STRUCTURED TURN MEMORY: SLOT-AWARE STORAGE, COMPRESSION, AND RETRIEVAL FOR LONG-HORIZON LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Model (LLM)-based agents that operate over extended horizons must store, compress, and retrieve interaction histories to sustain coherent planning across sessions and long reasoning chains. Existing memory systems treat each turn as a monolithic text block, so compression loses critical information, particularly the tool-call outputs, action decisions, and reasoning traces that constitute the structural spine of agent workflows. We propose **Structured Turn Memory (STM)**, a framework that decomposes each agent turn into four semantically typed slots: *Observation*, *Reasoning*, *Action*, and *Outcome*, and uses these slots as the unit of storage, compression, and retrieval. A rate-distortion analysis shows that slot-aware compression needs a lower rate than task-agnostic monolithic summarization whenever the gain from task-weighted allocation exceeds the redundancy among slots, a condition that holds with a wide margin on real agent traces. On four benchmarks spanning inter-session memory (LoCoMo, multi-session WebArena) and intra-session long-horizon memory (SWE-Bench Pro, Terminal-Bench 2.0), STM yields consistent improvements over strong baselines including Mem0, A-Mem, REMem, and Agent Workflow Memory: $+5.6$ memory recall F1 on LoCoMo, $+7.7\%$ task success rate on WebArena, $+6.7\%$ on SWE-Bench Pro, and $+7.6\%$ on Terminal-Bench 2.0. STM also outperforms the recent context managers MEM1, AgentFold, and GAM, and compared with keeping the full context, it offers a better cost-quality trade-off when histories fit in the window and larger gains when they do not. Ablations show that every slot contributes, with the action slot most critical for agentic tasks, and slot typing also improves existing memory systems.

1 INTRODUCTION

Large language model (LLM)-based agents have progressed from single-turn assistants to autonomous systems that reason, plan, and execute multi-step tasks through interaction with external environments(e.g., websites, codebases, and terminals) (Yao et al., 2022; Schick et al., 2023; Xi et al., 2025). A central challenge for deploying these agents in practical, long-running settings is *memory*: the ability to retain, organize, and recall information from prior interactions so that behavior remains coherent over extended time horizons (Weng, 2023; Sumers et al., 2023).

Existing agent memory architectures have approached this problem from several angles. MemGPT (Packer et al., 2023) pages information between main context and archival storage via LLM-initiated function calls, and Generative Agents (Park et al., 2023) synthesize higher-level reflections from a memory stream of natural language observations. More recent systems organize memory in richer ways: Mem0 (Chhikara et al., 2025) provides graph-structured storage with automatic extraction and consolidation, A-Mem (Xu et al., 2025) organizes memories through Zettelkasten-inspired linking, REMem (Shu et al., 2026) constructs hybrid memory graphs with time-aware episodic representations, and Agent Workflow Memory (Wang et al., 2025) extracts reusable task workflows from agent experience. Recent context managers instead learn to consolidate (Zhou et al., 2026), fold (Ye et al., 2025), or defer compression to query time (Yan et al., 2025).

Despite this progress, these systems share a common limitation: they treat each interaction turn as an *undifferentiated text block*. When compression is applied, whether through summarization, reflection,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

folding, or forgetting-curve decay, the result conflates observations, reasoning, actions, and outcomes into a single narrative. This creates three failure modes over long horizons. (1) **Action erasure.** Tool invocations, state mutations, and concrete outputs are among the most information-dense yet least narratively salient elements of a turn. Summarizers optimized for conversational coherence drop them, so over 20+ sessions the agent loses track of *what it actually did*, which is precisely the information needed for plan assessment and next-step selection. (2) **Reasoning collapse.** The chain of decisions that explains *why* an action was taken, including the alternatives considered and the binding constraints, disappears in a one-sentence summary, so the agent repeats failed strategies and cannot explain its past choices. (3) **Observation flattening.** Evidence, tool outputs, and environment signals are merged with conversational content, which erases the boundary between what the agent observed and what it inferred and blurs recalled facts with confabulations. Summarization-free strategies, such as keeping the full context or an append-only playbook (Zhang et al., 2026), avoid these losses only while the raw trace fits in the context window, which fails for a large share of long-horizon agentic episodes.

We argue that these failure modes reflect a structural mismatch between the *semantic heterogeneity* of agent interaction traces and the *homogeneous text assumption* underlying current memory systems. Existing designs vary how much to keep and when to compress, but the stored unit remains a homogeneous block of text. We propose **Structured Turn Memory (STM)**, which changes the unit itself: STM decomposes each turn into four typed semantic slots, Observation, Reasoning, Action, and Outcome, and stores, compresses, and retrieves each slot independently. Our approach rests on three insights: (1) slots differ in compressibility and task relevance, so they deserve different budgets, (2) slot-level indexing enables targeted retrieval that monolithic turn records cannot support, and (3) the action chain across turns is a queryable first-class object for long-horizon planning.

In summary, our contributions are:

- A structured turn representation with typed semantic slots, together with a rate-distortion analysis that characterizes when slot-aware compression outperforms monolithic compression.
- A slot-aware compression pipeline with four graduated levels: raw, slot-summarized, episode-merged, and archived, each with slot-specific strategies and invariants that preserve key information.
- A retrieval mechanism that combines slot-scoped queries, compression-level-aware ranking, and action-chain reconstruction, enabling work-aware recall.
- Experiments on four inter- and intra-session benchmarks, showing consistent gains over nine memory systems and context managers, a favorable comparison with summarization-free context management, and gains from transplanting slot typing into existing memory systems.

2 Related Works

Memory Architectures for LLM Agents. Persistent memory systems for LLM agents typically store interaction history as undifferentiated text and manage it through hierarchy, forgetting, or linking. MemGPT (Packer et al., 2023) manages a two-tier hierarchy through LLM-driven function calls, Generative Agents (Park et al., 2023) reflect over a timestamped memory stream without distinguishing structural roles within each entry, and MemoryBank (Zhong et al., 2024) applies an Ebbinghaus-inspired forgetting curve uniformly to all memory types. A-Mem (Xu et al., 2025) and Mem0 (Chhikara et al., 2025) add Zettelkasten-inspired linking and graph-structured storage, organizing *how* memories relate but not *what* internal structure each memory exposes. Recent context managers change the compression policy: MEM1 (Zhou et al., 2026) learns to overwrite a compact internal state at every turn, AgentFold (Ye et al., 2025) folds history at multiple scales, and GAM (Yan et al., 2025) keeps raw pages and compresses at query time, while ACE (Zhang et al., 2026) avoids summarization through an evolving playbook. These designs vary the rate and schedule of compression, but the stored unit remains a homogeneous turn, state, block, or page. STM makes typed sub-turn slots this unit and can be combined with these policies.

Learning and Reasoning from Agent Experience. Reflexion (Shinn et al., 2023) performs verbal reinforcement learning via an episodic reflection buffer, ExpeL (Zhao et al., 2024) extracts reusable insights from trajectories, Self-Refine (Madaan et al., 2023) iterates on self-feedback, and Voyager (Wang et al.) learns a skill library from open-ended exploration. For intra-turn reasoning, ReAct (Yao et al., 2022) interleaves reasoning traces with actions, while standard chain-of-thought

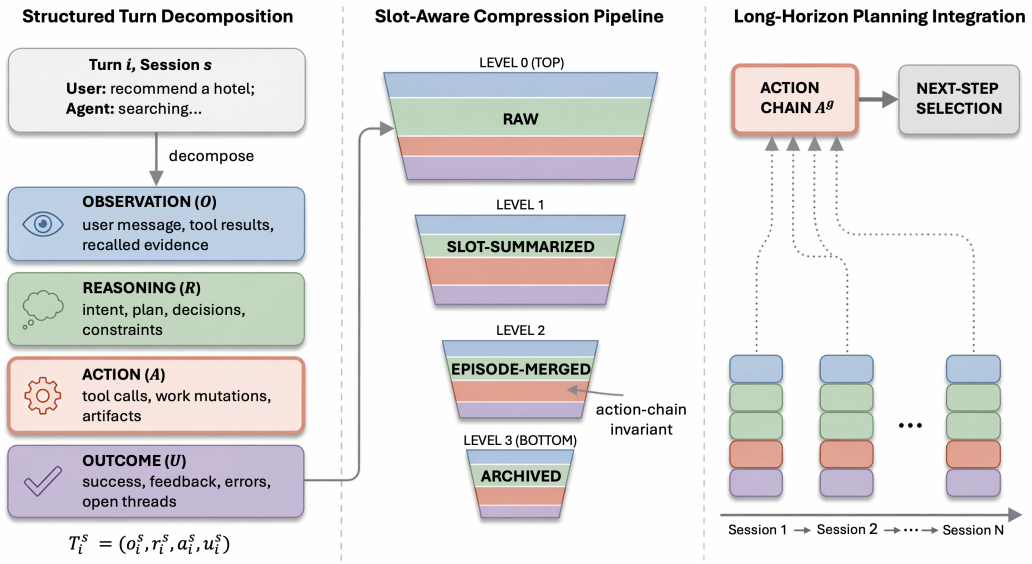

Figure 1: Illustration for the Structured Turn Memory framework.

prompting (Wei et al., 2022) and self-consistency (Wang et al., 2023) improve reasoning steps, and CoALA (Sumers et al., 2023) organizes agent capabilities into perception, memory, reasoning, and action modules. These works consume unstructured experience or structure the reasoning-action loop *within* a single turn. STM specifies how the products of that loop are preserved *across* turns, so that these mechanisms can operate on typed records rather than flat text.

Evaluation and Cognitive Foundations of Long-Horizon Memory. LoCoMo (Maharana et al., 2024) evaluates long-term conversational memory over multi-session dialogues of 300+ turns, and LongMemEval (Wu et al., 2025) benchmarks chat assistants on interactive memory. Recent work (He et al., 2026) shows that recall accuracy does not translate directly into effective memory use in goal-driven decision-making. Retrieval-augmented generation (Lewis et al., 2020; Gao et al., 2023; Karpukhin et al., 2020) and long-context modeling (Liu et al., 2024) sidestep explicit memory design but inherit the homogeneous-chunk assumption. Our slot decomposition draws on cognitive models in which heterogeneous memory roles demand distinct representations, including Atkinson and Shiffrin's multi-store model (Atkinson & Shiffrin, 1968), Tulving's episodic-semantic distinction (Tulving et al., 1972), and Anderson's ACT-R (Anderson, 2014); we operationalize this principle at the level of the *individual turn*.

3 Methodology

3.1 Structured Turn Representation

A *session* is a contiguous interaction episode that ends with a context reset, and a *turn* is one agent step together with the environment's response (e.g., a dialogue exchange, or a tool call and its result). Turn i of session s produces a raw trace $\mathbf{x}_i^s$ that is typically stored and compressed as a single text block. We instead decompose $\mathbf{x}_i^s$ into a structured turn record $\mathcal{T}_i^s = (\mathbf{o}_i^s, \mathbf{r}_i^s, \mathbf{a}_i^s, \mathbf{u}_i^s)$, where:

- $\mathbf{o}_i^s$ (**Observation**): the user message, tool results from prior dispatches, environment signals, and IDs of recalled evidence with retrieval reasons.
- $\mathbf{r}_i^s$ (**Reasoning**): intent resolution, plan state assessment, decision trace with rationale, and notes on binding constraints from the agent's profile or workspace.
- $\mathbf{a}_i^s$ (**Action**): the agent's response summary, tool invocations with input/output summaries, work-item mutations (created, advanced, blocked, closed), and artifact changes.
- $\mathbf{u}_i^s$ (**Outcome**): a success signal, explicit user feedback (corrections, approvals, rejections), error events, and unresolved threads deferred to future turns.

The decomposition is performed by an *Observation Pipeline* that runs after each turn completes. For LLMs that produce explicit chain-of-thought (e.g., via ReAct (Yao et al., 2022)), the pipeline extracts reasoning traces directly from the model output; otherwise, a lightweight extraction prompt identifies the implicit intent and decision points. Tool call records and state mutations are extracted programmatically from the execution context when the harness logs them, and are otherwise inferred from the emitted tool-call text.

