DUNE: KEEPING DERIVED MEMORIES CONSISTENT WITH CHANGING FACTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Research on long-term memory for agents increasingly studies how to replace an obsolete fact with its updated value. However, this overlooks a deeper problem, that is, updating a single stored fact can invalidate downstream conclusions, answers, or plans derived from it, even when their revised values are never explicitly stated. Drawing an analogy to materialized views and incremental view maintenance in databases, we define such downstream conclusions as *derived memories*. Derived memory maintenance requires revising the source evidence and any downstream derived memories whose validity or value depends on it. Based on this, we introduce DRM-bench, a collection of 3,200 two-session scenarios for evaluating derived memory maintenance across semantic, episodic, and procedural memory. To address this problem, we propose DUNE (Derived memory Updating through Networked Evidence), which maps evidence to nodes and dependencies to edges, then propagates source updates over the resulting graph. DUNE achieves the best source update (84.8%), and derived update (73.6%) among the tested methods on DRM-bench, and improves averaged F1, LLM Judge, SourceUpdate accuracy, and DerivedUpdate accuracy by 9.6, 14.8, 22.2, and 22.5 points, respectively.

1 INTRODUCTION

Long-term memory allows agents to carry user facts, past events, and task knowledge across sessions, which has motivated previous work on memory systems and benchmarks for long-term conversational and system memory (Park et al., 2023; Wang et al., 2023a; Maharana et al., 2024; Tan et al., 2025a). As the external information changes, researchers have also begun to study how to update stored evidence in memory. Memory systems such as A-MEM and MemoryBank allow new evidence to reorganize or revise earlier evidence (Zhong et al., 2024; Xu et al., 2025; Sun et al., 2026). Despite this progress, memory maintenance still focuses on the individual evidence being revised or removed.

However, stored evidence may already have produced an answer that was written back to memory. Such downstream conclusions can remain stale after the source has been corrected. Consider an assistant that stores a meeting at 16:00 (E1) and a 45-minute journey (E2), as in Figure 1. In Session 1, LLM answers that the user should leave at 15:15, and the memory module writes this answer back as E3. In Session 2, a new evidence U of the user moves the meeting 30 minutes earlier is added. In (a), an LLM without an memory module can recompute the answer from E1 and E2 and return 14:45. A memory-augmented agent may instead retrieve E3 because it directly matches the question, thereby repeating the stale answer 15:15 in (b).

To resolve the problem, we define stored evidence whose validity or value depends on other evidence as a *derived memory*, and a direct observation, event, or state serves as source evidence. By analogy with materialized-view maintenance in databases (Gupta & Mumick, 1995), we call the task *derived memory maintenance* (DMM). Unlike replacing an isolated piece of evidence, DMM requires tracing and revising the downstream consequences of an update, which presents three main challenges. First, existing benchmarks cover long conversations and direct knowledge updates (Maharana et al., 2024; Wu et al., 2024; Wei et al., 2025; Xu et al., 2026), but none is designed specifically to test whether a system can identify and repair derived memories affected by a source update. Second, dependencies among stored statements are often implicit and cannot be reliably recovered from surface similarity alone. Third, the appropriate scope of each update is difficult to determine. A system must decide

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.

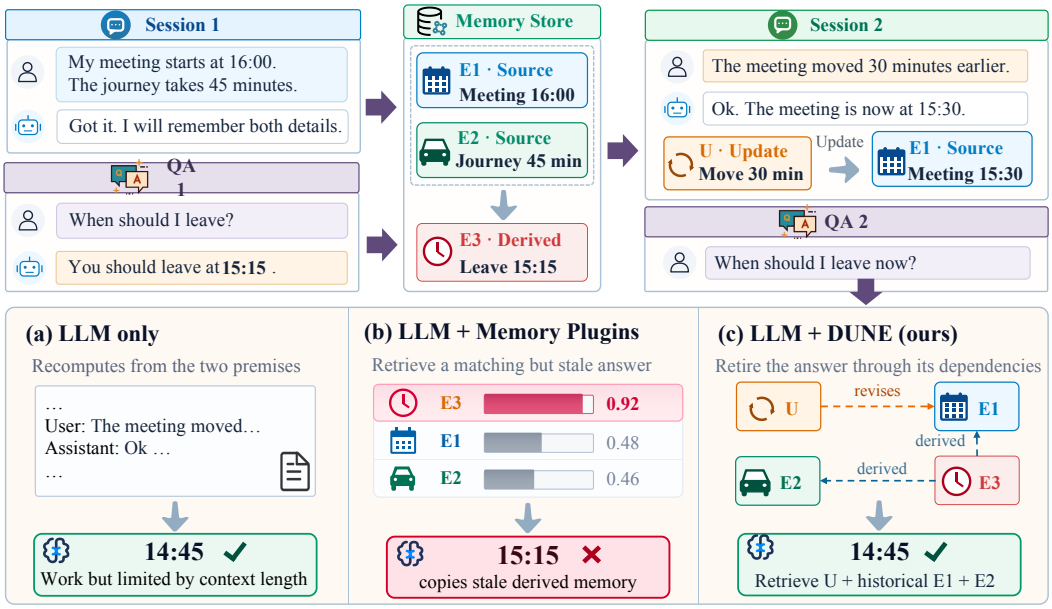

Figure 1: Running example. Session 1 produces a written-back answer E3 from E1 and E2. Session 2 supplies an update U that supersedes the old E1 but does not directly revise E3.

how broadly and how deeply its effects should propagate, while avoiding both missed downstream consequences and unnecessary changes to unrelated evidence.

We introduce DRM-bench to provide an evaluation, which contains 3,200 two-session scenarios across semantic, episodic, and procedural memory. Session 1 establishes source evidence and a dependent conclusion, then asks the model about both. The model's actual answers are then written back to the memory. Session 2 updates some of the information, and the model will be asked to answer both questions again. Moreover, we introduce DUNE (Derived memory Updating through Networked Evidence) to identify and retire stale derived memories. During the first interaction round, DUNE maps the available evidence and its dependencies to nodes and edges in a graph. The graph then grows incrementally after each round. Every node carries two values, where *validity* measures how strongly the available evidence supports its content, and *currentness* indicates whether that content still applies to the present state. At each inference checkpoint, DUNE propagates and recomputes both values across the full graph by minimizing a weighted hinge-loss objective over its nodes and edges. As in Figure 1 (c), DUNE records that E3 depends on E1 and E2. When E1 is updated, E3 loses currentness and the model recomputes from the active evidence. Across the three memory types, DUNE obtains the highest F1 (52.0%), source update (84.8%), and derived update (73.6%) among the evaluated methods in average, while improving averaged F1, Judge accuracy, SourceUpdate accuracy, and DerivedUpdate accuracy by 9.6, 14.8, 22.2, and 22.5 points, respectively.

Our contributions are: 1) We define *derived memory* and formulate DMM as the problem of revising changed source evidence together with the conclusions that depend on it. 2) We introduce DRM-bench, with 3,200 scenarios across semantic, episodic, and procedural memory. 3) We propose DUNE, which maps evidence to nodes and dependencies to edges, then propagates source updates while distinguishing historical validity from present currentness. It explicitly invalidates stale derived memories and enables answer-time recomputation from active evidence.

