EAV: EVIDENCE ANCHORING AND VERIFICATION FOR RELIABLE AGENTIC SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic search systems can return incorrect answers even after retrieving relevant evidence, a pattern we trace to three failure modes: evidence blindness, where retrieved documents go uninspected or are used ineffectively; verification failure, where self-review reconsiders existing reasoning without new falsifying evidence; and error propagation, where erroneous intermediate conclusions are reused downstream. Existing remedies either re-examine the agent's own reasoning or seek agreement across multiple complete runs; the former lacks new evidence, and the latter multiplies cost. We propose the Evidence Anchoring–Verification (EAV) framework. During execution, EAV archives raw tool observations and anchors the claims adopted by the executor, together with their sources and conflicts, into a compact evidence chain; before an answer is delivered, an independent tool-calling verifier re-checks high-risk premises against fresh evidence. A rejected candidate triggers diagnosis-guided re-execution that inherits the evidence memory; EAV-Flash and EAV-MAS apply this loop at the task level and at subtask handoffs, respectively. Across nine matched-backbone settings on GAIA (text-only validation), xBench-DeepSearch, and BrowseComp-ZH, EAV-Flash ranks among the leading methods in every setting and attains an average success rate of 82.64%, 3.69 points above MiroFlow, the previous state-of-the-art system, while using 56.1–72.4% fewer tokens; EAV-MAS reaches 80.91%, a 1.96-point gain over the same baseline. With DeepSeek-V4-Flash, EAV-Flash sets the open-source state of the art on xBench-DeepSearch (89.00%) and the state of the art on BrowseComp-Plus (92.89%). On a BrowseComp-Plus subset, archiving, anchoring, and verification contribute gains of 5.8, 7.1, and 7.8 points; anchoring mainly raises error-interception recall, and verification-driven sparse retries offer a better performance–cost trade-off than majority voting and can be combined with voting for further gains.

1 INTRODUCTION

Agentic search requires systems driven by large language models to autonomously complete the full loop of planning, retrieval, reading, reasoning, and answering within a limited budget Yao et al. (2023); Mialon et al. (2023). Retrieval alone does not guarantee a correct answer; recent analyses have also revealed limitations in *evidence use and verification*: on BrowseComp-Plus, the Pi-Serini configuration with 97.3% evidence recall achieved the lowest accuracy (76.0%) Hsu et al. (2026); ArcticSwarm found that in nearly six out of ten failure cases, the reference answer had already appeared in the trajectory, which the authors characterize as a "verification failure, not a retrieval failure" Yoon et al. (2026); and AtlasNav attributed 77.34% of the previous SOTA's errors to evidence being lost before it was fully localized Guo et al. (2026).

We reproduce the previous SOTA, MiroFlow Su et al. (2026), as the trajectory-analysis baseline and analyze its 227 failure cases on BrowseComp-Plus Chen et al. (2025b). The observed trajectory patterns, together with prior work, motivate the following three concerns: **evidence blindness** Guo et al. (2026), in which a relevant document may have been returned without being inspected or used effectively: in 23.3% of the failure cases, a gold document appeared in the search results but was never opened; in another 19.4%, the agent opened a gold document but still failed to use its evidence effectively; **verification failure**, in which self-review may merely reconsider existing reasoning without acquiring new evidence capable of falsifying its premises: 56.8% of the failure cases

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.

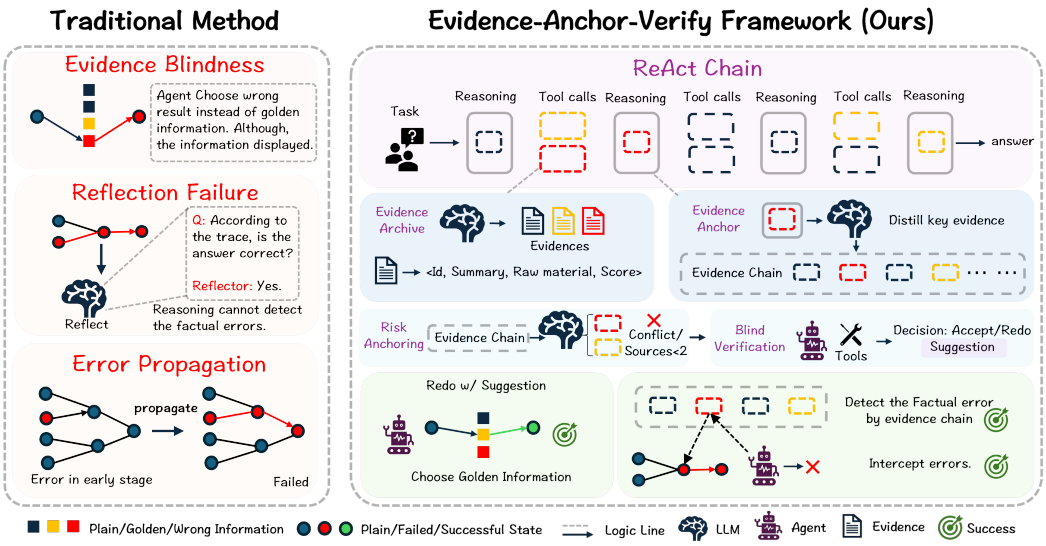

Figure 1: **Motivation and overview of EAV.** Evidence archiving, anchoring, and independent verification with diagnosis-guided re-execution address evidence blindness, verification failure, and error propagation.

contained some form of self-review, yet 76.7% of those cases still retained an incorrect candidate, corresponding to 43.6% of all failures Zhang et al. (2025); Dhuliawala et al. (2023); Shinn et al. (2023); **error propagation**, in which an erroneous intermediate conclusion may be reused in subsequent delegation, queries, or the final answer: the trajectory analysis identified this pattern in 64.3% of the failure cases; among propagation cases in which the first error could be localized to either planning or execution, 83.8% (109/130) originated during execution Su et al. (2026); Cai et al. (2026).

These descriptive frequencies are not estimates of causal effects. In this analysis, verification failure is a concern motivated by the related literature rather than a separately quantified category.

These observations motivate us to make the factual premises on which the executor relies explicit and to check them against external evidence before delivery. We propose the **Evidence Anchoring–Verification (EAV) framework** (Figure 1). After each tool return is interpreted, EAV extracts claims from the executor's reasoning and integrates them into a compact evidence chain while retaining source references and marking conflicts. The evidence chain records premises to be checked; it does not certify their correctness. It both supports subsequent reasoning and provides context for an independent, tool-using verification process. If a result is rejected, the system provides diagnostic guidance for another attempt at the task or subtask.

EAV constructs this memory within each execution attempt, without requiring agreement across sampled candidate trajectories. Table 1 compares its memory and feedback mechanisms with those of related approaches. EAV-Flash applies EAV to whole-task results, whereas EAV-MAS applies EAV at subtask handoffs. The method distinguishes premise retention, result checking, and the use of the resulting diagnosis to guide subsequent execution.

Our contributions are threefold:

- **Reasoning-derived evidence anchoring.** EAV jointly extracts and integrates claims along the reasoning trajectory of each execution attempt, without requiring consensus across candidates. It archives raw observations and retains claims, source references, and conflicts in a compact evidence chain. The same evidence chain both supports subsequent reasoning and makes the executor's premises explicit for result checking.
- **Diagnosis-guided re-execution.** EAV turns tool-grounded result checking into guidance for retrying the execution unit; it carries the diagnosis forward while inheriting the active

Table 1: Comparison of evidence management, verification, and feedback mechanisms.

Method	Evidence & verification		Feedback & target		
	Shared claim memory	Tool-based checks	Guided unit restart†	Feedback action	Check target
CoVe (2023)	—	—	—	Revise	Answer
Agent-as-a-Judge (2024)	—	✓	—	Evaluate	Requirement
Co-Sight (2025)	✓	✓	—	Repair / synthesize	Intermediates
FineVerify (2026)	—	✓	—	Stop / resample	Answer
ArcticSwarm (2026)	✓	✓	—	Corrective search	Findings / report
EAV (ours)	✓	✓	✓	**Guided rerun**	**Task / subtask**

✓: explicit in the described design; —: not specified as a component. †Diagnosis-guided re-execution of the same task or subtask with fresh active evidence memory. This is distinct from partial repair, corrective follow-up tasks, or score-based resampling. Shared claim memory serves execution and checking, not necessarily different agents; role restrictions may apply. CoVe denotes its factored variant.

evidence memory from the previous attempt. We instantiate this mechanism in EAV-Flash through bounded task-level retries and in EAV-MAS at subtask handoffs.

- **Evaluation across agentic search settings.** We evaluate EAV on GAIA val. text-only, BrowseComp-ZH, xBench-DeepSearch, and BrowseComp-Plus Mialon et al. (2023); Zhou et al. (2025); Chen et al. (2025a;b), covering open-web and fixed-corpus search as well as both single-agent and multi-agent execution.

