STARK: Moving Towards Personalized Agents That Truly Learn and Remember

Aryan Jain, Abhishek Jha*, Anish Nediyanchath, Promod Yenigalla

RBS Tech Sciences, Amazon
{arynjn, abhisjht, anishned, promy}@amazon.com

Abstract

Long-term memory is essential for personalized agentic systems that interact with users over long time periods, yet current architectures only capture what happened while leaving the reasoning of why it happened to be derived later by underlying LLMs.These events can, however, convey not only what a user experienced but also how that experience shaped their behavioral patterns. We introduce STARK, a dual-graph memory architecture which treats these events not just as nodes but also as edges between behavioral insights. STARK maintains an Episodic Graph which records interactions as temporally, causally, and semantically linked episodes; and an Insight Graph which represents LLM-extracted behavioral signals scored by how many events support them, connected by the very events that led to them. At inference, behavioral knowledge is retrieved directly from insight nodes grounded by event-as-edges, eliminating repeated episodic traversal. The event-as-edge structure also naturally enables projecting the knowledge graph into a relational embedding space where cross-user transfer becomes possible under shared contextual circumstances, allowing the system to infer missing insights for cold-start users. Evaluations show that STARK improves by 1.48 judge score on LongMemEval information extraction and 0.93 on LoCoMo open domain questions over state of the art. On the public LongLaMP abstract-generation benchmark, cross-user transfer improves open-ended judge scores by 0.7 points and ROUGE-L by 3 at 5% graph sparsity.

1 Introduction

As autonomous interactive agents are increasingly deployed to manage complex, long-running tasks such as workflow orchestration and support-ticket triage under evolving constraints (Wu et al. 2023; Wang et al. 2023; Jain, Ghosh, and Yenigalla 2026), their effectiveness relies on accumulating knowledge across long-horizon interactions. For example, for a deal negotiation agent, a compliance constraint surfaced during negotiation in one quarter must inform procurement strategy in the next; a failed negotiation strategy must not be repeated in the future. Moreover, many of these agents are domain-specific, wherein users pursue structurally similar goals and exhibit convergent behavioral patterns, making cross-user knowledge transfer a natural consequence of shared goal distributions.

Recent persistent-memory systems partially address this by constructing knowledge graphs from interactions (Rasmussen et al. 2025; Xu et al. 2025; Jiang et al. 2026), but maintain a single flat graph per user where events, preferences, constraints, and goals coexist as undifferentiated nodes. This design has two consequences. First, behavioral signals (the durable, reusable insights into how a user tends to act and decide, namely their preferences, constraints, goals, and values, distilled from many individual episodes and distinct from the episodes themselves) remain distributed across episodic nodes rather than being represented explicitly, which forces the LLM to re-derive them at query time, redundantly and often inconsistently (Richardson et al. 2023; Gutiérrez et al. 2024). Second, each user's graph is self-contained and provides no pathway for cross-user transfer. The common root cause is that these graphs capture *what happened* while leaving the reasoning behind *why it happened* to downstream inferencing.

We introduce **STARK**, a dual-graph memory architecture built around this distinction. Our core contributions are as follows:

1. A **dual-graph architecture** that separates episodic events from behavioral knowledge. It is composed of an **Episodic Graph** with temporally, causally and semantically linked episodes and an **Insight Graph** that contains behavioral signals, each carrying evidence vectors from the episodes that support it. The two are joined by an *event-as-edge bridge*: episodic identifiers serve as typed edge labels in the **Insight Graph**, keeping every insight grounded in the episodes that produced it.
2. A **cross-user transfer mechanism** for cold-start users. Insight nodes are embedded in a relational space where users with similar *preference fingerprints* cluster together, and link prediction over that space infers a new user's likely signals without accessing another user's raw interactions.
3. We deploy STARK for a production vendor-negotiation assistant and evaluate it in an **8-week A/B pilot** (N=40 managers) cutting negotiation rounds per deal by 18% and cycle time by 15% ($p < 0.05$).

2 Related Work

Memory-Augmented Generation: RAG (Lewis et al. 2020) treats knowledge as a static corpus, and write-back memories (MemoryBank (Zhong et al. 2024), Generative Agents (Park et al. 2023), MemGPT (Packer et al. 2024)) persist

*Work done during internship

Copyright © 2027, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

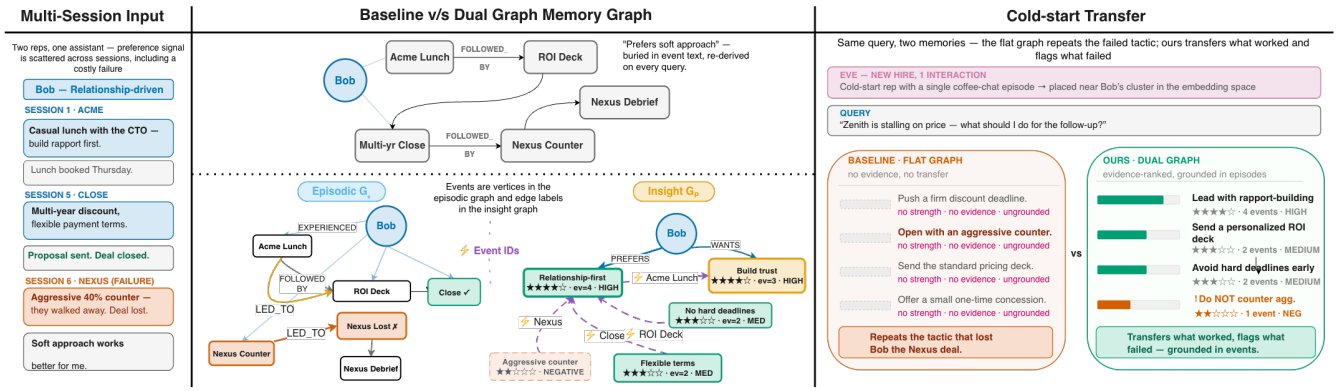


Figure 1: **From scattered multi-session dialogue to grounded cold-start personalization.** In the example shown, Bob tries different strategies with vendors ACME and NEXUS. The graph built for Bob, helps Eve (a new account manager) to close deal with their vendor using proven techniques. Baseline graph buries preferences while our *dual-graph memory* separates episodes and insights (evidence-bearing nodes, strength shown as border thickness), with the *same event as an edge label* bridging them.

interactions as flat text with no structure over the preferences, outcomes, or causal processes shaping behavior. Graph-based systems (GraphRAG (Edge et al. 2025), Zep (Rasmussen et al. 2025), MAGMA (Jiang et al. 2026), EverMemOS (Hu et al. 2026), A-MEM (Xu et al. 2025), Nemori (Ma, Nan, and Wu 2026)) add episodic structure but record only what happened, treating behavioral signals as incidental nodes; even MAGMA's multiple graphs partition events rather than promoting signals to evidence-scored nodes, so neither links an insight to the experiences that formed it.

Preference Modeling: Mem0 (Chhikara et al. 2025) and MemoryOS (Kang et al. 2025) store weighted user facts, but RealPref (Guo et al. 2026) and PAL-Bench (Huang et al. 2025) show explicit extraction fails when preferences must be inferred from behavior. STARK closes this gap with evidence-scored insight nodes bound to their forming episodes by an event-as-edge bridge.

3 STARK Framework

Consider users $\mathcal{U} = \{u_1, \ldots, u_N\}$ interacting with an LLM agent over many sessions spanning long periods. For each u_i the agent observes a temporally ordered stream of conversations $\mathcal{C}_i = \{c_1^i, \ldots, c_{T_i}^i\}$ of user-agent message pairs. The persistent-memory problem is to construct from $\mathcal{C}_i$ a structured memory $\mathcal{M}_i$ such that, given a query q at any future step, the agent retrieves enough context to meaningfully advance the user's underlying goal, from acquiring information to a measurable outcome such as resolving a ticket or closing a deal. Figure 2 illustrates the ingestion architecture, decomposing $\mathcal{M}_i = (\mathcal{G}_E^i, \mathcal{G}_I^i)$.

Fig. 1 shows how STARK handles insights and facts about an account manager and assists them to negotiate deals.

3.1 Episodic Graph

$\mathcal{G}_E^i = (V_E, E_E)$ captures the factual record of a user's interactions. Canonical node types are {User, Event, Milestone, Organization}, where Event nodes form the core, each a discrete interaction observed in conversation, and a Milestone marks a significant workflow or life-state change; the vocabulary is open, extended on demand by induced types such as Activity, Person, or Decision. Edges are temporal (FOLLOWED_BY), causal (LED_TO, CAUSED), or action edges (domain-specific entity interactions, e.g. between users and the entities they act on, such as topics or artifacts), each carrying a timestamp $t(e)$. Together they support temporal, causal, and entity-centric graph retrieval. We also provide qualitative extraction examples on the LongLaMP abstract-generation domain in Appendix A.8.