2 Related Work

2.1 Memory Updating in Agents

Recent agent-memory systems can be clearly distinguished by what they update. Evidence-level methods revise individual pieces of stored evidence, such as user information in MemoryBank and

Mem0 or preferences in PAMU(Zhong et al., 2024; Chhikara et al., 2025; Sun et al., 2026). Structure-level methods change how stored evidence is organized. For instance, A-MEM and THEANINE construct links among events, while MemoryOS, LightMem, GAM, and EverMemOS consolidate experience into graphs(Xu et al., 2025; Ong et al., 2025; Kang et al., 2025; Zhang et al., 2026; Wu et al., 2026; Hu et al., 2026). Policy-level methods decide when and how memory operations should be applied. Agentic Memory, AgeMem, Memory-R1, and Nemori learn policies for updating information (Tan et al., 2025b; Yu et al., 2026; Yan et al., 2026; Ma et al., 2026). Nevertheless, existing systems focus mainly on the evidence directly affected by new input, while our work instead studies the downstream updates induced by it.

2.2 Prior Work on Update Propagation

Research on propagating changes through dependent results spans two related traditions. First, incremental view maintenance updates materialized views when their base data change, limiting computation to results affected by the update (Blakeley et al., 1986; Gupta et al., 1993; Gupta & Mumick, 1995; Koch et al., 2014; Budiu et al., 2023). These methods provide a principled account of update scope, but assume that dependencies are already specified by a database schema and query plan. Second, truth maintenance and belief revision study how conclusions should change when their supporting assumptions are withdrawn or contradicted. Truth-maintenance systems attach justifications to beliefs and retract conclusions whose justifications no longer hold (Doyle, 1979; de Kleer, 1986), while belief-revision theory formalizes minimal change and repeated revision under new information (Alchourrón et al., 1985; Katsuno & Mendelzon, 1991; Darwiche & Pearl, 1997). Unlike these settings, DMM operates over natural-language evidence, where dependencies are implicit and the system must both recover them as graph edges and determine how far an update should propagate.

3 Preliminaries

3.1 Derived Memory and Maintenance

Derived memory. At time t, let $\mathcal{M}_t$ be the set of stored evidence, $\mathcal{S}_t \subseteq \mathcal{M}_t$ the source evidence stated or observed directly, and $\mathcal{R}_t$ the dependencies among evidence. We define

$$\mathcal{D}_t := \mathrm{Closure}_{\mathcal{R}_t}(\mathcal{S}_t) \setminus \mathcal{S}_t, \tag{1}$$

where $\mathrm{Closure}_{\mathcal{R}_t}(\mathcal{S}_t)$ contains the source evidence and all evidence derivable from it through one or more dependencies in $\mathcal{R}_t$. Each $d \in \mathcal{D}_t$ is a *derived memory*.

Derived memory maintenance. Given an update Δ_t, *derived memory maintenance* (DMM) transforms the memory state at time t into the state at $t+1$:

$$(\mathcal{S}_t, \mathcal{D}_t, \mathcal{R}_t) \xrightarrow{\Delta_t} (\mathcal{S}_{t+1}, \mathcal{D}_{t+1}, \mathcal{R}_{t+1}), \quad \mathcal{D}_{t+1} := \mathrm{Closure}_{\mathcal{R}_{t+1}}(\mathcal{S}_{t+1}) \setminus \mathcal{S}_{t+1}.$$

Thus, DMM revises the changed source evidence together with every derived memory affected by it. DUNE implements this process using continuous validity and currentness.

3.2 DRM-bench

Each benchmark scenario contains Session 1, a source query q_1^s, a derived query q_1^d, Session 2, and the corresponding Round 2 queries q_2^s, q_2^d. Evaluation follows this order: 1) Clear the memory for one method and ingest only Session 1; 2) (QA1) Answer and score q_1^s, q_1^d, Meanwhile, the LLM must reason the evidence sources used to arrive at the answer; 3) Write the model's actual Round 1 responses back as new evidence, the relation of recorded evidence is also added as new evidence; 4) Append Session 2 without resetting the store; 5) (QA2) Answer and score q_2^s, q_2^d. 6) Evaluate and calculate metrics for the answers in the QA2.

Data sources and construction. DRM-bench uses prior datasets in three distinct ways. QA benchmarks guide question form and answer granularity, narrative and tool-use benchmarks provide content reference, and inference and contradiction datasets motivate the dependency and update

instantiated in the benchmark. The dataset was generated with GPT-5.6-sol and verified by two human annotators to ensure the validity of the synthesized data. Specifically, DRM-bench is organized into semantic, episodic, and procedural memory, which together capture the major forms of long-term memory studied in current agents. Semantic memory contains 2,400 scenarios, which draws on established QA benchmarks, including Natural Questions, TriviaQA, SQuAD, HotpotQA, and RACE (Kwiatkowski et al., 2019; Rajpurkar et al., 2016; Joshi et al., 2017; Yang et al., 2018; Lai et al., 2017). Episodic memory contains 400 scenarios grounded in the three corpora released with EpBench, default, world-news, and science-fiction settings (Huet et al., 2025). Procedural memory (Tool) contains 400 scenarios constructed from ToolBench (Qin et al., 2023). Each scenario introduces a procedural update to an existing tool-use plan. Table 1 summarizes the three categories.

Dependency and update construction. The update and dependency patterns are constructed by abstracting features from four families of prior data, NLI and contradiction benchmarks, document and legal reasoning datasets, evidence-conflict corpora, and knowledge or narrative revision tasks (Bowman et al., 2015; Williams et al., 2018; Yin et al., 2021; Koreeda & Manning, 2021; Zheng et al., 2021; Liu et al., 2021; Cheng et al., 2024; Li et al., 2024; Wang et al., 2023b; Subbiah et al., 2024). These resources guide the relation patterns and update operators used in scenario generation.

Benchmark validation. Table 1 summarizes deterministic audits of the canonical benchmark data along two dimensions: structural coverage and diversity. *Derived hop* gives the minimum intended reasoning depth required to obtain the updated derived answer, while *QA/round* reports the number of source and derived questions evaluated in each round. *Unique Qs* is the percentage of distinct question strings after text normalization, and *Asst. dup.* is the percentage of assistant turns whose normalized text exactly duplicates another assistant turn. Thus, higher Unique Qs and lower Asst. dup. indicate less reliance on repeated templates. EP and Tool show particularly high question uniqueness and low assistant-turn duplication. SE remains largely diverse at the question level, although its larger scale and recurring conversational functions lead to more repeated assistant phrasing.

4 DUNE: Derived Memory Updating through Networked Evidence

Overview. DUNE alternates between two stages. *Evidence graph construction and maintenance* adds new evidence and dependencies while preserving the existing graph. Each node carries validity and currentness. *Full-graph update propagation* then updates these variables over the complete graph and freezes the result for retrieval. Writing M for graph maintenance, P for propagation, S_1, S_2 for Session1,2, and W_1 for the actual QA1 answers, then we have

$$G_0 := \mathsf{P}(\mathsf{M}(\varnothing, S_1)), \quad G_1 := \mathsf{P}(\mathsf{M}(G_0, W_1)), \quad G_2 := \mathsf{P}(\mathsf{M}(G_1, S_2)). \tag{2}$$

As in Figure 2, QA1 retrieves from G_0 while its answers are written back before Session 2 and QA2 retrieves from G_2. Construction is incremental, whereas propagation always covers the complete visible graph.

