Benchmarking Real-Time Question Answering via Executable Code Workflows

Wenjie Zhou[1], **Yuan Gao**[2], **Xin Zhou**[1], **Hao Fu**[1], **Zhongjian Miao**[1], **Wei Chen**[1], **Bo Chen**[2] and **Xiaobing Zhao**[2]
[1]Li Auto Inc.
[2]Minzu University of China
{zhouwenjie1, zhouxin12, miaozhongjian, chenwei10}@lixiang.com,
{ignitesun, nmzxb_cn}@163.com, haofuch@outlook.com, chenbomuc@muc.edu.cn

Abstract

Retrieving real-time information is a fundamental capability for search-integrated agents in real-world applications. However, existing benchmarks are predominantly static and therefore fail to capture the temporal dynamics of information and the continuously evolving nature of real-world knowledge. To address this limitation, we propose **RT-QA**, a dynamic evaluation framework that leverages executable code workflows to retrieve up-to-date answers at evaluation time. Specifically, we construct an agent-driven pipeline to automatically generate Python workflows for web crawling and DOM-based answer extraction. RT-QA spans 12 domains with 320 Chinese questions categorized into three difficulty levels. To ensure benchmark reliability, all final problem-workflow pairs undergo workflow execution verification and manual website validation. Extensive evaluations of state-of-the-art models reveal significant limitations in real-time adaptability: even the best-performing models achieve only around 46% accuracy. Our analysis highlights two primary failure modes: (1) **Lazy Retrieval**, where agents rely on search snippets instead of deeply scanning specific websites for information; and (2) **Temporal Confusion**, a cognitive error where agents retrieve a historical date (e.g., an event in 2024) and fail to re-anchor to the current time (2026) for subsequent reasoning. These findings suggest that future agents require not just better retrieval strategies, but robust temporal state management. The code and data are available at https://github.com/leaves-slient/RT-Bench.git.

1 Introduction

The capability to access and reason over real-time information is a fundamental requirement for general intelligence. In recent years, Large Language Models (LLMs) have evolved from static knowledge bases into dynamic, search-integrated agents. The emergence of commercial systems like SearchGPT [OpenAI, 2024], Perplexity [Perplexity, 2024], and the DeepResearch framework [Team, 2025] highlights this shift, demonstrating impressive performance in complex information-seeking tasks [Nakano *et al.*, 2021; Schick *et al.*, 2024] and effectively transitioning LLMs into comprehensive assistants.

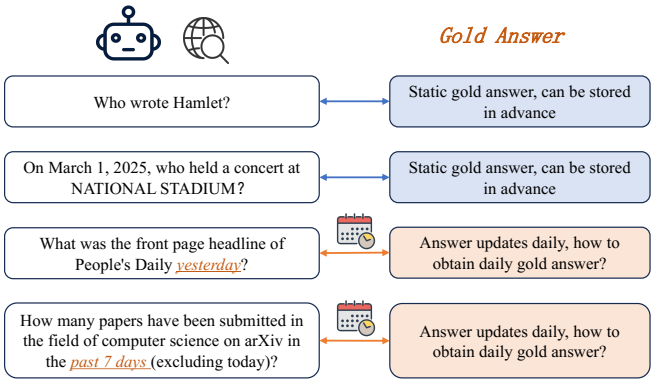

Figure 1: Comparison between Static QA and Real-time QA. Unlike static questions (top) where answers are fixed or can be pre-stored, real-time questions (bottom) typically involve relative temporal constraints (e.g., *"yesterday"*, *"past 7 days"*).

Despite this rapid development, evaluation methodologies for such agents remain limited. The majority of existing benchmarks rely on **static** question-answer pairs [Rajpurkar *et al.*, 2016; Kwiatkowski *et al.*, 2019; He *et al.*, 2024], suffering from two critical flaws: **Data Leakage** and **Knowledge Obsolescence**. The former enables agents to answer from memory instead of conducting necessary searches, while the latter renders the evaluation ground truth invalid as facts evolve.

To mitigate this, recent studies have introduced periodic benchmarks that update monthly or annually [Liska *et al.*, 2022; Vu *et al.*, 2023; Kasai *et al.*, 2023]. While some works propose automated pipelines [Meem *et al.*, 2024; Cheng and Dou, 2025], they remain limited by their dependency on pre-existing structured sources (e.g., Wikidata or specific APIs). Ideally, answering daily questions—such as tracking stock prices or arXiv submissions—requires querying official APIs. But such APIs are often scarce, paywalled, or simply do not exist. Consequently, existing benchmarks must compromise, covering only unchanged facts or a few API-supported domains. This leaves a critical gap: agents are rarely tested on high-frequency information from the open web, where no direct API exists.

In this paper, we introduce **RT-QA**, a dynamic benchmark designed to evaluate agents on rapidly changing, daily-updated information. The core innovation of RT-QA is the paradigm shift from Static Answers to **Executable Code Workflows**. Instead of a fixed answer string, each test item in RT-QA consists of a natural language question paired with a Python-based workflow. This workflow functions by visiting authoritative websites, parsing unstructured DOM elements, and generating the real-time ground truth at the exact moment of evaluation. To realize this, we propose the **RT Agent** framework, which leverages the tool-creation capabilities of LLMs to automatically generate these workflows and includes a self-repair mechanism to handle inevitable changes in web page structures. Finally we construct high-value questions across 12 domains (including Finance, Public Policy, and Science) that update daily, weekly, or monthly, minimizing manual effort.