2 Related Work

2.1 Agentic Search

Agentic search requires both effective retrieval and reliable use of the retrieved material. Related evaluations and failure analyses examine these complementary challenges, including cases in which evidence is overlooked or misinterpreted during long-horizon execution Mialon et al. (2023); Wei et al. (2025); Zhou et al. (2025); Chen et al. (2025a); Yoran et al. (2024); Hsu et al. (2026); Guo et al. (2026); Lu et al. (2026). ReAct unifies reasoning and tool calling in a single loop, WebGPT answers open-ended questions through browser interaction, and Toolformer enables models to learn to invoke tools autonomously; together, these works establish the basic paradigm of "interleaving reasoning and acting" Yao et al. (2023); Nakano et al. (2022); Schick et al. (2023). Building on this paradigm, subsequent work has developed along three directions: task decomposition and plan customization Cai et al. (2026); Zhu et al. (2025); Liu et al. (2026); Zhang et al. (2026a), parallel acceleration Li et al. (2025); Lee et al. (2026), and multi-agent orchestration Su et al. (2026); Zhang et al. (2026b). Among these methods, Flash-Searcher reduces execution latency through DAG-parallel subtasks Qin et al. (2025); this paper uses MiroFlow Su et al. (2026) as the trajectory-analysis baseline. Task decomposition, parallelization, and scheduling can improve execution efficiency, but do not by themselves establish that intermediate results are correct: MiroFlow's ablation experiments show that on strongly sequential tasks, the multi-agent system performs worse than the single-agent baseline Su et al. (2026); ArcticSwarm's failure analysis finds that in most failure cases, the reference answer had already appeared in the trajectory, which the authors characterize as a "verification failure" rather than a "retrieval failure" Yoon et al. (2026).

2.2 Reflection and Verification Mechanisms

Reflection can expose inconsistencies in reasoning, but checking factual premises may require evidence beyond the existing trajectory. EAV studies how execution-time claim memory can support such tool-grounded checking and subsequent re-execution. Checking generated outputs has long received attention: Self-Consistency replaces single-pass inference with agreement across multiple samples, multi-agent debate checks candidate responses through cross-examination, and Reflexion

verbalizes failure experience and stores it in memory Wang et al. (2023); Du et al. (2023); Shinn et al. (2023); CoVe first drafts an answer, then plans and answers verification questions, and finally uses the checks to generate a revised answer; its factored variant answers the checking questions *independently of the draft* Dhuliawala et al. (2023); Agent-as-a-Judge has a judge agent equipped with localization and retrieval tools read the trajectory and intermediate artifacts of the agent under evaluation and verify them item by item Zhuge et al. (2024). Within agentic search, verification further becomes an explicit component: FineVerify decomposes questions into checkable subquestions and uses them to verify candidate answers; Co-Sight combines conflict-aware verification with structured factual memory; ReThinker organizes parallel re-reasoning at test time; RGV reads retrieval-grounded voting signals from outside the contaminated context Zhao et al. (2026); Zhang et al. (2025); Tang et al. (2026); Kook et al. (2026); and ArcticSwarm performs structured review at commitment boundaries Yoon et al. (2026). Table 1 summarizes these mechanisms. Co-Sight's CAMV forms anchors through mutual support among candidates and audits disagreements among experts; its TRSF module also extracts facts from reasoning trajectories. FineVerify scores candidate answers against fixed requirements, whereas ArcticSwarm uses local checks, board audits, and a final commit gate to trigger corrective retrieval Yoon et al. (2026). In contrast, EAV accumulates claims along the reasoning trajectory of each execution attempt without requiring consensus across candidates. These claims remain hypotheses to be checked rather than premises certified by consensus. Result-level diagnosis guides a new attempt at the entire execution unit, whereas the repair described by Co-Sight targets affected subchains. EAV-Flash and EAV-MAS instantiate the same interface at task and subtask boundaries, respectively. The resulting design connects within-attempt premise retention with diagnosis-guided re-execution through a common interface.

3 METHOD

3.1 PRELIMINARIES

Task and execution unit Let T denote the specification of the current execution unit: the full task in EAV-Flash or the current subtask in EAV-MAS. Given T and tools $\mathcal{U}$, a ReAct executor interleaves reasoning with tool calls Yao et al. (2023). Let o_i denote the observation returned by the i-th tool call and r_{i+1} the subsequent reasoning that interprets it. An *execution unit* produces a candidate answer $\hat{y}$ and a final explanation r_f: in EAV-Flash, it solves the entire task; in EAV-MAS, it solves a subtask. We distinguish raw observations, claims inferred from those observations, and the verification verdict subsequently assigned to the submitted result. A recorded claim may be useful for reasoning even if it has not yet been independently verified.

Evidence anchoring and verification EAV combines execution-time evidence management with result-time checking through an Evidence Anchoring–Verification (EAV) layer. Its execution memory is $\mathcal{M} = (\mathcal{R}, \kappa)$, where $\mathcal{R}$ stores tool observations and κ is a compact, structured evidence chain. For a trajectory τ, we write

$$\mathcal{A} : \tau \longmapsto (\mathcal{R}, \kappa), \tag{1}$$

$$\mathcal{V} : (T, r_f, \hat{y}, \kappa) \longmapsto (\nu, g), \qquad \nu \in \{\alpha, \rho\}, \tag{2}$$

where α denotes acceptance by the verifier, ρ requests re-execution, and g contains diagnostic guidance when re-execution is requested. These symbols correspond to the implementation verdicts `accept` and `redo`; acceptance denotes a decision made by the verification procedure rather than an external correctness label. The anchoring and verification mappings define the interface shared by the two operational forms.

3.2 EVIDENCE ANCHORING–VERIFICATION LAYER

3.2.1 MEMORY MANAGEMENT

Evidence archive Each tool interaction is assigned an identifier. After its observation has been used by a subsequent reasoning step, the raw output is offloaded from the active context to a local archive. Its record contains

$$\mathcal{R} = \{(id, o_{id}, \sigma_{id}, \pi_{id})\}_{id \in \mathcal{I}}, \tag{3}$$

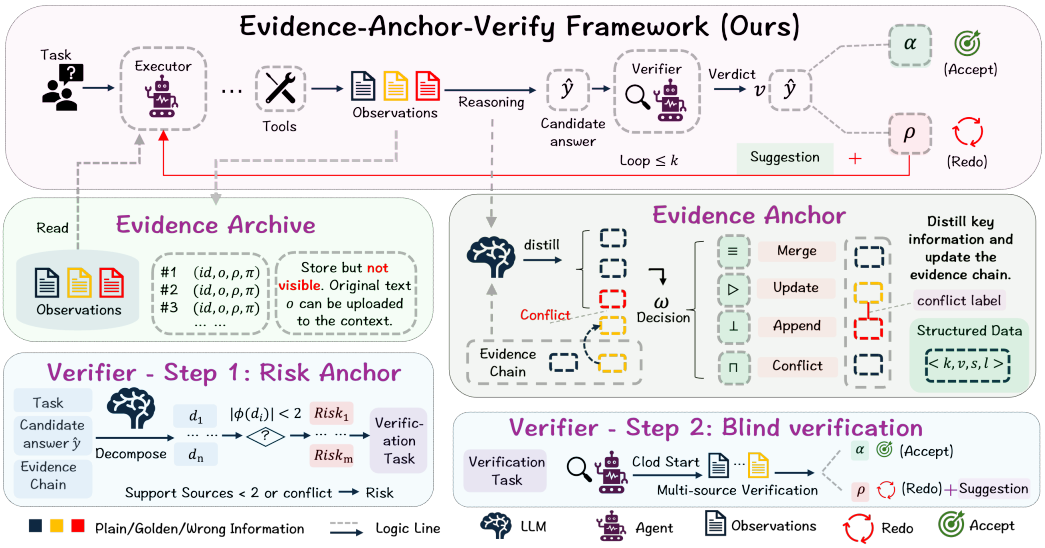

Figure 2: **EAV framework.** Archived observations and anchored claims support independent verification, whose diagnostic output guides re-execution.

where o_{id} is the original output, and a single summarization call produces its summary σ_{id} and importance score π_{id}. Summary records provide a compact overview of past interactions, while the stored observations provide material for subsequent inspection. Thus, fresh observations can first inform reasoning, after which their compressed representations are used for subsequent context management. An archive identifier refers to a tool interaction, whereas an external source identifier, such as a URL or corpus document ID, refers to the retrieved material. The former locates an interaction record, while the latter identifies its cited material; neither identifier alone can verify whether the cited source supports the associated claim. Source-to-record routing and context-policy details are provided in Appendix A.1.

Evidence chain The evidence chain κ stores task-relevant premises as entries $\mathcal{E} = \langle k, v, S, l \rangle$, where k is the topic, v is the claimed value, S is the set of cited sources, and $l \in \{0, 1\}$ is a conflict label. Unlike the archive $\mathcal{R}$, which preserves raw observations, κ records claims inferred by the executor and the sources associated with them. A source citation, chain membership, or conflict label identifies material for checking but does not constitute a correctness verdict. The chain is supplied to both subsequent reasoning and the verifier, supporting execution and checking without implying cross-agent consensus. Its serialized fields and update semantics are detailed in Appendix A.1.

