv:2510.14967.

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, San Diego, California, United States, 2026b. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1462. URL https://aclanthology.org/2026.acl-long.1462/.

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, 2025. arXiv:2505.15107.

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

Jian Xie, Tianhe Lin, Zilu Wang, Yuting Ning, Yuekun Yao, Tianci Xue, Zhehao Zhang, Zhongyang Li, Kai Zhang, Yufan Wu, Shijie Chen, Boyu Gou, Mingzhe Han, Yifei Wang, Vint Lee, Xinpeng Wei, Xiangjun Wang, Yu Su, and Huan Sun. QUEST: Training frontier deep research agents with fully synthetic tasks, 2026. arXiv:2605.24218.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.583. URL https://aclanthology.org/2026.acl-long.583/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. arXiv:2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. arXiv:2210.03629.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, Pengjun Xie, Fei Huang, Siheng Chen, Jingren Zhou, and Yong Jiang. AgentFold: Long-horizon web agents with proactive context management, 2025. arXiv:2510.24699.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models, 2025. arXiv:2506.05176.

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 Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12272–12290, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.596. URL https://aclanthology.org/2026.findings-acl.596/.

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

Jialiang Zhu, Gongrui Zhang, Xiaolong Ma, Lin Xu, Miaosen Zhang, Ruiqi Yang, Song Wang, Kai Qiu, Zhirong Wu, Qi Dai, Ruichun Ma, Bei Liu, Yifan Yang, Chong Luo, Zhengyuan Yang, Linjie Li, Lijuan Wang, Weizhu Chen, Xin Geng, and Baining Guo. RE-TRAC: Recursive trajectory compression for deep search agents. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/60790.

A RUNTIME INTERFACE AND STATE MANAGEMENT

This appendix summarizes the runtime mechanism behind Section 3.2: research state, tool interfaces, context construction, Browser provenance, submission rules, and the agent system prompt. Offline attribution, training, and evaluation are covered in Appendices B and C.

A.1 STATE AND TOOL INTERFACES

At the beginning of turn t, the persistent state is $S_t = (E_t, C_t)$. Evidence and Candidate records share one monotonic ID namespace (`c1`, `c2`, ...). Discarded Evidence is stored separately from the active state. Listing 1 summarizes the normalized record fields.

```
{
  "ClueRecord": {
    "id": "string (runtime-assigned cN)",
    "content": "string",
    "kind": "evidence | candidate",
    "metadata": "{} | CandidateMetadata",
    "status": "active | discarded"
  },
  "CandidateMetadata": {
    "answer": {"type": "string", "default": "parent content"},
    "status": {"enum": ["active", "ready", "rejected"], "default": "active"},
    "supporting_clues": {"type": "string[]", "default": []},
    "satisfied_constraints": {"type": "string[]", "default": []},
    "missing_constraints": {"type": "string[]", "default": []},
    "contradictions": {"type": "string[]", "default": []},
    "next_check": {"type": "string", "default": ""}
  }
}
```

Listing 1: Research-state fields and the shared Candidate metadata definition.

Evidence uses empty metadata and defaults to `kind=evidence`; all records default to top-level `status=active`. This lifecycle status is distinct from `metadata.status`, which tracks Candidate readiness or rejection. `supporting_clues` holds Evidence IDs, validated during offline attribution (Appendix B). Missing Candidate strings and lists become empty; scalar values supplied for list fields are wrapped in singleton lists. Invalid decision statuses become `active`, and a missing answer falls back to the record content. Discarded Evidence leaves the active rendering but remains in the trajectory and discarded-record collection; a rejected Candidate remains in the state with `metadata.status=rejected`.

The following are *interface summaries*, not verbatim copies of the full schemas passed to the model through native function calling. A question mark marks an optional argument; unmarked arguments are required.

```
{
  "browser.search": {"query": "string", "topn?": "integer = 10"},
  "browser.open": {"id?": "integer | URL = -1", "cursor?": "integer = -1",
                   "loc?": "integer = -1", "num_lines?": "integer = -1",
                   "view_source?": "boolean = false", "source?": "string"},
  "browser.find": {"pattern": "string", "cursor?": "integer = -1",
                   "context_lines?": "integer = 0 (range 0..5)"}
}
```

Listing 2: Compact Browser tool interfaces.

Browser search returns session-local numeric link IDs. Online results include URLs, whereas the fixed-corpus results omit raw URLs and document IDs; their search/open descriptions direct the model to use the displayed numeric IDs. Opening a page returns a cursor and line-numbered text, and `browser.find` returns at most 20 matches with local context. `view_source` and `source` remain in the interface but do not change the current executor. Runtime-only source records are described in Appendix A.2.

```
{
  "clue.add": {"content": "string", "kind?": "evidence | candidate",
               "reason?": "string", "metadata?": "CandidateMetadata"},
  "clue.update": {"id": "cN", "content?": "string",
                  "reason?": "string", "metadata?": "CandidateMetadata"},
  "clue.discard": {"id": "cN", "reason?": "string"}
}
```

Listing 3: Compact Clue tool interfaces; CandidateMetadata is defined in Listing 1.