3.2 Insight Graph

While $\mathcal{G}_E^i$ records what happened, $\mathcal{G}_I^i = (V_I, E_I)$ captures the behavior behind why it happened, with the $User^i$ node as the one fixed type anchoring each user to their signals.

Edges E_I carry two kinds of labels. Structural relations form a stable, closed set $R_s = \{$PREFERS, WANTS, VALUES, TRIED, AVOIDS, LEARNED$\}$; a Preference LEARNED a Constraint, for instance, is a signal-to-signal edge expressible only because signals are nodes rather than edge attributes. Event-linked relations $R_e = \{\mathrm{id}(e) : e \in V_E\}$ reuse episodic event identifiers as labels: an edge $(v_a, e_{id}, v_b) \in E_I$ means the relationship between v_a and v_b was observed during event e_{id}. This *event-as-edge bridge* enables bidirectional navigation between a signal and its originating events, anchoring each insight to its episodic provenance without merging the two graphs (Fig. 2).

Evidence vectors: An agent must decide whether to *act* on an inferred signal, and the cost is asymmetric: a one-off, weakly supported guess is risky, whereas a signal reinforced across many successful episodes warrants confident action. Each non-user signal $v \in V_I \setminus V_I^{User}$ (with V_I^{User} the set of User nodes) therefore carries an evidence vector that lets retrieval rank signals by their support,

$$ev(v) = \big(s(v),\, n(v),\, w(v),\, c(v)\big), \tag{1}$$

updated whenever a conversation reinforces or modifies the signal. The *evidence count* $n(v) \in \mathbb{N}^+$ is the number of distinct supporting episodes, and the *outcome weight* $w(v) = \frac{1}{|R(v)|} \sum_{j \in R(v)} o_j \in [0, 1]$ records how often acting

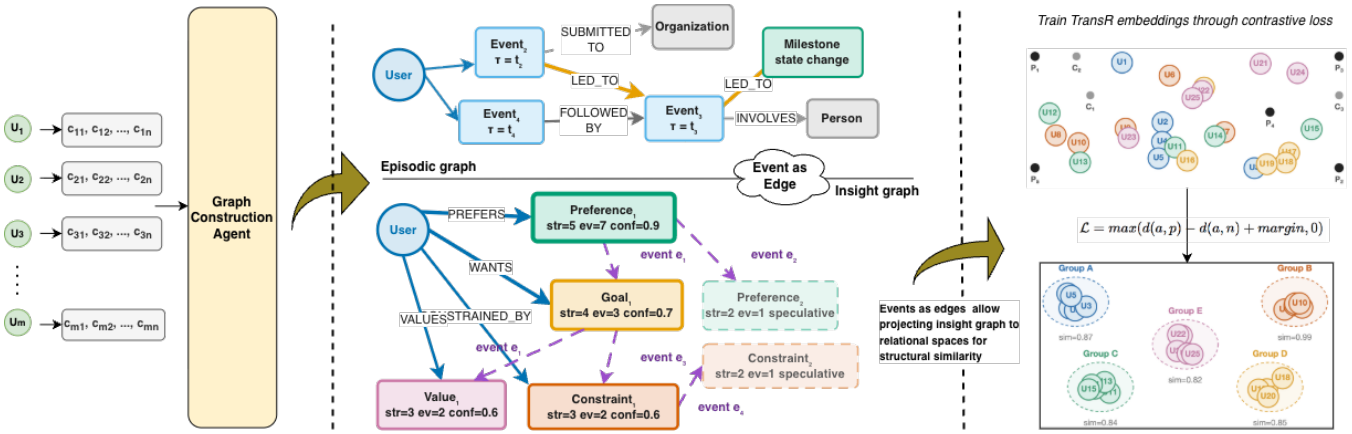


Figure 2: Detailed architecture of STARK ingestion framework. All conversations are first ingested into the dual graphs bridged by episodic events which are then used for obtaining the relational embedding space enabling cross-user insight transfer.

on the signal achieved the user's goal, where $R(v)$ is the set of *resolved* supporting episodes and each o_j is set at ingest from in-conversation markers of whether acting on the signal succeeded, such as whether a proposed approach was adopted or well received. Unresolved or partial episodes still count toward $n(v)$ but are excluded from $w(v)$ rather than scored as failures, so pending evidence does not deflate the outcome weight; when no episode is yet resolved, $w(v)$=0.5. *Strength* grows with evidence and a consistency bonus, $s(v) = \sigma(\alpha_s \log_2(1 + n(v)) + \lambda_s \mathbb{1}_{w(v)>\tau_w})$, with σ the sigmoid, α_s the saturation rate, λ_s the bonus magnitude, and τ_w the outcome-weight threshold; the log scaling yields diminishing returns. *Confidence* rewards volume but penalizes inconsistent outcomes, $c(v) = \sigma(\alpha_c\, n(v) - \lambda_c\, w(v)(1 - w(v)))$, with α_c scaling volume and λ_c the variance penalty $w(1-w)$.

Contradictory signals: Conflicting signals persist as separate nodes with independent evidence vectors rather than being resolved prematurely: a "heavy data augmentation" PREFERS signal that improved results in one subarea (high w) can coexist with an AVOIDS signal learned after it hurt results on a different task (low w), disambiguated by the topic/subarea context on their bridging events. Retrieval surfaces both and resolves the conflict from this stored support, weighing evidence, query context, and recency rather than re-deriving it from raw history; when these leave the signals tied, STARK abstains and presents the trade-off to the user.

4 Cross-User Structural Transfer

Each user's dual-graph is built independently, so every $(\mathcal{G}_E^i, \mathcal{G}_I^i)$ is a self-contained island: a sparse-history user has almost no signals to retrieve, exactly when guidance is most needed. Yet STARK targets domain-specific settings whose users share a goal distribution and converge on recurring patterns. For example, among authors writing in the same research area, one author establishes that a particular method suits a class of problems, and the next author working on that problem class rediscovers the same relevant concepts from scratch. This is the same shared-goal structure that arises among account managers negotiating with a common vendor base in our deployment (Section 7).

We therefore project every user's signal relations into one relational embedding space, learned from triples (h, r, t) over the *combined dual-graph*, the union of all users' $(\mathcal{G}_E^i, \mathcal{G}_I^i)$, and read transfer off the geometry by user-level similarity matching and link prediction over missing signals.

Why a relation-aware embedding? Collaborative filtering (Koren, Bell, and Volinsky 2009) collapses every interaction into one interacts-with edge, whereas our signals are heterogeneous: the same constraint node plays different geometric roles under PREFERS and under the collapsed SURFACED relation. We therefore use TransR (Lin et al. 2015), with entity embeddings $e \in \mathbb{R}^d$, relation embeddings $r \in \mathbb{R}^k$, and per-relation projections $M_r \in \mathbb{R}^{k \times d}$ that map entities into a relation-specific subspace before scoring $f(h, r, t) = \|M_r e_h + r - M_r e_t\|_2$. This projection, which a single-space model like TransE (Bordes et al. 2013) lacks, separates PREFERS from SURFACED and drives clustering, drawing two users who connect to the same signal under the same relation together. It holds only among users sharing a domain and goal distribution, so we claim no cross-domain transfer. We further compare TransR against RotatE and CompGCN in Appendix A.5. We use standard TransR notation: in a triple (h, r, t), t is the tail entity and e an entity embedding, distinct from the episodic timestamp $t(e)$ and event e.

Selective prefixing: The geometry produces transfer only if users share target nodes, so during triple extraction we prefix user-specific entities but leave behavioral-signal entities unprefixed, letting two users point at one shared signal node, the seam along which one user's geometry becomes informative for another (evidence in Appendix A.7). Event-linked labels in R_e appear mostly once, giving TransR no repeated examples to estimate M_r, so we collapse them to a single SURFACED relation while keeping R_s intact.