We conducted extensive evaluations of state-of-the-art agents, including GPT-5.2, Claude-4.5, and GLM-4.7 [Team *et al.*, 2025], on RT-QA. The results reveal a significant gap: while these models excel on static tasks, they struggle with real-time inquiries. Even the leading models achieve an average accuracy of only 46%, with many other models falling below 40%. To understand the root causes, we conducted a large scale analysis of over 7,000 agent trajectories. Our taxonomy of failure modes reveals that the primary bottleneck remains **information gathering**: Retrieval-related issues (including Lazy Retrieval, Incomplete Scan, and Source Quality) account for 45% of all errors. Specifically, 20% of failures stem from Lazy Retrieval, where agents rely on search snippets without visiting the actual page.

Moreover, we uncover a recurring reasoning error: Temporal Confusion. We observe that agents frequently perform worse on intermediate reasoning tasks (Level 2) than on highly complex ones (Level 3). Our analysis suggests this stems from poor temporal state management—agents mistakenly use historical dates retrieved in intermediate steps as the "current time" for subsequent reasoning, whereas the higher complexity of L3 tasks forces a more rigorous isolation of temporal variables.

Our contributions are summarized as follows:

- We propose RT-QA, a benchmark using executable workflows to dynamically generate ground truth at inference time.
- We develop the **RT Agent**, an automated framework for generating and maintaining these workflows, allowing users to easily extend the benchmark.
- We provide a comprehensive evaluation and a fine-grained failure taxonomy. We highlight **Lazy Retrieval** and **Temporal Confusion** as the two dominant failure modes, offering critical insights for future agents to improve retrieval depth and temporal state management.

2 Related Work

2.1 From Static RAG to Real-Time Agents

While LLMs excel at reasoning, their internal knowledge is static. To mitigate this Knowledge Obsolescence, Retrieval-Augmented Generation (RAG) methods [Lewis *et al.*, 2020; Izacard *et al.*, 2022] connect models to external corpora. Recent works like TimeRAG [Wang *et al.*, 2025] and Time-Aware LMs [Dhingra *et al.*, 2022] further improved temporal grounding. However, standard RAG relies on pre-indexed documents, which inevitably suffer from indexing latency—they cannot answer questions about events that happened minutes ago. To overcome this, the field is shifting towards autonomous agents that interact directly with the live web. Early works like WebGPT [Nakano *et al.*, 2021] and WebArena [Zhou *et al.*, 2024] demonstrated how agents can use browsers as tools. More recently, advanced frameworks like DeepResearch [Team, 2025] have enabled agents to perform concurrent browsing and complex planning. Our work builds on this trend and focuses on evaluation: providing a rigorous benchmark to measure how well these agents actually perform in dynamic, real-world scenarios.

2.2 Benchmarks for Time-Sensitive QA

Evaluating temporal reasoning has evolved significantly. Early datasets like TempQuestions [Jia *et al.*, 2018] and Situational QA [Zhang and Choi, 2022] focused on static text. Recent dynamic benchmarks have pushed the boundary: StreamingQA [Liska *et al.*, 2022] and FreshLLMs [Vu *et al.*, 2023] introduce periodic updates (e.g., monthly) to track world events, while DailyQA [Cheng and Dou, 2025] leverages Wikipedia revision logs. However, a fundamental distinction exists between these approaches and RT-QA. Existing benchmarks operate on a Static Answers paradigm: data is collected first and released at fixed intervals. This introduces an unavoidable latency gap—they cannot evaluate queries about events happening today or right now. In contrast, RT-QA ensures nearly zero latency. This allows us to cover high-frequency web information (e.g., live stock prices, instant weather warnings) that periodic benchmarks inevitably miss.

2.3 Agent-Driven Tool Creation

Recent breakthroughs in LLMs, particularly with models like GPT-5.2 [OpenAI, 2025], Claude 4.5, and DeepSeek-V3.2 [Liu *et al.*, 2025], have redefined autonomous code generation. Beyond merely utilizing existing APIs [Schick *et al.*, 2024; Patil *et al.*, 2023], advanced agents can now synthesize complex executable logic to interact with dynamic environments. This trend is exemplified by autonomous software engineers like Devin [AI, 2024] and OpenDevin [Team, 2024], as well as generalist web agents like Voyager [Wang *et al.*, 2023] and Mind2Web [Deng *et al.*, 2023]. RT-QA leverages this tool-creation capability to revolutionize benchmark construction. Instead of relying on manual scripting or rigid templates, we implement an Agent-in-the-Loop pipeline. By harnessing the advanced reasoning and coding capabilities of these models, our RT Agent autonomously analyzes unstructured web pages and generates precise Python workflows. This approach allows us to scale the construction of dynamic benchmarks far beyond what was previously possible with manual effort.

3 RT-QA Benchmark

RT-QA is a dynamic benchmark designed to evaluate agents on rapidly changing, daily-updated information.