`clue.add` requires `content` and defaults to Evidence; adding a Candidate must use structured metadata under the model instructions, even though no metadata member is schema-required. Candidate creation forces `metadata.status=active`; `clue.update` requires only `id` and can set a Candidate to `ready` or `rejected`, including via a metadata-only update. Omitted Candidate metadata on update inherits its previous normalized value. `clue.discard` requires only `id` and removes Evidence, not Candidates. The model cannot change record IDs, kinds, or runtime provenance directly. Clue tools return a one-line observation; the runtime separately logs mutations. Invalid calls, missing or inactive IDs, empty Evidence writes, and Candidate discard attempts produce no transition. Successful writes, including Evidence updates that preserve record content, are logged. Offline attribution selects a representative write for validated final Evidence content; Candidate transitions are assessed separately for progress (Appendix B). Submission is plain assistant text without tool calls and must satisfy the readiness rules in Appendix A.3.

A.2 Context Construction and Browser Provenance

At each turn, the model input follows Equation 2: fixed system instructions and the question, a freshly rendered S_t, and the bounded history $H_t^{(w)}$, containing up to w most recent complete turns. Tool definitions are supplied separately through native function calling. Research-state and task-state messages are rebuilt before each call. Active Evidence is rendered with IDs and content; Candidate rendering includes answers, status, support IDs, satisfied and missing constraints, contradictions, and next checks, marking ready candidates and summarizing rejected ones. Task state reports the budget and, when relevant, stagnation and recent queries. A history turn contains the assistant message, its tool calls, and their responses. Rendering S_t outside this rolling history keeps active records visible after their original observations leave the window; discarded Evidence may remain only in recent history until those turns leave the window. Runtime provenance, lifecycle indices, pending sources, and the full trajectory are not shown to the model.

All tool calls in one response are sampled from the same pre-turn input. The executor preserves order within each tool family but runs Clue calls before Browser calls: successful Clue operations update state, then Browser results enter history. Thus, browser observations returned in turn t can support Evidence writes only from turn $t + 1$ onward.

The runtime records each browser action's type (`search`, `open`, or `find`), query, URL, title, tool-call ID, and browser step/turn IDs. Clue records also log source candidates, creation/update steps, and discard reasons; attribution confidence and method are tracked separately. These bookkeeping fields are runtime-only and omitted from the rendered state.

An ordered pending pool $\mathcal{P}_t$ retains at most 80 recent successful, nonempty browser actions. Before executing a turn's calls, the runtime snapshots $\mathcal{P}_t$; every successful Evidence ADD or UPDATE in that turn records the same candidate-source snapshot $\widetilde{\mathcal{P}}(\tau) = \mathcal{P}_t$. A singleton is assigned as a direct source; larger snapshots remain unresolved online. Let $\mathcal{B}_t^+$ be the successful, nonempty browser actions from turn t, and let $\chi_t = 1$ if at least one Evidence write succeeds, including a semantic no-op. After the Clue batch, the old snapshot is consumed only once:

$$\mathcal{P}_{t+1} = \begin{cases} \mathcal{B}_t^+, & \chi_t = 1, \\ \mathcal{P}_t \cup \mathcal{B}_t^+, & \chi_t = 0. \end{cases}$$

Candidate operations, Evidence DISCARD, and failed Clue calls do not consume the pool. Malformed calls, empty or error observations, no-result searches, and no-match finds are excluded from $\mathcal{B}_t^+$. If the pool is empty but a current page exists, its URL/title may be copied as weak context without assigning a direct Browser-step source. The pending records capture possible sources given temporal visibility; offline source tracing selects the action that supports the claim (Appendix B).

A.3 Readiness and Submission

A Candidate is ready only if its decision status is `ready`, its answer is nonempty, and it has no missing constraints, contradictions, or unresolved rival. A rival is a nonequivalent, unrejected Candidate with no contradictions and at least as many satisfied and no more missing constraints. A plain-text submission is accepted only when exactly one Candidate is ready; matching the submitted text to that Candidate is diagnostic, not an acceptance condition. Runtime readiness does not require nonempty `supporting_clues`, whereas offline Candidate credit requires support IDs that resolve to active, approved Evidence.