Work items. STM tracks task-level intent through *work items*, persistent records with an ID, a title, a state in {open, active, blocked, closed}, and an optional parent (e.g., a GitHub issue and the files under modification). Work items are maintained by the Observation Pipeline without annotation: the extractor emits work mutations inside the action slot, and the memory controller applies them. A *created* mutation opens a new item unless it matches an open one (title-embedding cosine similarity above 0.82 and a shared artifact path), and other mutations update the matched item. Each slot is stored as an independent chunk, indexed by its turn ID, session ID, linked work-item IDs, slot type, and compression level. Appendix C defines sessions, turns, and work items for each benchmark.

3.2 Slot-Aware Compression

STM compresses structured turn records through four progressive levels, each with slot-specific strategies. Compression works by promotion: every slot chunk carries a compression level, and a promoted representation replaces the old one in the index, while the superseded text stays on disk if the storage budget allows. Because promotion is driven by age and work-item lifecycle, the live index always holds a mixture of levels, which the retriever exploits (§3.4).

Level 0: Raw. The full structured record is retained for the most recent N turns, where N is the raw retention window depending on the available context budget. No compression is applied at this level.

Level 1: Slot-Summarized. Each slot is compressed with a strategy matched to its type:

- $\mathbf{o}_i^s$: extract the user's intent, key entities, and task constraints; retain evidence IDs together with one-line recall reasons; drop verbose tool outputs, repeated context, and conversational filler.
- $\mathbf{r}_i^s$: retain each decision and its one-sentence rationale; drop unselected alternatives, intermediate evaluation steps, and speculative reasoning that does not affect the final action.
- $\mathbf{a}_i^s$: retain tool name, one-line input/output summary, mutation type, and target; drop full payloads. The action summary may not be shorter than τ_a tokens, which enforces a minimum action fidelity; τ_a depends on tool-use complexity and the available budget.
- $\mathbf{u}_i^s$: retain verbatim, since outcome slots are short; open threads are preserved until resolved.

The target compression ratio is 0.2–$0.4\times$ of raw. This level is triggered at session boundaries and whenever the raw buffer exceeds N turns.

Level 2: Episode-Merged. Level-1 turns that share a work-item ID within a grouping window of W turns are merged into *work episode summaries*:

$$\mathcal{E}_g^s = \text{Merge}(\{\mathcal{T}_i^s \mid \text{work_item}(\mathcal{T}_i^s) = g,\, i \in [i_{\text{start}}, i_{\text{end}}]\}), \quad i_{\text{end}} - i_{\text{start}} < W. \tag{1}$$

The merge preserves the *net* observation (first and last significant inputs), *net* decision chain, *net* state change, and final outcome, with a target ratio of 0.05–$0.15\times$ of raw. It is triggered by session end or a work-item state change in inter-session settings, and by more than M accumulated Level-1 turns in intra-session settings.

Level 3: Archived. Episodes are reduced to a single memory record stating the goal, the decisions, the products, the outcome, and the corrections applied, with a target ratio of $\leq 0.05\times$ of raw. This step is triggered by scheduled maintenance or work-item closure.

Compression invariants. Across all levels, four invariants are enforced: (1) the action chain for any work item is reconstructable, (2) every work mutation traces to at least one decision in the reasoning slot, (3) correction chains survive compression, and (4) slot boundaries remain queryable even in merged records.

3.3 INFORMATION-THEORETIC ANALYSIS

We analyze slot-aware compression with rate-distortion theory (Shannon, 1948; Cover, 1999). Let $X = (O, R, A, U)$ be a turn with slot variables X_s, $s \in \{o, r, a, u\}$, and let $\hat{X}$ be its compressed representation under a rate constraint ρ (maximum number of output tokens). Downstream use of memory is captured by a task-weighted distortion

$$d_{\text{task}}(X, \hat{X}) = \sum_{s \in \{o,r,a,u\}} w_s \, d_s(X_s, \hat{X}_s), \qquad \sum_s w_s = 1, \tag{2}$$

where d_s is a slot-specific distortion and a larger w_s marks a more task-critical slot (e.g., action).

Slot-aware compression. A slot-aware compressor f_{slot} codes each slot with its own codebook and splits the budget as $\rho = \rho_o + \rho_r + \rho_a + \rho_u$. Its minimum rate at task-weighted distortion D is

$$R_{\text{slot}}(D) = \min_{\{D_s\}} \sum_{s \in \{o,r,a,u\}} R_s(D_s) \quad \text{s.t.} \quad \sum_s w_s D_s \le D, \tag{3}$$

where $R_s(\cdot)$ is the rate-distortion function of slot s. At the optimum, $R_s'(D_s) = -\lambda w_s$ for a common multiplier $\lambda \ge 0$, so bits flow to slots with high task weight or steep rate-distortion curves.

Monolithic compression. A monolithic compressor f_{mono} codes X with a single joint codebook. Practical summarizers are task-agnostic: the per-slot distortions D_s^{mono} they induce follow their own objective, such as narrative coherence, rather than w. Let $R_{\text{mono}}(D)$ be the rate such a compressor needs to satisfy $\mathbb{E}[d_{\text{task}}(X, \hat{X})] \le D$. A joint codebook can, however, exploit redundancy among slots, measured by the total correlation $\mathrm{TC}(X) = \sum_s H(X_s) - H(X)$.

Proposition 1 (Slot-aware compression advantage)**.** *Let $(D_s^{\text{mono}})_s$ be the per-slot distortions of a monolithic compressor with $\sum_s w_s D_s^{\text{mono}} \le D$, and define the allocation gain $G_{\text{alloc}}(w) = \sum_s R_s(D_s^{\text{mono}}) - R_{\text{slot}}(D) \ge 0$. Then*

$$R_{\text{mono}}(D) - R_{\text{slot}}(D) \ge G_{\text{alloc}}(w) - \mathrm{TC}(X). \tag{4}$$

Slot-aware compression thus requires a strictly lower rate whenever $G_{\text{alloc}}(w) > \mathrm{TC}(X)$. For independent slots it is never worse, and it is strictly better unless the monolithic distortion profile already solves Eq. (3)*.*

Proof sketch. The monolithic profile is feasible for Eq. (3), so $G_{\text{alloc}}(w) \ge 0$. For any joint code, $H(X \mid \hat{X}) \le \sum_s H(X_s \mid \hat{X}_s)$, which gives $I(X; \hat{X}) \ge \sum_s I(X_s; \hat{X}_s) - \mathrm{TC}(X)$. Each marginal term is at least $R_s(D_s^{\text{mono}})$, so $R_{\text{mono}}(D) \ge \sum_s R_s(D_s^{\text{mono}}) - \mathrm{TC}(X)$, and subtracting $R_{\text{slot}}(D)$ yields Eq. (4). Appendix A gives the full proof. □

The allocation gain rewards separating slots whose task weights and compressibility differ, while the total correlation is the shared information that separate codebooks forgo. Much inter-slot dependence is relational (a decision explains a mutation), and STM stores it as explicit decision-to-mutation links (invariant 2) at a cost of a few tokens. We estimate both terms in §4.2.

3.4 SLOT-SCOPED RETRIEVAL

STM extends standard retrieval with slot-aware indexing, structured retrieval requests, compression-aware ranking, and action-chain reconstruction.

Per-chunk indexing. Each slot chunk is embedded independently with the same embedding model, and its slot type and compression level are stored as metadata. Using o, r, a, u as slot-type labels, both lexical (BM25 (Robertson & Zaragoza, 2009)) and dense (FAISS (Johnson et al., 2019)) indexes are maintained per chunk rather than per turn.

Retrieval requests. A request specifies a query q, a target slot set $\mathcal{S}_{\text{target}} \subseteq \{o, r, a, u\}$, a budget k, and a scope, which is *recent* if the query concerns the active plan step and *historical* otherwise. For explicit questions, as in LoCoMo, q is the question and all four slots are targeted. In agentic tasks, the memory controller emits the request as JSON before each step, using a short prompt over the task instruction, the active plan step, the last tool result, and the active work item. The prompt asks for concrete entities (file paths, URLs, identifiers, error strings) and maps what was seen, why, what was done, and what happened to o, r, a, and u (Appendix B.3).

Compression-aware ranking. Given a request and the active work item g, each candidate chunk c at compression level $\ell(c) \in \{0, 1, 2, 3\}$ is scored as

$$\begin{aligned}\text{Score}_{\text{final}}(c) = \alpha \cdot \text{Sim}(q, c) + \beta \cdot \text{WorkRel}(c, g) + \gamma \cdot \text{Recency}(c) \\ + \delta \cdot \text{SlotBonus}(c, \mathcal{S}_{\text{target}}) + \epsilon \cdot \text{LevelPrior}\big(\ell(c), \text{scope}(q)\big),\end{aligned} \tag{5}$$

where Sim combines lexical and dense similarity, and $\text{WorkRel}(c, g)$ is 1 if c is linked to g, 0.5 if it is linked to a related item in the same lineage, and 0 otherwise. $\text{Recency}(c) = \exp(-\Delta t/\tau)$ acts as a validity prior, since the agent changes its environment and older records are more likely to be stale. Records with open threads or inside correction chains are exempt from this decay, and superseded records are handled by explicit replacement links. $\text{SlotBonus}(c, \mathcal{S}_{\text{target}}) = \mathbb{I}[\text{slot}(c) \in \mathcal{S}_{\text{target}}]$ boosts chunks of the requested slot types. LevelPrior favors detailed records for recent scope and merged records for historical scope, with values $(1.0, 0.7, 0.3, 0.0)$ and $(0.2, 0.6, 1.0, 0.8)$ over Levels 0–3, respectively. The weights $(\alpha, \beta, \gamma, \delta, \epsilon)$ are non-negative and sum to one.

Slot-scoped queries. Let $\mathcal{C}_\sigma$ denote the chunk index set for slot type σ. The retriever restricts candidates to the requested slot types and returns the top-k chunks under the final score:

$$\text{Retrieve}(q, \mathcal{S}_{\text{target}}, k) = \text{TopK}\Big(\big\{\text{Score}_{\text{final}}(c) \,\big|\, c \in \bigcup_{\sigma \in \mathcal{S}_{\text{target}}} \mathcal{C}_\sigma\big\}, k\Big). \tag{6}$$

This enables targeted queries such as "what tools did we use for task X?" with target slot set $\{a\}$, or "why did we choose MongoDB?" with target slot set $\{r\}$. When a chunk and its merged ancestor are both live, only the more detailed one that fits the budget is kept, and a selected Level-2 or Level-3 action record may be expanded by one hop into its constituents within the remaining budget. Chunks enter the prompt in slot order O, R, A, U with boundaries preserved.

Action-chain reconstruction. For a work item g, the retriever also collects all linked action-slot chunks and orders them by turn index. The resulting chain $\mathbf{A}^g$ is injected into the context, giving long-horizon plan assessment access to the structural spine of prior work.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on four benchmarks that require long-horizon execution with persistent memory. *LoCoMo* (Maharana et al., 2024) evaluates *memory recall fidelity* over long conversations. We use its public release of 10 conversations, each spanning ∼300 turns and 9K tokens on average over up to 35 sessions, with 1,986 QA questions in five categories: single-hop, multi-hop, temporal-reasoning, open-domain-knowledge, and adversarial; following convention, adversarial questions are excluded from the aggregate F1. *WebArena* (Zhou et al., 2024) features 812 long-horizon tasks across four self-hosted web applications (shopping, forums, content management, and maps). We build a *multi-session* variant in which 200 complex tasks are split into dependent sub-goals across 2–4 sessions that refer to earlier findings only implicitly, so the agent must recall prior navigation states, extracted information, and partial completions (Appendix C.2). *SWE-Bench Pro* (Deng et al., 2025) extends SWE-Bench with long-horizon tasks that resolve real GitHub issues in large repositories, demanding memory of file structures, prior edits, and debugging decisions; we use its 731-task public subset. *Terminal-Bench 2.0* (Merrill et al., 2026) contains hard, realistic command-line tasks spanning software engineering, machine learning, security, data science, etc., where earlier command outputs inform later decisions; we use its 89 provided high-quality tasks.