Preference fingerprints: The first mechanism finds behaviorally similar users so a user, especially a cold-start one, can borrow their signals. Raw user-node embeddings fail, as a user node connects overwhelmingly to unshared episodic edges, so we represent each user only by their shared-signal

Algorithm 1: Dual-Graph Retrieval

Require: Query q, user graphs $(\mathcal{G}_E^i, \mathcal{G}_I^i)$, context budget B, optional TransR model
Ensure: Retrieved context $\mathcal{X}$
Extract entity mentions and query intent $\tau \in \{\textsc{behavioral}, \textsc{episodic}, \textsc{mixed}\}$ from q via LLM, producing seed set $\mathcal{S}$
if TransR model available **then**
    Augment $\mathcal{G}_I^i$ with TransR entity predictions
    Retrieve similar-user insights via preference fingerprints
end if
Insight retrieval: Look up $\mathcal{S}$ in $\mathcal{G}_I^i$; retrieve behavioral signal nodes ranked by $\text{rank}(v) = s(v) \cdot c(v) \cdot (0.5 + w(v))$, so failure-derived signals (w=0) keep half weight
Episodic retrieval: Look up $\mathcal{S}$ in $\mathcal{G}_E^i$; retrieve matching event nodes ranked by temporal recency and causal relevance
Bridge traversal: For each retrieved behavioral signal, follow event-as-edge labels to $\mathcal{G}_E^i$; retrieve originating events as provenance
Context assembly: Allocate budget B according to intent τ: behavioral signals first when $\tau = \textsc{behavioral}$, episodic content first when $\tau = \textsc{episodic}$, interleaved when $\tau = \textsc{mixed}$. Truncate lowest-ranked results to fit B. Cross-user signals are labeled separately.
return $\mathcal{X}$

edges, aggregated into a preference fingerprint:

$$\text{fp}(u) = \frac{\sum_{(u,r,p)\in T_u} w_r \, e_p}{\left\| \sum_{(u,r,p)\in T_u} w_r \, e_p \right\|}, \tag{2}$$

where T_u holds the triples connecting u to shared signal entities and w_r weights relation informativeness (PREFERS above TRIED). User similarity is the cosine between fingerprints, from which a user inherits candidate signals from its top-m most similar users.

Entity-level link prediction: Fingerprint matching needs a few existing signals to anchor the comparison, which cold-start users lack, so the second mechanism asks which specific signal the user is missing, scoring each candidate against its best-fitting link to what the user holds, $\text{predicted}(u) = \arg\min_{p \notin P_{\text{existing}}} \min_{r \in R_p} f(\text{seed}, r, p)$, where seed iterates over P_{existing} and $R_p \subset R_s$ are the preference-relevant relations obtained during graph construction. When no signals exist, the user node itself is the seed under PREFERS and WANTS, so a brand-new user still inherits common domain signals.

The dual-graph structure enables a retrieval strategy that queries both graphs with distinct objectives: the Insight Graph for evidence-ranked behavioral knowledge, and the Episodic Graph for narrative context and temporal grounding. Algorithm 1 describes how the retrieval system works and how context is assembled at the end of it.

5 Experimental Setup

We evaluate on five public benchmarks. LoCoMo (Maharana et al. 2024) and LongMemEval (Wu et al. 2025) target long-horizon memory (exercising the Episodic Graph via temporal, causal, and multi-hop retrieval); RealPref (Guo et al. 2026) and PAL-Bench (Huang et al. 2025) are preference-oriented (exercising the Insight Graph). Evidence scoring is active only where reliable outcome labels exist (RealPref, LoCoMo, LongMemEval); on PAL-Bench and LongLaMP, whose data record no reception signal, signals are ranked uniformly. All four are single-user, so we evaluate cross-user transfer on the public LongLaMP abstract-generation benchmark (Kumar et al. 2024; Salemi et al. 2024), whose authors act as users with disjoint train/test authors for a genuine cold-start setting. We report on the full public test split of each benchmark and simulate cold start by retaining $\{5 - 100\}\%$ of each user's graph operations (protocol in Appendix A.3).

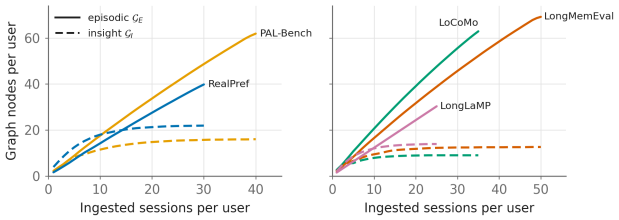


Figure 3: Per-dataset graph growth with ingested sessions. Episodic nodes $\mathcal{G}_E$ (solid) grow roughly linearly; insight nodes $\mathcal{G}_I$ (dashed) saturate as recurring signals merge in place, keeping the retrieved surface small as history accumulates.

We compare against memory systems Mem0 (Chhikara et al. 2025), A-MEM (Xu et al. 2025), Zep (Rasmussen et al. 2025), and MAGMA (Jiang et al. 2026), and full-context LLM baselines (Claude Haiku 4.5, Sonnet 4.5/4.6, and Opus 4.5). For fairness, every memory system and all three A/B arms use Claude Sonnet 4.6 as the backbone and the same B=4000-token retrieval budget as STARK (Table 5; full-context truncation policy in Appendix A.3). MCQ, true/false, and ranking are scored by exact match and are therefore judge-independent; open-ended answers are scored 1–5 (≥ 4 correct) by a GPT-5.1 judge.

6 Results

Graph structure matches full-context on short histories and dominates on long ones. On LoCoMo (Table 1, scores on a 1–5 scale), STARK is best across all four categories: it leads on single-hop (3.98 vs. MAGMA: 3.88, +0.10; vs. best LLM-only Opus 4.5: 3.84) and substantially outperforms all baselines on multi-hop (3.56 vs. MAGMA: 2.64, +0.92), temporal reasoning (3.77 vs. MAGMA: 3.25, +0.52), and open-domain queries (4.08 vs. Zep: 3.15, +0.93). Notably, on LoCoMo every other memory system does worse than full-context Opus 4.5 on multi-hop (2.50–2.70 vs. 3.34): lossy retrieval hurts when the history already fits, and STARK is the only system that avoids this cost. LongMemEval is the opposite case: histories no longer fit and full context collapses (information extraction around 1.6–1.8), so structured retrieval becomes essential. Here STARK achieves 4.38 on information extraction (+1.48 over MAGMA), 2.62 on temporal reasoning (+0.52), and 3.32 on knowledge update

Model	LoCoMo				LongMemEval				LongLaMP (abst gen.)		
	Single Hop	Multi Hop	Temporal	Open Domain	Information extraction	Abstention	Temporal	Knowledge update	RG-1	RG-L	Judge (/5)
LLM-Only (Full Context)											
Claude Haiku 4.5	3.69	3.10	2.12	2.77	1.60	1.75	1.10	1.65	24.0	18.5	2.6
Claude Sonnet 4.5	3.73	3.22	2.38	2.85	1.65	1.80	1.15	1.84	25.2	19.4	2.8
Claude Sonnet 4.6	3.82	3.31	2.58	2.89	1.70	1.90	1.20	1.85	25.8	19.9	2.9
Claude Opus 4.5	3.84	3.34	2.63	2.93	1.77	1.90	1.25	1.88	26.5	20.6	3.0
Long Term Memory Structures											
Mem0	2.90	2.70	2.30	3.10	2.10	2.25	1.55	2.20	28.0	21.5	3.3
MAGMA	3.88	2.64	3.25	2.58	2.90	2.75	2.10	2.70	29.2	22.6	3.5
A-MEM	2.75	2.50	2.10	2.40	1.90	2.00	1.40	1.95	26.8	20.8	3.1
Zep	2.80	2.55	2.20	3.15	2.00	2.15	1.50	2.10	28.5	22.0	3.4
STARK	**3.98**	**3.56**	**3.77**	**4.08**	**4.38**	**2.81**	**2.62**	**3.32**	**31.5**	**24.8**	**4.7**

Table 1: Comparison of STARK against baselines on LoCoMo, LongMemEval, and LongLaMP. All LoCoMo and LongMemEval judge scores are on a 1–5 scale. STARK achieves best LoCoMo single-hop (3.98), multi-hop (3.56), temporal reasoning (3.77), and open-domain (4.08), as well as best LongMemEval info extraction (4.38) and temporal reasoning (2.62).

Model	RealPref					PAL-Bench		
	O.E.	MCQ (%)			T/F	R.S.	Rank (%)	
	(/5)	Ex.	Ge.	To.	(%)	(/5)	T2	B2
LLM-Only (Full Context)								
Haiku 4.5	2.03	75.2	70.7	74.2	54.9	2.69	28.6	34.8
Sonnet 4.5	2.07	84.3	82.5	83.9	57.2	2.73	33.8	37.0
Sonnet 4.6	2.09	85.0	82.0	84.3	61.5	2.77	32.6	36.3
Opus 4.5	2.15	85.4	84.0	85.1	64.1	2.81	34.0	37.7
Long Term Memory Systems								
Mem0	4.02	92.6	89.6	91.9	78.9	3.87	37.7	38.8
MAGMA	3.80	94.0	91.5	93.4	80.0	3.70	40.0	42.0
A-MEM	3.50	89.0	85.0	88.1	74.0	3.40	35.0	36.0
Zep	3.90	91.0	88.0	90.3	77.0	3.60	36.0	37.5
STARK	**4.19**	**98.4**	**97.3**	**98.1**	**86.5**	**4.03**	**41.3**	**44.2**

Table 2: Comparison of STARK against baselines on RealPref and PAL-Bench. Ex./Ge./To. = MCQ accuracy on RealPref's original/generalized/all, macro-averaged over users; T2/B2 = share of the top/bottom 2 ranked candidates; O.E. = open-ended judge; R.S. = requirement-restatement score.