4.1 Evidence Graph Construction and Maintenance

The first stage determines graph topology. A dependency parser decomposes stored utterances into proposition-level nodes $i \in \mathcal{N}$ containing normalized subject, predicate, object, effective time and source. For G_2, we preserve G_1 and examine only pairs containing a new node. To extract edges, all node pairs are examined, while incremental construction considers only pairs containing at least one newly added node. A SIMILAR edge is created when two propositions share the same entity, predicate, typed value, and polarity, or when cross-speaker statements have sufficient lexical overlap. A CONTRADICT edge links two nodes that cannot be simultaneously valid, sharing the same entity predicate state yet holding incompatible values. A directed DERIVED(d, p) edge is added when proposition d explicitly uses p as a premise. During the generation of the answer for qa1, the LLM is required to perform reasoning simultaneously and record which retrieved evidence was used to derive the answer. When the QA1 answer is written back into memory, the system records the premise evidence and establishes DERIVED edges among them. A directed REVISE(u, o) edge is added from a later proposition u to an older proposition o when they target the same state and differ in value, or when update cues and overlapping entities or typed values indicate replacement.

Table 1: Composition and canonical-data audit of DRM-bench. Hop denotes the minimum intended reasoning depth after update. Duplicate rates are exact-string matches.

Category	Scenarios	Derived hop	QA/round	Unique Qs ()	Asst. dup. ()
SE	2,400	3	4,800	93.00	24.69
EP	400	3	800	99.75	0.80
Tool	400	2	800	97.25	5.67
Total	3,200	–	–	6,400	–

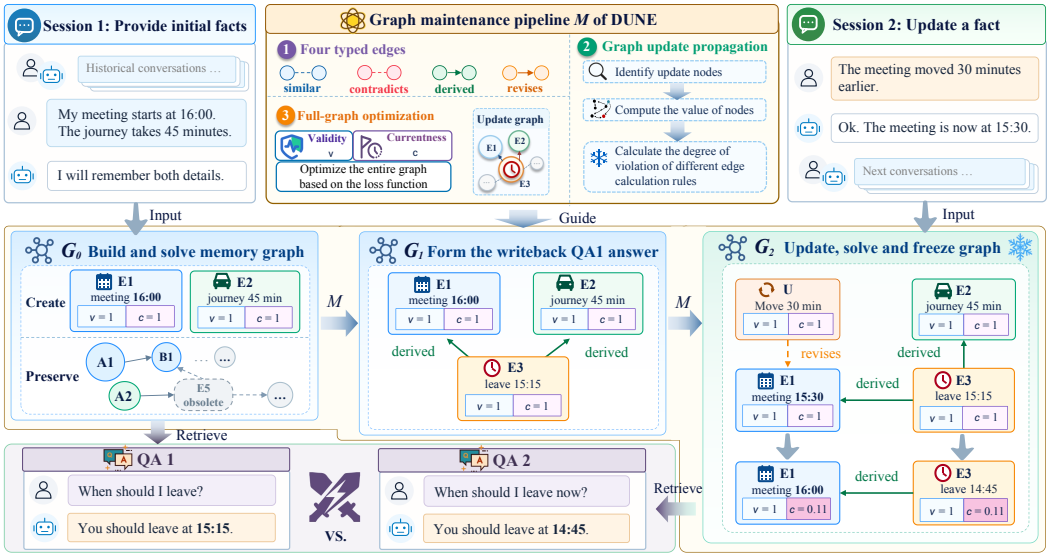

Figure 2: DUNE alternates incremental graph maintenance with full-graph propagation to produce G_0, G_1, and G_2.

4.2 Full-Graph Update Propagation

Each node $i \in \mathcal{N}$ has two variables in $[0, 1]$, validity v_i, which measures how strongly the available evidence supports the proposition, and currentness c_i, which measures whether the proposition still applies to the latest state. Both v_i and c_i are initialized to 1 upon node creation. Let $\mathcal{E}_{\text{sim}}$, $\mathcal{E}_{\text{con}}$, $\mathcal{E}_{\text{der}}$, and $\mathcal{E}_{\text{rev}}$ denote the sets of SIMILAR, CONTRADICT, DERIVED, and REVISE edges, respectively.

Edge constraints. Each edge type imposes distinct constraints on its connected nodes. A SIMILAR edge $(i, j) \in \mathcal{E}_{\text{sim}}$ connects propositions that describe a similar state with compatible values and polarity. Their validity scores should therefore differ by at most a tolerance $\delta_{\text{sim}} \geq 0$, which means $(v_i - v_j)^2 \leq \delta_{\text{sim}}^2$. A CONTRADICT edge $(i, j) \in \mathcal{E}_{\text{con}}$ connects propositions about the same state and effective time whose values or polarities are incompatible. They should not both be fully valid, which gives $v_i + v_j \leq 1$. A directed DERIVED edge $(d, p) \in \mathcal{E}_{\text{der}}$ records that proposition d was derived using proposition p as a premise. A conclusion cannot remain current after its premise becomes obsolete, such that $c_d \leq c_p$. Finally, a directed REVISE edge $(u, o) \in \mathcal{E}_{\text{rev}}$ indicates that a newer proposition u replaces an older proposition o. A valid update should reduce the currentness of the replaced proposition: $v_u + c_o \leq 1$.

From constraints to loss. Let $[x]_+ = \max(0, x)$. We denote the four nonnegative violation potentials by ϕ_{sim}, ϕ_{con}, ϕ_{der}, and ϕ_{rev}. Each potential is zero when its corresponding constraint is satisfied:

$$\begin{aligned} \phi_{\text{sim}}(i, j) &= [|v_i - v_j| - \delta_{\text{sim}}]_+^2, & \phi_{\text{con}}(i, j) &= [v_i + v_j - 1]_+^2, \\ \phi_{\text{der}}(d, p) &= [c_d - c_p]_+^2, & \phi_{\text{rev}}(u, o) &= [v_u + c_o - 1]_+^2. \end{aligned} \tag{3}$$

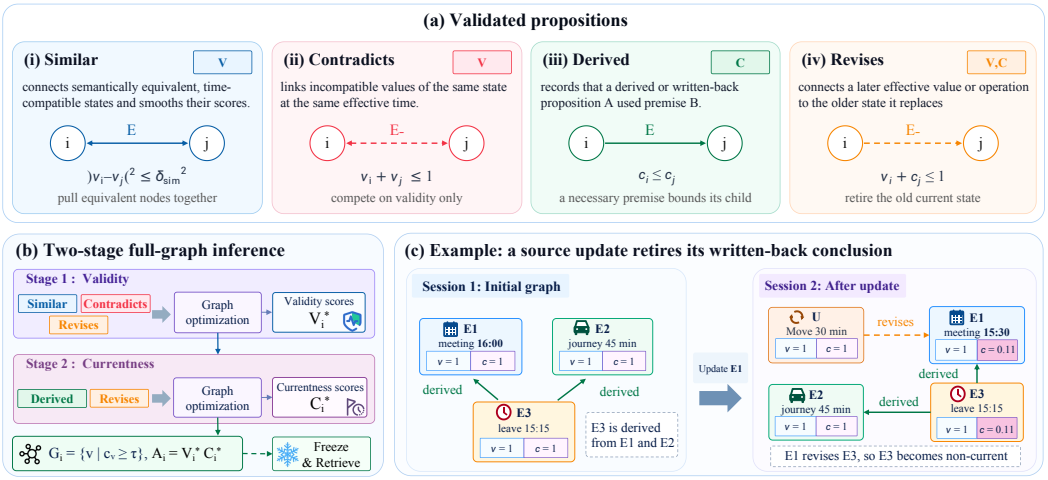

Figure 3: The four dependency factors and their joint effect on a written-back answer after a source update.

