DeepMem: Deep Memory Passing for LLM Agents

Anonymous ACL submission

Abstract

Recent agentic memory systems have progressed from flat vector indices to dynamic memory graphs to maintain evolving knowledge during long-horizon interactions, yet they operate under strict shallow topological constraints that limit memory traversals. This rigid single-hop design leaves critical challenges unresolved: it prevents state-changes from propagating to multi-hop dependent notes at write time, and fails to capture deep, mediated relevance at read time. To overcome these limitations, we present **DeepMem**, a novel framework designed to execute deep memory passing over both the write-time and read-time pathways. Specifically, DeepMem leverages a *Deep Memory Update* strategy to recursively propagate LLM-generated change summaries along graph edges via delta-anchored scoring, and symmetrically utilizes a *Deep Memory Retrieval* mechanism to diffuse query relevance through structural proxies. Comprehensive evaluations on the challenging LoCoMo benchmark demonstrate that DeepMem significantly outperforms state-of-the-art shallow methods, achieving a substantial weighted overall F1 improvement of $+15.7\%$ on Qwen2.5-1.5B-Instruct and $+58.0\%$ on Qwen2.5-3B-Instruct.

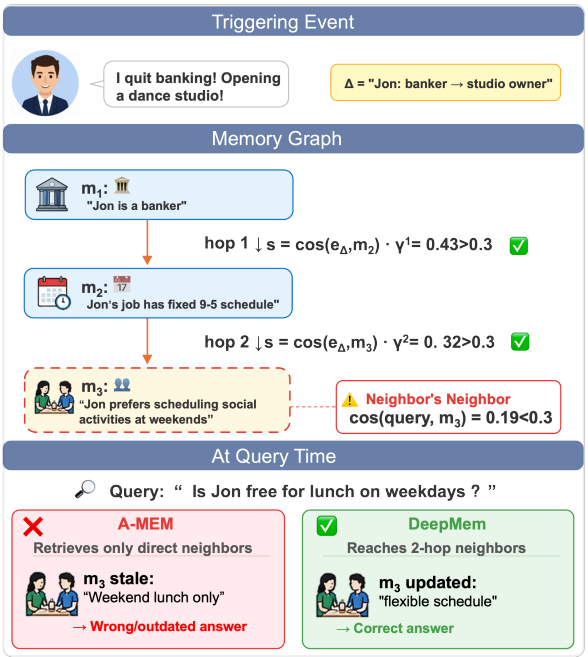

Figure 1: A motivating example for the necessity of deep memory passing. When Jon changes careers (Δ: banker $\rightarrow$ studio owner), the memory note m_3 is two-hop away from the updated m_1 and has low direct similarity, making it silently stale. In contrast, our **DeepMem** reaches m_3 via delta-anchored graph propagation, correcting it before the new query arrives.

1 Introduction

Large language model (LLM) agents are increasingly deployed in long-horizon interactions that require persistent and evolving knowledge of users and environments (Wang et al., 2023). Early systems typically treated past interactions as flat vector indices, retrieving historical notes based on semantic similarity and injecting them directly into the prompt context (Packer et al., 2023). While highly effective for semantic retrieval, this approach fails to represent how distinct facts interrelate: a state change in one memory note cannot automatically trigger updates in dependent notes. To overcome this limitation, recent agentic memory systems, such as A-Mem (Xu et al., 2025), introduce a dynamic memory graph actively managed by the LLM. By explicitly linking related notes, propagating updates to neighboring concepts, and querying the graph at retrieval time, these systems represent a crucial step forward in constructing relational structures over extended conversation histories.

However, existing frameworks operate under a strictly **shallow** topological constraint, typically restricting graph traversals to a single hop. This leaves two critical challenges unresolved. At write time, when a fact evolves, the LLM updates

only the explicitly retrieved note and its immediate neighbors, leaving dependent notes two or more hops away permanently stale, despite being implicitly conditioned on the change. At read time, the heavy reliance on direct query-to-note similarity under-exploits the graph's topology, as it floods the context window with tangential immediate neighbors while neglecting deeper nodes with mediated relevance, ultimately failing to capture the multi-hop dependencies necessary for coherent long-term memory.

Figure 1 illustrates the failure mode of shallow systems. Consider a scenario where Jon shifts his career from banking to managing a dance studio: while the job note m_1 and its direct neighbor m_2 (his schedule) are successfully updated, note m_3, which records that Jon is only available at weekends, remains untouched two hops away. Although its underlying fact is inherently affected by the career change, m_3 is bypassed by both single-hop write-time update and direct read-time retrieval. As a result, any subsequent query about Jon's current availability will draw upon the stale information in m_3, ultimately leading to a incorrect agent decision.

In this paper, we present **DeepMem**, a framework designed to execute **deep memory passing** over both the write-time and read-time dimensions. This core concept draws its inspiration from the multi-layer message passing paradigm established in Graph Neural Networks (GNNs) (Wu et al., 2020). Specifically, DeepMem consists of two core components (see Figure 2): *Deep Memory Update* and *Deep Memory Retrieval*. *Deep Memory Update* conducts long-range knowledge propagation by diffusing an LLM-generated change summary along graph edges via delta-anchored scoring (§3.2). When a new fact arrives, this mechanism successfully anchors on the change embedding to recursively update semantically dependent notes up to certain depths. Symmetrically, *Deep Memory Retrieval* propagates query relevance by utilizing the initial top-k semantic neighbors as bridging proxies; consequently, critical nodes beyond the immediate retrieval horizon can dynamically inherit relevance via their structural connections to these top-ranked seed notes (§3.3).

To sum up, our contributions are as follows:

- To the best of our knowledge, we are the first to conceptualize and advocate for deep memory passing in LLM agents, breaking the rigid single-hop limitations of existing agentic memory systems.

- We design and implement DeepMem via two symbiotic components: a write-time *Deep Memory Update* that propagates state-changes recursively, and a read-time *Deep Memory Retrieval* that diffuses query relevance through structural proxies.

- We conduct comprehensive evaluations on the challenging LoCoMo benchmark. Our results demonstrate that DeepMem significantly outperforms the state-of-the-art agentic memory baseline A-Mem, achieving a substantial weighted overall F1 improvement of $+15.7\%$ on Qwen2.5-1.5B-Instruct and $+58.0\%$ on Qwen2.5-3B-Instruct (§4).

