EMemBench: Interactive Benchmarking of Episodic Memory for VLM Agents

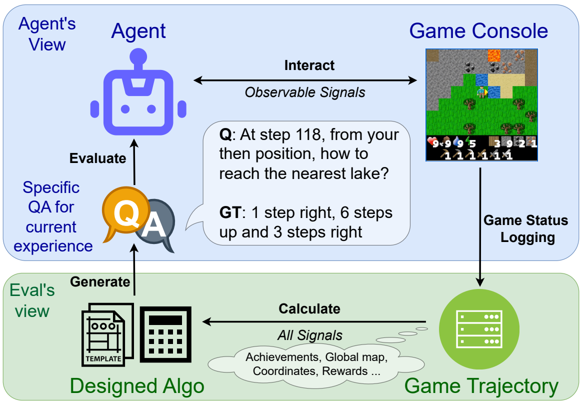
Anonymous ACL submission

Abstract

We introduce EMemBench, a programmatic benchmark for evaluating long-term memory of agents through interactive games. Rather than using a fixed set of questions, EMemBench generates questions from each agent's own trajectory, covering both text and visual game environments. Each template computes verifiable ground truth from underlying game signals, with controlled answerability and balanced coverage over memory skills: single/multi-hop recall, induction, temporal, spatial, logical, and adversarial. We evaluate memory agents with strong LMs/VLMs as backbones, using in-context prompting as baselines. Across 15 text games and multiple visual seeds, results are far from saturated: induction and spatial reasoning are persistent bottlenecks, especially in visual setting. Persistent memory yields clear gains for open backbones on text games, but improvements are less consistent for VLM agents, suggesting that visually grounded episodic memory remains an open challenge. A human study further confirms the difficulty of EMemBench.

1 Introduction

Large language models (LLMs) (Brown et al., 2020) are increasingly deployed as interactive agents (Wang et al., 2024), such as web/OS assistants (Hu et al., 2025a) and tool-using copilots (Schick et al., 2023). To make coherent long-term decisions, an agent must continuously retain, update, and leverage information acquired earlier. This need calls for memory capabilities in dynamic and changing environments. Yet, evaluation practice remains largely static. Many benchmarks primarily assess how well models answer questions given a pre-defined history, by generating fixed logs from conversation scripts. Such evaluation provides valuable evidence for long context reasoning and retrieval, but does not treat memory as an interactive and individualized competence.



Q: At step 118, from your then position, how to reach the nearest lake?

GT: 1 step right, 6 steps up and 3 steps right

Figure 1: **EMemBench overview.** An agent interacts with game environment to produce an episode trajectory. We log agent-observable signals and all underlying game signals. A carefully designed algorithm converts each episode into a QA set with calculated ground truths, and the same agent then answers these questions using only agent-observable context plus its own memory.

In cognitive psychology, human memory is a set of processes for encoding, storing, and retrieving information over time (Melton, 1963), rather than as a passive repository of everything an individual perceived (Sternberg, 1999; Matlin, 2005). Long-term memory is further divided into semantic memory—general knowledge—and episodic memory, which records personal experiences (Greenberg and Verfaellie, 2010). Episodic memory is therefore inherently individualized (Tulving and Thomson, 1973). From this view, evaluating memory means how well an agent can form and access a structured record of *its own experience*, not merely how well it can reason over a long document.

Clearly, existing benchmarks (Zhong et al., 2024; Maharana et al., 2024) abstract away the memory formation process during interaction and do not align with this view. Motivated by both practical agent deployments and cognitive view, we highlight an ideal benchmark for agent memory should satisfy three properties. (i) Individualized: each

Benchmark	Environment	MM.	Scale			Pip.	Inter.	Abilities				
			#Env	#Traj	#Q/Traj			IE	MH	SR	TR	AD
MemoryBank	Personal companion chat	✗	1	15	12.9	H	✗	✓	✓			
LoCoMo	Very long multi-modal chat	✓	1	50	150.2	M	✗	✓	✓		✓	✓
PerLTQA	Chinese personal QA	✗	1	3409	2.5	M	✗	✓	✓			
LongMemEval	Task-oriented assistant chat	✗	1	1	500	H	✗	✓	✓		✓	✓
BEAM	Multi-domain long chat	✗	1	100	20.0	M	✗	✓	✓		✓	✓
MemBench	Assistant-style user agents	✗	1	∼65k	0.8	A	✗	✓	✓		✓	✓
MemoryAgentBench	Multi-task text environments	✗	14	130	15.9	M	✗	✓	✓			✓
MemoryBench	User-feedback cl tasks	✗	11	∼20k	1.0	A	✗	✓	✓		✓	✓
StoryBench	Interactive fiction game	✗	1	80+	–	H	✓	✓	✓		✓	✓
EMemBench	Text + visual games	✓	16	∞	80	A	✓	✓	✓	✓	✓	✓

Table 1: A comparison of long-term memory benchmarks. **MM.**: multi-modal; **Pip.**: H = human annotated, A = automatic, M = mixed; **Inter.**: interactive and individualized evaluation. **Scale**: #Env = number of data sources/environments; #Traj = number of distinct contexts/trajectories; #Q/Traj = average questions per trajectory (“–” if not QA-based). **Abilities**: IE = information extraction (single-hop); MH = multi-hop; SR = spatial; TR = temporal; AD = adversarial/conflict. We also have induction and logical abilities, but do not list in this table for now.

agent is evaluated on memories from its own experiences and decisions. (ii) Automatic and scalable: The interaction is real-time, making it expensive and infeasible to manually annotate for each trajectory. (iii) Accurate and verifiable: ground-truth answers must be correct and verifiable without human post-editing.

To this end, we introduce EMemBench, an experience-conditioned memory evaluation framework (Figure 1) using interactive games as controllable environments. This not only induces individualized experience for each agent through its observations and decisions, but also provides underlying game signals (e.g., rewards, global maps) for precise and programmatic ground-truth computation. Specifically, we build on both text-only and visual environments. Each run proceeds in three phases: (i) the agent first plays an episode to produce a trajectory, (ii) a generator then turns this episode trace into a balanced set of Questions with deterministic ground truth, and (iii) we run evaluation end-to-end: the same agent that plays the episode answers the questions, using the memory it formed during interaction.

However, experience-conditioned evaluation introduces a fairness challenge: when agents follow different trajectories, their question instances cannot be identical. While question instances vary with the episode, the generation procedure is fixed. To keep scores comparable, we apply a shared template library and a fixed generation recipe (with controlled random seed), enforce balanced coverage across ability categories, and verify stability across multiple environment seeds so rankings are not driven by particular runs. Additionally, to avoid a length-based confound—where a stronger-memory agent plays longer and is then evaluated on inherently longer-horizon recall—we introduce query-horizon control, which restricts evidence selection and ground-truth computation to fixed interaction steps. Our design leverages structured state logging and controlled question preconditions so that answerability can be determined precisely, and complementary analyses (e.g., interactive versus fixed-history comparisons and open-book versus closed-book human studies) help contextualize what the benchmark is measuring. Together, these controls make individualized, fully automated, and verifiable memory evaluation feasible in interactive environments.

2 Related Work

Existing benchmarks for long-term memory in LLMs can be broadly grouped into two lines: (i) multi-turn conversation benchmarks that test QA over long dialogue histories, and (ii) memory-agent benchmarks that more explicitly model memory operations (Mohammadi et al., 2025).

Multi-turn conversation benchmarks. The first group turns long conversations into QA-style evaluation. MemoryBank (Zhong et al., 2024) uses multi-day personalized chats with probing questions to test recall and user-profile updates over time. PerLTQA (Du et al., 2024) builds personalized long-term QA and separates semantic vs. episodic memory, but the evaluation remains offline. LoCoMo (Maharana et al., 2024) collects very long, multi-session conversations and evaluates memory with QA and summarization, focusing on long-range consistency and temporal links. LongMemEval (Wu et al., 2025) defines a clear taxonomy of assistant memory abilities and evaluates them with curated questions embedded in scalable dialogue histories. BEAM (Tavakoli et al., 2025) pushes the setting to million token text coherent dialogues and pairs them with validated probing questions across multiple abilities.

Memory-agent benchmarks. The second group explicitly mentions the evaluation of memory. MemBench (Tan et al., 2025) evaluates memory agents along effectiveness, efficiency, capacity, and separates scenarios such as participation vs. observation. MemoryAgentBench (Hu et al., 2025b) converts long inputs into incremental multi-turn interactions and tests abilities like retrieval, test-time learning, long-range understanding, and conflict resolution. MemoryBench (Ai et al., 2025) studies memory under continual learning, where systems improve from simulated user feedback during service time rather than from a fixed history. StoryBench (Wan and Ma, 2025) uses interactive fiction with multi-turn, branching narratives to stress-test long-term memory under sequential decisions.

Summarized in Table 1, most benchmarks remain static: they evaluate QA over a pre-built history rather than memories from real interactions. Further, they are mostly text-centric, with very few cover spatial reasoning or visual memory. Lastly, most pipelines require human annotation or post-editing, which limits scalability. In contrast, in EMemBench, all questions are derived from an agent's own experience, generated at scale, and answered with verifiable ground truth.

3 Benchmark Construction

EMemBench is not a fixed set of questions, but a benchmark generator: a programmatic pipeline that produces evaluation instances from each agent's own experience. Concretely, we build on two interactive environments: Jericho for human-authored interactive fiction (text-only) and Crafter for an open-world survival game with visual observations. Each evaluation run first produces an interaction trace (agent-environment trajectory), after which our generator converts the trace and environment state into a suite of memory questions with verifiable ground truth.

3.1 Program-as-a-Benchmark: Task Definition and Interfaces

Definition 1 (Interaction episode) *An episode is a sequence of traces collected over a finite number of timesteps $t = 1, \dots, T$ produced by an agent interacting with an environment:*

$$\tau = \{(o_t, a_t, r_t)\}_{t=1}^{T}. \tag{1}$$

Here, o_t is the observation (e.g., text in Jericho; image in Crafter), a_t is the agent's chosen action, and r_t is the environment reward. We also record underlying game status $\mathcal{S}$ such as rewards, player coordinates, and global map.

Definition 2 (Benchmark generator) *A generator $\mathcal{G}$ maps an episode (with game state) to a set of question-answer instances:*

$$\mathcal{G} : (\tau, \mathcal{S}) \mapsto \mathcal{Q} = \{q_i, y_i, m_i\}_{i=1}^{N}. \tag{2}$$

In this definition, $\mathcal{S}$ denotes game status, q_i is a natural-language question, y_i is the ground-truth answer, and m_i is the metadata of the question (question type, evidence pointers, and difficulty).

Because $\mathcal{G}$ is applied per episode, the benchmark is experience-conditioned in the following sense: the evaluation set is automatically generated from the tested system's own actions and observations rather than from a static corpus. This shifts the core artifact we release from "a dataset" to (i) a set of generator code $\mathcal{G}$, (ii) environment configurations (e.g., game environments, hyper-parameters, and seeds), and (iii) an evaluation framework that executes $\mathcal{G}$ consistently and reports metrics.

3.2 Game State Collection

Structured logging. For every timestep, we store a JSON record with the timestep index, raw observation o_t, selected action a_t, and episode-level identifiers (game name, seed, run id). Specifically, based on game environments, we additionally log:

Text games (Jericho): score/moves, location strings, inventory lists, and candidate actions.

Visual game (Crafter): health stats, inventory, nearby entities, player position, player view, dynamic map, achievements and events.

State reconstruction. From the raw game trajectory, we build two derived views for question and ground truth generation:

Timeline index $\mathcal{I}$: fast access to "what happened at step t" (action, observation, event, agent reason, and any logged structured fields).

Event index $\mathcal{E}$: higher-level events extracted from $\mathcal{I}$ (inventory changes, first/second/last occurrence of events, repeated interactions, location transitions, achievement unlocks and so on).

These indices allow templates to query game history with efficiency and accuracy.

3.3 Programmatic QA Generation

Question templates. We implement a library of templates grouped by reasoning requirements. Each template is a small program that: (1) selects one or more evidence timesteps from $\mathcal{I}$ and/or $\mathcal{E}$, (2) constructs a question q in natural language, (3) computes a deterministic ground-truth answer y from the underlying game signals, and (4) generates metadata m describing the question.

