# EvoBrowseComp: Benchmarking Search Agents on Evolving Knowledge

**Yunhan Wang**[♠♣*], **Jiaan Wang**[♠†]
**Lianzhe Huang**[♠], **Xianfeng Zeng**[♠] and **Fandong Meng**[♠†]
[♠]Weixin AI, Tencent Inc, China [♣]Northeastern University, China
yunhannnan@gmail.com {torchwang,fandongmeng}@tencent.com

## Abstract

Search Agents—large language models augmented with search tools—have intensified the need for future-proof evaluation benchmarks. Existing benchmarks such as BrowseComp rely on static knowledge, making them vulnerable to test-set contamination and parametric memorization. Consequently, models can achieve high scores through fact recall rather than genuine retrieval, obscuring true browsing competence via reasoning shortcuts.

In this paper, we introduce EvoBrowseComp, an *evolving* benchmark of 400 English and 400 Chinese contamination-free complex questions synthesized via live-web traversal. To collect these questions, we design a three-agent collaborative framework: (1) a QA synthesis agent that retrieves *fresh* knowledge from the live web to synthesize QA pairs; (2) an information filtering agent that filters retrieved knowledge in terms of credibility and popularity to block parametric shortcuts; and (3) a high-level guidance agent that formalizes questions into reasoning graphs to reduce logical redundancy and shortcuts in synthesized QA pairs. Because the framework supports fully automated synthesis, EvoBrowseComp can be regularly updated to prevent data contamination and maintain temporal freshness. Extensive experiments confirm its great difficulty, requiring broad horizontal search. It establishes a scalable paradigm for auto-updatable, high-difficulty benchmarking that keeps pace with both evolving world knowledge and advancing agent capabilities.[1]

## 1 Introduction

Large Language Models (LLMs) augmented with web search tools, known as search agents (Wei et al., 2025; Chen et al., 2025; Zhou et al., 2025),

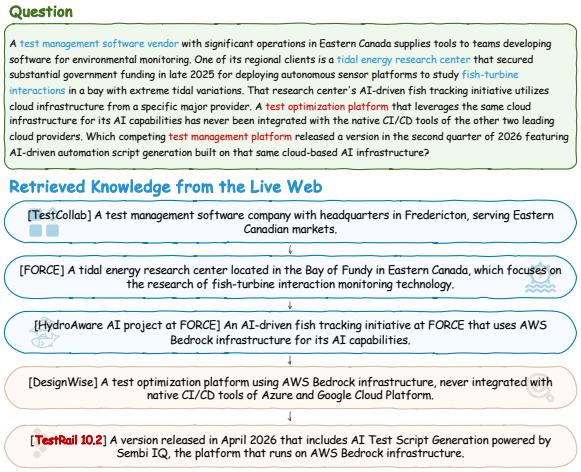



Figure 1: An illustrative example question from EvoBrowseComp. Orange highlights fresh knowledge (post-2026) lies in its reasoning paths, while red denotes its final answer.

have demonstrated remarkable performance on information-seeking tasks. These agents exercise *web browsing ability*—persistently navigating the open web, executing multi-hop questions, and gathering fragmented evidence across different sources (Wu et al., 2025; Gupta et al., 2026). To measure this ability, many benchmark datasets are proposed sequentially. BrowseComp (Wei et al., 2025) and BrowseComp-ZH (Zhou et al., 2025) focus on horizontal search, evaluating persistence and creativity in locating hard-to-find facts. GAIA (Mialon et al., 2024) tests general assistant competence through real-world, multi-step tool use. BFCL (Patil et al., 2025) assesses search orchestration via function calling, while WebWalker (Wu et al., 2025) isolates vertical traversal within structured websites. More recently, specialized benchmarks have targeted higher-order retrieval competencies: SealQA (Pham et al., 2025) probes robustness under noisy and conflicting retrieval conditions; while DeepSearchQA (Gupta et al., 2026) raises the bar by requiring exhaustive collation of

[*] Work was done when Yunhan Wang was interning at Weixin AI, Tencent Inc, China.
[†] Corresponding authors.
[1]We have released the data at https://hf.co/datasets/Krystalan/EvoBrowseComp



*Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 53176–53190
October 24-29, 2026 ©2026 Association for Computational Linguistics

answer sets across multiple sources. These efforts have established a rich landscape for benchmarking LLMs' web browsing ability.

However, existing benchmarks are typically anchored to *static* knowledge. For example, BrowseComp (Wei et al., 2025) and BrowseComp-ZH (Zhou et al., 2025) relies on question–answer pairs manually curated at a fixed point in time; BrowseComp-Plus (Chen et al., 2025) freezes a curated document snapshot to ensure reproducibility; GAIA (Mialon et al., 2024) grounds its tasks in specific and immutable versions of web pages or attached files; and DeepSearchQA (Gupta et al., 2026), though time-anchored, comprises a static prompt set evaluated against a fixed answer key. This static nature renders them acutely vulnerable to test-set contamination: as pre-training corpora expand, benchmark content inevitably leaks into model parameters, enabling models to solve questions via parametric memorization rather than genuine browsing and reasoning. As pointed out by Anthropic (2026a), the explicit leakage of BrowseComp answers into public data confirms that this benchmark has been compromised by data contamination.

To address these limitations, we introduce *EvoBrowseComp*, an evolving benchmark comprising 400 English and 400 Chinese complex questions automatically synthesized from live-web traversal. Our construction pipeline actively discovers and validates *fresh* knowledge and automatically constructs QA pairs through a three-agent collaborative framework. First, a *QA Synthesis Agent* retrieves fresh knowledge via web tools and provides (candidate) QA pairs based on the knowledge. Second, an *Information Filtering Agent* filters out retrieved knowledge in terms of credibility (verifying source credibility and cross-source consistency) and popularity (blocking parametric shortcuts through over-exposed knowledge). Third, a *High-level Guidance Agent* structures each question as a reasoning graph using three basic operations: projection, intersection, and complement. It identifies both structural redundancies and shortcuts, and directs the QA synthesis agent toward specific synthesis directions. Moreover, we adopt several strategies to ensure data quality, including the verification of textual quality, answer uniqueness, and question difficulty. In this manner, high-quality challenging questions involving fresh knowledge can be automatically collected (c.f., Figure 1). In contrast, prior benchmarks (Mialon et al., 2024; Wei et al., 2025; Zhou et al., 2025; Chen et al., 2025; Pham et al., 2025; Gupta et al., 2026) typically rely on labor-intensive human curation that makes regular updates prohibitively expensive. EvoBrowseComp removes this barrier: its synthesis pipeline is fully automated, requiring no costly manual annotation. This enables the benchmark to be refreshed continuously at minimal cost, swapping in newly emerged facts while retiring over-exposed ones.

Based on EvoBrowseComp, we evaluate various LLMs under both tool-based and tool-free settings. The results reveal two critical phenomena. First, even Claude-Opus-4.8 (Anthropic, 2026c), a cutting-edge reasoning LLM, achieves only 46.2% accuracy when equipped with tools, indicating that our temporally fresh, structurally complex questions are not easily retrieved. Second, when tool access is removed, Claude-Opus-4.8's performance drops to 9.0%, confirming that answering these questions demands genuine retrieval and multi-hop reasoning over fresh knowledge, rather than static recall. We believe this establishes a sustainable, contamination-resistant paradigm for future-proof evaluation of search agents.

In summary, our contributions are as follows:

- We introduce EvoBrowseComp, a search agent benchmark comprising 400 English and 400 Chinese complex questions. Grounding questions in fresh knowledge, it prevents models from exploiting parametric memorization.
- We propose a fully automated three-agent synthesis framework. It requires no costly human annotation, enabling continuous, low-cost regeneration that retires over-exposed questions and incorporates newly emerged facts and knowledge.
- Extensive evaluations demonstrate that even frontier LLMs achieve only modest accuracy ($<$50%) with web tools, and their performance collapses sharply when tool access is removed ($<$11%). This confirms that EvoBrowseComp effectively isolates genuine web browsing and multi-hop reasoning from static parametric recall.