Baselines. We compare against six memory systems: *Mem0* (Chhikara et al., 2025) (graph-structured extraction and consolidation), *A-Mem* (Xu et al., 2025) (Zettelkasten-inspired linking), *REMem* (Shu et al., 2026) (a hybrid graph of time-aware gists and facts with an agentic retriever), *Agent Workflow Memory (AWM)* (Wang et al., 2025) (reusable workflows induced from experience), *HippoRAG 2* (Gutiérrez et al., 2025) (structured non-parametric memory), and *ReasoningBank* (Ouyang et al., 2025) (reasoning patterns retrieved from prior tasks). We also compare with the context managers *MEM1* (Zhou et al., 2026), *AgentFold* (Ye et al., 2025), and *GAM* (Yan et al., 2025) on controlled subsets of up to 100 instances per benchmark, and with four summarization-free references: recency truncation of raw history, retrieval over raw turns, an ACE-style append-only playbook (Zhang et al., 2026), and full uncompressed context, truncated to the latest 400K tokens when the trace does not fit.

Table 1: Main results across four benchmarks. Best results in **bold**, second-best underlined. All numbers are mean ± std over 5 seeds. SR = Success Rate, PP = Partial Progress.

Method	LoCoMo	WebArena		SWE-Bench Pro	Terminal-Bench
	QA F1	SR	PP	SR	SR
Mem0	$46.3_{\pm 1.4}$	$28.4_{\pm 2.1}$	$51.7_{\pm 1.8}$	$32.6_{\pm 1.9}$	$34.8_{\pm 2.2}$
A-Mem	$48.7_{\pm 1.3}$	$30.1_{\pm 2.3}$	$53.2_{\pm 2.0}$	$34.1_{\pm 1.7}$	$36.4_{\pm 2.0}$
REMem	$51.2_{\pm 1.1}$	$27.6_{\pm 2.5}$	$49.8_{\pm 2.2}$	$31.4_{\pm 2.1}$	$33.0_{\pm 2.4}$
AWM	$44.1_{\pm 1.6}$	$33.8_{\pm 2.0}$	$57.4_{\pm 1.7}$	$37.5_{\pm 1.8}$	$39.1_{\pm 1.9}$
HippoRAG 2	$47.9_{\pm 1.2}$	$29.7_{\pm 2.2}$	$52.1_{\pm 1.9}$	$33.8_{\pm 2.0}$	$35.7_{\pm 2.1}$
ReasoningBank	$45.8_{\pm 1.5}$	$31.2_{\pm 2.4}$	$54.6_{\pm 2.1}$	$35.9_{\pm 1.6}$	$37.5_{\pm 1.8}$
STM (Ours)	$\mathbf{56.8}_{\pm 1.0}$	$\mathbf{41.5}_{\pm 1.8}$	$\mathbf{66.3}_{\pm 1.5}$	$\mathbf{44.2}_{\pm 1.4}$	$\mathbf{46.7}_{\pm 1.6}$

Table 2: Summarization-free context management on the full benchmarks (top) and recent context managers on controlled subsets with identical backbone, budgets, and splits (bottom). Means over 5 seeds; full context is truncated to the latest 400K tokens when needed.

Method	LoCoMo F1	WebArena SR	SWE-Bench Pro SR	Terminal-Bench SR
Recency truncation	41.2	24.7	26.8	29.4
Retrieval over raw turns	49.6	26.3	19.4	24.3
ACE-style playbook	50.8	35.9	39.1	41.3
Full context	54.1	37.2	38.6	40.2
STM (Ours)	56.8	41.5	44.2	46.7
STM + ACE-style playbook	**57.1**	**43.8**	**46.4**	**49.0**
MEM1 (in-context)	48.9	34.6	38.4	40.0
AgentFold	50.4	37.2	40.2	42.5
GAM	52.7	36.4	39.6	41.8
STM (Ours)	**56.3**	**41.2**	**44.6**	**46.7**

Evaluation Metrics. We report token-level *Memory Recall F1* on LoCoMo (Maharana et al., 2024) and *Task Success Rate* under each benchmark's native auto-grader on the agentic benchmarks. On WebArena, *Partial Progress* is the fraction of sub-goals completed before the first failure. On WebArena and SWE-Bench Pro, *Action Chain F1* compares the reconstructed action history with the ground-truth step sequence.

Implementation Details. All methods share the same backbone, embedding model, and budgets: GPT-5-mini (`openai/gpt-5-mini`, released August 7, 2025; temperature 0.7, top-p 1.0, at most 4,096 output tokens per step), text-embedding-3-small (`openai/text-embedding-3-small`) for dense retrieval, a 400K-token context cap, an 8,192-token per-step budget for retrieved memory, on-disk storage of at most $1.5\times$ the raw trace, and $k = 20$ retrieved chunks of at most 256 tokens. Each method is tuned with the same 24-configuration grid on a held-out tuning split disjoint from evaluation. STM uses $N = 8$, $\tau_a = 40$ tokens, $W = 10$, $M = 20$, and $(\alpha, \beta, \gamma, \delta, \epsilon) = (0.315, 0.270, 0.135, 0.180, 0.100)$, with a fixed 200-token slot decomposition prompt applied post-turn. We report mean ± std over 5 seeds (all controls in Appendix B.6).

4.2 Results and Analysis

Main Results. As shown in Table 1, STM improves over all baselines on all four benchmarks. On LoCoMo, it improves QA F1 by $+5.6$ points over the strongest baseline (REMem at 51.2), with the largest gains on multi-hop ($+7.5$) and temporal ($+6.9$) questions that require cross-session synthesis and event ordering (Appendix D.1). On WebArena, SWE-Bench Pro, and Terminal-Bench 2.0, STM outperforms AWM, the strongest agentic baseline, by $+7.7\%$, $+6.7\%$, and $+7.6\%$ in success rate. AWM captures reusable task patterns but not the action-decision provenance within each workflow instance, which slot decomposition preserves. All gains are significant (paired bootstrap, Holm-corrected $p \leq 0.002$; Cohen's d of 1.9–3.4) and persist under matched retrieved tokens and equally fine baseline chunking (Appendix D.2). The consistent gap across web navigation, code editing, and CLI operations suggests that structured turn storage is a domain-general improvement.

Comparison with Summarization-Free Context Management. Avoiding summarization is a strong strategy (Table 2, top): full context outperforms every memory system in Table 1, and the ACE-style

Table 3: Action-information retention at different compression levels on held-out SWE-Bench Pro turns. ROUGE-L compares the compressed action slot with the raw original.

Method	Comp. Ratio	Action ROUGE-L	Decision F1	Tokens/Turn
Monolithic Summary	0.18	$0.42_{\pm 0.05}$	$0.35_{\pm 0.04}$	$127_{\pm 18}$
Mem0 Consolidation	0.23	$0.46_{\pm 0.04}$	$0.39_{\pm 0.04}$	$162_{\pm 19}$
HippoRAG 2 Formation	0.21	$0.49_{\pm 0.05}$	$0.42_{\pm 0.03}$	$148_{\pm 17}$
STM Level 1	0.28	$\mathbf{0.79_{\pm 0.03}}$	$\mathbf{0.72_{\pm 0.03}}$	$198_{\pm 14}$
STM Level 2	0.10	$0.71_{\pm 0.04}$	$0.64_{\pm 0.04}$	$71_{\pm 11}$
STM Level 3	0.04	$0.58_{\pm 0.05}$	$0.53_{\pm 0.05}$	$28_{\pm 6}$

Table 4: Ablations on SWE-Bench Pro. Left: components; right: slot configurations.

Component	SR	Chain F1
Full STM	**44.2**	**78.6**
− Slot decomposition	34.5	51.8
− Slot-aware compression	39.8	67.3
− Action-chain retrieval	37.1	58.2
− Slot-scoped queries	41.6	73.5
− Compression invariants	40.9	71.2
Chronological retrieval	35.7	55.1

Slot configuration	SR	Chain F1
− Observation	40.2	74.1
− Reasoning	39.6	72.4
− Outcome	41.3	75.8
Merge O + U	42.0	76.9
Merge R + A	38.9	63.5
Action only	38.4	76.1
Without action (O, R, U)	33.1	36.4

playbook does so on the agentic benchmarks. Both, however, depend on the raw trace fitting in the 400K window, which holds for all LoCoMo dialogues but only for 78%, 39%, and 53% of WebArena, SWE-Bench Pro, and Terminal-Bench 2.0 episodes (Appendix D.3). On SWE-Bench Pro episodes that fit, STM slightly exceeds full context (49.6% vs. 47.8% SR) with 8–30% of its prompt tokens. On the remaining episodes, full context degrades to recency truncation (32.7%), while STM reaches 40.7%. Even on LoCoMo, full context trails STM most on temporal questions (44.2 vs. 49.7 F1), consistent with the lost-in-the-middle effect (Liu et al., 2024). STM thus offers a better cost-quality trade-off when histories fit and a clear accuracy gain when they do not. Since the playbook curates cross-task strategies and STM preserves within-task provenance, combining them adds 2.2–2.3 points on the agentic benchmarks.

Comparison with Recent Context Managers. Under identical backbone, budgets, top-k, and splits, STM leads the strongest of MEM1, AgentFold, and GAM by 3.6–4.4 points on every benchmark (Table 2, bottom). For GPT-5-mini, MEM1's consolidation objective is reimplemented in context, since its released policy is an RL-trained 7B model; on a Qwen3-14B backbone (Yang et al., 2025) that runs this policy directly, STM still leads on SWE-Bench Pro (30.4% vs. at most 26.8%; Appendix D.4). These managers adapt the compression rate or schedule over homogeneous blocks and thus compose with STM: folding STM slot records with AgentFold raises SR from 44.6% to 46.0% (AgentFold alone: 40.2%).

Compression Efficiency Analysis. At comparable compression ratios (Table 3; STM Level 2 at $0.10\times$ vs. monolithic at $0.18\times$), STM retains $+29$ points higher Action ROUGE-L and Decision F1. This is consistent with Proposition 1: slot-aware compression allocates more bits to task-critical action content. Mem0's consolidation and HippoRAG 2's memory formation outperform naive summarization but still treat action traces and conversational content as homogeneous.

Empirical Check of Proposition 1. On 500 held-out SWE-Bench Pro turns, the total correlation among slots is 0.42 ± 0.06 nats per turn, only about 11% of the turn entropy. The allocation gain at compression ratio 0.2 is 1.63 ± 0.11 nats, so Eq. (4) guarantees an advantage of at least 1.21 nats per turn (Appendix D.5). At ROUGE-L ≥ 0.85, observation, reasoning, action, and outcome compress to $0.31\times$, $0.27\times$, $0.48\times$, and $0.72\times$. The outcome slot is the least compressible but short (38 tokens), so STM keeps it verbatim. The action slot is the only one that is hard to compress, heavily weighted, and sizable (224 tokens), which motivates the τ_a floor. Monolithic compression also reduces the mutual information between the action chain $\mathbf{A}^g$ and the goal state by 41.3% at Level 2, versus 12.7% for slot-aware compression (Appendix D.6).

Ablation Studies. Table 4 reports ablations on SWE-Bench Pro. (1) **Slot decomposition is the most impactful component.** Monolithic storage costs 9.7% SR, and chronological trajectory retrieval

Table 5: Behavioral effect of removing each slot on SWE-Bench Pro. ↓: lower is better.

Removed	Behavioral metric	Full	Ablated
Reasoning	Redundant strategy (%) ↓	8.4	19.7
Observation	Redundant tool call (%) ↓	11.2	27.5
Outcome	Error recovery (%)	61.3	44.8
Action	Action Chain F1	78.6	36.4

Table 6: SWE-Bench Pro SR after transplanting slot-typed storage into baselines.

System	Native	+ Slots	Δ
Mem0	32.6	38.1	+5.5
A-Mem	34.1	38.8	+4.7
AWM	37.5	41.0	+3.5

without slot scoping costs 8.5%. (2) **Action-chain retrieval is critical for long horizons.** Removing it costs 7.1% SR and 20.4 Action Chain F1. (3) **No slot is redundant.** Removing observation, reasoning, or outcome costs 4.0, 4.6, and 2.9 points, and merging reasoning into action costs 5.3 points. (4) **The action slot is the most valuable for agentic tasks.** The action slot alone (38.4%) outperforms all other slots together (33.1%). On LoCoMo, by contrast, removing observation causes the largest drop (−8.9 F1; Appendix D.12). Slot-scoped queries (−2.6%) and compression invariants (−3.3%) contribute less but meaningfully.