Question: `Are there any performances at the National Centre for the Performing Arts tomorrow?`

Python Workflow Code

```
# Import section omitted

async def get_tomorrow_performances():
    # Specify official link
    url = "https://www.chncpa.org/ycgp_220/ycap/"
    async with async_playwright() as p:
        browser = ...  # Some code about browser omitted
        today = datetime.now()
        tomorrow = today + timedelta(days=1)

        # 🔍 Find all clickable date links、search for performance information
        clickable_dates = await page.locator("td a, .day a, [class*='day'] a").all()
        tomorrow_performances = []
        tomorrow_date_str = f"{tomorrow.year}-{tomorrow.month:02d}-{tomorrow.day:02d}"

        for date_link in clickable_dates:
    # Iterate through all clickable dates to find tomorrow's performances; part of the code is omitted..…
            link_text = await date_link.locator("p").first.inner_text()
            href = await date_link.get_attribute("href")

            if href and tomorrow_date_str in href:
                tomorrow_performances.append({
                    'title': link_text.strip(),
                    'link': href,
                    'date': tomorrow_date_str})
            # Also check if the text contains tomorrow's date information
            elif f"{tomorrow.month:02d}.{tomorrow.day:02d}" in link_text or f"{tomorrow.month}.{tomorrow.day}" in link_text:

                tomorrow_performances.append({
                    'title': link_text.strip(),
                    'link': href or '',
                    'date': tomorrow_date_str})
        await browser.close()

        if tomorrow_performances:   # Generate answer
            answer = f"Tomorrow({tomorrow.year}-{tomorrow.month}-{tomorrow.day}),the National Centre for the Performing Arts has the following performances:\n\n"
            for i, performance in enumerate(tomorrow_performances, 1):
                answer += f"{i}. {performance['title']}\n"
        else:
            answer = f" Tomorrow ({tomorrow.year}-{tomorrow.month}-{tomorrow.day}), there are no performances scheduled at the National Centre for the Performing Arts."
        return answer

async def main():
    result = await get_tomorrow_performances()
    print(result)
    return result

if __name__ == ""__main__"":
    result = asyncio.run(main())
    print(f"Return results: {result}")
```

Answer `(Result of Code Execution October 31, 2025):`

Tomorrow (2025-11-1), there are no performances scheduled at the National Centre for the Performing Arts.

Answer `(Result of Code Execution November 1, 2025):`

Tomorrow (2025-11-1), the National Centre for the Performing Arts has the following performances:
1. Concert by Daniele Gatti and the Staatskapelle Dresden
2. "Yabin and Her Friends" Season 7: 10th Anniversary Performance of the Dance Drama The Moon Opera
3. Drum Tower West Theatre Production: Drama A Report to an Academy

Figure 2: An Example of an RT-QA Executed Workflow. The figure demonstrates how a single Python workflow handles dynamic time anchoring. The code (left) explicitly calculates the target date (tomorrow) based on the execution time. Consequently, executing the same workflow on different dates (Oct 31 vs. Nov 1) yields correct, context-aware answers (bottom right), functioning as a dynamic pseudo-API.

3.1 Problem Formulation

Standard QA benchmarks represent a test sample as a static pair (q, a). However, this formulation is insufficient for real-time scenarios where the ground truth is time-dependent. In RT-QA, we redefine a test item as a tuple (q_τ, w_τ), where:

- q_τ is a natural language question anchored to a relative time constraint (e.g., "yesterday", "past 7 days").
- w_τ is an **Executable Code Workflow** serving as a dynamic retrieval script. It encapsulates the logic to visit a target URL and extract specific information by parsing raw DOM elements.

The ground truth answer a_t at any evaluation timestamp t is obtained dynamically:

$$a_t = \text{Exec}(w_\tau, t) \tag{1}$$

As illustrated in Figure 2, the execution function $\text{Exec}(\cdot)$ three key steps: (1) Temporal Resolution: converting relative terms in q_τ to absolute dates based on t; (2) Live Retrieval: navigating to the target website via a headless browser; and (3) DOM Parsing: locating and computing the target elements to produce a deterministic answer.

3.2 Real-Time Dataset Construction Pipeline

Constructing robust executable workflows needs to process complex, dynamic DOM structures. To automate this, we propose an Agent-in-the-Loop pipeline (Figure 3a). We implement the RT Agent (powered by Claude 4.5), equipped with the specific toolset listed in Step 1 of the pipeline. The process operates in three specific steps:

Step 1: Preparation Work. We curate a diverse set of authoritative websites across 12 domains (e.g., National Bureau of Statistics, Stock Exchanges) that update frequently. Simultaneously, we define the *Available Tools* (e.g., `Check_html_content`, `Code_interpreter`) that will be provided to the agent in the subsequent step.

Step 2: Agent-in-the-Loop. Given a target URL, the RT Agent autonomously generates both the question and the corresponding workflow through an iterative process:

