# SEARCHOS: TOWARDS ROBUST OPEN-DOMAIN INFORMATION-SEEKING AGENT COLLABORATION

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

## ABSTRACT

Recent advances in tool-integrated large language models have made web search a core capability of information-seeking agents. However, as interaction histories grow, agents increasingly struggle to track task progress. When search attempts fail to yield useful evidence, current single- and multi-agent systems can become trapped in repetitive loops, wasting search budgets and ultimately compromising the quality and completeness of the final output. We introduce **SearchOS**, a system-level multi-agent framework that turns fragile, implicit search progress into explicit, persistent, and shared state. **SearchOS** formulates open-domain information seeking as *relational schema completion with grounded citations*, where search agents discover entities, record attributes across linked tables, and anchor each value to source evidence. Inspired by this paradigm, we design three core modules of **SearchOS**: (1) **Search-Oriented Context Management** (SOCM), which externalizes and shares evolving search state across sub-agents and supports pipeline-parallel scheduling to improve utilization and throughput; (2) a **Search Tool Middleware Harness**, which intercepts model and tool interactions to record grounded evidence and respond to stalls or budget exhaustion; and (3) a **Hierarchical Search Skill System**, which provides strategy and access skills to guide task-level search and source-specific retrieval. Across multiple benchmarks, **SearchOS** achieves the best scores on all reported metrics among the evaluated single- and multi-agent baselines, paving the way toward robust information-seeking collaboration.

## 1 INTRODUCTION

Recent advances in tool-integrated large language models (LLMs) have made web search a core capability of information-seeking agents (Yao et al., 2023; Nakano et al., 2021; Jin et al., 2025; Li et al., 2026b). As the field moves toward broad, open-domain and long-horizon tasks (Wong et al., 2025; Zhu et al., 2026), an agent must gather large collections of facts, reconcile evidence across sources, and produce a structured answer with verifiable citations.

Stronger search capability alone, however, does not make this process reliable. As interaction histories grow, agents can easily lose track of what has been solved and what remains not. Simply adding more agents does not eliminate these problems: because search progress is neither consistently maintained nor traceable across agents, parallel workers may duplicate effort or disagree on the intended fields, leading to a poor-quality final answer.

These failures arise because agents treat plans, progress, evidence, and failures as transient conversation content. **Search state, however, should be maintained by the system rather than inferred by LLMs themselves repeatedly from interaction history**. A robust search system should track completed search subtasks and collected evidence, link each finding to its source, coordinate workers around unresolved questions, and intervene when execution fails. It should also retain effective search and access patterns across sessions instead of rediscovering them for every task.

Based on this principle, we present **SearchOS**, a multi-agent framework for long-horizon information seeking that maintains shared search state and coordinates agents around unresolved information gaps. Although flat tables make search state explicit (Lan et al., 2026), flattening multiple related entity types can duplicate facts or group multiple entities within cells, complicating coverage tracking. Inspired by modern relational database design (Codd, 1970; Bernstein, 1976), we formulate an



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.

information-seeking task as *relational schema completion with grounded citations*. A relational search schema organizes entities into one or more tables according to task structure, with primary keys identifying entities and foreign keys linking related tables. This formulation turns an open-ended request into a concrete search target and makes search progress measurable.

Building on this formulation, we design three core modules to enable **SearchOS** to efficiently schedule and coordinate search agents. First, **Search-Oriented Context Management** (SOCM) externalizes execution state and shares them across all agents. An orchestrator decomposes the schema into coverage-oriented tasks and coordinates specialized agents through this shared state. Rather than dispatching workers in synchronized batches, **SearchOS** uses pipeline-parallel scheduling to reduce straggler-induced idle time of multi-agents collaboration and increases search throughput.

Second, robust execution requires system-level controls that do not rely solely on instructions in prompts or interaction context. We introduce a **Search Tool Middleware Harness** that implements OS-like control logic by code execution. It intercepts model and tool interactions to inject relevant state, extract and anchor evidence, enforce budgets, and detect repeated or stalled trajectories. Agents can focus on local search decisions while the harness maintains global execution invariants.

Third, we design a **Hierarchical Search Skill System** that organizes complementary skills for relational schema design, search strategy, and site-specific access, including techniques for reaching and extracting information from websites that cannot be queried through APIs. Experiments across multiple benchmarks show that **SearchOS** achieves the best or tied-best scores on all reported metrics among the evaluated single- and multi-agent baselines. These results support its effectiveness in both structured information seeking and complex, multi-step autonomous navigation. Our main contributions are as follows:

1. **Relational Search Formulation.** We formulate open-domain information seeking as relational schema completion with grounded citations, providing a unified and verifiable objective for entity discovery, attribute completion, and evidence attribution.
2. **Stateful Search Architecture.** We design Search-Oriented Context Management (SOCM) and a Search Tool Middleware Harness of **SearchOS** to coordinate agents through shared state and enforce robust search execution.
3. **Reusable Search Knowledge.** We develop a Hierarchical Search Skill System that organizes orchestration, search-strategy, and source-access knowledge, and converts execution experience into validated access skills for reuse across tasks.

## 2 Problem Formulation

We formulate open-domain information seeking as **relational schema completion with grounded citations**. Following relational database design, we use primary keys to identify entities, foreign keys to express cross-table relations, and normalized tables to reduce redundant or inconsistent fact representations (Codd, 1970; Bernstein, 1976).

Given a natural-language request $q$, the system constructs a relational search schema

$$\mathcal{S} = \big(\{T_m\}_{m=1}^{M}, \mathcal{R}\big), \qquad T_m = \big(\mathcal{A}_m, \mathcal{P}_m\big), \tag{1}$$

where $T_m$ defines a table with attributes $\mathcal{A}_m$ and primary-key attributes $\mathcal{P}_m$, and $\mathcal{R}$ contains foreign-key relations between tables. The task is denoted by $\mathcal{T} = (q, \mathcal{S})$.

For each table, the system must discover a set of entities $\mathcal{E}_m = \{e_{m,i}\}_{i=1}^{N_m}$ and populate a value matrix $\mathbf{Y}_m \in \mathcal{V}^{N_m \times |\mathcal{A}_m|}$, and $\mathcal{V}$ is the space of attribute values. Unlike conventional table completion, every populated value must be grounded in source evidence. We therefore maintain a citation matrix $\mathbf{C}_m$ of the same shape, where each entry links a value to a source URL and an anchored excerpt:

$$\mathcal{O} = \big\{\big(\mathcal{E}_m, \mathbf{Y}_m, \mathbf{C}_m\big)\big\}_{m=1}^{M}. \tag{2}$$

The relational structure provides an explicit target for search, while the citation matrix makes each factual value independently verifiable. The completed schema is finally synthesized into a natural-language report with fine-grained inline citations.

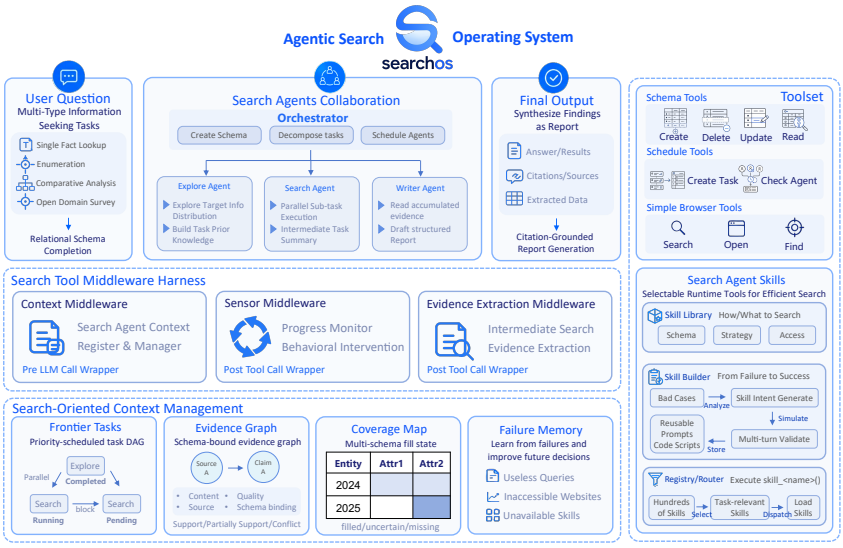



Figure 1: **SearchOS** architecture.

## 3 SEARCHOS

To keep execution aligned with the evolving search state, we implement **SearchOS** as a stateful closed loop (Figure 1). We dispatch unresolved schema gaps through the orchestrator, collect observations through SOCM, and ground each interaction through middleware. Once complete, an optional writer can produce the final report with inline citations.

### 3.1 SEARCH-ORIENTED CONTEXT MANAGEMENT

To coordinate search beyond individual interaction histories, SOCM maintains four complementary views of the same task: what to do, what evidence has been found, what remains unresolved, and which attempts have failed. At step $t$, the shared state is

$$\mathcal{M}_t = (\mathcal{F}_t, \mathcal{G}_t, \mathcal{C}_t, \mathcal{W}_t), \tag{3}$$

where $\mathcal{F}_t$ is Frontier Task, $\mathcal{G}_t$ the Evidence Graph, $\mathcal{C}_t$ the Coverage Map, and $\mathcal{W}_t$ Failure Memory. The system maintains this state outside agent conversations. An agent with role $r$ and assignment $z_t$ receives a task-relevant view

$$x_t^{(r)} = \phi_r(\mathcal{M}_t; z_t), \tag{4}$$

through the projection $\phi_r$. The orchestrator sees global gaps, search agents see assigned targets and relevant evidence and failures, and the writer sees findings with their citations. These views are regenerated from the current shared state before inference.