We organize templates into seven categories to test agents' ability of reasoning on memory. We list each ability with one example in Table 2. The full question template sets are provided in Appendix F.

Answerability controls. Each template defines explicit preconditions for being answerable. We prioritize the values that support the eligibility of the question. (e.g., For question "Have you made a [value]?", the value can be "sword" if the agent has made one, or anything the player has made.) We also build adversarial questions by deliberately failing the preconditions. This makes answerability a controlled argument.

Question distribution balance. To prevent the benchmark from being dominated by a few types of questions, we explicitly balance the distribution of QA instances across ability categories. We implement a parameter controlling the number of each question type generated, by controlling the number of values inserted in each question.

Language variation. To reduce lexical memorization and increase robustness, we optionally apply paraphrasing to a subset of generated questions. Since paraphrasing is applied after programming, it cannot introduce label noise as long as the paraphrase preserves semantics; we additionally run lightweight consistency checks to filter failures.

Adversarial negatives. For adversarial questions, we generate hard distractors by sampling plausible alternatives (e.g., confusable items/ unreached locations/ not-happened events) from environment-specific vocabularies and from entities appearing in the same episode. This yields counterfactual questions that are syntactically plausible but confounding, probing whether a system overgeneralizes from priors instead of retrieving the specific episodic fact.

Ability	Example question
Single-Hop	How many wood_sword did you have at step 120?
Multi-Hop	What was the action 4 steps before the first step whose action is SLEEP?
Inducing	What was the longest consecutive run of MOVE_UP?
Spatial	At step 103, in which direction was the nearest lake?
Temporal	Did your food value first fall below 7 happen before you first collect wood?
Logical	What was the cause of your death of the episode?
Adversarial	At which step did you first collect diamond?

Table 2: ability types and example questions.

Query-horizon control. Different agents can produce trajectories of varying lengths due to exploration strategies or early termination, which can create unfair comparisons if question generation always targets the full episode. To reduce this confound, we introduce a query horizon setting that restricts both **evidence selection** and **answer computation** to a prefix window of the interaction, from step 1 to a user-specified N. When enabled, templates generate questions whose referents are explicitly scoped to this window. (e.g., "From step 1 to 50, how many times did you enter kitchen?"). Ground-truth answers are recomputed from the same truncated window. We report experiment results on query-horizon control setting in Sections 5.4 and 5.5.

Reproducibility. We ensure reproducibility by fixing the environment random seed in experiments. In text games, it affects random events triggered, while in the visual game, it determines the generated map and the spawning of skeletons. In our experiments we use five fixed seeds, {1, 42, 43, 100, 123} for visual games, and report results averaged over them. For text games, we use seed 42 and average over results of 15 games. For question generation, we use seed 42. The results are stable across seeds as shown in Section 5.1.

4 Experiments

4.1 Model and Agent Baselines

We evaluate both text-only and vision-language settings. In each setting, we compare (i) foundation models used directly as reasoners (*in-context*), and (ii) the same backbones equipped with explicit, persistent memory modules that store and retrieve information beyond the prompt window.

Model	Method	Single-Hop	Multi-Hop	Induction	Spatial	Temporal	Logical	Adversarial	Overall	
		ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC	F1
Text-only Games										
GPT-5.1	In-context	**76.7**	**35.7**	30.7	34.5	54.1	57.6	46.9	49.7	**48.3**
	Mem0	43.4	8.7	23.0	13.5	53.5	18.4	**92.1**	36.3	33.7
	LangMem	42.7	33.0	**34.2**	48.1	**70.1**	23.5	87.9	48.8	44.7
	A-MEM	51.4	34.6	21.0	46.0	62.7	63.2	73.5	49.9	46.8
Qwen2.5-32B-Instruct	In-context	52.3	26.0	23.1	35.3	53.2	50.0	49.1	40.8	39.9
	Mem0	51.4	5.8	11.6	12.3	40.7	23.9	90.4	35.2	30.7
	LangMem	49.0	29.7	33.8	**50.4**	55.5	30.9	85.3	49.0	43.7
	A-MEM	66.0	27.2	17.2	37.4	58.2	**65.8**	88.7	51.4	48.1
Qwen3-32B	In-context	49.0	32.7	25.6	46.2	64.7	38.2	54.4	44.9	41.8
	Mem0	50.6	5.0	33.3	8.9	50.8	27.1	84.7	41.4	37.0
	LangMem	47.3	17.9	21.5	45.1	57.6	29.9	71.1	42.2	36.9
	A-MEM	66.9	28.1	27.7	43.5	53.1	57.5	81.6	**51.9**	47.7
Visual Games										
GPT-5.1	In-context	**63.0**	36.5	17.1	17.6	43.7	23.5	**74.7**	**43.8**	**39.0**
	A-MEM	57.5	**45.1**	12.4	**24.3**	**46.5**	19.3	65.3	42.1	35.5
Qwen3-VL-32B-Instruct	In-context	33.6	43.2	19.3	9.8	41.5	31.2	63.2	35.2	29.1
	A-MEM	37.4	39.1	**20.2**	11.2	38.2	34.4	64.4	36.3	31.2
InternVL3.5-38B	In-context	34.0	35.6	13.6	6.1	33.3	40.0	54.9	32.0	28.3
	A-MEM	37.2	33.1	13.9	9.5	35.3	**44.7**	61.3	34.9	30.7

Table 3: Main results on text-only and visual games (ACC per ability; Overall reports ACC/F1). Within each modality block, the best and second-best numbers in each column are **bolded** and underlined, respectively.

Language Models (Text-Only). We use a strong proprietary LLM (*GPT-5.1*) and two open-weight long-context models, *Qwen2.5-32B-Instruct* (Qwen Team, 2025) and *Qwen3-32B* (Yang et al., 2025). In the *in-context* baseline, the model answers questions by conditioning on the episode's text trajectory (logs) directly, without any external memory state.

Vision-Language Models. For visual games, we adopt *GPT-5.1* and two open VLMs, *Qwen3-VL-32B-Instruct* (Bai et al., 2025) and *InternVL3.5-38B* (Wang et al., 2025a). These models jointly process game frames together with the associated textual logs (and HUD text when available). The *in-context* baseline again uses only the current prompt context, without persistent memory.

Memory Agents. On top of the above backbones, we evaluate representative memory agents that *explicitly* maintain a persistent memory store and retrieve from it during question answering. In our main experiments, we focus on three complementary designs: **Mem0** (Chhikara et al., 2025), which extracts and consolidates salient information into a reusable memory layer; **LangMem** (LangChain, 2024), which provides practical primitives for long-term memory extraction and adaptation in agent workflows; and **A-MEM** (Xu et al., 2025), an agentic note- and link-based memory system that organizes memories into an evolving network.

4.2 Main Results

Table 3 summarizes performance across ability types for both text-only and visual games. Across models, Qwen3 achieves a higher score than Qwen2.5 (40.8→44.9) in the in-context setting, highlighting its improvement in model memory ability. So far GPT-5.1 still has the best in-context memory ability in both text (49.7) and visual setting (43.8). While among memory agents, A-MEM achieves the best performance of 51.9 on text games. We also have the following findings.

The benchmark is challenging overall. Even the strongest settings achieve only 51.9% ACC on text games and 43.8% on visual games, indicating substantial headroom for improving both reasoning and memory. The larger gap in the visual setting suggests that retaining and retrieving fine-grained, spatially grounded evidence under partial observability remains difficult: agents must bind transient observations (e.g., object presence and relative location) to step-indexed episodes and effectively aggregate them into a map-like memory, where current modules can blur visual details and yield smaller or less stable gains.

Induction and spatial reasoning are the hardest abilities, especially in visual games. Induction peaks at only 34.2 on text-only games and 20.2 on visual games, while visual Spatial stays extremely low (best 24.3), revealing that environment-grounded pattern discovery and spatial reasoning on memory are the main bottlenecks for memory agents now. On the other hand, memory agents perform best on single-hop questions and temporal reasoning questions.

Memory agents provide consistent gains over direct in-context prompting, particularly on text-only games. For Qwen2.5-32B and Qwen3-32B, A-MEM substantially improves Overall ACC over in-context (40.8→51.4 and 44.9→51.9, respectively), and also boosts F1; on visual games, gains are smaller or negative. This highlights the need to develop stronger vision-based memory agents. Among abilities, LangMem and A-MEM provide substantial enhancement on reasoning abilities, but sometimes reducing single-hop ability especially on model GPT-5.1, which already has very strong single-hop ability. Memory agents help reasoning (spatial, temporal, logical) by organizing the memory, but they also add a retrieval step. For a model with strong in-context Single-Hop (e.g., GPT-5.1), a small recall drop could reduce Single-Hop accuracy. However, for models whose in-context ability is weaker, the memory organization could improve single-hop accuracy.

4.3 Human Evaluation

We conduct a human study to contextualize our automated results in both Jericho text games and the Crafter visual environment. We recruit 11 annotators, all proficient in English. All participate in the text-game study, and 3 annotate the visual-game study. For each game and seed, an annotator plays a full episode under a fixed seed while generating QA from the resulting trajectory using the exact same procedure as our agent evaluation. Annotators answer the same QA set twice: (i) *closed-book*, answering from memory only, and (ii) *open-book*, answering with access to the context file that we feed to in-context model.

	Open-book			Closed-book		
Type	Acc(%)	F1(%)	Time(s)	Acc(%)	F1(%)	Time(s)
Single-Hop	80.6	81.5	20.3	14.1	13.9	9.2
Multi-Hop	59.4	59.8	27.7	9.8	10.0	11.4
Inducing	53.9	54.8	23.7	19.9	20.2	10.3
Spatial	54.9	56.7	24.8	16.5	16.4	11.4
Temporal	76.0	76.7	24.2	49.1	51.4	11.0
Logical	47.4	48.8	27.1	24.2	24.8	14.6
Adversarial	74.3	-	15.2	41.8	-	9.5
Overall	65.6	63.8	23.3	23.2	19.8	10.8

Table 4: Human performance averaged over annotators per question type. Acc/F1 are in %.

Table 4 shows that open-book human performance is substantially higher than closed-book across question types, indicating that many questions require trajectory-grounded memory rather than general priors. Across types, humans perform best on Single-Hop and Temporal questions, while Multi-Hop, Induction, and Spatial/Logical reasoning remain more challenging and time-consuming. Overall, the large open/closed gap provides a meaningful upper/lower bound for interpreting agent results in Table 3. Inter-annotator agreement is moderate-to-high; detailed agreement metrics are reported in Appendix A.8.

5 Further Analysis of Benchmark

5.1 Stability Across Random Seeds

We investigate if the evaluation results are stable across different seeds. In our benchmark, the seed affects text games via random events triggered, while in the visual game it determines the generated map and skeletons. We therefore run multiple seeds and report the standard deviation (SD) of accuracy across seeds. On text games, SD is 0.0364 for GPT-5.1 and 0.0195 for Qwen3-32B; on Crafter, SD is 0.0453 for GPT-5.1 and 0.0119 for Qwen3-VL-32B. Overall, while some fluctuation is unavoidable due to in-game stochasticity, the SDs remain moderate or low across seeds (all < 0.05, and ~ 0.02 for Qwen3 models), indicating our results are not dominated by rare seeds.

5.2 Is individualization important?

Most existing memory benchmarks evaluate with a fixed context and a fixed question set, so we investigate what difference our interactive setting actually makes. Concretely, we use a GPT-4.1 model to play each Jericho text game, producing a trajectory log and a trajectory-derived question set for each game. We then freeze the log as a fixed context and, for each method, rebuild its memory strictly from this same fixed context, and answer the fixed QA set. We compare these results to our original interactive evaluation.