## 2 EvoBrowseComp

EvoBrowseComp is built on two foundational principles. First, *questions should involve fresh knowledge*. By synthesizing questions from knowledge that emerges after training cutoffs, we prevent models from answering via parametric memorization. Second, *the construction pipeline should be fully automated and continuously evolvable*. This enables periodic regeneration in which over-exposed

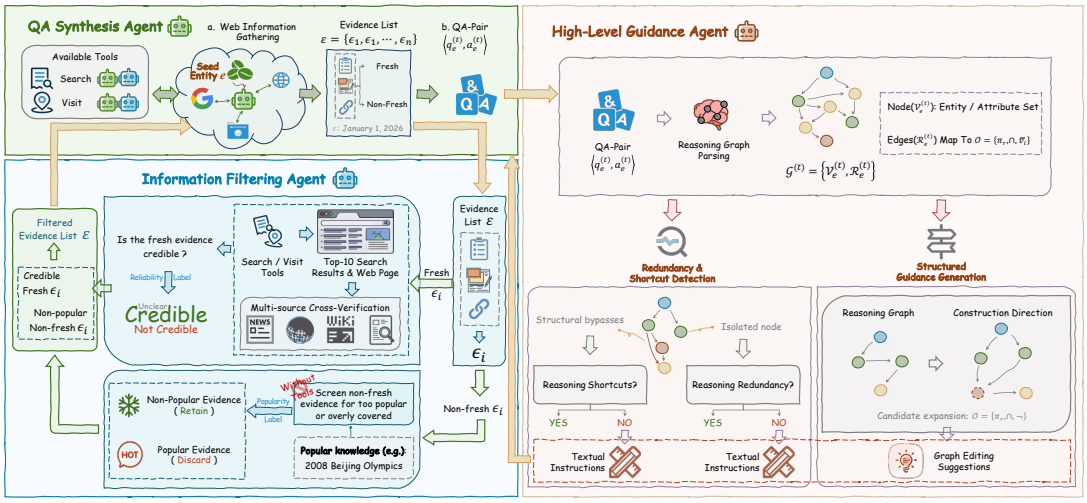



Figure 2: The illustration of the three-agent collaborative framework. (a) QA synthesis agent retrieves knowledge from the live web and generates (candidate) QA pairs; (b) Information filtering agent judges each retrieved knowledge in terms of credibility and popularity (popular/over-covered or not credible knowledge will be discarded); (c) High-level guidance agent detects the logical redundancy and shortcuts in the candidate QA pairs based on the constructed reasoning graphs, and gives suggestions to the QA synthesis agent in the next iteration.

questions are retired and replaced by newly surfaced knowledge, guaranteeing long-term benchmark validity without expensive human curation.

## 2.1 Data Collection

The data collection pipeline operates as an *iterative feedback loop* among three specialized agents (c.f., Figure 2). Beginning with seed entities, a QA synthesis agent searches the live web to propose candidate QA pairs with retrieved knowledge. Each retrieved knowledge is evaluated by an information filtering agent in terms of credibility and popularity. A high-level guidance agent formalizes the underlying reasoning structure of a candidate question generated in the $i$-th iteration, detects its logical redundancy and shortcuts, and guides the QA synthesis agent in the next iteration. In this way, the three agents collaborate automatically to synthesize highly complex, high-quality QA pairs.

**Seed Entity.** Synthesizing temporally fresh and logically complex QA pairs requires seed entities that tend to involve fresh knowledge. Rather than harvesting entities from a static knowledge graph—which risks stale facts—we collect seed entities through live-web retrieval. Specifically, we predefine 9 core domains (*e.g.*, science, economy and geography) and 50 fine-grained sub-domains. For each sub-domain, we equip an advanced LLM, *i.e.*, DeepSeek-V3.2 (Liu et al., 2025), with search tools to aggregate recently surfaced entities mentioned in high-coverage news or official websites w.r.t the sub-domain. This process yields about 50K seed entities, denoted as E. Illustration examples of seed entity collection are provided in Appendix A.1.

**QA Synthesis Agent.** For a given seed entity $e \in \mathrm{E}$, the QA synthesis agent iteratively mines information from the live web to construct a QA pair $\langle q, a \rangle$. The overall synthesis process can be formulated as a $m$-step iterative chain:

$$e \to \langle q_e^{(1)}, a_e^{(1)} \rangle \to \langle q_e^{(2)}, a_e^{(2)} \rangle \to \cdots \to \langle q_e^{(m)}, a_e^{(m)} \rangle \quad (1)$$

where $q_e^{(t)}, a_e^{(t)}$ denotes the question and its answer generated in $t$-th iteration, respectively. In detail, the agent involves two sub-steps in each iteration:

(1) *Web Information Gathering*: The agent collects information via engaging in multi-turn interactions with web tools: a *search* tool uses the Google search engine to retrieve information and a *visit* tool extracts targeted information from specific web pages. In the course of this multi-turn interaction, we encourage the agent to gather *fresh* knowledge, defined as information that becomes available after a specified timestamp $t$.[2] The agent then refines the gathered knowledge into an evidence list, denoted as $\mathcal{E} = \{\epsilon_1, \epsilon_2, ..., \epsilon_n\}$, where each $\epsilon_i$ indicates a concise knowledge statement (*e.g.*, entity $e_i$ has some specific attributes).

(2) *QA Pair Construction*: Leveraging the evidence

[2]In this paper, we set $t$ to January 1, 2026, and it can be trivially adjusted to other timestamps (*e.g.*, training cutoffs of specific LLMs).

list $\mathcal{E}$, the agent incorporates these pieces of evidence to synthesize a complex QA pair $\langle q_e^{(t)}, a_e^{(t)} \rangle$. Ideally, all evidence in $\mathcal{E}$ is fresh knowledge, ensuring that the synthesized $q_e^{(t)}$ is free from data contamination because it falls completely outside the search agents' parametric memorization. However, fresh knowledge appears much less frequently on the live web than its counterpart, namely non-fresh knowledge (*i.e.*, information already available before the timestamp $t$). Consequently, although we encourage the agent to gather fresh knowledge, $\mathcal{E}$ inevitably contains non-fresh knowledge. If we strictly require all evidence to be fresh knowledge, the scale of $\mathcal{E}$ will be too limited to synthesize a complex question. Therefore, we allow some non-fresh knowledge in $\mathcal{E}$, and require the agent to classify each $\epsilon_i$ as either fresh or non-fresh. To avoid overly covered answers induced by the non-fresh knowledge in $\mathcal{E}$, we limit the final answer to be based on fresh knowledge. In this way, a preliminary question $\hat{q}_e^{(t)}$ together with its answer $a_e^{(t)}$ is generated. To further enhance the difficulty of $\hat{q}_e^{(t)}$, wo follow Li et al. (2025); Lu et al. (2025) and obfuscate features and relationships within $\hat{q}_e^{(t)}$ (*e.g.*, vague time references and and non-specific descriptors) to obtain final question $q_e^{(t)}$. The prompts employed by the QA synthesis agent in these two sub-steps are presented in the Appendix A.2.

**Information Filtering Agent.** Directly using the gathered evidence list $\mathcal{E}$ to synthesize QA pairs might involve the following issues: (1) The evidence might suffer from rumor propagation, making the synthesized QA pairs unreliable. This issue is particularly pronounced in the context of fresh knowledge compared to non-fresh knowledge (Alkhodair et al., 2020). (2) Too popular or over-covered non-fresh knowledge in $\mathcal{E}$ will make the relevant reasoning in the QA pairs too predictable for search agents (Lu et al., 2025).