#### 3.1.1 FRONTIER TASK

Frontier Task turns unresolved information needs into schedulable work. We represent each task and the ready set as

$$f_j = (\kappa_j, s_j, p_j, B_j, Z_j, a_j, n_j), \qquad \mathcal{R}_t = \{f_j \in \mathcal{F}_t \mid s_j = \textsc{Pending} \wedge B_j \subseteq \mathcal{T}_t\}, \tag{5}$$

where $\kappa_j$, $s_j$, $p_j$, $B_j$, $Z_j$, $a_j$, and $n_j$ denote type, status, priority, dependencies, target cells, assigned agent, and attempt count; $\mathcal{T}_t$ contains terminal task IDs. This separates the work to be done from its assignment to agents; the orchestrator dispatches ready tasks as execution slots become available.

#### 3.1.2 EVIDENCE GRAPH

The Evidence Graph connects atomic findings to the schema and their sources:

$$\begin{aligned} \mathcal{G}_t &= (\mathcal{N}_t, \mathcal{L}_t), \qquad g_i = (v_i, u_i, x_i, \mathbf{b}_i, \gamma_i, \tau_i, s_i), \\ \mathcal{L}_t &\subseteq \mathcal{N}_t \times \mathcal{R}_E \times \mathcal{N}_t, \qquad \mathcal{R}_E = \{\textsc{Support}, \textsc{Conflict}, \textsc{Refine}\}. \end{aligned} \tag{6}$$

A node $g_i \in \mathcal{N}_t$ records value $v_i$, source URL $u_i$, supporting excerpt $x_i$, schema binding $\mathbf{b}_i = (m_i, e_i, a_i)$, extraction confidence $\gamma_i \in [0, 1]$, provenance tier $\tau_i$, and status $s_i$. The binding identifies a table, entity, and attribute; edges express support, conflict, or refinement between findings. Binding evidence to individual cells preserves provenance when agents combine results across sources.

#### 3.1.3 COVERAGE MAP

The Coverage Map translates evidence into task progress. Each schema cell $c = (m, e, a)$ has a status (missing, filled, uncertain, or unreachable), supporting findings $H_c$, primary evidence, and a conflict flag. These statuses distinguish missing information from unresolved or inaccessible evidence and guide subsequent search. When several findings support a cell, we select

$$g_c^* = \arg\max_{g \in H_c} \big(\tau(g), \alpha(g), \widetilde{\gamma}(g)\big), \tag{7}$$

in lexicographic order, where $\tau(g)$ is the recorded provenance tier and $\alpha(g)$ ranks schema alignment derived from extraction labels (full $>$ partial $>$ loose). The adjusted confidence is $\widetilde{\gamma}(g) = w(g)\gamma(g)$, where $\gamma(g)$ is the recorded extraction confidence and $w(g)$ combines a fixed source-authority weight with a discount for derived agent summaries. Let $\Omega_t$ be the currently known schema cells and $\mathcal{E}_m(t)$ the discovered rows of table $m$. Coverage is computed as

$$\mathrm{Cov}(\mathcal{C}_t) = \frac{\sum_{c \in \Omega_t} \mathbf{1}[s(c) = \textsc{Filled}]}{|\Omega_t| + \sum_{m:\,|\mathcal{E}_m(t)|=0} |\mathcal{A}_m|}. \tag{8}$$

Here $s(c)$ denotes cell status. The denominator keeps a declared empty table unfinished. Discovering entities expands $\Omega_t$, so coverage measures completion of known rows, not exhaustive entity enumeration.

#### 3.1.4 FAILURE MEMORY

Failure Memory associates unsuccessful attempts with their search context and corrective guidance:

$$w_k = (\eta_k, \sigma_k, \chi_k), \qquad \mathcal{W}_t = \{w_k\}, \tag{9}$$

where $\eta_k$ is the failure type, $\sigma_k$ its task context, and $\chi_k$ the corrective guidance. Relevant records enter each agent's projected view, helping it avoid uninformative queries or inaccessible sources already encountered by other agents.

**Shared state updates.** The four components evolve through a common state transition,

$$\mathcal{M}_{t+1} = \mathcal{U}(\mathcal{M}_t, \delta_t), \tag{10}$$

where $\delta_t$ denotes a tool observation, agent report, or middleware signal. Accepted evidence updates both the Evidence Graph and Coverage Map, while task outcomes and detected failures update Frontier Task and Failure Memory. A shared locked interface keeps related updates within the same state transition. Subsequent projections expose these changes to agents, connecting search outcomes to future decisions. Middleware realizes these transitions in Sections 3.3.2 and 3.3.3.

### 3.2 PIPELINE-PARALLEL MULTI-AGENT ORCHESTRATION

**Agent roles.** In **SearchOS**, we adopt an orchestrator–worker architecture (Wu et al., 2023). The *orchestrator* builds the schema, prioritizes gaps, and decides when to stop; *explore agents* identify candidates and sources to provide priors for schema design and following subtasks; *search agents* collect grounded evidence for scoped gaps; and the *writer* produces a cited report.

**Continuous dispatch.** To avoid idle time from synchronized batches, we apply a pipeline parallelism (Huang et al., 2019; Narayanan et al., 2019) to role-scoped work, synchronized through SOCM. At time $t$, let $b_t$ be the number of available execution slots and $\mathcal{R}_t$ the ready set defined by Frontier Task. We dispatch

$$\mathcal{D}_t = \mathrm{Top}_{\min(b_t, |\mathcal{R}_t|)}\big(\mathcal{R}_t;\, p\big), \tag{11}$$

where $p$ is frontier priority. After each completion, we update SOCM, recompute $\mathcal{R}_t$, and refill the released slot. This event-driven policy overlaps roles when dependencies permit and avoids batch stragglers.

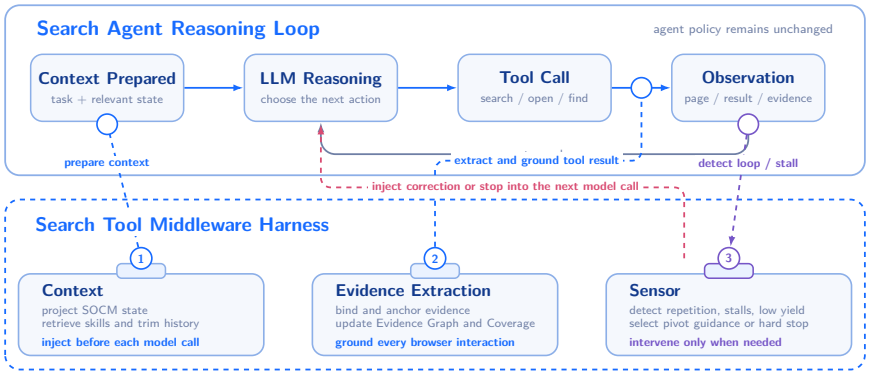



Figure 2: Illustration of middleware interventions in the Search Agent loop.

**Toolset design.** We carefully design a rich set of tools for search tasks and expose role-scoped toolsets (Appendix G) to keep each role's action space focused. We reserve schema and task scheduling for the orchestrator, browsing for explore and search agents, skill for search agents, and outline and state access for the writer. Browser observations update $\mathcal{G}$ and $\mathcal{C}$ only through middleware.

### 3.3 Search Tool Middleware Harness

During long-horizon search process, agents face both model-level failures and system-level disruptions, including tool errors, malformed outputs, stalled loops. Because **prompt-level safeguards cannot reliably cover heterogeneous post-trained agents**, we introduce a Search Tool Middleware Harness that intercepts the agent loop at model and tool calls. As illustrated in Figure 2, its three components form the following execution flow:

$$\widetilde{h}_t^{(r)} = \mathcal{H}_{\text{ctx}}(h_t, \mathcal{M}_t), \qquad \mathcal{M}_{t+1} = \mathcal{H}_{\text{evidence}}(o_t, \mathcal{M}_t), \qquad a_{t+1} = \mathcal{H}_{\text{sensor}}(\mathcal{M}_t, \mathcal{M}_{t+1}, \xi_t), \tag{12}$$

where $\widetilde{h}_t^{(r)}$ is the role-specific model context, $o_t$ is a tool observation, $\xi_t$ contains runtime counters, and $a_{t+1}$ is a continue, correct, or stop action. Context is prepared before inference; after tool execution, evidence is grounded and committed before sensors evaluate progress.

#### 3.3.1 Context Middleware

Before each model call, Context Middleware composes recent history $h_t$, shared state $\mathcal{M}_t$, and retrieved skills for role $r$ and assignment $z_t$:

$$\begin{aligned} x_t^{(r)} &= \phi_r(\mathcal{M}_t; z_t), \qquad \mathcal{K}_t = \text{TopK}\Big(\text{Retrieve}(q, \mathcal{S}, x_t^{(r)})\Big), \\ \widetilde{h}_t^{(r)} &= h_t \oplus x_t^{(r)} \oplus \psi(\mathcal{K}_t), \end{aligned} \tag{13}$$

where $\phi_r$ projects shared state for the assigned task, $\mathcal{K}_t$ is the selected skill set, $\psi$ renders skills, and $\oplus$ composes context. The projection retains targets, evidence, gaps, and failures without serializing the full history.

#### 3.3.2 Evidence Extraction Middleware

To keep sub-agents' reasoning **focused and coherent** during long-horizon search, we offload evidence extraction and persistence to middleware, freeing sub-agents from repeated database tool calls. Evidence Extraction Middleware extracts schema-bound candidates from observation $o_t$:

$$\widehat{\mathcal{E}}_t = \text{Extract}(o_t, \mathcal{S}, \mathcal{C}_t), \tag{14}$$

where each candidate contains an entity, attribute, value, source, and supporting span. Acceptance requires both schema binding and span anchoring:

$$\mathcal{E}_t^+ = \Big\{ z \in \widehat{\mathcal{E}}_t \;\Big|\; \text{Bind}(z, \mathcal{S}) = 1 \wedge \text{Anchor}(z, o_t) = 1 \Big\}. \tag{15}$$

We commit accepted evidence in one transition:

$$(\mathcal{G}_{t+1}, \mathcal{C}_{t+1}) = \mathcal{U}_{\text{evidence}}(\mathcal{G}_t, \mathcal{C}_t, \mathcal{E}_t^+). \tag{16}$$

This atomically records provenance and updates Coverage Map cells, making evidence capture a consequence of tool execution.