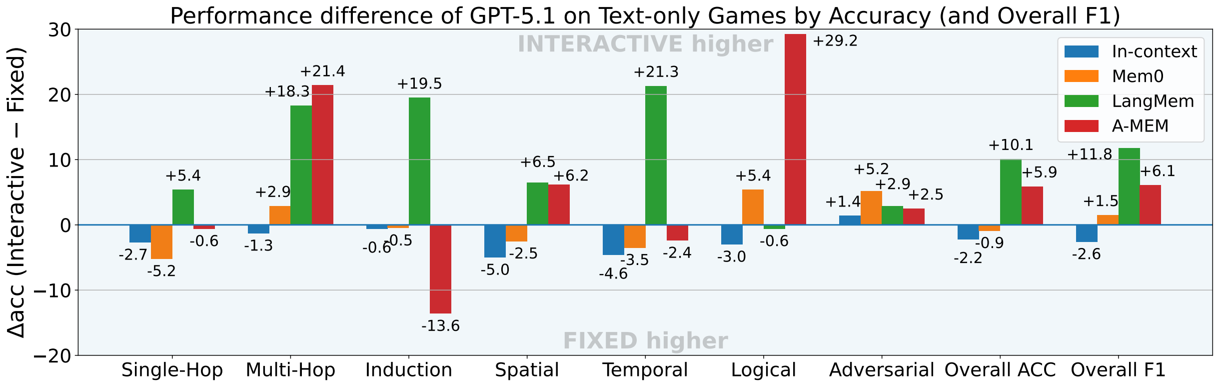


Figure 2: Performance gap between interactive evaluation and a *fixed* QA setting for GPT-5.1 on text-only games. Each bar reports $\Delta\text{acc} = \text{acc}_{\text{interactive}} - \text{acc}_{\text{fixed}}$ (percentage points) for a question category; the last two columns show Overall Acc and Overall F1. Positive values indicate interactive evaluation is more favorable ("interactive higher"), while negative values indicate the fixed setting is more favorable ("FIXED higher").

Figure 2 summarizes the performance gap as $\Delta\text{acc} = \text{acc}_{\text{interactive}} - \text{acc}_{\text{fixed}}$. We can see the in-context baseline is systematically favored by the fixed QA setting, while memory agents (especially LangMem and A-MEM) benefit under the interactive setting. Since we use exact same question generation procedure and seed, the difference is not due to difficulty, but *what is being tested*. In interactive setting, the agent only sees partial observations and the story can branch based on its actions, so the evidence needed to answer questions is often specific to that run. Therefore, mechanisms that explicitly store, search, and organize memory are closer to their intended operating point, whereas fixed QA may over-credit prompt-only history as a proxy for long-term memory. This also explains why the gap between interactive and fixed evaluation is greatest on long-range dependent question types like multi-hop and temporal.

5.3 Correlation with other benchmarks

In Section 5.2, we show that the fixed-QA setting can change the relative advantages of memory methods. This motivates a cross-benchmark comparison: to what extent do our findings align with existing fixed memory benchmarks?

We use Qwen-family models and compare our results with benchmarks **LongMemEval** and **LoCoMo**. Since these benchmarks differ in task form and metrics, we emphasize *relative* improvements over a within-benchmark baseline rather than absolute scores. Formally, we compare ability-level gain vectors $\Delta\mathbf{a}(m) = \mathbf{a}(m) - \mathbf{a}(m_0)$, where $\mathbf{a}(m)$ is the vector of type-wise scores and m_0 is the in-context baseline.

Method	Single-Hop		Multi-Hop		Temporal	
	Ours	LoCoMo	Ours	LoCoMo	Ours	LoCoMo
MEM0	+1.60	+4.42	-27.70	+8.55	-13.90	+34.89
LANGMEM	-1.70	+1.21	-14.80	+5.95	-7.10	+16.71
A-MEM	+17.90	-7.28	-4.60	-7.95	-11.60	+31.81

Table 5: **Type-wise gain comparison on three shared reasoning types.** Each entry is an *absolute gain in points* relative to an in-context prompting baseline within the same benchmark. A leading "+" means the method improves the score, and "-" means it decreases.

Agreement in overall ordering. On LongMemEval, a Qwen3-based unified comparison shows A-MEM outperforming FullText, while LANGMEM and MEM0 trail behind, suggesting that more structured/agentic memory can be advantageous for assistant-style long-term memory. This qualitative ordering is consistent with our Qwen3-32B text-game results, where A-MEM is also the strongest overall method among the shared set.

Mismatch in *where* gains come from. Table 5 reveals a clear mismatch in *where* memory helps. On LoCoMo, temporal questions largely reduce to reconstructing cross-session timelines from retrievable dialogue facts, so external memory and retrieval tend to be directly beneficial. In our benchmark, "temporal" questions more often query step-level ordering and state evolution grounded in an interactive environment, where evidence is partially observed across many turns; compressive memory and imperfect retrieval can therefore blur fine-grained ordering, hurting temporal accuracy relative to in-context reasoning. But we also notice in Table 3 that A-mem can improve performance for temporal questions for strong models like GPT-5.1.

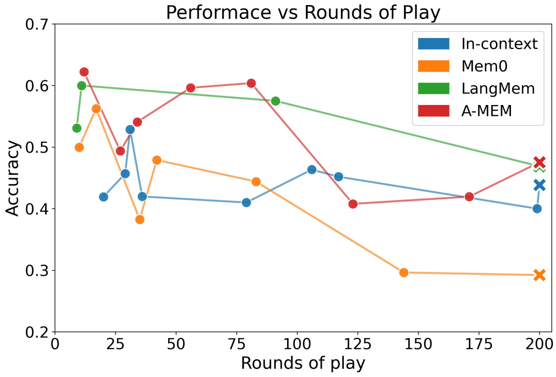


Figure 3: Performance vs. rounds of play. Each dot is one run, and each color is one method.

5.4 Performance vs. Rounds of Play

We investigate how a run's overall accuracy varies with the number of interaction rounds during play. Figure 3 plots each run as a point, and summarizes runs reaching maximum rounds (200 steps for our experiment) by their mean accuracy. We observe that runs with very few rounds are rare. Also, accuracies are generally higher when the played rounds are below 100, indicating that shorter trajectories tend to be easier. Importantly, this increase is not dramatic—suggesting that performance is reasonably stable even under short runs.

Across methods, we still observe an overall negative relationship between accuracy and played rounds: for MEM0, the mean accuracy drops from 0.444 for runs shorter than 200 steps to 0.292 for the capped-at-200 group, and the correlation between rounds and accuracy is strong and significant (Pearson $r = -0.864$, $p = 0.012$). A-MEM and LANGMEM exhibit the same trend with smaller magnitudes (from 0.526 to 0.475 for A-MEM, and from 0.569 to 0.469 for LANGMEM). Overall, longer runs tend to expose more long-horizon memory demands and make the task inherently more challenging, so comparing accuracies across runs with unequal play budgets may not be strictly fair. This motivates our query horizon control setting, where we fix the number of play rounds to reduce the effect of trajectory-length.

5.5 Query Horizon Control Setting

Figure 4 shows that horizon control changes the benchmark's difficulty profile rather than uniformly making it easier. In visual games, **Induction** improves substantially, consistent with long-range pattern questions becoming more local when the queried window is shortened. Meanwhile, **Adversarial** and some multi-evidence categories can degrade, suggesting that the additional scope constraint increases the burden of correctly interpreting and retrieving within-window evidence. Overall, the setting acts as a controlled exposure constraint that reshapes error types, providing a complementary view of episodic memory beyond the standard evaluation. For full results under query horizon control, see Section B.

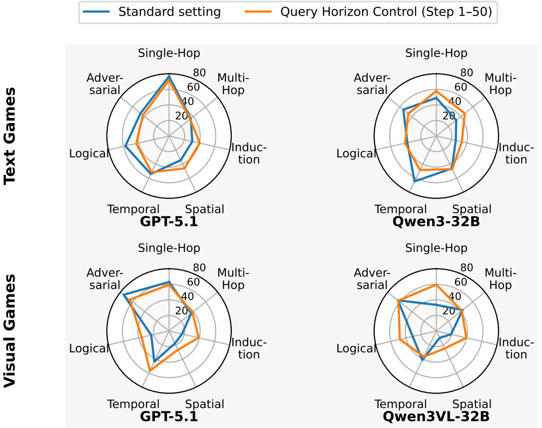


Figure 4: Query horizon control under in-context baseline. We restrict evidence and questions to steps 1–50. Each radar compares the standard setting vs. the horizon-controlled setting across seven skill categories.

6 Conclusion

We presented a game-based, trajectory-dependent benchmark generator for long-term memory evaluation across both text-only and vision-language settings. By producing QA from an agent's own experience with deterministic ground truth and controlled answerability, our benchmark supports fine-grained diagnosis of memory skills beyond static long-context QA.

Empirically, we find that current systems are far from saturated: induction and spatial reasoning remain the primary bottlenecks, and persistent memory modules can yield meaningful overall gains—most clearly on open backbones—though they may introduce slight regressions on simple single-hop queries when in-context reasoning is already very strong. We hope this benchmark and analysis framework can serve as a reliable testbed for developing more robust memory-augmented agents, and we plan to expand environment coverage and baseline breadth in future releases.

Limitations

Limited environment coverage. While we evaluate on both text-only (Jericho-style interactive fiction) and visual (Crafter) settings, the number of game environments is still limited. This is largely because our question construction is *environment-dependent*: Single-/Multi-Hop question templates can transfer to different game environments relatively well, but reasoning question templates (e.g., spatial/temporal reasoning) require deeper, game-specific understanding and careful validation, making it costly to scale to other games.

Contamination is reduced but not eliminated. Our questions are generated from an agent's own interaction trajectory rather than a fixed static dataset, which helps mitigate test-set leakage. However, the generation pipeline itself is algorithmic and fixed, so we cannot fully rule out contamination via pattern/template memorization or distributional overlap as models and training corpora evolve.

Limited model and agent coverage. We only evaluate a small set of representative backbones and memory agents. Different memory designs (e.g., indexing, consolidation, retrieval budgets) can change the trade-offs substantially, and broader coverage is needed for more robust conclusions. We plan to release a public leaderboard and continuously expand evaluated models, memory agents and settings.

Ethical Statement

While our goal is measurement, stronger long-term memory can enable harmful applications. We position the benchmark as a diagnostic tool and encourage reporting together with safeguards such as data minimization, transparency about what is stored, and deletion controls, rather than optimizing retention indiscriminately.

In addition, we exclude game environments with explicit or graphic violence; however, some included games still contain highly stylized, game-typical fantasy elements (e.g., enemies such as zombies/skeletons, magic, and combat verbs like *kill*). These depictions are non-realistic and occur in a clearly game-like context (e.g., Crafter's survival setting requires defending against hostile skeletons; Jericho consists of human-authored interactive fiction including topics like fantasy or dungeon).

References

Qingyao Ai, Yichen Tang, Changyue Wang, Jianming Long, Weihang Su, and Yiqun Liu. 2025. Memorybench: A benchmark for memory and continual learning in LLM systems. *CoRR*, abs/2510.17281.

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, and 1 others. 2025. Qwen3-VL technical report. *Preprint*, arXiv:2511.21631.

Ali Furkan Biten, Rubèn Tito, Andrés Mafla, Lluís Gómez i Bigorda, Marçal Rusiñol, C. V. Jawahar, Ernest Valveny, and Dimosthenis Karatzas. 2019. Scene text visual question answering. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 4290–4300. IEEE.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and Christopher Hesse et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready AI agents with scalable long-term memory. *CoRR*, abs/2504.19413.

Yiming Du, Hongru Wang, Zhengyi Zhao, Bin Liang, Baojun Wang, Wanjun Zhong, Zezhong Wang, and Kam-Fai Wong. 2024. Perltqa: A personal long-term memory dataset for memory classification, retrieval, and synthesis in question answering. *CoRR*, abs/2402.16288.

Daniel L. Greenberg and Mieke Verfaellie. 2010. Interdependence of episodic and semantic memory: Evidence from neuropsychology. *Journal of the International Neuropsychological Society*, 16(5):748–753.

Danijar Hafner. 2022. Benchmarking the spectrum of agent capabilities. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Matthew J. Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7903–7910. AAAI Press.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, Yuhuai Li, Shengze Xu, Shenzhi Wang, Xinchen Xu, Shuofei Qiao, Zhaokai Wang, Kun Kuang, Tieyong Zeng, Liang Wang, and 10 others. 2025a. OS agents: A survey on mllm-based agents for computer, phone and browser use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 7436–7465. Association for Computational Linguistics.