(+0.62). On LongMemEval abstention we match MAGMA, as our retrieval passes nodes similar to the query entities and biases the model toward answering.

Explicit preference nodes drive performance gains on preference-oriented datasets. On RealPref (Table 2), the gain concentrates where memory is irreplaceable: the open-ended score, where the model must produce the preference itself, jumps $+2.04$ over LLM-only ($2.15 \rightarrow 4.19$), whereas MCQ, where the answer is one of the given options, rises far less ($+13.0$pp, to 98.1%). Both gains share one source: preferences exist as evidence-scored nodes retrieved and ranked directly by support and consistency, not re-derived from raw text at query time. This ranking is what free-text stores cannot match: Mem0 nearly matches our open-ended

Method	.05	.1	.2	.3	.5	1.0
No-matching	1.3/12	2.0/14	2.7/16	3.6/18	4.0/19	4.4/20
Random-matching	1.7/12	2.3/14	3.1/16	3.8/18	4.2/20	4.4/21
Label-matching	1.8/13	2.4/15	3.3/17	4.0/19	4.3/21	4.5/22
TransR(user)	1.8/13	2.5/15	3.4/18	4.1/20	4.4/22	4.6/23
TransR(entity)	2.0/14	2.6/16	3.5/18	4.3/21	4.5/22	4.6/24
TransR(combined)	2.0/15	2.7/16	3.7/19	4.4/21	4.6/23	4.7/25

Table 3: Cross-user transfer impact on Open-Ended score (/5) / ROUGE-L at varying graph sparsity levels on the public LongLaMP abstract-generation benchmark. TransR combined = fingerprint similarity + entity link prediction.

fluency (4.02 vs. 4.19) yet trails on MCQ and T/F (91.9%, 78.9% vs. 98.1%, 86.5%), and STARK has the smallest Explicit-to-Generalized MCQ drop of any system (1.1pp vs. 1.4–4.5pp), showing the explicit structure also helps infer unstated preferences. PAL-Bench isolates this ranking step by removing it: it has no action-outcome signal, so STARK skips evidence scoring and treats signals uniformly. Its small residual edge (Top-2 41.3% vs. MAGMA 40.0%) therefore comes from the dual-graph structure alone, and the low scores across all methods reflect how hard it is to order preferences with no success signal to rank on.

TransR enables $1.5\times$ quality recovery at extreme cold-start. At 5% graph sparsity (Table 3, Figure 4b), TransR combined achieves 2.0/5 and 15 ROUGE-L vs. 1.3/12 for text-only. This is where transfer helps most: in a shared-goal domain most of a new user's signals already exist in the collective graph, and coverage grows with the user base (Fig. 4a). Entity-level link prediction dominates across all sparsity levels; combining it with fingerprint similarity adds a further gain. We however see that random matching almost always performs better than no-matching. This means in a shared domain, cross-user insights are prevalent and useful.

Each graph matters only for its own kind of query.

Dataset	Full System	W/o TransR	W/o G_I	W/o Evidence	W/o G_E	W/o Graph
LoCoMo (Single-Hop)	3.98 / 1786	3.98 / 1786	3.90 / 1400	3.95 / 1786	3.55 / 1350	3.82 / 2466
LongMemEval (Info extraction)	4.38 / 1839	4.38 / 1839	4.05 / 1500	4.30 / 1839	2.60 / 1400	1.70 / 3823
LongLaMP (Open ended)	4.70 / 1900	4.41 / 1810	4.05 / 1500	4.70 / 1900	4.20 / 1400	2.90 / 4500
RealPref (Open ended)	4.19 / 5531	4.19 / 5531	2.95 / 4800	3.70 / 5531	3.85 / 4500	2.09 / 3388
PAL-Bench (R.S.)	4.03 / 7295	4.03 / 7295	3.10 / 6200	4.03 / 7295	3.75 / 5800	2.77 / 7242

Table 4: Full system performance when removing one component at a time. Each cell: STARK score (1–5) / response generation latency (ms) with LLM backbone as Claude Sonnet 4.6. Evidence scoring is disabled on PAL-Bench and LongLaMP, which have no outcome signal; TransR affects only LongLaMP, the only shared-domain benchmark.

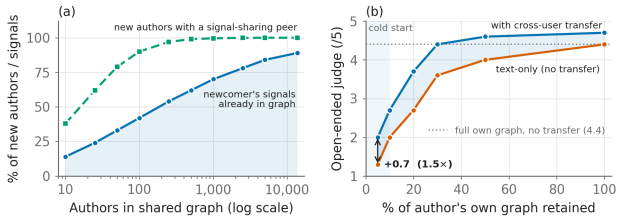


Figure 4: Cross-user transfer scales on both deployment axes. (a) With more users, more of a newcomer's signals already exist in the collective graph (blue) and a signal-sharing peer becomes near-certain (green). (b) At 5% of their own history, transfer recovers 2.0/5 vs. 1.3/5 text-only (1.5×).

Removing the **Episodic** Graph badly hurts the factual and temporal benchmarks (LongMemEval $4.38 \rightarrow 2.60$; LoCoMo -0.43) but barely affects preferences (RealPref -0.34); removing the **Insight** Graph does the opposite (RealPref $4.19 \rightarrow 2.95$; PAL-Bench -0.93). Evidence scoring matters most for preferences (RealPref -0.49) and only marginally for factual recall (LoCoMo -0.03, LongMemEval -0.08), whose answers come mainly from the Episodic Graph; on LongLaMP and PAL-Bench, whose data carry no reception signal, it is disabled, so the two columns coincide. TransR affects only the shared-domain LongLaMP (-0.29, $+90$ ms at $O(1)$). On long histories STARK is also faster than full context (e.g. LongMemEval 1839 vs. 3823 ms) as it sends a small subgraph, while insights saturate as episodes grow (Fig. 3), keeping the read path flat.

7 Deployment and Impact

STARK is deployed as a drop-in *memory layer* for a **vendor-negotiation assistant** used by Account Managers (AMs) who negotiate terms with thousands of vendors. Negotiations span many sessions across quarters, constraints surface mid-deal and must inform later ones, and closing strategies sit buried in each AM's history; new hires start cold, and the team rediscovers the same constraints independently.

We ran an 8-week live A/B test over the 40 AMs (18 new hires under 6 months, 22 with 3+ years). It is paired at the vendor level: each of a manager's five vendors gets three otherwise identical deals, run separately to avoid carryover: memory-on, memory-on with cross-user transfer disabled (transfer-off), and memory-off control. This yields 200 vendor triples (600 deals), with the order of the three arms randomized within each triple, and the transfer-off arm isolates cross-user transfer with the rest of the memory layer fixed. Each AM's dual-graph was initialized from up to a year of their prior history (under six months for new hires), so every arm retrieves over the AM's full available history from the first deal; the 8-week window scopes only the deals conducted in the pilot, all 600 of which closed, so cycle time carries no closure-based selection. Vendors respond live, each deal carries an AM-provided target, and all arms use Claude Sonnet 4.6. Memory-on cut negotiation rounds per deal by 18% and cycle time by 15% (paired t-test, n=200, both $p < 0.05$), raised deals closed at or above target by 9 pp, and lifted blind guidance ratings by 0.8/5. The gains concentrate in new hires ($\sim$24%/$\sim$19% vs. experienced $\sim$13%/$\sim$11%). The transfer-off arm pins this on cross-user transfer: without it, memory-on retained only $\sim$11%/$\sim$9%, so transfer supplies roughly 40% of the gain overall and most of it for new hires (new $\sim$9% vs. $\sim$24%; experienced $\sim$12% vs. $\sim$13%). Through the pilot we learned that the AMs preferred inferred signals *surfaced* as accept-or-dismiss options rather than acted on, so we leave the decision to them. We also observed that new–experienced fingerprint similarity (Eq. 2) rose $0.31 \rightarrow 0.58$, a descriptive within-arm trend (not isolated by the transfer-off arm).

Why it works? Surfacing known constraints and concession patterns up front removes rounds spent rediscovering them; outcome-weighting (Eq. 1) front-loads previously successful tactics and down-ranks rejected ones, shortening the cycle; and for new hires this knowledge arrives via cross-user transfer. Cost, governance, and rollout are in Appendix A.6.

Conclusion

We presented STARK, a dual-graph memory that decouples episodic events from behavioral knowledge via an event-as-edge bridge, enabling evidence-ranked retrieval and cross-user transfer for cold-start inference. However, STARK has several important limitations. TransR updates in $O(1)$ but its per-relation projections have fixed capacity, so embedding quality degrades as the graph densifies. Privacy is enforced agentically, not formally: we do not bound residual leakage of aggregate fingerprints under membership inference. Transfer is domain-specific, holding only among users who share a goal distribution. In future work, we plan to use the graph to look forward rather than back. Using behavioral paths as a search tree and running an MCTS-style rollout over likely trajectories can help the agent act on the path most likely to reach the user's goal.