#### 3.3.3 SENSOR MIDDLEWARE

To detect stalls and enforce resource limits, we use Sensor Middleware to measure coverage and evidence progress. Let $\mathrm{Cov}(\mathcal{C}_t)$ denote grounded schema coverage:

$$\Delta_t^{\text{cov}} = \mathrm{Cov}(\mathcal{C}_t) - \mathrm{Cov}(\mathcal{C}_{t-1}), \qquad \Delta_t^{\text{ev}} = |\mathcal{G}_t| - |\mathcal{G}_{t-1}|. \tag{17}$$

Over window $w$, the trajectory stalls when neither quantity grows:

$$s_t = \mathbb{I}\left[\sum_{j=t-w+1}^{t} \Delta_j^{\text{cov}} = 0 \ \wedge \sum_{j=t-w+1}^{t} \Delta_j^{\text{ev}} = 0\right]. \tag{18}$$

Budget pressure is the maximum normalized use of iterations, tool calls, and time:

$$\rho_t = \max\left(\frac{n_t^{\text{iter}}}{B^{\text{iter}}}, \frac{n_t^{\text{tool}}}{B^{\text{tool}}}, \frac{\ell_t}{B^{\text{time}}}\right). \tag{19}$$

where $\ell_t$ is execution time. Sensor Middleware computes the stall state $s_t$ and budget pressure $\rho_t$.

### 3.4 HIERARCHICAL SEARCH SKILL SYSTEM

Search requires reusable knowledge at three levels: *orchestrator skills* guide task decomposition, schema alignment, and answer synthesis; *strategy skills* provide source-independent methods for query formulation, entity discovery, and verification; and *access skills* encode executable procedures for retrieving structured information from specific sources.

Orchestrator skills provide global guidance, while relevant strategy and access skills are selected for individual search tasks and exposed as instructions and callable tools, respectively.

**Initialization.** We cold-start strategy skills from Web2BigTable (Huang et al., 2026) and provide orchestrator skills as global playbooks. Access skills are generated from **SearchOS** execution trajectories, allowing the library to expand as the system encounters new sources.

**Skill generation & update.** To turn search experience into reusable procedures, the generation pipeline examines completed task trajectories for sources that require repeated visits or yield little evidence through generic browsing.

As each session ends, **SearchOS** selects websites with recurring information accessing need, probes representative pages, and constructs executable code scripts. Candidate skills undergo interface checks and execution probes before entering the skill library. Let $\mathcal{B}_j$ be the skill library before task $j$ and $\zeta_j$ its execution trajectory. We model the update as

$$\mathcal{B}_{j+1} = \mathcal{B}_j \cup \mathrm{Validate}(\mathrm{Generate}(\zeta_j; \mathcal{B}_j)), \tag{20}$$

where $\mathrm{Generate}$ identifies reusable access needs and constructs candidate skills, and $\mathrm{Validate}$ retains candidates that pass these checks. The updated library supplies additional guidance to subsequent tasks. Figure 6 summarizes the skill library, and Appendix E illustrates an executable access skill.

## 4 EXPERIMENTS

### 4.1 EXPERIMENTAL SETUP

**Benchmarks.** We evaluate **SearchOS** on four benchmarks spanning two task categories. For *structured information seeking*, **WideSearch** (Wong et al., 2025) comprises 200 manually curated questions (100 English and 100 Chinese) across more than 15 domains, each requiring large-scale

Table 1: Main results. The best score is in **bold**, and the second-best is underlined. $\Delta$ is the improvement of SearchOS over the strongest baseline.

| | WideSearch | | | | | | GISA | | | | | XBench | GAIA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Item | | | Row | | | Table | | Set | List | Item | ACC | ACC |
| **Method** | **P** | **R** | **F1** | **P** | **R** | **F1** | **Item F1** | **Row F1** | **F1** | **F1** | **EM** | | |
| *Single-Agent Methods* | | | | | | | | | | | | | |
| ReAct | 73.8 | 63.8 | 66.3 | 48.2 | 41.6 | 43.3 | 66.8 | 45.9 | 49.1 | 52.3 | 0.0 | 41.0 | 45.1 |
| Plan-and-Solve | 77.1 | 65.8 | 68.7 | 48.6 | 41.5 | 43.3 | 66.2 | 42.9 | 51.1 | 50.3 | 11.1 | 41.0 | 50.0 |
| *Multi-Agent Systems* | | | | | | | | | | | | | |
| MindSearch | 68.5 | 52.4 | 56.1 | 37.5 | 28.3 | 30.4 | 52.4 | 28.9 | 15.2 | 17.9 | 11.1 | 30.0 | 41.2 |
| Table-as-Search | 74.2 | 64.8 | 67.0 | 45.1 | 40.3 | 41.5 | 63.9 | 42.7 | 43.3 | 41.8 | 16.7 | 44.0 | 53.9 |
| A-MapReduce | 74.1 | 65.2 | 67.2 | 43.8 | 38.8 | 40.1 | 63.3 | 41.6 | 49.9 | 44.6 | 11.1 | 25.0 | 41.2 |
| Web2BigTable | 69.3 | 64.4 | 65.2 | 48.9 | 45.6 | 46.3 | 57.9 | 35.6 | 45.4 | 51.3 | **27.8** | 42.0 | 56.9 |
| **SearchOS** | **79.1** | **71.2** | **73.2** | **51.3** | **46.3** | **47.6** | **68.9** | **48.2** | **63.2** | **59.4** | **27.8** | **49.0** | **59.8** |
| $\Delta$ | +2.0 | +5.4 | +4.5 | +2.4 | +0.7 | +1.3 | +2.1 | +2.3 | +12.1 | +7.1 | 0.0 | +5.0 | +2.9 |

collection of objectively verifiable facts into a complete table; **GISA** (Zhu et al., 2026) contains 373 human-crafted queries in four structured formats (item, set, list, and table), coupling multi-hop reasoning with broad cross-source aggregation.

For *complex, multi-step autonomous navigation tasks*, we use text-only questions from **GAIA** (Mialon et al., 2023) and Chinese deep-search tasks from **XBench** (Chen et al., 2025a) (XBench-DeepSearch), which require planning, web browsing, and reasoning across multiple steps.

**Baselines.** We compare **SearchOS** against baselines in two categories: *single-agent* methods, **ReAct** (Yao et al., 2023) and **Plan-and-Solve** (Wang et al., 2023), and *multi-agent* systems, **MindSearch** (Chen et al., 2025c), **Table-as-Search** (Lan et al., 2026), **A-MapReduce** (Chen et al., 2026) and **Web2BigTable** (Huang et al., 2026).

**Configuration.** We use `GLM-5` as the backbone for all methods and `Qwen3.5-35B-A3B` for Evidence Extraction of **SearchOS**.For **SearchOS**, we first run each case, generate access skills for up to three websites from its execution trajectory, and then rerun the case with these skills for evaluation.

For WideSearch and GISA, we report item-/row-level Precision, Recall, and F1, scoring individual cells or fully correct rows and Exact Match (EM) for item tasks. We report averages over three runs for WideSearch and GISA, and single-run answer accuracy (ACC) for GAIA and XBench. We cap each session at 50 orchestrator iterations, 8 parallel sub-agents, and a 1800-second runtime budget.

### 4.2 Main Results

As shown in Table 1, **SearchOS** achieves best performance across diffrenent benchmarks. It improves WideSearch Item F1 from the strongest baseline's 68.7 to 73.2 and Row F1 from 46.3 to 47.6. On GISA, **SearchOS** leads Table Item F1, Table Row F1, Set F1, and List F1 by 2.1, 2.3, 12.1, and 7.1 points, respectively. It also obtains 49.0 ACC on XBench and 59.8 ACC on GAIA, outperforming the strongest baselines by 5.0 and 2.9 points.

## 5 Ablations & Analysis

### 5.1 Ablating Search-Oriented Context Management

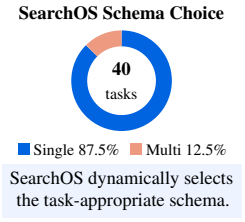
We ablate role-specific SOCM projections and Failure Memory while preserving the remaining system components (Table 2). Removing SOCM projections reduces WideSearch Item F1 from 73.2 to 66.5, Row F1 from 47.6 to 40.0, and GISA Set F1 from 63.2 to 53.0. Removing Failure Memory yields 67.6, 39.6, and 54.4 on the same metrics. The larger reductions on WideSearch row-level completion and GISA set aggregation suggest that both mechanisms primarily help the system maintain coherent progress across distributed search results.