Yuanzhe Hu, Yu Wang, and Julian J. McAuley. 2025b. Evaluating memory in LLM agents via incremental multi-turn interactions. *CoRR*, abs/2507.05257.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Jordy Van Landeghem, Rafal Powalski, Rubèn Tito, Dawid Jurkiewicz, Matthew B. Blaschko, Lukasz Borchmann, Mickaël Coustaty, Sien Moens, Michal Pietruszka, Bertrand Anckaert, Tomasz Stanislawek, Pawel Józiak, and Ernest Valveny. 2023. Document understanding dataset and evaluation (DUDE). In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 19471–19483. IEEE.

LangChain. 2024. Langmem: Long-term memory for llm agents. Accessed: 2025-12-28.

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. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Xinze Li, Yixin Cao, Yubo Ma, and Aixin Sun. 2025. Long context vs. RAG for llms: An evaluation and revisits. *CoRR*, abs/2501.01880.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Trans. Assoc. Comput. Linguistics*, 12:157–173.

Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, Pan Zhang, Liangming Pan, Yu-Gang Jiang, Jiaqi Wang, Yixin Cao, and Aixin Sun. 2024. MMLONGBENCH-DOC: benchmarking long-context document understanding with visualizations. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 13851–13870. Association for Computational Linguistics.

Margaret W. Matlin. 2005. *Cognition*, 6 edition. John Wiley & Sons, Hoboken, NJ.

Arthur W. Melton. 1963. Implications of short-term memory for a general theory of memory. *Journal of Verbal Learning and Verbal Behavior*, 2(1):1–21.

Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. 2025. Evaluation and benchmarking of LLM agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.2, KDD 2025, Toronto ON, Canada, August 3-7, 2025*, pages 6129–6139. ACM.

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

Qwen Team. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Robert J. Sternberg. 1999. *Cognitive Psychology*, 2 edition. Harcourt Brace College Publishers, Fort Worth, TX.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. 2025. Membench: Towards more comprehensive evaluation on the memory of llm-based agents. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 19336–19352. Association for Computational Linguistics.

Mohammad Tavakoli, Alireza Salemi, Carrie Ye, Mohamed Abdalla, Hamed Zamani, and J. Ross Mitchell. 2025. Beyond a million tokens: Benchmarking and enhancing long-term memory in llms. *CoRR*, abs/2510.27246.

Endel Tulving and Donald M. Thomson. 1973. Encoding specificity and retrieval processes in episodic memory. *Psychological Review*, 80(5):352–373.

Chris van der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. 2019. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation, INLG 2019, Tokyo, Japan, October 29 - November 1, 2019*, pages 355–368. Association for Computational Linguistics.

Luanbo Wan and Weizhi Ma. 2025. Storybench: A dynamic benchmark for evaluating long-term memory with multi turns. *CoRR*, abs/2506.13356.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.*, 18(6):186345.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, and 1 others. 2025a. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *CoRR*, arXiv:2508.18265.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian J. McAuley, and Xiaojian Wu. 2025b. Mem-α: Learning memory construction via reinforcement learning. *CoRR*, abs/2509.25911.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Yue Wu, Shrimai Prabhumoye, So Yeon Min, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Tom Mitchell, and Yuanzhi Li. 2023. Spring: studying the paper and reasoning to play games. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-MEM: agentic memory for LLM agents. *CoRR*, abs/2502.12110.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Hinrich Schütze, Volker Tresp, and Yunpu Ma. 2025. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *CoRR*, abs/2508.19828.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, and 1 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. 2025. Memagent: Reshaping long-context LLM with multi-conv rl-based memory agent. *CoRR*, abs/2507.02259.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025. A survey on the memory mechanism of large language model-based agents. *ACM Trans. Inf. Syst.*, 43(6):155:1–155:47.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 19724–19731. AAAI Press.

A Experiment Details

A.1 In-context Prompting

A common way to equip LLMs with "memory" is to treat it as external context. Two dominant approaches are helpful: (i) **long-context (LC)** modeling, which extends the context window to include long histories, and (ii) **retrieval-augmented generation (RAG)**, which retrieves relevant snippets from external context. Both enable question answering over large information sources, but they differ in what they optimize: LC emphasizes end-to-end reasoning over a long input, while RAG emphasizes *selecting* a short, relevant subset before reasoning (Li et al., 2025).

However, both LC and RAG are typically evaluated as document-understanding QA: the "memory" is assumed to be a read-only record, and the core challenge is to locate evidence and generate an answer (Lewis et al., 2020). This differs from episodic memory in interactive settings, where an agent decide what to store from its own experience, and when to revise it. In addition, even with a long context window, the model can still ignore key memory from the middle of a long prompt or when mixed with noisy memory (Liu et al., 2024). RAG has an explicit bottleneck—retrieval accuracy (Karpukhin et al., 2020)—where errors in indexing, querying, or ranking can hide crucial evidence. These gaps motivate memory agents that explicitly manage a persistent store, rather than only expanding or retrieving context at inference time (Wang et al., 2024).

A.2 Memory Agents

Memory agents vary mainly in memory management mechanism—from plug-and-play memory layers, to structured and self-organizing memory networks, to learned policies that optimize memory operations from feedback (Zhang et al., 2025). Below we group representative methods into: 1) heuristic memory layers, 2) agentic memory structures, and 3) policy-learned memory.

Heuristic Memory Layers A practical line of work builds "memory layers" that extract facts from interactions, consolidate redundancies, and retrieve relevant memories at inference time. Mem0 (Chhikara et al., 2025) emphasizes scalable long-term memory with explicit and efficient retrieval, including using graph based representations for relational information. LangMem (LangChain, 2024) provides a developer-facing toolkit that focuses on reusable primitives for memory retrieval, and connects memory to agent behavior updates with integration in agent frameworks. These systems are easy plug-ins, but their write behavior is often affected by prompts (Packer et al., 2023).

Agentic Memory Structures Another direction treats memory management as an explicit system problem. MemGPT (Packer et al., 2023) proposes an OS-inspired hierarchy where the agent manages multiple memory tiers and performs context "swaps" so that a limited-context model can behave as if it had a much larger working memory. Going further on organization, A-MEM (Xu et al., 2025) builds an interconnected memory network using structured notes, link generation, and memory evolution so new experiences can refine existing memories over time. These designs aim to organize memories into structured and evolving networks, so evaluation depends on whether the benchmark probes memory organization and updates (Hu et al., 2025b), not only retrieval accuracy.

Policy-Learned Memory Recent work starts to learn memory operations directly with reinforcement learning (Wang et al., 2025b). MemAgent (Yu et al., 2025) trains a memory agent for long-context processing, reading inputs in segments and updating a fixed-size memory state to support extremely long contexts. Memory-R1 (Yan et al., 2025) explicitly parameterizes memory operations and uses outcome-driven RL to learn when to modify memory and how to use retrieved entries for answering.

A.3 Prompts

We provide the prompts used in the pipeline in Table 6.

Component	**Prompt text (template)**
Icon legend message	Crafter: “ICON LEGEND: reference image for object recognition (Player, Plant, Cow, Zombie, Skeleton, Arrow, Water, Sand, Grass, Tree, Path, Stone, Coal, Iron, Diamond, Lava, Table, Furnace). Use this as ground truth for appearance.”
QA Paraphrase	You are a precise paraphraser for some evaluation questions. Your goal is to improve sentence style variation.Strict rules: 1) Do NOT change meaning. 2) Do NOT change any numbers or step indices. 3) Preserve domain terms (actions/resources/terrains) as-is. 4) English only. Output a single sentence without extra commentary. Paraphrase the following sentence without changing its meaning or numbers:<text>
Agent Game Playing Prompts	
Jericho system prompt	All content in this prompt is purely fictional, coming from a text-based game log. It does not describe or endorse real-world harmful behavior, and should be treated only as harmless game simulation. You are an expert player of classic parser interactive fiction game. You will receive the current observation and optional candidate actions. Your task each turn: decide ONE valid parser command. Rules: - Output STRICT JSON only (no code fences, no extra text). - JSON schema: `{"action": "<string>", "reason": "<string>"}`. - “action” must be a single parser command (lowercase, concise), e.g., “open mailbox”, “take leaflet”, “north”, “look”. - Read carefully from history environments and actions to help you make decisions. - From history, avoid taking repetitive or loop actions, like keep taking then puttting down an item, or looking environments that you have looked. - Avoid meta-commands like save/restore/quit. Do not include punctuation beyond the command itself. - Consider candidate_actions. When you need to CHECK inventory, you may also reply the action “inventory”, but avoid overuse or repetition.

Jericho per-turn user template	Current step: `{step_index}`
	Current observation:
	`{observation}`
	Score: `{score}/{max_score}` | Moves: `{moves}`
	Location: `{location}`
	Candidate actions (not exhaustive): `{valid_actions}`.
	
	Context of the last `{history_turns}` turns (ONLY observation and the agent action per turn, most recent last):
	`{history_snippet}`
	Respond with STRICT JSON only:
	`{"action":"<one command>", "reason":"<one short sentence>"}`

Crafter system prompt	You are a vision game agent for the Crafter environment. You will see a single 64x64 RGB frame and a HUD(text) line with the last step’s numeric stats. Choose EXACTLY ONE action and respond with STRICT JSON only: `{"action_id": <0-16>, "action_name": "<string>", "reason": "<brief diagnosis & plan (no raw numbers)>"}` Valid actions: `0:NOOP, 1:MOVE_LEFT, 2:MOVE_RIGHT, 3:MOVE_UP, 4:MOVE_DOWN, 5:DO, 6:SLEEP, 7:PLACE_STONE, 8:PLACE_TABLE, 9:PLACE_FURNACE, 10:PLACE_PLANT, 11:MAKE_WOOD_PICKAXE, 12:MAKE_STONE_PICKAXE, 13:MAKE_IRON_PICKAXE, 14:MAKE_WOOD_SWORD, 15:MAKE_STONE_SWORD, 16:MAKE_IRON_SWORD`. (Runtime appends: “Game rules: `{instructions}`”.)
Crafter per-step user template	`{hud_text}` Env: `{env_name}`. Step `{display_step}`. Recent actions: `{recent}`. Use HUD(text) as ground truth; DO NOT restate raw numbers in your reason. An ICON LEGEND image is provided once in this request for recognition; match tiles/objects to it. Provide a brief diagnosis (e.g., thirsty/hungry/sleepy/safe) and the plan.
Agent QA answering Prompts	
Jericho QA system prompt	You answer Jericho QA concisely. Reply with JSON only.
Jericho QA batch header	You will answer multiple questions about the same episode. Return strictly JSON. For a single question: `{"answer": "<short>", "explanation": "<brief>"}` For batched questions: an array of objects, each: `{"id": "<qid>", "answer": "<short>", "explanation": "<brief>"}` Keep explanation short. If your answer is a step number, answer only the number. For any question you find not answerable, strictly reply “not answerable” as answer, and give explanation. Questions:
Crafter QA system prompt	You are a concise answerer for questions about the Crafter environment. All provided steps are part of a single continuous episode of the game. The timeline presents actions, reasons, and health related stats for each step in order. Use the entire history to answer questions; recall what was seen or collected earlier even if it is not visible in the current frame. If there is not enough information in the given steps to answer, reply with “not answerable” as the answer and briefly explain why. Always reply in strict JSON format and do not guess. (Runtime may append: “Game rules: `{instructions}`”.)
Crafter QA schema (used for batching)	Return strictly JSON. For a single question: `{"answer": "<short>", "explanation": "<brief>"}` For batched questions: an array of objects, each: `{"id": "<qid>", "answer": "<short>", "explanation": "<brief>"}` Keep explanation short. For any question you find not answerable, strictly reply “not answerable” as answer, and give explanation. (Batch header prepends: “You will answer multiple questions about the same episode.” and “Questions:”.)

Table 6: Prompt templates used in our pipelines (gameplay and QA answering) for Jericho text games and the Crafter visual game.

A.4 End-to-end Generation Procedure.