3.2.2 Evidence Anchoring

After the executor reasons about a tool return, the anchoring call updates the evidence chain as

$$\kappa_{i+1} = \mathcal{F}(r_{i+1}, \kappa_i). \tag{4}$$

Within an attempt, $\mathcal{F}$ extracts and consolidates task-relevant claims together with their source and conflict annotations, without requiring cross-candidate consensus. The resulting chain exposes the premises used by the executor to subsequent reasoning and verification but does not certify their correctness.

3.2.3 Evidence Verification

Risk Anchor When an execution unit submits a result, Risk Anchor receives the task T, final explanation r_f, candidate answer $\hat{y}$, and evidence chain κ. It first decomposes the candidate answer and its explanation into checkable factual claims and maps each claim to the sources recorded in

the evidence chain. A claim is marked as a high-risk evidence point when it has a conflict label, lacks a locatable supporting source, or has fewer than two independent supporting sources. The risk marking is used only to prioritize subsequent checks and does not constitute a judgment about the correctness of the claim; the absence of a high-risk marking also does not mean that the claim has been verified.

Blind Verification The second stage of Eq. (2) is performed by a tool-using ReAct verifier. In a new verification context, the verifier independently checks high-risk evidence points and task constraints without directly treating prior reasoning, source counts, or conflict labels as correctness conclusions. It retrieves evidence to check the factual premises and task requirements of the candidate answer, and returns the verdict in Eq. (2), together with diagnostic suggestions when necessary. On open-web tasks, the verifier uses web retrieval tools; on fixed-corpus tasks, it uses only that corpus and its permitted tools.

Diagnostic feedback An acceptance verdict releases the candidate answer from the verification loop. A re-execution verdict provides guidance concerning unsupported premises, conflicting evidence, or unmet constraints. If another attempt remains available, this guidance is passed together with the original task or subtask to a new execution attempt. The guidance targets suspected problems, and past memory is retained for the next round of execution.

3.3 Operational Forms

EAV-Flash EAV-Flash applies EAV to a ReAct executor that solves the entire task. Rejected candidates trigger diagnosis-guided re-execution with inherited evidence memory, up to a total-attempt budget K. An accepted candidate terminates the loop; if the budget is exhausted, the latest candidate is returned as an unverified fallback. Appendix A.2 details memory inheritance, candidate history, retry accounting, and interruption handling.

EAV-MAS EAV-MAS applies EAV at subtask submission boundaries in a DAG-based multi-agent workflow. The orchestrator schedules dependency-ready subtasks in parallel, and each subtask returns a result and evidence chain for verification. Rejected subtasks are re-executed with diagnostic feedback while retaining their active evidence memory, after which the aggregator combines the accepted outputs. Because verification is local to each subtask, it does not by itself validate cross-subtask relationships or claims newly introduced during aggregation.

4 Experiments

4.1 Experimental Setup

We evaluate EAV on GAIA val. text-only Mialon et al. (2023), xBench-DeepSearch Chen et al. (2025a), BrowseComp-ZH Zhou et al. (2025), and BrowseComp-Plus Chen et al. (2025b), spanning English and Chinese tasks in open-web and fixed-corpus settings. The comparison includes four open-source agentic-search systems that achieved state-of-the-art results on different benchmarks: Yunque Cai et al. (2026), Co-Sight Zhang et al. (2025), MiroFlow Su et al. (2026), and Flash-Searcher Qin et al. (2025). We compare them with EAV-Flash and EAV-MAS under matched backbone models wherever results are available; benchmark, method, and model details appear in Appendices B.1 and B.3.

All reproduced systems share the same retrieval interfaces on each benchmark: the three open-web benchmarks use web search and fetching, whereas BrowseComp-Plus follows the official closed-corpus setting. We report success rate for effectiveness and average token consumption, execution steps, and tool calls for efficiency. Tool configurations, grading protocols, metrics, and prompts are detailed in Appendices B.2, B.4, and D.1.

4.2 Main Results

Table 2 presents same-backbone results under the unified tool environment, while Figure 3 compares EAV-Flash with publicly reported results.

Table 2: Main results on GAIA val. text-only, xbench-ds, and BrowseComp-ZH. EAV ranks among the top two methods in eight of the nine model–benchmark settings.

Method	GAIA			xbench-ds			BrowseComp-ZH		
	DS3.2	DS4-F	GLM5.2	DS3.2	DS4-F	GLM5.2	DS3.2	DS4-F	GLM5.2
Yunque	74.76	79.61	72.82	77.00	84.00	78.00	57.09	71.63	62.28
Co-Sight	78.64	75.73	84.47	82.00	83.00	77.00	**72.66**	74.74	-
MiroFlow	71.84	**85.44**	**88.35**	81.00	86.00	82.00	65.05	**76.82**	**74.05**
Flash-Search	71.84	84.47	81.55	75.00	**88.00**	**88.00**	-	69.20	68.17
EAV-Flash	**84.47**	**91.26**	**88.35**	**84.00**	**89.00**	**88.00**	**71.63**	**76.12**	**70.93**
EAV-MAS	**83.50**	**85.44**	**86.41**	**87.00**	87.00	**85.00**	70.24	74.74	68.86
Δ *EAV-Flash*	+12.63	+5.82	0.00	+3.00	+3	+6.00	+6.58	-0.70	-3.12
Δ *EAV-MAS*	+11.66	0.00	-1.94	+6.00	+1.00	+3.00	+5.19	-2.08	-5.19

Notes. DS3.2: DeepSeek-V3.2; DS4-F: DeepSeek-V4-Flash; GLM5.2: GLM-5.2. Bold marks the top two distinct scores in each model–benchmark setting, retaining all ties. Δ rows report percentage-point changes over MiroFlow; the macro-average gains are +3.36 pp for EAV-Flash and +1.96 pp for EAV-MAS. Externally reported results marked by symbols in the source are excluded here and presented separately.

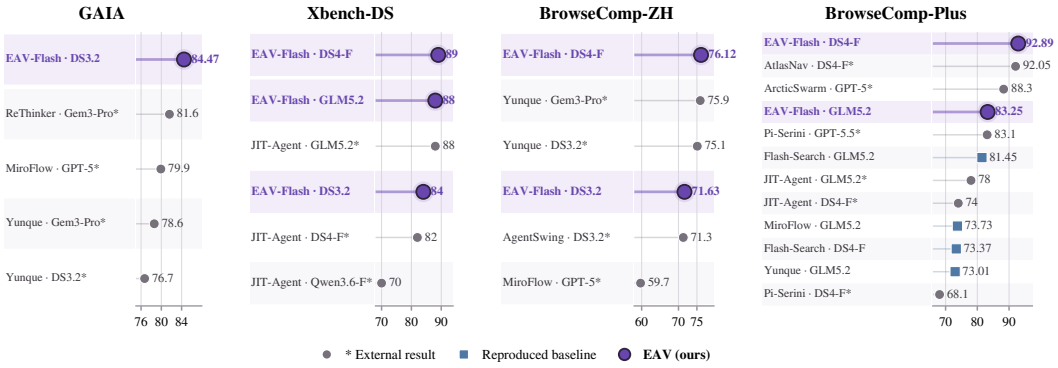

Figure 3: **Comparison with representative baselines.** Purple markers denote EAV-Flash, squares denote reproduced baselines, and * denotes externally reported results; panels use independent score ranges.

Overall, EAV-Flash ranks among the top two methods in all nine matched open-web settings and attains the best or tied-best score in five. Its macro-average success rate is 82.64, the highest among methods with complete coverage, while EAV-MAS reaches 80.91 and exceeds MiroFlow by 1.96 percentage points. EAV-MAS does not consistently outperform EAV-Flash, indicating that finer-grained subtask verification does not always justify its additional execution cost.

On xBench-DeepSearch, EAV-MAS reaches 87.00 with DeepSeek-V3.2, while EAV-Flash reaches 89.00 with DeepSeek-V4-Flash and 88.00 with GLM-5.2. The 89.00 result is the highest among the included open-source systems and exceeds the best publicly reported JIT-Agent result by 1.00 percentage point; with GLM-5.2, EAV-Flash ties Flash-Searcher for first place. These results indicate that the gains are not tied to a single backbone model.

On BrowseComp-Plus, EAV-Flash reaches 92.89 with DeepSeek-V4-Flash, ranking first among the collected public results and exceeding AtlasNav with the same backbone by 0.84 percentage points. With GLM-5.2, it scores 83.25, exceeding our reproduced Flash-Searcher result by 1.80 percentage points. Because external systems may use different retrieval and execution settings, same-backbone comparisons under the unified environment remain the primary evidence for framework-level gains.