Table 2: Ablation of SOCM projection and Failure Memory.

| | WideSearch | | | | | | GISA | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Item | | | Row | | | Table Item | | | Table Row | | | Set | | | List |
| **Setting** | **P** | **R** | **F1** | **P** | **R** | **F1** | **P** | **R** | **F1** | **P** | **R** | **F1** | **P** | **R** | **F1** | **F1** |
| **SearchOS** | **79.1** | **71.2** | **73.2** | **51.3** | **46.3** | **47.6** | 68.9 | **71.6** | **68.9** | 48.2 | **50.0** | **48.2** | **69.0** | **66.0** | **63.2** | **59.4** |
| w/o SOCM Projection | 73.3 | 64.4 | 66.5 | 43.7 | 38.7 | 40.0 | **69.6** | 69.7 | 68.5 | 47.1 | 46.7 | 46.3 | 58.8 | 53.4 | 53.0 | 58.6 |
| w/o Failure Memory | 74.7 | 65.0 | 67.6 | 43.7 | 37.9 | 39.6 | 69.1 | 69.0 | 67.6 | **48.7** | 47.7 | 47.4 | 60.8 | 56.2 | 54.4 | 56.3 |

Table 3: Relational schema completion analysis.

| | Item-level | | | Row-level | | |
|---|---|---|---|---|---|---|
| **Setting** | **Precision** | **Recall** | **F1** | **Precision** | **Recall** | **F1** |
| Fixed Single-Table | 54.7 | 46.7 | 47.9 | 33.5 | 28.2 | 29.2 |
| Fixed Multi-Table | 66.9$_{(+12.2)}$ | 55.7$_{(+9.0)}$ | 58.3$_{(+10.4)}$ | 37.4$_{(+3.9)}$ | 32.6$_{(+4.4)}$ | 34.2$_{(+5.0)}$ |
| Oracle Single/Multi | 70.3$_{(+3.4)}$ | 60.7$_{(+5.0)}$ | 62.4$_{(+4.1)}$ | 44.8$_{(+7.4)}$ | 39.7$_{(+7.1)}$ | 41.2$_{(+7.0)}$ |
| **SearchOS** | **76.3**$_{(+6.0)}$ | **68.3**$_{(+7.6)}$ | **70.6**$_{(+8.2)}$ | **52.7**$_{(+7.9)}$ | **47.3**$_{(+7.6)}$ | **48.9**$_{(+7.7)}$ |




### 5.2 Fixed Schemas vs. Search-Time Schema Planning

**Table-as-Search** (Lan et al., 2026) organizes search around a single table. **SearchOS** extends this formulation to task-adaptive relational schemas, supporting both single-table tasks and tasks involving multiple related entity types. To assess the value of this flexibility, we randomly select 40 questions amenable to multi-table decomposition and use `GPT-5.5` to construct semantically matched fixed single-table and multi-table schemas with the same required attributes.

We compare both fixed-schema settings with autonomous schema planning by explore agents within **SearchOS**, isolating the effect of schema design from differences between systems.

As shown in Table 3, fixed multi-table outperforms fixed single-table on average, but falls short of an oracle that selects the better fixed schema for each case by Item F1, suggesting that **relational schemas can expand the available search representations**, while task-adaptive planning determines which representation is appropriate.

### 5.3 Pipeline-Parallel Scheduling Ablation

We compare batch and continuous scheduling over three paired rounds on 10 WideSearch cases, holding queries and model configuration fixed. Continuous scheduling refills released slots without waiting for the batch to finish. Table 4 shows 24.3% lower average end-to-end time, higher slot utilization and throughput, fewer LLM calls, and higher Item F1.

### 5.4 Middleware-Governed Search Agent Execution Analysis

We further analyze whether our **Search Tool Middleware Harness** keeps long-horizon search productive after detecting stagnation. Figure 3 presents three representative WideSearch trajectories with early, mid-run, and late interventions. In each column, the upper figure shows table coverage and the lower figure shows the cumulative number of discovered entities. The dashed line marks the point at which the Loop Sensor detects a low-progress search loop and triggers a strategy switch.

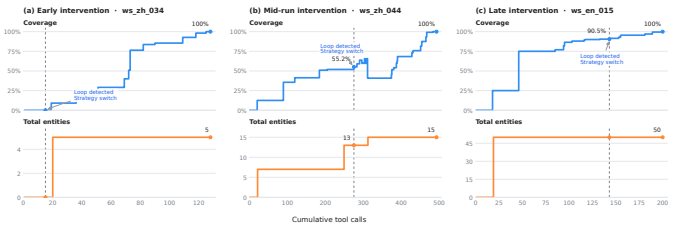



Figure 3: Middleware-governance trajectories on WideSearch.

Table 4: Pipeline scheduling ablation on WideSearch (mean across 30 trajectories per policy).

| Scheduling Policy | Time (s) (↓) | Slot Utilization (↑) | Tasks/min (↑) | LLM Calls (↓) | Item F1 (↑) |
|---|---|---|---|---|---|
| Batch (Control) | 629.1 | 34.6% | 3.0 | 341.4 | 79.7 |
| **Continuous (Ours)** | **476.3** | **41.7%** | **3.4** | **296.6** | **86.8** |

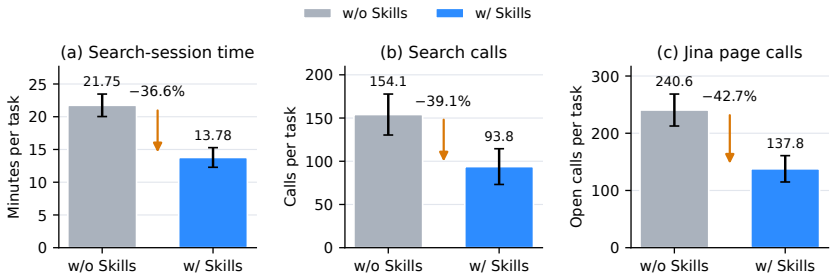



Figure 4: Efficiency analysis. Error bars show 95% confidence intervals across questions.

Across the different cases, interventions at early, middle, and late stages all restore search progress beyond the preceding stagnation period, although they affect coverage and entity discovery to different degrees. These trajectories provide mechanism-level evidence that the Loop Sensor can redirect stalled search toward productive work.

### 5.5 Skill Ablation and Efficiency

In this section, we compare **SearchOS** with and without hierarchical skills to present their contribution to search quality and efficiency.

**Effectiveness and Efficiency.** As shown in Table 5, our search skills improve average Item F1 by 3.4 points and Row F1 by 6.2 points. The larger row-level gain suggests that reusable decomposition, search, and source-access knowledge primarily helps agents assemble coherent entity records rather than recover isolated cells. For efficiency, figure 4 shows reductions of 36.6% in session time, 39.1% in search calls, and 42.7% in Jina API calls.

Table 5: Skill ablation results.

| | Item-level | | | Row-level | | |
|---|---|---|---|---|---|---|
| **Setting** | **Precision** | **Recall** | **F1** | **Precision** | **Recall** | **F1** |
| w/o Skills | 73.2 | 68.8 | 69.8 | 43.4 | 40.6 | 41.4 |
| **Full SearchOS** | **79.1** | **71.2** | **73.2** | **51.3** | **46.3** | **47.6** |
| Δ | +5.9 | +2.4 | +3.4 | +7.9 | +5.7 | +6.2 |

## 6 Related Work

Our work is mainly related to agentic information seeking, multi-agent orchestration, and agent harness engineering. Due to limited space, we discuss these lines of research in Appendix A.

## 7 Conclusion

In this paper, we introduce **SearchOS**, a multi-agent framework that externalizes long-horizon open-domain search as system-maintained state. We first formulate information seeking as relational schema completion with grounded citations. Building on this formulation, we design three core modules: (1) Search-Oriented Context Management (SOCM), which maintains shared search state and supports pipeline-parallel scheduling; (2) a Search Tool Middleware Harness, which grounds observations and handles stalls and budget limits; and (3) a Hierarchical Search Skill System, which supplies reusable orchestration, search-strategy, and source-access knowledge. Experiments across multiple benchmarks and efficiency analysis show that **SearchOS** improves answer quality and search efficiency. These results suggest that explicit search state, grounded evidence, and system-level execution control provide a practical foundation for reliable multi-agent information seeking.

## AI USE STATEMENT

In this paper, we use ChatGPT to improve the presentations of this paper and Codex to assist in experiments.[1] The authors reviewed all AI-assisted text and take responsibility for the final manuscript and all artifacts.

## REFERENCES

Anthropic. Effective context engineering for AI agents. Anthropic Engineering, September 2025. URL https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents.

Philip A Bernstein. Synthesizing third normal form relations from functional dependencies. *ACM Transactions on Database Systems (TODS)*, 1(4):277–298, 1976.

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. *CoRR*, abs/2506.13651, 2025a. doi: 10.48550/ARXIV.2506.13651. URL https://doi.org/10.48550/arXiv.2506.13651.

Mingju Chen, Guibin Zhang, Heng Chang, Yuchen Guo, and Shiji Zhou. A-mapreduce: Executing wide search via agentic mapreduce. *CoRR*, abs/2602.01331, 2026. doi: 10.48550/ARXIV.2602.01331. URL https://doi.org/10.48550/arXiv.2602.01331.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. Research: Learning to reason with search for llms via reinforcement learning. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025b. URL http://papers.nips.cc/paper_files/paper/2025/hash/7b3a0d3a0864b66f229c0a9c84f9e950-Abstract-Conference.html.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=EHg5GDnyq1.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Jiangning Liu, Wenwei Zhang, Kai Chen, and Feng Zhao. MindSearch: Mimicking human minds elicits deep ai searcher. In *International Conference on Learning Representations*, 2025c. URL https://openreview.net/forum?id=xgtXkyqw1f.

Edgar F Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema Amershi. Magentic-One: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024. URL https://arxiv.org/abs/2411.04468.

Dawei Gao, Zitao Li, Weirui Kuang, Xuchen Pan, Daoyuan Chen, Zhijian Ma, Bingchen Qian, Liuyi Yao, Lin Zhu, Chen Cheng, Hongzhu Shi, Yaliang Li, Bolin Ding, and Jingren Zhou. Agentscope: A flexible yet robust multi-agent platform. *CoRR*, abs/2402.14034, 2024. doi: 10.48550/ARXIV.2402.14034. URL https://doi.org/10.48550/arXiv.2402.14034.

[1] https://chatgpt.com/

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous RL. *CoRR*, abs/2508.07976, 2025. doi: 10.48550/ARXIV.2508.07976. URL https://doi.org/10.48550/arXiv.2508.07976.

Yu Gu, Yiheng Shu, Hao Yu, Xiao Liu, Yuxiao Dong, Jie Tang, Jayanth Srinivasa, Hugo Latapie, and Yu Su. Middleware for LLMs: Tools are instrumental for language agents in complex environments. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7646–7663, 2024. doi: 10.18653/v1/2024.emnlp-main.436. URL https://aclanthology.org/2024.emnlp-main.436/.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven K. S. Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Xu Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 103–112, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/093f65e080a295f8076b1c5722a46aa2-Abstract.html.