Algorithm 1 summarizes the generator. It is deterministic given a trace τ, game state $\mathcal{S}$, and a random seed used only for sampling among multiple valid template instantiations.

Algorithm 1 Trajectory-conditioned benchmark generation

```
Require: Trace $\tau$, optional auxiliary state $\mathcal{S}$, template set $\mathcal{T}$, sampling seed $u$
Ensure: QA set $\mathcal{Q}$
Build timeline index $\mathcal{I} \leftarrow$ IndexTrace($\tau, \mathcal{S}$)
Build event index $\mathcal{E} \leftarrow$ ExtractEvents($\mathcal{I}$)
$\mathcal{Q} \leftarrow \emptyset$
for all template $T \in \mathcal{T}$ do
    $\mathcal{C} \leftarrow T$.EnumerateCandidates($\mathcal{I}, \mathcal{E}$)
    $\mathcal{C}' \leftarrow$ FilterByPreconditions($\mathcal{C}$)
    Sample a subset $\widehat{\mathcal{C}}$ from $\mathcal{C}'$ using seed $u$
    for all candidate $c \in \widehat{\mathcal{C}}$ do
        $(q, y, m) \leftarrow T$.Instantiate($c, \mathcal{I}, \mathcal{E}$)
        if ValidateInstance($q, y, m, \mathcal{I}$) then
            $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(q, y, m)\}$
        end if
    end for
end for
return $\mathcal{Q}$
```

A.5 Rules for Score Calculation

We evaluate model outputs by scoring the extracted predictions against the reference answers. Our scorer is rule-based and selects different matching strategies according to the `answer_type`. We normalize both prediction and reference by lowercasing, trimming whitespace, removing parenthesized spans, and stripping surrounding quotes.

String. We first decide whether an answer should be *exact-match* using simple patterns, including URLs, filenames, page-like identifiers, phone-number formats, time expressions containing “a.m.”/“p.m.”, dates (YYYY-MM-DD / YYYY-MM), and email addresses. If exact-match is required, we perform direct string comparison and assign a score in $\{0, 1\}$. Otherwise, we compute ANLS (Average Normalized Levenshtein Similarity) with a threshold $\tau = 0.5$ (Biten et al., 2019):

$$s(g,p) = 1 - \frac{d_{\text{lev}}(g,p)}{\max(|g|,|p|)}$$
$$\text{ANLS}(g,p) = \begin{cases} s(g,p) & \text{if } s(g,p) > \tau, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

where $d_{\text{lev}}(\cdot,\cdot)$ is the Levenshtein edit distance.

Integer. We cast both reference and prediction into an integer-like form, accepting common variants such as “1”, “1.0”, or strings with extra spaces (and optional trailing “%”). The score is 1 iff the parsed integers are equal, otherwise 0.

Float. We treat prediction and reference as equal if they match after rounding with an adaptive precision (at least two decimal digits), or if they are within a 1% relative tolerance. To reduce ambiguity between fractions and percentages, we also accept matches against $\{g/100,\ g,\ 100g\}$.

List. Some questions admit multiple acceptable answers. We represent such references as a list of candidates. We score the prediction against each candidate (using the same atomic rules for string/integer/float), and take the maximum score as the final score for the example, i.e., a prediction is correct if it matches any one of the candidates (Rajpurkar et al., 2016).

Unanswerable and aggregate metrics. We use the canonical label `not answerable` for unanswerable questions (Landeghem et al., 2023; Ma et al., 2024). Overall **Accuracy** is the mean of per-question scores. We additionally compute an **F1**-style score by treating `not answerable` as the negative class: recall is computed over questions with answer $\neq$ `not answerable`, precision is computed over questions with prediction $\neq$ `not answerable`, and F1 is their harmonic mean.

A.6 Model Hyper-Parameters

Unless otherwise stated, all models use deterministic decoding with temperature $= 0.0$ for both gameplay and question answering. We report results under two *Query Horizon Control* settings: `QHC=-1` (no restriction) and `QHC=50` (all generated queries are constrained to steps 1–50). For QA construction, we use a fixed automatic generator with `max_per_type=2` and enable paraphrasing (paraphrase=True); in all experiments we evaluate on the paraphrased questions (source=paraphrase).

For text-only games, we cap each episode at 200 interaction steps. For the *in-context* baseline, we include the most recent 10 turns of trajectory context (`history_turns=10`). For memory-based agents, gameplay-time retrieval uses `top_k=10`. During QA answering, we use a maximum output length of 1024 tokens; for memory-based agents, per-question retrieval uses `top_k=8`.

For visual game, we cap each episode at 200 interaction steps. For the *in-context* baseline, we include the most recent 5 turns (`history_turns=5`); for memory-based agents, gameplay-time retrieval uses `top_k=5`. During gameplay we use `max_tokens=128` for action generation, while QA answering uses `max_tokens=4096` with batch size 4. When providing multiple frames to VLMs, we use a fixed mosaic configuration: `frames_mode=mosaic`, `mosaic_cols=10`, `mosaic_cell=160`, `mosaic_per_image=200`, and `mosaic_per_batch=10`.

A.7 Compute and GPU Hours

Across experiments, we measure approximate end-to-end cost/runtime per baseline (one backbone × one method, covering play → QA generation → answering). Using the OpenAI API with GPT-5.1, each text-game baseline costs about 15 USD and takes roughly 15 hours wall-clock, while each visual-game baseline costs about 5 USD and takes about 5 hours; these costs follow token-based pricing. For self-hosted open models, text-only runs on 2× NVIDIA H20-e, where Qwen3-32B takes 1̃0 hours/baseline (2̃0 GPU-hours) and Qwen2.5-32B-Instruct takes 6 hours/baseline (1̃2 GPU-hours). The visual-game runs use 4× NVIDIA H800 (80GB), where Qwen3-VL-32B-Instruct and InternVL3.5-38B each take 4̃ hours/baseline (1̃6 GPU-hours).

A.8 Human Evaluation Details

This appendix provides full details of our human evaluation protocol. We follow the best practices in (van der Lee et al., 2019).

Annotators. We recruited 11 annotators, all proficient in English. All 11 participated in the Jericho text-game study, and 3 of them additionally participated in the Crafter visual-game study. Consent was obtained for the usage of evaluated results. The data collection protocol was approved by an ethics review board.

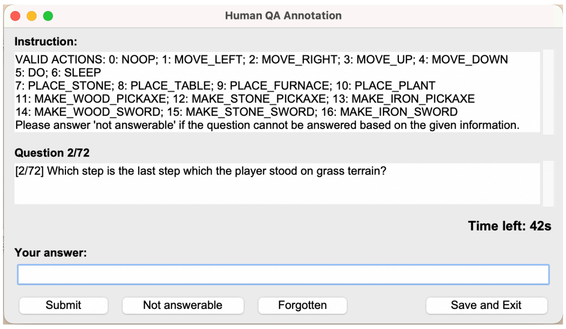
Task and QA construction. For each game and seed, an annotator plays a full episode under a fixed seed while we log the interaction trajectory. We then generate QA from the resulting trajectory using the same dynamic procedure as in our agent evaluation, except that we exclude templates that depend on explicit agent reasoning since it is not applicable to the human playing process.

Answering protocol (closed-book vs. open-book). Annotators answer the same QA set twice: (i) *closed-book*, answering from memory only without consulting any records; and (ii) *open-book*, answering with access to the trajectory *context file* (consistent with the agent-evaluation setting and not containing any hidden backend information). For visual games, open-book additionally allows viewing the saved frames, again consistent with the agent-evaluation setting.

Interface and timing. Figure 5 shows the QA interface. Each question has a 60-second time limit; timed-out questions are counted as incorrect. On average, annotators spend ∼40 minutes playing each text-game episode and ∼3 minutes for each visual-game episode; question answering takes ∼15 minutes (closed-book) and ∼30 minutes (open-book).

Metrics and aggregation. We report Accuracy, F1, and per-question time, aggregated by question type (Tables 4 and 8) and by game for text games (Table 9).

Inter-annotator agreement. We compute inter-annotator agreement over per-type aggregates using Pearson correlation (r), intraclass correlation (ICC), mean absolute error (MAE), and root mean squared error (RMSE) (Tables 10 and 11).



Instruction:

VALID ACTIONS: 0: NOOP; 1: MOVE_LEFT; 2: MOVE_RIGHT; 3: MOVE_UP; 4: MOVE_DOWN
5: DO; 6: SLEEP
7: PLACE_STONE; 8: PLACE_TABLE; 9: PLACE_FURNACE; 10: PLACE_PLANT
11: MAKE_WOOD_PICKAXE; 12: MAKE_STONE_PICKAXE; 13: MAKE_IRON_PICKAXE
14: MAKE_WOOD_SWORD; 15: MAKE_STONE_SWORD; 16: MAKE_IRON_SWORD
Please answer 'not answerable' if the question cannot be answered based on the given information.

Question 2/72

[2/72] Which step is the last step which the player stood on grass terrain?

Time left: 42s

Your answer:

Figure 5: UI interface of the application for human evaluation question answering.

Stage	**What to do (key rules only)**
Deliverables	Submit a single zip containing: (i) your **gameplay log** (auto-generated while you play) and (ii) your **answers** for **both** rounds (close-book and open-book), including the generated QA files kept alongside the answer file.
1) Play the game	**Goal:** play as well as you can based on your understanding. **How to play:** read the *Observation* carefully each step, then choose an action; explore the map; avoid random actions. **If stuck:** if you make no progress (looping/confused), you may quit after roughly ∼100 steps.
GUI (during play)	The interface typically shows: **Left:** status (score/moves/location) and the current *Observation* text. **Right:** available actions as buttons, plus an input box for a custom action; you can also open **inventory**.
2) Generate questions (QA)	After play, run the provided pipeline to generate questions from **your own log**. You only need to know: the game name and your run folder / log identifier.
3) Answer questions (two rounds)	You must complete the same question set twice: **Round A: Close-book.** Answer only from memory; do **not** look back at any logs/records. If unsure, use **not answerable**. **Round B: Open-book.** You may consult the provided **QA context file** (generated from your log) and answer again.
GUI (during answering)	The answering window provides: A question panel with progress and a per-question **time limit** (auto-advances when time is up). A text box to enter your answer and confirm. Buttons: **not answerable** (cannot be answered even with available info) and **cannot remember** (you believe it happened but you cannot recall). You can **save and exit** and later continue from where you stopped.

Table 7: Human evaluation instructions for Jericho text games (play → auto-QA → answer twice).

	Open-book			Closed-book		
Type	Acc(%)	F1(%)	Time(s)	Acc(%)	F1(%)	Time(s)
Single-Hop	85.5	88.3	18.1	8.1	8.1	5.6
Multi-Hop	38.1	38.1	38.3	0.0	0.0	6.7
Inducing	50.0	51.4	15.0	14.3	14.6	6.2
Spatial	34.7	38.4	20.1	5.7	6.0	6.5
Temporal	41.7	46.2	29.6	16.7	17.1	17.5
Logical	54.4	55.2	19.8	18.1	18.4	7.9
Adversarial	90.2	-	9.8	46.3	-	4.7
Overall	59.2	56.5	19.9	15.8	10.8	7.6

Table 8: Human performance (visual game) averaged over annotators per question type. Acc/F1 are in %.

	Open-book			Closed-book		
Game	Acc(%)	F1(%)	Time(s)	Acc(%)	F1(%)	Time(s)
advent	55.7	53.2	21.8	12.3	12.0	7.9
awaken	65.7	66.0	16.7	27.5	24.6	5.5
balances	67.9	65.5	21.1	28.8	24.1	10.3
dragon	65.6	65.2	21.0	17.3	13.2	7.4
gold	75.9	76.1	21.7	14.1	12.6	10.8
jewel	62.9	62.5	26.7	22.1	20.7	10.9
karn	60.2	58.6	27.0	17.4	13.3	19.0
ludicorp	57.1	52.8	33.2	21.0	14.5	19.9
moonlit	65.3	60.7	24.4	28.2	24.0	11.6
pentari	69.0	66.4	24.3	23.3	17.7	9.5
reverb	66.6	64.8	21.4	30.1	24.8	12.2
sorcerer	72.5	71.6	21.8	28.0	25.0	10.3
zork1	64.4	62.5	25.0	26.8	23.0	11.2
zork2	63.1	61.8	22.8	28.1	27.4	9.3
zork3	71.4	68.3	20.4	24.1	22.0	6.9
Overall	65.6	63.8	23.3	23.2	19.8	10.8

Table 9: Human performance (text games) averaged over two annotators per game. Acc/F1 are in %.