References

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating Embeddings for Modeling Multi-relational Data. In Burges, C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Chhikara, P.; Khant, D.; Aryan, S.; Singh, T.; and Yadav, D. 2025. Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. arXiv:2504.19413.

Edge, D.; Trinh, H.; Cheng, N.; Bradley, J.; Chao, A.; Mody, A.; Truitt, S.; Metropolitansky, D.; Ness, R. O.; and Larson, J. 2025. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130.

Guo, Q.; Li, Y.; Liu, Y.; and Hooi, B. 2026. Towards Natural Personalization: Evaluating Long-Horizon Preference Following in Personalized User-LLM Interactions. arXiv:2603.04191.

Gutiérrez, B. J.; Shu, Y.; Gu, Y.; Yasunaga, M.; and Su, Y. 2024. HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. In Globerson, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J.; and Zhang, C., eds., *Advances in Neural Information Processing Systems*, volume 37, 59532–59569. Curran Associates, Inc.

Hu, C.; Gao, X.; Zhou, Z.; Xu, D.; Bai, Y.; Li, X.; Zhang, H.; Li, T.; Zhang, C.; Bing, L.; and Deng, Y. 2026. EverMemOS: A Self-Organizing Memory Operating System for Structured Long-Horizon Reasoning. arXiv:2601.02163.

Huang, Z.; Dai, Q.; Wu, G.; Wu, X.; Chen, K.; Yu, C.; Li, X.; Ge, T.; Wang, W.; and Jin, Q. 2025. Mem-PAL: Towards Memory-based Personalized Dialogue Assistants for Long-term User-Agent Interaction. arXiv:2511.13410.

Jain, A.; Ghosh, P.; and Yenigalla, P. 2026. CASPER: Bridging Discrete and Continuous Prompt Optimization through Feedback-Guided Gradient Descent. In Matusevych, Y.; Eryiğit, G.; and Aletras, N., eds., *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track)*, 425–437. Rabat, Morocco: Association for Computational Linguistics. ISBN 979-8-89176-384-5.

Jiang, D.; Li, Y.; Li, G.; and Li, B. 2026. MAGMA: A Multi-Graph based Agentic Memory Architecture for AI Agents. arXiv:2601.03236.

Kang, J.; Ji, M.; Zhao, Z.; and Bai, T. 2025. Memory OS of AI Agent. arXiv:2506.06326.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8): 30–37.

Kumar, I.; Viswanathan, S.; Yerra, S.; Salemi, A.; Rossi, R. A.; Dernoncourt, F.; Deilamsalehy, H.; Chen, X.; Zhang, R.; Agarwal, S.; Lipka, N.; Nguyen, C. V.; Nguyen, T. H.; and Zamani, H. 2024. LongLaMP: A Benchmark for Personalized Long-form Text Generation. arXiv:2407.11016.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.; Rocktäschel, T.; Riedel, S.; and Kiela, D. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781713829546.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, 2181–2187. AAAI Press. ISBN 0262511290.

Ma, W.; Nan, J.; and Wu, W. 2026. What Deserves Memory: Adaptive Memory Distillation for LLM Agents. In Liakata, M.; Moreira, V. P.; Zhang, J.; and Jurgens, D., eds., *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 34789–34812. San Diego, California, United States: Association for Computational Linguistics. ISBN 979-8-89176-390-6. Earlier arXiv versions (arXiv:2508.03341) titled "Nemori: Self-Organizing Agent Memory Inspired by Cognitive Science".

Maharana, A.; Lee, D.; Tulyakov, S.; Bansal, M.; Barbieri, F.; and Fang, Y. 2024. Evaluating Very Long-Term Conversational Memory of LLM Agents. arXiv:2402.17753.

Packer, C.; Wooders, S.; Lin, K.; Fang, V.; Patil, S. G.; Stoica, I.; and Gonzalez, J. E. 2024. MemGPT: Towards LLMs as Operating Systems. arXiv:2310.08560.

Park, J. S.; O'Brien, J.; Cai, C. J.; Morris, M. R.; Liang, P.; and Bernstein, M. S. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23. New York, NY, USA: Association for Computing Machinery. ISBN 9798400701320.

Rasmussen, P.; Paliychuk, P.; Beauvais, T.; Ryan, J.; and Chalef, D. 2025. Zep: A Temporal Knowledge Graph Architecture for Agent Memory. arXiv:2501.13956.

Richardson, C.; Zhang, Y.; Gillespie, K.; Kar, S.; Singh, A.; Raeesy, Z.; Khan, O. Z.; and Sethy, A. 2023. Integrating Summarization and Retrieval for Enhanced Personalization via Large Language Models. arXiv:2310.20081.

Salemi, A.; Mysore, S.; Bendersky, M.; and Zamani, H. 2024. LaMP: When Large Language Models Meet Personalization. arXiv:2304.11406.

Sun, Z.; Deng, Z.; Nie, J.; and Tang, J. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. arXiv:1902.10197.

Vashishth, S.; Sanyal, S.; Nitin, V.; and Talukdar, P. 2020. Composition-based Multi-Relational Graph Convolutional Networks. arXiv:1911.03082.

Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; Zhao, W. X.; Wei, Z.; and Wen, J. 2023. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Wu, D.; Wang, H.; Yu, W.; Zhang, Y.; Chang, K.; and Yu, D. 2025. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. arXiv:2410.10813.

Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; Awadallah, A. H.; White, R. W.; Burger, D.; and Wang, C. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. arXiv:2308.08155.

Xu, W.; Liang, Z.; Mei, K.; Gao, H.; Tan, J.; and Zhang, Y. 2025. A-MEM: Agentic Memory for LLM Agents. arXiv:2502.12110.

Zhong, W.; Guo, L.; Gao, Q.; Ye, H.; and Wang, Y. 2024. MemoryBank: Enhancing Large Language Models with Long-Term Memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17): 19724–19731.

A Appendix

A.1 Hyperparameter setup

Table 5 consolidates the hyperparameters of STARK that are free rather than fixed by the architecture, grouped by the component that uses them. The evidence-scoring constants govern how strength and confidence saturate with accumulated evidence; the retrieval constant sets the read-path context budget (Algorithm 1); and the transfer constants configure neighbor selection, fingerprint weighting, and the TransR embedding. All remaining settings follow standard practice.

The evidence-scoring constants are set so that strength rises smoothly from ≈ 0.67 at a single supporting episode to ≈ 0.94 at fifteen, with the consistency bonus applied once a signal's outcome weight clears the majority threshold $\tau_w = 0.5$. The fingerprint weights w_r rank structural relations by how strongly they express a durable preference: $w_{\text{PREFERS}} = w_{\text{WANTS}} = 1.0$, $w_{\text{VALUES}} = 0.8$, $w_{\text{LEARNED}} = 0.7$, $w_{\text{AVOIDS}} = 0.6$, and $w_{\text{TRIED}} = 0.5$.

Symbol	Description	Value
Evidence scoring		
α_s	strength saturation rate	0.7
λ_s	consistency-bonus magnitude	0.5
τ_w	outcome-weight threshold	0.5
α_c	confidence volume sensitivity	0.5
λ_c	outcome-variance penalty	2.0
Retrieval (Algorithm 1)		
B	context budget (tokens)	4000
Cross-user transfer		
m	similar users per user	5
TransR embedding		
d	entity embedding dimension	64
k	relation embedding dimension	64
γ	ranking-loss margin	1.0
η	learning rate	0.01
–	training epochs	100
–	negatives per positive triple	1

Table 5: Hyperparameter settings for STARK, grouped by the component that uses them. Evidence-scoring constants are set by design, not tuned; m is tuned on the LongLaMP validation split, whose authors are disjoint from the 4,560 test authors; TransR and retrieval-budget values follow standard practice.

A.2 LongLaMP Abstract-Generation Benchmark

We evaluate on the public LongLaMP abstract-generation benchmark (Kumar et al. 2024; Salemi et al. 2024) (the `abstract_generation_user` split), whose users are authors drawn from a large citation-network corpus. The split places disjoint authors in train and test (13,693 train / 4,560 test, zero overlap), so every test author is genuinely unseen—a true cold-start setting for cross-user transfer. A separate validation split, whose authors are disjoint from the test authors, is used only for hyperparameter tuning (Appendix A.1). Each author's profile is their set of prior paper abstracts, and the personalization target is the abstract of a held-out paper.

We treat each author as a user, each prior paper as an episode in the Episodic Graph, and the research topics, methods, and concepts recurring across an author's abstracts as the behavioral-signal nodes of the Insight Graph. Because authors in the same research area converge on shared topics and methods, these signal nodes recur across users, providing the shared-domain structure that cross-user transfer relies on. For the clustering metric of Appendix A.7 we group authors by research field (assigned per author by an LLM from their profile abstracts) to obtain the ground-truth grouping.