To deal with the above issues, we introduce an information filtering agent to filter out flawed $\epsilon_i \in \mathcal{E}$: (1) For each fresh $\epsilon_i$, the agent equipped with web tools (*i.e.*, search and visit) to cross-validate the reliability of $\epsilon_i$ on the live web, and finally output a reliability label, *i.e.*, "credible", "not credible", or "unclear". (2) For each non-fresh $\epsilon_j$, the agent directly judges whether it is too popular or overly covered, without using any tools. It will output a popularity label, *i.e.*, "popular" or "non-popular". Only credible fresh evidence and non-popular non-fresh evidence are retained in $\mathcal{E}$. If the length of $\mathcal{E}$ is less than a pre-defined threshold $k$, the QA synthesis agent retries the web information gathering process to collect additional evidence. The prompts used to obtain the reliability and popularity labels are provided in the Appendix A.3.

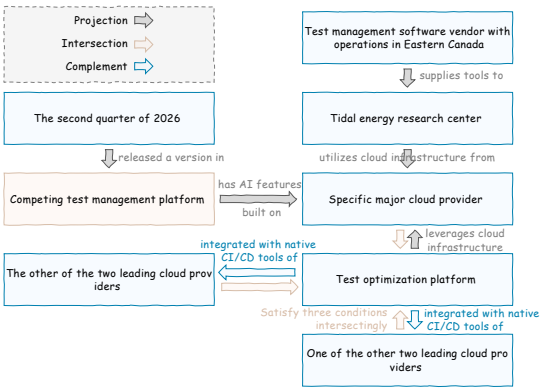



Figure 3: The illustration of a reasoning graph.

**High-Level Guidance Agent.** As pointed out by Tao et al. (2025), information-driven text-only synthesis paradigms struggle to capture underlying complex topologies (*e.g.*, entity–relation structures) and lack systematic control. As a result, the synthesized QA pairs tend to be *redundant* or *shortcut-prone* reasoning paths. In contrast, graphs offer a structured, semantically rich environment for multi-hop reasoning, enabling explicit control over reasoning paths (Lu et al., 2025).

To mitigate reasoning redundancy and shortcuts inherent in text-only synthesis paradigms and to provide high-level, explicit control over QA synthesis, we introduce a high-level guidance agent. This agent structures each synthesized question $q_e^{(t)}$ into a reasoning graph $\mathcal{G}_e^t = \{\mathcal{V}_e^t, \mathcal{R}_e^t\}$, where $\mathcal{V}_e^t$ and $\mathcal{R}_e^t$ denote the note set and the edge set in $\mathcal{G}_e^t$, respectively. Figure 3 shows an example of $\mathcal{G}_e^t$ using the example question in Figure 1. Each node $v_i \in \mathcal{V}_e^t$ indicates an object, *i.e.*, an entity (set) or an attribute (set). As for edges $\mathcal{R}_e^t$, we employ logical operations to capture the complex topologies within $\mathcal{V}_e^t$: (1) **Intersection** returns the intersection of $v_i$ and $v_j$, denoted as $v_i \cap v_j$. For example, the intersection of Turing Award laureates and females is the set of female Turing Award laureates. (2) **Complement** returns the complement of $v_i$, denoted as $\overline{v_i}$. For example, the complement of Turing Award laureates (with respect to all people) is the set of people who have not received the Turing Award. These two operations form a complete basis for ex-

pressing any entity set (Enderton, 2001). To further support relations and attributes, we introduce: (3) **Projection**, which projects $v_i$ to $v_j$ via a specific relation $r$, denoted as $v_j = \pi_r(v_i)$. For example, $\pi_{\text{winner}}(\text{Turing Award})$ denotes Turing Award laureates, while $\pi_{\text{gender}}(\text{Jackie Chan})$ denotes Jackie Chan's gender. Based on the above definition, each edge in the graph is denoted by one of the three operations.

After parsing $q_e^{(t)}$ into $\mathcal{G}_e^t$, we (1) detect reasoning redundancy by checking for isolated nodes or subgraphs in $\mathcal{G}_e^t$, and (2) detect reasoning shortcuts by examining whether there exist structural bypasses leading to the answer node. These detections can be achieved by an off-the-shelf toolkit, *i.e.*, NetworkX[3]. Further, using $\mathcal{G}_e^t$ together with the detected reasoning redundancies and shortcuts, the high-level guidance agent generates a textual instruction (denoted as $\mathcal{I}_e^t$) that specifies the synthesis direction (*e.g.*, which logical operations to add), and any redundancy or shortcuts should be avoided in the subsequent iteration. The instructions are fed into the QA synthesis agent and guide its behavior in the next iteration:

$$\left(\mathcal{I}_e^t, \langle q_e^{(t)}, a_e^{(t)} \rangle\right) \xrightarrow{\text{QA Synthesis Agent}} \langle q_e^{(t+1)}, a_e^{(t+1)} \rangle \quad (2)$$

The prompts of graph parsing and instruction generation are provided in the Appendix A.4.

**Iteration Termination.** The iteration of the three-agent collaborative framework is repeated until all of the following conditions are satisfied: (1) the synthesized question $q_e^{(t)}$ contains no redundancy or shortcuts; (2) at least five iterations have been executed; (3) the reasoning graph contains at least five edges, *i.e.*, $|\mathcal{R}_e^t| \geq 5$.

## 2.2 Data Quality

In the previous section, we mainly use the high-level guidance agent to control the data quality, *i.e.*, avoid reasoning redundancy and shortcuts in the synthesized questions. To further ensure the textual quality, uniqueness and difficulty of the collected QA pairs, we employ the following strategies:

**Textual Quality.** We evaluate whether each synthesized QA pair is fluent, clear, and unambiguous by using DeepSeek-V3.2 (DeepSeek-AI, 2025) as the judge model. Then, we filter out the low-quality QA pairs. The prompt is shown in Appendix B.

[3] https://github.com/networkx/networkx

| | Gave up after 2h | Solved within 2h |
|---|---|---|
| EvoBrowseComp-EN | 76.0% | 24.0% |
| EvoBrowseComp-ZH | 80.0% | 20.0% |
| Overall | 78.0% | 22.0% |

Table 1: Human baseline results on EvoBrowseComp under a 2h-per-question limit.

**Uniqueness and Difficulty.** To avoid alternative answers in the synthesized questions, we adopt a cross-validation method inspired by xbench-DeepSearch (Xbench-Team, 2025). In detail, for each question, we employ six cutting-edge LLMs[4] as search agents to answer the question three times independently[5], resulting in 18 solutions. If more than 80% of the solutions converge on the same incorrect answer, the question is treated as a multiple-answer question and is discarded. As for difficulty, if more than five (out of six) LLMs correctly answer the question, it will be discarded.

**Human Validation.** After quality filtering, we balance the data distribution for each domain and ultimately obtain 400 English and 400 Chinese QA pairs, which constitute our EvoBrowseComp. To further reveal the data quality, we conduct human analyses on randomly selected 400 QA samples (200 English and 200 Chinese). Since directly asking humans to answer questions in a web environment is overly challenging, we instead use the evidence list $\mathcal{E}$ as an anchor and require human evaluators to verify (1) whether each $\epsilon_i \in \mathcal{E}$ is correct, and (2) whether each synthesized question is consistent with its corresponding evidence list $\mathcal{E}$ and is unambiguous; (3) whether each answer can be inferred by the evidence list $\mathcal{E}$ (for more details, please refer to Appendix C.1). The results indicate that 94.2% of evidence lists are entirely correct, and 90.8% of questions are both consistent with their corresponding evidence lists and unambiguous.[6] 100.0% of answers can be inferred from $\mathcal{E}$. Overall, *87.8%* of QA pairs simultaneously pass the above three verifications, indicating the superiority of our synthesis framework.