For each edge $e \in \mathcal{E}$, we define its effective weight as w. We use a uniform default validity prior, $\widehat{v}_i = 1$, for all nodes. We assume that a node admitted by the extraction stage is initially supported by its associated source evidence and the node validity is subsequently revised according to the relational constraints in the graph. We distinguish this prior from the optimization initialization. During incremental inference, existing nodes are warm-started from their previously optimized values and newly added nodes are initialized to one. Meanwhile, λ denotes the weight that controls the trade-off between retaining the default validity assumption and satisfying the relational constraints. We write $\mathbf{v} = (v_i)_{i\in\mathcal{N}}$ and $\mathbf{c} = (c_i)_{i\in\mathcal{N}}$. The validity loss is

$$\mathcal{L}_v(\mathbf{v}) = \lambda \sum_{i\in\mathcal{N}} (v_i - \widehat{v}_i)^2 + \sum_{e=(i,j)\in\mathcal{E}_{\text{sim}}} w\phi_{\text{sim}}(i,j) + \sum_{e=(i,j)\in\mathcal{E}_{\text{con}}} w\phi_{\text{con}}(i,j). \tag{4}$$

For a fixed validity vector $\mathbf{v}$, the currentness loss is

$$\mathcal{L}_c(\mathbf{c};\mathbf{v}) = \lambda \sum_{i\in\mathcal{N}} (c_i - 1)^2 + \sum_{e=(d,p)\in\mathcal{E}_{\text{der}}} w\phi_{\text{der}}(d,p) + \sum_{e=(u,o)\in\mathcal{E}_{\text{rev}}} w\phi_{\text{rev}}(u,o). \tag{5}$$

The first loss keeps validity close to the extracted evidence. The second assigns currentness one in the absence of evidence that a proposition has become obsolete.

Solving the objective. Both subproblems are convex once validity is fixed for the currentness stage. We first compute $\mathbf{v}^* = \arg\min_{\mathbf{v}\in[0,1]^n} \mathcal{L}_v(\mathbf{v})$ by projected gradient descent with backtracking. Revision does not feed back into this stage, since whether a proposition has been replaced should not change whether it was supported by evidence. We then compute $\mathbf{c}^* = \arg\min_{\mathbf{c}\in[0,1]^n} \mathcal{L}_c(\mathbf{c};\mathbf{v}^*)$ by coordinate updates. After each update, every derived pair is projected onto $c_d \le c_p$ to enforce dependency consistency. Each stage stops when the maximum variable change is below the convergence tolerance or the iteration limit is reached. For the incremental graph G_2, the states from G_1 provide the initialization. Finally, after defining the activity of node i as $a_i = v_i^* c_i^*$, retrieval ranks propositions by a_i, so active evidence must be both valid and current.

5 Experiments

5.1 Evaluation Setup

Settings. We use Qwen-Flash for answer generation and for methods that require an internal language model, memory extraction and update. Retrieval-based methods return the top $k = 10$ memory items. In DUNE, we set $\lambda = 1$, $w = 8.0$ and optimize the full graph for at most 200

iterations, terminating when the residual falls below 10^{-5}. After all answers are fixed, GLM-5.3 is used to evaluate the answer. Each experiment is repeated five times, and the final results are reported as averages.

DRM-bench Metrics. Each scenario contains two probes, a *source probe*, which asks for the updated source evidence, and a *derived probe*, which asks for a consequence that depends on it. For each probe, the memory system generates an answer. We report token-level F1 and an independent LLM-judge score(J). Let $y_{i,s}^{(2)}$ and $y_{i,d}^{(2)}$ denote J for the source and derived probes in the second round. We compute

$$\textsc{SourceUpdate} = \frac{\sum_{i=1}^{N} y_{i,s}^{(2)}}{N}, \qquad \textsc{DerivedUpdate} = \frac{\sum_{i=1}^{N} y_{i,d}^{(2)}}{N}. \tag{6}$$

Thus, SOURCEUPDATE is the proportion of second-round source probes answered correctly, while DERIVEDUPDATE is the corresponding proportion for derived probes.

Baselines. We compare DUNE with three retrieval baselines: a deterministic token-hash retriever, dense RAG using sentence embeddings (Lewis et al., 2020; Reimers & Gurevych, 2019), and BM25 RAG (Robertson & Zaragoza, 2009). We also evaluate five general-purpose memory systems: Mem0 (Chhikara et al., 2025), A-MEM (Xu et al., 2025), LangMem (LangChain, 2026), LightMem (Zhang et al., 2026), and PowerMem, evaluated through its PowerContext implementation (PowerContext Team, 2026). These systems differ in how they construct, organize, and retrieve memories. All baselines are evaluated under the same datasets, language model, top-k, and evaluation protocol to ensure a fair comparison.

5.2 MAIN RESULTS

Experiments are shown in Table 2. Three main conclusions can be derived as below.

1) Overall, DUNE achieves state-of-the-art performance across most metrics. As shown in Table 2, DUNE outperforms all eight baselines on F1 (52.0%, 9.6 points higher on average), SourceUpdate (84.8%, 22.2 points higher), and DerivedUpdate (73.6%, 22.5 points higher). Compared against the strongest individual baseline for each metric, DUNE leads by 2.0% in F1, 0.3% in SourceUpdate, and 12.2% in DerivedUpdate. Notably, its advantage is substantially larger on DerivedUpdate than on SourceUpdate. This is a gap consistent with our central claim: systems can often recover an updated source, yet still fail to revise conclusions that depend on it.

2) The magnitude of DUNE 's advantage varies across datasets, suggesting that different memory types require different retrieval strategies. DUNE 's DerivedUpdate advantage is most pronounced on EP (89.3%), indicating that dependency structured memory is particularly suited for episodic reasoning. On Tool, however, DUNE 's DerivedUpdate score (65.3%) is only 4.8 points above the strongest RAG baselines (Hash RAG and BM25 RAG, both 60.5%, the Tool scores are 60.7% and 58.0% respectively). This smaller gap may reflect the procedural nature of tool workflows, that is, updating a tool pipeline requires preserving call order, parameter dependencies, and branching conditions, not merely revising individual evidence nodes.

3) A-MEM achieved considerable performance, which may stem from its associative memory organization. By explicitly linking related evidence, A-MEM facilitates retrieval of relevant sources and maintains coherent context, which explains its competitive LLM Judge. Nevertheless, its average DerivedUpdate (61.4%) is 12.2 points below DUNE, which suggests that associative retrieval alone cannot substitute for explicit dependency propagation when conclusions must be revised in light of upstream changes.

5.3 ABLATIONS

We conduct ablation experiments on a random 200 scenarios from the SE dataset to isolate the contribution of DUNE's key components. Two variants are evaluated: (1) w/o propagation, which retains the evidence graph structure but removes full-graph HL-MRF propagation; and (2) w/o derived edge, which removes the representation and maintenance of derived evidence.

