

Beyond the Context Window

Why AI Agents Need
Memory That Works



Shahzaib Awan

CEO, Deaimer

MEMORY / AGENTS / CONTINUITY

Inside this article

A research perspective on memory that helps agents retain useful experience, revise outdated assumptions, and resume meaningful work.

1. The work continues after the conversation ends
2. Context, memory, and learning describe different mechanisms
3. A longer context does not settle the problem
4. What memory research already teaches us
5. Four properties of memory that supports work
6. From factual recall to operational consequences
7. Remembering mistakes without inventing lessons
8. Time changes the meaning of stored information
9. Memory and the environment must remain distinguishable
10. Memory access needs a defined scope
11. The dataset unit becomes a sequence of experiences
12. How Deaimer could support memory research
13. A practical study of memory-dependent work
14. Continuity with the ability to reconsider

Includes a four-property memory framework, five illustrative environment families, and a proposed research program for model developers and dataset buyers.

Abstract

AI agents that perform extended work must make decisions using information acquired at different moments, under different conditions, and sometimes across separate sessions. Retaining more text is one part of this challenge. An agent must also identify relevant experience, distinguish evidence from inference, recognize superseded information, and determine when a previous conclusion needs to be checked again. This research perspective examines memory as a capability expressed through action. It draws on primary studies of long-context use, conversational memory, hierarchical memory management, reflective feedback, and reusable skill libraries. It then proposes a framework for designing memory-dependent tasks around persistence, provenance, revision, and reuse. Illustrative scenarios span software engineering, terminal operations, scientific workflows, and recovery of existing systems. The article explains how Deaimer's custom environment and dataset work could support these research questions while distinguishing that opportunity from a claim that an existing collection already measures every aspect of memory. No new model experiment or customer performance result is reported.

1. The work continues after the conversation ends

An AI agent begins investigating a failing service. It discovers that a recent migration changed the expected data format, rules out a network fault, and identifies one configuration value that requires further examination. The session ends before the repair is complete.

When the agent resumes, the service is still failing, but the environment has changed. A teammate has updated a dependency. A log file has rotated. The previous configuration has been replaced. The agent has access to a record of its earlier work.

What should it do with that record?

It could repeat the entire investigation. It could trust the old diagnosis and apply the proposed repair immediately. Or it could recover the useful parts of the earlier investigation, identify which assumptions may have changed, and gather the evidence needed to continue.

This third possibility is the focus of the article. Useful memory helps an agent maintain continuity while remaining responsive to the present. It preserves the reasoning context behind a decision and allows that decision to be reconsidered.

The distinction matters because real work rarely consists of an isolated question with a timeless answer. A repository evolves. A scientific analysis changes as new data arrives. An operational incident develops while people and services act on it. The agent's history can be valuable precisely because it contains information that would otherwise be expensive to reconstruct.

Yet the same history can become misleading. A once-correct conclusion may no longer apply. A temporary workaround may be mistaken for a general rule. A failed experiment may have been inconclusive rather than definitive.

Developing memory-capable agents therefore calls for tasks that make these differences consequential. If every historical fact remains valid forever, a system may demonstrate recall without demonstrating judgment about what it remembers.

2. Context, memory, and learning describe different mechanisms

A context window contains the information supplied to a model for a particular computation. An external memory system stores information that can be selected and brought into later computations. Changes to model parameters provide another route through which training can influence future behavior.

These mechanisms can interact, but they should be named separately. An agent may adapt during a task because its current context contains a useful observation. It may behave differently in a later session because a stored note is retrieved. Or it may develop a capability through training that changes its parameters.

For this article, **agent memory** means retained information that can influence future agent decisions. This operational definition includes task records, summaries, structured facts, and reusable procedures. It does not assume that an AI memory system works like biological memory.

The distinction gives researchers a way to specify an experiment. Does the agent retain a transcript? Can it create its own notes? Does a retrieval component select material? Is the environment itself persistent? Are model weights frozen? Each choice changes what the resulting evidence can support.

MemGPT provides an influential research example of memory management. Its authors propose virtual context management inspired by hierarchical storage in operating systems, moving information between memory tiers to support document analysis and multi-session conversation beyond a model's immediate context limits. [1]

Our inference is that memory should be examined as part of the configured agent system. The behavior depends on what is stored, how it is retrieved, what enters the context, and how the model uses it. Merely identifying the underlying model leaves much of this mechanism unspecified.

For a dataset buyer, that has an immediate implication: an environment should state what information can persist and which memory operations are available. Otherwise, comparisons may confuse memory architecture with model capability or differences in access to prior evidence.