Following BrowseComp (Wei et al., 2025), we further conduct a human baseline to reveal the difficulty of EvoBrowseComp. We recruit 6 evaluators and ask each to independently solve 100 QAs un-

[4] DeepSeek-V4, DeepSeek-V3.2, GLM-5, Kimi-K2.6, Qwen3.5-397B-A17B and Qwen3.5-122B-A10B.

[5] Temperature is set to 1.0; top_p is set to 0.95.

[6] The remaining 5.8% of evidence lists may involve hallucinations.

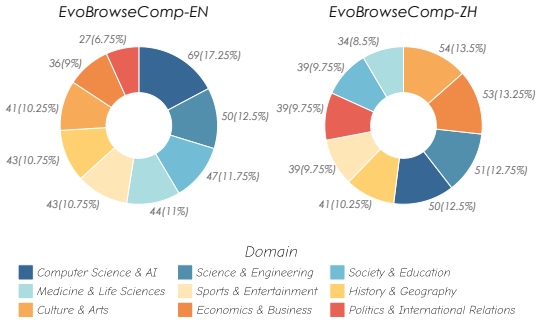



Figure 4: Distribution across nine domains in EvoBrowseComp.

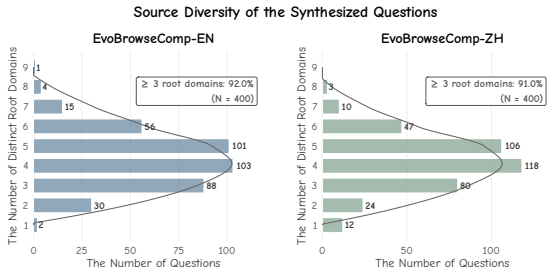



Figure 5: Distribution of the number of distinct root domains per question in EvoBrowseComp.

| Data | Scale | Lang. | Length | Node |
|---|---|---|---|---|
| BrowseComp (Wei et al., 2025) | 1266 | En | 128.86 | 8.85 |
| DeepSearchQA (Gupta et al., 2026) | 900 | En | 65.33 | 5.83 |
| EvoBrowseComp-EN (Our) | 400 | En | 142.48 | 8.62 |
| BrowseComp-ZH (Zhou et al., 2025) | 289 | Zh | 73.88 | 6.63 |
| WebWalkerQA (Wu et al., 2025) | 680 | Zh | 33.99 | 4.63 |
| EvoBrowseComp-ZH (Our) | 400 | Zh | 162.33 | 8.07 |

Table 2: Data Statistics of EvoBrowseComp compared with previous benchmark datasets (Lang.: Language). "Length" denotes the average question length, and "Node" denotes the average number of nodes in the reasoning graphs.

der a 2h-per-question limit, during which they may use a web browser but no AI assistant (details are provided in Appendix C.2). As shown in Table 1, human evaluators solve only 22.0% of the QAs and give up on 78.0% after 2h.

### 2.3 Data Statistics

EvoBrowseComp contains 800 high-quality QA pairs (400 in English and 400 in Chinese). We analyze the data statistics from the following aspects:

**Domain Distribution.** As shown in Figure 4, EvoBrowseComp is evenly distributed across nine predefined domains, ensuring broad coverage of the knowledge areas.

**Length.** We calculate the average question length of EvoBrowseComp and previous benchmark datasets. As shown in Table 2, The average question lengths in EvoBrowseComp are 142.48 and 162.33 tokens for English and Chinese, respectively, which are generally longer than those in previous datasets, particularly for Chinese.

**Complexity.** We use reasoning graphs to structure complex questions in Section 2.1. In addition to EvoBrowseComp, we also extract reasoning graphs for complex questions of previous datasets. The average number of nodes in reasoning graphs, which can reflect questions' complexity, is also reported in Table 2: The average number of nodes in EvoBrowComp-EN is similar to that in BrowseComp (8.62 vs. 8.85). On Chinese data, the average number of nodes in EvoBrowComp-Zh is significantly greater than others, *i.e.*, 8.07 vs. 6.63/4.63, indicating the complexity of our data.

**Source Diversity.** During collecting the evidence list $\mathcal{E}$ in data synthesis, we also record the source URL of each evidence $\epsilon_i \in \mathcal{E}$. To characterize the source diversity of the synthesized questions, we calculate, for each question, the number of distinct root domains present in the corresponding $\mathcal{E}$. For example, if the evidence for a given question is drawn from "a.com" and "b.com", there are two distinct root domains. Theoretically, the greater the number of distinct root domains, the more complex and difficult the question becomes. As shown in Figure 5, the number of distinct root domains per question in EvoBrowseComp exhibits a distinct bell-shaped distribution. On average, each question involves 4.2 distinct root domains, and over 90% of questions require reasoning across at least three independent sources. This makes cross-site evidence aggregation and verification a necessary prerequisite for answering.

### 2.4 Data Updation

We intend to update the benchmark data annually for the following two reasons: (1) Frequent updates are impractical due to the inherent scarcity of fresh knowledge (Section 2.1). (2) The high cost of data generation further renders more frequent updates unfeasible (Section 3.1).

Each annual update consists of two major steps: (1) Contamination Detection: We use the most advanced LLMs available at that time as search agents to answer each question without any web tools; questions that the majority can answer correctly are flagged as contaminated and removed. (2) New Question Generation: By adjusting the timestamp

$t$ and re-running the three-agent synthesis framework, we generate new complex questions from much fresher knowledge, keeping the benchmark up-to-date.

### 2.5 Evaluation Protocol

In model evaluation, search agents are required to answer the given complex questions under multi-turn interactions with the following two web tools: (1) *Search* uses the Google search engine to retrieve information. (2) *Visit* extracts targeted information from specific web pages. The definition and implementation of these tools are derived from Team et al. (2025).[7] Following previous benchmark datasets (Wei et al., 2025; Zhou et al., 2025), we also employ the LLM-as-a-judge paradigm in model evaluation. Specifically, a judge model is used to verify whether a model prediction is correct or not. The judge prompt is derived from Team et al. (2025); Xbench-Team (2025), and is provided in the Appendix D. We use GLM-5-Chat (Zeng et al., 2026) as the judge model since it shows a strong correlation with human judgments (please refer to the Appendix E). The evaluation metric is accuracy based on the model judgments.

## 3 Experiments

### 3.1 Experimental Setup

**LLMs.** Based on our EvoBrowseComp, we evaluate the following cutting-edge LLMs: Claude-Opus-4.8 (Anthropic, 2026c), Claude-Opus-4.6 (Anthropic, 2026b), Qwen3.5-397B-A17B-FP8[8], Qwen3.5-122B-A10B[9], Qwen3.5-35B-A3B[10], Qwen3.5-27B[11] (Qwen, 2026), Qwen3-235B-A22B-Thinking-2507[12] (Yang et al., 2025), DeepSeek-V4-Pro[13], DeepSeek-V4-Flash[14] (DeepSeek-AI, 2026), DeepSeek-V3.2[15] (DeepSeek-AI, 2025), GLM-5[16] (Zeng et al., 2026) and Kimi-K2.6[17] (Kimi, 2026).

