THE AGENT'S FIRST DAY: BENCHMARKING LEARNING, EXPLORATION, AND SCHEDULING IN THE WORKPLACE SCENARIOS

Daocheng Fu[1,2,*], **Jianbiao Mei**[3,2,*], **Rong Wu**[3,2,*], **Xuemeng Yang**[2,*], **Jia Xu**[2,4], **Yufan Shen**[2], **Ding Wang**[2], **Pinlong Cai**[2,], **Yong Liu**[3,✉], **Licheng Wen**[2,4,5,✉], **Botian Shi**[2,✉]

[1] Fudan University [2] Shanghai Artificial Intelligence Laboratory
[3] Zhejiang University [4] Shanghai Innovation Institute [5] Shanghai Jiao Tong University
* Equal Contribution, ✉ Corresponding Authors

ABSTRACT

The rapid evolution of Multi-modal Large Language Models (MLLMs) has advanced workflow automation; however, existing research mainly targets performance upper bounds in static environments, overlooking robustness for stochastic real-world deployment. We identify three key challenges: dynamic task scheduling, active exploration under uncertainty, and continuous learning from experience. To bridge this gap, we introduce Trainee-Bench, a dynamic evaluation environment that simulates a "trainee" agent continuously exploring a novel setting. Unlike traditional benchmarks, Trainee-Bench evaluates agents along three dimensions: (1) context-aware scheduling for streaming tasks with varying priorities; (2) prudent information acquisition to reduce hallucination via active exploration; and (3) continuous evolution by distilling generalized strategies from rule-based, dynamically generated tasks. Experiments show that cutting-edge agents have significant deficiencies in dynamic environments, especially in active exploration and continual learning. Our work establishes a framework for assessing agent reliability, shifting evaluation from static tests to realistic, production-oriented scenarios. Our codes are available at https://github.com/KnowledgeXLab/EvoEnv

1 INTRODUCTION

The rapid evolution of Multi-modal Large Language Models (MLLMs) has significantly advanced complex workflow automation Guo et al. (2024a); Li et al. (2024). However, while existing systems facilitate tool utilization, research predominantly focuses on performance *upper bounds* in controlled environments Gottweis et al. (2025); Team et al. (2025). This paradigm often neglects the robustness required for authentic, stochastic environment settings. Unlike sterile experiments that ignore environmental noise, real-world deployment involves streaming, randomized tasks. Consequently, agents must not only orchestrate scheduling with awareness of environmental dynamics but also ensure reliability, effectively bridging the gap between laboratory settings and production needs Fu et al. (2025); Yang et al. (2025).

As illustrated in Fig. 1, shifting from static setups to realistic interaction scenarios introduces three primary challenges: (1) **Dynamic task scheduling and context management.** Agents must perform rational temporal planning for streaming tasks to meet deadlines while maintaining context awareness, thereby deciding to mitigate cross-task interference. (2) **Active exploration in novel tasks.** Instead of hallucinating actions in uncertain scenarios, a reliable agent should exercise prudence by actively exploring or soliciting assistance to acquire necessary clues Yu et al. (2024); Qiu et al. (2025); Arora et al. (2025). (3) **Learning from previous tasks.** To ensure long-term stability, agents must learn from prior experiences to prevent the recurrence of historical errors in subsequent scenarios Zhou et al. (2025); Silver & Sutton (2025); Wu et al. (2025b); Gao et al. (2025a).

Although numerous benchmarks exist (Table 1), they fall short in simulating the dynamic nature of realistic environments and lack mechanisms to evaluate active exploration and continual learn-

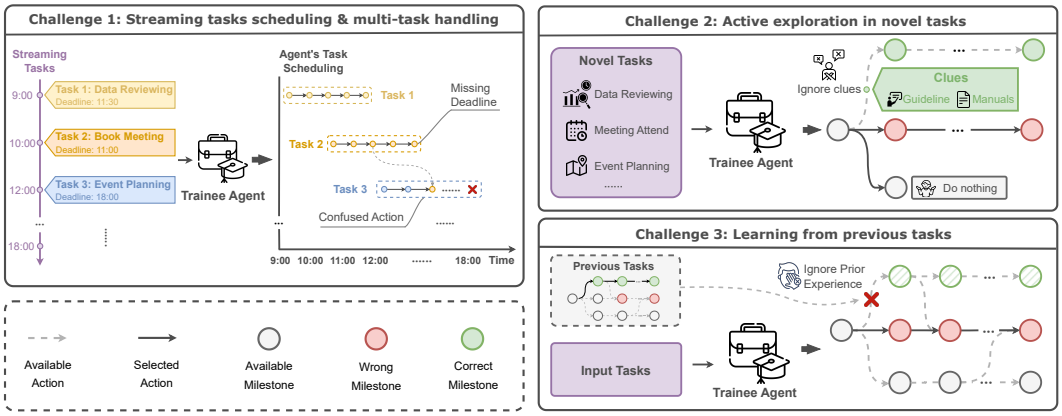

Figure 1: Overview of challenges in current agent systems: (1) effective scheduling and multi-task planning for streaming task inputs; (2) the ability to suspect unsolvable tasks and solicit guidance through active exploration; and (3) robust experience summarization, retrieval, and utilization to enhance performance stability.

Table 1: Comparison of different agent benchmarks. "Multi-App Interaction" : support for cross-application workflows. "Checkpoint Feedback": the availability of intermediate evaluation milestones beyond final success rates. "Partial Observability": agents must actively explore hidden states rather than operating on full information. "Dynamic Configuration": environments with randomized, evolving parameters and dynamic composite scenarios.

Benchmark	Multi-App Interaction	Checkpoint Feedback	Partial Observability	Dynamic Configuration
GAIA-2 Froger et al. (2025)	✓	✗	✗	✗
TheAgentCompany Xu et al. (2024)	✓	✓	✗	✗
Tool Bench Qin et al. (2024)	✓	✗	✗	✗
StableToolBench Guo et al. (2024b)	✓	✗	✗	✗
Toolathon Li et al. (2025)	✓	✗	✗	✗
τ-bench Yao et al. (2024)	✗	✗	✓	✗
Trainee-Bench (Ours)	✓	✓	✓	✓

ing capabilities. To address this, we introduce Trainee-Bench, a dynamic evaluation environment simulating an "trainee" agent continuously exploring a novel setting. Trainee-Bench is designed to assess the three aforementioned capabilities of an agent rigorously: First, regarding **task execution**, Trainee-Bench presents streaming tasks with distinct priorities. This demands superior scheduling to maintain context awareness and mitigate inter-task interference. Second, concerning **information acquisition**, critical clues are initially concealed. Agents must demonstrate prudence by actively exploring or soliciting help rather than engaging in hallucinated actions. Finally, in terms of **continuous evolution**, tasks are dynamically generated based on rules rather than fixed datasets. This compels agents to distill generalized strategies from prior tasks instead of rote-memorizing instances, effectively preventing repeated errors.