3. A longer context does not settle the problem

Longer contexts can make more history directly available. Their usefulness still depends on whether the relevant information is found, interpreted correctly, and applied under appropriate conditions.

Liu and colleagues' *Lost in the Middle* studies multi-document question answering and key-value retrieval. In the models and tasks they examined, performance changed with the position of relevant information, often favoring material near the beginning or end. This is historical evidence about the tested systems, rather than a universal statement about every current model. [2]

The deeper design question remains useful even when a newer model handles positional retrieval well. What counts as relevant? A transcript may contain an original assumption, a later correction, and several intermediate guesses. Supplying all three does not automatically specify their relationship.

Imagine a repository note stating that a component uses version A of an interface. Later, a migration changes that interface to version B. A future task asks the agent to modify a dependent component. A retrieval system may correctly find both statements. The agent must still understand which applies to the current repository state.

Compression adds another consideration. A summary that says "the interface is unstable" may discard the distinction between a past migration and an unresolved defect. The compressed record can remain linguistically plausible while losing the information needed for action.

The practical objective is therefore selective continuity: preserving what matters, recovering it when needed, and retaining enough context to assess its applicability. A context window can support that objective, but the objective extends beyond input capacity.

This suggests several dataset opportunities. Researchers can vary the distance between observation and use, the amount of irrelevant material, the frequency of corrections, and the relationship between stored notes and current evidence. These variations should be chosen around a learning question, rather than used simply to make histories longer.

4. What memory research already teaches us

Several research directions provide useful foundations for studying agent memory. They address different settings and should be interpreted within those settings.

4.1 Remembering across conversations

LongMemEval evaluates five abilities in sustained assistant interactions: information extraction, multi-session reasoning, temporal reasoning, knowledge updates, and abstention. It also organizes memory-system design into indexing, retrieval, and reading. Its scope shows why remembering one fact and reasoning correctly over changing histories are separate demands. [3]

LoCoMo provides very long conversational data and evaluates capabilities including question answering, event summarization, and multimodal dialogue generation. Its machine-human construction process offers a research example of building histories whose events and relationships support memory-dependent questions. [4]

These benchmarks motivate more precise descriptions of memory tasks. A question can ask what happened, when it happened, how information changed, or whether the available history supports an answer. Each exercises a different relationship between retention and reasoning.

4.2 Carrying feedback into another attempt

Reflexion explores an agent framework that stores reflective text derived from feedback in an episodic memory buffer. Its approach influences subsequent attempts through linguistic feedback rather than updating model weights. [5]

For environment design, this suggests studying whether a retained lesson changes the next useful decision. The research question becomes particularly interesting when a past failure has several possible explanations and the agent must avoid converting an uncertain diagnosis into an unconditional rule.

4.3 Remembering how to do something

Voyager combines an automatic curriculum, an expanding library of executable skills, and iterative prompting with environmental feedback in Minecraft. It illustrates a form of procedural reuse in which accumulated code can support future behavior. [6]

These directions differ in architecture, domain, and experimental objective. Together they support an inference for this article: useful memory involves the relationship between experience and future decisions. Conversational recall, reflective notes, and reusable procedures provide complementary ways to investigate that relationship.

5. Four properties of memory that supports work

We propose four properties for organizing memory-dependent environments: persistence, provenance, revision, and reuse. This is an authoring framework developed for this article, rather than a standardized memory score.

5.1 Persistence: carrying the right information forward

The agent needs a record of information that would otherwise disappear. That record may concern an unresolved hypothesis, an important constraint, a completed action, or the state of a partial result.

Persistence should be purposeful. Keeping every raw tool output can consume resources and make relevant evidence harder to locate. Keeping only a polished conclusion can conceal uncertainty. A useful task creates a reason to preserve a particular piece of information and a later decision that depends on it.

5.2 Provenance: knowing where a belief came from

A remembered observation differs from a remembered interpretation. "The test failed after the dependency update" describes an event. "The dependency update caused the failure" is a causal inference. Treating the second as established evidence can narrow a later investigation prematurely.

A memory record can preserve the source, time, environment version, and degree of confidence associated with a statement. Researchers can then ask whether the agent uses those distinctions when deciding what to trust.

5.3 Revision: changing a belief when conditions change

Memory becomes useful over time when it can accommodate updates. The task might introduce a revised instruction, a corrected measurement, or a changed software interface. The agent should recognize whether the update replaces a prior fact, narrows its scope, or leaves it intact.

5.4 Reuse: applying experience at the right level

A lesson can be too specific or too broad. Remembering the exact command from one environment may fail when paths change. Remembering "always restart the service" may be dangerous when the earlier success depended on a temporary condition.