On GAIA, EAV-Flash is best under DeepSeek-V3.2 and DeepSeek-V4-Flash and ties MiroFlow under GLM-5.2, with scores of 84.47, 91.26, and 88.35, respectively. On BrowseComp-ZH, it ranks second in all three matched-backbone comparisons, trailing the corresponding best scores by 1.03, 0.70, and 3.12 percentage points. Together, these results show consistently strong performance across English and Chinese open-web search without implying uniform dominance in every setting.

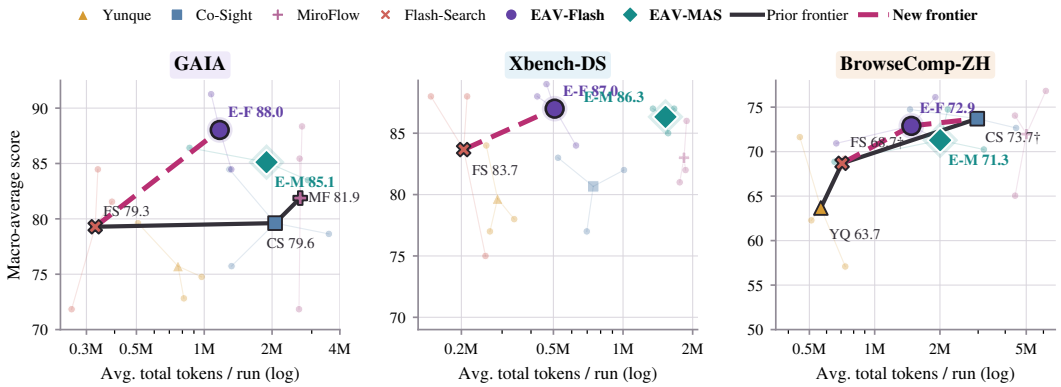

Figure 4: **Performance–token Pareto analysis.** Large and small markers denote backbone averages and individual results; solid black and dashed magenta curves show the frontiers before and after EAV, and † marks a two-model average.

Table 3: **Ablation configurations and success rates.** M1, M2, and M3 denote evidence archiving, evidence anchoring, and evidence verification, respectively; K is the maximum number of total attempts.

Method	M1	M2	M3	K	Success rate (%)
EAV (Ours)	✓	✓	✓	10	**81.4 ± 1.6**
EAV w/o M3	✓	✓	–	1	73.7 ± 1.4
EAV w/o M2	✓	–	✓	10	72.7 ± 1.0
EAV w/o M2, M3	✓	–	–	1	66.6 ± 1.2
ReAct w/ M3	–	–	✓	10	63.9 ± 1.3
ReAct	–	–	–	1	58.1 ± 0.9

Notes. ± denotes the population standard deviation over three runs. EAV w/o M3 and EAV w/o M2, M3 use the first-attempt windows of EAV and EAV w/o M2, respectively. Because M3 has not been invoked at the end of the first attempt, these windows are equivalent to the corresponding $K=1$ configurations. ReAct is an independently executed clean baseline.

4.3 Efficiency and Cost Analysis

Figure 4 compares average total-token consumption with macro-average success rate. Large markers represent averages across backbone models, while small markers represent individual-backbone results; fewer tokens and higher success rates are preferable.

Across all three open-web benchmarks, EAV-Flash advances the high-performance end of the Pareto frontier. On GAIA and xBench-DeepSearch, it reaches macro-average success rates of 88.03 and 87.00 while using 1.17M and 0.51M tokens, improving over MiroFlow by 6.15 and 4.00 percentage points with 56.1% and 72.4% fewer tokens, respectively. On BrowseComp-ZH, EAV-Flash is only 0.81 percentage points below Co-Sight while using 50.3% fewer tokens, and it exceeds MiroFlow by 0.92 percentage points with 70.5% fewer tokens.

Tool-call counts follow the same trend: EAV-Flash averages 14.9, 10.4, and 16.2 calls on GAIA, xBench-DeepSearch, and BrowseComp-ZH, compared with 52.1, 49.6, and 102.0 for MiroFlow. EAV-MAS is more costly than EAV-Flash on all three benchmarks and does not enter the new Pareto frontier, showing that finer-grained verification does not always translate into higher final accuracy.

4.4 Ablation Analysis

We evaluate the incremental effects of evidence archiving (M1), evidence anchoring (M2), and evidence verification (M3) using Qwen3.8-27B on a 400-question BrowseComp-Plus subset; full setup and configuration details are provided in Appendix C.1.

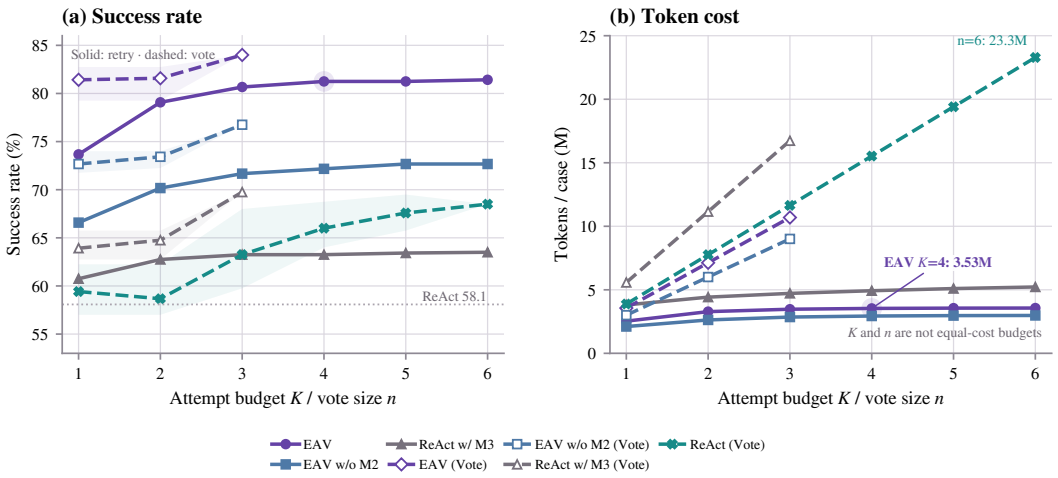

Figure 5: **Performance and cost of re-execution and majority voting.** The left panel shows success rates at different strategy scales, and the right panel shows the corresponding measured token consumption. Solid curves denote re-execution, dashed curves denote voting, and shaded regions span the minimum and maximum over different combinations of voting members. Equal values of K and n do not imply equal compute budgets, and the heterogeneous voting pools do not support strict component-level causal comparisons.

Table 3 summarizes the configurations; across the matched comparisons defined in Appendix C.1, M1, M2, and M3 yield gains of 5.8, 7.1, and 7.8 percentage points, respectively, with full EAV achieving the highest success rate of 81.4%. After Bonferroni correction, M2 and M3 remain significant under both paired tests, while M1 remains significant under the instance-level Wilcoxon test; full results are provided in Appendix C.

Beyond the component gains, Figure 5 compares EAV's verification-driven sparse retries with majority voting over independent runs. At $K{=}4$, EAV reaches 81.3% with 3.53M tokens per question, whereas ReAct voting at $n{=}6$ reaches 68.5% with 23.29M tokens. Voting over three full EAV runs further raises the success rate to 84.0%, showing that EAV and voting are complementary. Full budget and cost details are provided in Appendix C.

To examine why verification helps, Appendix Figure 7 compares how first-attempt errors are intercepted and repaired across three random seeds. Full EAV attains 63.0% interception recall, compared with 41.1% for EAV w/o M2 and 30.4% for ReAct w/ M3, while all three retain high interception precision of 95.7%, 97.6%, and 89.9%, respectively. Once an error is intercepted, full EAV and EAV w/o M2 achieve similar repair rates of 47.2% and 44.8%, compared with 29.4% for ReAct w/ M3. These results indicate that evidence anchoring contributes mainly by exposing more errors to verification rather than by changing retry effectiveness after interception.

5 Conclusion

This paper presents EAV, an evidence anchoring–verification framework that addresses evidence blindness, verification failure, and error propagation in agentic search. EAV archives tool observations, organizes executor-derived premises into an evidence chain, and checks results with an independent tool-using verifier whose diagnosis guides re-execution. Across four benchmarks, EAV-Flash and EAV-MAS achieve a favorable performance–cost trade-off, ranking first among the compared open-source systems on xBench-DeepSearch and among the collected public results on BrowseComp-Plus; ablations confirm gains from all three mechanisms. Future work includes reliable assessment of source independence, cross-subtask evidence consistency, and end-to-end evidence attribution.

REFERENCES

Brave Software, Inc. Brave search api. https://search.brave.com/, 2026. Backend of the web_search tool; accessed September 2026.

Yuxuan Cai, Xinyi Lai, Peng Yuan, Weiting Liu, Huajian Li, Mingda Li, Xinghua Wang, Shengxie Zheng, Yanchao Hao, Yuyang Yin, and Zheng Wei. Yunque deepresearch technical report, 2026. URL https://arxiv.org/abs/2601.19578.