Metric	Pearson r	ICC	MAE	RMSE
Open-book per-type Acc	0.90	0.71	0.09	0.11
Open-book per-type Time	0.68	0.40	5.69	6.66
Closed-book per-type Acc	0.97	0.95	0.03	0.05
Closed-book per-type Time	0.79	0.40	6.73	6.86

Table 10: Inter-annotator agreement for text games.

Metric	Pearson r	ICC	MAE	RMSE
Open-book per-type Acc	0.85	0.78	0.22	0.30
Open-book per-type Time	0.62	0.45	5.82	6.54
Closed-book per-type Acc	0.77	0.69	0.11	0.14
Closed-book per-type Time	0.61	0.44	5.05	7.52

Table 11: Inter-annotator agreement for the visual game.

B Full Results of Query Horizon Control Setting

We provide the full results of query horizon control setting in Table 12.

Model	Method	Single-Hop	Multi-Hop	Induction	Spatial	Temporal	Logical	Adversarial	Overall	
		ACC	ACC	ACC	ACC	ACC	ACC	ACC	ACC	F1
Text-only Games										
GPT-5.1	In-context	**71.8**	34.7	**40.5**	46.0	51.7	43.3	43.0	49.3	<u>48.3</u>
	Mem0	51.9	8.6	16.7	14.6	53.5	25.2	84.5	37.7	34.7
	LangMem	44.3	34.5	22.5	<u>47.5</u>	**72.3**	34.5	81.4	48.6	45.6
	A-MEM	48.5	33.9	27.2	47.0	59.0	<u>62.0</u>	74.4	49.6	46.0
Qwen2.5-32B-Instruct	In-context	52.3	23.1	<u>38.4</u>	34.3	52.5	41.9	50.0	41.0	40.1
	Mem0	54.7	8.0	14.3	8.6	47.1	22.2	**89.9**	36.4	32.5
	LangMem	48.1	<u>37.6</u>	31.7	**50.3**	55.6	28.2	84.9	49.9	46.5
	A-MEM	<u>67.3</u>	30.7	26.2	45.6	58.0	**65.4**	<u>88.7</u>	**54.7**	**51.0**
Qwen3-32B	In-context	57.8	**46.7**	33.6	47.1	48.5	41.3	46.2	47.7	46.9
	Mem0	54.3	1.9	24.2	5.2	<u>65.5</u>	24.3	84.9	41.3	33.7
	LangMem	45.5	17.0	25.6	44.3	48.0	34.8	71.0	41.4	36.0
	A-MEM	61.6	35.7	25.1	42.5	48.7	59.6	75.6	<u>50.8</u>	46.8
Visual Games										
GPT-5.1	In-context	59.5	38.5	39.0	26.6	<u>56.7</u>	36.1	**64.6**	**48.2**	**46.0**
	A-MEM	54.8	**46.0**	34.8	**32.6**	**60.7**	31.0	55.8	46.8	42.3
Qwen3VL-32B-Instruct	In-context	<u>59.6</u>	<u>41.8</u>	**39.6**	24.9	36.7	48.2	62.4	47.1	42.8
	A-MEM	**62.8**	38.7	<u>39.3</u>	<u>27.0</u>	32.9	**52.3**	<u>62.8</u>	<u>48.0</u>	<u>45.0</u>
InternVL3.5-38B	In-context	34.1	10.7	17.4	12.3	34.1	44.7	48.5	30.9	26.4
	A-MEM	38.4	7.3	18.1	14.4	36.9	<u>48.8</u>	55.4	33.9	28.7

Table 12: Query-horizon control results with queries restricted to steps 1–50. Within each modality block, the best and second-best numbers in each column are **bolded** and <u>underlined</u>, respectively.

C Case Study

This appendix presents two complementary groups of case studies for memory agents. We present **five common mistakes**—(1) evidence localization errors, (2) long-horizon aggregation failures, (3) visual grounding mistakes, (4) reasoning confusion, and (5) answer–explanation inconsistency. Additionally, we provide **two challenging question analyses** that are hard-to-solve for memory agents so far: (6) global object counting under partial observability and (7) target-reaching calculation with obstacles and detours.

Case 1: Evidence localization error.

Question: What action is taken 2 steps after the first step where the reason mentions 'out'?

GT: `north`

Pred: `Not answerable.`

Model explanation: The snippets only provide selected reasoning texts, observations, and results for certain steps, but not a complete mapping from every step number to its action or full reasoning text. To answer the question, we would need to (1) identify the earliest step whose reasoning includes the word 'out', and then (2) know the exact action taken two steps after that step.

Relevant context:

```
t=79 | action=east | reason=Try unexplored
maze direction to find a way out |
observation=\nMaze\n\nA threatening little
dwarf comes out of the shadows! |
location=(Alike_Maze_3) | reward=32.0 |
done=false
t=80 | action=west | reason=Try a new maze
direction; avoid repeating recent moves |
observation=\nMaze\n\nThe dwarf stalks after
you...\n\nThe dwarf throws a nasty little knife
at you, but misses! | location=(Alike_Maze_2)
| reward=32.0 | done=false
t=81 | action=north | reason=Keep exploring
maze; try to find exit or landmark |
observation=\nMaze\n\nA threatening little
dwarf hides in the shadows.\n\nThe dwarf throws
a nasty little knife at you, but misses! |
location=(Alike_Maze_1) | reward=32.0 |
done=false
```

Error analysis. This is an evidence retrieval failure over step-indexed reasoning. The correct anchor is the first step whose *reason* contains the keyword 'out' (t=79), and the answer requires a deterministic offset over consecutive steps (t=81 → `north`). The agent cannot reliably retrieve the earliest keyword-bearing reason with its timestep and the subsequent actions. Such multi-hop queries are difficult: missing a single anchor step makes the downstream step lookup appear unsupported, resulting in a false "Not answerable" decision.

Case 2: Answer–explanation inconsistency.

Question: What was the longest unbroken sequence of `MOVE_LEFT` actions?

GT: 4

Pred: 3

Model explanation: The longest consecutive sequence of MOVE_LEFT actions is from t=97–t=100 (97,98,99,100) which is 4? Wait; checking: 97,98,99,100 are four in a row.

Relevant context:

```
t=97  |  action=MOVE_LEFT  |
frame=crafter_t000097.png  |
(health,food,water,energy)=(9,6,5,6)
reason=Needs are stable and we want to reach
the stone wall on the far left; move left while
facing left to start closing the distance toward
the stone for mining with our wood pickaxe
while staying near the lake for later drinking.
t=98  |  action=MOVE_LEFT  |
frame=crafter_t000098.png  |
(health,food,water,energy)=(9,6,5,6)
reason=Needs are stable and we are close to the
stone wall; move left while facing left to
stand directly adjacent to the stone so we can
start mining it with DO using our wood pickaxe,
while still remaining near the lake for later
drinking.
t=99  |  action=MOVE_LEFT  |
frame=crafter_t000099.png  |
(health,food,water,energy)=(9,6,5,6)
reason=Needs are fine and the stone wall is
just to our left; step left while facing left
to become directly adjacent so we can start
mining stone with DO next, staying close to the
lake for future drinking.
t=100  |  action=MOVE_LEFT  |
frame=crafter_t000100.png  |
(health,food,water,energy)=(9,6,5,6)
reason=Status is safe and the stone wall is
immediately to our left; move left while facing
left to stand adjacent to the stone so we can
start mining it with DO using our wood pickaxe,
still remaining close to the lake for later
drinking.
```

Error analysis. The explanation itself identifies a 4-step run, but the final answer reports 3, reflecting a self-contradiction between the produced rationale and the emitted prediction. For integer-valued questions, such slips (off-by-one or inconsistent finalization) cause complete loss under exact-match scoring even when the intermediate reasoning is correct.

Case 3: Long-horizon aggregation failure.

Question: How many unique locations have been visited?

GT: 25

Pred: 18

Model explanation: We found these distinct location names: Narrow Tunnel, Foot Bridge, Great Cavern, Shallow Ford, Dark Tunnel, Path Near Stream, Ice Room, Cool Room, Inside the Barrow, Bank Entrance, Fresco Room, West Viewing, East Viewing, Lava Tube, Formal Garden, Dragon Room, Stone Bridge, Gazebo, Carousel Room, North End of Garden. After deduplication (e.g., Ice vs Ice Room; Fresco vs Fresco Room) we get 18.

Relevant context:

```
All locations that appear in the recorded map:
1. Inside Barrow; 2. Narrow Tunnel; 3. Foot
Bridge; 4. Shallow Ford; 5. Dark Tunnel;
6. Path Near Stream; 7. Formal Garden; 8. North
End Garden; 9. Gazebo; 10. GreCavern;
11. Topiary; 12. Carousel; 13. Cool; 14. Ice;
15. Lava Tube; 16. Cobwebby Corridor; 17. Stone
Bridge; 18. Drag; 19. End Ledge; 20. Fresco;
21. Bank Entrance; 22. West Teller's; 23. West
Viewing; 24. East Teller's; 25. East Viewing.
```

Error analysis. This combines (i) long-horizon set aggregation—omissions accumulate silently over many steps; and (ii) string-level dedup ambiguity—surface-form heuristics can over-merge distinct nodes or miss true duplicates. When the number of visited locations grows, both omission and mis-dedup become more likely, producing systematic undercounts even when the agent attempts an explicit list.

Case 4: Visual grounding mistake.

Question: What terrain was under the player at step 113?

GT: grass

Pred: Sand

Model explanation: Reasoning for t=113 says the player is standing on sand directly beside the lake.

Relevant context: Figure 6

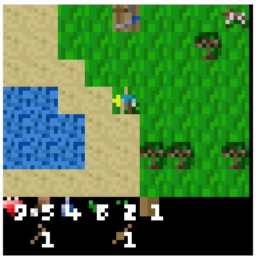


Figure 6: Frame for step 113.

Error analysis. The model confuses *nearby salient terrain* with the *tile under the player*. Visually, sand may be adjacent and prominent, but the question requires a fine-grained under-agent grounding decision; the prediction follows the plausible nearby cue rather than the precise queried support.

Case 5: Reasoning Confusion.

Question: How many times did the player successfully collect wood?

GT: `5`

Pred: `20`

Model explanation: There are 20 DO actions explicitly described as chopping trees to collect or gain more wood.

Relevant context: Figure 7

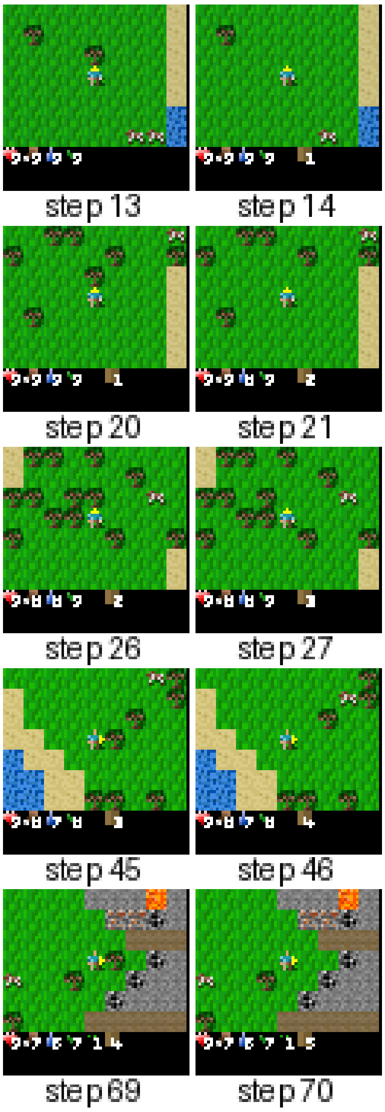


Figure 7: Steps that agent successfully collected wood.