Yuxuan Huang, Yihang Chen, Zhiyuan He, Yuxiang Chen, Ka Yiu Lee, Huichi Zhou, Weilin Luo, Meng Fang, and Jun Wang. Web2bigtable: A bi-level multi-agent LLM system for internet-scale information search and extraction. *CoRR*, abs/2604.27221, 2026. doi: 10.48550/ARXIV.2604.27221. URL https://doi.org/10.48550/arXiv.2604.27221.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 1658–1677. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.91. URL https://doi.org/10.18653/v1/2024.acl-long.91.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516, 2025. doi: 10.48550/ARXIV.2503.09516. URL https://doi.org/10.48550/arXiv.2503.09516.

Jiajie Jin, Xiaoxi Li, Yuyao Zhang, Guanting Dong, Zhao Yang, Yutao Zhu, and Zhicheng Dou. Hira: Decoupling planning and execution with hierarchical reasoning in deep search. In Alistair Moffat, Falk Scholer, Hannah Bast, Marc Najork, and Min Zhang (eds.), *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2026, MelbourneVICAustralia, July 20-24, 2026*, pp. 657–667. ACM, 2026. doi: 10.1145/3805712.3809666. URL https://doi.org/10.1145/3805712.3809666.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An LLM compiler for parallel function calling. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 24370–24391. PMLR, 2024. URL https://proceedings.mlr.press/v235/kim24y.html.

Tian Lan, Felix Henry, Bin Zhu, Qianghuai Jia, Junyang Ren, Qihang Pu, Haijun Li, Longyue Wang, Zhao Xu, and Weihua Luo. Table-as-search: Formulate long-horizon agentic information seeking as table completion. *CoRR*, abs/2602.06724, 2026. doi: 10.48550/ARXIV.2602.06724. URL https://doi.org/10.48550/arXiv.2602.06724.

Junjie Li, Xi Xiao, Yunbei Zhang, Chen Liu, Lin Zhao, Xiaoying Liao, Yingrui Ji, Janet Wang, Jianyang Gu, Yingqiang Ge, Weijie Xu, Xi Fang, Xiang Xu, Tianchen Zhao, Youngeun Kim, Tianyang Wang, Jihun Hamm, Smita Krishnaswamy, Jun Huan, and Chandan K. Reddy. Agent harness engineering: A survey. OpenReview preprint, 2026a. URL https://openreview.net/forum?id=eONq7FdiHa.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Rui Ye, Yida Zhao, Liwen Zhang, Litu Ou, Dingchu Zhang, Xixi Wu, Jialong Wu, Xinyu Wang, Zile Qiao, Zhen Zhang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Websailor-v2: Bridging the chasm to proprietary agents via synthetic data and scalable reinforcement learning. *CoRR*, abs/2509.13305, 2025. doi: 10.48550/ARXIV.2509.13305. URL https://doi.org/10.48550/arXiv.2509.13305.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. Webthinker: Empowering large reasoning models with deep research capability. *Advances in Neural Information Processing Systems*, 38:120091–120131, 2026b.

Ryan Lopopolo. Harness engineering: Leveraging Codex in an agent-first world. OpenAI Engineering, February 2026. URL https://openai.com/index/harness-engineering/.

Kun Luo, Hongjin Qian, Zheng Liu, Ziyi Xia, Shitao Xiao, Siqi Bao, Jun Zhao, and Kang Liu. Infoflow: Reinforcing search agent via reward density optimization. *CoRR*, abs/2510.26575, 2025. doi: 10.48550/ARXIV.2510.26575. URL https://doi.org/10.48550/arXiv.2510.26575.

Kai Mei, Zelong Li, Shuyuan Xu, Ruosong Ye, Yingqiang Ge, and Yongfeng Zhang. AIOS: LLM agent operating system. *CoRR*, abs/2403.16971, 2024. doi: 10.48550/ARXIV.2403.16971. URL https://doi.org/10.48550/arXiv.2403.16971.

Qianyu Meng, Yanan Wang, Liyi Chen, Yihang Li, Wei Wu, Wenyuan Jiang, Qimeng Wang, Chengqiang Lu, Yan Gao, Yi Wu, and Yao Hu. Agent harness for large language model agents: A survey. *Preprints*, April 2026. doi: 10.20944/preprints202604.0428.v3. URL https://doi.org/10.20944/preprints202604.0428.v3.

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. *CoRR*, abs/2112.09332, 2021. URL https://arxiv.org/abs/2112.09332.

Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. Pipedream: generalized pipeline parallelism for DNN training. In Tim Brecht and Carey Williamson (eds.), *Proceedings of the 27th ACM Symposium on Operating Systems Principles, SOSP 2019, Huntsville, ON, Canada, October 27-30, 2019*, pp. 1–15. ACM, 2019. doi: 10.1145/3341301.3359646. URL https://doi.org/10.1145/3341301.3359646.

Xuying Ning, Katherine Tieu, Dongqi Fu, Tianxin Wei, Zihao Li, Yuanchen Bei, Jiaru Zou, Mengting Ai, Zhining Liu, Ting-Wei Li, Lingjie Chen, Yanjun Zhao, Ke Yang, Bingxuan Li, Cheng Qian, Gaotang Li, Xiao Lin, Zhichen Zeng, Ruizhong Qiu, Sirui Chen, Yifan Sun, Xiyuan Yang, Ruida Wang, Rui Pan, Chenyuan Yang, Dylan Zhang, Liri Fang, Zikun Cui, Yang Cao, Pan Chen, Dorothy Sun, Ren Chen, Mahesh Srinivasan, Nipun Mathur, Yinglong Xia, Hong Li, Hong Yan, Pan Lu, Lingming Zhang, Tong Zhang, Hanghang Tong, and Jingrui He. Code as agent harness. *CoRR*, abs/2605.18747, 2026. doi: 10.48550/ARXIV.2605.18747. URL https://doi.org/10.48550/arXiv.2605.18747.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560, 2023. doi: 10.48550/ARXIV.2310.08560. URL https://doi.org/10.48550/arXiv.2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In Sean Follmer, Jeff Han, Jürgen Steimle, and Nathalie Henry Riche (eds.), *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA, 29 October 2023- 1 November 2023*, pp. 2:1–2:22. ACM, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186, 2024. doi: 10.18653/v1/2024.acl-long.810. URL https://aclanthology.org/2024.acl-long.810/.

Hongjin Qian and Zheng Liu. Scent of knowledge: Optimizing search-enhanced reasoning with information foraging. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/e4724af0e2a0d52ce5a0a4e084b87f59-Abstract-Conference.html.

Yujia Qin, Zihan Cai, Dian Jin, Lan Yan, Shihao Liang, Kunlun Zhu, Yankai Lin, Xu Han, Ning Ding, Huadong Wang, et al. Webcpm: Interactive web search for chinese long-form question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8968–8988, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Webshaper: Agentically data synthesizing via information-seeking formalization. *CoRR*, abs/2507.15061, 2025. doi: 10.48550/ARXIV.2507.15061. URL https://doi.org/10.48550/arXiv.2507.15061.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 10014–10037, 2023.

Haoyu Wang, Christopher M. Poskitt, and Jun Sun. Agentspec: Customizable runtime enforcement for safe and reliable LLM agents. *CoRR*, abs/2503.18666, 2025. doi: 10.48550/ARXIV.2503.18666. URL https://doi.org/10.48550/arXiv.2503.18666.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 2609–2634. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.ACL-LONG.147. URL https://doi.org/10.18653/v1/2023.acl-long.147.

Ryan Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, Wenhao Huang, Yang Wang, and Ke Wang. Widesearch: Benchmarking

agentic broad info-seeking. *CoRR*, abs/2508.07999, 2025. doi: 10.48550/ARXIV.2508.07999. URL https://doi.org/10.48550/arXiv.2508.07999.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversation. *CoRR*, abs/2308.08155, 2023. doi: 10.48550/ARXIV.2308.08155. URL https://doi.org/10.48550/arXiv.2308.08155.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

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. *CoRR*, abs/2510.24699, 2025. doi: 10.48550/ARXIV.2510.24699. URL https://doi.org/10.48550/arXiv.2510.24699.

Yuyao Zhang, Zhicheng Dou, Xiaoxi Li, Jiajie Jin, Yongkang Wu, Zhonghua Li, Ye Qi, and Ji-Rong Wen. Neuro-symbolic query compiler. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 12138–12155. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.628/.

Yuyao Zhang, Hongyu Lu, Jiajie Jin, Hongjin Qian, Shiyu Li, Zhao Yang, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Web sitemap knowledge can enhance autonomous browsing. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 29307–29321, 2026.

Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 21805–21830. Association for Computational Linguistics, 2025a. doi: 10.18653/V1/2025.EMNLP-MAIN.1106. URL https://doi.org/10.18653/v1/2025.emnlp-main.1106.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 414–431, 2025b.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 19724–19731. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29946. URL https://doi.org/10.1609/aaai.v38i17.29946.

Yutao Zhu, Xingshuo Zhang, Maosen Zhang, Jiajie Jin, Liancheng Zhang, Xiaoshuai Song, Kangzhi Zhao, Wencong Zeng, Ruiming Tang, Han Li, Ji-Rong Wen, and Zhicheng Dou. GISA: A benchmark for general information-seeking assistant. *CoRR*, abs/2602.08543, 2026. doi: 10.48550/ARXIV.2602.08543. URL https://doi.org/10.48550/arXiv.2602.08543.