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. URL https://arxiv.org/abs/2506.13651.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent, 2025b. URL https://arxiv.org/abs/2508.06600.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models, 2023. URL https://arxiv.org/abs/2309.11495.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate, 2023. URL https://arxiv.org/abs/2305.14325.

Zhaopeng Feng, Liangcai Su, Zhen Zhang, Xinyu Wang, Xiaotian Zhang, Xiaobin Wang, Runnan Fang, Qi Zhang, Baixuan Li, Shihao Cai, Rui Ye, Hui Chen, Jiang Yong, Joey Tianyi Zhou, Chenxiong Qian, Pengjun Xie, Bryan Hooi, Zuozhu Liu, and Jingren Zhou. Agentswing: Adaptive parallel context management routing for long-horizon web agents, 2026. URL https://arxiv.org/abs/2603.27490.

Hongyu Guo, Zhiyu Zheng, and Zhao Cao. Evidence blindness in direct corpus interaction: Persistent navigation with atlasnav, 2026. URL https://arxiv.org/abs/2608.24764.

Tz-Huan Hsu, Jheng-Hong Yang, and Jimmy Lin. Rethinking agentic search with pi-serini: Is lexical retrieval sufficient?, 2026. URL https://arxiv.org/abs/2605.10848.

Hyunho Kook, Junhyuk So, Tianyu Fu, Haizhong Zheng, and Beidi Chen. Beyond confidence: Test-time scaling for multi-turn search agents via retrieval grounding, 2026. URL https://arxiv.org/abs/2608.24024.

Yoonsang Lee, Howard Yen, Xi Ye, and Danqi Chen. Agentic aggregation for parallel scaling of long-horizon agentic tasks, 2026. URL https://arxiv.org/abs/2604.11753.

Baixuan Li, Dingchu Zhang, Jialong Wu, Wenbiao Yin, Zhengwei Tao, Yida Zhao, Liwen Zhang, Haiyang Shen, Runnan Fang, Pengjun Xie, Jingren Zhou, and Yong Jiang. Parallelmuse: Agentic parallel thinking for deep information seeking, 2025. URL https://arxiv.org/abs/2510.24698.

Jiaxi Liu, Yanzuo Jiang, Guibin Zhang, Zihan Zhang, Heng Chang, Zhenfei Yin, Qibing Ren, and Junchi Yan. Todoevolve: Learning to architect agent planning systems, 2026. URL https://arxiv.org/abs/2602.07839.

Yi Lu, Zhuofeng Li, Ping Nie, Haoxiang Zhang, Yuyu Zhang, Kai Zou, Wenhu Chen, Jimmy Lin, Dongfu Jiang, and Yu Zhang. Dr-dci: Scaling direct corpus interaction via dynamic workspace expansion, 2026. URL https://arxiv.org/abs/2606.14885.

Mendable AI. Firecrawl. https://www.firecrawl.dev/, 2026. Backend of the web_fetch tool; accessed September 2026.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants, 2023. URL https://arxiv.org/abs/2311.12983.

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. Webgpt: Browser-assisted question-answering with human feedback, 2022. URL https://arxiv.org/abs/2112.09332.

Tianrui Qin, Qianben Chen, Sinuo Wang, He Xing, King Zhu, He Zhu, Dingfeng Shi, Xinxin Liu, Ge Zhang, Jiaheng Liu, Yuchen Eleanor Jiang, Xitong Gao, and Wangchunshu Zhou. Flash-searcher: Fast and effective web agents via dag-based parallel execution, 2025. URL https://arxiv.org/abs/2509.25301.

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

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Shiqian Su, Sen Xing, Xuan Dong, Muyan Zhong, Bin Wang, Xizhou Zhu, Yuntao Chen, Wenhai Wang, Yue Deng, Pengxiang Zhu, Ziyuan Liu, Tiantong Li, Jiaheng Yu, Zhe Chen, Lidong Bing, and Jifeng Dai. Miroflow: Towards high-performance and robust open-source agent framework for general deep research tasks, 2026. URL https://arxiv.org/abs/2602.22808.

Zhentao Tang, Yuqi Cui, Shixiong Kai, Wenqian Zhao, Ke Ye, Xing Li, Anxin Tian, Zehua Pei, Hui-Ling Zhen, Shoubo Hu, Xiaoguang Li, Yunhe Wang, and Mingxuan Yuan. Rethinker: Scientific reasoning by rethinking with guided reflection and confidence control, 2026. URL https://arxiv.org/abs/2602.04496.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Soyoung Yoon, Boyi Liu, Yite Wang, Ruofan Wu, Canwen Xu, Nikki Lijing Kuang, Seung won Hwang, Yuxiong He, and Zhewei Yao. Arcticswarm: Deferring early consensus in long-horizon multi-agent research, 2026. URL https://arxiv.org/abs/2609.01870.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. Assistantbench: Can web agents solve realistic and time-consuming tasks?, 2024. URL https://arxiv.org/abs/2407.15711.

Guibin Zhang, Leo Lu, Fangzhou Xie, Kang Zhu, Junhao Wang, Zhifei Xie, Zhaochen Yu, Zihang Liu, Zhongxiang Sun, Qiankun Li, Yue Liao, Heng Chang, Xiaobin Hu, Qibing Ren, Wangchunshu Zhou, Chuanrui Hu, Yafeng Deng, and Shuicheng Yan. Jit-agent: Scaling harness intelligence via just-in-time harness evolution, 2026a. URL https://arxiv.org/abs/2608.25593.

Hongwei Zhang, Ji Lu, Shiqing Jiang, Chenxiang Zhu, Li Xie, Chen Zhong, Haoran Chen, Yurui Zhu, Yongsheng Du, Yanqin Gao, Lingjun Huang, Baoli Wang, Fang Tan, and Peng Zou. Co-sight: Enhancing llm-based agents via conflict-aware meta-verification and trustworthy reasoning with structured facts, 2025. URL https://arxiv.org/abs/2510.21557.

Wentao Zhang, Liang Zeng, Yuzhen Xiao, Yongcong Li, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. Agentorchestra: Orchestrating multi-agent intelligence with the tool-environment-agent(tea) protocol, 2026b. URL https://arxiv.org/abs/2506.12508.

James Xu Zhao, Hui Chen, Bryan Hooi, and See-Kiong Ng. Fineverify: Scaling test-time compute with fine-grained self-verification for agentic search, 2026. URL https://arxiv.org/abs/2606.00660.

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

He Zhu, Tianrui Qin, King Zhu, Heyuan Huang, Yeyi Guan, Jinxiang Xia, Yi Yao, Hanhao Li, Ningning Wang, Pai Liu, Tianhao Peng, Xin Gui, Xiaowan Li, Yuhui Liu, Yuchen Eleanor Jiang, Jun Wang, Changwang Zhang, Xiangru Tang, Ge Zhang, Jian Yang, Minghao Liu, Xitong Gao, Jiaheng Liu, and Wangchunshu Zhou. Oagents: An empirical study of building effective agents, 2025. URL https://arxiv.org/abs/2506.15741.

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents, 2024. URL https://arxiv.org/abs/2410.10934.

A Additional Method Details

A.1 Memory and Anchoring Details

Context lifecycle A new tool observation is available to the executor's next reasoning step. After that step, the raw output is offloaded from the active context and retained in the archive together with its summary and importance score. The evidence chain is appended to subsequent reasoning inputs. Archive records, the active context, and the rendered evidence chain are distinct representations: retaining a record on disk does not mean that its raw text remains in every model call.

Joint extraction and consolidation As shown in Eq. (4), the anchoring call operates on the reasoning produced after a tool result and on the current evidence chain. It may add claims, merge matching claims and their sources, or mark disagreements. Serialized entries implement $\mathcal{E} = \langle k, v, S, l \rangle$ through `key`, `value`, `sources`, and `conflict`, together with the execution metadata `step`, `tool`, and `timestamp`. For example, an entry associates `key` = "Chigozie Obioma's birthplace" with `value` = "Akure, Nigeria" and `sources` = ["docid: 8674"]. This illustrates the representation only and is not an independently verified factual label. When entries under the same topic contain inconsistent values, `conflict` is set, corresponding to $l = 1$; otherwise, $l = 0$. The label marks an unresolved disagreement for subsequent reasoning or verification and neither selects the correct value nor constitutes a verification verdict. The chain cannot recover facts that the executor never read or incorporated into its reasoning; such omissions require further retrieval or verification. If there is no useful change, the evidence chain remains unchanged.

Source references and archive identifiers The `sources` field of the evidence chain identifies cited material, whereas an archive identifier identifies a tool interaction. Repeated retrieval of the same document may produce multiple interaction records. A source reference therefore does not uniquely identify an archived observation. The semantic update in Eq. (4) does not specify how step, tool, or source metadata are routed. In particular, if a source identifier is omitted from both the reasoning and the existing evidence chain, it cannot be recovered from these inputs alone.