Slot-Level Mechanisms. Table 5 measures the failure mode that each slot is meant to prevent. Removing reasoning more than doubles re-attempts of abandoned strategies, and removing observation more than doubles redundant tool calls. Without the outcome slot, error recovery drops from 61.3% to 44.8% and open-thread completion from 82.1% to 66.5%. Without the action slot, the agent can no longer reconstruct its prior edits. Adding slot-typed storage to Mem0, A-Mem, and AWM, without changing their consolidation or retrieval, also improves them by 3.5–5.5 points (Table 6).

Computational Overhead. STM has the lowest per-turn latency among the memory systems (142ms vs. 156ms for Mem0, 198ms for AWM, and 224ms for REMem), because slot extraction is a single ∼200-token prompt per completed turn and compression is triggered lazily. It retrieves 21–31% fewer tokens per step (2,890) with comparable storage (820 tokens per turn), since slot scoping and the level prior surface compact records (Appendix D.8). Retrieval over raw turns is faster (∼75ms) but reaches only 19.4% SR on SWE-Bench Pro.

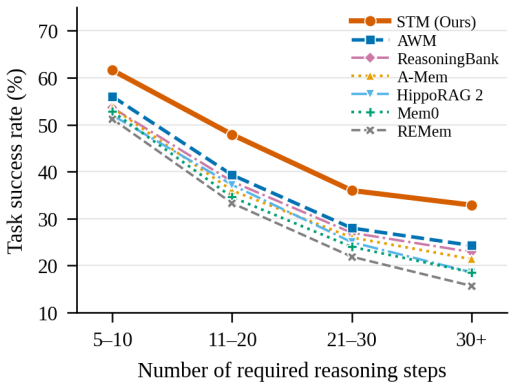

Figure 2: Task success rate on Terminal-Bench 2.0, grouped by number of reasoning steps required.

Scaling with Task Horizon. Figure 2 bins Terminal-Bench 2.0 tasks by the number of required reasoning steps. All methods degrade with horizon length, but STM degrades most gracefully. From 5–10 to 30+ steps, STM falls from 61.6% to 32.9% and retains 53% of its short-horizon success rate, while the baselines retain 31–43% (e.g., 56.0% → 24.3% for AWM). STM leads the best baseline, AWM, in every bin, and its lead is larger beyond 10 steps (+8.0 to +8.6) than at 5–10 steps (+5.6), consistent with episode-merged records preserving the net decision chain.

5 Conclusions and Future Work

We presented Structured Turn Memory, a framework that decomposes agent interaction traces into semantically typed slots and uses them as the unit of storage, compression, and retrieval. Our analysis shows that slot-aware compression needs a lower rate than task-agnostic monolithic compression whenever the allocation gain exceeds inter-slot redundancy, which holds with a wide margin on real agent traces. Across four benchmarks, STM outperforms memory systems and recent context managers. Relative to full context, it is slightly more accurate at a fraction of the cost when histories fit and clearly more accurate when they do not, and slot typing also improves existing memory systems when transplanted. The action slot stands out as hard to compress, sizable, and the most valuable for agentic tasks, while the observation slot dominates conversational recall. Future directions include learning consolidation policies over slots, adapting budget allocation to downstream performance, supporting richer modalities (images, code diffs, structured data), and scaling STM to production agents operating over months-long horizons.

AI USE STATEMENT

In this work, we used generative AI tools solely to edit the manuscript for readability. Concretely, an LLM-based writing assistant was applied to author-written drafts to correct grammatical errors, adjust word choice, and improve the clarity and flow of the prose. We did not use generative AI tools for any of the tasks that require disclosure. The conceptual framework of Structured Turn Memory, the formulation and proof of Proposition 1, the design of the slot-aware compression and slot-scoped retrieval methodology, the implementation of our system and of the baseline comparisons, the benchmark adaptation protocols, and the interpretation of all experimental results were developed entirely by the authors. We did not generate synthetic data, propose or refine hypotheses, clean or reformat datasets, or perform translation with generative AI tools. All AI-assisted edits were reviewed line by line by at least two authors to confirm that the technical meaning of the original text was preserved and that no unsupported claims, altered numerical results, or fabricated citations were introduced. We take full responsibility for the final content of this work, including all text, claims, and artifacts.

ETHICS STATEMENT

The authors have read and adhere to the ICLR Code of Ethics. This work does not involve human subjects, user studies, or the collection of any new data. All experiments are conducted on publicly released benchmarks (LoCoMo, WebArena, SWE-Bench Pro, and Terminal-Bench 2.0) in accordance with their respective licenses and intended research use; our multi-session WebArena variant is constructed by recomposing existing task specifications and introduces no new human-generated content. We note one consideration that is intrinsic to the problem setting: STM is designed to persist agent interaction histories across sessions, and the observation slot may retain user messages, retrieved evidence, and tool outputs that contain personal or proprietary information. Practical deployments should therefore pair the framework with appropriate access control, retention limits, and deletion mechanisms. Our schema supports this through explicit lifecycle states and per record identifiers that allow targeted removal of stored content, and the slot boundaries make it possible to purge sensitive observation records while preserving non sensitive action and outcome traces. Because the compression and extraction components are LLM based, stored records may inherit the biases and factual errors of the underlying model, and we do not claim that retained summaries are faithful representations of user intent in safety critical settings. The proposed method improves the coherence of long horizon agents rather than expanding their action space, so we do not foresee direct dual use risks beyond those already associated with general purpose LLM agents. The authors declare no conflicts of interest or sponsorship that could have influenced the reported results.

REPRODUCIBILITY STATEMENT

We have taken several steps to ensure that the results reported in this paper can be reproduced. The structured turn representation, the four level compression pipeline, and the slot scoped retrieval mechanism are specified in Section 3, and the main hyperparameters and budgets are listed in Section 4.1. Appendix B provides the slot extraction prompt, the per slot compression instructions, the retrieval request template, the retrieval pipeline configuration including BM25 and HNSW index settings, the storage schema, the computational resources and API cost, and a consolidated table of all control variables shared across methods. Appendix C defines sessions, turns, and work items for each benchmark and documents how each benchmark was used or adapted, including the construction rules for our multi session WebArena variant and the compression triggers for the inter session and intra session regimes, so that the same protocols can be reconstructed from the public benchmark releases. For our theoretical claim, the assumptions and a complete proof of Proposition 1 are given in Appendix A, together with a remark on the gap between the asymptotic analysis and finite length compression by language models, and Appendix D.5 and D.6 report the empirical estimates of the quantities in that analysis. All reported numbers are means and standard deviations over five random seeds. An anonymized implementation for our proposed method, including all prompts and configurations, is available at `https://anonymous.4open.science/r/STM-2687`.

REFERENCES

John R Anderson. *Rules of the mind*. Psychology Press, 2014.

Anthropic. Introducing Claude Opus 4.5. https://www.anthropic.com/news/claude-opus-4-5, 2025. Released November 24, 2025.

Richard C Atkinson and Richard M Shiffrin. Human memory: A proposed system and its control processes. In *Psychology of learning and motivation*, volume 2, pp. 89–195. Elsevier, 1968.

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

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, et al. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, Haofen Wang, et al. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1):32, 2023.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From rag to memory: Non-parametric continual learning for large language models. In *International Conference on Machine Learning*, pp. 21497–21515. PMLR, 2025.

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, et al. Memoryarena: Benchmarking agent memory in interdependent multi-session agentic tasks. *arXiv preprint arXiv:2602.16313*, 2026.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE transactions on big data*, 7(3):535–547, 2019.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 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)*, pp. 13851–13870, 2024.

Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

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

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

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

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Yiheng Shu, Saisri Padmaja Jonnalagedda, Xiang Gao, Bernal Jiménez Gutiérrez, Weijian Qi, Kamalika Das, Huan Sun, and Yu Su. Remem: Reasoning with episodic memory in language agent. In *The Fourteenth International Conference on Learning Representations*, 2026.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Theodore Sumers, Shunyu Yao, Karthik R Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2023.

Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Endel Tulving et al. Episodic and semantic memory. *Organization of memory*, 1(381-403):1, 1972.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *International Conference on Machine Learning*, pp. 63897–63911. PMLR, 2025.

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

Lilian Weng. Llm powered autonomous agents, June 2023. URL `https://lilianweng.github.io/posts/2023-06-23-agent/`. Lil'Log.

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

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

BY Yan, Chaofan Li, Hongjin Qian, Shuqi Lu, and Zheng Liu. General agentic memory via deep research. *arXiv preprint arXiv:2511.18423*, 2025.

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

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

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, et al. Agentfold: Long-horizon web agents with proactive context management. *arXiv preprint arXiv:2510.24699*, 2025.

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

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024.

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

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *International Conference on Learning Representations*, volume 2026, pp. 58413–58438, 2026.

CONTENTS

1 Introduction **1**

2 Related Works **2**

3 Methodology **3**

3.1 Structured Turn Representation 3

3.2 Slot-Aware Compression 4

3.3 Information-Theoretic Analysis 5

3.4 Slot-Scoped Retrieval 5

4 Experiments **6**

4.1 Experimental Setup 6

4.2 Results and Analysis 7

5 Conclusions and Future Work **9**

A Proof of Proposition 1 **16**

B Implementation Details **17**

B.1 Slot Extraction Prompts 17

B.2 Compression Strategies by Slot 17

B.3 Retrieval Pipeline Configuration 18

B.4 Storage and Indexing 18

B.5 Computational Resources 19

B.6 Experimental Controls 19

C Benchmark Adaptation Details **19**

C.1 LoCoMo Protocol 20

C.2 Multi-Session WebArena Construction 20

C.3 SWE-Bench Pro Protocol 20

C.4 Terminal-Bench 2.0 Protocol 21

C.5 Memory Regime Alignment 21

D Additional Results and Analyses **21**

D.1 LoCoMo Results by Question Category 21

D.2 Budget Fairness and Statistical Significance 22

D.3 Summarization-Free Context Management 22

D.4 Recent Context Managers 23

D.5 Empirical Rate-Distortion Analysis 24

D.6 Mutual Information Decay of the Action Chain 25

D.7 Retrieval Analysis 26
D.8 Computational Overhead 26
D.9 Dependence on Execution Metadata 27
D.10 Robustness to the Backbone Model 27
D.11 Hyperparameter Sensitivity 28
D.12 Per-Benchmark Slot Importance 28
D.13 Failure Case Analysis 29
D.14 Qualitative Example 29

E Limitations **29**

A Proof of Proposition 1

We state the setting, restate Proposition 1, and give the complete proof.

Setting. Each slot variable X_s, $s \in \{o, r, a, u\}$, takes values in a finite alphabet (token sequences of bounded length over a finite vocabulary), so all entropies are finite. A compressor is a test channel $p(\hat{x} \mid x)$ with $\hat{x} = (\hat{x}_o, \hat{x}_r, \hat{x}_a, \hat{x}_u)$, where $\hat{x}_s$ is the reconstruction of slot s, and its per-slot distortions are $D_s = \mathbb{E}[d_s(X_s, \hat{X}_s)]$. The rate-distortion function of slot s is

$$R_s(D_s) = \min_{p(\hat{x}_s|x_s):\, \mathbb{E}[d_s(X_s,\hat{X}_s)] \le D_s} I(X_s; \hat{X}_s), \tag{7}$$

which is convex and non-increasing in D_s. The slot-aware rate $R_{\text{slot}}(D)$ is defined in Eq. (3). A task-agnostic monolithic compressor can realize a set of joint channels $\mathcal{P}_{\text{mono}}$ (e.g., summaries of different lengths produced under its own objective), and

$$R_{\text{mono}}(D) = \min_{p \in \mathcal{P}_{\text{mono}}:\, \mathbb{E}[d_{\text{task}}(X,\hat{X})] \le D} I(X; \hat{X}). \tag{8}$$

We write $p^\star$ for the minimizing channel and $(D_s^{\text{mono}})_s$ for its per-slot distortions, which satisfy $\sum_s w_s D_s^{\text{mono}} \le D$. Finally, $\mathrm{TC}(X) = \sum_s H(X_s) - H(X)$ is the total correlation among slots.

Proposition 1. *Define the allocation gain* $G_{\text{alloc}}(w) = \sum_s R_s(D_s^{\text{mono}}) - R_{\text{slot}}(D)$. *Then* $G_{\text{alloc}}(w) \ge 0$ *and*

$$R_{\text{mono}}(D) - R_{\text{slot}}(D) \ge G_{\text{alloc}}(w) - \mathrm{TC}(X). \tag{9}$$

We first prove a slot-wise lower bound on the mutual information of any joint code.