# A Related Work

## A.1 Agentic Information Seeking

Tool-integrated language agents interleave search, browsing, reasoning, evidence collection, and synthesis (Nakano et al., 2021; Yao et al., 2023; Trivedi et al., 2023; Qin et al., 2023; Chen et al.,

2025c; Jin et al., 2025; Li et al., 2026b; Zheng et al., 2025b; Li et al., 2025). Their capabilities have advanced through search policy optimization (Qian & Liu, 2025; Luo et al., 2025), task decomposition (Chen et al., 2025c; 2026), synthetic interaction data (Tao et al., 2025; Li et al., 2025), and reinforcement-based post-training (Zheng et al., 2025a; Jin et al., 2025; Gao et al., 2025; Chen et al., 2025b; Zheng et al., 2025b). Yet progress typically remains in growing interaction histories, obscuring established evidence and unresolved questions.

Broad information seeking has also been formulated as structured or table-oriented completion (Tao et al., 2025; Wong et al., 2025; Zhu et al., 2026; Lan et al., 2026; Chen et al., 2026; Huang et al., 2026). Such outputs expose missing entities and attributes and support parallel collection, but do not capture task dependencies, conflicting observations, provenance, or failed attempts. **SearchOS** instead formulates the task as *relational schema completion with grounded citations* and maintains these intermediate artifacts outside the interaction history.

### A.2 Multi-Agent Orchestration Systems

Multi-agent frameworks decompose complex tasks through specialized roles, message exchange, and structured workflows (Wu et al., 2023; Hong et al., 2024; Chen et al., 2024; Qian et al., 2024; Chen et al., 2025c; Fourney et al., 2024); hierarchical approaches decouple strategic planning from specialized execution (Jin et al., 2026), while structured execution systems distinguish dependent from parallel sub-tasks and execute independent branches concurrently (Kim et al., 2024; Zhang et al., 2025). However, coordination state usually remains in conversations, plans, or general task ledgers, making redundant collection, stale work, coverage gaps, and idle capacity difficult to manage explicitly during long-horizon search.

Recent advances in agent memory and context management address related challenges through reflection, memory stores, virtual context, compression, or trajectory consolidation (Park et al., 2023; Shinn et al., 2023; Zhong et al., 2024; Packer et al., 2023; Jiang et al., 2024; Ye et al., 2025). In contrast, Search-Oriented Context Management (SOCM) externalizes intermediate search state as a Frontier Task, Evidence Graph, Coverage Map, and Failure Memory, with role-specific projections for each agent.

### A.3 Agent Harness Engineering

Recent surveys treat the agent harness as a first-class infrastructure layer governing execution, tools, context, state, lifecycle, observability, and evaluation, while engineering reports emphasize inspectable environments, feedback loops, and context curation for reliable long-horizon agents (Li et al., 2026a; Meng et al., 2026; Lopopolo, 2026; Anthropic, 2025). Prior systems instantiate parts of this layer through environment middleware, message-driven and fault-tolerant runtimes, operating-system abstractions, efficient model–tool interception, and runtime constraints (Gu et al., 2024; Gao et al., 2024; Mei et al., 2024; Wang et al., 2025; Ning et al., 2026; Zhang et al., 2026). The **SearchOS** Search Tool Middleware Harness applies these ideas to long-horizon information seeking by preparing role-specific context, validating and grounding evidence-producing observations, and monitoring repetition, stalls, and budget exhaustion.

## B System Interface

The **SearchOS-WebUI** provides a workbench for inspecting and organizing diverse long-horizon search task (Figure 5), powered by our well-designed **SearchOS** backend. Users submit requests through a conversational interface, while an execution panel exposes orchestrator activity, parallel agents, task status, and resource usage. The coverage view organizes collected values and unresolved gaps by table; users can inspect a value's supporting evidence, source URL, and cited excerpt. Agent traces and tool interactions provide a complementary view of how the search proceeds. Users can also supply follow-up instructions, stop a run, request targeted searches for selected cells, or manage their own search tasks through histories and favorites panel. By visualizing execution traces with structured results and their provenance, the workbench makes system-maintained search state directly inspectable.




Figure 5: **SearchOS** interface for a long-horizon information-seeking task. The workspace exposes the orchestration trace, pipeline parallel agent activity, and relational schema coverage.

## C SOCM State Example

This appendix provides a concrete example of the SOCM state as injected into the orchestrator's prompt during a search session.

**SOCM State Snapshot (Simplified)**

**== Frontier Memory ==**
Active: 3/200 — Completed: 7 — Blocked: 1

| ID | Question | Kind | Status |
|---|---|---|---|
| T-008 | Revenue of Company X in 2024? | search | RUNNING |
| T-009 | CEO of Company Y? | search | PENDING |
| T-010 | Compose final report | write | BLOCKED (by T-008) |

**== Coverage Map ==**
Table: "Fortune 500 Comparison" — Filled: 18/30 (60%)

| Entity | Revenue | CEO | Founded |
|---|---|---|---|
| Company X | ✓ | ✓ | – |
| Company Y | ✓ | – | ✓ |
| Company Z | – | – | – |

**== Evidence Summary ==**
Total nodes: 24 — Active: 21 — Conflicts: 2
Pending resolution: Company X revenue (source A: $1.2B vs source B: $1.4B)

## D Skill Library Statistics

Figure 6 summarizes the pre-built skill library. The word cloud shows frequent concepts in skill metadata, while the access-skill panel reports skill counts and the mean number of callable functions per skill across source domains. These views characterize the library's topical coverage and retrieval interfaces, rather than how often skills are invoked. Their contribution to search quality and efficiency is evaluated separately in Section 5.5.




(a) Frequent skill keywords.

(b) Access-skill counts and functions.

Figure 6: Pre-built skills: (a) weighted keyword frequency; (b) access-skill counts and mean functions by domain.

## E Executable Access Skill Case

An executable access skill is packaged as three aligned artifacts: `skill.md` provides source-specific usage guidance, `manifest.yaml` declares the typed parameter schema exposed to the agent, and `executor.py` implements dispatch, retrieval, parsing, and normalization. The following abridged case is taken from the current Senate.gov access skill.

**Access Skill Case: Senate.gov (Abridged)**

```
# manifest.yaml
description: Fetch current U.S. Senators by state from senate.gov
params_schema:
  function:
    type: string
    required: true
    description: one of
      [get_senators_by_state, list_all_states, get_state_history]
  state_code:
    type: string
    required: false

# typed invocation selected by the agent
{"function": "get_senators_by_state", "state_code": "OK"}

# executor.py dispatch (simplified)
async def execute(params, ctx=None):
    function = params.get("function")
    async with httpx.AsyncClient(timeout=30, follow_redirects=True)
    as client:
        if function == "get_senators_by_state":
            return await get_senators_by_state(client,
            params["state_code"])
        if function == "list_all_states":
            return await list_all_states(client)
        if function == "get_state_history":
            return await get_state_history(client,
            params["state_code"])

# normalized output schema
{"success": true, "state_code": "OK", "state_name": "Oklahoma",
 "senators": [{"name": ..., "party": ..., "website": ...,
               "office_address": ..., "phone": ...}],
 "senator_count": 2}
```

The manifest constrains the agent to a small operation vocabulary, while the executor hides source-specific URL construction, HTTP error handling, and BeautifulSoup parsing. Consequently, the agent receives stable structured records even though the underlying evidence is extracted from heterogeneous Senate state-profile pages.

Table 6: Round-wise scheduling results (median across 10 cases).

| Round | Time Δ (↓) | Tokens Δ (↓) | Item F1 Δ (pp) |
|---|---|---|---|
| 1 | −32.6% | −27.7% | +2.35 |
| 2 | −32.3% | −31.2% | +15.00 |
| 3 | −28.6% | −10.8% | +1.85 |

Table 7: Role-scoped tools. ✓ denotes availability.

| Tool Group | Orchestrator | Search | Explore | Writer |
|---|---|---|---|---|
| Simple Browser (search/open/find) | – | ✓ | ✓ | – |
| Schema & Entity CRUD | ✓ | – | – | – |
| Task Queue & Coordination | ✓ | – | – | – |
| Outline Management | – | – | – | ✓ |
| SOCM Read | – | – | – | ✓ |
| Skill Catalog (list/load) | – | ✓ | – | ✓ |

# F Additional Scheduling Results

Table 6 complements the pooled scheduling results in Section 5.3 with round-wise median changes across the same 10 WideSearch cases. Changes are measured relative to batch scheduling: negative time and token changes indicate savings, while positive Item F1 changes indicate quality gains. The median changes favor continuous dispatch in all three rounds, although their magnitudes vary. These repetitions assess run-to-run variability on a fixed query set rather than performance on additional queries.

# G Implementation Reference

For reproducibility, this appendix records the tool interfaces summarized in Section 3.2: the Simple Browser operations and the orchestrator's schema, task-queue, and writer tools. These interfaces are implementation choices rather than part of the core method.

## G.1 Simple Browser

**Browser State Abstraction.** We model the browser state as a tuple $\mathcal{B} = (\mathcal{P}, \mathcal{H}, \sigma)$, where $\mathcal{P}$ is a page cache mapping URLs to rendered page contents, $\mathcal{H}$ is a LIFO page stack recording the navigation history, and $\sigma$ is a scroll cursor tracking the current viewport position within the top-of-stack page. Every browser operation that produces a new page—a search query, a page fetch, or a find operation—pushes the result onto $\mathcal{H}$, establishing a natural navigation history.

This stack-based design provides several advantages over a flat URL-based model:

- **Contextual link resolution**: When the agent issues `open(3)`, the browser walks the stack backwards to find the most recent page containing link ID 3, resolving the numeric reference unambiguously.
- **Implicit back-navigation**: The stack preserves the full navigation path, enabling further trajectory analysis.
- **Cross-page find**: `find` results reference the *source page* (the top non-find page on the stack), and subsequent `open(match_id)` calls auto-jump to the matched line, enabling efficient in-page exploration without losing context.

**Browser Tools.** Table 8 summarizes the three browser operations.

Figure 7 illustrates the rendered outputs an agent receives from `open()` and `find()` calls.