Table 2: Full results (%) on DRM-bench. **Src.** and **Der.** denote source and derived update accuracy; Avg. is the macro average across the three memory types.

	DRM-SE				DRM-EP				DRM-Tool				Avg.			
Method	F1	J	Src.	Der.	F1	J	Src.	Der.	F1	J	Src.	Der.	F1	J	Src.	Der.
Hash RAG	37.4	44.6	47.5	41.7	**78.2**	**83.1**	86.5	79.8	27.6	71.4	76.2	60.5	47.7	66.4	70.1	60.7
Dense RAG	37.9	42.0	46.9	37.1	77.2	81.6	85.2	78.0	27.2	69.5	80.0	59.0	47.4	64.4	70.7	58.0
BM25 RAG	36.2	39.1	42.5	35.6	77.8	82.8	87.5	78.0	27.8	71.9	77.2	60.5	47.3	64.6	69.1	58.0
Mem0	40.6	20.8	45.8	29.5	59.8	25.3	61.5	38.0	18.0	9.0	59.8	13.5	39.5	18.4	55.7	27.0
A-MEM	47.2	**71.2**	83.8	58.5	74.1	75.5	85.5	65.5	28.8	**76.8**	**84.2**	60.2	50.0	**74.5**	84.5	61.4
LangMem	41.5	65.4	70.5	60.4	52.7	57.2	59.2	55.2	18.0	52.4	54.5	50.2	37.4	58.3	61.4	55.3
LightMem	31.7	48.1	56.0	40.3	40.9	33.1	35.8	30.5	15.1	32.2	24.5	40.0	29.2	37.8	38.8	36.9
PowerMem	33.8	34.6	36.5	32.8	69.6	69.9	64.8	75.0	18.6	49.1	51.2	47.0	40.7	51.2	50.8	51.6
DUNE	**48.3**	58.3	**85.5**	**66.3**	73.7	81.3	**90.0**	**89.3**	**34.0**	68.0	78.8	**65.3**	**52.0**	69.2	**84.8**	**73.6**

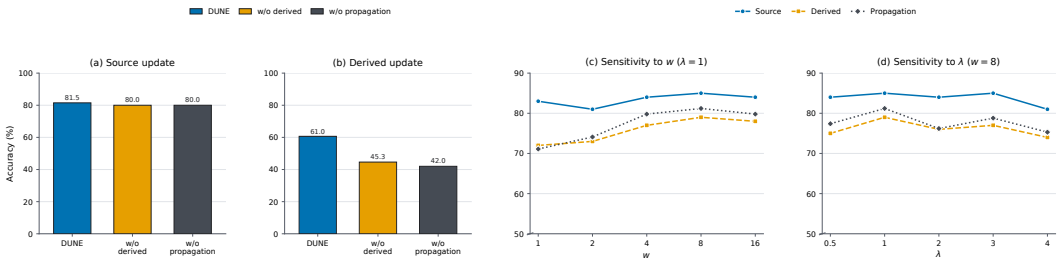

Figure 4: DUNE ablation and hyperparameter sensitivity on DRM-SE. The left figure compares source- and derived-update accuracy across DUNE and its ablated variants. The right figure shows how source, derived, and propagation accuracy vary with w and λ, respectively.

As shown in Figure 4, both ablations cause a minimal drop in source update (81.5% $\rightarrow$ 80.0%, a reduction of 1.5 points), while derived update drops substantially from 61.0% to 45.3% and 42.0% under two conditions. This pattern suggests that the two ablated components contribute to derived evidence revision through a common mechanism than to source evidence. Specifically, the evidence graph provides the structural scaffold, while the derived edge and propagation process together enable changes at the source to cascade through the dependency hierarchy and update downstream conclusions. When either component is removed, the system can still retrieve updated sources, but it fails to propagate those changes to dependent derived facts—precisely matching the observed decline in derived update.

5.4 Error Analysis

We examine two complementary metrics after an update. PropagationAccuracy measures whether the derived probe is answered correctly among scenarios in which the updated source has already been recovered. OldAnswerResidual measures how often the model repeats the obsolete Round 1 answer when the old and new targets differ. Table 3 reports the results.

The methods differ substantially across domains. DUNE has the highest macro-average propagation accuracy (69.7%) and the lowest macro-average old-answer residual (2.4%). LangMem performs best on SE (80.1%), whereas DUNE performs best on both EP (87.2%) and Tool (68.1%); it also has the lowest EP residual (1.4%). On Tool, all methods have zero exact old-answer residual, so the remaining errors are not explained by verbatim reuse of stale answers alone; they more likely reflect difficulty producing the updated procedure after avoiding the obsolete answer.

5.5 Sensitivity to Propagation Weights

We use a fixed random sample of 100 SE scenarios and vary one weight at a time around the default $(\lambda, w) = (1, 8)$. Claude Haiku 4.5 generates the answers, while GLM-5.3 serves as the offline judge.

Table 3: Conditional propagation accuracy (Prop.) and exact old-answer residual (OldRes.) on the full benchmark. All values are percentages. Avg. is the macro average across the three datasets.

	DRM-SE		DRM-EP		DRM-Tool		Avg.	
Method	Prop.	OldRes.	Prop.	OldRes.	Prop.	OldRes.	Prop.	OldRes.
Hash RAG	54.2%	12.8%	81.8%	6.0%	67.5%	**0.0%**	**67.8%**	6.3%
Dense RAG	47.3%	12.7%	79.8%	6.6%	59.1%	**0.0%**	62.1%	6.4%
BM25 RAG	48.4%	15.3%	80.3%	7.0%	67.6%	**0.0%**	65.4%	7.4%
Mem0	43.6%	20.8%	40.1%	35.8%	19.7%	**0.0%**	34.5%	18.9%
A-MEM	61.0%	6.5%	68.1%	13.8%	58.8%	**0.0%**	62.6%	6.8%
LangMem	**80.1%**	5.3%	64.1%	9.1%	57.3%	**0.0%**	67.2%	4.8%
LightMem	59.3%	6.9%	35.7%	29.0%	58.2%	**0.0%**	51.1%	12.0%
PowerMem	47.6%	15.8%	78.4%	13.0%	51.7%	**0.0%**	59.2%	9.6%
DUNE	53.8%	5.7%	**87.2%**	**1.4%**	**68.1%**	**0.0%**	**69.7%**	**2.4%**

All settings use the same sample IDs and Round 1 model responses for writeback. Details can be found in Appendix B. At $\lambda = 1$, increasing w from 1 to 8 raises derived update from 72% to 79%, increasing it further to 16 changes these scores only slightly, to 78%. With $w = 8$, $\lambda = 1$ gives the highest derived update and propagation accuracy. The range $w \in [4, 16]$ is comparatively stable, and the default $(1, 8)$ is the best observed setting. Overall, the performance curves exhibit acceptable fluctuations, which indicates that DUNE is relatively insensitive to hyperparameters. Accordingly, a subset of samples can be reserved as a development set for hyperparameter tuning prior to formal experimentation.