A.4 Agent System Prompt

The following is the standard ClueFlow agent-facing system prompt. Tool schemas are supplied separately through native function calling; the question and rendered state are separate messages (Appendix A.2).

```
You are a clue-based search agent.

Your task is to answer the user's question using browser tools to search
    and read web pages, and clue tools to maintain a structured evidence
    memory.

Available tools:

Browser tools:
- browser.search: search the web.
- browser.open: open a search result or URL.
- browser.find: find text in an opened page.

Clue tools:
- clue.add: add a typed clue from the latest observation. Use kind='
    evidence' for facts and kind='candidate' for plausible answers with
    structured metadata.
- clue.update: update an existing evidence or candidate clue. For
    candidates, this is the only way to change status to ready or
    rejected.
- clue.discard: discard an evidence clue. Do not use it for candidate
    rejection.

Clue policy:
Clues are persistent evidence memory, not private thoughts.
Only ADD facts stated in tool results (browser observations). Do not ADD
    clues from the question itself.

Be active: if a tool result names a candidate entity, title, date, role,
    relation, event, author, work, adaptation, office, or constraint-
    relevant fact, prefer ADD.

The question itself is not an evidence source, but it is a source of
    search constraints.
You may use phrases, entities, dates, roles, relations, and tentative
    interpretations from the question to form search queries.

Source-specific constraint:
For questions asking "according to [source]", "on page X", "in a
    newspaper", "in a review", or similar source-specific phrasing, the
    final answer MUST come from that specific source context. Do not ADD
    general background facts as answer-supporting clues unless they come
    from the required source or directly help identify/locate that source
    .
```

```
Binding clues:
When an observation fact satisfies a question constraint, ADD a binding
    clue that states the observation fact together with which constraint
    it satisfies.

UPDATE only when a tool result corrects, broadens, narrows, or
    supplements an existing clue.
DISCARD clues that are contradicted, stale, over-specific, misleading, or
     repeatedly unproductive.


Candidate policy:
Evidence clues are not enough to finish. You may answer only through
    Candidate State.
Use clue.add with kind='candidate' when a plausible answer appears.
    Candidate add always creates an active candidate, even if another
    status is supplied. Candidate clues must include metadata:
- answer: proposed final answer string.
- status: active on add; use clue.update later for ready or rejected.
- supporting_clues: active evidence clue ids supporting this candidate.
- satisfied_constraints: question constraints already covered by evidence
    .
- missing_constraints: hard constraints or checks still not proven.
- contradictions: observed evidence that conflicts with this candidate.
- next_check: the next concrete browser action needed to resolve the
    candidate.

Update candidate clues whenever new evidence changes which answer is
    plausible, which hard constraints are covered, what remains missing,
    or what contradiction was found.
Reject a candidate only with clue.update by setting metadata.status='
    rejected' when a hard contradiction is found.
Mark metadata.status='ready' only with clue.update, and only when the
    answer is explicit, missing_constraints is empty, contradictions is
    empty, and no stronger rival candidate remains.

A free-text candidate clue without structured metadata is not sufficient
    for finish readiness.

Do not treat many related evidence clues as sufficient to answer. If no
    candidate is ready, use the candidate's next_check to decide the next
     browser.search/open/find action. If a candidate is ready, finish
    with that candidate instead of continuing to search.

Answer guidance:
When a candidate clue is ready, provide your final answer as plain text
    with NO tool calls.
The answer must be a short plain string: only the entity, title, name,
    number, date, location, or phrase being asked for.
Do not use markdown, bold text, explanations, citations, or leading
    phrases.

Search/open guidance:
When search results contain a promising source, prefer opening it instead
     of repeatedly reformulating queries.
If two consecutive searches return empty or low-information results,
    switch strategy: open a result, broaden the query, or answer from
    grounded clues.
```

B ATTRIBUTION DETAILS

B.1 EVIDENCE VALIDATION

Attribution is performed after trajectory execution without exposing reference answers or judge outputs to the search policy. Listing 4 summarizes the Evidence judge input and output (strings denote field types). Evidence comes from active and discarded records, deduplicated by ID. Candidate records are validated separately against the reference answer, with additional structural checks on their transitions. For long trajectories, batches of at most 20 Evidence records repeat the question, answers, and allowed labels.

```
{
  "input": {
    "question": "string", "gold_answer": "string",
    "gold_aliases": ["string"],
    "predicted_answer": "string | NO_ANSWER",
    "allowed_labels": ["label"],
    "clues": [{"clue_id": "string", "kind": "evidence",
      "status": "active | discarded", "discard_reason": "string | null",
      "source_action": "search | open | find | null",
      "source_url": "string | null", "source_title": "string | null",
      "content": "string"}]
  },
  "output": {"labels": [{"clue_id": "cN",
    "label": "<allowed label>", "rationale": "<one sentence>"}]}
}
```

Listing 4: Schematic input and output contract for Evidence validation.

The judge assigns one of eight labels per Evidence record:

Label	Meaning	Positive?
SUPPORT_GOLD	Establishes the gold answer, alias, or required component.	Yes
EXCLUSION_GOLD	Rules out a wrong candidate or narrows toward gold.	Yes
SCAFFOLDING_RELEVANT	Provides an intermediate entity, date, or constraint anchor.	No
SUPPORT_PRED	Supports a non-gold prediction; absent for no-answer runs.	No
SUPPORT_WRONG	Favors another incorrect candidate as the answer.	No
CONTRADICT_GOLD	Conflicts with gold or its required entity binding.	No
IRRELEVANT	Adds no useful support, exclusion, or intermediate anchor.	No
UNCLEAR	Insufficient to judge; fallback for an invalid label.	No

The prompt requires one output object per input clue in the same order and a short rationale. Its principal semantic rules are that a gold-string match is insufficient without the correct entity binding; mentioning a wrong candidate may constitute useful exclusion rather than support for a wrong answer; intermediate anchors should not be collapsed into irrelevance; entity identity outranks coincident dates or numbers; and unsuccessful search is not evidence of absence. Invalid labels are normalized to UNCLEAR. Incomplete or unparseable batches trigger retry and fallback handling; unresolved records do not enter a strict positive chain.