Scope of the current specification The lifecycle above specifies when observations are used and offloaded, and when the evidence chain is supplied to reasoning and verification. It does not specify the retention policy for historical reasoning and summaries, how importance scores are used in context selection, or which roles may retrieve archived observations. A reproducible implementation still needs to record these settings, along with source-metadata routing and the verification prompt. If internal archive access is enabled, it is distinct from external corpus retrieval.

A.2 Execution and Retry Details

This section specifies the operational details summarized in Section 3.3, including memory initialization and inheritance, candidate history, total-attempt accounting, fallback behavior, and interruption handling.

Cross-attempt memory and candidate history The first attempt initializes an empty active archive and evidence chain, while each subsequent attempt inherits the active archive and evidence chain accumulated during the preceding attempt. Diagnostic guidance is provided together with the original task and the existing evidence memory, whereas completed rationales and candidate answers are stored separately in the history H. In Algorithm 1, g is replaced by the latest verification output; the pseudocode does not separately maintain a cumulative history of diagnostic feedback.

Normal execution flow Algorithm 1 summarizes the normal EAV-Flash loop. The total-attempt cap K includes the initial attempt, so at most K attempts are executed. The inner loop describes a serialized tool–reasoning sequence at the executor-event level. Capturing the reasoning that uses an observation does not require an additional model call. Any tool action or final delivery produced in the same executor turn remains part of the normal execution state. The algorithm does not prescribe batching multiple tool returns into a single reasoning event.

Algorithm 1 EAV-Flash: Normal execution and bounded re-execution.

Require: Task T, tools $\mathcal{U}$, total-attempt cap K
Ensure: Returned candidate answer $\hat{y}$ (accepted or fallback)

$H \leftarrow [\,]$, $g \leftarrow \varnothing$, $(\mathcal{R}, \kappa) \leftarrow (\varnothing, \varnothing)$
for $k = 1, \ldots, K$ **do**
  Initialize the executor with $(T, g, \mathcal{R}, \kappa)$
  while the executor has not delivered a result **do**
    Advance normal execution and append κ before reasoning calls
    if a tool call returns observation o_i **then**
      Capture the next normal reasoning step r_{i+1} that uses o_i
      $\kappa \leftarrow \mathcal{F}(r_{i+1}, \kappa)$ ▷ Joint extraction and consolidation
      Offload o_i to $\mathcal{R}$ together with its identifier, summary, and importance score
    end if
  end while
  $(r_f, \hat{y}^{(k)}) \leftarrow$ the delivered rationale and candidate answer
  Append $(r_f, \hat{y}^{(k)})$ to H
  $(\nu, g) \leftarrow \mathcal{V}(T, r_f, \hat{y}^{(k)}, \kappa)$
  if $\nu = \alpha$ **then**
    return $\hat{y}^{(k)}$ ▷ Accepted by the verifier
  end if
end for
return the latest candidate answer in H ▷ Fallback; not necessarily verified

Fallback and interruption When the total attempt budget is exhausted, the latest available candidate answer is returned. If execution terminates unexpectedly after a candidate answer has been saved, the saved candidate answer can likewise serve as the fallback. This policy preserves a usable answer, but it does not establish that accuracy improves monotonically across subsequent attempts or that the returned answer has passed verification. Algorithm 1 describes normally completed attempts; exception handling and termination when no candidate answer exists fall outside this normal flow.

Subtask orchestration EAV-MAS applies the same execution and verification components to each subtask in a dependency DAG. Scheduling and aggregation operate at the task level, whereas verification provides a judgment and diagnostic feedback for a delivered subtask. This separation makes explicit where checks occur while avoiding an equivalence between execution completion and factual correctness. The task-level fallback in Algorithm 1 does not specify how the MAS scheduler should handle a subtask result whose retry budget has been exhausted.

B Experimental Details

B.1 Benchmarks

Table 4 summarizes the number of questions, language, retrieval environment, and scoring scheme for the four benchmarks.

Table 4: Overview of the evaluation benchmarks.

Benchmark	Questions	Language	Retrieval environment	Scoring
GAIA validation set (text-only)	103	English	Open web	LLM judge
BrowseComp-ZH	289	Chinese	Open web	LLM judge
xBench-DeepSearch	100	Chinese	Open web	LLM judge
BrowseComp-Plus	830	English	Fixed corpus	LLM judge

GAIA GAIA contains several mutually incompatible evaluation subsets (103-question text-only subset / 165-question full subset / 300-question test set); we use the text-only subset of the validation set (103 questions), which removes tasks requiring multimodal perception or local file opera-

tions while retaining text-only general-assistant tasks that combine web search with multi-step tool use Mialon et al. (2023). This evaluation setup is consistent with MiroFlow Su et al. (2026) and Yunque Cai et al. (2026).

BrowseComp-ZH We use all 289 questions from the April 2025 revision Zhou et al. (2025). The questions, retrieval chains, and browsing steps were authored directly in Chinese rather than translated, preserving the retrieval challenges inherent to the Chinese web, such as information scattered across fragmented platforms and implicitly phrased key facts.

xBench-DeepSearch We use all 100 search tasks from version 2505, which are posed in Chinese Chen et al. (2025a).

BrowseComp-Plus We use all 830 questions in the official test set; answers are restricted to a fixed corpus containing 100,195 documents (approximately 530M tokens) Chen et al. (2025b). Corpus retrieval uses two tools: `search`(query) returns the IDs, scores, and snippets of the top-k documents, and `get_document`(docid) returns the full text; Internet access is not allowed.

Retrieval service implementation We implement the corpus retrieval service using BM25 with k_1=1.2 and b=0.75. Document identifiers use the official qrels namespace so that retrieved documents can be matched against relevance judgments. Matching identifiers and scoring parameters do not imply identical rankings or evidence recall: tokenization, index construction, tie handling, and retrieval settings can all affect the returned results.

B.2 Tool Configuration

On the open-web benchmarks, the agent uses `web_search` and `web_fetch`: `web_search`(query) returns web search results and is powered by the Brave Search API Brave Software, Inc. (2026); `web_fetch`(url) retrieves and parses the main content of a web page and is powered by Firecrawl Mendable AI (2026). EAV additionally provides `read_memory` for retrieving archived raw observations on demand. BrowseComp-Plus follows the official protocol and uses the two closed-corpus retrieval tools, `search` and `get_document`; the retrieval service implementation is described in Section B.1.

B.3 Comparison Methods and Backbone Models

We reproduce four representative open-source agentic-search systems (Yunque Cai et al. (2026), Co-Sight Zhang et al. (2025), MiroFlow Su et al. (2026), and Flash-Searcher Qin et al. (2025)) and compare them with EAV-Flash and EAV-MAS. On GAIA, xBench-DeepSearch, and BrowseComp-ZH, where results are available, we evaluate each method with DeepSeek-V3.2 (DS3.2), DeepSeek-V4-Flash (DS4-F), and GLM-5.2 (GLM5.2). These matched-backbone comparisons isolate the effect of the agent framework as far as possible. Figure 3 additionally includes publicly reported results from ReThinker Tang et al. (2026), AgentSwing Feng et al. (2026), JIT-Agent Zhang et al. (2026a), Pi-Serini Hsu et al. (2026), AtlasNav Guo et al. (2026), and ArcticSwarm Yoon et al. (2026). Because these external results may use different backbone models, retrieval backends, execution budgets, or grading protocols, we use them only to contextualize the public performance frontier and exclude them from paired improvement calculations.

B.4 Evaluation Protocol and Metrics

We report success rate, defined as the fraction of questions for which the single final answer returned by the system is judged correct (Pass@1). The judge model makes a binary decision using the standardized prompt in Appendix D.1. GAIA, xBench-DeepSearch, and BrowseComp-ZH use Qwen3.8-Max to avoid evaluation by a model from the same family as the tested backbones, whereas BrowseComp-Plus follows the official grading template and uses Qwen3-32B. For efficiency, we record the average input, output, and total token counts per question, together with the average numbers of execution steps and tool calls. Figure 4 measures the performance–cost trade-off using average total tokens and the macro-average success rate across backbone models.

Table 5: **Paired significance tests for mechanism increments.** Gains are measured in percentage points, and the Bonferroni-corrected significance threshold is 0.0167.