5.6 Cost Measurement

We compare DUNE and BM25 RAG with Claude Haiku 4.5 on 50 scenarios. We organize the measurements into time cost, space cost and API cost. For runtime, DUNE's graph construction and propagation take 0.0564 s per scenario, while BM25 index construction takes 0.00135 s, Both are small relative to QA. DUNE uses twice as many calls and about 4.4 times as many prompt tokens as BM25 RAG. For memory, DUNE has a higher peak resident footprint. Overall, the result suggests that DUNE's added graph computation and memory overhead are acceptable, while its main additional cost comes from increased model API usage.

Table 4: Average cost per scenario on the pilot (pilot totals divided by 50). QA times are cumulative per-question processing times. Peak RSS is amortized.

Time cost

Metric	DUNE	BM25
Graph/index setup (s)	0.0551	0.00135
Graph propagation (s)	0.0013	–
QA1 (s)	45.485	29.42
QA2 (s)	52.96	22.985

Space and API cost

Metric	DUNE	BM25
Space: peak RSS (MiB)	33.7	24.25
API: model calls	24	12
API: prompt tokens	135.8K	31K
API: completion tokens	7.5K	0.95K

6 Conclusion

Long-term memory systems typically update the evidence directly mentioned in new input, but an update can also invalidate conclusions stored earlier. We frame this overlooked problem as DMM, introduce DRM-bench across semantic, episodic, and procedural memory, and propose DUNE, which maintains a graph and propagates updates over the full graph before retrieval. Experiments have shown that DUNE improves averaged F1, Judge accuracy, source-update accuracy, and derived-update accuracy by 9.6, 14.8, 22.2, and 22.5 percentage points, respectively. Future work could refine the extraction of nodes and edges, and incorporate a broader range of scenarios into the parameterization of the propagation mechanism.

AI USE STATEMENT

We used generative AI tools to generate and diversify synthetic benchmark data, translate research materials, clean and reformat datasets and create or edit experimental software. We have reviewed AI-assisted data and code by 2 authors. AI outputs were treated as suggestions in our writing. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50(2):510–530, 1985. doi: 10.2307/2274239.

Jose A. Blakeley, Per-Åke Larson, and Frank Wm. Tompa. Efficiently updating materialized views. *ACM SIGMOD Record*, 15(2):61–71, 1986. doi: 10.1145/16856.16861.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642, 2015. doi: 10.18653/v1/D15-1075.

Mihai Budiu, Tej Chajed, Frank McSherry, Leonid Ryzhyk, and Val Tannen. DBSP: Automatic incremental view maintenance for rich query languages. *Proceedings of the VLDB Endowment*, 16 (7):1601–1614, 2023. doi: 10.14778/3587136.3587137.

Jiayang Cheng, Chunkit Chan, Qianqian Zhuang, Lin Qiu, Tianhang Zhang, Tengxiao Liu, Yangqiu Song, Yue Zhang, Pengfei Liu, and Zheng Zhang. ECON: On the detection and resolution of evidence conflicts. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7816–7844, 2024. doi: 10.18653/v1/2024.emnlp-main.447.

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.

Adnan Darwiche and Judea Pearl. On the logic of iterated belief revision. *Artificial Intelligence*, 89 (1–2):1–29, 1997. doi: 10.1016/S0004-3702(96)00038-0.

Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28(2):127–162, 1986. doi: 10.1016/0004-3702(86)90080-9.

Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12(3):231–272, 1979. doi: 10.1016/0004-3702(79)90032-7.

Ashish Gupta and Inderpal Singh Mumick. Maintenance of materialized views: Problems, techniques, and applications. *IEEE Data Engineering Bulletin*, 18(2):3–18, 1995.

Ashish Gupta, Inderpal Singh Mumick, and V. S. Subrahmanian. Maintaining views incrementally. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp. 157–166, 1993. doi: 10.1145/170035.170066.

Chuanrui Hu, Xingze Gao, Zuyi Zhou, Dannong Xu, Yi Bai, Xintong Li, Hui Zhang, Tong Li, Chong Zhang, Lidong Bing, and Yafeng Deng. EverMemOS: A self-organizing memory operating system for structured long-horizon reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 45836–45853, 2026. doi: 10.18653/v1/2026.acl-long.2125.

Alexis Huet, Zied Ben Houidi, and Dario Rossi. Episodic memories generation and evaluation benchmark for large language models. *arXiv preprint arXiv:2501.13121*, 2025. doi: 10.48550/arXiv.2501.13121.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017. doi: 10.18653/v1/P17-1147.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25961–25970, 2025. doi: 10.18653/v1/2025.emnlp-main.1318.

Hirofumi Katsuno and Alberto O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52(3):263–294, 1991. doi: 10.1016/0004-3702(91)90069-V.

Christoph Koch, Yanif Ahmad, Oliver Kennedy, Milos Nikolic, Andres Nötzli, Daniel Lupei, and Amir Shaikhha. DBToaster: Higher-order delta processing for dynamic, frequently fresh views. *The VLDB Journal*, 23(2):253–278, 2014. doi: 10.1007/s00778-013-0348-4.

Yuta Koreeda and Christopher D. Manning. ContractNLI: A dataset for document-level natural language inference for contracts. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 1907–1919, 2021. doi: 10.18653/v1/2021.findings-emnlp.164.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, 2017. doi: 10.18653/v1/D17-1082.

LangChain. LangMem: Long-term memory for language-model agents. Software repository, 2026. URL `https://github.com/langchain-ai/langmem`. Evaluated revision 29cbe41e58528f92e9efa773c12e15c47be3808c; accessed 2026-09-04.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rocktaschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Jierui Li, Vipul Raheja, and Dhruv Kumar. ContraDoc: Understanding self-contradictions in documents with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6509–6523, 2024. doi: 10.18653/v1/2024.naacl-long.362.

Hanmeng Liu, Leyang Cui, Jian Liu, and Yue Zhang. Natural language inference in context: Investigating contextual reasoning over long texts. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13388–13396, 2021. doi: 10.1609/aaai.v35i15.17580.

Wenquan Ma, Jiayan Nan, and WenLong Wu. What deserves memory: Adaptive memory distillation for LLM agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34789–34812, 2026. doi: 10.18653/v1/2026.acl-long.1607.

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. doi: 10.18653/v1/2024.acl-long.747.

Kai Tzu-iunn Ong, Namyoung Kim, Minju Gwak, Hyungjoo Chae, Taeyoon Kwon, Yohan Jo, Seungwon Hwang, Dongha Lee, and Jinyoung Yeo. Towards lifelong dialogue agents via timeline-based memory management. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8631–8661, 2025. doi: 10.18653/v1/2025.naacl-long.435.

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. doi: 10.1145/3586183.3606763.

PowerContext Team. PowerContext: Handoff-ready context for human-agent work. Software repository, 2026. URL https://github.com/oceanbase/powercontext. Evaluated revision 652e11ccddaad4039442f259596c7cdb7f78962a; accessed 2026-09-04.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. *arXiv preprint arXiv:2307.16789*, 2023. doi: 10.48550/arXiv.2307.16789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016. doi: 10.18653/v1/D16-1264.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of EMNLP-IJCNLP*, pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

Melanie Subbiah, Faisal Ladhak, Akankshya Mishra, Griffin Thomas Adams, Lydia Chilton, and Kathleen McKeown. STORYSUMM: Evaluating faithfulness in story summarization. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9988–10005, 2024. doi: 10.18653/v1/2024.emnlp-main.557.