2 Related Work

2.1 Long-term Conversational Memory

Persistent memory for LLM agents is most commonly realised as an external store of atomic notes retrieved by query similarity and injected into the prompt (Lewis et al., 2020; Wang et al., 2023). MemoryBank (Zhong et al., 2024) logs conversation events and user traits in a vector database, refreshing entries on a forgetting-curve schedule. ReadAgent (Lee et al., 2024) compresses long contexts into gist memories to reduce retrieval noise. MemGPT (Packer et al., 2023) introduces OS-style paging between a working-memory buffer and a persistent store. MemoryOS (Kang et al., 2025) formalises this into a four-module hierarchy with heat-based cross-tier migration. HiGMem (Cao et al., 2026) organises memories into a two-level event-turn hierarchy; at retrieval time, event summaries serve as semantic anchors for an LLM to predict which linked turns are worth reading, achieving high precision with a compact evidence set. Despite their differences in storage design, all these systems lack a propagation layer: when an upstream fact changes, updates apply only to the directly-modified entry, leaving dependent notes unaware of the change. DeepMem inherits this retrieve-and-inject foundation while adding explicit propagation to keep the store consistent as facts evolve.

2.2 Agentic Memory with Graphs

A-MEM (Xu et al., 2025) moves beyond passive indexing by letting an LLM editor manage a self-organising note graph: a `strengthen` action adds a directed edge, while `update_neighbor` rewrites the context and tags of directly retrieved neighbouring notes. Despite this richer design, A-MEM propagates changes only to directly retrieved neighbours in a single step, without cascading updates to multi-hop dependents; query-time retrieval relies on flat top-k similarity and then unconditionally expands immediate graph neighbours, admitting all linked nodes regardless of their relevance to the query. Beyond A-MEM, a growing body of concurrent work refines graph memory design along three axes. **Structural quality:** GAAMA (Paul et al., 2026) uses four node types—episode, fact, reflection, and concept—to avoid mega-hub effects; GAM (Wu et al., 2026) decouples a volatile interaction graph from a stable knowledge graph to prevent noise propagation. **Write-time selectivity:** Song and Xin (2026) gate writes by a dopamine-inspired reward-prediction-error score; All-Mem (Lv et al., 2026) applies offline SPLIT, MERGE, and UPDATE operators to reorganise topology non-destructively. **Long-horizon adaptation:** You et al. (2026) retain fast vector retrieval for routine queries and escalate to a full-deliberation fallback when quality gates flag insufficient evidence; APEX-MEM (Banerjee et al., 2026) structures facts as temporally grounded events with append-only storage. None of these systems propagates the consequences of an update beyond the directly-modified note, leaving write-time consistency across multi-hop neighbours unaddressed. DeepMem fills this gap with Deep Memory Update that extends updates to all semantically implied dependents, guided by the change embedding rather than proximity to the source node.

2.3 Graph-based Retrieval

Graph structure has also been applied at retrieval time. GraphRAG (Edge et al., 2024) and G-Retriever (He et al., 2024) leverage community summaries and subgraph selection respectively, but both assume a static, externally curated graph incompatible with the dynamic memory graphs we consider. HeLa-Mem (Zhu et al., 2026) propagates retrieval signals across Hebbian-reinforced links via spreading activation: high-scoring seed nodes spread their base scores to neighbours weighted by learned co-activation edge weights, potentially surfacing historically associated but semantically distant memories. However, Hebbian edge weights must accumulate through repeated co-activation across conversation history, creating a cold-start limitation where associative retrieval provides little benefit in early stages before sufficient co-occurrences have been observed. DeepMem's Deep Memory Retrieval uses seed-proxy scoring on pre-computed embeddings and requires no training history, enabling effective graph traversal from the first conversation turn; furthermore, seed-proxy scoring propagates through multiple hops, whereas HeLa-Mem's spreading activation propagates only one hop from the initial seeds.

3 DeepMem

DeepMem extends the agentic memory paradigm by equipping the memory graph with two cooperating mechanisms. At write time, *Deep Memory Update* propagates the consequences of a factual change across multiple hops, guided by the change itself rather than by similarity to the source note. At read time, *Deep Memory Retrieval* walks the graph outward from the seed top-k, routing query relevance through graph edges to nodes that are reachable but not directly query-similar. Figure 2 gives an overview.

3.1 Memory Construction

Following A-Mem (Xu et al., 2025), graph memory is maintained as a set $\mathcal{M} = \{m_1, \ldots, m_N\}$ of notes . Each note is represented as:

$$m_j = \{c_j,\ t_j,\ K_j,\ G_j,\ X_j,\ e_j,\ L_j\}, \qquad (1)$$

where c_j is the raw interaction content, t_j denotes the timestamp, K_j represents LLM-extracted keywords, G_j refers to categorical tags, X_j is a one-sentence contextual summary, $e_j \in \mathbb{R}^h$ computes a dense embedding, and $L_j \subseteq [N]$ returns an out-edge list of linked notes.

When a new observation (c_n, t_n) arrives, a new note m_n is constructed by extracting semantic metadata with a *construction prompt* P_{con}:

$$(K_n,\ G_n,\ X_n) \leftarrow \texttt{LLM}(c_n \parallel t_n \parallel P_{\text{con}}). \qquad (2)$$

The note embedding is then computed over all textual fields:

$$e_n = f_{\text{enc}}\big[\texttt{cat}(c_n,\ K_n,\ G_n,\ X_n)\big]. \qquad (3)$$