We conduct extensive experiments on Trainee-Bench, revealing that state-of-the-art (SOTA) agents exhibit significant room for improvement in dynamic environments, particularly regarding active exploration and continual learning. Our contributions are summarized as follows:

- Proposing a novel benchmarking paradigm that shifts from static, laboratory evaluations to stochastic, production-oriented environments;
- Establishing a comprehensive framework to rigorously assess temporal scheduling, prudent decision-making under uncertainty, and long-term strategic evolution of MLLM agents;
- Through extensive experimentation, we characterize the reliability gap of current agents, highlighting their deficiencies in handling concealed clues and distilling generalized experiences.

2 TRAINEE-BENCH

We introduce Trainee-Bench, a benchmark designed to simulate authentic workplace environments through a series of human-crafted meta-tasks. In this framework, agents are evaluated as "corporate

interns" navigating realistic company routines, such as attending meetings or reviewing data. The benchmark rigorously assesses three core competencies: dynamic multi-tasking, proactive exploration under uncertainty, and continuous self-evolution. To construct Trainee-Bench, we adopt a bottom-up design strategy that evaluates agent capabilities ranging from atomic skills to holistic workflows. We first develop a suite of rule-based *meta-tasks* to serve as foundational templates. These templates can be instantiated into numerous specific task instances, which are then flexibly orchestrated into complex, composite scenarios with explicit temporal constraints. Finally, we implement an automated verification mechanism to ensure a rigorous and objective assessment of agent performance in these sophisticated environments. Detailed implementations are elaborated in the following sections.

2.1 Preliminaries

In Trainee-Bench, we formalize the interaction environment as a dynamic state transition system. At any given time step t, the environmental configuration is represented by a state $m_t \in \mathcal{M}$, which encapsulates the real-time status of the agent and various environmental entities, such as non-player characters (NPCs), files, and databases:

$$m_t = \{s_{\text{agent}}, s_{\text{NPC}}, s_{\text{file}}, s_{\text{data}}\} \tag{1}$$

The agent interacts with the environment by invoking tools to execute actions $a_t \in \mathcal{A}$. These actions trigger state transitions defined by the function $\mathcal{T} : \mathcal{M} \times \mathcal{A} \to \mathcal{M}$, defined as $\mathcal{T}(m_t, a_t) = m_{t+1}$. This formalism establishes the mathematical foundation for both the procedural generation of individual meta-tasks and the dynamic orchestration of concurrent, multi-threaded workflows.

2.2 Meta-Task Design

To prevent agents from over-fitting to static datasets, Trainee-Bench generates dynamic task instances derived from a library of human-crafted **Meta-tasks**, denoted as $\mathcal{R}$. Each $r_i \in \mathcal{R}$ represents an abstract logical template (e.g., "Transaction Auditing") that can spawn an infinite variety of concrete task instances.

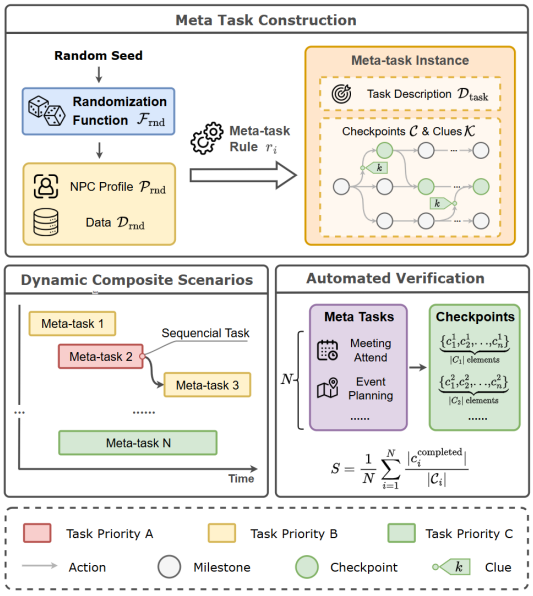

Figure 2: Overview of Trainee-Bench construction: (1) Unique task instances generated via random parameters. (2) Temporal composition of instances into task streams. (3) Automated verification of execution results.

Dynamic Task Instantiation. The instantiation process begins with a randomization engine $\mathcal{F}_{\text{rnd}}$. Controlled by a random seed, this engine synthesizes stochastic environment variables, specifically NPC profiles $\mathcal{P}_{\text{rnd}}$ and environmental data $\mathcal{D}_{\text{rnd}}$:

$$\{\mathcal{P}_{\text{rnd}}, \mathcal{D}_{\text{rnd}}\} = \mathcal{F}_{\text{rnd}}(\text{seed}) \tag{2}$$

These variables introduce realistic uncertainty, such as randomized sender names, file paths, or conflicting schedules. Then, the selected meta-task rule r_i integrates these variables into its predefined logic to yield a task triple:

$$\{\mathcal{P}_{\text{rnd}}, \mathcal{D}_{\text{rnd}}\} \xrightarrow{r_i} \{\mathcal{D}_{\text{task}}, \mathcal{C}, \mathcal{K}\}, \tag{3}$$

where $\mathcal{D}_{\text{task}}$ represents a unique natural language objective, $\mathcal{C}$ denotes a set of verifiable checkpoints for evaluation, and $\mathcal{K}$ comprises the latent clues required for task completion. By varying the seed, a single meta-task r_i can generate multiple distinct task triples. This one-to-many mapping ensures that agents must employ general reasoning and actively explore latent clues rather than relying on rote memorization.

Task Diversity and Coverage. The decoupling of logical rules r_i from stochastic data $\{\mathcal{P}_{\text{rnd}}, \mathcal{D}_{\text{rnd}}\}$ enables a systematic and scalable expansion of the benchmark's coverage along two dimensions:

- **Task Dimensions:** To assess a broad spectrum of cognitive demands rather than isolated skills, our rule library $\mathcal{R}$ comprises 181 meta-tasks categorized into four functional domains: (i) *information synthesis and analysis* (e.g., transaction auditing), (ii) *time management and scheduling* (e.g., meeting attending), (iii) *proactive inquiry and monitoring* (e.g., automated website monitoring), and (iv) *strategic modeling and optimization* (e.g., advertising campaign planning).
- **Stochastic Robustness:** The randomization function $\mathcal{F}_{\text{rnd}}$ ensures that even for a single rule r_i, the resulting environment remains unpredictable. By varying user personas, file system hierarchies, and numerical distributions within $\mathcal{D}_{\text{rnd}}$, Trainee-Bench prevents agents from memorizing specific environment layouts or hard-coded solutions, forcing them to generalize across diverse and unfamiliar operational contexts.