Haoran Sun, Zekun Zhang, and Shaoning Zeng. Preference-aware memory update for long-term LLM agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 783–793, 2026. doi: 10.18653/v1/2026.findings-acl.38.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. MemBench: Towards more comprehensive evaluation on the memory of LLM-based agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19336–19352, 2025a. doi: 10.18653/v1/2025.findings-acl.989.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8416–8439, 2025b. doi: 10.18653/v1/2025.acl-long.413.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In *Advances in Neural Information Processing Systems*, volume 36, pp. 74530–74543, 2023a. doi: 10.52202/075280-3259.

Yike Wang, Shangbin Feng, Heng Wang, Weijia Shi, Vidhisha Balachandran, Tianxing He, and Yulia Tsvetkov. Resolving knowledge conflicts in large language models. *arXiv preprint arXiv:2310.00935*, 2023b. doi: 10.48550/arXiv.2310.00935.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. Evo-Memory: Benchmarking LLM agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL-HLT*, pp. 1112–1122, 2018. doi: 10.18653/v1/N18-1101.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Zhaofen Wu, Hanrong Zhang, Fulin Lin, Wujiang Xu, Xinran Xu, Yankai Chen, Henry Peng Zou, Shaowen Chen, Weizhi Zhang, Xue Liu, Philip S. Yu, and Hongwei Wang. GAM: Hierarchical graph-based agentic memory for LLM agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34647–34664, 2026. doi: 10.18653/v1/2026.acl-long.1600.

Jundong Xu, Qingchuan Li, Jiaying Wu, Yihuai Lan, Shuyue Stella Li, Huichi Zhou, Bowen Jiang, Lei Wang, Jun Wang, Anh Tuan Luu, Caiming Xiong, Hae Won Park, Bryan Hooi, and Zhiyuan Hu. EvoArena: Tracking memory evolution for robust LLM agents in dynamic environments. *arXiv preprint arXiv:2606.13681*, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 17577–17604, 2025. doi: 10.52202/085713-0593.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825, 2026. doi: 10.18653/v1/2026.acl-long.583.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259.

Wenpeng Yin, Dragomir Radev, and Caiming Xiong. DocNLI: A large-scale dataset for document-level natural language inference. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 4913–4922, 2021. doi: 10.18653/v1/2021.findings-acl.435.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21457–21483, 2026. doi: 10.18653/v1/2026.acl-long.981.

Jiaquan Zhang, Chaoning Zhang, Shuxu Chen, Zhenzhen Huang, Pengcheng Zheng, Zhicheng Wang, Ping Guo, Fan Mo, Sung-Ho Bae, Jie Zou, Jiwei Wei, and Yang Yang. Lightweight LLM agent memory with small language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12914–12929, 2026. doi: 10.18653/v1/2026.acl-long.588.

Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. When does pretraining help? assessing self-supervised learning for law and the CaseHOLD dataset of 53,000+ legal holdings. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, pp. 159–168, 2021. doi: 10.1145/3462757.3466088.

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. doi: 10.1609/aaai.v38i17.29946.

A Component Ablation

Table 5: Component ablation on 200 SE scenarios with identical sample IDs. All values are percentages; bold marks the best judge-based result.

Variant	F1	Judge	Source	Derived	Prop.
DUNE	49.1	**66.3**	**81.5**	**61.0**	**51.5**
w/o DERIVED	51.7	61.0	80.0	45.3	44.4
w/o propagation	52.3	61.0	80.0	42.0	43.8

We conduct a component ablation on 200 DRM-SE scenarios, using identical sample IDs across all variants. The full DUNE system is compared with two variants: *w/o* DERIVED, which removes the representation and maintenance of derived evidence, and *w/o propagation*, which retains the derived-evidence setup but disables update propagation over the graph. This comparison isolates the contribution of representing derived evidence from the contribution of propagating an update through dependencies.

Table 5 shows that the full model obtains the strongest Judge-based performance: 66.3% overall Judge accuracy, 81.5% Source Update, 61.0% Derived Update, and 51.5% Propagation Accuracy. Removing either component leaves Source Update nearly unchanged, decreasing it by only 1.5 percentage points. The larger differences appear in downstream measures. Both ablations reduce Derived Update by 15.7, 19.0 points relative to DUNE without derived and without propagation, respectively. Propagation Accuracy also falls, from 51.5% to 44.4% without derived-evidence handling and 43.8% without propagation.

The similar scores of the two ablations suggest that neither component alone accounts for the full system's improvement: representing derived evidence without propagating updates is insufficient, and propagation without the derived-evidence component likewise fails to preserve the same downstream gains. The results are consistent with the two components working together to maintain dependencies after a source update. The pattern is also specific to update-sensitive metrics: removing components slightly increases answer F1 (from 49.1% to 51.7–52.3%), even as Judge accuracy and all three update measures decline. Thus, F1 alone does not capture whether a system correctly revises both a changed source and the conclusions that depend on it.

B Sensitivity Analysis

Table 6: Sensitivity to the currentness prior λ and revise-edge weight w on a fixed random sample of 100 SE scenarios. In the general objective in the main text, these correspond to the currentness prior coefficient λ and the weight w on revise-edge penalties, respectively. One weight varies while the other remains at its default. Prop. denotes conditional propagation accuracy. All values are percentages.

Value	Source	Derived	Joint	Prop.
$\lambda = 1$, varying w				
1	83	72	59	71.1
2	81	73	60	74.1
4	84	77	67	79.8
8	**85**	**79**	**69**	**81.2**
16	84	78	67	79.8
$w = 8$, varying λ				
0.5	84	75	65	77.4
1	**85**	**79**	**69**	**81.2**
2	84	76	64	76.2
3	**85**	77	67	78.8
4	81	74	61	75.3

C Formal Evaluation Protocol

For each dataset D and method m, the evaluator creates a separate empty memory store and performs the following operations in order:

1. ingest all selected Session 1 evidence into the store for m;
2. if $m =$ DUNE, build and solve the complete sample-scoped graph G_1;
3. answer all Round 1 source and derived questions from the current store or frozen G_1;
4. write every actual Round 1 model response into the same method-specific store with `source_type=qa_writeback`;
5. append Session 2 evidence without clearing the store;
6. if $m =$ DUNE, rebuild and solve G_2 from Session 1, QA1 writeback, and Session 2;
7. answer all Round 2 questions from the current store or frozen G_2; and
8. after generation has finished, score all outputs with a fixed offline judge.

In a 20-scenario batch, each track has two QA1 questions per scenario, so each mode must add exactly 40 pieces of writeback evidence; a joint default/derived log therefore reports 80 additions, zero skips, and zero failures. Round 2 retrieval traces must include evidence tagged with `qa_writeback`. A result is invalid if Session 1 and Session 2 are ingested before QA1, if QA1 responses are not written back, if gold answers replace model responses, or if the two methods share a memory store.

D Benchmark Details

D.1 SE categories

The 12 balanced categories are temporal update, rule change, relation update, precedent update, contract-clause update, evidence reweighting, negation update, quantifier update, self-contradiction, contextual exception, knowledge conflict, and story-fact revision. Each category contributes 200 source and 200 derived questions. Source questions have an intended hop level of two, and derived questions an intended hop level of three. In the canonical set, 30.67% of scenarios contain more than two update rounds, and 18% of derived targets are explicitly marked as requiring uncertainty-aware answers.