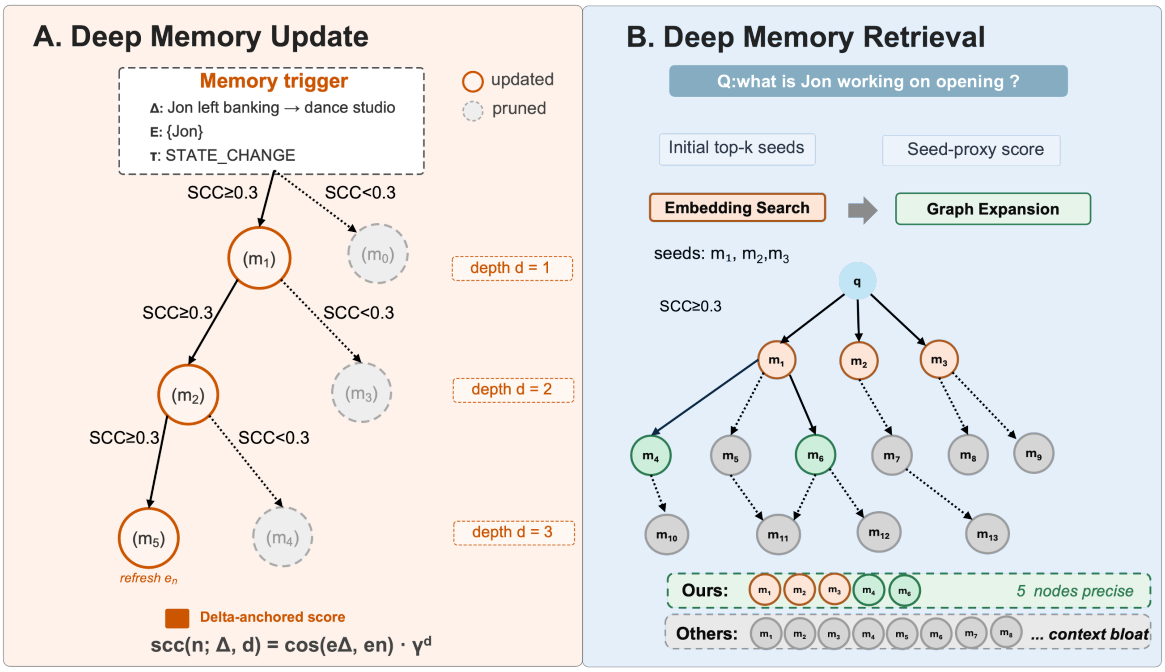

Figure 2: Overview of DeepMem's two mechanisms. **(A) Deep Memory Update.** When the LLM editor detects a state change (Δ: *Jon left banking* $\rightarrow$ *dance studio*; E: {Jon}; τ: STATE_CHANGE), it triggers propagation from the directly-updated node. Solid arrows indicate updates ($s_{\text{cc}} \geq 0.3$); dashed arrows indicate pruned candidates ($s_{\text{cc}} < 0.3$), where s_{cc} is the delta-anchored score. Propagation proceeds to depth $d = 1, 2, 3$; embeddings e_n are refreshed after each update. Scoring: $s_{\text{cc}}(n; \Delta, d) = \cos(e_\Delta, e_n) \cdot \gamma^d$. **(B) Deep Memory Retrieval.** Given query q (e.g. "What is Jon working on opening?"), embedding search retrieves initial top-k seeds $\{m_1, m_2, m_3\}$. A layered BFS then propagates seed relevance along graph edges via the seed-proxy score, surfacing m_4 and m_6 while pruning low-score neighbours. The final context (5 nodes, precise) contrasts with A-MEM's unconditional expansion (context bloat).

This joint encoding ensures that retrieval is sensitive to both the raw content and the LLM-generated semantic enrichment.

3.2 Deep Memory Update

We score each existing note $m_j \in \mathcal{M}$ by cosine similarity to the new note:

$$s_{n,j} = \frac{e_n \cdot e_j}{\|e_n\| \|e_j\|}, \tag{4}$$

and retrieve the top-k most similar as the neighbour candidate set:

$$\mathcal{M}_n = \{m_j \mid \text{rank}(s_{n,j}) \leq k,\ m_j \in \mathcal{M}\}. \tag{5}$$

For each $m_i \in \mathcal{M}_n$, a single LLM call with an *evolution prompt* P_{evo} then jointly handles link formation and content updates:

$$(L_n,\ m'_j,\ \Delta,\ E,\ \tau) \leftarrow \texttt{LLM}(m_n \parallel m_j \parallel P_{\text{evo}}). \tag{6}$$

L_n denotes the link set for m_n; m'_j are the immediately-updated neighbours with rewritten (X_j, G_j); Δ is a one-sentence *delta summary* describing what changed (e.g. "Jon left banking to start a dance studio"); E is a set of 1–2 *primary entities*; and τ is the *change type*, implemented as STATE_CHANGE, CORRECTION, TEMPORAL_UPDATE, or NEW_FACT. The LLM may execute one or both of two actions: `strengthen` appends suggested neighbour indices to L_n, forming graph edges; `update_neighbor` rewrites (X_j, G_j) of selected neighbours when $\tau \neq$ NEW_FACT. When $\tau =$ NEW_FACT, only `strengthen` is executed and $\Delta = \emptyset$ in this case.

When $\Delta \neq \emptyset$, the change propagates beyond the first hop to reach all semantically implied dependents. Let $U_0 = \{j \mid m_j$ was updated in Eq. 6$\}$ be the set of first-hop updated nodes. For each topological depth $d = 1, \ldots, D_{\max}$, we identify the set of unique, non-repetitive adjacent nodes v for each vertex $u \in U_{d-1}$. These candidate nodes are subsequently evaluated via the *delta-anchored score*:

$$s_{\text{cc}}(v;\ \Delta,\ d) = \cos\!\big(e_\Delta,\ e_v\big) \cdot \gamma^d, \tag{7}$$

where $e_\Delta = f_{\text{enc}}(\Delta)$ is the delta embedding of note u and $\gamma = 0.7$ is a depth-decay factor. Candi-

Algorithm 1 Deep Memory Update

Require: Updated seeds U_0, delta summary Δ, primary entities E, depth limit $D_{\max}$