**Implementation Details.** In data collection, all three agents are employed based on DeepSeek-V3.2 (DeepSeek-AI, 2025), which is self-hosted (16 NVIDIA H20 GPUs per deployment). The pre-defined threshold $k$ is set to 5 in the information filtering agent. The whole data construction takes $\sim$ 40K GPU hours, with $\sim$ 57K search and $\sim$ 64K visit tool calls. In model evaluation, all open-source LLMs are deployed on NVIDIA H20 GPUs (96G) using SGLang[18]. Most models are run on 8 GPUs, except DeepSeek-V4-Pro and GLM-5, which each use 32 GPUs, and DeepSeek-V3.2, which uses 16 GPUs. We adopt sampling-based decoding with a temperature of 0.6 and a top_p of 0.95. We set the maximum context length to 128K for all LLMs to ensure a fair comparison, and limit the maximum number of tool calls to 40. All LLMs are evaluated with the (maximum) thinking mode. After model prediction, we use GLM-5-Chat (zero temperature) as the judge model. For each LLM, we run three independent evaluations and report the average.

### 3.2 Results & Analyses

Table 3 presents the model performances on EvoBrowseComp, and we analyze the results from the following aspects:

**Tool-Free Setting.** Without tools, most LLMs' accuracy is below 5% (*e.g.*, Kimi-K2.6 only achieves 0% and 1.6% in English and Chinese, respectively), and even the best-performing LLM (Claude-Opus-4.8) only achieves 9.0% in English and 14.0% in Chinese. The limited tool-free performance suggests that EvoBrowseComp effectively prevents reliance on parameterized memory by introducing fresh knowledge in the complex questions.

**Tool-Based Setting.** With access to web tools, in English, Claude-Opus-4.8 ranks first with 46.2%, followed by Claude-Opus-4.6 (44.8%), and Qwen3.5-397B (42.0%). The results of the Chinese evaluation also show similar trends. Taking Qwen3.5-397B, GLM-5 and DeepSeek-V3.2 as example LLMs, we also compare their performance on EvoBrowseComp and previous BrowseComp(-

[7] https://github.com/Alibaba-NLP/DeepResearch
[8] https://huggingface.co/Qwen/Qwen3.5-397B-A17B-FP8
[9] https://huggingface.co/Qwen/Qwen3.5-122B-A10B
[10] https://huggingface.co/Qwen/Qwen3.5-35B-A3B
[11] https://huggingface.co/Qwen/Qwen3.5-27B
[12] https://huggingface.co/Qwen/Qwen3-235B-A22B-Thinking-2507
[13] https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro
[14] https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash
[15] https://huggingface.co/deepseek-ai/DeepSeek-V3.2
[16] https://huggingface.co/zai-org/GLM-5
[17] https://huggingface.co/moonshotai/Kimi-K2.6
[18] https://github.com/sgl-project/sglang

| Model | English | | Chinese | |
|---|---|---|---|---|
| | w/ tools | w/o tools | w/ tools | w/o tools |
| Claude-Opus-4.8 | **46.2** | **9.0** | **38.3** | **14.0** |
| Claude-Opus-4.6 | <u>44.8</u> | 6.0 | <u>36.8</u> | 8.8 |
| GLM-5 | 39.2 | 1.1 | 30.5 | 2.5 |
| Kimi-K2.6 | 31.0 | 0.0 | 19.8 | 1.6 |
| Qwen3.5-397B-A17B-FP8 | 42.0 | 2.9 | 34.5 | 3.1 |
| Qwen3.5-122B-A10B | 29.2 | 2.9 | 21.0 | 2.6 |
| Qwen3.5-35B-A3B | 29.2 | 0.8 | 14.2 | 1.2 |
| Qwen3.5-27B | 25.0 | 2.8 | 18.6 | 4.5 |
| Qwen3-235B-A22B-Thinking-2507 | 16.5 | 3.5 | 17.5 | 7.0 |
| DeepSeek-V4-Pro | 16.5 | 0.8 | 15.0 | 3.5 |
| DeepSeek-V4-Flash | 16.5 | 2.0 | 10.8 | 4.5 |
| DeepSeek-V3.2 | 23.0 | <u>6.3</u> | 30.5 | <u>10.3</u> |

Table 3: Experimental results on EvoBrowseComp. The **bold** and the <u>underline</u> denote the best and second-best performances, respectively.

| Model | BC | EvoBC.-EN | BC-ZH | EvoBC.-ZH |
|---|---|---|---|---|
| DeepSeek-V3.2 | 51.4 | 23.0 | 65.0 | 30.5 |
| GLM-5 | 62.0 | 39.2 | 72.7 | 30.5 |
| Qwen3.5-397B | 69.0 | 42.0 | 70.3 | 34.5 |

Table 4: The performance of three example LLMs on BrowseComp (BC), BrowseComp-ZH (BC-ZH) and EvoBrowseComp-EN/ZH (EvoBC.-EN/ZH).

| Model | English | | Chinese | |
|---|---|---|---|---|
| | Acc. | ER. | Acc. | ER. |
| DS-V4-Chat | 27.2 | 51.8 | 20.0 | 47.5 |
| DS-V4-High | 34.5 | 38.8 | 24.8 | 54.5 |
| DS-V4-Max | 16.5 | 75.5 | 10.8 | 82.5 |
| Claude-Opus-4.6 | 44.8 | 26.2 | 36.8 | 25.2 |

Table 5: The performance of DeepSeek-V4-Flash on EvoBrowseComp using different reasoning efforts. DS.: DeepSeek; ACC.: "accuracy"; ER.: "Exceed Ratio" indicates the proportion of evaluation samples that exceed the maximum allowed number of tool calls.

ZH) (Wei et al., 2025; Zhou et al., 2025).[19] As shown in Table 4, we find that the model performances on our data are significantly lower than those on BrowseComp / BrowseComp-ZH, indicating the difficulty of our complex questions.

**Scaling Laws in Different Tool Settings.** We also evaluate the performance of the Qwen3.5 family (27B → 35B-A3B → 122B-A10B → 397B-A17B), and identify the following scaling behaviors: When tools are utilized, accuracy improves with increased model capability (EN: 25.0 → 42.0; ZH: 18.6 → 34.5 overall). In contrast, without tools, accuracy remains consistently low, hovering around 1–5%, regardless of model scale. This indicates that no scaling law is evident, as the necessary episodic or fresh knowledge is not embedded within the parameters at any model size.

**The Effect of Reasoning Effort.** We also observe that DeepSeek-V4-Pro/Flash underperforms DeepSeek-V3.2. We manually inspect the predictions of DeepSeek-V4 and find that many evaluation samples exceed the maximum allowed number of tool calls (40; c.f., § 3.1), which leads to its unsatisfactory performance. To figure out the effect of reasoning effort, we use DeepSeek-V4-Flash as an example and evaluate its performance under three settings: the original max-reasoning setting, a high-reasoning setting (DS-V4-High), and a non-reasoning setting (DS-V4-Chat). In addition to accuracy, we also report the proportion of evaluation samples that exceed the maximum allowed number of tool calls (abbr. ER). As shown in Table 5, we find that DeepSeek-V4-Flash-High achieves the best performance among the three settings, while DeepSeek-V4-Flash-Max even underperforms DeepSeek-V4-Flash-Chat. In terms of ER, we find that DeepSeek-V4 achieves much high ER scores than Claude-Opus-4.6. For example, DeepSeek-V4-Max achieves 75.5% and 82.5% ER scores in English and Chinese. This phenomenon raises concerns about reasoning efficiency. Although state-of-the-art LLMs exhibit strong reasoning capabilities, enhancing their reasoning effi-

[19]The results on BrowseComp and BrowseComp-Zh are directly borrowed from Zeng et al. (2026); DeepSeek-AI (2025); Qwen (2026).

ciency remains crucial for practical applications.