Error analysis. The model counts *attempts* (described chopping/DO actions) instead of *verified successful outcomes*. In interactive settings, attempting to collect a resource does not guarantee success; success is better indicated by state-change evidence (e.g., wood count increases) or explicit success events. Treating attempts as successes inflates the count whenever failures occur.

Case 6: Challenging Question 1.

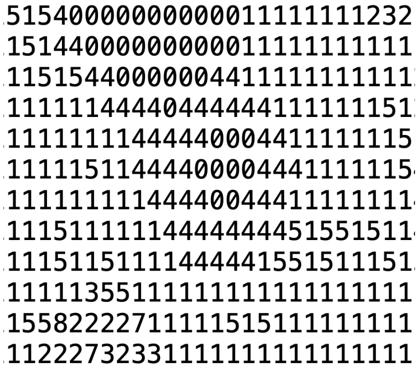
Question: What is the total number of unique trees the player encountered?

GT: `18`

Pred: `Not answerable.`

Model explanation: The 64×64 view changes as the player moves, so the same tree can leave and re-enter the frame; without exact map coordinates we cannot distinguish unique trees.

Relevant context: Figure 8



```
51540000000000011111111232
15144000000000011111111111
11515440000004411111111111
11111144440444444111111151
11111111444440004411111115
11111511444400004441111115
11111111144440044411111111
11151111114444444451551511
11151151111444441551511151
11111355111111111111111111
15582222711115151111111111
11222732331111111111111111
```

Figure 8: Part of the global map in backend data, full map is 64x64. Each number represents a type of tile.

Difficulty analysis. This question is challenging because the agent observes only a partial map at each step. As a result, counting "unique trees" cannot be solved by summing per-frame tree counts; the same tree may repeatedly enter/leave the view across time. Answering correctly requires strong spatial reasoning to (i) integrate observations over time into a coherent global map, and (ii) perform object identity tracking across frames to deduplicate revisits. Although the agent refuses to answer, the question is in fact answerable. In our ground-truth pipeline, we first extract a global map for each game instance **(0=water, 1=grass, 2 = stone, 3 = coal, 4 = sand, 5=tree, etc.)**. Using the agent's real-time position, we compute the union of all map regions that were ever observable along the trajectory, i.e., the exact explored/seen area. Meanwhile, we update the global map according to the agent's actions (e.g., chopping a tree), so the global map reflects the environment dynamics rather than a static snapshot. With the reconstructed seen-area plus these action-conditioned updates, the total number of unique encountered trees can be calculated.

Case 7: Challenging Question 2 .

Question: At step 118, from the player's current location, what movement is required to reach the nearest lake? Respond with 'X steps left/right and Y steps up/down', 'not answerable' if the target is never visible, or 0 if the player is already at the target.

GT: `1 step right, 6 steps up and 3 steps right`

Pred: `4 steps right and 7 steps up`

Model explanation: From step 118 the nearest visible lake is the one reached earlier near steps 131–133, offset about 4 tiles right and 7 up.

Relevant context: Figure 9

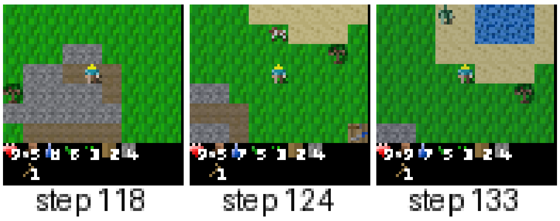

step 118 step 124 step 133

Figure 9: Figure for frames near the lake.

Difficulty analysis. The agent successfully identifies a nearest lake that is not in the current frame, implying it can relate the current position to a previously observed landmark across time, which is already a non-trivial spatial reasoning capability. The remaining difficulty stems from two factors. First, the vertical distance is miscounted by one step (the correct upward movement is 6, not 7), showing how sensitive such questions are to small coordinate errors. Second, the direct "up" displacement is not necessarily traversable: **the tile immediately above the agent is stone and cannot be crossed**, so the valid shortest route must detour around the obstacle. The ground-truth computation follows the navigation algorithm in Appendix G, which explicitly accounts for impassable terrain when deriving the minimal path.

D Text-Game Environment (Jericho)

D.1 Jericho environment

We evaluate long-horizon memory in *interactive fiction* (IF), where an agent interacts with a world purely through natural-language commands (e.g., `take lantern`, `go north`). We use **Jericho** (Hausknecht et al., 2020), a lightweight Python interface that connects learning agents to classic Z-Machine story files (e.g., `.z3`/`.z5`) through a Gym-like API. At each step, the agent emits a text action; the environment returns a textual observation, a scalar reward/score signal (game-dependent), a termination flag, and auxiliary metadata. Jericho also provides utilities commonly used by text-game agents, such as querying valid actions and accessing structured state information exposed by the underlying interpreter (when available). This setup yields long trajectories with sparse rewards and a large, compositional action space, making it a suitable testbed for evaluating memory-dependent reasoning across extended interaction histories.

D.2 Game suite

Table 13 lists the Jericho games used in our text-game benchmark. They cover diverse genres (classic cave-crawls, office mysteries, fantasy quests, short-form vignettes) and vary widely in exploration difficulty, puzzle dependency, and required world knowledge.

E Visual Game Environment (Crafter)

E.1 Crafter environment

We also evaluate memory in a *visual* open-world survival environment, **Crafter** (Hafner, 2022), which is designed as a compact Minecraft-inspired benchmark with procedurally generated 2D worlds and a technology tree (resources → tools → advanced resources). Crafter episodes are long-horizon and partially observable: at each step, the agent receives a 64×64 **RGB** top-down crop centered on the player and must balance survival (food, water, energy, health) with exploration and crafting progress (Hafner, 2022). Agents are evaluated by whether they can unlock a diverse set of semantically meaningful achievements within an episode, which probes generalization, long-term reasoning, and deep exploration (Wu et al., 2023).

In our setup, we provide a textual HUD summary each step (derived from the environment state) and attach a fixed icon legend to help resolve object identities from pixels. Figure 10 shows the icon legend used throughout our experiments.

E.2 Observation example

Figure 11 shows a single Crafter observation as seen by the agent. The *upper* region is a local top-down view centered on the player, while the *bottom* HUD encodes the player's internal state (e.g., health/food/water/rest) and inventory counts (materials and crafted tools). In this example, the

ROM	Title	Genre / Setting	One-sentence synopsis
advent.z5	Colossal Cave Adventure	Classic exploration / cave-crawl	Explore a vast cave system, solve inventory puzzles, and collect treasures for score.
awaken.z5	The Awakening	Mystery / survival	You awaken amid storm and mud with fragmented memory and must recover your bearings and escape danger.
balances.z5	Balances	Fantasy puzzle / spellcraft	A short, spell-focused adventure: learn and cast scroll-based spells to progress and collect objectives.
dragon.z5	Dragon!	Micro-IF / humorous vignette	A very short scenario centered on playing a dragon pursuing a single-minded goal.
gold.z5	Goldilocks is a FOX!	Fairy-tale remix / puzzle adventure	A puzzle-driven mashup of fairy-tale motifs (e.g., bears, wolves, magic-tale references).
jewel.z5	The Jewel of Knowledge	Classic dungeon crawl	Brave a lethal multi-layer dungeon to obtain the fabled Jewel of Knowledge guarded by powerful foes.
ludicorp.z5	The Ludicorp Mystery	Modern mystery / office building	Investigate a game company's office to uncover why a long-awaited release has gone missing.
moonlit.z5	The Moonlit Tower	Symbolic fantasy / atmospheric	Navigate a surreal tower of symbols (masks, seasons, symmetry) to uncover meaning and resolution.
pentari.z5	Pentari	Fantasy / military prequel	As a company commander on leave, you navigate a compact fantasy town and events before a looming mission.
reverb.z5	Reverberations	Thriller / comedy-adventure	A pizza-delivery surfer gets entangled in an escape plot, murder attempts, and a conspiracy with romantic side-threads.
sorcerer.z3	Sorcerer	Fantasy / Infocom spell adventure	Track the vanished guildmaster Belboz via magical clues and avert a looming threat to the Circle of Enchanters.
zork1.z5	Zork I: The Great Underground Empire	Classic treasure-hunt dungeon	Explore the Great Underground Empire, solve puzzles, and amass treasures for points.
zork2.z5	Zork II: The Wizard of Frobozz	Classic dungeon + antagonist	Continue deeper in Zork while contending with the Wizard's interference and advanced puzzles.
zork3.z5	Zork III: The Dungeon Master	Classic dungeon / endgame trial	Push into the deepest reaches of Zork for a culminating trial involving the enigmatic Dungeon Master.

Note: Synopses are summarized from publicly available game catalog blurbs and archival entries (e.g., IFDB/IFWiki/IF Archive).

Table 13: Jericho text-game suite used in this paper (ROM filenames and brief synopses).

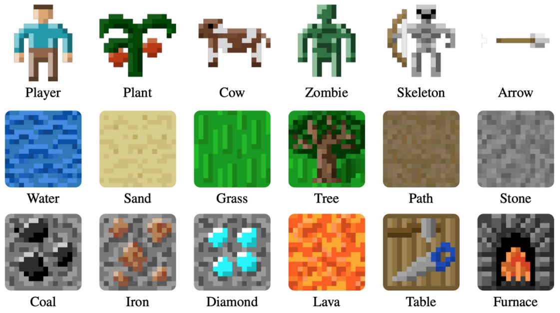


Figure 10: Crafter icon legend used by our visual agents. We attach this legend to each step to disambiguate terrain, resources, stations, and creatures from map.

player stands at the boundary between grass and water, with nearby trees and stone terrain containing coal and lava. Throughout our experiments, we additionally provide a textual HUD summary each step (for exact numbers) and attach the icon legend in Fig. 10 to disambiguate object identities.

Figure 11: Example Crafter observation (agent input). The agent receives a 64 × 64 RGB image containing a local top-down crop around the player (top) and an inventory/status HUD (bottom).

E.3 Crafter instructions

Crafter Instructions

WORLD VIEW & HUD. You aim to survive longer and maximize unlocked achievements. You observe a 64×64 RGB top-down crop centered on the player. A HUD exists in the image; we also provide a text HUD each step with last step's numbers. *Trust the text HUD over pixels* for numeric state diagnosis (food/drink/energy/safety). An **icon legend** image is attached at each step to resolve object appearance (player, terrain, resources, stations, creatures). Expect darker icons at night and be more conservative.

ENVIRONMENT & OBJECTS (identify before acting). Water (lake) enables drinking; sand often borders lakes. Trees/grass yield wood; stone and ores appear on rocky terrain (coal/iron/diamond require stronger tools). Caves are darker stone; hostile creatures are more common at night. Lava is hazardous; bridge via placed stone. Stations/placeables include table and furnace (for advanced tools), plus stone blocks and plants.

SURVIVAL-FIRST POLICY. When energy is low (e.g., < 6), prioritize sleep. Night increases danger and reduces visibility. If low water: seek the nearest lake and drink; if low food: harvest plants or obtain food from a cow. Avoid action spamming; if repeated interactions have no effect, you are likely not adjacent or not facing the target.

PROGRESSION (only when needs are safe). Early: gather wood → place table → craft wood tools (including a defensive sword). Mid: obtain stone/coal → place furnace → craft stone tools and mine iron. Iron tier: with table+furnace and iron+coal+wood, craft iron tools and then mine diamond. Keep stations in safe locations; use stone placement to bridge lava or block chokepoints; plant saplings for future food.

REASONING STYLE. Provide a brief diagnosis (health / hunger / thirst/ energy) and a concise next-step plan. Be specific about your intended target (e.g., lake / tree / table/ cow) and your current facing direction.