$e_\Delta \leftarrow \texttt{encode}(\Delta)$; $\mathcal{V} \leftarrow U_0$; $d \leftarrow 1$
while $d \le D_{\max}$ **and** $U_{d-1} \neq \emptyset$ **do**
  cands $\leftarrow \{v \mid u \in U_{d-1}, v \in L_u, v \notin \mathcal{V}, v \text{ matches } E\}$
  $C \leftarrow \{(v, s_{\text{cc}}(v; \Delta, d)) \mid v \in \text{cands}, s_{\text{cc}} \ge \tau_{\min}\}$
  $C \leftarrow$ top-B of C by score
  $U_d \leftarrow \emptyset$
  for $(v, \cdot) \in C$ **do**
    $\mathit{flag}, X'_v, G'_v \leftarrow \texttt{LLM_judge}(\Delta, m_v)$
    if flag **then**
      $X_v \leftarrow X'_v$; $G_v \leftarrow G'_v$; refresh e_v
      $U_d \leftarrow U_d \cup \{v\}$; $\mathcal{V} \leftarrow \mathcal{V} \cup \{v\}$
    end if
  end for
  $d \leftarrow d + 1$
end while

dates additionally pass an entity pre-filter (content of v must mention at least one entity from E) and a score threshold ($s_{\text{cc}} \ge \tau_{\min} = 0.3$); per layer we retain at most the top-B candidates ($B = 3$). For each surviving candidate v, a secondary LLM call verifies whether the update is applicable:

$$(\mathit{flag},\ X'_v,\ G'_v) \leftarrow \texttt{LLM_judge}(\Delta,\ c_v,\ X_v,\ G_v). \tag{8}$$

If *flag* is set, we overwrite $X_v \leftarrow X'_v$ and $G_v \leftarrow G'_v$, and immediately re-encode e_v, so that subsequent propagation steps operate on the updated representation rather than waiting for the periodic `consolidate` pass. All updated nodes at depth d form U_d, the seed set for the next layer. Algorithm 1 summarises the full procedure.

The key insight is to anchor the propagation score on the *change embedding* e_Δ rather than on the source note: notes that are semantically implied by the change are reached even when they are lexically distant from the triggering note.

Complexity. DeepMem adds at most $D_{\max} \times B$ LLM calls per triggered evolution step; when $\Delta = \emptyset$ or $\tau = \text{NEW_FACT}$, the overhead is zero.

3.3 Deep Memory Retrieval

At read time, the query q is embedded as $e_q = f_{\text{enc}}(q)$, and each note is scored by cosine similarity:

$$s(q, m_j) = \cos(e_q,\ e_j). \tag{9}$$

Top-k notes are retrieved as the initial seed set:

$$M^{\text{ret}} = \{m_j \mid \text{rank}(s(q, m_j)) \le k,\ m_j \in \mathcal{M}\}. \tag{10}$$

A-MEM expands M^{ret} by unconditionally admitting all immediate graph neighbours, regardless of their relevance to the query; this floods the context with tangential entries while leaving nodes beyond one hop unreachable. Adding a depth-discounted cosine score $\cos(e_q, e_v) \cdot \gamma^d$ does not resolve the deeper problem: any node absent from the seed top-k already has a lower direct similarity to q, which γ^d only attenuates further.

Algorithm 2 Deep Memory Retrieval

Require: Query embedding e_q, seeds S, depth $D_{\max}$, decay γ, threshold τ

$score[s] \leftarrow \cos(e_q, e_s)$ for $s \in S$; $\mathcal{V} \leftarrow S$; $prev \leftarrow S$
for $d = 1, \ldots, D_{\max}$ **do**
  $next \leftarrow \{\}$
  for $u \in prev$; $v \in L_u \setminus \mathcal{V}$ **do**
    $s' \leftarrow score[u] \cdot \cos(e_u, e_v) \cdot \gamma$
    if $s' > next.\text{get}(v, 0)$ **then** $next[v] \leftarrow s'$
    end if
  end for
  $score \leftarrow score \cup next$; $\mathcal{V} \leftarrow \mathcal{V} \cup next.\text{keys}()$; $prev \leftarrow next.\text{keys}()$
end for
return $\{v : score[v] \ge \tau\}$ sorted by $score[v]$

We avoid this circularity by routing relevance through seed nodes as proxies. For a candidate node v reached from seed s via path $p = (u_0 = s, u_1, \ldots, u_{|p|} = v)$, let $w(p) = \prod_{(u,u') \in p} \cos(e_u, e_{u'})$ be the edge-weight product along p. The score of v is the maximum propagated relevance over all paths from any seed:

$$s_{\text{rc}}(v) = \max_{p \in \mathcal{P}(s \to v)} \cos(e_q,\ e_s) \cdot w(p) \cdot \gamma^{|p|}, \tag{11}$$

where $\mathcal{P}(s \to v)$ denotes all paths of length at most $D_{\max}$ from any seed s to v. A node strongly connected to a query-relevant seed inherits high relevance; a node loosely connected or far from relevant seeds does not.

We implement this via a layered BFS that avoids the $\mathcal{O}(N^2)$ cost of naive all-pairs expansion; the full procedure is given in Algorithm 2.

The final context is $M^{\text{ret}} \cup \{v : s_{\text{rc}}(v) \ge \tau\}$. Since all scores are computed from precomputed embeddings, the graph expansion requires no additional LLM calls and runs in $O(|E| \cdot D_{\max})$ time. The two mechanisms are independent and can be enabled or disabled separately for ablation (§4.3).

4 Experiments

4.1 Experimental Settings

Benchmark. We evaluate on **LoCoMo** (Maharana et al., 2024), a long-term conversational QA benchmark consisting of 10 multi-session dialogues totalling 1,986 questions across five cate-

Table 1: Main results on LoCoMo (%; F1 and BLEU-1 across four question categories). Baseline numbers are taken from Xu et al. (2025) (LoCoMo, ReadAgent, MemoryBank, MemGPT, A-MEM) and Zhu et al. (2026) (MemoryOS, HeLa-Mem). †Our faithful reproduction. Avg Rk = average rank across four categories (↓ lower is better). **Bold** = best per column; underline = second best. Shaded row = our method. Full results including ADVERSARIAL and context length are in Table 6 (Appendix **??**).