Mechanism increment	Gain (pp)	Median exact McNemar p-value	Case-level Wilcoxon p-value
+M1: Evidence archiving	5.8	0.028	0.0004
+M2: Evidence anchoring	7.1	0.0056	$< 10^{-5}$
+M3: Evidence verification	7.8	$< 10^{-5}$	$< 10^{-5}$

C Additional Ablation Studies

C.1 Ablation Setup and Increment Construction

We use Qwen3.8-27B on a 400-question diagnostic subset of BrowseComp-Plus, with all configurations sharing the same closed corpus and retrieval tools and each configuration evaluated with three random seeds. M1 archives retrieval results for later access, M2 organizes executor-used premises into an evidence chain, and M3 verifies submitted answers and triggers diagnosis-guided re-execution after rejection. Table 3 reports the six evaluated configurations, while the mechanism increments follow the sequential activation of M1, M2, and M3. The M1 increment compares the first-attempt result of ReAct w/ M3, before M3 can act, with EAV w/o M2, M3, increasing success from 60.8% to 66.6%. The M2 increment compares EAV w/o M2, M3 with EAV w/o M3, increasing success from 66.6% to 73.7%. The M3 increment compares the first-attempt result of full EAV with its verified final result, increasing success from 73.7% to 81.4%. The 60.8% endpoint used for the matched M1 increment is therefore distinct from the independently executed clean ReAct baseline of 58.1% shown in Table 3.

C.2 Significance Tests for Mechanism Increments

Table 5 reports paired significance tests for the M1, M2, and M3 increments discussed in Section 4.4. The McNemar test reports the median exact p-value across matched seed pairs, while the Wilcoxon test is computed from case-level paired outcomes.

At the corrected threshold, M2 and M3 remain significant under both tests. M1 is significant under the case-level Wilcoxon test, whereas the median seed-pair McNemar result is only marginal. All three increments are directionally consistent, but M1 should be interpreted as a configuration-level increment because it also activates the associated context-management path.

C.3 Attempt-Budget Analysis

Figure 6 presents the accuracy and measured token cost of the three verifier-enabled configurations as the total attempt budget K increases. At $K{=}1$, full EAV, EAV w/o M2, and ReAct w/ M3 achieve 73.7%, 66.6%, and 60.8%, respectively. Their success rates increase to 79.1%, 70.2%, and 62.8% at $K{=}2$, and to 81.3%, 72.2%, and 63.3% at $K{=}4$. By $K{=}10$, the corresponding rates are only 81.4%, 72.7%, and 63.9%, showing that most accuracy gains occur within the first few attempts. For full EAV, the second attempt realizes approximately 69% of the total gain available by $K{=}10$, while $K{=}4$ realizes approximately 98%.

The token curves explain why a larger maximum budget does not imply proportional cost growth under verification-driven retry. Full EAV consumes 2.53M tokens per case at $K{=}1$, 3.28M at $K{=}2$, 3.53M at $K{=}4$, and 3.56M at $K{=}10$ because accepted cases stop early. EAV w/o M2 similarly grows from 2.11M to 3.00M tokens, whereas ReAct w/ M3 grows from 3.81M to 5.58M tokens while gaining only 3.2 percentage points. The continued cost increase of ReAct w/ M3 after $K{=}4$ produces little additional accuracy, indicating that later retries are less effective without archived and anchored evidence. Because the configurations have different per-attempt costs, the curves compare realized policies rather than equal-compute budgets.

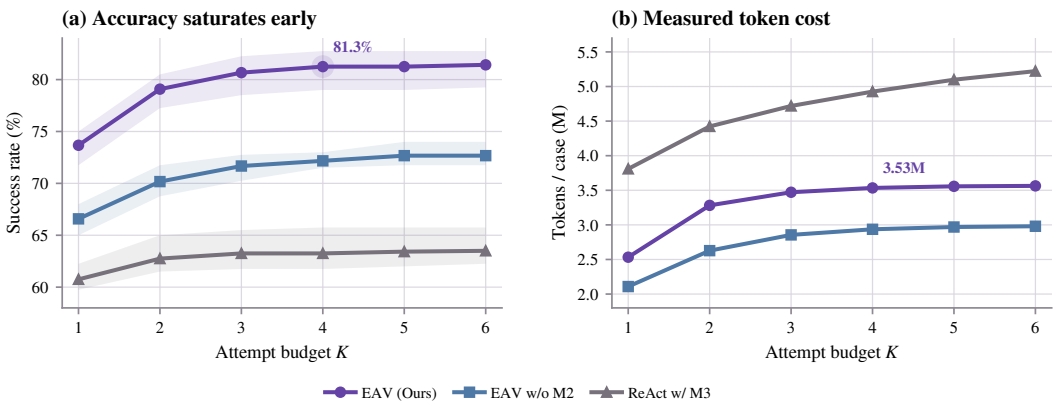

Figure 6: **Attempt-budget sweep.** Shaded regions indicate the minimum and maximum across three random seeds, and the curves show the mean. Most of the gains from full EAV occur within the first 2–4 attempts; by contrast, later attempts by ReAct w/ M3 continue to increase cost while providing almost no further improvement in success rate.

C.4 Sparse Re-execution and Majority Voting

EAV initiates another execution only after the verifier rejects a candidate answer, rather than generating a fixed number of independent samples in advance. Consequently, increasing the maximum attempt budget from $K{=}1$ to $K{=}4$ raises the success rate from 73.7% to 81.3%, while the measured cost reaches only 3.53M tokens per question because accepted cases terminate early. By contrast, majority voting requires multiple complete independent runs, so its cost grows approximately linearly with the vote size n. At $n{=}6$, ReAct voting consumes 23.29M tokens per question and reaches 68.5%; relative to EAV at $K{=}4$, this is approximately $6.6\times$ the token cost and 12.8 percentage points lower success. The two strategies are nevertheless complementary: voting over three full EAV runs raises success from 81.4% for one run to 84.0% at 10.69M tokens per question.

C.5 Verification Diagnostics

Figure 7 separates verification quality into interception recall, interception precision, and eventual repair. Interception recall is the fraction of incorrect first-attempt answers rejected by the verifier, interception precision is the fraction of rejected answers that are actually incorrect, and eventual repair is the fraction of intercepted errors corrected by later attempts. Pooling the three seeds, full EAV intercepts 199 of 316 first-attempt errors, yielding 63.0% recall. EAV w/o M2 intercepts 165 of 401 errors and ReAct w/ M3 intercepts 143 of 471 errors, corresponding to recalls of 41.1% and 30.4%. Their interception precisions are 95.7%, 97.6%, and 89.9%, respectively, so the higher recall of full EAV does not arise from indiscriminately rejecting candidate answers. Among intercepted errors, full EAV repairs 94 of 199, EAV w/o M2 repairs 74 of 165, and ReAct w/ M3 repairs 42 of 143, giving repair rates of 47.2%, 44.8%, and 29.4%. Relative to all first-attempt errors, these repairs correspond to recovery rates of 29.7%, 18.5%, and 8.9%. The similar post-interception repair rates of full EAV and EAV w/o M2 indicate that evidence anchoring contributes mainly by exposing more errors to verification rather than by changing the success rate of a retry once an error has been intercepted. The Wilson intervals in the left panel quantify uncertainty in the pooled proportions, while the right-panel funnel visualizes absolute error attrition after normalizing each method's initial errors to 100%. These comparisons are descriptive rather than equal-difficulty causal estimates because the configurations begin with different sets and numbers of first-attempt errors.

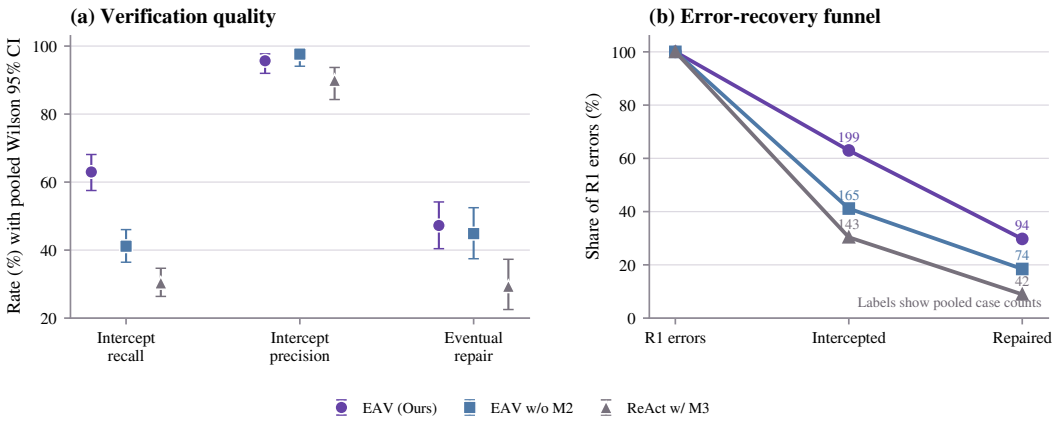

Figure 7: **Error interception and eventual repair.** The left panel reports pooled estimates with Wilson 95% confidence intervals; the right panel normalizes the first-attempt errors of each method to 100% and shows the fractions intercepted and eventually repaired. Labels give case counts pooled across three random seeds.

D PROMPTS

D.1 LLM-AS-JUDGE PROMPT

The following prompt is used to judge whether a response matches the reference answer.

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

[question]: {question}

[response]: {response}

[correct_answer]: {correct_answer}

Your judgement must end with a line exactly of the form:
correct: yes
or
correct: no
```

D.2 EAV FRAMEWORK PROMPTS

D.2.1 SYSTEM PROMPT

For the BrowseComp-Plus closed-corpus setting, the executor uses the following system prompt. The retrieval tools are the executor's only information channel.