## 4 Related Work

A growing body of work introduces benchmarks to evaluate the browsing, reasoning, and retrieval capabilities of LLMs. Early datasets such as NaturalQuestions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and HotpotQA (Yang et al., 2018) focus on single-hop or multi-hop fact retrieval; however, many of these datasets are effectively handled by cutting-edge LLMs (Mialon et al., 2024). To raise the difficulty ceiling, BrowseComp (Wei et al., 2025) introduces reverse-engineered questions requiring persistence and creativity in information seeking, while BrowseComp-Plus (Chen et al., 2025) provides a fixed, human-verified corpus to disentangle retriever performance from search agent reasoning. Parallel efforts such as BrowseComp-ZH (Zhou et al., 2025) extend BrowseComp to Chinese, and WebWalkerQA (Wu et al., 2025) emphasizes vertical web traversal through structured official websites. GAIA (Mialon et al., 2024) proposes conceptually simple yet execution-heavy tasks for general AI assistants, requiring diverse tool use and multi-step planning. DeepSearchQA (Gupta et al., 2026) shifts the evaluation focus from single-answer retrieval to exhaustive set generation, stressing systematic collation, entity resolution, and stopping criteria. More recently, SealQA (Pham et al., 2025) highlights the brittleness of search-augmented models under conflicting, noisy, or misleading retrieval results. Despite their respective strengths, these benchmarks largely rely on *static* or *fixed* corpora. As noted by Pham et al. (2025), this static nature renders them susceptible to progressive data contamination. Different from previous datasets, we design a three-agent framework to automatically discover *fresh* knowledge and construct contamination-free complex questions. Without costly manual annotation, our data can be regularly updated to prevent data contamination and ensure temporal freshness.

## 5 Conclusion

In this paper, we introduce EvoBrowseComp, an evolving search agent benchmark, which contains 400 English and 400 Chinese contamination-free complex QA pairs. We design a three-agent collaborative framework to discover fresh knowledge from the live web and synthesize the QA data. Multiple strategies are implemented to ensure data quality, including the avoidance of reasoning redundancy and shortcuts, as well as the verification of textual quality, answer uniqueness, and question difficulty. Human analyses further indicate that the synthesized data achieves a high level of quality. Since the framework operates entirely automatically and does not require costly manual annotation, it can be regularly updated to prevent data contamination and ensure temporal freshness. Furthermore, experimental results on cutting-edge LLMs underscore the challenges posed by this data. Thus, it establishes a sustainable paradigm for the future-proof evaluation of search agents.

## Limitations

While we show the effectiveness of the three-agent synthesis framework, there are some limitations worth noting: (1) We employ DeepSeek-V3.2 (DeepSeek-AI, 2025) as the backbone of the three agents, and thus, the synthesized data might involve the same biases and toxic behaviors exhibited by the model. (2) In the model evaluation, we use the judge model to assess only the final answers rather than the entire reasoning trajectory. Consequently, it becomes difficult to distinguish an agent that reasoned correctly from one that obtained the correct answer through inefficient or accidental means (*e.g.*, lucky guessing).

## Ethical Considerations

We discuss the main ethical considerations of EvoBrowseComp as follows: (1) Licenses. We will release our synthesized data under CC-BY-NC-SA 4.0 license. (2) Privacy Information. We extract knowledge from the publicly available web pages, and we filter out potential privacy information via LLMs.

## References

Sarah A Alkhodair, Steven HH Ding, Benjamin CM Fung, and Junqiang Liu. 2020. Detecting breaking news rumors of emerging topics in social media. *Information Processing & Management*, 57(2):102018.

Anthropic. 2026a. Eval awareness in claude opus 4.6's browsecomp performance.

Anthropic. 2026b. System card: Claude opus 4.6.

Anthropic. 2026c. System card: Claude opus 4.8.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama

Patel, Ruoxi Meng, Mingyi Su, et al. 2025. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*.

DeepSeek-AI. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *CoRR*, abs/2512.02556.

DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence.

H B Enderton. 2001. *A Mathematical Introduction to Logic*. Elsevier.

Nikita Gupta, Riju Chatterjee, Lukas Haas, Connie Tao, Andrew Wang, Chang Liu, Hidekazu Oiwa, Elena Gribovskaya, Jan Ackermann, John Blitzer, et al. 2026. Deepsearchqa: Bridging the comprehensiveness gap for deep research agents. *arXiv preprint arXiv:2601.20975*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Kimi. 2026. Kimi k2.6: Advancing open-source coding.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

K Li, Z Zhang, H Yin, et al. 2025. Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

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

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*.

OpenAI Team. 2025. OpenAI o3 and o4-mini system card.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Thinh Pham, Nguyen Nguyen, Pratibha Zunjare, Weiyuan Chen, Yu-Min Tseng, and Tu Vu. 2025. Sealqa: Raising the bar for reasoning in search-augmented language models. *arXiv preprint arXiv:2506.01062*.

Qwen. 2026. Qwen3.5: Towards native multimodal agents.

Z Tao, J Wu, W Yin, et al. 2025. Webshaper: Agentically data synthesizing via information-seeking formalization. *arXiv preprint arXiv:2507.15061*.

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

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

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. 2025. WebWalker: Benchmarking LLMs in web traversal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10290–10305, Vienna, Austria. Association for Computational Linguistics.

Xbench-Team. 2025. Xbench-deepsearch.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Jerrold H Zar. 2005. Spearman rank correlation. *Encyclopedia of biostatistics*, 7.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. 2026. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*.

Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, et al. 2025. Browsecomp-zh: Benchmarking web browsing ability of large language models in chinese. *arXiv preprint arXiv:2504.19314*.

# A Construction Details

## A.1 Examples of Seed Entity Construction

### Prompt for Seed Entity Construction

You are an expert in seed entity generation. Given a "primary domain : sub-domain" pair, your task is to aggregate entities that have recently surfaced (in 2026) under the specified sub-domain, as reported by authoritative news outlets or official websites.

**Input**

- Primary domain: `{primary_domain}`
- Sub-domain: `{secondary_domain}`
- Number of seeds to generate: `{num_seeds}`

**Tool**

- `search(query)`: performs a live web search and returns a list of webpage snippets.

**Requirements**

1. *Freshness*: Each entity must have either newly emerged in 2026 or undergone a significant attribute change in 2026 (e.g., newly founded organizations, newly released products, newly appointed individuals, newly occurred events).
2. *Source reliability*: Each entity must be reported by authoritative news sources (e.g., Reuters, BBC, Xinhua) or official websites (e.g., .gov sites, official homepages, reputable encyclopedias). Avoid personal blogs, self-media, and content farms.
3. *Diversity*: Cover a broad range of entity types, including people, organizations, events, products, concepts, and locations.
4. *Multi-round search*: Issue multiple `search()` calls with diversified queries to comprehensively cover the different facets of the sub-domain.

**Output Format**

```
{
  "domain": {
    "primary": "<primary domain>",
    "secondary": "<sub-domain>"},
  "total_count": {num_seeds},
  "seeds": [
    {"id": 1,
     "entity_name": "<entity name>",
     "is_fresh": true}
  ]
}
```

Now begin: first plan your search queries, then perform multiple rounds of `search()`, and finally output `{num_seeds}` seed entities.

## A.2 Examples of the QA Synthesis Agent

### Prompt for Evidence Collection

**Role: Knowledge Collection Agent**
Starting from the given seed, perform multiple rounds of `search` and `visit` to collect structured evidence that will support downstream QA construction.
**Definition of Fresh Knowledge**
Events, relations, attributes updates that occurred on or after January 1, 2026.
**Tools**

- `search(query)`: live web search.
- `visit(url)`: fetch the full content of a specific page.

**Strategy**

1. *Iterative exploration*: Starting from the seed, iteratively invoke search/visit, following hyperlinks to expand the entity network.
2. *Prioritize fresh knowledge*: Augment queries with time-aware keywords such as "2026", "latest", or "current".
3. *Mandatory verification*: Every piece of evidence must come from a page you have actually visited; do not record evidence based on search snippets alone.
4. *Chainability*: Evidence items should share entities with one another so that they can later be linked into evidence chains.