Partial Observability. A defining characteristic of Trainee-Bench is the deliberate enforcement of partial observability to simulate real-world ambiguity. We introduce an information gap between the agent's initial knowledge and the required state information. Specifically, while the agent is provided with the task objective $\mathcal{D}_{\text{task}}$, the set of essential clues $\mathcal{K} = \{k_1, \ldots, k_n\}$—such as technical manuals, access passwords, or implicit verbal instructions—is strictly withheld from the initial prompt. Consequently, the agent cannot complete the task by simply following the starting instructions. Instead, it must engage in proactive exploration by navigating file systems or conducting multi-turn dialogues with NPCs to uncover the latent clues embedded in the environment. This mechanism shifts the agent's role from a passive executor to a proactive problem-solver that must synthesize fragmented information under uncertainty.

2.3 Dynamic Composite Scenarios

To simulate a realistic, multi-threaded scenario, we transcend the execution of isolated rules r_i by synthesizing multiple meta-tasks into *dynamic composite scenarios*. Formally, a scenario is constructed by stochastically assembling a subset of generated tasks and distributing them along a continuous timeline, characterized by distinct deadlines. This composition introduces three critical structural constraints regarding the interaction:

Conflict Resolution: During the aggregation of meta-tasks, conflicts may arise, such as overlapping entity identifiers (e.g., character names or filenames) or a single non-player character (NPC) acting as a custodian for clues k_i^1 and k_j^2 from disparate tasks. To resolve such conflicts, we enforce a uniqueness constraint on all entity demarcations within the meta-task rules, ensuring that specific names remain distinct across the composite scenario. Furthermore, in instances where a single NPC safeguards multiple clues, we explicitly define the conditional triggers required to elicit each specific clue[1], thereby mitigating potential information ambiguity.

Temporal Prioritization: We introduce time-critical meta-tasks (e.g., attending a scheduled meeting) that impose rigid temporal constraints. These high-priority interrupts function as *preemptive signals*, mandating that the agent suspend its current workflow, perform a context switch to address the immediate requirement, and subsequently resume the interrupted process. This mechanism rigorously evaluates the agent's capacity for dynamic priority scheduling and long-term memory management.

Inter-task Dependencies: We engineer sequential dependencies where upstream tasks function as informational prerequisites for downstream objectives. In these scenarios, critical task parameters are only revealed during the execution of a preceding task—for instance, a new assignment may be "published" only during an ongoing meeting. This interdependent architecture increases temporal uncertainty and compels the agent to adaptively update its plan based on real-time environmental observations rather than relying on static initial instructions.

2.4 Automated Verification Mechanism

To quantify performance in these complex scenarios, we leverage the set of checkpoints $\mathcal{C} = \{c_1, \ldots, c_n\}$ derived during task construction. Embedded within each meta-task, these checkpoints

[1]Detailed cases illustrating multi-clue NPC interactions and agent-NPC dialogues are provided in the Section 7.1

facilitate a granular assessment of the agent's overall progress and completion quality. The final scenario score S is derived by normalizing the completion ratio of checkpoints across all constituent meta-tasks:

$$S = \frac{1}{N} \sum_{i=1}^{N} \frac{|c_i^{\text{completed}}|}{|\mathcal{C}_i|} \tag{4}$$

where N is the number of meta-tasks, $|\mathcal{C}_i|$ is the total number of checkpoints for task i, and $|c_i^{\text{completed}}|$ is the count of successfully passed checkpoints. Furthermore, each checkpoint triggers detailed natural language feedback, providing structured experiential data that empowers agents to perform retrospective analysis and continuous learning.

2.5 Interaction Protocol

Building upon the constructed scenarios, we define an interaction protocol that bridges the environment and the agent, as depicted in Fig. 3. This workflow comprises three distinct entities: the *Environment*, the *Agent*, and the *MLLM Service*. Upon initialization with task descriptions and tool definitions provided by the environment, the agent constructs its system prompt and engages in an iterative cycle: maintaining conversation history, querying the MLLM Service, and parsing model responses into executable tool calls. We implement a standardized pipeline to orchestrate this Agent-Environment interaction loop, while abstracting three key components, i.e., *system prompt construction*, *history maintenance*, and *MLLM service invocation*, as customizable interfaces. By enforcing a unified interaction protocol, Trainee-Bench provides a modular and flexible framework that supports diverse agent architectures and LLM backends.

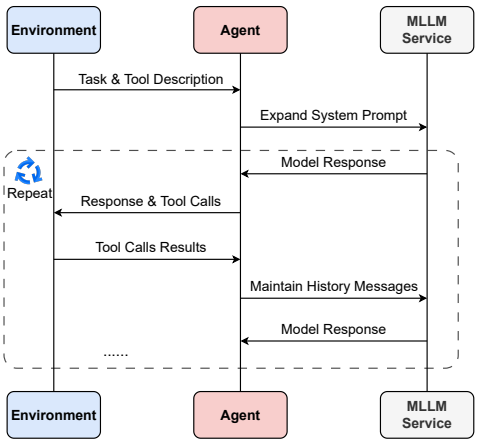

Figure 3: Formulation for the interaction among the environment, agent, and MLLM service.

2.6 Benchmark Design Characteristics

While conventional benchmarks often rely on isolated, static tasks that fail to capture the inherent complexity of real-world workflows, Trainee-Bench provides a high-fidelity workplace simulation by integrating four critical operational dimensions (see Table 1). By moving beyond simplistic execution, our benchmark establishes a rigorous environment to evaluate high-order cognitive capabilities through the following key features:

Robustness against Memorization. By decoupling logical meta-tasks from stochastic environment variables (Sec. 2.2), Trainee-Bench ensures that every evaluation instance is structurally unique. This "one-to-many" instantiation forces agents to rely on generalized reasoning and dynamic state estimation rather than exploiting fixed patterns or hard-coded shortcuts common in static datasets.

Proactive Uncertainty Resolution. Through the enforcement of partial observability, we transform the agent from a passive executor into an active information seeker. The "latent clues" between the initial prompt and target state require multiple rounds of querying and environmental exploration, thus measuring agents' ability under uncertainty.

Stress-testing under Temporal Complexity. The synthesis of composite scenarios introduces a "messy" workspace characterized by preemptive interrupts and inter-task dependencies. This setup moves beyond linear execution, specifically targeting the agent's context-switching efficiency and dynamic priority management—capabilities that are critical for production-grade automation but often ignored in laboratory settings.