Method	MultiHop F1	MultiHop B-1	Temporal F1	Temporal B-1	OpenDom. F1	OpenDom. B-1	SingleHop F1	SingleHop B-1	Avg Rk↓ F1	Avg Rk↓ B-1
Qwen2.5-1.5B-Instruct										
LoCoMo (Maharana et al., 2024)	9.05	6.55	4.25	4.04	9.91	8.50	11.15	8.67	5.3	5.3
ReadAgent (Lee et al., 2024)	6.61	4.93	2.55	2.51	5.31	12.24	10.13	7.54	6.8	5.8
MemoryBank (Zhong et al., 2024)	11.14	8.25	4.46	2.87	8.05	6.21	13.42	11.01	4.5	5.3
MemGPT (Packer et al., 2023)	10.44	7.61	4.21	3.89	13.42	11.64	9.56	7.34	5.5	5.3
A-MEM (Xu et al., 2025)	18.23	11.94	**24.32**	**19.74**	**16.48**	**14.31**	23.63	19.23	2.0	**2.0**
A-MEM† (Xu et al., 2025)	19.29	12.29	21.58	15.98	15.27	13.30	23.93	19.53	2.3	2.3
DeepMem	**21.06**	**13.09**	23.40	18.88	14.92	10.94	**29.70**	**25.00**	**1.8**	2.3
Qwen2.5-3B-Instruct										
LoCoMo (Maharana et al., 2024)	4.61	4.29	3.11	2.71	4.55	5.97	7.03	5.69	7.3	7.0
ReadAgent (Lee et al., 2024)	2.47	1.78	3.01	3.01	5.57	5.22	3.25	2.51	8.3	8.3
MemoryBank (Zhong et al., 2024)	3.60	3.39	1.72	1.97	6.63	6.58	4.11	3.32	7.8	7.8
MemGPT (Packer et al., 2023)	5.07	4.31	2.94	2.95	7.04	7.10	7.26	5.52	6.3	6.3
A-MEM (Xu et al., 2025)	12.57	9.01	27.59	25.07	7.12	7.28	17.23	13.12	3.8	3.8
A-MEM† (Xu et al., 2025)	14.45	10.13	24.37	20.60	6.18	5.70	16.13	11.43	5.0	5.3
MemoryOS (Kang et al., 2025)	19.20	**14.84**	20.85	16.05	13.57	10.86	25.65	18.78	3.3	2.8
HeLa-Mem (Zhu et al., 2026)	20.12	14.59	24.79	21.35	12.24	10.24	29.51	25.91	2.5	2.5
DeepMem	**21.44**	14.55	**35.56**	**28.20**	**14.16**	**12.75**	**35.39**	**29.04**	**1.0**	**1.5**

gories (Table 2). ADVERSARIAL questions intentionally contain a wrong premise; the gold answer asserts that the premise is unsupported by the conversation.

Table 2: Distribution of question types in the LoCoMo evaluation set (10 multi-session dialogues, 1,986 questions).

Category	Count	Percentage
SINGLEHOP	841	42.3%
ADVERSARIAL	446	22.4%
TEMPORAL	321	16.1%
MULTIHOP	282	14.2%
OPENDOMAIN	96	4.8%
Total	**1,986**	**100.0%**

Baselines. We compare against seven baselines. **LoCoMo** (Maharana et al., 2024), **ReadAgent** (Lee et al., 2024), **MemoryBank** (Zhong et al., 2024), **MemGPT** (Packer et al., 2023), and **A-MEM** (Xu et al., 2025) are five standard baselines; for A-MEM we report both the original paper numbers and our faithful reproduction. We additionally compare against two recent strong baselines on Qwen2.5-3B-Instruct: **MemoryOS** (Kang et al., 2025) frames agent memory management as an operating-system problem, organising memories into a hierarchical segment-page structure that supports dynamic insertion, eviction, and retrieval. **HeLa-Mem** (Zhu et al., 2026) is a bio-inspired architecture that models conversation history as a dynamic Hebbian graph; co-activated memories strengthen their connections over time, and a dual-path retrieval strategy combines spreading activation over Hebbian edges with semantic similarity search.

Backbones. We use Qwen2.5-1.5B-Instruct and Qwen2.5-3B-Instruct (Qwen Team, 2024) as the LLM editor/answerer. Models are served locally with vLLM v0.20.2 (Kwon et al., 2023) on a single H20 GPU. We deliberately use small open-weight backbones to stress the system: agentic-memory pipelines that depend on perfect editor behaviour are exactly the regime where mid-sized open models are weakest.

Implementation. The retriever uses Sentence-BERT (Reimers and Gurevych, 2019) embeddings (`all-MiniLM-L6-v2`) with cosine similarity for both indexing and scoring. We set a unified propagation depth $D_{\max} = 3$ governing both memory passing and memory retrieval. For Deep Memory Update we set depth-decay $\gamma = 0.7$, per-layer budget $B = 3$, and candidate threshold $\tau_{\min} = 0.3$. For Deep Memory Retrieval we set $\gamma = 0.85$ and expansion threshold $\tau = 0.3$.

Table 3: Ablation on LoCoMo (F1 %; Qwen2.5-3B-Instruct; MultiHop, SingleHop, Adversarial). Overall gain vs. A-MEM: +12.52. Shaded = full system.

Method	Multi.	Single.	Adv.	All
DeepMem	**21.44**	**35.39**	**32.83**	**31.83**
w/o DM Update	15.31	26.18	28.74	23.38
w/o DM Retrieval	18.10	30.32	30.57	25.83
w/o Both	14.45	16.13	27.58	19.31

Top-k at retrieval is 10 for all categories. All LLM calls (memory construction, neighbour update judging, and answer generation) use temperature 0.7; ADVERSARIAL answer generation uses $T = 0.5$ following A-MEM. Each LoCoMo conversation is processed once and its memory cache reused across all configurations of the same memory-construction setting.

Metrics. We report token-level F1 and BLEU-1 (Papineni et al., 2002), both computed on the lowercased answer/prediction strings, averaged within each category and then macro-averaged across questions to obtain OVERALL.

4.2 Main Results

Table 1 reports results across four question categories on both backbones; full results including ADVERSARIAL are in Table 6 (Appendix **??**). On Qwen2.5-3B-Instruct, DeepMem leads all methods on every category. The gains are largest on SINGLEHOP (35.39 F1, +5.88 over HeLa-Mem and +19.26 over A-MEM†) and TEMPORAL (35.56 F1, +10.77 over HeLa-Mem), followed by MULTIHOP (+1.32 over HeLa-Mem) and OPENDOMAIN (+0.59 over MemoryOS). On Qwen2.5-1.5B-Instruct, DeepMem leads on SINGLEHOP (29.70, +5.77 over A-MEM†) and MULTIHOP (21.06, +1.77); it trails A-MEM (paper) on TEMPORAL (23.40 vs. 24.32) and OPENDOMAIN (14.92 vs. 16.48), consistent with the smaller model's limited ability to follow multi-step temporal reasoning prompts.