Table 8: Simple Browser tool specifications.

| Tool | Parameters | Description |
|---|---|---|
| `search` | `query`: str | Web search; return URLs, titles, and snippets. |
| `open` | `id_or_url`: int\|str; `loc`: int | Visit a page by URL or link ID, starting at a given line. |
| `find` | `pattern`: str\|list | Grep keywords from the currently open page. |

`open(0)`

```
[Now viewing] GPT-4 Technical Report - OpenAI
URL: https://openai.com/research/gpt-4
**viewing lines [0 - 42] of 318**

L0: # GPT-4 Technical Report
L1:
L2: [0+OpenAI+openai.com] | [1+API
Docs+platform.openai.com]
L3:
L4: GPT-4 is a large multimodal model that
accepts image
L5: and text inputs and produces text outputs.
While less
L6: capable than humans in many real-world
scenarios, it
L7: exhibits human-level performance on various
professional
L8: and academic benchmarks, including passing
the bar exam
L9: with a score in the top 10% of test takers
L10: [2+PDF+arxiv.org].
L11:
L12: ## Performance Highlights
L13: | Benchmark      | GPT-4 Score | GPT-3.5
Score |
L14:
|----------------|-------------|---------------|
L15: | Bar Exam (MBE) | 298/400     | 213/400
|
L16: | SAT Math       | 700/800     | 590/800
|
L17: | GRE Quantit.   | 163/170     | 147/170
|
...
```

`find("bar exam")`

```
[Find results] 3 matches on current page

# Match 0 -- call open(0) to jump here
  L6:  capable than humans in many real-world
  scenarios, it
  L7:  exhibits human-level performance on
  various professional
> L8:  and academic benchmarks, including passing
the bar exam
  L9:  with a score in the top 10% of test takers
  L10: [2+PDF+arxiv.org].

# Match 1 -- call open(1) to jump here
  L13: | Benchmark      | GPT-4 Score | GPT-3.5
  Score |
  L14: |----------------|-------------|---------|
  ------|
> L15: | Bar Exam (MBE) | 298/400     | 213/400
|
  L16: | SAT Math       | 700/800     | 590/800
  |
  L17: | GRE Quantit.   | 163/170     | 147/170
  |

# Match 2 -- call open(2) to jump here
  L156: ## Appendix A: Detailed Exam Performance
  L157: On the Uniform Bar Examination, GPT-4
  achieved a
> L158: scaled score of 298, placing it in the
top 10% of bar
  L159: exam takers across all tested
  jurisdictions in 2023.
  L160: This represents a 40% improvement over
  GPT-3.5.
...
```

Figure 7: Browser tool outputs as seen by a search agent. **Left**: `open()` renders a page with line-numbered markdown and bracket-style link references (`[id†title†domain]`). **Right**: `find()` returns numbered matches with context; calling `open(match_id)` jumps to the source line.

### G.2 Schema and Entity Management Tools

The orchestrator manages the relational search schema $\mathcal{S}$ through a suite of CRUD tools that operate on the Coverage Map (Section 3.1.3). Table 9 lists the full tool set.

Table 9: Schema and entity management tools (orchestrator only).

| Tool | Key Parameters | Description |
|---|---|---|
| `create_schema` | `tables`, `relations` | Define schema with tables, attributes, PKs, and FKs. |
| `add_entities` | `entities`, `attributes` | Add rows with optional inline values (stored as evidence). |
| `remove_entities` | `entities`, `reason` | Soft-delete rows with mandatory justification. |
| `edit_entities` | `edits` | Rename PKs or edit cells. Preserves evidence refs. |
| `inspect_table` | `table_id`, `status` | Return row list by status (filled/partial/empty). |

### G.3 Task Queue and Agent Coordination Tools

The orchestrator manages the Frontier Memory (Section 3.1.1) and sub-agent lifecycle through the tools listed in Table 10.

Table 10: Task queue and agent coordination tools (orchestrator only).

| Tool | Key Parameters | Description |
|---|---|---|
| `enqueue_tasks` | `items` | Append tasks with priority, target table, and dependencies. |
| `check_agents` | `timeout`: float | Block until $\geq$1 agent completes; return reports. |
| `stop_task` | `task_ids`, `reason` | Cancel tasks and their running sub-agents. |

### G.4 Writer Tools

The writer agent operates on a structured outline rather than producing free-form text, ensuring consistent organization and complete citation coverage. Table 11 lists the writer-specific tools.

Table 11: Writer agent tools for outline and section management.

| Tool | Key Parameters | Description |
|---|---|---|
| `read_outline` | `section_id`, `toc_only` | Return TOC or full section content. |
| `update_outline` | `ops` (JSON array) | Batch add/remove/reorder sections. |
| `write_section` | `section_id`, `cited_evidence_ids` | Write section with mandatory citations. |
| `edit_section` | `section_id`, `old`, `new` | Exact substring replacement for surgical edits. |
| `annotate_section` | `section_id`, `annotation` | Attach annotation to a section. |

In addition, the writer agent has read-only access to the full SOCM state and the skill catalog, enabling it to ground its writing in the latest evidence.

## H Agent Trajectory Case Studies

This section presents process-level case studies from complete WideSearch trajectories. We summarize the logged decision rationale, tool action, and observable state transition at each consequential step rather than reproducing unabridged token-level reasoning. The cases were selected to expose different behaviors—parallel enrichment, scope auditing, and recovery from inaccessible sources—and are illustrative rather than aggregate evidence.

### H.1 Detailed Case: Spotify 2024 Rankings

**Task and outcome.** Case `ws_en_022` asks for the global and U.S. top-ten songs on Spotify in 2024, enriched with artist, language, songwriter, producer, and release date. The run finishes in 362 seconds with 221 evidence nodes, 100% known-cell coverage, 97.5 Item F1, and 85.0 Row F1.

**Decision trace.** Table 12 reconstructs the consequential intermediate decisions from the stored trajectory. The key design choice is to establish the two authoritative ranked lists before dispatching metadata enrichment, so each sub-agent writes into a stable composite key rather than independently rediscovering row identities.

**SOCM evolution.** The orchestrator receives compact state snapshots while the Search Agents work. Early in the run, all 20 seeded rows have gaps. A mid-run snapshot reports only six rows with missing or uncertain columns. The final snapshot reports zero gaps, while explicitly warning that "all known rows filled" is not proof that the requested row set is exhaustive. This warning triggers the Stage-5 row-set audit rather than immediate termination.

Table 12: Condensed agent trajectory for `ws_en_022`. Rationale is summarized from logged steps; state values are taken from SOCM deltas.

| Stage | Decision rationale | Action | Observable state |
|---|---|---|---|
| 1. Explore | First identify an authoritative source that fixes both top-ten lists. | Dispatch one Explore Agent to inspect Spotify Wrapped and chart sources. | No schema; coverage 0%. |
| 2. Structure | The official lists expose two row dimensions: chart category and rank. | Create one $20 \times 8$ table with primary key `[Category, Rank]`; seed both lists. | 40/160 cells known; coverage 25%. |
| 3. Parallelize | Metadata lookup is independent after row identities are fixed. | Split the 20 songs into four five-song tasks; dispatch four Search Agents with the Spotify access skill. | Four agents run concurrently. |
| 4. Monitor | Merge completed shards without waiting for the slowest agent. | Repeatedly call `check_agents`; retain shared Coverage Map updates. | Coverage 25.0→68.8→89.4→100.0%. |
| 5. Audit | Full known-cell coverage does not guarantee formatting consistency or scope completion. | Inspect all 20 rows; compare known row set to the two official top-ten lists. | 20 expected rows present; zero missing rows. |
| 6. Repair | Two release dates use inconsistent separators. | Apply two targeted `edit_entities` operations, then export. | 20 complete rows; 221 evidence nodes. |

**Coverage and Governance Events (Condensed)**

| Moment | SOCM / middleware signal | Coverage |
|---|---|---|
| After schema creation | 20 known rows; all 20 have missing attributes. | 25.0% |
| First shard completes | Five rows receive title, singer, language, credits, and date evidence. | 68.8% |
| Mid-run snapshot | 20 known rows; six still have missing or uncertain attributes. | 89.4% |
| Search stalls | Loop Sensor emits four `bare_search_without_progress` reminders across two agents. | – |
| Final snapshot | 20 known rows; zero gaps; row-set audit still required. | 100.0% |

**Output excerpt.** The same song may occur in both charts, but the composite key preserves its category-specific rank while shared metadata is independently grounded.

Table 13: Selected final rows from `ws_en_022`.

| List | Rank | Title | Singer | Release Date |
|---|---|---|---|---|
| Global | 1 | Espresso | Sabrina Carpenter | 2024/04/11 |
| Global | 2 | Beautiful Things | Benson Boone | 2024/01/19 |
| U.S. | 1 | Espresso | Sabrina Carpenter | 2024/04/11 |
| U.S. | 2 | Not Like Us | Kendrick Lamar | 2024/05/04 |

### H.2 Additional Process Cases

**False saturation followed by scope repair (`ws_en_016`).** The task enumerates Michael Phelps's individual-event medals across the Olympic Games and World Aquatics Championships. The orchestrator initially splits the work by competition family. After both agents return, known-cell coverage reaches 100%, but the table contains only 16 rows while the agent reports imply roughly 31 eligible events. Instead of treating saturation as completion, the orchestrator inspects the row set,

identifies missing years and competitions, and dispatches targeted backfill tasks. The table grows to 33 and then 35 rows. A final verification pass detects a duplicate 2003 World Championships 400m individual-medley record, removes the incorrect `4:11.04` row, adds the missing 2005 100m butterfly result, and corrects medalists in several 200m freestyle rows. This trajectory illustrates why Coverage Map saturation is paired with explicit scope auditing.

**Key Logged Decisions for `ws_en_016`**

**Observation:** coverage is 100%, but only 16 rows are present; two completed agents report approximately 31 eligible medal events.
**Decision:** audit row scope before synthesis.
**Action:** inspect the table, then dispatch Olympic and World-Aquatics backfill agents.