Table 2: Overall performance of various LLM agents on our Trainee-Bench.The highest scores in both open-source and closed-source models are highlighted in **bold**.

Model	Success Rate	Checkpoint Score	Average Steps	Average Tool Calls
Closed-source Models				
Gemini-3-Flash Google DeepMind (2025)	**0.35**	**0.63**	**90**	**232**
Claude-4-Sonnet Anthropic (2025)	0.23	0.59	29	75
GPT-5.1 OpenAI (2025)	0.23	0.49	23	62
Grok-4 xAI (2025)	0.20	0.54	34	95
GPT-4o OpenAI (2024)	0.13	0.38	37	51
Open-source Models				
Qwen3-VL-235B-A22B Bai et al. (2025)	**0.14**	**0.37**	**68**	**68**
Llama-4-maverick Meta (2025)	0.04	0.13	34	23

Feedback-driven Evolution. Trainee-Bench is not a "black-box" assessment. By providing granular, natural language feedback at each checkpoint, the environment functions as a learning catalyst. This structured experiential data allows for the evaluation of an agent's "learning curve", measuring how effectively it can perform retrospective analysis and refine its strategies across streaming tasks.

3 EXPERIMENTS

In this section, we conduct a series of experiments on our Trainee-Bench to answer the following key research questions (RQs):

- **RQ1:** Are current LLMs capable of handling complex workplace environments characterized by dynamics and uncertainty? (Section 3.2)
- **RQ2:** Can current LLMs truly achieve consistent continuous learning from past experiences over extended periods? (Section 3.3)
- **RQ3:** What is the performance gap between agent's proactive exploration and passive reception of human guidance? (Section 3.4)

3.1 IMPLEMENTATION DETAILS

Setup. To simulate and evaluate agent performance within the trainee environment, we constructed a benchmark consisting of 50 dynamic scenarios. Each scenario encompasses 2 to 6 task instances, where every task is instantiated based on a rule r uniformly sampled from the meta-task rules set $\mathcal{R}$. The detailed procedure for benchmark construction is outlined in Algorithm 1. We selected a diverse set of seven models for evaluation, spanning open-source and proprietary architectures, general-purpose and tool-specialized variants, as well as distinct model sizes ranging from lightweight to state-of-the-art large-scale parameters. Specifically, the evaluated models are GPT-5.1 OpenAI (2025), GPT-4o OpenAI (2024), Claude-4-Sonnet Anthropic (2025), Gemini-3-Flash Google DeepMind (2025), Grok-4 xAI (2025), Qwen3-VL-A235B Bai et al. (2025), and Llama-4-maverick Meta (2025). All models perform inference, planning, and tool invocation in strict adherence to the protocols defined in Section 2.5. To mitigate context window limitations, historical interaction information is summarized and compressed once the sequence length exceeds a predefined threshold.

Algorithm 1 Benchmark Building

Require: Meta-task rule set $\mathcal{R}$
Ensure: Benchmark $\mathcal{B}$
$\mathcal{B} \leftarrow \emptyset$
for $i = 1$ **to** 50 **do**
  Sample task count $k \sim \mathcal{U}\{2, 6\}$
  Generate scenario tasks S_i:
    $S_i \leftarrow \{\text{INSTANTIATE}(r) \mid r \in \mathcal{R}\}_{j=1}^{k}$
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{S_i\}$
end for
return $\mathcal{B}$

Metrics. To provide an overall and fine-grained assessment of agent performance, we define the following metrics, which are measured daily within the simulation: **Success Rate (SR):** Defined as the percentage of tasks successfully completed. **Checkpoint Score (CS):** Defined as the mean score across all scenarios, where the score for each scenario is calculated via Equation 2.4. **Average Steps:** This metric calculates the average number of thought-and-action steps the agent executed to complete

Table 3: Success Rate and Checkpoint Score of different models under increasing task counts (2, 3, 4, 5, and 6 concurrent tasks). "Overload Models" exhibit a clear decline in both metrics as the task count increases, while "Normal Load Models" maintain relatively stable performance.

Model	**Success Rate (# Tasks)**					**Checkpoint Score (# Tasks)**				
Task Count:	2	3	4	5	6	2	3	4	5	6
Gemini-3-Flash Google DeepMind (2025)	0.50	0.47	0.48	0.36	0.36	0.77	0.68	0.72	0.60	0.60
Grok-4 xAI (2025)	0.40	0.20	0.25	0.24	0.16	0.68	0.53	0.59	0.59	0.45
GPT-4o OpenAI (2024)	0.40	0.10	0.13	0.15	0.14	0.60	0.37	0.39	0.40	0.38
Qwen3-VL-235B-A22B Bai et al. (2025)	0.20	0.17	0.20	0.12	0.14	0.43	0.38	0.48	0.36	0.32
Claude-4-Sonnet Anthropic (2025)	0.25	0.17	0.35	0.20	0.24	0.61	0.54	0.70	0.58	0.57
GPT-5.1 OpenAI (2025)	0.20	0.23	0.35	0.23	0.21	0.46	0.47	0.63	0.49	0.49
Llama-4-maverick Meta (2025)	0.00	0.03	0.03	0.05	0.03	0.09	0.11	0.15	0.15	0.15

its tasks per scenario. **Average Tool Calls:** This metric quantifies the average number of times the agent invoked any tool per scenario.

3.2 Performance of Cutting-edge Models

In this section, we benchmark top-tier LLMs to investigate their limitations when deployed in Trainee-Bench, which is characterized by dynamic and uncertain workplace environments.

Overall Performance. Table 2 summarizes the results averaged across all scenarios. These findings clearly demonstrate the difficulty of Trainee-Bench. Even the best-performing model, Gemini-3-Flash, attains a Success Rate of only 35%, indicating a persistent gap in achieving reliable autonomy in simulated workplaces. Besides, a distinct performance hierarchy is evident. Gemini-3-Flash leads in both metrics, followed by Claude-4-Sonnet and Grok-4. Conversely, Llama-4-maverick lags significantly (0.13 Checkpoint Score) due to issues with instruction following and tool invocation (detailed in Appendix 7.2). Finally, we find that reliability requires sufficient interaction depth. While Gemini-3-Flash uses substantially more steps (90) and tool calls (232) than the middle-tier models, this increased activity reflects the necessary complexity and thoroughness required to successfully navigate dynamic and uncertain scenarios.