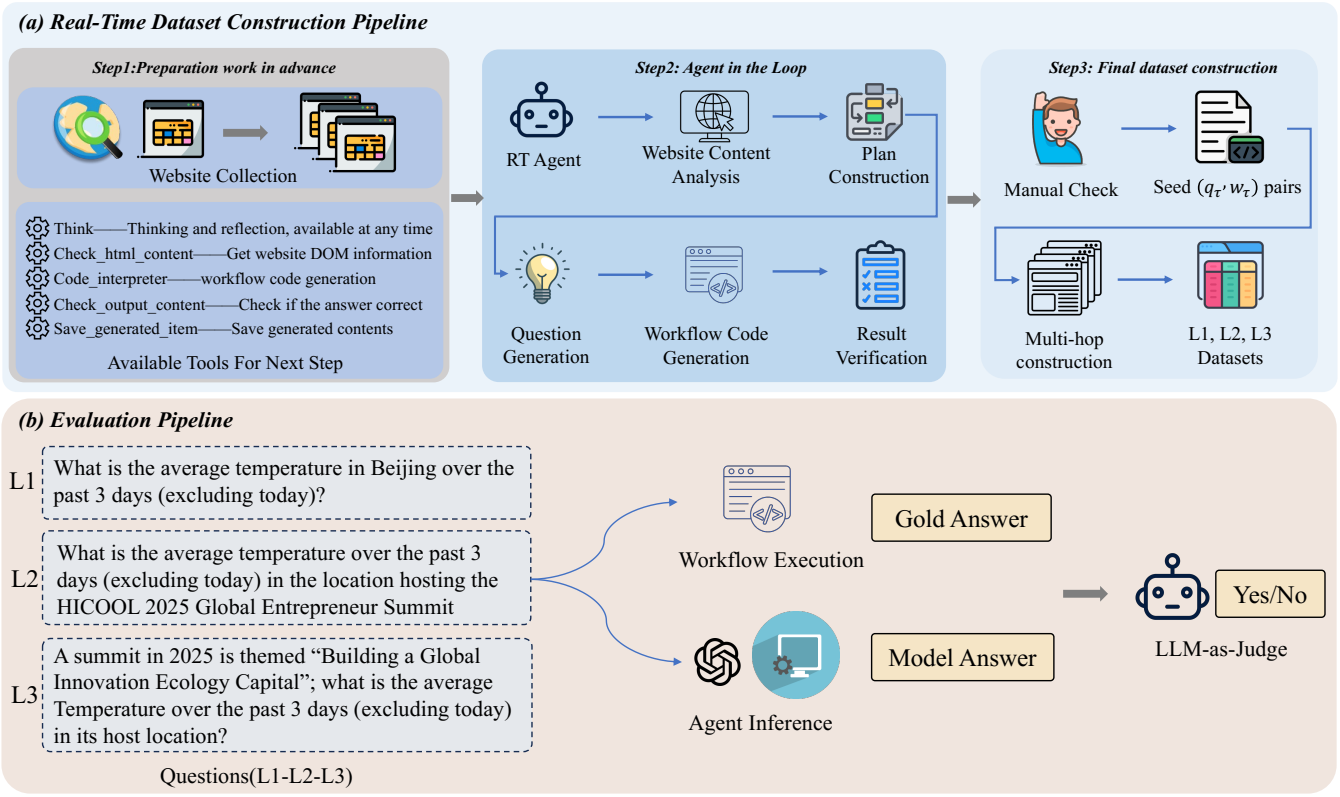

Figure 3: Overview of the RT-QA Framework. (a) The Construction Pipeline consists of three steps: preparation and tool definition, Agent-in-the-Loop workflow generation, and final dataset construction. (b) The Evaluation Pipeline, where models and workflows are executed simultaneously to assess real-time accuracy.

- *Exploration & Coding:* The agent utilizes `Check_html_content` tool (backed by Playwright) to fetch the raw DOM tree and identify dynamic elements. It then uses the `Think` tool to plan a strategy and writes Python scripts via the `Code_interpreter` tool. We enforce strict coding standards that prohibit silent failure (e.g., broad try-except blocks), ensuring that runtime errors are propagated back to the agent for debugging and self-repair.
- *Result Verification:* Once a candidate code workflow is generated, the agent invokes `Check_output_content`. This tool executes the generated code to retrieve the text answer while simultaneously capturing a page screenshot. The agent then cross-validates the text against the visual evidence to ensure alignment with human perception.

Step 3: Final Dataset Construction. The output from Step 2 is a set of candidate pairs (q_τ, w_τ). We then apply a rigorous two-phase process to finalize the dataset:

- *Phase 1: Human Validation.* We conduct a comprehensive manual review on the candidate pairs. Annotators verify that the generated questions (q_τ) are fluent and unambiguous, and that the workflows (w_τ) consistently retrieve the correct information. Verified pairs serve as the L1 (Direct) seed dataset.
- *Phase 2: Multi-hop Expansion.* To construct L2 and L3 datasets, we manually expand these L1 seeds based on the reasoning logic defined in Table 1. For example, an L1 question about "Beijing's temperature" is expanded into an L2 question by replacing "Beijing" with an entity description (e.g., "the host city of the 2025 HICOOL Summit"). By reusing the verified L1 workflow for L2 and L3, we ensure the ground truth remains consistent, regardless of how complex the question becomes.

During construction, the RT Agent successfully returned generated workflows for 71% of attempted cases, and 96.3% of these returned workflows passed manual validation. All final problem-workflow pairs were manually reviewed before formal evaluation while others were revised or discarded. The resulting benchmark comprises a total of **320 test samples** spanning 12 domains, categorized into **L1 (Direct,** $N = 154$**)**, **L2 (1-Hop,** $N = 109$**)**, and **L3 (2-Hop+,** $N = 59$**)**. This design tests whether an agent can switch between retrieving historical facts to identify the target entity and fetching real-time data to answer the core question.

Human Effort and Cost. As we mentioned before, RT-QA is not fully automatic and final human verification is required, the agent-assisted construction pipeline substantially reduces manual effort. Without the RT Agent, manually designing and maintaining one real-time QA workflow takes approximately 4 hours per case. With the proposed pipeline, the required human time is reduced to about 0.05 hours per case, mainly

Panel A: Domain Distribution (12 Categories)

Domain	#	Domain	#
Finance & Economy	42	Public Safety	34
Weather & Env.	40	News & Media	33
Culture Activities	35	Space & Astronomy	28
Sports Events	28	Energy & Industry	20
Consumer Products	20	Transportation	18
Sci & Tech	14	Policy & Gov.	10