On ADVERSARIAL (Table 6 in Appendix), DeepMem leads all methods (53.04 F1 on 1.5B; 32.83 on 3B): keeping retrieved facts current is directly necessary for resisting wrong-premise questions. The Avg Rk column shows that on 3B, DeepMem holds the top F1 rank on every category simultaneously.

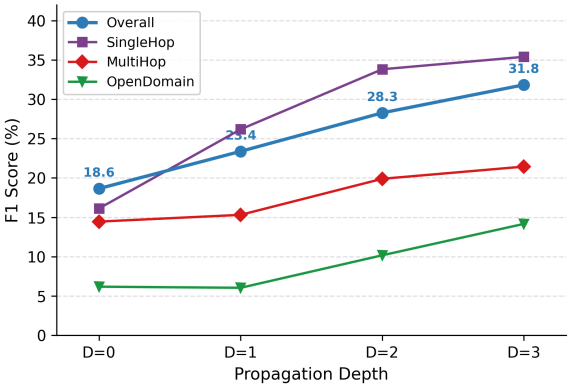

Figure 3: Effect of $D_{\max}$ on LoCoMo (Qwen2.5-3B-Instruct). $D_{\max}$ uniformly controls propagation depth for both DM Update and DM Retrieval. $D_{\max}$=0: no mechanism; $D_{\max}$=3: full DeepMem. Overall F1 rises monotonically.

4.3 Ablation Study

Effect of Each Component. To isolate the contribution of each mechanism, we disable them individually on the Qwen2.5-3B-Instruct backbone over the full LoCoMo benchmark (1,986 questions). As shown in Table 3, removing Deep Memory Update (*w/o DM Update*) reduces overall F1 to 23.38, barely above the no-mechanism baseline (19.31). Removing Deep Memory Retrieval (*w/o DM Retrieval*) yields 25.83, well below the full system (31.83). Neither mechanism reaches its full potential in isolation.

Effect of Propagation Depth. We sweep $D_{\max} \in \{0, 1, 2, 3\}$, which uniformly controls the propagation depth of both DM Update and DM Retrieval (Figure 3). Overall F1 grows monotonically (18.65 $\rightarrow$ 23.38 $\rightarrow$ 28.27 $\rightarrow$ 31.83); the jump from $D_{\max}$=0 to $D_{\max}$=1 (+4.73) confirms that even one hop of joint propagation substantially enriches the memory graph, while the continued gain from $D_{\max}$=1 to $D_{\max}$=2 (+4.89) shows that deeper traversal yields further improvement. SINGLEHOP is the largest beneficiary (+16.86 across $D_{\max}$=0$\rightarrow$3). TEMPORAL (35.56) and OPENDOMAIN (14.16) both reach their peak at $D_{\max}$=3, confirming that deeper graph propagation is essential for date-anchored and broad-domain evidence retrieval.

4.4 Analysis

Token Efficiency. Table 4 shows that DeepMem maintains competitive context efficiency despite enabling multi-hop graph traversal. On 1.5B,

DeepMem feeds 1,156 tokens per question to the answer LLM—below A-MEM (1,300) and comparable to HeLa-Mem (1,072 on 3B). On 3B, DeepMem uses 1,524 tokens, modestly above A-MEM† (1,137) due to graph expansion, yet still orders of magnitude below full-context methods such as LoCoMo (16,910) and MemGPT (16,953). The retrieval threshold $\tau = 0.3$ gates neighbour inclusion: only nodes whose propagated relevance score clears the threshold are added to context, preventing unbounded growth.

Table 4: Average context tokens fed to the answer LLM per question (lower is better). DeepMem uses selective graph expansion ($\tau = 0.3$) rather than unconditional neighbour expansion, keeping context compact while enabling multi-hop retrieval.

Method	1.5B	3B
LoCoMo (Maharana et al., 2024)	16,910	16,910
MemGPT (Packer et al., 2023)	16,953	16,961
ReadAgent (Lee et al., 2024)	752	776
A-MEM (Xu et al., 2025)	1,300	1,137
MemoryOS (Kang et al., 2025)	–	2,000
HeLa-Mem (Zhu et al., 2026)	–	1,072
MemoryBank (Zhong et al., 2024)	284	298
DeepMem	1,156	1,524

Deep Memory Update Activation. Table 5 summarises activation statistics collected on the full LoCoMo benchmark (Qwen2.5-1.5B-Instruct, 10 conversations, 2,812 evolution steps). Deep Memory Update fires selectively: only 21.8% of evolution steps trigger at least one neighbour update. Among triggered events, the system updates on average 1.48 neighbours per event, with a mean delta-anchored score of 0.354. This selectivity is crucial—indiscriminate propagation would contaminate unrelated memories with spurious context.

Update Distribution. Figure 4 shows the distribution of neighbour updates per propagation event. The distribution is heavily zero-inflated (78.2% of events produce no update at all), confirming that the delta-anchored scoring sets a high bar for propagation. Among triggered events, most update only 1–3 neighbours per event, consistent with the per-layer budget $B = 3$, which prevents blow-up while allowing meaningful multi-hop propagation when the change signal is strong enough.

Table 5: Deep Memory Update event statistics on LoCoMo (Qwen2.5-1.5B-Instruct, 10 conversations). DM Update fires on 21.8% of steps; when triggered, it updates 1.48 neighbours per event on average, with mean delta-anchored score 0.354 ($\tau_{\min} = 0.3$).

Statistic	Value
Total evolution steps	2,812
DM Update triggered ($\geq$1 update)	612 (21.8%)
Total neighbour updates	903
Avg updates per triggered event	1.48
Avg propagation score	0.354

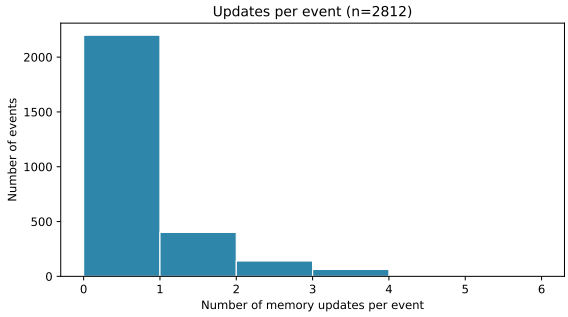

Figure 4: Distribution of neighbour updates per propagation event ($n = 2{,}812$ events, Qwen2.5-1.5B-Instruct). Most events (78.2%) produce zero updates; among triggered events, most update only 1–3 neighbours per event, confirming the per-layer budget $B = 3$ prevents blow-up.