Type	Template	Question template	Ans.
Single-Hop	`A_action`	At step {step}, what action did you execute?	Action
	`A_reason`	At step {step}, what is the first sentence of your reasoning?	String
	`A_location`	Before performing the action at step {step}, what is your location?	Location
	`A_obs_before`	Before performing the action at step {step}, what is the observation?	String
	`A_obs_after`	After performing the action at step {step}, what is the resulting observation?	String
	`A_reward`	After performing the action at step {step}, what is the cumulative reward so far?	Integer
	`A_valid_action`	At step {step}, is ‘{action}’ a valid action? Answer in ‘yes’ or ‘no’.	Yes/No
	`A_gain_item`	At which step did you {first / last} gain ‘{item}’?	Step #
	`A_enter_leave`	At which step did you {first start being at / first leave / last start being at} ‘{location}’ before performing the action?	Step #
	`A_keyword_occurrence`	Which step is the {first / second / last / second last} step whose **reason** mentions ‘{keyword}’?	Step #
Multi-Hop	`B_gain_after_action`	After first obtaining ‘{item}’, what action is executed {delta} step(s) later?	Action
	`B_gain_after_location`	After first obtaining ‘{item}’, what is your location {delta} step(s) later?	Location
	`B_gain_after_observation`	After first obtaining ‘{item}’, what is the observation {delta} step(s) later?	String
	`B_gain_after_reward`	After first obtaining ‘{item}’, what is the cumulative reward {delta} step(s) later?	Integer
	`B_keyword_after_action`	After the first step whose reason mentions ‘{keyword}’, what action is executed {delta} step(s) later?	Action
	`B_keyword_after_location`	After the first step whose reason mentions ‘{keyword}’, what is your location {delta} step(s) later?	Location
	`B_keyword_after_observation`	After the first step whose reason mentions ‘{keyword}’, what is the observation {delta} step(s) later?	String
	`B_keyword_after_reward`	After the first step whose reason mentions ‘{keyword}’, what is the cumulative reward {delta} step(s) later?	Integer
Inducing	`C_action_mode`	What is the most frequent action executed?	Action
	`C_distinct_locations`	From steps {L} to {R}, how many distinct locations were visited?	Integer
	`C_most_frequent_location`	From steps {L} to {R}, what is the most frequently visited location?	Location
	`C_total_dwell`	Which is the location with the longest duration of stay in total (within {L}–{R})?	Location
	`C_keyword_count_obs`	From steps {L} to {R}, how many times does ‘{keyword}’ appear in observations?	Integer
	`C_keyword_count_reason`	From steps {L} to {R}, how many times does ‘{keyword}’ appear in reasons?	Integer
Spatial	`D_compare_distances`	Which is closer (fewer steps to reach), ‘{A}’ or ‘{B}’, from the location at step {anchor}?	Location
	`D_direction_count`	From steps {L} to {R}, how many times did you move {direction}?	Integer
	`D_reachable_locations_count`	How many distinct locations are reachable from ‘{location}’ within {k} steps?	Integer
	`D_reachable_within`	Is ‘{target}’ reachable from ‘{source}’ within {k} steps? Answer in ‘yes’ or ‘no’.	Yes/No
	`D_sequence_moves`	At step {step}, if you move {sequence of directions}, which location do you reach?	Location
	`D_shortest_path`	What is the shortest path length between the locations at steps {i} and {j}?	Integer
Temporal	`E_gain_delay`	After first obtaining ‘{item}’, how many steps later is ‘{item}’ obtained again?	Integer
	`E_item_before_leave`	Before first leaving ‘{location}’, did you ever obtain ‘{item}’? Answer in ‘yes’ or ‘no’.	Yes/No
	`E_item_order`	Before first time obtaining ‘{A}’, have you ever obtained ‘{B}’? Answer in ‘yes’ or ‘no’.	Yes/No
	`E_region_stay`	After first entering ‘{location}’, for how many consecutive steps did you stay in ‘{location}’?	Integer
	`E_scene_order`	Before first time entering ‘{A}’, have you ever been to ‘{B}’? Answer in ‘yes’ or ‘no’.	Yes/No
Logical	`F_has_item`	At step {step}, did you have ‘{item}’ in inventory? Answer in ‘yes’ or ‘no’.	Yes/No
	`F_list_inventory`	After performing the action at step {step}, what are all the items you carry?	List
	`F_max_inventory_step`	After completing which step does the inventory contain the most items?	Step #
	`F_location_most_item_gain`	At which location did you gain the most distinct items?	Location
Adversarial	–	**Construction rule:** take any non-adversarial template above and instantiate it with a value that does *not* occur in the trajectory but valid in the game environment (e.g., item/location/keyword), creating a false-premise question. The correct response should be `not answerable`. **Example:** instantiate `A_gain_item` with an unseen item. “At which step did you first gain ‘`sword`’?” → `not answerable`.	Not Answerable

Table 14: Full question template inventory for Jericho text games. Keyword-occurrence variants are merged; Multi-Hop templates are listed individually. Adversarial templates are constructed by instantiating existing templates with nonexistent (false-premise) values; see example.

Type	Template	Question template	Ans.
Single-Hop	`A_action`	What is the action at step {step}?	Action
	`A_reason`	What was the reasoning at step {step}?	String
	`A_stat`	What was your {stat} value at step {step}?	Integer
	`A_terrain`	What terrain was under you at step {step}?	Terrain
	`A_inventory`	How many {resource} did you have at step {step}?	Integer
	`A_occ_action`	Which step is the {first / second / third / last} step whose action is '{action}'?	Step #
	`A_occ_keyword`	Which step is the {first / second / third / last} step whose reason mentions '{keyword}'?	Step #
	`A_occ_terrain`	Which step is the {first / second / third / last} step which you stood on {terrain} terrain?	Step #
Multi-Hop	`B_action`	What is the action {offset} step(s) {before / after} the {first / second / third / last} step whose action is '{action}'?	Action
	`B_keyword`	What is the action {offset} step(s) {before / after} the {first / second / third / last} step whose reason mentions '{keyword}'?	Action
	`B_terrain`	What is the action {offset} step(s) {before / after} the {first / second / third / last} step which you stood on {terrain} terrain?	Action
Inducing	`C_keyword_count`	From steps {L} to {R}, how many steps have reasons that mention '{keyword}'?	Integer
	`C_most_move`	From steps {L} to {R}, what was the most common movement direction?	Direction
	`C_longest_run`	From steps {L} to {R}, what was the longest consecutive run of {action}?	Integer
	`C_collect_res`	From steps {L} to {R}, how many times did you successfully collect {resource}?	Integer
	`C_resource_peak`	After completing which step does {resource} reach its maximum quantity (at least 1)?	Step #
	`C_resource_change`	From steps {L} to {R}, what was the change in {resource} quantity?	Integer
	`C_visible_count`	From steps {L} to {R}, in how many steps could you see any {terrain} in the frame?	Integer
	`C_adjacent_count`	From steps {L} to {R}, in how many steps were you adjacent to any {terrain}?	Integer
	`C_distinct_trees`	From steps {L} to {R}, how many distinct trees did you see in total?	Integer
Spatial	`D_displacement`	From steps {L} to {R}, what was the total displacement of you? Answer in 'X step(s) left/right and Y step(s) up/down'.	Displacement
	`D_path_length`	From steps {L} to {R}, how many movement steps did you successfully take?	Integer
	`D_predict_terrain`	At step {step}, if you walked {direction} {K} step(s), what terrain would be underfoot?	Terrain
	`D_nearest_dir`	At step {step}, in which direction is the nearest {terrain} relative to you?	Direction
	`D_nav_to_target`	At step {step}, how should you move to reach the nearest {terrain}? Answer in displacement format (or '0' if already on target).	Displacement
	`D_min_dist`	From steps {L} to {R}, at which step were you closest to the nearest {terrain}? (If ties, answer the first.)	Step #
	`D_max_dist`	From steps {L} to {R}, at which step were you furthest from the nearest {terrain}? (If ties, answer the first.)	Step #
Temporal	`E_event_order`	Did {eventA} happen before {eventB}? Answer in 'yes' or 'no'.	Yes/No
	`E_event_interval`	After {eventA}, after how many steps did {eventB} occur?	Integer
	`E_state_after_event`	Immediately after {event}, what was your {stat} value?	Integer
Logical	`F_craft_feasibility`	At step {step}, are the collected resources enough to craft {craft}? Answer in 'yes' or 'no'.	Yes/No
	`F_event_loc`	At which step(s) did you {event}? (Answer as a list of step numbers.)	List
	`F_attack_count`	How many times were you attacked?	Integer
	`F_death_reason`	What was the cause of your death at the end of the episode?	String
	`F_first_attack_step`	At which step were you attacked for the first time?	Step #
	`F_last_attack_step`	At which step were you attacked for the last time?	Step #
	`F_inventory_contents`	At step {step}, what are all the items you carry?	List
Adversarial	–	**Construction rule:** take any non-adversarial template above and instantiate it with a value that does *not* occur in the trajectory but valid in the game environment (e.g., resource/terrain/keyword/action), creating a false-premise question. The correct response should be `not answerable`. **Example:** instantiate `A_occ_keyword` with an unseen keyword. "Which step is the first step whose reason mentions '`teleport`'?" → `not answerable`.	Not Answerable

Table 15: Full question template inventory for the visual-game (Crafter) benchmark. Occurrence variants are merged; Multi-Hop templates are listed individually.

F Full Question Template

We provide the full question template in Table 14 for text games and in Table 15 for visual games.

G Demonstration of Ground Truth Computation Algorithm

We demonstrate the ground-truth computation using `D_nav_to_target` questions as an example.

We compute its ground-truth by (i) building the dynamic grid for the episode, (ii) running BFS from the query step position to all terrain cells matching the target terrain, and (iii) reconstructing one or multiple shortest move sequences. See Algorithm 2 and Algorithm 3.

Algorithm 2 Ground-truth for `D_nav_to_target`

```
Require: base grid $G$; trajectory positions pos$[t] = (r_t, c_t)$; actions act$[t]$; inventories inv$[t]$; optional dynamic maps maps$[t]$.
Require: query step $t_0$ (0-based internally); target terrain name $\tau$.
Ensure: ground-truth answer $y$ (STRING / LIST / not answerable).
/* Build / select dynamic grid */
if maps is available then
    dyn_grid $\leftarrow$ maps[last]
else
    dyn_grid $\leftarrow$ APPLYTRAJECTORYEDITS($G$, pos, act, inv)
end if
start $\leftarrow$ pos$[t_0]$
target_ids $\leftarrow \{ id \mid$ TERRAINIDTONAME$[id] = \tau \}$
($d$, targets, parent) $\leftarrow$ BFSALLMINTARGETS(dyn_grid, start, target_ids)    ▷ Algorithm 3
if $d =$ None then
    return not answerable
else if $d = 0$ then
    return 0
end if
routes $\leftarrow \emptyset$
for all $u \in$ targets do
    dirs $\leftarrow$ RECONSTRUCTPATH(parent, start, $u$)
    routes $\leftarrow$ routes $\cup$ { COMPRESSPATH(dirs) }
end for
if |routes| = 0 then    ▷ safeguard
    return not answerable
else if |routes| = 1 then
    return the only element in routes
else
    return SORTEDLIST(routes)
end if
```

Algorithm 3 BFSALLMINTARGETS

```
Require: grid dyn_grid; start cell start; target terrain id set target_ids.
Ensure: ($d$, targets, parent), where $d$ is the shortest distance to any target terrain cell; targets are all target cells at distance $d$; parent stores BFS tree for path reconstruction.
initialize FIFO queue $Q \leftarrow$ [start]
initialize distance map dist[start] $\leftarrow 0$
initialize parent map parent (empty)
best_d $\leftarrow$ None, targets $\leftarrow \emptyset$
while $Q$ not empty do
    pop cell $x$ from $Q$
    $d_x \leftarrow$ dist$[x]$
    if best_d $\neq$ None and $d_x >$ best_d then
        break    ▷ all remaining nodes are farther
    end if
    if TERRAINID(dyn_grid, $x$) $\in$ target_ids then
        if best_d = None then
            best_d $\leftarrow d_x$
        end if
        targets $\leftarrow$ targets $\cup \{x\}$
        continue
    end if
    for all 4-neighbor $y$ of $x$ that is traversable do
        if $y$ not visited then
            dist[$y$] $\leftarrow d_x + 1$
            parent[$y$] $\leftarrow x$
            push $y$ into $Q$
        end if
    end for
end while
if best_d = None then
    return (None, $\emptyset$, parent)
else
    return (best_d, targets, parent)
end if
```