A more useful record identifies the conditions under which the procedure worked and what the outcome established. Environments can make those conditions vary, allowing researchers to examine whether the agent transfers a lesson appropriately.

6. From factual recall to operational consequences

A conversational benchmark may ask the agent to answer from its history. An operational environment can require the agent to act on that history. This creates an additional connection between memory and consequences.

Suppose the agent previously learned that a directory contains customer-generated artifacts that must be preserved. A later repair task includes a cleanup step. The relevant memory should influence which files the agent changes or removes.

Now suppose the directory has been moved. Blindly preserving the old path is insufficient. The agent needs to recover the underlying constraint, inspect the current state, and apply that constraint to the right location.

The following scenarios are proposed environment families. They are illustrative designs and do not describe an existing delivered Deaimer memory benchmark.

Scenario	What must persist	What changes later	Decision under examination
Interrupted repository repair	Evidence gathered and unresolved hypotheses	A dependency or module changes	Which earlier conclusions still apply?
Repeated terminal workflow	A discovered tool convention	Syntax or available utilities change	Can experience guide fresh interface discovery?
Scientific analysis revision	Data meaning and method assumptions	New observations or a corrected unit	Should the prior analysis be revised?
System recovery follow-up	Protected state and repair rationale	A restart or new failure condition	Can continuity coexist with reassessment?
Multi-session project	Decisions, constraints, and unfinished work	Requirements are amended	Can the agent resume without obsolete commitments?

The value of these tasks comes from the causal role of memory. If an agent can complete the later task using only newly supplied information, success may reveal little about persistence. Conversely, if all essential information is withheld from a memory-free baseline, the result should be interpreted as a test of retained access, rather than automatically as evidence of sophisticated reasoning.

A carefully scoped environment can distinguish these cases. Some tasks exercise factual retention. Others exercise selection, revision, or transfer. A portfolio should state which is intended.

7. Remembering mistakes without inventing lessons

Failure records can help agents avoid repeating costly actions. They can also reinforce a mistaken explanation.

Consider an illustrative debugging attempt. The agent changes a timeout, runs a test, and observes failure. It stores the lesson "increasing the timeout does not help." During the attempt, however, a separate dependency was unavailable. The failed test may not have evaluated the timeout hypothesis at all.

A better record would separate the action, the observation, and the unresolved explanation. It might state that the timeout was increased, the run failed before reaching the relevant stage, and the effect of the change remains unknown.

This distinction is useful for dataset design because it makes reflection accountable to evidence. An agent should not receive an easy path merely by producing a confident retrospective summary. The later environment can create a situation in which an overgeneralized lesson leads to the wrong decision.

There are several dimensions to explore. The earlier attempt may have succeeded for an incidental reason. It may have failed under a temporary condition. The next task may resemble the earlier one while differing in the feature that made the original lesson applicable.

These tasks can examine conditional reuse: does the agent remember what it learned, what it did not establish, and when its conclusion should be revisited?

For model developers, this offers a richer target than avoiding repeated commands. Sometimes repeating an action is justified because the conditions changed. The capability of interest is whether the agent can explain that difference through the evidence available to it.

8. Time changes the meaning of stored information

Temporal reasoning matters whenever an agent's history contains changes. A statement can be accurate about the past and unsuitable for a current decision.

An interface migration, an amended requirement, or an updated dataset can create several relevant moments: when a change was announced, when it took effect, and when the agent observed it. These moments need not be identical.

Imagine a task sequence in which a project requires output format A. A later instruction announces that new jobs must use format B, while previously completed jobs should remain in format A. The agent must apply the update to the correct objects. Replacing every old artifact would violate the distinction; ignoring the update would also be wrong.

A memory-dependent dataset can represent this through explicit events and evolving environment state. The difficulty should come from reasoning over the transition, with enough information to determine what applies.

Another useful variation concerns uncertainty. A stored observation may say that a service was healthy yesterday. That observation does not establish its current health. The agent needs to recognize when a fresh check is appropriate.

This is particularly relevant to coding and scientific work. A repository description can become stale after a commit. A numerical result can become obsolete after a unit correction. A procedure can remain valid while its input assumptions change.

The authoring objective is to preserve these distinctions rather than collapse them into a simplistic rule that newer information always wins. Source authority, scope, and effective time can matter alongside recency.

9. Memory and the environment must remain distinguishable

An agent can encounter two kinds of persistence: a record of prior experience and the current state left behind by earlier actions. These can agree, diverge, or reveal different parts of the task.