Lemma 1. For any joint channel $p(\hat{x} \mid x)$, $I(X; \hat{X}) \ge \sum_s I(X_s; \hat{X}_s) - \mathrm{TC}(X)$.

Proof of Lemma 1. By the chain rule and because conditioning cannot increase entropy,

$$H(X \mid \hat{X}) \le \sum_s H(X_s \mid \hat{X}) \le \sum_s H(X_s \mid \hat{X}_s). \tag{10}$$

Therefore

$$\begin{aligned} I(X; \hat{X}) &= H(X) - H(X \mid \hat{X}) \ge H(X) - \sum_s H(X_s \mid \hat{X}_s) \\ &= \sum_s I(X_s; \hat{X}_s) - \Big(\sum_s H(X_s) - H(X) \Big), \end{aligned} \tag{11}$$

and the last term in parentheses is $\mathrm{TC}(X)$. □

Proof of Proposition 1. Step 1: slot-aware optimum. Problem (3) minimizes a sum of convex functions under a linear constraint, so it is convex. Where R_s is differentiable and the optimal D_s^* is interior, the Karush–Kuhn–Tucker conditions give $R_s'(D_s^*) = -\lambda w_s$ for a common multiplier $\lambda \ge 0$. Slots with a larger task weight or a steeper rate-distortion curve therefore receive more rate.

Step 2: non-negative allocation gain. The monolithic profile satisfies $\sum_s w_s D_s^{\text{mono}} \le D$ and is therefore feasible for Problem (3). Hence $\sum_s R_s(D_s^{\text{mono}}) \ge R_{\text{slot}}(D)$, i.e., $G_{\text{alloc}}(w) \ge 0$.

Step 3: monolithic lower bound. Applying Lemma 1 to $p^\star$ gives $R_{\text{mono}}(D) \ge \sum_s I(X_s; \hat{X}_s) - \mathrm{TC}(X)$. Under $p^\star$, the induced channel from X_s to $\hat{X}_s$ achieves distortion D_s^{mono} on slot s, so $I(X_s; \hat{X}_s) \ge R_s(D_s^{\text{mono}})$ by the definition of R_s. Combining the two inequalities,

$$R_{\text{mono}}(D) \ge \sum_s R_s(D_s^{\text{mono}}) - \mathrm{TC}(X) = R_{\text{slot}}(D) + G_{\text{alloc}}(w) - \mathrm{TC}(X), \tag{12}$$

which is the claimed bound. □

Consequences. Three corollaries follow directly. (i) *Independent slots.* If $\mathrm{TC}(X) = 0$, then $R_{\text{mono}}(D) \ge R_{\text{slot}}(D) + G_{\text{alloc}}(w)$, so slot-aware compression is never worse and is strictly better whenever the monolithic profile is not optimal for Problem (3). A task-agnostic compressor attains this optimum only by coincidence, because its profile is set by its own objective rather than by w.

(ii) *Dependent slots.* Slot-aware compression is guaranteed to need a strictly lower rate whenever $G_{\text{alloc}}(w) > \text{TC}(X)$. (iii) *Cost of separation.* Let $R^*_{\text{joint}}(D)$ be the minimum rate of any joint code, including task-aware ones, that satisfies $\mathbb{E}[d_{\text{task}}] \leq D$. Applying Lemma 1 to its optimal channel and using feasibility of its profile gives $R^*_{\text{joint}}(D) \geq R_{\text{slot}}(D) - \text{TC}(X)$. Separate coding thus costs at most $\text{TC}(X)$ relative to the best joint code, and nothing when slots are independent. Redundancy matters in the other direction as well: if two slots were identical copies, $\text{TC}(X)$ would equal the entropy of the copy, which a joint code could exploit. STM keeps inter-slot redundancy low by storing cross-slot relations as explicit links (invariant 2) rather than duplicating content across slots.

Remark on finite-length coding. The analysis above operates in the asymptotic regime. In practice, compression is performed by finite-context language models that do not achieve the information-theoretic lower bound. We therefore estimate both terms of the bound on real agent traces and measure the compression-quality trade-off of monolithic and slot-aware compressors in Appendix D.5.

B Implementation Details

B.1 Slot Extraction Prompts

The Observation Pipeline uses a single extraction prompt applied after each turn completes. The prompt is reproduced below in shortened form.

```
You are extracting a structured record from a completed
agent turn.  Given the raw turn transcript, tool calls, and
execution context, produce a JSON object with exactly four
keys:  "observation", "reasoning", "action", "outcome".
- observation:  user message, tool results from previous
dispatches, environment signals, IDs of recalled evidence
with one-line reasons.
- reasoning:  intent resolution (1 sentence), plan state
assessment, decision trace (list of decisions with
one-sentence rationale each), binding constraints.
- action:  response summary, tool calls [{name,
input_summary, output_summary}], work_mutations [{type,
target, new_state}], artifact changes.
- outcome:  success_signal (true/false/partial),
user_feedback (if any), error_events, open_threads.
Preserve every tool call and every work mutation.  Do not
omit actions even if they seem minor.  Keep the action
section at least 40 tokens.
```

The memory controller applies the emitted `work_mutations` to the work-item table (Appendix B.4), following the matching rule in §3.1. We use GPT-5-mini for extraction (the same model serving as the agent backbone) to avoid introducing a separate model dependency. Extraction cost averages $0.0002 per turn at GPT-5-mini pricing. STM uses seven prompts in total: slot extraction, the four Level-1 slot compressors, the Level-2 merge prompt, and the retrieval request builder. All of them are released verbatim with the code.

B.2 Compression Strategies by Slot

Each slot-specific compressor at Level 1 uses a dedicated prompt. We summarize the key instructions:

Observation compressor: "Extract user intent and key entities. Retain all evidence IDs with their recall reasons. Drop verbose tool outputs except for numerical results and direct quotes."

Reasoning compressor: "Retain each decision and its one-sentence rationale. Drop deliberation alternatives and intermediate evaluation unless they were ultimately selected."

Action compressor: "Preserve every tool call with name, one-line input summary, one-line output summary. Preserve every work mutation with type, target, and new state. Minimum output length: 40 tokens."

Outcome compressor: "Retain success signal, user feedback verbatim if present, and any open threads. Typical output unchanged from input."

Level 2 merging groups Level-1 turns that share a work-item ID within a window of $W = 10$ turns and applies a dedicated merge prompt that produces the net observation, net decision chain, net state change, and final outcome. In intra-session settings, merging is triggered once more than $M = 20$ Level-1 turns have accumulated; in inter-session settings, it is triggered by session end or a work-item state change (Appendix C.5). Each promotion writes the new chunk, links it to its constituents through the `replacement_chain` field, and removes the constituents from the live index. Their text stays on disk while the storage budget allows, which enables one-hop expansion at retrieval time.

B.3 Retrieval Pipeline Configuration

Retrieval requests. In the agentic benchmarks, the memory controller issues one retrieval request before each step using the following 118-token template:

```
Given the task instruction, the active plan step, and the last tool result,
write ONE retrieval query for the agent's memory.
Output JSON: {"query": str, "target_slots": [subset of o,r,a,u],
 "k": int <= 20, "scope": "recent" | "historical"}
Rules: name concrete entities (file paths, URLs, identifiers, error strings)
rather than paraphrasing the task. Choose target_slots by question type:
what was seen -> o ; why -> r ; what was done -> a ; what happened -> u.
Set scope=recent if the query concerns the active plan step, else historical.
Inputs: [task instruction] [active plan step, taken from the previous turn's
reasoning slot] [last tool result, truncated to 500 tokens] [active work-item
title]
```

The `scope` field selects the level prior. If the output is not valid JSON (0.7% of calls), STM falls back to a query formed by the task instruction and the last 200 tokens of the previous observation slot, with all four slots targeted. On LoCoMo, the query is the question verbatim, all four slots are targeted, and $k = 20$.

Indexing and scoring. The pipeline combines BM25 and dense retrieval. BM25 parameters are $k_1 = 1.2$ and $b = 0.75$. Dense retrieval uses OpenAI text-embedding-3-small (1536 dimensions) with cosine similarity and an HNSW index (M=16, efConstruction=200, efSearch=100). Top-100 candidates per index are merged and reranked with

$$\begin{aligned}\text{Score}(c) = {} & 0.315 \cdot \text{Sim}(q, c) + 0.270 \cdot \text{WorkRel}(c, g) + 0.135 \cdot \text{Recency}(c) \\ & + 0.180 \cdot \text{SlotBonus}(c, \mathcal{S}_{\text{target}}) + 0.100 \cdot \text{LevelPrior}\big(\ell(c), \text{scope}(q)\big),\end{aligned} \tag{13}$$

where $\text{WorkRel}(c, g) = 1$ if chunk c is linked to work item g, 0.5 if it is linked to a related item by lineage, and 0 otherwise. If no work item can be induced, WorkRel is 0 for all candidates and the remaining weights are renormalized. $\text{Recency}(c) = \exp(-\Delta t/\tau)$ with $\tau = 14$ days for inter-session benchmarks and $\tau = 50$ steps for intra-session benchmarks; records with unresolved open threads or inside a correction chain are exempt from decay. $\text{SlotBonus}(c, \mathcal{S}_{\text{target}}) = 1$ if c's slot is in $\mathcal{S}_{\text{target}}$, else 0. LevelPrior takes the values given in §3.4.

Assembly. If a chunk and its merged ancestor are both live during a promotion window, the more detailed one that fits the budget is kept and the other is dropped, so no content is counted twice. One-hop expansion applies only to action-slot records at Levels 2 and 3 whose constituents remain on disk, and it uses only the budget left after top-k selection. Each chunk is capped at 256 tokens, and the retrieved memory is capped at 8,192 tokens per step. The final chunks are placed in the prompt in slot order O, R, A, U with slot boundaries preserved. Appendix D.7 analyzes each of these components.

B.4 Storage and Indexing

Structured turn records are stored in SQLite with the following schema:

- `evidence_records(evidence_id, session_id, workspace_id, work_item_ids, kind, content_json, created_at, lifecycle_state, replacement_chain, embedding_version)`
- `evidence_chunks(chunk_id, evidence_id, slot, sub_slot, content, summary, token_count, compression_level, created_at)`

- `evidence_links(source_id, target_id, link_type)` for cross-references (super-session, correction, work linkage, and decision-to-mutation links)
- `work_items(item_id, title, state, parent_id, created_at, closed_at)` for work-item tracking
- FTS5 virtual table over `evidence_chunks.content` for lexical retrieval
- External HNSW index file for dense retrieval, rebuildable from canonical rows

B.5 Computational Resources

All experiments were run on a single workstation with 1x NVIDIA A100 (80GB) for local embedding computation and OpenAI API calls for the main agent and compression. Total API cost for the reported experiments, including the additional baselines and analyses, was approximately $5,300. Per-condition wall-clock time averaged 18 hours for the full evaluation suite across all four benchmarks.

B.6 Experimental Controls

Memory methods are sensitive to budgets and implementation details, so all methods share the controls in Table 7. We distinguish three budgets: the context window cap, the per-step budget for retrieved memory, and the on-disk storage budget. Retrieval budgets are equalized in tokens as well as in chunk count.

Table 7: Control variables shared by all methods.

Control	**Value**
Backbone and decoding	GPT-5-mini, temperature 0.7, top-p 1.0, $\leq$ 4,096 output tokens per step
Embedding model	text-embedding-3-small (1536 dimensions)
Context window cap	400K tokens
Retrieved-memory budget	8,192 tokens per step
Storage budget	$\leq 1.5\times$ the raw trace token count
Retrieval budget	$k = 20$ chunks, ≤ 256 tokens per chunk
Step caps	80 (SWE-Bench Pro), 60 (Terminal-Bench 2.0), 30 per sub-goal (WebArena)
Seeds	5, with the same seed set for every method
Evaluation sets	LoCoMo 1,986 QA; WebArena 200 chains; SWE-Bench Pro 731 tasks; Terminal-Bench 2.0 89 tasks
Hyperparameter tuning	24-configuration grid over each method's published knobs
Tuning data	held-out tuning split per benchmark, disjoint from evaluation
Compression analyses	500 held-out SWE-Bench Pro turns
Significance testing	paired bootstrap, 10,000 resamples, Holm correction

C Benchmark Adaptation Details

Our four benchmarks span two memory regimes. In the *inter-session* regime (LoCoMo and multi-session WebArena), the context is reset between sessions, so information must be carried across sessions by memory. In the *intra-session* regime (SWE-Bench Pro and Terminal-Bench 2.0), each task runs as one long session, and memory must keep a growing history within the context budget. Table 8 lists how sessions, turns, and work items are instantiated on each benchmark. Work items are induced by the Observation Pipeline (§3.1) and never use dataset annotations. We detail the protocol for each benchmark below.