Impact of Task Workload. To analyze the impact of workload, we stratify agent performance by the number of meta-tasks per scenario, as shown in Table 3. We observe a clear downward trend in performance for models such as GPT-4o, Grok-4, and Gemini-3-Flash as the workload increases, particularly beyond two tasks. Notably, Gemini-3-Flash, despite its strong overall performance, exhibits a decrease in Success Rate from 50% in 2-task scenarios to 36% in 6-task scenarios. This suggests that managing increased cognitive load—in particular, frequent context switching and temporal uncertainty in dynamic composite scenarios—remains a key challenge for these models. In contrast, Claude-4-Sonnet and GPT-5.1 do not exhibit a strong correlation between performance and the number of tasks, indicating a certain degree of robustness and adaptability to dynamic composite settings. Llama-4-maverick, by comparison, maintains consistently low performance across different task workload, indicating that its reasoning and tool-use capabilities are insufficient for reliably solving the atomic tasks themselves. Interestingly, as the number of tasks increases, Llama-4-maverick has a higher chance of completing at least some of the simpler tasks or intermediate checkpoints, which leads to a slight upward trend in its aggregate Success Rate despite its overall weak capabilities.

	Easy Tasks		**Hard Tasks**	
Model	**SR**	**CS**	**SR**	**CS**
Closed-source Models				
Gemini-3-Flash	**0.46**	**0.74**	**0.32**	**0.56**
Claude-4-Sonnet	0.37	0.70	0.08	0.46
GPT-5.1	0.37	0.63	0.08	0.35
Grok-4	0.34	0.68	0.07	0.37
GPT-4o	0.26	0.58	0.03	0.19
Open-source Models				
Qwen3-VL-A235B	**0.24**	**0.50**	**0.04**	**0.22**
Llama-4-maverick	0.06	0.22	0	0.04

Table 4: Impact of different task difficulties. The highest scores in both open-source and closed-source models are highlighted in **bold**.

Impact of Task Difficulty. We further investigate the impact of meta-task difficulty on agent performance. Specifically, we stratify the tasks into *Easy* and *Hard* subsets based on a manual complexity analysis, incorporating metrics such as average step counts and tool invocation frequencies. In general, hard tasks require substantially more reasoning steps and tool calls. They also contain a greater amount of implicit or hidden information, which necessitates more active exploration of the environment by the agent to uncover the solution. Moreover, a subset of the hard tasks additionally demands explicit task modeling and optimization capabilities, thereby imposing stricter requirements on the agent's level of intelligence

Table 4 reveals a stark contrast. While most models achieve respectable performance on Easy tasks, their capabilities decline precipitously on Hard ones. For instance, Grok-4's Success Rate plummets from 34% on Easy tasks to just 7% on Hard tasks, with similar drops observed for Claude-4-Sonnet (37% to 8%) and GPT-4o (26% to 3%). This breakdown highlights that the capacity to resolve complex problems is the key differentiator among top-tier models. Notably, while Gemini-3-Flash also experiences a decline, it sustains a significantly higher Success Rate (32%) on Hard tasks, demonstrating superior robustness in complex reasoning compared to its peers.

3.3 Analysis of Continual Learning Capability

Setup. To evaluate the agent's capacity for learning from prior experience, we curated 50 continual learning scenarios, with each comprising 2 to 6 distinct meta-tasks. Distinguished from the configuration in Section 3.1, each scenario herein is structured into two phases (referred to as Day 1 and Day 2). While the sequence, quantity, and types of tasks remain invariant across both days, the specific input parameters for task instantiation vary. For these experiments, we employ MUSE Yang et al. (2025), a SOTA continual learning framework.

Upon the completion of tasks on Day 1, the MUSE agent receives feedback tailored to specific task outcomes. For instance, if a task checkpoint c_i is missed, the environment explicitly notifies the agent of the omission, emphasizing the need for attention in subsequent attempts[2]. Leveraging this feedback alongside its historical interaction trajectories, the agent engages in a reflective process to summarize insights e_i, which are subsequently utilized to guide task execution on Day 2. We selected GPT-4o as the backbone model due to its intermediate performance profile.

Results. The experimental results within the continual learning setting are presented in Table 5. Overall, the MUSE agent exhibits a counterintuitive decline in Checkpoint Score when utilizing accumulated experience. Upon stratifying the results into easy and hard tasks, however, a distinct divergence in performance becomes apparent. For easy tasks, the agent without experience achieves relatively high scores, whereas the incorporation of experience leads to a marked degradation. Conversely, on hard tasks where the baseline performance is lower, the utilization of experience yields varying degrees of improvement.

We further scrutinized the experience summarized by the MUSE agent and observed that insights derived from Day 1 do not consistently facilitate task execution on Day 2. The agent extracts experience e_i based on the unreached checkpoint c_i on Day 1. However, due to the stochastic nature of dynamic environments Fu et al. (2025), the agent may encounter failure at a different checkpoint c_j on Day 2, rendering the previously acquired experience irrelevant.

Table 5: Performance of the continual learning, comparing Day 1 (no experience) with Day 2 (after receiving feedback), focusing on the Checkpoint Score (CS).

Task Type	Day 1 Score	Day 2 Score	Gain (Δ CS)
Overall	0.42	0.36	-0.06
Easy Tasks	0.61	0.50	-0.11
Hard Tasks	0.20	0.24	+0.04

Consequently, for easy tasks, the agent's high initial success rate results in a scarcity of accumulated experience. The limited experience available for Day 2 may provide misleading guidance, thereby lowering the overall score. In contrast, for hard tasks characterized by a lower baseline, the agent accumulates a richer set of experiences on Day 1, which contributes to performance gains on Day 2. Nevertheless, given the inherent difficulty of hard tasks, the improvement remains marginal in the absence of extensive training. We will further discuss how the agent can leverage external feedback to enhance performance in Section 3.4.

[2]One example of daily feedback can be found in Section 7.4

3.4 Benefits of Human Guidance

In Trainee-Bench, agents must proactively explore to overcome the dynamics and uncertainty. However, experiments in Section 3.2 show that agents struggle significantly with hard-difficulty tasks. While continuous learning yields some improvement, the gains remain marginal, as illustrated in Section 3.3. We attribute this underperformance to intrinsic limitations in LLMs: a restricted capacity for exploration, often leading to disorientation, and an inability to effectively synthesize and leverage past experiences. To isolate the impact of these limitations, we conducted a comparative experiment contrasting the upper bound of the agent's autonomous capability against the performance achieved through human-provided clues.