```
You are a helpful AI assistant solving BrowseComp-Plus benchmark questions -- hard,
↪  multi-hop, information-seeking deep research questions in English, answered over a
↪  **fixed, closed corpus** of ~100K web documents. There is **NO internet access** and
↪  no file attachments: the retrieval tools below are your only information channel.

Every question in this benchmark is guaranteed to have a definite answer. Never refuse
↪  and never tell the user to look it up themselves. If, after exhaustive retrieval, the
↪  evidence does not fully pin it down, you MUST still commit to your single
↪  best-supported candidate as the Exact Answer (with a lower Confidence). Refusal
↪  phrasing -- "unable to determine", "cannot confirm", "insufficient information", "not
↪  mentioned in the corpus", "no definitive answer" -- always scores zero.

## Your Retrieval Tools (the only information channel)

- `bcp_search(query)` -- keyword (BM25) search over the closed corpus; returns a JSON
↪  list.
- `bcp_get_document(docid)` -- returns the full text of one corpus document (very long
↪  documents are truncated with a marker).
- `read_memory(step_id)` -- re-reads the full content of a previous tool result stored in
↪  step memory.
```

```
## IMPORTANT -- Final Answer Format

When you have gathered all necessary information and are ready to answer:
- Stop using tools.
- `sub_task_answer` (the Exact Answer) must always contain a concrete best-guess
↪  candidate -- refusal wording is forbidden anywhere in the final answer and always
↪  scores zero; when uncertain, still state the most likely candidate and lower the
↪  Confidence.
- Respond in this JSON format only:

{
    "reasoning": " [1] First, I searched... [2] Then, document 5412 confirmed... [3]
    ↪  Based on these findings, I concluded... (Cite the docids you relied on)",
    "sub_task_answer": "The exact answer of this subtask ONLY (one succinct entity,
    ↪  number or short phrase -- the Exact Answer content, no full sentences)",
    "raw_question_answer": "The complete response to the raw question, strictly following
    ↪  the format required at the end of the question (Explanation with [docid]
    ↪  citations / Exact Answer / Confidence)"
}
```

D.2.2 VERIFY PROMPT

Claims marked as unsupported or uncorroborated are independently rechecked by a verifier that receives the claim and disambiguation context, but not the original trajectory or reasoning.

```
Independently verify the following factual claim. Do NOT assume it is correct;
↪  investigate from scratch.

Claim to verify: {claim}

Context (for disambiguation only):
- Original question: {original_question}
- Subtask goal: {task_description}

Requirements:
1. Find at least 2 INDEPENDENT sources. Sources are NOT independent if they share a
↪  publisher, cite each other, republish the same wire copy, or draw on the same
↪  upstream database - such copies count as ONE source.
2. Prefer authoritative sources: official/primary (Tier 1) > well-cited encyclopedia /
↪  major wire services (Tier 2) > general media (Tier 3) > blogs/forums (Tier 4-5).
3. For numeric claims, normalize units, currency and time basis before comparing; do
↪  conversions with code execution, never mentally.
4. If sources conflict, the higher tier wins, unless it is clearly stale - then the
↪  fresher authoritative source wins.
{JSON_OUTPUT_INSTRUCTION}
Respond with ONLY this JSON object:
{"verdict": "confirmed|refuted|uncertain", "correct_value": "<value best supported by the
↪  evidence>", "evidence": [{"url": "<source url>", "span": "<supporting text>", "tier":
↪  1}], "new_evidence": "<new evidence you found (URL/file/computation); empty if
↪  none>", "rationale": "<one sentence: which rule decided - tier / recency / scope /
↪  independence>"}

Verdict rules:
- "confirmed": the claim matches the best-supported value (>=2 independent sources, or a
↪  single unambiguous Tier-1 source).
- "refuted": the evidence contradicts the claim; put the best-supported value in
↪  correct_value.
- "uncertain": evidence is insufficient or the conflict cannot be reconciled.
```

If the first investigation round does not produce a valid verdict, the following closing section is appended for a final constrained round.

```
A previous investigation round produced the following inconclusive output (reference
↪  only, you may correct it):
{phase1_output}

This is your FINAL round with a strictly limited step budget. Do NOT open new broad
↪  investigation directions. Go directly to the most authoritative source for the claim,
↪  then output the final verdict JSON (verdict / correct_value / evidence / new_evidence
↪  / rationale).
```

D.2.3 TOOL SUMMARIZATION PROMPT

The following prompt filters a tool result while preserving information relevant to the task.

```
You are a content filter. Extract relevant info from tool output.
```

```
Task Goal: {task_goal}
Previous AI Reasoning:
{previous_ai_message_content}

Tool Output:
{content}

Instructions:
1. Remove garbled text/binary data.
2. Extract info relevant to the Task Goal.
3. Preserve key details (numbers, names, URLs).
4. Output JSON: {"filtered_content": "...", "garbled_text_detected": false, "confidence":
↪  "high"}
```

D.2.4 EVIDENCE ANCHOR PROMPT

The following prompt extracts one reusable, confirmed fact from a reasoning step and its associated tool result.

```
You are an information distillation assistant. Below are a task goal, the intent of one
↪  reasoning step, and the result of one tool call. First judge: does the tool result
↪  contain a key conclusion that helps complete the task, is reusable, and has been
↪  confirmed?

If there is no such key conclusion (e.g., purely procedural output, internal reasoning,
↪  empty results, mere restatement of already-known information, intermediate steps of a
↪  calculation or command, skill or meta-operation records), output only:
{"skip": true}

If there is one, distill the single most important confirmed fact and output it as JSON:
{"key": "fact topic (short, e.g., 'film release year')", "value": "conclusion value
↪  (short, no more than 50 characters)", "source": "verifiable source"}

Requirements:
1. key is the fact topic and value is the conclusion value; keep both as short as
↪  possible;
2. source must be a verifiable source: url / file path / website name; fill in
↪  "[unverified]" if no source can be found in the result;
3. output only this one JSON (either skip or the triplet -- nothing else).

[Task Goal]
{task_goal}

[Reasoning Intent of This Step]
{reasoning}

[Tool] {tool_name}

[Tool Result] (may be very long; only the beginning is included)
{result}
```

The extracted entry is injected at the end of the message list in the following format.

```
<task_scratchpad>
[{step} | {tool}] <{key}, {value}, {sources}> (conflicts with Nxxx conclusion / updated
↪  from Nxxx)
</task_scratchpad>
```

The following prompt determines the relation between a new entry and an existing entry on the same topic.

```
Judge the relationship between the NEW and OLD pieces of information below; output only
↪  one English word and nothing else:
- EQUIVALENT: same topic and consistent conclusion (including differences only in
↪  wording/units/format, e.g., "1998" vs. "1998 (year)")
- CONFLICT: same topic but mutually contradictory conclusions
- UPDATE: same topic; the new information is a correction / update / more precise version
↪  of the old
- COMPLEMENTARY: related topics describing different aspects that do not contradict each
↪  other
- NONE: completely different topics

[Task Goal]
{task_goal}

[OLD] key={old_key} | value={old_value}
[NEW] key={new_key} | value={new_value}
```

E CASE STUDY

E.1 EAV-FLASH

Task The query asks for a Chinese pop song that satisfies four linked clues: it is performed by a well-known male singer, appears on the singer's self-titled album, was arranged by a former band keyboardist, and contains a food name in its backing-vocal lyrics whose final Chinese character matches the final character of the song title.

Initial failure The first execution returned *Yellow Maple Leaf* by Xue Zhiqian. The evidence chain supported that the song appears on Xue Zhiqian's self-titled album and that its arranger, Peng Cheng, had worked as a keyboardist. However, the candidate depended on the claim that the backing-vocal lyrics contain "tea leaves." The verifier checked multiple independent lyric sources and found neither this phrase nor another food name, so it rejected the otherwise plausible answer.

Diagnosis-guided re-execution The verifier explicitly identified the unsupported lyric clue and instructed the next executor to reconsider the song rather than reuse the inferred "tea leaves" match. With this diagnosis and the inherited evidence memory, the second execution returned *Istanbul* by Jay Chou. The subsequent check confirmed that the song belongs to Jay Chou's self-titled album *Jay*, that arranger Hong Jingyao had served as a keyboardist in EZ Band, and that the backing-vocal lyrics contain the Chinese word for "hamburger." Both that food name and the Chinese title of *Istanbul* end with the same character, pronounced *bao*, satisfying the remaining wordplay constraint, and the verifier accepted the answer.

Takeaway This case illustrates that the evidence chain can preserve supported partial matches while exposing a single unsupported premise that invalidates the conclusion. Rather than repeating the same search or voting over independently generated answers, diagnostic feedback redirects re-execution toward the failed constraint and enables a corrected answer on the next attempt.