Table 8: Instantiation of sessions, turns, and work items on each benchmark. The last column gives the mean number of work items per episode.

Benchmark	**Session**	**Turn**	**Root work item**	**Child work items**	**Items**
LoCoMo	dataset session boundary	one dialogue exchange	recurring dialogue thread	none	7.2
WebArena (multi-session)	one sub-goal episode	one action with its observation	task chain	one per sub-goal	3.1
SWE-Bench Pro	whole task (30–80 steps)	one tool call with its result	GitHub issue	one per file under modification	4.6
Terminal-Bench 2.0	whole task	one command with its output	task	one per touched subsystem	2.8

C.1 LoCoMo Protocol

We use LoCoMo (Maharana et al., 2024) as-is with no modification. Its dialogues each span about 300 turns and 9K tokens on average over up to 35 sessions, and we use the public release of 10 conversations with 1,986 QA queries. The agent receives the full dialogue history through its memory system and then answers the queries, using each question verbatim as the retrieval query. Each dataset session is one STM session, and each dialogue exchange is one turn. For STM, each dialogue turn is decomposed into slots at ingestion time. For baselines, each method ingests dialogue turns using its native storage mechanism. We compute token-level F1 following the standard protocol. Adversarial questions are excluded from the aggregate F1, following convention, and are reported separately in Appendix D.1. Because the agent's own reasoning is also recorded, a full LoCoMo trajectory is longer than the dialogue itself (Appendix D.3).

C.2 Multi-Session WebArena Construction

WebArena (Zhou et al., 2024) provides 812 single-session web tasks. We construct a multi-session variant by selecting 200 complex tasks and decomposing each into 2–4 dependent sub-goals that are assigned to separate sessions. The decomposition follows four rules:

1. **State-dependent chaining.** Each sub-goal after the first references an entity, state, or information artifact produced by a prior sub-goal. For example, a task "find the cheapest laptop with 16GB RAM and add it to cart" becomes: session 1 = "search for laptops with 16GB RAM and identify the cheapest option"; session 2 = "navigate to the product page for [the laptop identified in session 1] and add it to cart."
2. **Context reset.** Between sessions, the browser state is fully reset (cleared cookies, fresh page). The agent must recall prior findings from memory alone, not from browser state.
3. **Implicit reference.** Sub-goal descriptions in later sessions use implicit references ("the laptop you found," "the forum post you identified") rather than explicit entity names, so answering requires genuine memory retrieval. These references never contain the answer.
4. **No gold information in the input.** The decomposition is produced offline from the task instruction alone, never from a gold trajectory or gold end state. The auto-grader is invoked only after an episode terminates, and all methods receive identical inputs.

We construct chains across all four WebArena domains: shopping (68 chains), forums (52), content management (44), and maps (36). A chain is scored as successful only if the final sub-goal achieves the correct end state as judged by WebArena's native auto-grader. Partial progress is the fraction of sub-goals completed before the first failure. Each session is capped at 30 steps. WebArena exposes no privileged environment state, so work mutations are derived from the emitted action string, the URL change, and a DOM-diff heuristic.

Single-session WebArena. We also evaluate on the original single-session WebArena with all 812 tasks, where no decomposition is applied. Table 9 shows that STM improves over Mem0 and AWM by 7.8 and 4.2 points. The gains are smaller than in the multi-session variant, which requires cross-session recall, but consistent in direction.

Table 9: Success rate on the original single-session WebArena (812 tasks).

Method	SR
Mem0	25.8
AWM	29.4
STM (Ours)	**33.6**

C.3 SWE-Bench Pro Protocol

SWE-Bench Pro (Deng et al., 2025) provides 731 public long-horizon software engineering tasks, each requiring multi-step codebase exploration, editing, and testing within a single session. We use this public subset without modification to the task structure.

For memory evaluation, the key challenge is *intra-session* context management: tasks typically require 30–80 reasoning steps (capped at 80), during which earlier file explorations, grep results, and edit decisions are progressively compressed or evicted from the context window. Each tool call with its result is one turn, the GitHub issue is the root work item, and each file under modification is a child work item. We instrument each method's memory system to manage this growing history:

- **STM:** Each tool call (file read, grep, edit, test run) produces a structured turn record. As the turn count grows, older turns are compressed from Level 0 to Level 1, with action slots (file edits, test results) protected by the τ_a floor. The action-chain retrieval mechanism assembles the full edit history for the files under active modification.
- **Memory baselines:** Each baseline uses its native compression or eviction strategy as the session grows. Mem0 consolidates facts into its graph; A-Mem links related memories via its Zettelkasten structure; AWM extracts workflow patterns from repeated edit-test cycles; REMem indexes tool outputs as episodic events; HippoRAG 2 applies continual learning formation; ReasoningBank stores reasoning chains for retrieval.
- **Context managers:** MEM1 overwrites a compact internal state at every step, reimplemented in context for GPT-5-mini; AgentFold folds the step history at two scales; GAM stores raw pages and compresses them at query time with its researcher agent.

All methods share the backbone, embedding model, and budgets in Appendix B.6. The auto-grader evaluates whether the agent's final patch resolves the target GitHub issue by running the repository's test suite. The compression analyses (Table 3 and Appendix D.5) use 500 held-out SWE-Bench Pro turns.

C.4 Terminal-Bench 2.0 Protocol

Terminal-Bench 2.0 (Merrill et al., 2026) provides 89 verified hard CLI tasks spanning software engineering, machine learning, security, data science, and other domains. Like SWE-Bench Pro, these are single-session tasks (capped at 60 steps), but with a distinctive memory challenge: command outputs, often hundreds of lines of logs, directory listings, or configuration dumps, must be retained and referenced by later commands. Each command with its output is one turn, the task is the root work item, and each touched subsystem is a child work item.

For the scaling analysis in Figure 2, we bin tasks by the number of reasoning steps required, as annotated in the benchmark's metadata: 5–10, 11–20, 21–30, and 30+ steps. The auto-grader checks the final system state against the gold specification.

For STM, command outputs are stored in `observation.tool_results` of the subsequent turn, and the action slot preserves the exact command issued. This separation is particularly useful for CLI tasks: when a later step needs an earlier command's output, slot-scoped retrieval on $\{o\}$ returns the raw output without mixing in the agent's interpretation of that output.

C.5 Memory Regime Alignment

Table 10 summarizes how STM's compression triggers differ across the two memory regimes. The grouping window $W = 10$ bounds the span of turns merged into one Level-2 episode in both regimes, while $M = 20$ sets how many Level-1 turns may accumulate in a long single session before merging starts.

The slot decomposition, compression strategies, and retrieval mechanisms are identical across regimes, and only the trigger conditions differ. The same framework therefore handles both inter-session and intra-session memory without regime-specific engineering.

D Additional Results and Analyses

D.1 LoCoMo Results by Question Category

Table 11 breaks down LoCoMo performance by question category. The largest improvements over REMem appear on multi-hop ($+7.5$) and temporal ($+6.9$) questions. Multi-hop questions require

Table 10: Compression trigger conditions across memory regimes.

Trigger	Inter-session (LoCoMo, WebArena)	Intra-session (SWE-Bench Pro, Terminal-Bench 2.0)
Level 0 $\rightarrow$ 1	Session boundary, or raw buffer exceeds N turns	Raw buffer exceeds N turns in the active session
Level 1 $\rightarrow$ 2	Session end or work-item state change, grouping turns of a shared work item within W	More than M Level-1 turns accumulated, grouping turns of a shared work item within W
Level 2 $\rightarrow$ 3	Scheduled maintenance or work-item closure	Task completion or work-item closure
Context eviction	Full context reset at session start	Oldest chunks evicted when budget exceeded

synthesizing facts across sessions, and slot-scoped retrieval can target the slot most likely to contain the needed fact. Temporal questions require event ordering, which benefits from the structured turn index and timestamps preserved in STM records. REMem's hybrid memory graph is strong at episodic recollection but does not differentiate internal turn structure. On single-hop questions, both REMem and STM clearly outperform Mem0, suggesting that structured memory organizations, whether graph-based or slot-based, give broad recall advantages. STM also leads on adversarial questions, which are excluded from the aggregate F1.

Table 11: LoCoMo QA F1 by question category. STM shows the largest gains on multi-hop and temporal questions, which require cross-session reasoning.

Method	Single-hop	Multi-hop	Temporal	Open	Adversarial
Mem0	$55.8_{\pm 1.3}$	$37.4_{\pm 1.8}$	$33.2_{\pm 2.1}$	$50.1_{\pm 1.5}$	$42.6_{\pm 1.7}$
REMem	$58.4_{\pm 1.1}$	$44.6_{\pm 1.5}$	$42.8_{\pm 1.7}$	$53.9_{\pm 1.2}$	$46.3_{\pm 1.4}$
STM (Ours)	$\mathbf{62.3_{\pm 0.9}}$	$\mathbf{52.1_{\pm 1.3}}$	$\mathbf{49.7_{\pm 1.5}}$	$\mathbf{58.6_{\pm 1.1}}$	$\mathbf{51.4_{\pm 1.2}}$

D.2 Budget Fairness and Statistical Significance

All methods share the retrieval budget in tokens as well as in chunk count, and STM already retrieves fewer tokens per step than the baselines (Appendix D.8). To test whether retrieval volume or chunk granularity could explain the gap, we run two controls on SWE-Bench Pro for AWM and REMem (Table 12). Under a strictly token-matched condition with 4,096 retrieved tokens per step, AWM reaches 37.9% and REMem 31.9%, against 44.0% for STM. Giving both baselines the same fine-grained 128-token chunking raises them only to 38.3% and 32.6%. Neither control closes more than a small fraction of the gap.

Table 12: SWE-Bench Pro success rate under budget and granularity controls.

Method	Default budget	Token-matched (4,096)	Token-matched + 128-token chunks
AWM	37.5	37.9	38.3
REMem	31.4	31.9	32.6
STM (Ours)	**44.2**	**44.0**	–

For significance, we compare STM with the strongest baseline on each benchmark using a paired bootstrap with 10,000 resamples and Holm correction. The resulting p-values are 0.002 on LoCoMo, below 0.001 on WebArena, 0.0007 on SWE-Bench Pro, and below 0.001 on Terminal-Bench 2.0. Effect sizes (Cohen's d) range from 1.9 to 3.4.

D.3 Summarization-Free Context Management

References. Recency truncation keeps the most recent raw history up to the 400K-token cap. Retrieval over raw turns indexes unsummarized turns with the same retriever and budget as STM.

The ACE-style playbook is append-only and grows through delta updates without summarization. Full context passes the whole trajectory and truncates the oldest tokens once it exceeds 400K. For STM + ACE, the playbook is added to the prompt next to STM's retrieved memory. Per step, the history in the prompt amounts to up to 400K tokens for recency truncation, 3.4K for retrieval over raw turns, 38K for the playbook, and 31K to 400K for full context. STM uses 12.7K, which includes its raw window of N turns in addition to at most 8,192 tokens of retrieved memory, and STM + ACE uses 21K.

When does the trajectory fit? Table 13 reports the mean trajectory length, including the agent's reasoning, and the fraction of episodes that fit in the 400K window. Every LoCoMo trajectory fits, whereas only 39% of SWE-Bench Pro trajectories do.

Table 13: Mean trajectory length (including the agent's reasoning) and fraction of episodes that fit in the 400K-token context window.

Benchmark	Mean trajectory tokens	Episodes within 400K
LoCoMo (per dialogue)	31K	100%
WebArena (per chain)	214K	78%
SWE-Bench Pro	486K	39%
Terminal-Bench 2.0	372K	53%

Split analysis. Table 14 splits SWE-Bench Pro by whether the trajectory fits. Where it fits, full context is a strong reference and STM's advantage is modest (+1.8 SR), while STM uses roughly 8% to 30% of the prompt tokens. Where it does not fit, full context degrades to recency truncation and STM's advantage grows to +8.0 SR, which accounts for most of the aggregate gain. On LoCoMo, where every dialogue fits, STM leads full context by 2.7 F1 (56.8 vs. 54.1). The largest per-category deficit of full context is on temporal questions (44.2 vs. 49.7 for STM; cf. Table 11), consistent with the lost-in-the-middle effect. For tasks that fit, STM is therefore mainly a cost-quality improvement; for tasks that do not, it is an accuracy improvement.