A.3 Evaluation Protocol

We report on the full public test split of each benchmark: LoCoMo's 10 conversations, LongMemEval-S's 500 questions, RealPref's 100 users and 1,300 preferences, and PAL-Bench's 100 users; the LongLaMP split is detailed above (Appendix A.2). Cold start is simulated by retaining $\{5, 10, 20, 30, 50, 100\}\%$ of each user's graph operations to measure resilience to sparsity; at low sparsity we exclude factual MCQ and T/F questions whose answers are absent from the sparse graph. Evidence scoring uses per-episode outcome labels o_j inferred at ingest by an LLM on RealPref, LoCoMo, and LongMemEval, where in-conversation reception signals exist; PAL-Bench and LongLaMP carry no such signal, so evidence scoring is disabled and their signals are ranked uniformly. Every memory system retrieves under the same B=4000-token budget as STARK; the full-context LLM baselines receive the entire history when it fits the backbone's context window and, on histories that exceed it, are truncated to the most recent tokens that fit.

A.4 TransR training details

We train a single TransR model over the combined multi-user graph, with entity embeddings $e \in \mathbb{R}^d$, relation embeddings $r \in \mathbb{R}^k$, and per-relation projection matrices $M_r \in \mathbb{R}^{k \times d}$, scored by $f(h, r, t) = \|M_r e_h + r - M_r e_t\|_2$. Training minimizes a margin-based ranking loss over the positive triples in the combined set against tail-corrupted negatives (a positive (h, r, t) paired with (h, r, t') for a sampled $t' \neq t$), pushing the geometry to score observed structure below corrupted

structure; we use otherwise standard TransR hyperparameters and do not tune the embedding dimensions d and k per dataset. A full offline train takes 58 s on the LongLaMP training set. Between full retrains the model is updated incrementally in $O(1)$ per new triple as conversations arrive, and a periodic full retrain admits new users and entities; in the interval before a user appears in a retrain, cold-start link prediction serves them off the existing embedding space. We adopt TransR because a live memory needs incremental, no-full-graph-recompute updates, which rules out CompGCN (it requires full message-passing at inference); we prefer it to RotatE, which is competitive on our metrics, for its interpretable per-relation subspaces (see Appendix A.5).

A.5 TransR comparison with other embedding models

The Insight Graph contains semantically heterogeneous relations (PREFERS, SURFACED, WANTS) that require distinct geometric subspaces: the space in which a user *prefers* a behavioral pattern differs from the space in which that pattern *surfaces* a constraint. TransR addresses this directly via per-relation projection matrices $\mathbf{M}_r$ that map entities into relation-specific subspaces before translational scoring ($\mathbf{M}_r\mathbf{h} + \mathbf{r} \approx \mathbf{M}_r\mathbf{t}$). RotatE (Sun et al. 2019) models relations as rotations in a single complex space; empirically it clusters comparably to TransR (Table 9), so we claim no quality advantage over it. We nonetheless adopt TransR because its explicit per-relation projection matrices expose an inspectable subspace per relation, which suits the provenance and audit requirements of the deployment, and because its real entity space matches the fingerprint's real-vector aggregation $\mathrm{fp}(u) = \sum w_r \cdot \mathrm{emb}(p_i)$. CompGCN (Vashishth et al. 2020) entangles entity representations across relation types during neighborhood aggregation and requires full message-passing at inference, making it incompatible with the $O(1)$ per-session incremental updates a live memory needs; TransR and RotatE both support such updates, CompGCN does not.

A.6 Deployment: Cost, Governance, and Rollout

Engineering readiness. STARK concentrates cost on the one-time write path and keeps the read path flat as a user's graph grows. Ingestion is the cost center: each conversation is processed by two sequential LLM passes (episodic then insight), and a repeated behavioral signal is folded into an existing node by recomputing its evidence vector in place (Eq. 1) rather than recomputing the graph, so write cost is bounded by the two passes and is independent of accumulated graph size. The read path is a fixed-budget filter over evidence-ranked signals: a query assembles about 3.8K tokens from about 74 entities in 710–770 ms regardless of how many sessions the user has logged, with cross-user transfer adding only about 90 ms. The embedding component is similarly write-heavy and read-cheap: TransR is trained offline (58 s on the LongLaMP training set used in our experiments) and served with $O(1)$ incremental updates per new triple, with periodic full retraining ($\sim$11 min at 10K users) to admit new users and entities; between retrains, cold-start link prediction serves users the model has not yet seen. Table 6 consolidates the cost and governance profile.

Phase	Metric	Value
Ingestion	Passes per conversation	2 (episodic, insight)
	Latency / tokens per conv.	6.2 s / 4.3K
Retrieval	Entities / tokens / latency	$\sim$74 / $\sim$3.8K / 710–770 ms
	Cross-user add-on	+90 ms
Embedding	Full train (LongLaMP)	58 s
	Incremental / retrain at scale	$O(1)$ / $\sim$11 min @ 10K
Governance	Cross-user channel	signal nodes + embeddings
	Forget / PII policy	`DELETE_NODE` / redaction

Table 6: Production cost and governance profile for STARK. Cost concentrates on the one-time write path while the read path stays cheap and graph-size-stable.

Stage	Go/no-go gate	Status
0. Offline replay	Recover $\geq 80\%$ known constraints; judge uplift $\geq +0.5$	Passed
1. Shadow	p95 read $\leq$ 900 ms; 0 PII leaks; $\geq 90\%$ traceable	Passed
2. Pilot A/B	$\geq 15\%$ fewer rounds; positive blind guidance-rating lift	Passed
3. GA (all managers)	Sustained pilot gains; $< 1\%$ rollback	Underway

Table 7: Staged deployment plan for the vendor-negotiation assistant. Each stage advanced only on its go/no-go gate.

Governance, privacy, and compliance. Cross-user transfer shares only the behavioral-signal layer: the LLM-extracted names and short descriptions of unprefixed signal nodes, plus the learned embeddings and fingerprint aggregates over them. Episodic records, raw conversation text, and user-specific entities are user-prefixed and never cross a user boundary, and transferred signals are rendered under explicit caveat headers kept separate from a user's own memory. Erasure maps directly onto the `DELETE_NODE` and `UNLINK` operations, with a forgotten user's contribution to the shared embedding clearing at the next full retrain. The event-as-edge bridge keeps every acted-on signal traceable to its supporting episodes, giving the provenance an enterprise audit requires. These properties cleared the security and privacy sign-off required before the pilot went live.

Rollout plan. We advanced through four staged gates (Table 7), each with an explicit go/no-go criterion, so that no stage advanced on academic proxies alone: offline replay on historical logs, shadow mode with recommendations hidden from managers, the A/B pilot, and general availability. Stages 0–2 have passed their gates; on the strength of the pilot we are now beginning the Stage 3 rollout to all managers, with scheduled retraining, monitoring, and rollback in place.

A.7 Cross-User Transfer: Design Justification

For enabling cross user transfer we make three design choices that drive cross-user transfer: selective prefixing, relation normalization, and preference fingerprints. This appendix reports the experiments that justify each.

Setup. All measurements use the public LongLaMP abstract-generation benchmark (Kumar et al. 2024), whose

Sharing mode	Separation	Judge
Isolated (per-user, no sharing)	0.022	2.45
Shared relations only	0.082	2.55
Shared relations + entities (ours)	**0.174**	**2.66**

Table 8: Selective prefixing ablation. Clustering separation (within-group minus between-group fingerprint cosine) and average low-sparsity judge score, as entity/relation sharing varies under a fixed TransR model. Leaving behavioral-signal entities unprefixed (full sharing) is what produces clustering.

users are authors with disjoint train and test splits; for the clustering metric we group authors by research field. The primary metric is *clustering separation*: the mean cosine similarity between fingerprints of users in the same group minus the mean cosine similarity between fingerprints of users in different groups. A higher value means same-group users sit closer in the embedding space, which is the precondition for transferring one user's signals to a behaviorally similar one; a value near zero means the space carries no transferable structure.

Selective prefixing is what creates the clustering: Table 8 varies how entities are shared across users while holding TransR fixed. With per-user (isolated) training, no entity is shared and separation is 0.022, essentially the no-structure floor: same-group users are no closer than unrelated ones. Sharing only relation projections raises it to 0.082, and sharing behavioral-signal entities as well (our full design, in which signal nodes stay unprefixed) raises it to 0.174, almost an eightfold gain over the isolated baseline. The shared signal node, not the embedding model alone, is what makes behaviorally similar users cluster.