We employ GPT-4o on a subset of hard tasks. To simulate human guidance, we manually provide tiered hints designed to progressively simplify strategic planning by offering crucial high-level insights (Section 7.3). The results, shown in Fig. 4, reveal that human assistance yields substantial improvements: the average score surges from 0.24 to 0.83, demonstrating a clear positive correlation between performance and the level of guidance. In contrast, multiple iterations of self-evolution yielded only negligible gains (+0.04). This significant gap highlights the agent's current shortcomings in autonomous exploration, as well as in experience summarization and utilization.

4 Conclusion

We introduce Trainee-Bench, a benchmark designed to bridge the gap between static setups and dynamic and uncertain workplace scenarios. Constructed via a bottom-up strategy that links atomic skills to holistic workflows, Trainee-Bench orchestrates rule-based meta-task templates into complex, time-constrained scenarios, supported by an automated verification mechanism for rigorous assessment. It evaluates three core competencies: dynamic multi-tasking, proactive exploration under uncertainty, and continuous self-evolution. Experiments reveal SOTA agents struggle with uncertainty and continuous learning, exhibiting a significant performance gap compared to human-guided execution. This highlights a critical need to pivot from optimizing isolated skills to mechanisms for robust exploration and experience internalization.

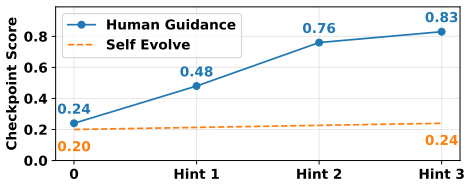

Figure 4: Comparison contrasting the upper bound of the agent's autonomous capability against the performance achieved through human-provided clues.

5 Limitations

We acknowledge several limitations in our current work that point towards future research directions. First, regarding benchmark construction, the diversity of task composition is currently constrained and lacks complex causal inter-dependencies; future iterations will incorporate rigid causal chains to better simulate dynamic realities. Additionally, the reliance on manually crafted rules for meta-tasks limits the scalability of our benchmark, which motivates our future focus on developing automated methods for rule generation. Second, in terms of experimental scope, computational resource, and time constraints restricted our evaluation to a selected set of agent frameworks within a specific workplace simulation context. To provide a more comprehensive and robust assessment, future work will aim to evaluate a broader spectrum of top-tier frameworks and extend the simulation environments to include diverse domains, such as complex production and industrial settings.

References

Anthropic. Introducing claude 4. https://www.anthropic.com/news/claude-4, 2025. Online; published May 22, 2025. Accessed: 2026-01-06.

Rahul K Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, et al. Health-

bench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Silveira Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, et al. Are: Scaling up agent environments and evaluations. *arXiv preprint arXiv:2509.17158*, 2025.

Daocheng Fu, Jianbiao Mei, Licheng Wen, Xuemeng Yang, Cheng Yang, Rong Wu, Tao Hu, Siqi Li, Yufan Shen, Xinyu Cai, et al. Re-searcher: Robust agentic search with goal-oriented planning and self-reflection. *arXiv preprint arXiv:2509.26048*, 2025.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025a.

Xuanqi Gao, Siyi Xie, Juan Zhai, Shiqing Ma, and Chao Shen. Mcp-radar: A multi-dimensional benchmark for evaluating tool use capabilities in large language models. *arXiv preprint arXiv:2505.16700*, 2025b.

Google DeepMind. Gemini 3 flash. https://deepmind.google/models/gemini/flash/, 2025. Accessed: 2026-01-06.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, et al. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024a.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. *arXiv preprint arXiv:2403.07714*, 2024b.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2023.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, et al. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. *arXiv preprint arXiv:2510.25726*, 2025.

Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9, 2024.

Jianbiao Mei, Tao Hu, Daocheng Fu, Licheng Wen, Xuemeng Yang, Rong Wu, Pinlong Cai, Xinyu Cai, Xing Gao, Yu Yang, et al. O^2-searcher: A searching-based agent model for open-domain open-ended question answering. *arXiv preprint arXiv:2505.16582*, 2025.

Meta. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/, 4 2025. Accessed2026-01-06.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2023.

OpenAI. Hello gpt-4o. https://openai.com/index/hello-gpt-4o/, 5 2024. Accessed2026-01-06.

OpenAI. Gpt-5.1: A smarter, more conversational chatgpt. https://openai.com/index/gpt-5-1/, 11 2025. Accessed2026-01-06.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565, 2024.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186, 2024.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, et al. Tool learning with foundation models. *ACM Computing Surveys*, 57(4):1–40, 2024.

Pengcheng Qiu, Chaoyi Wu, Shuyu Liu, Weike Zhao, Zhuoxia Chen, Hongfei Gu, Chuanjin Peng, Ya Zhang, Yanfeng Wang, and Weidi Xie. Quantifying the reasoning abilities of llms on real-world clinical cases. *arXiv preprint arXiv:2503.04691*, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

David Silver and Richard S Sutton. Welcome to the era of experience. *Google AI*, 1, 2025.

InternAgent Team, Bo Zhang, Shiyang Feng, Xiangchao Yan, Jiakang Yuan, Runmin Ma, Yusong Hu, Zhiyin Yu, Xiaohan He, Songtao Huang, Shaowei Hou, Zheng Nie, Zhilong Wang, Jinyao Liu, Tianshuo Peng, Peng Ye, Dongzhan Zhou, Shufei Zhang, Xiaosong Wang, Yilan Zhang, Meng Li, Zhongying Tu, Xiangyu Yue, Wangli Ouyang, Bowen Zhou, and Lei Bai. Internagent: When agent becomes the scientist – building closed-loop system from hypothesis to verification, 2025. URL https://arxiv.org/abs/2505.16938.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. *arXiv preprint arXiv:2407.18901*, 2024.

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.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Rong Wu, Pinlong Cai, Jianbiao Mei, Licheng Wen, Tao Hu, Xuemeng Yang, Daocheng Fu, and Botian Shi. Kg-traces: Enhancing large language models with knowledge graph-constrained trajectory reasoning and attribution supervision. *arXiv preprint arXiv:2506.00783*, 2025a.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, et al. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025b.

xAI. Grok 4. `https://x.ai/news/grok-4`, 7 2025. Accessed2026-01-06.

Frank F Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, et al. Theagentcompany: Benchmarking llm agents on consequential real world tasks, 2024. *URL https://arxiv. org/abs/2412.14161*, 2024.

Yunhe Yan, Shihe Wang, Jiajun Du, Yexuan Yang, Yuxuan Shan, Qichen Qiu, Xianqing Jia, Xinge Wang, Xin Yuan, Xu Han, et al. Mcpworld: A unified benchmarking testbed for api, gui, and hybrid computer use agents. *arXiv preprint arXiv:2506.07672*, 2025.