For example, a stored note may report that a configuration was changed. The environment can show whether that change remains present. If another actor reverted it, the memory describes history while the configuration describes the current system.

An experiment should specify which state persists across sessions. Does the workspace continue unchanged? Is it reset while the memory remains? Do both survive, with controlled modifications between episodes? Each setup asks a different question.

One useful comparison resets the environment but retains a learned procedure. Another preserves a partially completed workspace and asks the agent to resume. A third introduces an external change to determine whether the agent notices that its record has become incomplete.

These designs should make the roles of memory and observation explicit. An agent that reopens a file and discovers the relevant information is demonstrating environment investigation. An agent that retrieves a prior observation may avoid redundant work. Either can be sensible; their contribution to success should be assessed under the experimental question.

The relationship also affects interruption tasks. A good handoff record may help the agent resume even when the original transcript is unavailable. Its quality can be examined through what the agent does next, rather than through how comprehensive the record sounds.

10. Memory access needs a defined scope

Persistent information should be relevant to the user, project, and task for which it is used. Mixing unrelated histories can produce incorrect assumptions and inappropriate disclosure.

In an enterprise setting, two projects may use similarly named files but have different constraints. A lesson from one repository may refer to an internal dependency unavailable in another. A stored user preference may apply to one workflow and conflict with an explicit instruction in a new task.

An environment program can explore these boundaries with fictional, non-sensitive records. It can require the agent to retrieve only information belonging to the appropriate project, recognize the scope of a preference, and follow authorized changes to what should be retained.

Forgetting also has an operational meaning. If a fact is corrected or removed from an authorized memory store, the system should have a defined policy for derived summaries and cached copies. A study must specify which stores are under test before interpreting whether removal was effective.

The point is to make memory access part of task competence. Correct retention is useful when coupled with appropriate scope. An agent that remembers an irrelevant detail and applies it confidently can be less dependable than one that asks for clarification or inspects current evidence.

This section describes design considerations for memory environments. It does not imply that Deaimer has implemented a specific enterprise memory architecture or completed a particular privacy evaluation.

11. The dataset unit becomes a sequence of experiences

Memory changes what constitutes a useful example. A single instruction and a final answer may be insufficient to represent the capability being studied. The meaningful unit can become a sequence: acquire information, encounter an interruption or change, and later make a decision that depends on the earlier experience.

A proposed task sequence can include an initial environment, events that occur between sessions, a persistence policy, later objectives, and information about the intended capability. The sequence should also define what the agent can observe at each point.

This gives researchers control over causality. A fact introduced in an early episode may become relevant later. A correction can alter its scope. Distracting events can increase retrieval demands without changing the underlying answer. Different sequences can reserve distinct tools or failure mechanisms for transfer studies.

Recorded trajectories offer another possible resource. They can show how an agent wrote a memory, which information it retrieved, and how that information influenced a decision. The availability and permitted use of such records should be agreed for a particular delivery.

The authoring challenge is to make sequences coherent. A later event should follow logically from the environment, and the retained information should have a plausible role in work. Artificial chronology that exists only to hide an answer can produce a narrow memory puzzle with weak relevance to deployment.

For a dataset buyer, useful questions include: What persists? What changes? Which decisions require history? Which can be resolved by current observation? What is the intended transfer boundary? These questions clarify the value of a collection before a model experiment begins.

12. How Deaimer could support memory research

Deaimer publicly describes custom environments and private task collections spanning terminal operations, repository-level software engineering, scientific computing, and longer workflows. Its offerings include runnable environments and supporting assets, with package contents dependent on the collection and agreed scope. [7, 8]

These capabilities provide a foundation for discussing custom memory-dependent work. The extension proposed here is to organize selected tasks into purposeful sequences and introduce controlled interruptions or changes. Such a program could help a model developer study continuity, revision, and conditional reuse in familiar engineering domains.

A coding-agent team might commission repository sequences in which an investigation is interrupted and later resumed after a change. A scientific-agent team might request analyses whose assumptions must be revisited after a correction to data meaning. An operational-agent team might focus on retaining incident knowledge while reassessing current state.

These are possible custom scopes. They should not be read as claims that Deaimer already offers a published memory benchmark, that every existing dataset supports persistent sessions, or that an improvement has been measured.

The commercial opportunity is concrete. A model developer can identify a memory-related weakness and ask for environments that make that weakness observable. Deaimer can discuss authoring, domain coverage, runtime constraints, and the materials required for the intended study.

The buyer's experimental results would establish whether exposure to the collection improves a defined capability. Research on MemGPT, Reflexion, and conversational benchmarks supplies context for the field; it does not establish a performance effect for Deaimer's work.