TransR is the right embedding for heterogeneous relations: Table 9 compares embedding models under full sharing. A single-space model (TransE) reaches separation 0.095 because it cannot give a signal different geometry under PREFERS versus SURFACED; per-relation projection (TransR) nearly doubles this to 0.174. RotatE is marginally higher (0.185 separation, 2.68 judge vs. our 0.174/2.66), but the gap is within single-run noise, so the two are effectively tied on quality; we adopt TransR for its interpretable per-relation subspaces and its fit with the real-vector fingerprint, both models supporting the incremental updates a live memory needs.

Relation normalization makes the event relations trainable: Event-linked labels in R_e are episodic identifiers that are unique and appear mostly once, so TransR has no repeated examples from which to estimate their projection matrix M_r. Left raw, the relation vocabulary holds 197 relations, 83 of them seen exactly once.

Fingerprints isolate the transferable structure: A user node connects to about 82% episodic edges and only 18% shared-signal edges, so its raw embedding is dominated by user-specific episodic neighborhoods, the part that is by construction not shared and not transferable; raw user-to-user similarity is therefore near random. The preference fingerprint (Eq. 2) discards the episodic mass and aggregates only the shared-signal 18%, which is what turns near-random raw similarity into the clustering reported in Table 8.

Embedding	Separation	Judge
TransE (single space)	0.095	2.60
TransH (Wang et al. 2014)	0.120	2.62
TransR (per-relation, ours)	0.174	2.66
RotatE (complex rotation)	0.185	2.68

Table 9: Embedding-model ablation under full sharing. Per-relation projection (TransR) clusters far better than single-space TransE; RotatE is marginally higher but within single-run noise, and we adopt TransR for its interpretable per-relation subspaces (both support incremental updates).

Source	**Signal the graph extracts**
Draft note	"Strengthen the evaluation" the *intent* only; not the specifics.
Tool I/O	Experiment-tracker / citation-manager results: added a low-resource retrieval baseline, +2 datasets, ablation FAIL → PASS after a fix. Stored as typed Event nodes with exact attributes.
Revision state	Paper task BLOCKED → ACTIVE; camera-ready checklist unresolved 12 days. Stored as a state-change milestone and a revision-delay pattern.

Table 10: Multi-source signal extraction for a single paper-writing episode (`p-4a7f2e`; low-resource retrieval work). The draft note conveys only intent; the exact experimental changes come from tool I/O and the delay pattern from revision state. STARK extracts all three as typed, queryable nodes, whereas raw context stuffing would re-read the entire history to recover any of them.

A.8 Qualitative Samples

We present three qualitative examples showing how the dual-graph improves agent behavior over generic LLM baselines.

Multi-Source Signal Extraction: A single paper-writing episode showing how different data sources contribute distinct signals extracted as typed, queryable nodes (Table 10).

Learning Through Failure: Without vs. With Personalization: Author A drafts papers in the same line of work across three successive submissions. Table 11 contrasts a generic LLM (no memory; red) with STARK (graph memory; green).

Cross-User Research-Concept Transfer: Two authors work in the same research area. Author A1 establishes a method/concept first; Author A2, a cold-start author, then benefits from A1's structure without ever communicating with them (Table 12).

	Generic LLM (no memory)	STARK (graph memory)
P1 (first paper)	Frames the method as a purely supervised pipeline; reviewers REJECT ("insufficient low-resource evidence"). Advises generically: "add experiments or soften claims."	Same rejection, but *stores* `supervised_only_lowresource_weakness`: "supervised-only framing rejected : weak in low-resource settings."
P2 (months later)	"What framing would you like?" : no memory of the rejection; starts from zero.	Recalls the failure; recommends a contrastive-pretraining framing instead. Paper ACCEPTED; graph logs the strategy evolution.
P3 (later still)	"What framing would you like?" : again no memory; author re-specifies everything.	Proactively applies the proven contrastive-pretraining framing and cites the reason from graph memory. No re-specification.

Table 11: Learning through failure across three successive papers by the same author. Without memory (red), the agent repeats generic advice and forgets each rejection; with the dual-graph (green), the P1 failure is stored as a behavioral signal, recalled in P2 to recommend the fix, and reused proactively in P3.

A1 - donor (signal contributed to the shared embedding space)	
Behavior	Across several papers on low-resource retrieval, A1 consistently frames contrastive pretraining as the method of choice, establishing a recurring topic–method association in their abstracts.
Signal formed	A "contrastive pretraining → low-resource retrieval" concept signal (PREFERS). As an unprefixed behavioral-signal node, it is shared across users in the combined embedding space; A1's own papers (episodic records) stay private.
A2 - receiver, cold-start author in the same area, no prior papers	
Without graph	With no profile to condition on, the abstract is generated generically, omitting the contrastive-pretraining framing that characterizes the area.
With graph	Entity-level link prediction over the shared space infers A2 is likely to hold the same contrastive-pretraining concept and surfaces it under a cross-user caveat header. The generated abstract adopts the field-appropriate framing on the first attempt.

Table 12: Cross-user research-concept transfer. A1's recurring topic–method association forms a behavioral-signal node that, left unprefixed, is shared across users in the TransR embedding space. Because A2 is cold-start with no signals to anchor fingerprint matching, entity-level link prediction infers the concept for A2 from the geometry alone—no raw text or episodic records cross the user boundary—surfacing it as clearly labeled cross-user evidence (green). An LLM baseline sees only A2's own (empty) profile (red). As an inferred signal it is advisory and may be wrong (Limitations).

A.9 Step-by-Step Pipeline Traces on Public Benchmarks

To complement the personalized-generation examples in the qualitative samples mentioned in appendix (LongLaMP abstract-generation dataset), we trace the full pipeline on two long-horizon conversational benchmarks, LoCoMo and LongMemEval. For each we report (i) the *progressive ingestion* of the dual-graph $(\mathcal{G}_E^i, \mathcal{G}_I^i)$ (Table 13; Figure 5, which visualizes the episodic and preference graphs separately as a growth flow toward the full dataset-scale graph); and (ii) a *retrieval trace* following one real query per dataset, from the dataset-scale memory graph down to the retrieved subgraph (Figure 6), with each sub-figure caption reporting that query's intent, pipeline-stage flow, and latencies.

A.10 Prompt Library

Episodic Extraction Prompt

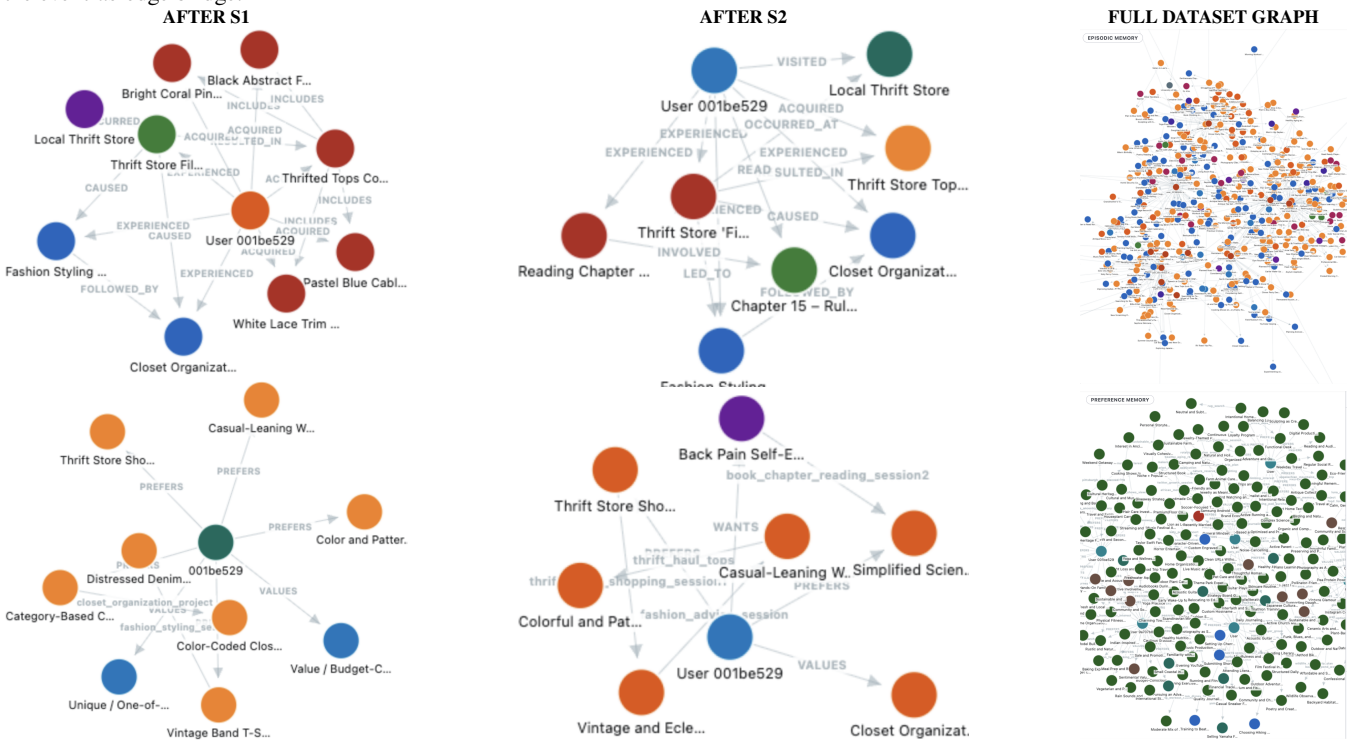
Episodic Extraction Prompt

```
You are an expert episodic-memory
extraction agent for personal
conversation analysis.
YOU MUST RESPOND WITH **ONLY** VALID JSON
- ABSOLUTELY NO OTHER TEXT.
TASK: Extract an EPISODIC MEMORY GRAPH
from a conversation using memory
operations.
You will receive: 1. A user_id
identifying the person 2. A conversation
(and optionally persona context) 3.
Optionally, a session_date (the date/time
when this conversation session took
place)
== OPERATIONS ==
You have 6 primitive operations:
ADD_NODE - Create a new node (event,
person, location, pet, item, etc.)
Required: node_id (snake_case), node_type
(dynamic), properties (dict) Properties
MUST include: user_id, graph_type
```