**Output Format**

```
[
  {
    "evidence_id": "e_001",
    "head_entity": "Entity A",
    "relation": "relation or
    ↪ attribute description",
    "tail_entity": "Entity B or
    ↪ attribute value",
    "evidence_text": "verbatim
    ↪ snippet from the source page",
    "source_url": "URL actually
    ↪ visited",
    "is_fresh_knowledge": true
  }
]
```

**Quantity Requirement**
Collect at least **{num_evidence}** evidence items, with no less than 50% marked as fresh knowledge.
`<seed>{seed}</seed>`

### Prompt for QA Construction

**Role: QA Construction Agent**
Based on the provided evidence list, construct **{num}** reasoning QA pairs.
`<evidence_list> {evidence_list} </evidence_list>`
**Hard Requirements**
**1. Reasoning Chain**

- *At least 5 hops*, with each hop introducing a fresh entity or relation (no looping in place).

- *The final hop must be fresh knowledge.* (`is_fresh_knowledge = true`)

**2. Answer**
The answer must fall into one of the following types to ensure uniqueness and evaluability:

- Deterministic attributes (time, number, or proper nouns with unique referents).
- The intersection, difference, or ranking result derived from multiple entities along the chain.

**3. Obfuscation (Core)**
*Goal*: prevent the model from identifying entities via memorization or a single search; the full reasoning chain must be traversed.

- *Non-fresh knowledge*: use descriptions whose individual words are common but whose combination points to a rare attribute.
- *Fresh knowledge*: anchor the description on a non-core but retrievable attribute, and phrase it with moderate vagueness.
- *Time / location*: replace explicit values with event anchors or rare attribute combinations.
- *The answer entity must not be directly or indirectly hinted at within the question.*

**4. Anti-Shortcut**
The final interrogative clause must be tightly bound to the last unresolved entity in the reasoning chain, and must not pivot to a public event or globally known attribute.
**Output Format**

```
[
  {
    "id": "qa_001",
    "question": "obfuscated question
    ↪ ",
    "answer": "final answer",
    "hop_count": 5,
    "knowledge_order": "nonfresh-
    ↪ nonfresh-fresh-fresh-fresh",
    "reasoning_chain": [
      {
        "hop": 1,
        "evidence_id": "e_xxx",
        "entity": "...",
        "relation": "...",
        "is_fresh_knowledge": false
      }
    ]
  }
]
```

Output the JSON array directly.

## A.3 Examples of Information Filtering Agent

### Prompt for Fresh Knowledge Reliability Assessment

**Role: Knowledge Reliability Verification Assistant**
Use the `search` and `visit` tools to **cross-validate** the given evidence list and assess its reliability.
**Evidence List to Verify**
Each item is a triple `(head_entity, relation, tail_entity)` supported by `evidence_text` from `source_url`, and flagged as fresh knowledge via `is_fresh_knowledge`.
`{evidence_list}`
**Tools**

- `search(query)`: live web search.
- `visit(url)`: fetch the full content of a specific page.

**Verification Procedure**

1. For each evidence item, extract the key fact represented by the triple `(head_entity, relation, tail_entity)`.
2. Issue `search` queries to retrieve relevant results, and `visit` authoritative sources (official websites, mainstream media, reputable encyclopedias) when further confirmation is needed. The provided `source_url` may be visited as one reference, but **must not** be counted as an independent corroboration.
3. *Cross-validation*: for each triple, find at least two independent sources that corroborate it.
4. Assign an overall reliability label for the entire evidence list:
   - `credible`: every triple is consistently supported by multiple independent and trustworthy sources.
   - `not credible`: at least one triple is contradicted by sources, or supported only by untrustworthy sources.
   - `unclear`: available information is insufficient to make a definitive judgment on one or more triples.

**Output Format**
Output only the following JSON. Do not add any additional text or code-block markers:

```
{
  "reliability": "credible | not
    ↪ credible | unclear"
}
```

### Prompt for Non-fresh Knowledge Popularity Assessment

**Role: Knowledge Assessment Assistant**
Based **solely on your internal knowledge**, judge whether the given evidence list is too popular or overly covered.
**Evidence List to Assess**
Each item is a triple `(head_entity, relation, tail_entity)` supported by `evidence_text` from `source_url`, and flagged as non-fresh knowledge via `is_fresh_knowledge`.
`{evidence_list}`
**Criteria**

- `popular`: the entities or facts described by the triples have prominent, widely known features, allowing you to identify the referents directly and unambiguously.
- `non-popular`: the descriptions are relatively

obscure or lack distinctive features, requiring additional clues to identify the referents.

**Instructions**

- Assess every triple in the list, then assign one overall label:
  - Label as `popular` only if the majority of triples clearly describe widely known entities or facts.
  - Otherwise, label as `non-popular`.
- If the content lies beyond your knowledge scope (e.g., involves future information or entirely unfamiliar concepts), label it as `non-popular`.

**Output Format**
Output only the following JSON. Do not add any additional text or code-block markers:

```
{
  "popularity": "popular | non-popular"
}
```

## A.4 Examples of High-level Guidance Agent

### Prompt for Question Graph Parsing

**Role: Question Graph Parsing Expert**
Parse the given (`question`, `answer`, `reasoning_chain`) into a strict Question Graph.
**Graph Definition**

- *Node*: a set of entities or attribute values.
- *Edge*: one of exactly three operations.
  - `projection` ($\pi_r$): traverse one step along relation $r$, either forward or backward.
  - `intersection` ($\cap$): take the intersection of multiple entity sets.
  - `complement` ($\neg$): take the complement of an entity set.

**Core Principles**

1. *Faithful to the literal semantics of the question*: the `reasoning_chain` serves only as a reference for entity and relation names; do not use it to forcibly connect subgraphs that have no logical dependency in the question itself.
2. *Local connectivity*: even when an isolated subgraph functions as mere "background filler," its internal edges must still be built according to the literal logic expressed in the question.

**Node Fields**

- `node_id`: starts from `n0`.
- `label`: semantic description of the node.
- `known_entities`: entities explicitly given in the question; otherwise `null`.
- `reference_entities`: real entity names not mentioned in the question but provided by the reasoning_chain; otherwise `null`.
- `is_root`: whether this node is the answer node.

**Input**

- `question`: {question}
- `answer`: {answer}
- `reasoning_chain`: {reasoning_chain}

**Output Format**
Output only the following JSON, with no additional text:

```
{
  "question": "...",
  "answer": "...",
  "root_node_id": "n_x",
  "nodes": [
    {
      "node_id": "n0",
      "label": "...",
      "known_entities": [...] or null
    ↪ ,
      "reference_entities": [...] or
    ↪ null,
      "is_root": false}
  ],
  "edges": [
    {
      "edge_id": "e0",
      "source_ids": ["n0"],
      "target_id": "n1",
      "op": "projection |
    ↪ intersection | complement",
      "relation": "...",
      "direction": "forward |
    ↪ backward"}
  ]
}
```

### Prompt for QA Synthesis Direction

**Role: QA Synthesis Direction**
Given the parsed Question Graph, propose **one** concrete expansion suggestion that makes the question more complex and adversarial.
**Input**

- `question`: {question}
- `answer`: {answer}
- `graph_json`: {graph_json}

**Allowed Operations** (only the following three edge types may be added)

- `projection`: traverse one step along a relation.
- `intersection`: add an additional condition that must be jointly satisfied.
- `complement`: add an exclusionary constraint.

**Candidate Strategies**

- *Strategy A (Complement)*: attach a "never / excluding ..." constraint to a node, producing a hard negative that retrieval systems struggle to handle.
- *Strategy B (Inverse Projection)*: flip a forward projection into its inverse (effect $\rightarrow$ cause), which is inherently harder to reason about than forward inference.