Under strict Source–State–Final attribution, an Evidence record must receive an eligible positive label, remain active at the end of the trajectory, and have its ID listed in the supporting_clues of the Candidate associated with the submitted answer. SCAFFOLDING_RELEVANT remains useful for analysis but is not credited in the strict final-support chain.

B.2 EFFECTIVE TRANSITION RECOVERY

The trajectory stores an operation ledger of applied, non-skipped ADD, UPDATE, and DISCARD events $v = (\mathrm{op}, \mathrm{kind}, S^-, S^+, \mathrm{prov})$. For a judge-labeled Evidence record, we normalize whitespace and select the last Evidence ADD or UPDATE whose post-operation content matches its final content. This representative-write rule may select an identical later update that leaves the content unchanged; earlier matching writes receive no separate credit, and differing writes are superseded. A DISCARD and its record remain available for auditing and judging but are never selected as a positive content write. If no matching write exists, or the selected write lacks a mapped assistant

turn or reliable browser source, the Evidence chain is dropped rather than reassigned to an earlier write.

Candidate recovery is performed separately after semantic validation. The operation must modify the final Candidate associated with the submission, its normalized answer must equal the submitted answer, and its `supporting_clues` must contain at least one Evidence ID whose semantic and source links both survive validation. Such an operation advances the chain if it adds newly approved support, first creates or revises the final answer content, or performs a genuine transition into `metadata.status = ready`. An operation that changes none of the answer, approved support, or readiness is a semantic no-op and is excluded.

B.3 Source Tracing

Each effective Evidence write is matched only to browser observations available before the write (Appendix A.2). We first consider the pending browser actions recorded at write time. If no reliable match is found, we expand the candidate set to actions with an exact recorded URL and the 80 most recent prior browser actions.

A deterministic matcher combines document IDs, URLs, search queries, and content coverage, defined as the fraction of normalized Evidence tokens present in an observation. Titles and recency provide auxiliary ranking signals but cannot independently establish a reliable link. The best-matching source is retained only if it passes fixed source-identity or content-based reliability checks. Accepted sources must be successful `search`, `open`, or `find` actions, and both the source and Evidence write must map to assistant turns. Evidence without a qualifying source is excluded from positive attribution.

For a correct trajectory, each surviving chain connects the matched browser action, validated state transitions, and the final submission:

$$\text{Source} \longrightarrow \Delta^{E} \longrightarrow [\Delta^{C}]_{\text{optional}} \longrightarrow \text{Final}. \tag{10}$$

Incorrect trajectories retain only validated Source–State partial chains within the final support set of the Candidate associated with the submitted answer. Unfinished trajectories receive no positive attribution. If no valid Source–State chain survives, PPO falls back to outcome-only learning.

C Training and Evaluation Details

C.1 SFT Data and Training

Table 4 summarizes the data and principal training settings. Both backbones receive full-parameter SFT in each environment; the controlled SFT ablations use Qwen3-30B-A3B-Thinking-2507.

Setting	Fixed corpus	Live web
QA source	OpenResearcher QA	REDSearcher QA
Training questions	3,657	7,551
Correct / incorrect teacher trajectories	2,927 / 730	6,300 / 1,251
SFT examples (one-turn / multi-target chunk)	34,783 / 44,031	38,962 / 75,557
Chunk size K	5	10
Global batch size	64	64
Maximum sequence length	131,072	163,840
Shared optimization		
Epochs / learning rate	3 / 1×10^{-5}	
Schedule / warmup	Cosine decay / 0.1 warmup ratio	

Table 4: SFT data and training for the two environments.

Supervision selection. In correct trajectories, assistant responses are supervised unless they contain state operations judged harmful after execution. In incorrect trajectories, only validated Source or State turns without harmful operations are supervised; incorrect final submissions and other turns are masked. When a turn combines validated and harmful operations, its entire response is masked rather than supervising individual tool calls. Masked responses may still appear in later recorded contexts.

Chunk construction and masking. These counts are complementary parts of one hybrid export, not two encodings of the same turns. We group successive assistant turns into non-overlapping multi-target chunks of at most K turns using the last turn's recorded input. Only supervised, non-attributed assistant responses receive loss within a chunk; attributed turns use separate one-turn examples in their original contexts. Other responses and context tokens are masked. Fixed-corpus and live-web SFT retain 10 and 20 visible turns, respectively.

C.2 PPO Data and Training

Table 5 reports the PPO sampled training questions and rollout settings, rather than SFT teacher-trajectory counts. Fixed-corpus 35B follows the fixed-corpus 30B configuration.