Table 14: SWE-Bench Pro success rate split by whether the trajectory fits in the 400K-token window.

Method	**Fits** (39%)	**Does not fit** (61%)	All
Full context	47.8	32.7	38.6
STM (Ours)	**49.6**	**40.7**	**44.2**

Complementarity with playbooks. Summarization-free playbooks and STM target different content. The playbook curates cross-task strategy knowledge (what to do), whereas STM preserves within-task episodic provenance (what was actually done, and why). Combining them adds 2.2 to 2.3 points over STM alone on the agentic benchmarks (Table 2).

D.4 Recent Context Managers

Positioning. Table 15 contrasts STM with episodic and graph memories and with recent context managers along five axes. Existing designs vary the rate (how much to keep) and the schedule (when to compress) over a homogeneous unit of storage. STM instead changes the unit, making a typed sub-turn slot the object that receives its own budget and that retrieval can address. State, action, and outcome decompositions are common in the agent loop, as in ReAct and CoALA, and in offline execution logs. STM uses this decomposition as the unit of memory compression and retrieval.

Methods. MEM1 learns with reinforcement learning to overwrite one compact internal state per turn, which keeps memory nearly constant. The state is unstructured and not addressable, and the trajectory is discarded, so past content cannot be queried again. AgentFold folds history at two scales and chooses the scale at every step. This is a flexible schedule over homogeneous blocks, without typed sub-blocks, per-type floors, or slot-scoped retrieval. GAM keeps a raw page store and defers compression to query time through a researcher agent, so it retains everything and pays at retrieval. STM changes what the store exposes, which reduces the search space of such a researcher rather than replacing it.

Table 15: Design axes of memory systems, recent context managers, and STM.

Axis	Episodic and graph memory (Mem0, A-Mem, REMem)	Context managers (MEM1, AgentFold, GAM)	STM
Unit of storage	turn, event, or fact node	evolving state, folded block, or raw page	typed sub-turn slot
Per-component rate allocation	no	no (uniform per state or fold)	yes, per-slot budgets with an action floor
Retrieval addressability	turn, event, or node	none (in context) or page	slot $\times$ level $\times$ work item
Guarantees under compression	none	none	four invariants, incl. action-chain reconstructability
Cross-slot provenance	implicit in prose	none	explicit decision-to-mutation links

Protocol. The controlled subsets contain 100 QA questions for LoCoMo, 100 chains for WebArena, and 100 tasks for SWE-Bench Pro. Terminal-Bench 2.0 has only 89 tasks, so its full set is used. Each subset is drawn once and shared by all methods, which use the same backbone, budgets, retrieval top-k, and tuning procedure (Appendix B.6). MEM1's released consolidation policy is a 7B model trained with reinforcement learning, so with GPT-5-mini we reimplement its consolidation objective in context. We therefore also compare on a Qwen3-14B backbone, where the released policy runs directly (Table 16). STM leads the strongest manager, AgentFold, by 3.6 points (30.4% vs. 26.8%).

Table 16: SWE-Bench Pro success rate on the 100-task controlled subset with a Qwen3-14B backbone. MEM1 uses its released consolidation policy.

Method	SR
MEM1 (released policy)	24.2
GAM	25.8
AgentFold	26.8
STM (Ours)	**30.4**

Composability. Because these managers change the rate or schedule of compression while STM changes the unit, they can be combined. Applying AgentFold's folding over STM slot records reaches 46.0% SR on the 100-task SWE-Bench Pro subset, compared with 44.6% for STM and 40.2% for AgentFold alone. In the same way, a learned consolidation policy such as MEM1's could operate over STM slots with per-slot budgets, replacing the hand-set action floor.

D.5 Empirical Rate-Distortion Analysis

Slot profiles. Table 17 profiles the four slots on 500 held-out SWE-Bench Pro turns. Compressibility is the compression ratio needed to reach ROUGE-L ≥ 0.85 against the raw slot, so a lower value means a more compressible slot. Task weights are derived from slot ablations. The outcome slot is the least compressible in ratio terms but small in absolute size, so STM keeps it verbatim and its ratio never binds the budget. The action slot is the only slot that is hard to compress, heavily weighted, and sizable at the same time, which is why it needs an explicit fidelity floor. The relevant quantity for budget allocation is thus the combination of task weight, size, and incompressibility, not compressibility alone.

Monolithic versus slot-aware compression. We measure the rate-distortion trade-off of monolithic and slot-aware compression on the same turns. For each turn and each compression ratio $\rho \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, the task-weighted distortion is

$$D_{\text{task}} = 1 - \Big(w_o \cdot \text{ROUGE-L}_o + w_r \cdot \text{ROUGE-L}_r + w_a \cdot \text{ROUGE-L}_a + w_u \cdot \text{ROUGE-L}_u \Big), \quad (14)$$

with the task weights $(w_o, w_r, w_a, w_u) = (0.2, 0.25, 0.4, 0.15)$ from Table 17. Table 18 shows that slot-aware compression achieves lower task-weighted distortion at every compression ratio, with a

Table 17: Slot profiles on 500 held-out SWE-Bench Pro turns. Compressibility is the ratio needed to reach ROUGE-L ≥ 0.85 (lower is more compressible).

Slot	Mean raw tokens	Compressibility	Task weight	Tokens at ROUGE-L ≥ 0.85
Observation	412	0.31	0.20	128
Reasoning	186	0.27	0.25	50
Action	224	0.48	0.40	108
Outcome	38	0.72	0.15	27

relative improvement of 32–39%. The absolute gap is largest under aggressive compression ($\rho \leq 0.3$), where slot-aware allocation can protect the action channel while a monolithic summarizer does not favor it.

Table 18: Task-weighted distortion at matched compression ratios on SWE-Bench Pro turns (lower is better). The absolute gap widens under stronger compression (smaller ρ).

Compression ratio ρ	0.1	0.2	0.3	0.4	0.5
Monolithic D_{task}	$0.68_{\pm 0.04}$	$0.52_{\pm 0.03}$	$0.41_{\pm 0.03}$	$0.33_{\pm 0.02}$	$0.27_{\pm 0.02}$
Slot-aware D_{task}	$0.46_{\pm 0.03}$	$0.33_{\pm 0.02}$	$0.25_{\pm 0.02}$	$0.20_{\pm 0.02}$	$0.17_{\pm 0.01}$
Relative improvement	32.4%	36.5%	39.0%	39.4%	37.0%

Estimating the bound. Table 19 reports both terms of Proposition 1 on the same 500 turns, with mutual information estimated by the InfoNCE estimator of Appendix D.6. The measured redundancy is small, about 11% of the turn entropy, plausibly because the slots differ in surface form: user prose, rationale, tool JSON, and status flags. The allocation gain at compression ratio 0.2 therefore exceeds the total correlation by a wide margin, and the bound guarantees an advantage of at least 1.21 nats per turn. The advantage would vanish only if the total correlation exceeded roughly 1.6 nats per turn, about four times the measured value.

Table 19: Empirical estimates of the terms in Proposition 1 (nats per turn, 500 held-out SWE-Bench Pro turns).

Quantity	Estimate
Total correlation $\mathrm{TC}(X)$ across the four slots	0.42 ± 0.06
Normalized redundancy, $\mathrm{TC}(X)/H(X)$	0.11
Allocation gain $G_{\text{alloc}}(w)$ at compression ratio 0.2	1.63 ± 0.11
Lower bound on the advantage, $G_{\text{alloc}}(w) - \mathrm{TC}(X)$	1.21

Conditional coding. Much of the dependence between slots is relational rather than statistical. The reasoning slot explains the action slot, but this relation is a single pointer from a decision ID to a mutation ID, which STM stores under invariant 2 at a cost of about four tokens. This predicts that coding the action slot conditionally on the reasoning slot should add little. Compressing the action slot conditioned on the already compressed reasoning slot gives 44.8% SR on SWE-Bench Pro, versus 44.2% with independent slot coding, a gain of 0.6 points that is consistent with the small measured total correlation.

D.6 Mutual Information Decay of the Action Chain

We measure how much planning-relevant information survives compression by estimating the mutual information $I(\mathbf{A}^g; G_{\text{state}})$ with the InfoNCE lower bound and 1000 negative samples. On SWE-Bench Pro, $\mathbf{A}^g$ is the sequence of file-edit actions for a given issue and G_{state} is the final patch correctness. Table 20 reports the estimates at each compression level. At Level 2, monolithic compression loses 41.3% of the action-chain information, whereas slot-aware compression loses only 12.7%, a $3.25\times$ difference. The gap persists at Level 3 (68.3% vs. 33.5%). This is consistent with the downstream success gap: on SWE-Bench Pro, the agent needs to recall which files were edited, what the test output was, and which approach was abandoned, which is the information that monolithic compression discards first.

Table 20: Estimated mutual information (nats) between the action chain and the task-completion state at different compression levels on SWE-Bench Pro. Relative loss with respect to Level 0 in parentheses.

Strategy	Level 0 (raw)	Level 1	Level 2	Level 3
Monolithic	1.82	1.41 (−22.5%)	1.07 (−41.3%)	0.58 (−68.3%)
Slot-aware (STM)	1.82	1.73 (−5.0%)	1.59 (−12.7%)	1.21 (−33.5%)

D.7 Retrieval Analysis

Table 21 isolates the retrieval components described in §3.4 and Appendix B.3.

Table 21: Retrieval component analysis. SWE-Bench Pro success rate and LoCoMo F1; – marks settings not evaluated on LoCoMo.

Variant	SWE-Bench Pro SR	LoCoMo F1
Full STM	44.2	56.8
Fallback query at every step	42.0	–
Without work items (WorkRel $= 0$)	41.8	55.7
Uniform level prior	42.3	54.9
Without one-hop expansion	42.4	–
Without recency ($\gamma = 0$)	42.9	–
Worst case under $\pm 50\%$ weight perturbation	42.8	–
Weights learned by logistic regression	44.6	–

Query construction. Replacing the request builder with the fallback query at every step costs 2.2 SR, so STM remains well above all baselines even with a simple query. **Work items.** When no work items are induced, STM loses 2.4 SR on SWE-Bench Pro and 1.1 F1 on LoCoMo; work linkage helps, but the gains do not depend on it. **Level prior.** A uniform prior over levels costs 1.9 SR and 1.9 F1. **One-hop expansion.** Expansion fires on 23% of retrievals, and disabling it costs 1.8 SR. **Scoring weights.** Perturbing each weight by $\pm 50\%$ changes SR by at most 1.4 points. Fitting the weights by logistic regression on retrieval-utility labels from 200 held-out steps gives 44.6%, close to the hand-set 44.2%.

Recency as a validity prior. Because the agent changes its environment, older records are more likely to be invalid. On SWE-Bench Pro, a stored chunk is contradicted by a later chunk with probability 4.1% within 5 steps, 20.8% within 20 steps, and 37.6% within 50 steps. Setting $\gamma = 0$ costs 1.3 SR on SWE-Bench Pro and lowers LoCoMo temporal F1 from 49.7 to 46.9, while adversarial F1 is essentially unchanged (51.4 vs. 51.6). The term thus helps where staleness or event ordering matters and is neutral elsewhere.

Level mixture. Table 22 shows the levels of retrieved chunks at a late point of an episode. On SWE-Bench Pro, recent raw and slot-summarized records dominate, while in a long LoCoMo conversation most retrieved chunks are slot-summarized or episode-merged.

Table 22: Compression levels of retrieved chunks.

Setting	Level 0	Level 1	Level 2	Level 3
SWE-Bench Pro, step 40	34%	41%	19%	6%
LoCoMo, session 30	9%	38%	36%	17%

D.8 Computational Overhead

Table 23 compares the cost of STM with the memory baselines. STM has the lowest per-turn latency (142ms) despite the additional slot decomposition. Its slot extraction is a single lightweight prompt (GPT-5-mini, ∼200 tokens) applied once per completed turn, which avoids the iterative graph construction of REMem (224ms), the workflow induction pass of AWM (198ms), and the recursive

consolidation cycles of Mem0 (156ms). Compression is triggered lazily at session boundaries or when a buffer threshold is reached (Appendix C.5) rather than after every turn. STM also retrieves 31% fewer tokens than AWM and 27% fewer than REMem, because slot-scoped queries restrict retrieval to the relevant slot types and the level prior surfaces compact records instead of verbose raw entries. Storage per turn is comparable across methods. Retrieval over raw turns without memory management is faster (∼75ms) but reaches only 19.4% SR on SWE-Bench Pro, so it is omitted from the table.