Panel B: Reasoning Complexity & Logic

Level	Hops	Example Question (Translated) & Logic
L1	**0-Hop**	"What is the average temperature in **Beijing** over the past 3 days (excluding today)?" $\hookrightarrow$ *Logic: Locate Beijing* $\rightarrow$ *Search.*
L2	**1-Hop**	"What is the avg. temp. over the past 3 days (excluding today) in the location hosting the **HICOOL 2025 Global Entrepreneur Summit**?" $\hookrightarrow$ *Logic: HICOOL 2025 Summit* $\xrightarrow{Map}$ *Beijing* $\rightarrow$ *Search.*
L3	**2-Hop+**	"A summit in 2025 is themed **'Building a Global Innovation Ecology Capital'**; what is the avg. temp. over the past 3 days (excluding today) in its host location?" $\hookrightarrow$ *Logic: Theme* $\xrightarrow{Map}$ *HICOOL 2025* $\xrightarrow{Map}$ *Beijing* $\rightarrow$ *Search.*

Table 1: **RT-QA Statistics.** Top: Domain distribution. Bottom: Difficulty definitions based on reasoning hops. *Note: Example questions are translated from Chinese to demonstrate the reasoning complexity.*

for final verification. The token cost is approximately 160k tokens per case.

3.3 Maintenance & Extensibility

Automated Self-Repair. Although all workflows are verified before inclusion, website layouts may change over time, causing previously valid selectors or extraction logic to fail. To ensure long-term stability, we deploy a Repair Agent with a similar architecture to the RT Agent. When a workflow fails during execution, for example by throwing an exception, returning an empty result, or producing an obviously invalid answer, the failed case is sent to the Repair Agent. The Repair Agent re-analyzes the live DOM of the target website, identifies structural changes such as modified class names or shifted content blocks, and patches the Python workflow accordingly.

During the formal evaluation period, about 5% of workflows required repair due to website-side changes. For the observed repair cases, the Repair Agent successfully restored execution in all cases after re-analyzing the live page and updating the extraction logic. This maintenance burden is acceptable in our process.

Extensibility as a Service. Beyond the initial 320 questions, RT-QA serves as an open framework. Researchers can simply provide a target URL and a topic of interest; our system will automatically synthesize questions and verify the corresponding workflows. This allows the community to easily expand RT-QA into specific domains without manual coding.

4 Experiments

4.1 Experimental Setup

Agent Framework for evaluation. Evaluating agents via official web interfaces (e.g., ChatGPT-Web) presents two challenges: lack of transparency (intermediate steps are invisible) and scalability issues (manual testing is labor-intensive). Therefore, to ensure reproducibility and fine-grained analysis, we standardize our evaluation using an controllable open-source agent framework (adapted from DeepResearch [Team, 2025]) powered by the respective model APIs.To tailor the framework for real-time QA, we introduced two key modifications compared to the original implementation:

(1) Native Function Calling & Toolset: The original framework relies on XML-based prompting for tool use. We replaced this with native function calling interfaces to enhance compatibility and robustness. Under this protocol, agents are equipped with two main tools:

- `Search`: Powered by Serper API, retrieving top-10 results. It includes an automatic locale routing strategy (e.g., mapping Chinese queries to `cn`) to ensure linguistic relevance.
- `Visit`: Uses the Jina Reader API to fetch page content. To handle context limits, we implement a Goal-Driven Extraction mechanism where the model filters raw content to extract only evidence relevant to the current browsing goal.

(2) Temporal Injection: We inject the precise current timestamp into the system prompt. This enforces a "Real-Time Protocol," instructing agents to convert relative expressions (e.g., *yesterday*) into absolute dates before tool execution.

Prompt Optimization To examine whether performance can be substantially improved by prompt engineering alone, we use Qwen3-235B as a representative backbone and keep the agent framework, tools, decoding setting, and judge unchanged while varying only the system prompt. Starting from the original DeepResearch prompt, accuracy improves from 16.0% to 24.6% after emphasizing real-time awareness, further to 28.2% after requiring time anchoring and relative-time resolution, and finally to 30.6% after explicitly distinguishing retrieved historical dates from the current evaluation time. These gains confirm the importance of temporal instructions, but the diminishing returns suggest that the remaining bottleneck lies beyond prompting and requires deeper retrieval, stronger evidence verification, and explicit temporal state management.

Evaluation Setup. We evaluated 8 state-of-the-art models (as of Jan 2026), including proprietary ones (GPT-5.2, Gemini-3-Pro, Claude-4.5, Doubao-Seed) and open-weights models (DeepSeek-V3.2, GLM-4.7, Qwen3, Kimi-K2), all set to temperature temperature $T = 0.6$. Experiments were conducted over a 6-day period from January 10 to January 15, 2026. As shown in the Evaluation Pipeline (Figure 3b), for each question, the ground truth a_t is generated by executing the workflow w_τ at the exact same timestamp as the model inference. To assess correctness, we adopt the LLM-as-a-Judge paradigm [Zheng *et al.*, 2023]. We utilize GPT-5.2 as the evaluator to assess semantic equivalence between the model's response and the dynamic ground truth. *(Note: GPT-5.2 was selected due to its high correlation with human annotations in*