**Observation:** two rows claim the 2003 400m individual medley with times `4:09.09` and `4:11.04`.
**Decision:** treat the duplicate identity as a verification target.
**Action:** verify against championship sources and remove the incorrect row.

**Outcome:** 35 final rows, 486 evidence nodes, 80.1 Item F1.

**Blocked publisher pages and bounded recovery (`ws_en_023`).** This task enumerates migration-related articles from three journals over 2020–2024. The orchestrator first corrects an ambiguous journal name, then assigns one Search Agent per journal. After the initial pass, it removes an out-of-scope 2026 article and dispatches targeted metadata tasks for five incomplete rows. Wiley and JSTOR pages repeatedly return access checks or HTTP 402 responses. The affected agents pivot to Crossref, OpenAlex, and academic-paper skills; the Loop Sensor records two explicit strategy switches for one stalled agent. When the remaining sources and per-agent budgets are exhausted, the system preserves the unresolved cells instead of fabricating metadata. Despite these access failures, the run returns 57 rows with 96.3% coverage and 93.8 Item and Row F1.

Table 14: Process summary for the three trajectory cases. Scores are scaled by $\times 100$.

| Case | Primary behavior | Tasks | Evidence | Item F1 | Row F1 |
|---|---|---|---|---|---|
| `ws_en_022` | Parallel enrichment + final audit | 5 | 221 | 97.5 | 85.0 |
| `ws_en_016` | False-saturation repair + verification | 8 | 486 | 80.1 | 28.2 |
| `ws_en_023` | Source fallback + bounded completion | 11 | 302 | 93.8 | 93.8 |

**Cross-case observations.** The three traces expose complementary aspects of the system that are hidden by final-answer scores:

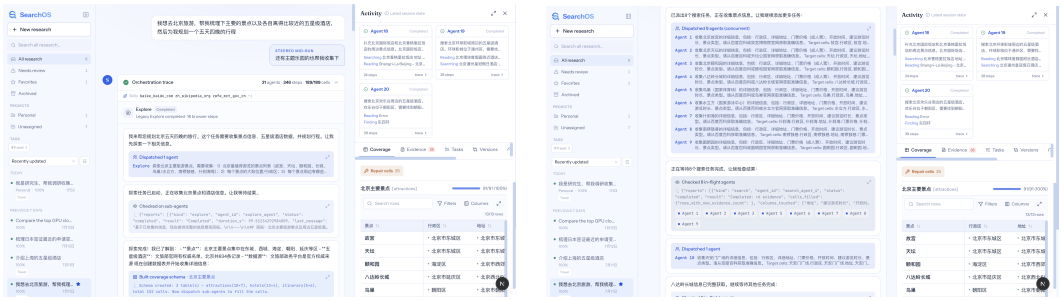
- **Explore before committing to row identities.** In the Spotify case, an authoritative list establishes the complete key space before metadata agents are launched. This prevents independently discovered song records from drifting across the two chart categories.
- **Cell saturation and scope completion are separate conditions.** In the Phelps case, 100% coverage over 16 known rows is correctly rejected as completion because the expected event count is much larger. Coverage controls attribute filling; an explicit row-set audit controls recall.
- **Recovery remains bounded and evidence preserving.** In the journal case, sensors encourage source and tool changes after repeated failures, while resource limits eventually stop unproductive search. Unresolved cells remain visible instead of being filled from unsupported inference.

Together, these cases show the intended control loop: exploration defines a defensible search space, parallel agents accumulate evidence into shared structured state, SOCM exposes concrete residual gaps, and governance redirects or stops work according to observable progress.

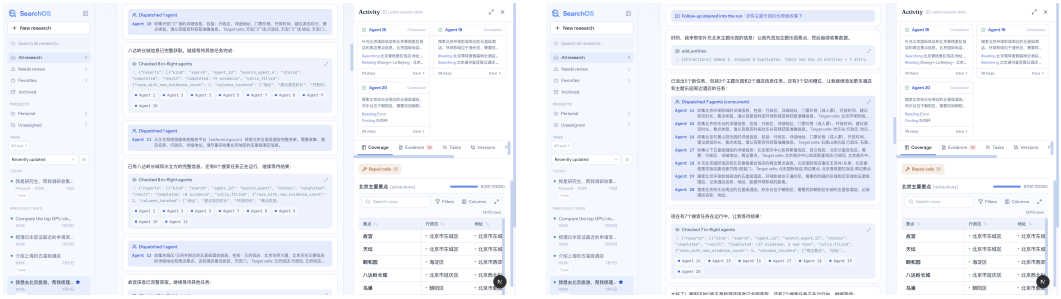
# I SEARCHOS CASE WALKTHROUGHS

The following interface snapshots are ordered by case number and then by step number. Each row contains two snapshots.

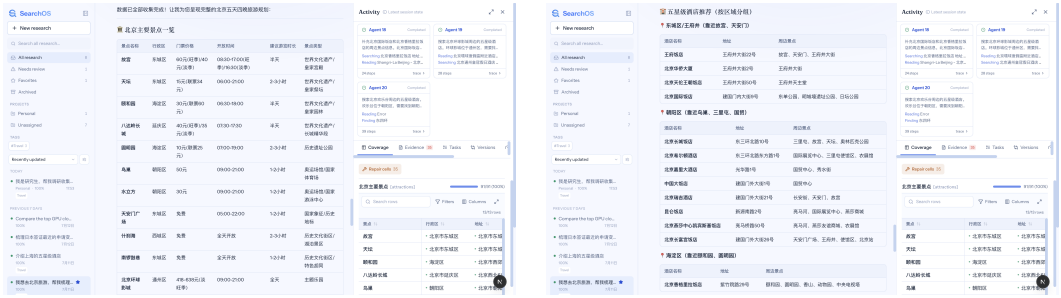
## I.1 CASE 1: BEIJING TRAVEL PLANNING



**Case 1-0** **Case 1-1**



**Case 1-2** **Case 1-3**



**Case 1-4** **Case 1-5**



Case 1-6

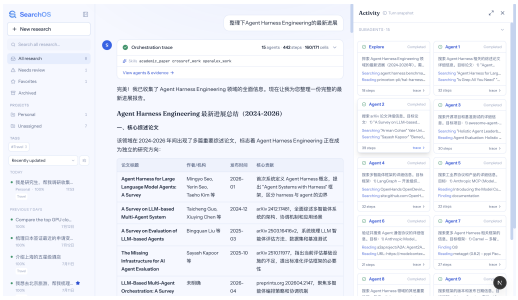


Case 1-7



Case 1-8

## I.2 CASE 2: AGENT HARNESS ENGINEERING SURVEY



Case 2-0



Case 2-1

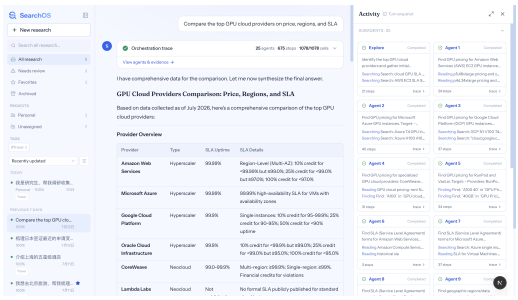


Case 2-2

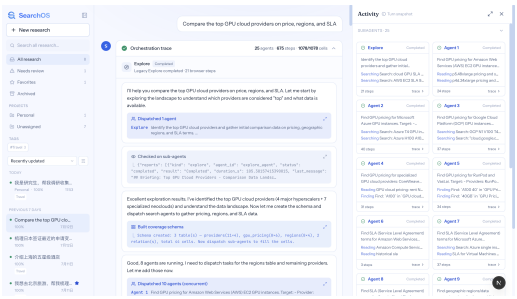


Case 2-3

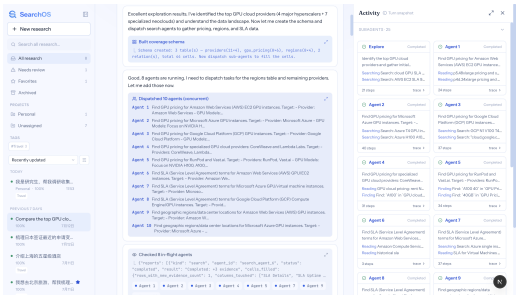
### I.3 CASE 3: GPU CLOUD PROVIDER COMPARISON



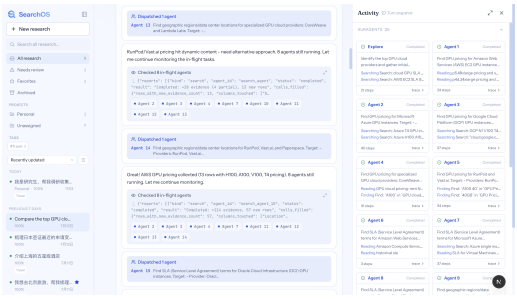
Case 3-0



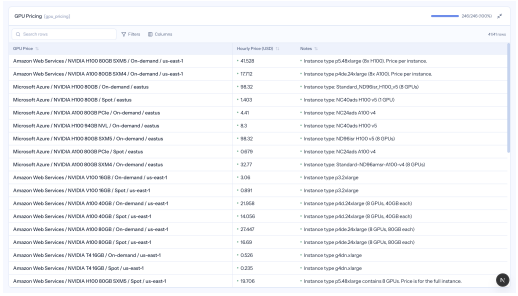
Case 3-1



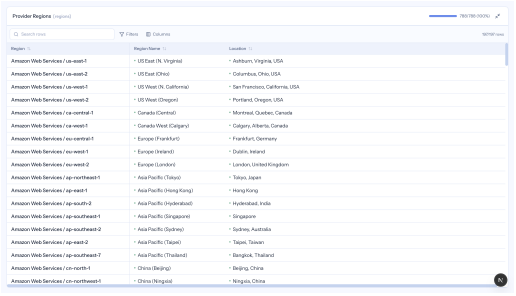
Case 3-2



Case 3-3



Case 3-4



Case 3-5