Setting	Fixed 30B / 35B	Live 30B / 35B
QA source	OpenResearcher QA	REDSearcher QA
Training questions sampled	800	1,000
Trajectories per rollout batch	20	40
Rollout batches	40	25
Optimization batch size	20	40
History window (turns)	10	20
Shared configuration		
Samples per rollout prompt	1	
PPO epochs per rollout batch	2	
Actor / critic learning rate	10^{-6} / 10^{-5}	
Policy / value clip	0.1 / 0.2	
GAE γ / λ	1 / 0.995	
Temperature / top-p	1.0 / 0.95	

Table 5: PPO data and training.

The outcome reward and Source/State/Final advantage bonuses are defined in Section 3.5 (Equation 7). Advantage normalization is disabled, preserving the scale of these fixed bonuses; the KL and entropy coefficients are both zero. The critic bounds its values to $[-0.1, 1]$ and applies loss only at the pre-response boundary of each valid assistant turn, against the unmodified turn-level return.

C.3 Evaluation Protocols

We evaluate on all 830 BrowseComp-Plus (BCP), 1,266 BrowseComp (BC), 289 BrowseComp-ZH (BC-ZH), and 100 xBench-DeepSearch-2510 questions. Offline training uses the OpenResearcher[1] corpus and released document embeddings, indexed locally with FAISS IVF. BCP evaluation uses the BrowseComp-Plus[2] corpus and precomputed embeddings released by Tevatron. Both use Qwen3-Embedding-8B for retrieval. Live-web benchmarks use Serper search and Jina Reader for page fetching.[3] Both live-web and fixed-corpus evaluations allow up to 800 Browser tool calls in total per question. xBench-DeepSearch-2510 scores are averaged over six evaluation runs; all other benchmarks are evaluated once.

Inference-time context management. For the GLM-5.1 BrowseComp comparison in Section 4.4, Discard-all drops the visible trajectory once it exceeds 64K tokens. Keep-k uses $k = 5$, retaining

[1] `https://huggingface.co/collections/TIGER-Lab/openresearcher`

[2] `https://huggingface.co/collections/Tevatron/browsecomp-plus`

[3] Serper: `https://serper.dev/`; Jina Reader: `https://github.com/jina-ai/reader`.

the five most recent rounds' observations while folding older ones. HCM applies the same keep-5 folding and discards the tool-call history if the folded context still exceeds 64K tokens. ClueFlow retains the most recent $w = 20$ turns alongside its persistent research state. The same 64K threshold is used by HCM and Discard-all on BCP in Table 7.

The gold-conditioned GPT-5.5 attribution judge validates Evidence and Candidate records for training and PPO credit, not benchmark accuracy. For final-answer scoring, normalized matches are accepted first; otherwise an answer-only GPT-5.5 judge compares the submitted answer with the reference without viewing the search trace. Unanswered or unresolved questions count as incorrect; PPO evaluation retries failed judge calls and may use a configured fallback.

D Additional Results and Analyses

D.1 Additional Context-Management Baselines

We evaluate additional context-management baselines on the full BrowseComp-Plus evaluation set. Four inference-time strategies reuse the same ReAct SFT model. We separately train Keep-5 SFT on teacher trajectories collected under Keep-k ($k = 5$) and evaluate it under Keep-5 and HCM. Table 6 compares their training data with ClueFlow SFT.

Training-data property	ReAct SFT	Keep-5 SFT	ClueFlow SFT
Teacher rollout configuration	ReAct	Keep-5	ClueFlow
Distinct training questions	2,927	2,927	3,657
Successful teacher trajectories	2,927	2,927	2,927
Incorrect trajectories used	0	0	730[†]
SFT serialization	Full trajectory	One-turn	One-turn + chunks
Serialized training examples	2,927	117,877	78,814
Attribution-guided construction	No	No	Yes

Table 6: SFT data and trajectory construction for the context-management comparison.

Notes. All three use Qwen3-30B-A3B-Thinking-2507, a GLM-5.1 teacher, and OpenResearcher QA with offline retrieval. The three training-question sets are not identical. [†] Of ClueFlow's 730 incorrect trajectories, only validated intermediate steps, not complete trajectories, are used. Keep-5 one-turn examples retain the teacher-visible folded context. Serialized-example counts do not directly measure training tokens.

SFT model	Inference-time context strategy	Acc. (%)
ReAct	Full history	28.55
ReAct	Keep-5	29.64
ReAct	HCM	35.06
ReAct	Discard-all	41.81
Keep-5	Keep-5	32.89
Keep-5	HCM	37.11
ClueFlow w/o failure recovery	Persistent state + bounded window	46.27
ClueFlow	Persistent state + bounded window	**48.31**

Table 7: Additional context-management baselines on BrowseComp-Plus. The four ReAct rows share the same SFT checkpoint and differ only in inference-time context management. HCM and Discard-all use a 64K-token context-management threshold. All evaluations use a maximum budget of 800 browsing calls per question, and all 830 questions remain in the accuracy denominator.

Context management substantially improves the ReAct SFT baseline. Discard-all achieves the strongest result among the tested ReAct configurations, increasing accuracy from 28.55% to 41.81% (+13.26 percentage points). Keep-5 SFT exceeds ReAct SFT under the corresponding Keep-5 and HCM inference strategies by 3.25 and 2.05 percentage points, respectively. Because their teacher trajectories and training questions differ, this comparison does not isolate the effect of bounded-context training.