5 Conclusion

In this paper, we presented DeepMem, a graph-based agentic memory system built upon a critical observation: while existing architectures construct memory graphs, their relational structures remain fundamentally under-exploited due to rigid single-hop constraints. DeepMem closes this structural gap through the synergy of two complementary mechanisms. At write time, Deep Memory Update traces the cascading consequences of factual changes via delta-anchored scoring, seamlessly propagating updates to implicitly affected dependent nodes across multi-hop topological depths. Symmetrically, at read time, Deep Memory Retrieval leverages initial top-k semantic matches as structural proxies to diffuse query relevance along graph pathways, successfully surfacing critical, deeper evidence that direct similarity metrics fail to reach. Empirical evaluations on the challenging LoCoMo benchmark validate the efficacy of our method. Ultimately, our study suggest that deep propagation should be considered in the design of next-generation agentic memory systems.

Limitations

While DeepMem demonstrates consistent gains in multi-hop memory consistency, we acknowledge several areas for future exploration. Although the delta-anchored scoring mechanism is designed to be model-agnostic, the quality of write-time propagation still depends on the LLM editor's ability to recognise factual changes accurately; on smaller backbones, occasional misclassification of change types can lead to missed propagation. Additionally, the memory graph currently grows without any forgetting mechanism, which may affect retrieval quality in very long conversations where stale nodes accumulate. Finally, our evaluation is conducted on a single English benchmark, and extending DeepMem to multilingual or multi-domain settings remains an open direction for future work.

References

Pratyay Banerjee, Masud Moshtaghi, Shivashankar Subramanian, Amita Misra, and Ankit Chadha. 2026. APEX-MEM: Agentic semi-structured memory with temporal reasoning for long-term conversational AI. *arXiv preprint arXiv:2604.14362*. ACL 2026.

Shuqi Cao, Jingyi He, and Fei Tan. 2026. HiGMem: A hierarchical and LLM-guided memory system for long-term conversational agents. *arXiv preprint arXiv:2604.18349*. ACL 2026 Findings.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems (NeurIPS)*.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory OS of AI agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. 2024. A human-inspired reading agent with gist memory of very long contexts. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems (NeurIPS)*.

Can Lv, Heng Chang, Yuchen Guo, Shengyu Tao, and Shiji Zhou. 2026. All-Mem: Agentic lifelong memory via dynamic topology evolution. *arXiv preprint arXiv:2603.19595*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Swarna Kamal Paul, Shubhendu Sharma, and Nitin Sareen. 2026. GAAMA: Graph augmented associative memory for agents. *arXiv preprint arXiv:2603.27910*.

Qwen Team. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yuru Song and Qi Xin. 2026. D-MEM: Dopamine-gated agentic memory via reward prediction error routing. *arXiv preprint arXiv:2603.14597*.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhaofen Wu, Hanrong Zhang, Fulin Lin, Hongwei Wang, and 1 others. 2026. GAM: Hierarchical graph-based agentic memory for LLM agents. *arXiv preprint arXiv:2604.12285*.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-MEM: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhixing You, Jiachen Yuan, and Jason Cai. 2026. D-Mem: A dual-process memory system for LLM agents. *arXiv preprint arXiv:2603.18631*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. MemoryBank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Jinchang Zhu, Jindong Li, Cheng Zhang, Jiahong Liu, and Menglin Yang. 2026. HeLa-Mem: Hebbian learning and associative memory for LLM agents. *arXiv preprint arXiv:2604.16839*. ACL 2026.

Appendix

Deep Memory Update relies on two LLM prompts: (1) the *evolution prompt* that extracts the delta summary, primary entities, and change type during memory construction, and (2) the *cascade judge prompt* that decides whether each candidate neighbour should be updated.

Evolution Prompt (Delta Extraction)

```
You are an AI memory evolution agent
responsible for managing and evolving a
knowledge base. Analyze the new memory note
according to keywords and context, also with
their nearest neighbors.

The new memory context: {context}
content: {content}
keywords: {keywords}

The nearest neighbors memories:
{nearest_neighbors_memories}

Determine:
1. Should this memory be evolved?
2. Actions (strengthen is ALWAYS required):
- strengthen: provide suggested_connections
as 0-based indices of related neighbors.
- update_neighbor: include ONLY when the new
memory changes the meaning/state of existing
neighbors.
ACTION-CHANGE_TYPE CONSISTENCY RULE:
If change_type is "state_change",
"correction", or "temporal_update", actions
MUST be ["strengthen", "update_neighbor"]
AND delta_summary MUST be non-empty.
3. Cascade propagation fields (always
required):
- delta_summary: ONE sentence describing
WHAT CHANGED. Empty string if change_type is
new_fact.
- primary_entities: 1-2 entity names that are
the subject of the change.
- change_type: exactly one of: state_change
| correction | temporal_update | new_fact

Return JSON:
{"should_evolve": bool, "actions": [...],
"suggested_connections": [int, ...],
"delta_summary": "...", "primary_entities":
[...], "change_type": "..."}
```

Cascade Judge Prompt (Neighbour Update)

```
A factual change has occurred:
{delta_summary}

Review this related memory:
Content : {target_content}
Context : {target_context}
Tags    : {target_tags}

Does this memory need updating to reflect the
change?
- If YES: provide a revised one-sentence
context and updated tags.
- If NO: set should_update=false.

Return JSON:
{"should_update": bool, "new_context": "...",
"new_tags": [...]}
```

Table 6: Full results on LoCoMo (%; F1 and BLEU-1 across all five question categories). †Our faithful reproduction. Baseline numbers are taken from Xu et al. (2025) and Zhu et al. (2026) (MemoryOS and HeLa-Mem on Qwen2.5-3B-Instruct only; Adversarial not reported). **Bold** = best per column; underline = second best. Shaded row = our method.