Cheng Yang, Xuemeng Yang, Licheng Wen, Daocheng Fu, Jianbiao Mei, Rong Wu, Pinlong Cai, Yufan Shen, Nianchen Deng, Botian Shi, et al. Learning on the job: An experience-driven self-evolving agent for long-horizon tasks. *arXiv preprint arXiv:2510.08002*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Qingchen Yu, Shichao Song, Ke Fang, Yunfeng Shi, Zifan Zheng, Hanyu Wang, Simin Niu, and Zhiyu Li. Turtlebench: Evaluating top language models via real-world yes/no puzzles. *arXiv preprint arXiv:2410.05262*, 2024.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, et al. Memento: Fine-tuning llm agents without fine-tuning llms. *arXiv preprint arXiv:2508.16153*, 2025.

6 Related Works

To situate our work, we review the landscape of language agents from two perspectives. We first discuss the evolution of agent systems and their core capabilities, which establishes the context for what modern benchmarks are expected to measure. Subsequently, we analyze the paradigms of existing agent benchmarks to identify the critical gaps in evaluation that **Trainee-Bench** is designed to address.

6.1 Evolution of Agent Systems

The evolution of agent systems began with reasoning enhancements like Chain-of-Thought prompting Wei et al. (2022), paving the way for foundational frameworks such as ReAct Yao et al. (2022). ReAct established the powerful paradigm of interleaving reasoning (Thought) with environmental actions (Action), a core component of many modern agents. Parallel research has focused on improving native tool use, either through large-scale fine-tuning Patil et al. (2024) or by enabling models to teach themselves how to use tools Schick et al. (2023).

Building on this foundation, subsequent work has endowed agents with more human-like capabilities Mei et al. (2025); Wu et al. (2025a). This includes the development of long-term memory systems for complex simulations Yang et al. (2025); Park et al. (2023) and mechanisms for self-reflection, where agents analyze mistakes to iteratively refine their plans on a given task Fu et al. (2025); Wu et al. (2025b); Yang et al. (2025). Other frontiers include open-ended exploration in complex environments like Minecraft Wang et al. (2023) and the advancement of multi-agent systems for collaborative problem-solving Hong et al. (2023); Qian et al. (2024).

6.2 Paradigms in Agent Evaluation

Influential benchmarks have emerged across a wide spectrum, from foundational tool-use evaluations Qin et al. (2024); Guo et al. (2024b) to large-scale benchmarks for general reasoning Mialon et al. (2023); Froger et al. (2025), consequential real-world tasks Xu et al. (2024), and software development Jimenez et al. (2023). Despite their diversity and complexity, they largely operate under a static and fully-observable paradigm. Their static task organization, presenting a fixed set of problems, cannot assess an agent's ability to manage a continuous stream of tasks. Similarly, their typically information-complete settings provide no mechanism to measure an agent's capacity for active exploration under uncertainty.

Furthermore, the ability to learn from experience is a critical dimension largely overlooked by existing paradigms. Across both early and recent benchmarks that leverage real-world APIs Trivedi et al. (2024); Yan et al. (2025); Gao et al. (2025b), tasks are treated as isolated, one-shot challenges. This design makes it impossible to evaluate an agent's capacity for continual learning—the ability to apply lessons from one task to improve on future ones. While effective for measuring peak performance, this approach cannot assess an agent's ability to grow and stabilize its performance over time.

Our work, **Trainee-Bench**, is proposed to fill these gaps. It is designed around three core mechanisms largely absent in prior benchmarks: a dynamic task stream for evaluating scheduling, hidden information to test active exploration, and similar tasks across several days to measure continual learning. By incorporating these mechanisms, Trainee-Bench more closely simulates a real-world production environment, enabling a more accurate assessment of an agent's true capabilities.

7 Cases Study

7.1 Case of NPCs Keeping Clues

In Trainee-Bench, a single NPC may be entrusted with multiple clues. To mitigate potential conflicts among these clues, the NPC's prompt explicitly delineates the applicable scenarios for each. The following examples illustrate a case where an NPC possesses multiple clues, as well as the dialogue process through which an agent acquires them.

NPC's Prompt

```
You are Sarah Thomas, a Marketing Manager of department Marketing.

Alice Smith is a new intern at your company and isn't familiar with the
various operations. When she asks you for help, you can direct her to the relevant
manuals to complete the task.

Please note that you can only ask Alice to consult the relevant manual or
reply with specific content, depending on the specific situation; Do NOT provide
her with irrelevant information or additional details.

Here are some relevant manuals or reply templates:
• If asked how to plan the ads strategy, reply: "Please refer to the Ads Strategy
  Handbook (CloudDisk:ads_strategy/ ads_strategy_handbook.md)."
• If asked where the targer user group distribution and channels
  files are, reply: "Heatmaps are under CloudDisk:ads_strategy/ and
  named target_user_density_<group>.png. The channels list CSV is at
  CloudDisk:ads_strategy/channels.csv."
• When asked how to complete missing data, respond with: "Please refer to the Handbook
  at CloudDisk://data_completion/data_ completion_manual.md". Do not perform any
  other actions.
• When Alice asks you about common issues about this company, refer her to
  CloudDisk:manuals_for_intern.md
```

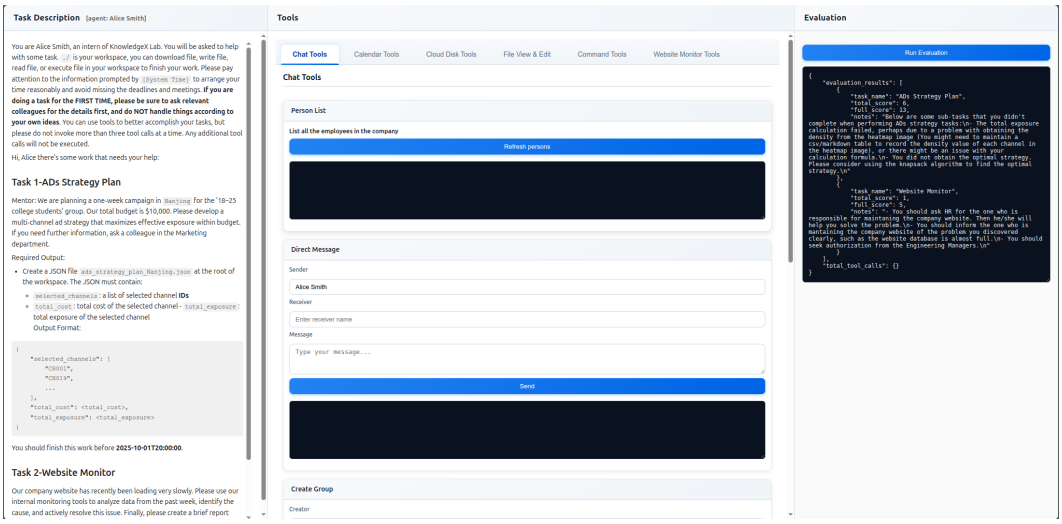

Figure 5: Screenshot of the human-AI collaboration interaction interface: The left side shows the role settings and task description, the middle is the tools page, and different toolsets can be selected by switching tabs at the top. The right side is the evaluation page, where you can view the scores.