Model	Daily Overall Trend (Timeline)						Level Performance			Total Avg
	Jan 10	Jan 11	Jan 12	Jan 13	Jan 14	Jan 15	L1	L2	L3	
Open-Weights Models										
GLM-4.7	43.6	46.0	46.1	48.4	43.8	43.1	**46.9**	41.9	46.9	**45.2**
DeepSeek-V3.2	30.1	41.6	33.9	34.4	34.2	36.3	34.2	34.0	**39.3**	35.1
Qwen3-235B	24.5	29.9	34.2	25.6	34.5	35.6	**33.0**	26.9	31.9	30.7
Kimi-K2	37.6	41.5	40.4	40.6	41.0	41.6	**43.0**	37.4	39.6	40.4
Proprietary Models										
Doubao-Seed-1.8	46.9	47.4	49.4	45.6	43.2	46.3	44.3	45.6	**53.7**	**46.5**
GPT-5.2	38.2	42.2	44.7	38.4	36.6	45.9	**43.2**	38.0	41.0	41.0
Gemini-3-Pro	34.8	36.6	38.1	37.0	39.1	38.4	**41.0**	33.2	35.4	37.3
Claude-4.5	42.9	43.4	48.1	42.2	40.7	40.9	44.3	39.2	**46.9**	43.0

Table 2: **Main Results on RT-QA (Jan 10–15, 2026).** The table reports the **Accuracy (%)** evaluated by GPT-5.2 Judge. The left section shows the daily trend. The right section breaks down average performance by different level. **Total Avg** represents the aggregated accuracy across the entire 6-day period for all questions.

our validation. In a human agreement check on Qwen3-235B's responses to 109 questions, three independent GPT-5.2 judging runs agreed with human annotations on 109/109, 109/109, and 108/109 cases, respectively.)

4.2 Main Results

Table 2 presents the comprehensive performance of 8 leading models over a continuous 6-day evaluation period (Jan 10–15, 2026).

Low Overall Accuracy in Real-Time Settings. Contrary to their high performance on static benchmarks, models struggle significantly with real-time information. Even the leading models (Doubao-Seed and GLM-4.7) achieve an average accuracy of only 46%, while most open-weights models fall below 40%. This confirms that current agents still lack reliable information-seeking capabilities for the open web.

High Volatility Caused by Lazy Retrieval. Model performance fluctuates drastically day-to-day. For instance, DeepSeek-V3.2 varied by over 11 points (30.1% to 41.6%) within 24 hours. Our analysis links this instability to *Lazy Retrieval*. Agents often rely on search snippets rather than navigating to authoritative websites for step-by-step verification. Consequently, their success depends largely on whether the search engine surfaces a direct answer or relevant article in its immediate results, leading to inconsistent results.

This underscores the critical importance of *genuine deep search*, where agents must actively browse and verify source content instead of passively consuming summaries.

Harder Tasks Perform Better. We initially expected performance to degrade linearly with difficulty ($L1 > L2 > L3$). However, the results exhibit a **U-shaped pattern**, where intermediate-level (L2) tasks often yield the lowest accuracy. For instance, GPT-5.2 achieves 41.0% accuracy on the most complex L3 tasks, yet drops to 38.0% on the ostensibly simpler L2 tasks. A typical example is shown in Figure 4.

- **L2 Failure (Temporal Confusion):** L2 tasks typically consist of two major phases: first, **identifying the target entity** associated with a past event, and second, **retrieving information** regarding this entity's status at the current query time. The first phase introduces a strong historical date (e.g., "Oct 2025"). We observe that agents often get "stuck" on this retrieved date, failing to shift their focus back to the current date ("Jan 2026") when addressing the second phase.
- **L3 Recovery (Rigorous Planning):** L3 tasks require a more intricate reasoning chain involving multiple entities and timeframes. This complexity forces the agent to engage in more rigorous planning, explicitly distinguishing between historical event times and the current evaluation time within its working memory.

This contrast highlights that *planning capability* is decisive—complexity can sometimes force a model to plan better, acting as a safeguard against simple retrieval errors.

To verify that L2 failures stem from a lack of explicit temporal planning (rather than inherent difficulty), we conducted an additional experiment. We injected a strict Time Anchor instruction into the system prompt, explicitly warning agents to prioritize the current timestamp over retrieved historical dates. As shown in Table 3, the results are illuminating. With the Time Anchor, L2 accuracy rises to 40.4%, surpassing L3 (37.3%). This reversal confirms our hypothesis: L2 tasks are not intrinsically harder than L3; rather, they are cognitively deceptive. Without explicit guidance, agents fail to plan their temporal alignment, but once alerted (via prompt injection) or forced to strategize (via L3 complexity), they can navigate the timeline correctly.

4.3 Failure Mode Analysis

To better understand the root causes of low performance, we conducted a large scale analysis of over 7000 error trajectories. As illustrated in **Figure 5**, we identify seven distinct failure modes ranging from retrieval strategy to reasoning logic.

Retrieval & Scanning Issues (45%). The most prevalent failures stem from inadequate information gathering. Lazy

Ground Truth Context (Eval Time: *Jan 14, 2026*)

Original Question: What exhibitions are starting at the Shen-Zhen Museum ***this month***?
Correct Answer: Inheritance of Craftsmanship (*Jan 2026*)

L2 Case: 1-Hop (Failure)

L2 Question: On Oct 28, 2025, the Spring Institute was established in a prefecture-level city. What exhibitions are starting ***this month*** in that city's museum?
Agent Trajectory:
[Action1] <think>none</think> *Search*: Spring Institute, Oct 28, 2025
[Observation1] Result: Oct 26, 2025 *Temporal Confusion*
[Action2] <think> I need to search for exhibitions that will begin at the Shenzhen Museum in October 2025. </think> *Visit*: Shenzhen Museum exhibitions, Oct 2025
[Action…]

L3 Case: 2-Hop (Success)

L3 Question: Two days after the 2025 World Cities Day opened in Shanghai, the Spring Institute was established in a city... Whatexhibitions are starting this month there?
Agent Trajectory:
[Action1] <think>The question is conflict, firstly I need…</think> *Search*: 2025 World Cities Day, Shanghai
[Observation1] Result: Oct 26, 2025
[Action2] <think>I neet to …</think> *Search*: Spring Institute, Oct 28, 2025 *Rigorous Planning*
[Observation2] Result: Shenzhen
[Action3] <think>Now I need to confirm the current time (January 14, 2026) and the upcoming exhibition at Shenzhen Museum</think> *Visit*: Shenzhen Museum exhibitions, Jan 2026
[Action…]

Figure 4: Case Study: Temporal Confusion (L2) vs. Rigorous Planning (L3). Comparison of agent trajectories on the Shenzhen Museum question.