D.2 EP categories

The five source categories revise location, time, actor, object, or completion status. Each is paired with a three-hop downstream category: location-to-desk, time-to-window, actor-to-receiver-status, object-to-corridor, or completion-to-escalation-queue propagation. Narrative surface forms draw on three source-corpus styles and rotate across 22 everyday activities and seven evidence families. The stories deliberately avoid using access credentials or cameras as default plot devices.

D.3 Tool categories

The 20 tool categories are tool substitution, parameter correction, entity-ID correction, dependency migration, permission-scope update, rate-limit constraint, batch-to-streaming, idempotency, approval gates, source-of-truth switch, tenant routing, time-zone correction, unit correction, SLA tightening, policy override, constraint injection, environment-state change, objective shift, partial-completion recovery, and failure-driven replanning.

D.4 Data validation

After automatic generation, every scenario was manually inspected for logical and linguistic correctness. The inspection checked: (i) whether the Session 1 source evidence and its gold answer were correct; (ii) whether the derived answer followed from the stated premises; (iii) whether Session 2 contained a valid update to the source evidence; (iv) whether the Round 2 source and derived gold answers were correctly recomputed; (v) whether the derived answer depended on the claimed premises;

and (vi) whether Session 2 directly revealed the Round 2 derived answer. Invalid examples were corrected when the intended scenario remained unambiguous. Examples with ambiguous premises, inconsistent gold answers, incomplete dependencies, or direct leakage of the derived answer were removed. We retained 3,200 scenarios after this quality-control process.

E Node and Edge Construction

The parser first splits each memory into sentences and uses spaCy's `en_core_web_sm` dependency parse to extract proposition nodes. For each verbal predicate, it collects the grammatical subject and object, lemmatizes the predicate, and records negation, modality, time expressions, speaker, session, timestamp, and source. A sentence may therefore yield multiple nodes when it contains multiple predicates or objects. Node IDs are derived from the memory identifier, sentence and predicate positions, and extracted proposition fields. Candidate edge pairs are restricted to the same scenario; in G_2, the existing G_1 graph is retained and new edges are considered only for pairs containing a newly added node. Each candidate receives a rule-based confidence score; candidates below 0.6 are discarded, and duplicate source–target–relation triples are retained only once.

Edge candidates are generated from the normalized subject–predicate–object tuple, polarity, temporal order, and local discourse cues. The four edge types used in the main text are extracted as follows:

- SIMILAR(i, j) links nodes with the same normalized subject, predicate, object, and polarity, or cross-speaker statements with sufficient lexical overlap. Cross-speaker candidates require at least three shared content tokens and a Jaccard overlap of at least 0.14. The edge is symmetric when no temporal order is available; otherwise it points from earlier to later corroborating evidence.
- CONTRADICT(i, j) links nodes with the same subject and predicate but incompatible object values or polarity, when the statements cannot be reconciled as a time-ordered revision. This relation is symmetric; the same-slot conflict candidate is assigned confidence 0.82.
- DERIVED(d, p) points from a conclusion to a premise when a result connector (e.g., "therefore" or "as a result") links adjacent propositions, or when QA1 writeback records proposition IDs used to produce the answer. The connector-based candidate receives confidence 0.70. Provenance IDs are checked against the nodes visible to that answer; unresolved or cross-scenario references are discarded.
- REVISE(u, o) points from a later update u to an earlier proposition o when they address the same subject–predicate state but differ in value or polarity. Temporal order is determined first by session, then timestamp or event order; update markers and typed-value overlap provide additional evidence that the later statement replaces rather than merely contrasts with the earlier one. Same-slot newer-value candidates receive confidence 0.88, while candidates supported by an update marker receive confidence 0.76.

Repeated statements are treated as support rather than revision when their extracted values and polarity agree. A pair with incompatible values is not labeled as a revision solely because one sentence occurs later: the shared state target and update evidence must also match. Duplicate edges are removed before propagation. Session 2 updates may revise source nodes selected from a QA1 answer's recorded provenance, but may not directly revise the QA answer node; its staleness follows the derived edge to its premise.

F Metric Definitions

Let U_s and U_d be the sets of updated Round 2 source and derived questions. With correctness indicator $C(x)$,

$$\textsc{SourceUpdate} = |U_s|^{-1} \sum_{x \in U_s} C(x), \tag{7}$$

$$\textsc{DerivedUpdate} = |U_d|^{-1} \sum_{x \in U_d} C(x). \tag{8}$$

For each scenario whose source questions are all correct, let U_d^+ contain its updated derived questions. Then

$$\text{PROPAGATIONACCURACY} = |U_d^+|^{-1} \sum_{x \in U_d^+} C(x). \tag{9}$$

For old-answer residual, let E_p be the Round 2 examples for probe type $p \in \{s, d\}$ whose old and new gold answers differ after normalization. Normalization lowercases text, trims leading and trailing whitespace, collapses internal whitespace, and removes every character except ASCII letters, digits, spaces, and colons. The exact residual for probe p is

$$R_p^{\text{exact}} = \frac{\sum_{x \in E_p} \mathbb{1}[N(r_x) = N(a_x^{\text{old}})]}{|E_p|}, \tag{10}$$

where r_x is the generated Round 2 response, a_x^{old} is the old answer, and $N(\cdot)$ is the normalization above. Thus, the denominator includes only updated cases, separately for source and derived probes. The reported overall residual is the unweighted mean of the source and derived rates when both are defined.

The containment-aware residual uses the same denominator and normalization, but counts a match when the normalized old answer occurs in the normalized response as a complete phrase bounded by the beginning/end of the response or a space. It does not count an old answer that appears only as a substring inside a longer token. Minimal justification is strict when the highest-ranked memory contains new evidence and the top three contain no old evidence; the loose variant allows new evidence anywhere among the top three. State-evidence analysis separately records old and new evidence retrieval rates, old-only/new-only/both/none compositions, and $\Delta_{\text{state}} = P(\text{new retrieved}) - P(\text{old retrieved})$.

G QUALITATIVE EXAMPLE

Table 7: A schematic SE instance. The Round 2 derived answer is not written in Session 2.

Stage	Content
Session 1	The client review is at 08:00. Travel to the office takes two hours.
QA1 source	Current review time? → 08:00.
QA1 derived	When should travel begin? → 06:00.
Writeback	The model's two actual answers are stored.
Session 2	Move the appointment 90 minutes earlier; the travel rule is unchanged.
QA2 source	Current review time? → 06:30.
QA2 derived	When should travel begin? → 04:30.

A default retriever may surface "06:00" because it is a short, direct answer written during QA1. DUNE records that this answer was derived from the earlier appointment, lets the relative update revise that premise, and then lowers the answer's currentness through the DERIVED edge constraint. The earlier time remains available as historical evidence for computing 06:30, after which the model applies the unchanged two-hour travel rule. This example also illustrates why evaluating only the source question is insufficient: a system may correctly return 06:30 yet still answer 06:00 downstream.

H EVALUATION SCOPE

The full-set comparison evaluates Mem0, DUNE, three RAG variants, A-MEM, LangMem, LightMem, and PowerMem on all 3,200 scenarios. All reported runs use method-specific memory stores and the required QA1 writeback chronology; development runs that violate runtime blindness or use incomplete writeback are excluded.