Method	MultiHop		Temporal		OpenDom.		SingleHop		Adversarial	
	F1	B-1	F1	B-1	F1	B-1	F1	B-1	F1	B-1
Qwen2.5-1.5B-Instruct										
LoCoMo	9.05	6.55	4.25	4.04	9.91	8.50	11.15	8.67	40.38	40.23
ReadAgent	6.61	4.93	2.55	2.51	5.31	12.24	10.13	7.54	5.42	27.32
MemoryBank	11.14	8.25	4.46	2.87	8.05	6.21	13.42	11.01	36.76	34.00
MemGPT	10.44	7.61	4.21	3.89	13.42	11.64	9.56	7.34	31.51	28.90
A-MEM	18.23	11.94	**24.32**	**19.74**	**16.48**	**14.31**	23.63	19.23	46.00	43.26
A-MEM†	19.29	12.29	21.58	15.98	15.27	13.30	23.93	19.53	49.87	46.72
DeepMem	**21.06**	**13.09**	23.40	18.88	14.92	10.94	**29.70**	**25.00**	**53.04**	**49.69**
Qwen2.5-3B-Instruct										
LoCoMo	4.61	4.29	3.11	2.71	4.55	5.97	7.03	5.69	16.95	14.81
ReadAgent	2.47	1.78	3.01	3.01	5.57	5.22	3.25	2.51	15.78	14.01
MemoryBank	3.60	3.39	1.72	1.97	6.63	6.58	4.11	3.32	13.07	10.30
MemGPT	5.07	4.31	2.94	2.95	7.04	7.10	7.26	5.52	14.47	12.39
A-MEM	12.57	9.01	27.59	25.07	7.12	7.28	17.23	13.12	27.91	25.15
A-MEM†	14.45	10.13	24.37	20.60	6.18	5.70	16.13	11.43	27.58	25.30
MemoryOS	19.20	**14.84**	20.85	16.05	13.57	10.86	25.65	18.78	–	–
HeLa-Mem	20.12	14.59	24.79	21.35	12.24	10.24	29.51	25.91	–	–
DeepMem	**21.44**	14.55	**35.56**	**28.20**	**14.16**	**12.75**	**35.39**	**29.04**	**32.83**	**30.04**

Table 7: Representative Deep Memory Update examples (Qwen2.5-1.5B-Instruct on LoCoMo, 466 valid update events). Rows 1–8 are STATE_CHANGE events; rows 9–15 are TEMPORAL_UPDATE events. The delta-anchored score ($\tau_{\min} = 0.3$) ensures only semantically aligned neighbours pass the filter. Examples cover diverse update scenarios: job transition (rows 1–2), outdoor activity and social plans (rows 3–4, 8), dietary and health habits (rows 5, 14), professional impact (row 6), lifestyle experiences (row 7), gaming and hobbies (rows 9–10, 13), possessions (row 11), mindfulness (row 12), and incremental refinement of existing context (rows 14–15).

#	Delta Summary	Score	Neighbour Context — Before	Neighbour Context — After
state_change				
1	Andrew: Financial Analyst → Marketing Manager	**0.413**	*"I want to apply my programming skills to causes I care about."*	*"John recently volunteered at a non-profit, creating software tools that improved operations."*
2	Andrew's job transition from Financial Analyst to Marketing Manager	**0.397**	*"I started a new job as a Financial Analyst last week—it's been quite a change."*	*"I just got promoted to be a Marketing Manager. It's been an exciting journey."*
3	Speaker updated outdoor activity to include mountains and camping	**0.472**	*"We enjoyed an adventurous day combining skiing, snowshoeing, and hiking."*	*"Shared recent travels to explore different regions, including mountainous areas for hiking and camping."*
4	Sam planning trip to Rocky Mountains	**0.346**	*"I went up to the Rocky Mountains recently."*	*"Planning a trip to the Rocky Mountains; enjoys hiking and camping."*
5	Dietary changes toward more fruits and vegetables	**0.360**	*"I'm trying to eat less processed food and sugary snacks to be extra careful with my health."*	*"I've been making dietary changes, adding more fruits and vegetables, which is helping me feel better."*
6	Engineering skills used to improve education in underserved communities	**0.372**	*"I want to keep working in engineering and make a positive impact on communities in need."*	*"Your engineering skills truly made a significant impact in improving education and infrastructure in underserved areas."*
7	Explored multiple cafes and found the best pastries	**0.367**	*"We found an awesome cafe with amazing pastries—the smell was just irresistible!"*	*"We explored a variety of cafes and discovered the best pastries in town—they were heavenly!"*
8	Joining friends for a hike with pets	**0.412**	*"The new memory connects speakers discussing various parks and dog-walking experiences."*	*"The speaker reminisces about joining friends for a hike with their pets, highlighting the joy of enjoying nature together."*
temporal_update				
9	James recently acquired a new gaming system	**0.452**	*"James shares his excitement about playing games on the device."*	*"James recently acquired a new gaming system and enjoys spending time with friends."*
10	Recent gaming experiences and online community	**0.399**	*"Gaming can bring people together and create shared experiences."*	*"I recently joined an online multiplayer game that quickly became a source of community support."*
11	Speaker finally got a classic car and took it for a drive	**0.374**	*"I finally got myself this beauty. Here's a pic from yesterday when I took it for a ride."*	*"Finally got this beautiful classic car—can't believe I'm actually driving it."*
12	Mindfulness practices now actively reducing stress	**0.345**	*"It helps me make sense of everything and relieves stress. It's like a restart."*	*"It has helped me reduce stress and keep my mind centered."*
13	Painting recommended as a calming activity with new materials	**0.389**	*"I'm interested in trying painting. Could you help me pick out materials?"*	*"I'm interested in painting as a calming activity. Could you recommend some suitable materials?"*
14	Diet and exercise routine showing health improvements	**0.337**	*"I've been experiencing improvements in my overall well-being since implementing these changes."*	*"I recently started incorporating more fruits, vegetables, and whole grains into my diet to improve my health."*
15	Photography style updated to reflect sharper, more recent shots	**0.412**	*"Hey Dave, do you use this camera for photos? They always turn out so good!"*	*"Hey Dave, do you use this camera for photos? Their shots are always so sharp!"*