Using only successful teacher trajectories, ClueFlow SFT without failure recovery reaches 46.27%, 4.46 percentage points above the strongest tested non-ClueFlow baseline, ReAct SFT with Discard-

all (41.81%). Adding validated intermediate steps from incorrect trajectories raises accuracy by 2.04 points to 48.31% (6.50 points above the same baseline). Thus, context management narrows the gap between full-history ReAct SFT and ClueFlow SFT, but the tested alternatives do not fully close it. These comparisons assess complete training and execution configurations; the controlled SFT ablations in Section 4.3 separately examine the contribution of attribution-guided context selection.

The relative ranking of HCM and Discard-all is reversed across the two evaluation settings. HCM performs better with GLM-5.1, whereas Discard-all is substantially stronger for the Qwen3-30B SFT policy. This suggests that the utility of retaining historical reasoning context may be policy dependent. One plausible explanation is that a stronger policy such as GLM-5.1 can more effectively exploit retained or hierarchically compressed reasoning traces to preserve long-horizon continuity, while a weaker SFT policy may be more susceptible to distraction from stale, incomplete, or incorrect intermediate reasoning. In the latter case, aggressively discarding old interaction history can act as a more robust form of context control. We emphasize, however, that the two settings also differ in backbone, training regime, and evaluation environment; therefore, the observed inversion should be treated as a model-dependent empirical observation rather than evidence that model capability alone determines the preferred context-management strategy.

D.2 PPO Rollout Behavior

Table 8 complements the PPO ablations in Table 2(b) with rollout statistics and post-hoc attribution diagnostics.

Table 8: Additional rollout, behavioral, and post-hoc attribution statistics for the BrowseComp-Plus PPO ablations.

(a) Rollout statistics

	Avg. browse calls		
Variant	Correct	Non-correct	Unfinished (#)
Outcome-only	82.21	319.69	20
Source-only attribution	106.91	415.37	64
Full-chain attribution	88.76	396.30	24
+ Failure recovery	87.26	428.76	15

(b) Behavioral and attribution diagnostics

Variant	Inspect (%)	Raw State / traj	Valid State / traj	State conv. (%)	S–S Cov. (%)
Outcome-only	16.01	4.743	1.871	39.45	55.54
Source-only attribution	15.93	3.908	1.636	41.86	48.43
Full-chain attribution	23.25	4.367	1.933	44.25	57.71
+ Failure recovery	27.09	4.301	1.904	44.26	59.04

In (a), Non-correct includes incorrect-answer and unfinished trajectories; Unfinished counts the unfinished subset. In (b), Inspect denotes the fraction of Browser calls that are `open` or `find`; Raw State counts state-update turns before validation, while Valid State counts turns retained by strict post-execution attribution. State conv. is the fraction of raw State turns retained as validated State progress. S–S Cov. is the percentage of evaluation trajectories containing at least one strictly valid Source–State chain. All variants use the same 800-call browsing budget. Post-hoc attribution is used only for analysis and does not affect inference or benchmark scoring.

Full-chain attribution changes more than the frequency of browser actions. Relative to Source-only attribution, it increases both the fraction of inspection actions and validated State progress, with Source–State coverage rising from 48.43% to 57.71%. This is not simply due to more state writing: relative to outcome-only PPO, Full-chain produces fewer raw State-update turns (4.367 vs. 4.743) but more validated State turns (1.933 vs. 1.871), increasing State conversion from 39.45% to 44.25%. Failure recovery further increases inspection share, while its validated State statistics remain similar to Full-chain attribution.

E QUALITATIVE CASE STUDIES

Turn numbers, tool calls, observations, and quoted Think excerpts come from saved trajectories; ellipses mark omissions and line wraps are editorial. The turn-8 input snapshot abbreviates a logged request while preserving role order. Task summaries, state annotations, highlights, and Source/State/Final labels are post-hoc, not agent-visible. Bonuses are PPO actor advantage bonuses (Equation 7), not environment rewards. Neither trajectory calls `clue.discard`.

Correct submission.

Question (abridged). Name the annual two-day music and hot-air-balloon festival in Clark Global City, a planned district in a Central Luzon economic zone; its name evokes a natural phenomenon.
Reference answer. Aurora Music Festival. **Submitted answer.** Aurora Music Festival.