Retrieval (20%) occurs when agents rely on search snippets without clicking the authoritative websites. Even when visiting the correct page, agents exhibit Incomplete Scanning (15%), often reading only the first few items of a list. Furthermore, Source Quality (10%) issues arise when agents prioritize blogs over official government portals.

Temporal Misalignment (20%). Agents struggle with dynamic time anchoring. A common failure is Rigid Grounding, where agents search for a literal date string but fail to recognize data published under relative terms. **Crucially, this category also includes the Temporal Confusion failures observed in L2 tasks, where agents fail to disengage from a retrieved past date (e.g., the event time) and correctly re-anchor to the evaluation time.**

Hallucination & Logic (25%). When retrieval fails, agents often resort to Hallucination (15%), Logic Errors (10%) include calculation mistakes or misclassification.

Level	Jan 10	Jan 15	Jan 20 (w/ Time Anchor)
L2	33.0	40.2	40.4
L3	39.0	44.1	37.3

Table 3: Performance of L2 and L3 tasks, with a Time Anchor added to the prompt for the Jan 20 setting.

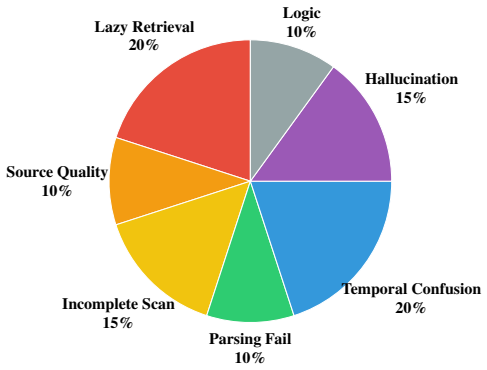

Figure 5: Fine-grained Distribution of Failure Modes. Failures are decomposed into 7 specific behavioral flaws.

5 Conclusion

We introduced RT-QA, a dynamic benchmark for evaluating real-time question answering agents under an evaluation-via-execution paradigm. Instead of relying on static gold answers, RT-QA uses executable workflows to retrieve and compute ground truth at evaluation time. To make this paradigm practical, we further designed an agent-assisted pipeline for workflow generation, validation, and self-repair, together with manual verification to ensure benchmark reliability.

Our extensive evaluation reveals that while current agents excel at static reasoning, they remain fragile in dynamic environments. The discovery of Temporal Confusion reveals a weakness that agents may get stuck in the past, mistakenly using historical dates to answer questions about the present. This implies that simply giving agents better search tools is not enough. Instead, they need better planning capabilities to distinguish between past events and current goals. Ultimately, to handle the changing web, agents must evolve from simple searchers into proactive investigators that can actively verify and manage information. We release RT-QA not just as a dataset, but as an extensible infrastructure. By democratizing the creation of live benchmarks, we hope to accelerate the transition of Large Language Models from knowledgeable chatbots to truly reliable, real-time assistants.

Contribution Statement

Wenjie Zhou and Yuan Gao contributed equally to this work. Yuan Gao conducted part of this work during an internship at Li Auto Inc. Xin Zhou is the corresponding author.

References

[AI, 2024] Cognition AI. Devin: The first ai software engineer. https://www.cognition-labs.com/blog, 2024. Ac-

cessed: 2026-01-15.

[Cheng and Dou, 2025] Jiehan Cheng and Zhicheng Dou. Dailyqa: A benchmark to evaluate web retrieval augmented llms based on capturing real-world changes. *arXiv preprint arXiv:2505.17162*, 2025.

[Deng *et al.*, 2023] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[Dhingra *et al.*, 2022] Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273, 2022.

[He *et al.*, 2024] Yancheng He, Shilong Li, Jiaheng Liu, Yingshui Tan, Weixun Wang, Hui Huang, Xingyuan Bu, Hangyu Guo, Chengwei Hu, Boren Zheng, Zhuoran Lin, Xuepeng Liu, Dekai Sun, Shirong Lin, Zhicheng Zheng, Xiaoyong Zhu, Wenbo Su, and Bo Zheng. Chinese simpleqa: A chinese factuality evaluation for large language models. *arXiv preprint arXiv:2411.07140*, 2024.

[Izacard *et al.*, 2022] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 23(1):1–40, 2022.

[Jia *et al.*, 2018] Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. Tempquestions: A benchmark for temporal question answering. In *Proceedings of the 2018 World Wide Web Conference*, pages 1057–1062, 2018.

[Kasai *et al.*, 2023] Jungo Kasai, Keisuke Sakaguchi, Yoichi Takahashi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A. Smith, Yejin Choi, and Kentaro Inui. Realtime qa: What's the answer right now? *Advances in Neural Information Processing Systems*, 36, 2023.

[Kwiatkowski *et al.*, 2019] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and Kristina Toutanova. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[Lewis *et al.*, 2020] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9459–9474, 2020.