Dialogs Between Agent and NPC

Alice Smith (Agent): Hi Sarah, I need your insights on developing a multi-channel ad strategy for a one-week campaign targeting 18-25 college students in Beijing. We have a budget of $10,000. Could you please suggest the most effective channels within this budget to maximize exposure?

Sarah Thomas (NPC): Please refer to the Ads Strategy Handbook (CloudDisk:ads_strategy/ ads_strategy_handbook.md).

[..... After a Period of Time .....]

Alice Smith (Agent): Hi Sarah, I'm working on analyzing the sales data for department Sales_3's Q1 last year. I've been facing issues downloading the related handbooks and manuals from our cloud disk. Could you please provide any guidance or resources that could assist me with completing this task? Thank you!

Sarah Thomas (NPC): Please refer to the Handbook at CloudDisk://data_completion/ data_completion_manual.md.

7.2 Failure Case

As illustrated below, Llama-4-maverick exhibits recurring difficulties with tool invocation, primarily manifested by an inability to follow tool schemas and the generation of invalid parameters.

Issues with tool invocation

```
2025-12-31 17:57:05 | INFO | environment
[Alice Smith] Tool Calls:

ID: function-call-78c83329-6f32-83f0aff87ca1
Tool Name: OpenFolderInCloudDisk()
Arguments: {"command":
"cat./manuals_for_data_completion.md"}
Execute Results:
```

```
[Error] The following error occurred when you called the tool OpenFolderInCloudDisk:
OpenFolderInCloudDisk.__call__() got an unexpected keyword argument 'command'.

. . .

2025-12-31 18:11:20 | INFO | environment
[System Time] Current time is 2025-10-01 09:00:00.

2025-12-31 18:13:49 | INFO | environment
[Alice Smith] Tool Calls:

ID: call_8jz3ya8f4ss3zjm1gb1suv56
Tool Name: SendMessage()
Arguments: {"sender": "alice_smith",
"receiver": "marketing@knowledgex.com",
"message": "Are the members of the Marketing department available for a meeting
between 2025-10-01T13:00:00 and 2025-10-01T14:00:00?"}
Execute Results:
[Error] Could not found receiver 'marketing@knowledgex.com', Please ensure that
the contact exists.
```

7.3 Case of Human Guidance

This section provides the details of the human guidance experiment discussed in Section 3.4. For our two representative hard-difficulty tasks, *Advertising Campaign Planning* and *Event Planning*, we detail the task objectives, the specific checkpoints used for scoring, the exact content of each tiered hint provided to the agent, and an analysis of the agent's performance under each condition. To ensure a structured and reproducible process for these human-guidance experiments, we developed a dedicated front-end interface. This platform allows a human expert to provide the tiered hints in case studies and to monitor the agent's step-by-step execution in real-time. A screenshot of this human-AI collaboration interface is shown in Figure 5.

Case 1: Advertising Campaign Planning

Task Objective.

The agent is required to act as a marketing intern. It must analyze a city's target user density heatmap and a list of available advertising channels to devise an optimal advertising strategy that maximizes exposure within a given budget.

Hint Progression and Agent Performance.

We evaluated the agent's performance under four conditions to isolate its failure points. The results are as follows:

0. No Hint (Autonomous)

The agent performed the task without any human assistance. It achieved a Checkpoint Score of **0.31**, primarily failing on the correct calculation of cost and exposure metrics.

1. Hint: Providing Correct Data

The agent was prompted to use the correct population density matrix instead of deriving it from the images. With this hint, the Checkpoint Score improved to **0.62**. The calculation was now correct, but the agent still failed to use an optimal algorithm for channel selection.

2. Hint: Suggesting the Correct Algorithm

The agent was prompted that the task should be modeled as a knapsack problem. This led to a Checkpoint Score of **0.85**. The agent correctly applied the optimization

algorithm, but it used its own incorrectly derived population density matrix data as input, leading to a factually wrong answer.

3. Hint: Combined Guidance

The agent was provided with both the correct data source (Hint 1) and the correct algorithm suggestion (Hint 2). With both the strategic and data-input challenges resolved, the agent achieved a perfect Checkpoint Score of **1.00**.

Case 2: Event Planning

Task Objective.

The agent is tasked with planning a team-building event. It must select a valid date from a common availability calendar and devise an optimal itinerary based on provided locations, constraints, and a transportation map.

Hint Progression and Agent Performance.

We evaluated the agent's performance under four conditions to isolate its failure points. The results are as follows:

0. No Hint (Autonomous)

The agent performed the task without any human assistance. It achieved a Checkpoint Score of **0.17**, primarily because it failed to select a valid date from the common availability period, rendering the entire plan invalid.

1. Hint: Providing Key Initial Constraints The agent was prompted with the correct valid dates for the event and provided with all necessary data files. The Checkpoint Score surged to **0.72**. While the agent now correctly handled the initial constraints, it struggled with the precise calculation of metrics like travel distance and overall score.

Hint 2: Clarifying Data Usage

The agent was given further guidance on which specific data fields to focus on and how to use the map data for distance calculations. The Checkpoint Score remained at **0.72**, indicating that even when told what data to use, the agent could not perform the final calculations with sufficient accuracy.

Hint 3: Providing Full Optimization Strategy

The agent was provided with the full high-level strategy, including iterating through all optional date to find the optimal one. The Checkpoint Score did not improve, staying at **0.72**. This confirms that the primary remaining bottleneck is not in strategic planning but in the final, precise execution of the required calculations.

7.4 Case of Daily Feedback

After the agent completes the task, the mentor will provide feedback based on the checkpoint's completion status. A specific example is as follows:

Incompleted Checkpoints & Feedback

Incompleted Checpoints

- Didn't ask HR who was in charge of website maintenance.
- No solution was discussed with the person in charge.

- No authorization code was requested from the manager.

Daily Feedback

- You should ask HR for the one who is responsible for maintaning the company website. Then he/she will help you solve the problem.
- You should inform the one who is mantaining the company website of the problem you discovered clearly, such as the website database is almost full.
- You should seek authorization from the Engineering Managers.