This positioning connects the dataset to the development objective: helping agents maintain useful continuity while handling changes that make old information incomplete or misleading.

13. A practical study of memory-dependent work

The following experimental outline is a proposal for buyers and researchers. It avoids treating memory as a single number and focuses on the contribution of retained information to action.

First, define a narrow capability. Examples include resuming an interrupted repair, incorporating an amended requirement, or reusing a procedure under changed conditions. State what a successful decision would demonstrate.

Second, establish the memory conditions. Compare a system without cross-session memory with one that receives a full historical record and one that uses a selective memory mechanism, where feasible. Keep the underlying model and tool interfaces fixed unless they are the variable under study.

Third, disclose the budgets. Storage size, retrieval calls, supplied context, and total execution resources can affect results. A method that uses substantially more history or compute should be evaluated with that difference visible.

Fourth, reserve meaningful changes for evaluation. New repositories, tools, event patterns, or combinations can help distinguish transferable memory use from memorization of repeated task sequences. Choose the separation around the capability claim.

Fifth, examine outcomes and behavior together. Completion can be accompanied by measures of repeated investigative work, use of superseded facts, justified reassessment, and correct handling of scoped information. The categories should be limited to those relevant to the study.

Finally, include cases in which memory should have little effect. These can help reveal whether the mechanism introduces irrelevant assumptions or extra cost when current evidence is sufficient. A memory system should be useful selectively.

The strongest resulting claim will describe the conditions under which a particular system benefits from retained experience. It may also identify failures that guide the next environment brief: summaries that erase uncertainty, retrieval that favors obsolete records, or reuse that ignores changed assumptions.

14. Continuity with the ability to reconsider

An agent that performs extended work needs a usable relationship with its own history. It should carry forward relevant observations, preserve the reasons behind decisions, and recognize when new evidence changes what those decisions mean.

Memory research offers several approaches to that problem: managing information across storage tiers, evaluating sustained conversational histories, reflecting on feedback, and retrieving executable skills. Their differences make the field richer and make careful task design more important.

For coding, scientific, and operational agents, the next opportunity is to connect memory to meaningful consequences. Can the agent resume a repair without repeating the entire investigation? Can it revise an analysis after a corrected assumption? Can it reuse a lesson without treating it as a universal rule?

Deaimer's role in this direction can be practical: working with model developers to define custom sequences, construct executable environments, and provide relevant domain experiences. The resulting work should be assessed through experiments that separate retained access, reasoning, and adaptation.

The goal is continuity that remains open to correction. An agent's earlier experience should help it understand what to do next, while leaving room for the world to have changed.

Work with Deaimer

If your team is studying agent memory, interrupted workflows, or learning across repeated tasks, start with a concrete capability question. Deaimer can discuss whether a custom environment program or private task collection fits your intended study. Explore [custom RL environments](#) and [private coding datasets](#).

References

1. Packer, C., et al. (2023; revised 2024). MemGPT: Towards LLMs as Operating Systems. arXiv:2310.08560. <https://arxiv.org/abs/2310.08560>
2. Liu, N. F., et al. (2024). Lost in the Middle: How Language Models Use Long Contexts. Transactions of the Association for Computational Linguistics, 12, 157-173. https://doi.org/10.1162/tacl_a_00638
3. Wu, D., et al. (2025). LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. International Conference on Learning Representations. <https://arxiv.org/abs/2410.10813>
4. Maharana, A., Lee, D.-H., Tulyakov, S., Bansal, M., Barbieri, F., and Fang, Y. (2024). Evaluating Very Long-Term Conversational Memory of LLM Agents. Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 13851-13870. <https://aclanthology.org/2024.acl-long.747/>
5. Shinn, N., et al. (2023). Reflexion: Language Agents with Verbal Reinforcement Learning. arXiv:2303.11366. <https://arxiv.org/abs/2303.11366>
6. Wang, G., et al. (2023). Voyager: An Open-Ended Embodied Agent with Large Language Models. arXiv:2305.16291. <https://arxiv.org/abs/2305.16291>
7. Deaimer. RL Environments & Coding-Agent Evaluations. Public offering description. Accessed 12 October 2026. <https://www.deaimer.com/rl-environments>
8. Deaimer. Private Coding-Agent Datasets for Licensing. Public offering description. Accessed 12 October 2026. <https://www.deaimer.com/datasets>

Research perspective. The four-property framework, illustrative environments, and experimental outline are proposals developed for this article. Published findings belong to their cited studies. No private task content, customer data, or internal implementation details are disclosed.