Turn 4: Competing event [turns 1–3 omitted] Unattributed

Think (excerpt). *"Let me check if there's an event specifically at Clark Global City."*

Assistant tool calls

```
browser.open({"id":12})
browser.search({"query":"Lubao International Balloon Music
  Festival art installations","topn":10})
```

Tool observations (excerpts)

```
browser.open: [2] Philippine International Hot Air Balloon Fiesta
  L22: ... an annual four-day air-sporting event ...
browser.search: [33] Lubao International Balloon and Music Festival
    2026 ...
```

Turn 6: Source page [turn 5 omitted] Source
Think (excerpt). *"Let me check the Wikipedia page for the Aurora Music Festival."*
Assistant tool call: browser.open({"id":43})
Tool observation (excerpt; hyperlinks omitted)

```
[3] ## First edition and COVID-19 pandemic (73 lines total)
L1: Title: Aurora Music Festival
L3: URL Source: https://en.wikipedia.org/wiki/Aurora_Music_Festival
L24: ... art installations, and hot air balloons held in
     Clark Global City ... over the course of two days.
```

Turn 7: Record three Evidence clues State

Think (excerpt). *"The Wikipedia page for the Aurora Music Festival explicitly says it's held in Clark Global City. … Let me add clues and create a candidate."*

Assistant tool calls (one turn; selected arguments)

```
clue.add({"kind":"evidence","content":"The Aurora Music Festival
  ... held in Clark Global City ... two days. [Wikipedia]",
  "reason":"Matches all key constraints ..."})
clue.add({"kind":"evidence","content":"Clark Global City is a planned
  central business district ... [Search results 18, 23]", ...})
clue.add({"kind":"evidence","content":"Pampanga province was created
  in 1571 ... [Wikipedia/Doc 49-52]", ...})
```

Tool observations (same call order)

```
ok: added c1 (1 active); ok: added c2 (2 active);
ok: added c3 (3 active)
```

State after execution (author annotation). The complete `c1` Evidence record (line-wrapped) is:

```
{"id":"c1","kind":"evidence","metadata":{},"status":"active",
 "content":"The Aurora Music Festival (also called the
 \"Clark Hot Air Balloon Music, Arts and Food Festival\") is
 an annual music and arts festival featuring live music, food,
```

```
 art installations, and hot air balloons held in Clark Global
 City, Pampanga, Philippines over the course of two days.
 [Wikipedia]"}
```

`c2` records Clark Global City, the SEZ, and the Bank of China loan; `c3` records Pampanga's 1571 establishment. Both remain active.

Turn 8: Create the Candidate State

Model input (logged request, abridged). The rendered state precedes the recent assistant/tool history:

```
system: You are a clue-based search agent. ...
user: In a Southeast Asian nation ... two-day event ... name?
system: [Evidence Clues] c1: Aurora Music Festival ...
        c2: Clark Global City ...; c3: Pampanga ... 1571 ...
        [Candidate State] (no candidate clues)
assistant (t6; prior turns omitted): browser.open({"id":43})
tool (browser.open): [3] Aurora Music Festival ... L24: ... two days.
assistant (t7): clue.add(...) x 3
tool (clue.add): ok: added c1; ok: added c2; ok: added c3
```

Think (excerpt). *"Now let me add the candidate clue with the answer."*

Assistant tool call (selected arguments)

```
clue.add({"content":"Aurora Music Festival","kind":"candidate",
  "metadata":{"answer":"Aurora Music Festival","status":"ready",
  "supporting_clues":["c1","c2","c3"], ...}})
```

Tool observation

```
ok: added c4 (4 active)
```

State after execution (author annotation). `c4` (constraints list abridged):

```
{"id":"c4","content":"Aurora Music Festival","kind":"candidate",
 "status":"active",
 "metadata":{"answer":"Aurora Music Festival","status":"active",
  "supporting_clues":["c1","c2","c3"],
  "satisfied_constraints":["Southeast Asian nation (Philippines)",
    ... [six other items omitted] ...],
  "missing_constraints":[],"contradictions":[],"next_check":""}}
```

The top-level `status` tracks record lifecycle; `metadata.status` tracks Candidate readiness. The call requested `ready`, but creation initialized `metadata.status=active`.

Turn 10: Update readiness [turn 9 omitted] State

Think (excerpt). *"The candidate has all constraints satisfied, no missing constraints, no contradictions, and no next check needed. ... Let me try updating the candidate with status ready."*

Assistant tool call (unchanged fields omitted)

```
clue.update({"id":"c4","metadata":{"answer":"Aurora Music Festival",
  "status":"ready","supporting_clues":["c1","c2","c3"], ...}})
```

Tool observation

```
ok: updated c4 (4 active)
```

State change (author annotation; this update). `c4.metadata.status: active → ready`; `c4.status` stays `active`. The answer and `supporting_clues=[c1,c2,c3]` do not change.

Turn 11: Submit the answer Final

Think (excerpt). *"I should answer with the plain text answer."*

Assistant final answer (no tool call): `Aurora Music Festival`

Post-hoc attribution. Validated `c1` (`SUPPORT_GOLD`) links turn 6 to turn 7 and supports `c4`; `c2`/`c3` supply context without independent credited chains.

Turn	Role	Reason	Actor advantage bonus
4	Unattributed	Competing festival and exploratory search	0
6	Source	Matched page for `c1`	+0.03
7	State	Validated `c1` write (one turn, three `clue.add` calls)	+0.01
8	State	Validated Candidate creation with `c1` support	+0.01
10	State	`c4.metadata.status`: `active`→`ready`	+0.01
11	Final	Correct answer submission	+0.04

Incorrect submission.