[Liska *et al.*, 2022] Adam Liska, Tomas Kocisky, Elena Gribovskaya, Tayfun Terzi, Eren Sezener, Devang Agrawal, Cyprien de Masson d'Autume, Tim Scholtes, Mose Manzagol, Peter Young, Geoffrey Irving, Phil Blunsom, and Jack W. Rae. Streamingqa: A benchmark for adaptation to new knowledge over time in question answering models. *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

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

[Meem *et al.*, 2024] Jannat Ara Meem, Muhammad Shihab Rashid, Yue Dong, and Vagelis Hristidis. Pat-questions: A self-updating benchmark for present-anchored temporal question-answering. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13129–13148, 2024.

[Nakano *et al.*, 2021] 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 Elias, Clemens Winter, Lonnie Bickel, Matthew Tezak, Jerry Tworek, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[OpenAI, 2024] OpenAI. Searchgpt prototype: A temporary prototype of new ai search features. https://openai.com/index/searchgpt/, 2024. Accessed: 2026-01-15.

[OpenAI, 2025] OpenAI. OpenAI GPT-5 system card, 2025. arXiv preprint arXiv:2601.03267.

[Patil *et al.*, 2023] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

[Perplexity, 2024] Perplexity. Perplexity ai: An ai-powered answer engine. https://www.perplexity.ai, 2024. Accessed: 2026-01-15.

[Rajpurkar *et al.*, 2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[Schick *et al.*, 2024] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.

[Team *et al.*, 2025] GLM Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, Yean Cheng, Yifan An, Yilin Niu, Yuanhao Wen, Yushi Bai, Zhengxiao Du, Zihan Wang, Zilin Zhu, Bohan Zhang, Bosi Wen, Bowen Wu, Bowen Xu, Can Huang, Casey Zhao, Changpeng Cai, Chao Yu, Chen Li, Chendi Ge, Chenghua Huang, Chenhui Zhang, Chenxi Xu, Chenzheng Zhu, Chuang Li, Congfeng Yin, Daoyan Lin, Dayong Yang, Dazhi Jiang, Ding Ai, Erle Zhu, Fei Wang, Gengzheng Pan, Guo Wang, Hailong

Sun, Haitao Li, Haiyang Li, Haiyi Hu, Hanyu Zhang, Hao Peng, Hao Tai, Haoke Zhang, Haoran Wang, Haoyu Yang, He Liu, He Zhao, Hongwei Liu, Hongxi Yan, Huan Liu, Huilong Chen, Ji Li, Jiajing Zhao, Jiamin Ren, Jian Jiao, Jiani Zhao, Jianyang Yan, Jiaqi Wang, Jiayi Gui, Jiayue Zhao, Jie Liu, Jijie Li, Jing Li, Jing Lu, Jingsen Wang, Jingwei Yuan, Jingxuan Li, Jingzhao Du, Jinhua Du, Jinxin Liu, Junkai Zhi, Junli Gao, Ke Wang, Lekang Yang, Liang Xu, Lin Fan, Lindong Wu, Lintao Ding, Lu Wang, Man Zhang, Minghao Li, Minghuan Xu, Mingming Zhao, Mingshu Zhai, Pengfan Du, Qian Dong, Shangde Lei, Shangqing Tu, Shangtong Yang, Shaoyou Lu, Shijie Li, Shuang Li, Shuang-Li, Shuxun Yang, Sibo Yi, Tianshu Yu, Wei Tian, Weihan Wang, Wenbo Yu, Weng Lam Tam, Wenjie Liang, Wentao Liu, Xiao Wang, Xiaohan Jia, Xiaotao Gu, Xiaoying Ling, Xin Wang, Xing Fan, Xingru Pan, Xinyuan Zhang, Xinze Zhang, Xiuqing Fu, Xunkai Zhang, Yabo Xu, Yandong Wu, Yida Lu, Yidong Wang, Yilin Zhou, Yiming Pan, Ying Zhang, Yingli Wang, Yingru Li, Yinpei Su, Yipeng Geng, Yitong Zhu, Yongkun Yang, Yuhang Li, Yuhao Wu, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yuxuan Zhang, Zezhen Liu, Zhen Yang, Zhengda Zhou, Zhongpei Qiao, Zhuoer Feng, Zhuorui Liu, Zichen Zhang, Zihan Wang, Zijun Yao, Zikang Wang, Ziqiang Liu, Ziwei Chai, Zixuan Li, Zuodong Zhao, Wenguang Chen, Jidong Zhai, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models, 2025.

[Team, 2024] OpenDevin Team. Opendevin: An open platform for ai software engineers as collaborators. *arXiv preprint arXiv:2407.16741*, 2024.

[Team, 2025] Alibaba-NLP Team. Deepresearch: A unified framework for autonomous web navigation and reasoning. *Open Source Software*, 2025.

[Vu *et al.*, 2023] Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, and Thang Luong. Freshllms: Refreshing large language models with search engine augmentation. *arXiv preprint arXiv:2310.03214*, 2023.

[Wang *et al.*, 2023] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[Wang *et al.*, 2025] Zhao Wang, Ziliang Zhao, and Zhicheng Dou. TimeRAG: Enhancing complex temporal reasoning with search engine augmentation. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM)*, 2025.

[Zhang and Choi, 2022] Michael J. Q. Zhang and Eunsol Choi. Situational qa: Incorporating extra-linguistic contexts into qa. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[Zheng *et al.*, 2023] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2023.

[Zhou *et al.*, 2024] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.