Table 23: Computational overhead. STM has the lowest per-turn latency and retrieves the fewest tokens.

Method	Per-Turn Latency (ms)	Retrieval Tokens (avg)	Storage Tokens (per turn)
Mem0	$156_{\pm 28}$	$3{,}680_{\pm 390}$	$840_{\pm 95}$
AWM	$198_{\pm 35}$	$4{,}210_{\pm 480}$	$920_{\pm 115}$
REMem	$224_{\pm 41}$	$3{,}950_{\pm 440}$	$870_{\pm 100}$
STM (Ours)	$\mathbf{142_{\pm 24}}$	$\mathbf{2{,}890_{\pm 290}}$	$\mathbf{820_{\pm 90}}$

D.9 Dependence on Execution Metadata

STM extracts tool calls and state mutations from the execution context when the harness logs them. To measure how much this metadata matters, we compare three regimes (Table 24): full harness instrumentation, as in all other experiments; transcript-only extraction, where slots are built from the agent's own messages and emitted tool-call text without harness state diffs; and noisy metadata, where 30% of mutation records are dropped or mislabeled.

Table 24: Performance under different metadata regimes. The last row gives the strongest baseline on each benchmark for reference.

Regime	WebArena SR	SWE-Bench Pro SR	Terminal-Bench SR	LoCoMo F1
Full instrumentation	41.5	44.2	46.7	56.8
Transcript only	39.8	41.6	43.8	56.8
Noisy mutations (30%)	38.4	40.1	42.5	56.8
Strongest baseline	33.8	37.5	39.1	51.2

Four observations follow. (1) About 68% of action-slot tokens can be recovered from the transcript alone, because tool names and arguments are emitted by the model; the harness mainly contributes the effect of each call. (2) LoCoMo is unaffected, since it contains no tool calls. (3) WebArena already operates without privileged state, with mutations derived from the action string, the URL change, and a DOM-diff heuristic (Appendix C.2), and STM still clearly exceeds the strongest baseline there. (4) MCP-style tool ecosystems return structured call and response JSON, which is the signal STM consumes, so they are close to the instrumented case. In all regimes, STM remains above the strongest baseline on every benchmark.

D.10 Robustness to the Backbone Model

To test whether STM's gains depend on a single backbone, we replicate the main experiments on SWE-Bench Pro with four frontier models from proprietary and open-source families: GPT-5 (Singh et al., 2025) (released August 2025, unified reasoning with adaptive routing), Claude Opus 4.5 (Anthropic, 2025) (released November 2025, optimized for long-horizon agentic workflows), GLM-5 (Zeng et al., 2026) (released February 2026, open-source 744B MoE with 40B active parameters), and Kimi K2 (Team et al., 2025) (released July 2025, open-source 1T MoE with 32B active parameters optimized for agentic tool use). Table 25 reports the results.

Absolute performance varies with backbone capability, with GPT-5 and Claude Opus 4.5 leading, but the improvement from STM stays within $+6.6$ to $+7.1$ points. The advantage also holds for the open-source models GLM-5 and Kimi K2, and with a Qwen3-14B backbone in the comparison with context managers (Appendix D.4). These results indicate that the benefit comes from the storage-side decomposition rather than from a model-specific capability.

Table 25: SWE-Bench Pro task success rate across frontier LLM backbones. STM's advantage over AWM (the strongest baseline) is consistent across proprietary and open-source models.

Method	**GPT-5**	**Claude Opus 4.5**	**GLM-5**	**Kimi K2**
AWM	$41.8_{\pm 1.5}$	$40.6_{\pm 1.7}$	$37.2_{\pm 1.9}$	$35.8_{\pm 2.1}$
STM (Ours)	$48.9_{\pm 1.3}$	$47.6_{\pm 1.4}$	$44.1_{\pm 1.6}$	$42.4_{\pm 1.8}$
Δ (STM $-$ AWM)	$+7.1$	$+7.0$	$+6.9$	$+6.6$

D.11 Hyperparameter Sensitivity

We vary three key hyperparameters and measure their effect on SWE-Bench Pro task success rate (Table 26).

Table 26: Hyperparameter sensitivity on SWE-Bench Pro. STM is robust across a broad range of settings.

Raw retention N	4	6	8 (default)	10	12
Task SR (%)	$40.8_{\pm 1.7}$	$42.9_{\pm 1.5}$	$44.2_{\pm 1.4}$	$44.4_{\pm 1.5}$	$44.3_{\pm 1.6}$
Min action length τ_a	20	30	40 (default)	60	80
Task SR (%)	$38.5_{\pm 1.9}$	$41.6_{\pm 1.6}$	$44.2_{\pm 1.4}$	$44.5_{\pm 1.4}$	$44.1_{\pm 1.5}$
Retrieval k	10	15	20 (default)	30	50
Task SR (%)	$41.1_{\pm 1.7}$	$43.2_{\pm 1.5}$	$44.2_{\pm 1.4}$	$44.5_{\pm 1.5}$	$43.9_{\pm 1.7}$

Performance plateaus for $N \geq 8$, $\tau_a \geq 40$, and $k \geq 20$. The minimum action length τ_a is the most sensitive parameter: lowering it to 20 costs 5.7 points of success rate. This matches the slot profile in Appendix D.5, where the action slot is the one that is hard to compress, heavily weighted, and sizable. Increasing k beyond 30 slightly hurts performance due to retrieval noise. The sensitivity of the retrieval weights is analyzed in Appendix D.7.

D.12 Per-Benchmark Slot Importance

To test whether the relative importance of slots varies across task domains, we run the single-slot ablation on all four benchmarks (Table 27).

Table 27: Performance when retaining only a single slot, relative to the full STM system (= 100%). Higher values indicate a greater standalone contribution of that slot.

Retained slot	**LoCoMo**	**WebArena**	**SWE-Bench Pro**	**Terminal-Bench**
Observation only	71.3%	58.2%	54.6%	62.7%
Reasoning only	64.8%	52.1%	57.3%	49.8%
Action only	62.1%	78.4%	86.9%	84.3%
Outcome only	48.6%	41.7%	43.2%	45.1%

On the three agentic benchmarks, the action slot alone retains 78–87% of full-system performance, which makes it the most task-critical slot. On LoCoMo, the observation slot is more valuable (71.3%) than the action slot (62.1%), since conversational recall depends on what the speakers said rather than on actions. The removal ablations on LoCoMo show the same ordering: removing the observation slot lowers F1 from 56.8 to 47.9 (-8.9), merging observation and outcome gives 53.6 (-3.2), removing outcome gives 54.3 (-2.5), and removing reasoning gives 55.1 (-1.7). Together with the SWE-Bench Pro ablations in Table 4, these results show that no slot is redundant and that the dominant slot is domain-dependent: action for agentic execution and observation for conversational recall. Both benefits come from the same slot decomposition.

D.13 Failure Case Analysis

We manually inspect 50 failure cases on SWE-Bench Pro, i.e., tasks where STM fails to produce a correct patch, and categorize them:

- **Inherent task difficulty (28%):** The issue requires domain knowledge or multi-file reasoning that exceeds the backbone model's capability regardless of memory quality. These failures are shared across all methods.
- **Decomposition extraction errors (22%):** The slot extractor misclassifies a test output as part of the reasoning slot, or merges two distinct tool calls into one action entry, corrupting the action chain.
- **Compression over-reduction (16%):** A file exploration result compressed at Level 1 loses a critical detail (e.g., an import path or error line number) needed 20+ steps later.
- **Retrieval miss (18%):** The needed evidence existed in storage but was not retrieved in the top-k due to vocabulary mismatch between the query and the stored chunk, particularly for variable names and file paths.
- **Genuine reasoning failure (16%):** All relevant information was correctly retrieved, but the agent generated an incorrect edit, typically due to misunderstanding the codebase's control flow.

The three pipeline-related modes (22% + 16% + 18%) account for 56% of all inspected failures, i.e., 78% of the failures not caused by inherent task difficulty. This suggests three concrete improvement directions: (1) fine-tuning a dedicated small model for extraction to reduce misclassification, (2) adaptive compression budgets that allocate more tokens to turns containing file paths and line numbers, and (3) hybrid retrieval with code-specific embeddings for software engineering tasks.

D.14 Qualitative Example

We include one representative trace from SWE-Bench Pro showing how STM reconstructs a long-horizon debugging context that baselines fail to recover.

> **Task:** Resolve a Django ORM issue where `QuerySet.union()` silently drops `order_by()` clauses.
>
> **Turn 8 (step 8 of 47).** The agent greps for `union` in `django/db/models/sql/` and finds the relevant code in `compiler.py` lines 420–435. STM records this in `observation.tool_results = [{"grep", "compiler.py:420-435: SQL generation for combined queries"}]` and `reasoning.decision_trace = [{"The union SQL compiler strips ORDER BY from subqueries per SQL standard; the bug is that it also strips the outer ORDER BY"}]`.
>
> **Turn 31 (step 31 of 47).** After exploring several failed approaches, the agent needs to revisit the original finding. A slot-scoped query on $\{o\}$ with "compiler.py union ORDER BY" retrieves the exact grep result from turn 8. Action-chain retrieval for the task returns the sequence of 6 file edits attempted so far, showing which approaches were tried and reverted.
>
> **AWM baseline:** AWM's workflow memory captured a general "grep → read → edit → test" pattern but did not retain the specific file location or the reasoning about *why* the outer ORDER BY was stripped. At turn 31, the agent re-greps for the same information (wasting 3 steps) and misses the connection to the earlier failed approaches, re-attempting a strategy that was already abandoned at turn 19.

The two AWM errors are instances of the redundant tool calls and redundant strategies measured in Table 5. The example illustrates how slot-aware storage preserves both observational detail and decision provenance that workflow-level and monolithic memory systems discard.

E Limitations

We note several limitations of the current work.

Extraction quality ceiling. The slot decomposition is only as reliable as the extraction prompt. On turns with unusual structure (nested tool calls, multi-party dialogue, interrupted reasoning), the

extractor's error rate rises to roughly 15% based on manual inspection, and extraction errors account for 22% of the inspected failures (Appendix D.13). Improving extraction robustness by fine-tuning a dedicated small model is a natural next step.

Fixed slot taxonomy. The four-slot taxonomy (O, R, A, U) is defined by hand and motivated by the structure of typical LLM agent turns. The dominant slot is already domain-dependent (Appendix D.12), and multi-agent interactions, embodied agents with continuous perception, or agents with rich internal deliberation may benefit from richer slot schemas. Learning the taxonomy or the per-slot budgets from downstream performance is a promising extension.

Compression model dependence. The per-slot compressors are LLM-based and inherit the biases and failure modes of the underlying model. The action-fidelity floor τ_a mitigates but does not eliminate the tendency to drop tool-call details. Proposition 1 is an asymptotic statement, and our estimates of its terms rely on a neural mutual-information estimator applied to one benchmark (Appendix D.5).

Access to an action log. STM needs an accessible record of the agent's actions. It works with transcript-only extraction and tolerates noisy mutation records (Appendix D.9), but a pure GUI agent without any accessible action log is outside the current scope.

Histories that fit in context. When the full trajectory fits in the context window, keeping the full context is a strong reference, and STM's benefit is mainly a lower prompt cost with a modest accuracy gain (Appendix D.3). The accuracy gains are concentrated on histories that exceed the window.

Benchmark coverage. While our four benchmarks span two memory regimes and diverse task domains, they do not cover embodied agents, multi-agent collaboration, scientific research workflows, or inputs with images and other modalities. The multi-session WebArena variant is our own construction. We encourage future work to extend evaluation to these settings.

Computational cost. STM adds extraction latency and slot-level indexing overhead, although its total per-turn latency ($\sim$142ms) is the lowest among the compared memory systems (Appendix D.8). For deployments where per-turn latency is critical, asynchronous extraction and delta-only indexing are promising directions.