- *Strategy C (Intersection Expansion)*: add an intersection branch on an intermediate node. Placing it on an early node (left-deep tree) or on a node close to the answer (right-deep tree) modulates how the reasoning difficulty is distributed along the path.

**Output Format**
Output only the following JSON, with no additional text:

```
{
  "expansion_advice": {
    "target_text_span": "the exact
    ↪ text span of the target entity
    ↪  copied verbatim from the
    ↪ original question",

    "suggested_operation": "
    ↪ complement | projection |
    ↪ intersection",

    "selected_strategy": "Complement
    ↪ | Inverse Projection |
    ↪ Intersection Expansion",

    "reasoning": "why this operation
    ↪ and strategy are chosen, and
    ↪ how they increase the graph's
    ↪ complexity and reasoning
    ↪ difficulty",

    "semantic_suggestion": "a natural
    ↪ -language description of the
    ↪ concrete modification to apply
    ↪ "
  }
}
```

# B Low Quality QA pairs filtering

**Prompt for QA Data Quality Filtering**

**Role: QA Data Quality Filtering**
Inspect the given QA pair and output only "pass" or "fail".
**Input**

- `question`: {question}
- `answer`: {answer}

**Inspection Criteria** (failing any single item results in "fail")
**1. Completeness**

- Both question and answer are non-empty, non-whitespace, and free of truncation or garbled text.

**2. Question Language Quality**

- Fluent and natural, consistent with how a human would naturally ask, with no grammatical errors or signs of machine-generated patchwork.
- Sentence structure is clear, avoiding excessive nested modifiers.
- Semantically unique and unambiguous, with no unclear references, vague scope, or multiple valid interpretations.
- Forms a complete interrogative sentence with an explicit target of inquiry.

**3. Answer Quality**

- Concise and definite, easy to evaluate automatically (e.g., a specific entity, number, or date).
- Semantically unique, with no equally valid alternative answers.
- Free of ambiguous expressions or vague qualifiers.

**Output Format**
Output only the following JSON, with no additional text or code-block markers:

```
{
  "thinking": "brief reasoning",

  "result": "pass | fail"
}
```

# C Human Validation

## C.1 Human Analyses on Data Quality

Two data experts from the author team, proficient in both English and Chinese, are involved in the human analyses. For (1) whether each $\epsilon_i \in \mathcal{E}$ is correct: each expert verifies whether $\epsilon_i$ is consistent with or can be inferred from its source web pages. For (2) whether each synthesized question is consistent with its corresponding $\mathcal{E}$ and is unambiguous; and (3) whether each answer can be inferred by the corresponding $\mathcal{E}$, the data experts also give their judgments independently. In cases where the two experts have differing judgments on a specific issue, the final decision will be reached through discussion.

## C.2 Human Baseline Details

Six evaluators proficient in both English and Chinese all have NLP/LLM backgrounds, hold a Master's or PhD degree, and had no prior exposure to the QA samples in EvoBrowseComp. Two data experts judge the correctness of the answers: a QA is marked as "solved within 2h" if the answer is judged correct; otherwise the evaluator keeps working, and if the QA remains unsolved after 2h, it is marked as "gave up after 2h".

# D Prompt of LLM-as-a-judge

**Prompt for LLM-as-a-Judge**

You are a general artificial intelligence assistant. Based on the `[Correct Answer]` provided below, judge whether the `[Response]` to the `[Original Question]` below is correct.

`[Original Question]`: {question}

`[Correct Answer]`: {correct_answer}

`[Response]`: {response}

| Model | Spearman |
|---|---|
| GPT-4.1 | 0.830 |
| DeepSeek-V4-Flash-Chat | 0.812 |
| DeepSeek-V4-Flash-Max | 0.828 |
| DeepSeek-V3.2-Chat | 0.824 |
| DeepSeek-V3.2-Think | 0.803 |
| Kimi-K2.6-Chat | 0.730 |
| Kimi-K2.6-Think | 0.843 |
| GLM-5-Chat | **0.864** |
| GLM-5-Think | 0.847 |

Table 6: The Spearman correlation between LLM judge and human judge.

Your judgment must follow the following format and standards:

**Final Answer**: The final accurate answer extracted from the `[Response]`. If there is no clear final answer in the `[Response]`, fill in `'None'`.

**Explanation**: Explain why the `[Final Answer]` is correct or incorrect based on the `[Correct Answer]`. Only focus on whether there is a substantial difference between the `[Final Answer]` and the `[Correct Answer]`; do not comment on the background of the question, do not attempt to re-solve the problem, do not defend any answer different from the `[Correct Answer]`, and only focus on judging whether the answers are consistent.

**Conclusion**: If the `[Final Answer]` is consistent with the `[Correct Answer]` provided above, or within an acceptable small error range for numerical questions, fill in `'Correct'`; otherwise (i.e., in case of any inconsistency, ambiguity, inequivalence, or incorrect extracted answer), fill in `'Incorrect'`.

## E Selection of the Judge Model

To select a reliable judge model in model evaluation, we randomly select 800 model predictions from the main experiments (Section 3.1), and respectively use the following cutting-edge LLMs as the judge models: GPT-4.1 (OpenAI Team, 2025), DeepSeek-V4-Flash-Chat, DeepSeek-V4-Flash-Max (DeepSeek-AI, 2026), DeepSeek-V3.2-Chat, DeepSeek-V3.2-Think (DeepSeek-AI, 2025), Kimi-K2.6-Chat, Kimi-K2.6-Think (Kimi, 2026), GLM-5-Chat and GLM-5-Think (Zeng et al., 2026). To figure out their correlation with humans, we also manually annotate whether each model prediction is correct or not. Similar to the human analyses (Appendix C.1), two data experts participate in the human annotation process, independently making their assessments and subsequently discussing any discrepancies in their judgments. As shown in Table 6, GLM-5-Chat achieves the best correlation with human judgments in terms of Spearman correlation (Zar, 2005).

| Model | EN | | ZH | |
|---|---|---|---|---|
| | Search | Visit | Search | Visit |
| GLM-5 | 15.5 | 11.2 | 17.3 | 9.3 |
| Kimi-K2.6 | 16.6 | 2.9 | 19.5 | 2.0 |
| Qwen3.5-397B-A17B-FP8 | 10.7 | 4.8 | 10.1 | 4.3 |
| Qwen3.5-122B-A10B | 13.8 | 5.0 | 11.8 | 4.0 |
| Qwen3.5-35B | 21.4 | 7.8 | 24.6 | 8.2 |
| Qwen3.5-27B | 12.7 | 2.0 | 11.8 | 1.7 |
| Qwen3-235B-A22B-Thinking | 11.6 | 2.3 | 12.4 | 1.8 |
| DeepSeek-V3.2 | 23.8 | 8.9 | 24.7 | 4.9 |
| DeepSeek-V4-Flash | 24.2 | 12.8 | 30.6 | 6.9 |
| DeepSeek-V4-Pro | 24.8 | 11.0 | 29.2 | 6.4 |
| Claude-Opus-4.6 | 14.4 | 8.8 | 17.8 | 8.6 |

Table 7: Average number of tool calls per question for each model on EvoBrowseComp.

## F Average Number of Tool Calls

To further investigate whether EvoBrowseComp meaningfully evaluates multi-hop agentic search, we report the average number of tool calls per question for each model in Table 7, where "search" and "visit" denote the call counts for the search and visit tools, respectively. Across all models, solving a single question requires a substantial number of tool calls (*e.g.*, over 20 search calls for DeepSeek-V4 series), reflecting the multi-hop nature of EvoBrowseComp and confirming that it effectively evaluates agentic search capabilities.