Question (abridged). Which specific Pakistani TV, fashion, and music awards ceremony was held at a major Dubai convention venue? The full question adds historical-government and public-figure clues. **Reference answer.** 3rd Hum Awards. **Submitted answer.** Hum Awards.

Turn 13: Unrelated find [turns 1–12 omitted] Unattributed

Think (excerpt). *"Let me look at the notable residents section."*

Assistant tool call

```
browser.find({"pattern":"Notable past residents",
  "cursor":1,"context_lines":5})
```

Tool observation (excerpt)

```
Found 2 match(es) for 'Notable past residents' in [1]:
L230: ... Aqeela Asifi ... Nansen Refugee Award ...
```

Turn 18: Awards page [turns 14–17 omitted] Source

Think (excerpt). *"Actually, let me do one more verification. Let me check the specific name of the ceremony."*

Assistant tool call

```
browser.open({"id":6,"loc":0,"num_lines":50})
```

Tool observation (excerpt; hyperlinks omitted)

```
[2] Hum Awards (81 lines total) [showing L1-L50]
L1: Title: Hum Awards
L15: Country: Pakistan
L22: Hum Awards are annual Pakistani entertainment awards
     ... Pakistani television, fashion and music ...
L28: ... beginning with the 3rd Hum Awards in Dubai in 2015.
L40: | 3rd Hum Awards | 9 April 2015 | ... |
     Dubai World Trade Centre
```

Turn 19: Write valid Evidence State

Think (excerpt). *"The remaining parts of the chain (three historical polities, Afghanistan public figure) ... but they're not strictly necessary to answer the question ..."*

Assistant tool call (content excerpted)

```
clue.add({"kind":"evidence","content":"Hum Awards match all
  core criteria ... The 3rd Hum Awards was held at Dubai
  World Trade Centre in 2015.","reason":"Confirms all three
  core identifying features of the ceremony ..."})
```

Tool observation

```
ok: added c3 (3 active)
```

State after execution (author annotation). The `c3` Evidence record has these fields (content excerpted):

```
{"id":"c3","kind":"evidence","metadata":{},"status":"active",
 "content":"Hum Awards match all core criteria ... The
  3rd Hum Awards was held at Dubai World Trade Centre in 2015."}
```

Turn 20: Create an incorrect Candidate Unattributed

Think (excerpt). *"Now let me create a candidate clue with the answer."*

Assistant tool call (selected arguments)

```
clue.add({"content":"Hum Awards","kind":"candidate",
  "metadata":{"answer":"Hum Awards","status":"ready",
  "supporting_clues":["c1","c2","c3"],
  "missing_constraints":["Verifying Afghanistan public figure
  connection","Verifying three historical polities ..."], ...}})
```

Tool observation

```
ok: added c4 (4 active)
```

State after execution (author annotation; selected fields). `c4` has `kind=candidate`, `content="Hum Awards"`, top-level `status=active`, `metadata.answer="Hum Awards"`, `metadata.status=active` (despite requested `ready`), and `supporting_clues=[c1,c2,c3]`. Two constraints remain missing. Evidence `c3` already names the *3rd* edition, but `c4` does not.

Turn 36: Update readiness [turns 21–35 omitted] Unattributed

Think (excerpt). *"So I'm confident the answer is the Hum Awards. Let me update the candidate and mark it as ready."*

Assistant tool call (unchanged fields omitted)

```
clue.update({"id":"c4","metadata":{"answer":"Hum Awards",
  "status":"ready","missing_constraints":[], ...}})
```

Tool observation

```
ok: updated c4 (4 active)
```

State change (author annotation; before/after in turn 36). `c4.metadata.status`: `active` → `ready`; `missing_constraints`: two checks → `[]`. `metadata.answer="Hum Awards"` and `supporting_clues=[c1,c2,c3]` remain unchanged.

Turn 37: Submit an incorrect answer Unattributed

Think (excerpt). *"Let me provide the final answer."*

Assistant final answer (no tool call): `Hum Awards`

Post-hoc attribution. Evidence `c3` is `SUPPORT_GOLD`, links to turn 18, and is cited in the submitted Candidate's `supporting_clues`. It retains a Source–State partial chain even though `c4` and the answer omit the edition. Incorrect Candidate operations and the final submission receive no positive credit.

Turn	Role	Reason	Actor advantage bonus
13	Unattributed	Unrelated `find`	0
18	Source	Matched awards page for `c3`	+0.03
19	State	Validated `c3` write naming the 3rd edition	+0.03
20	Unattributed	Candidate answer remains *Hum Awards*	0
36	Unattributed	Marks that answer `ready`, still no edition	0
37	No Final credit	Incorrect answer submission	0

F LIMITATIONS AND FUTURE WORK

The present study instantiates state-transition attribution for long-horizon search using Evidence and Candidate records. Generalizing this learning interface to broader agent settings calls for state abstractions and progress-validation criteria tailored to different tasks. Future work will explore extensions to open-ended research and multimodal interaction, examining how attributed state transitions can support learning beyond answer-oriented search.