8 ep / 5 pref, 19 e | 9 ep / 7 pref, 19 e | 650 nodes, 827 edges, 7 users

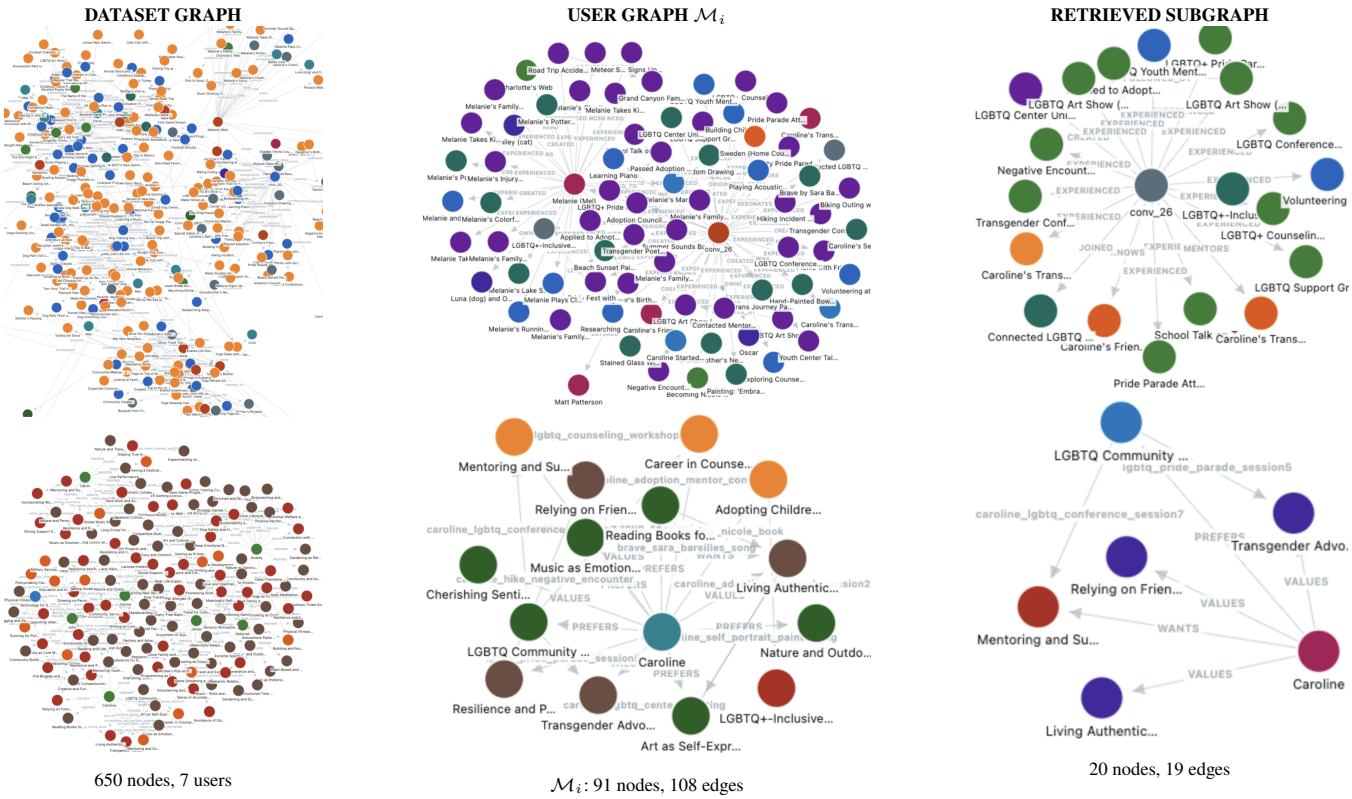
(a) **LoCoMo.** One trace user's dual-graph grows session by session (After S1 → After S2), then the Full Dataset Graph shows the scale of memory the system holds across all users. Each stage is split into its episodic (top) and preference (bottom) graphs, co-evolving and linked by the event-as-edge bridge.



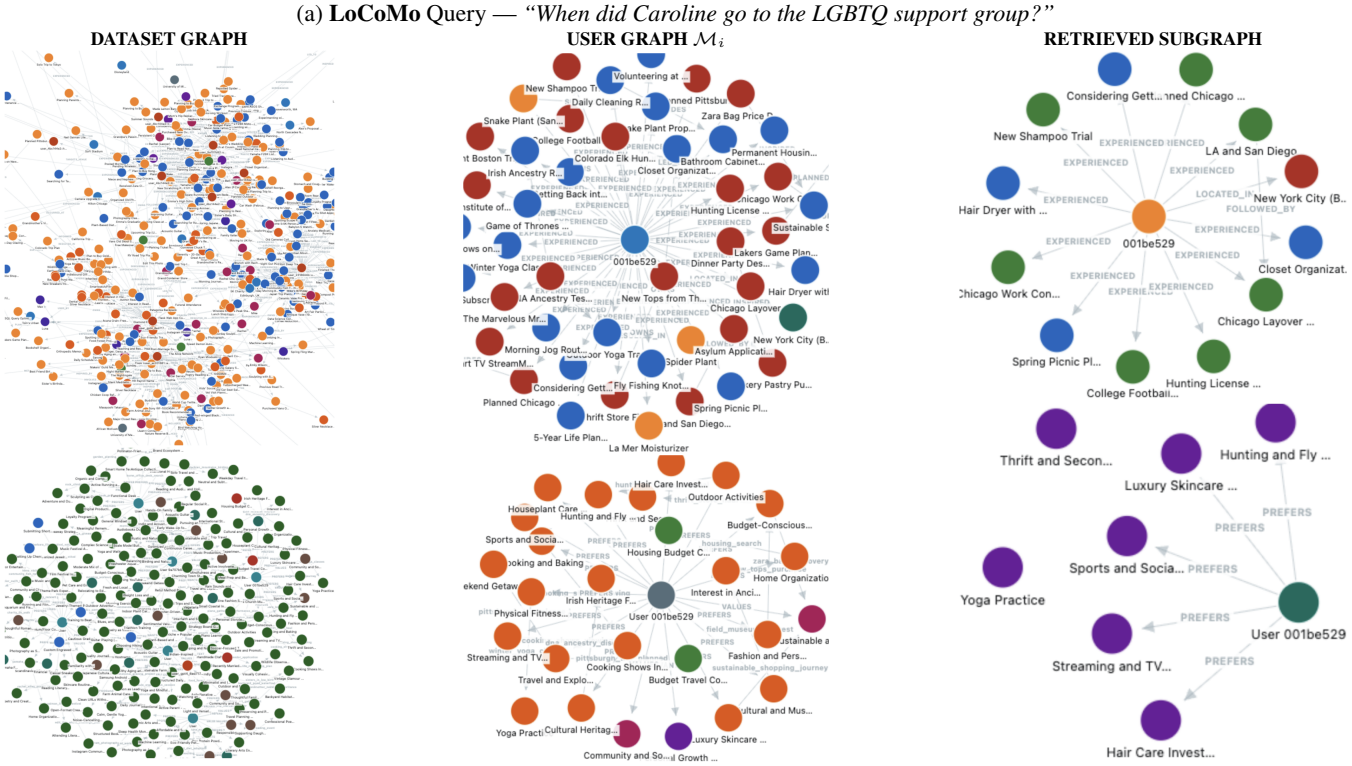
10 ep / 9 pref, 26 e | 8 ep / 8 pref, 21 e | 610 nodes, 719 edges, 6 users

(b) **LongMemEval.** Same layout: per-session growth (After S1 → After S2) followed by the Full Dataset Graph. LongMemEval's S1→S2 node count fluctuates under LLM extraction non-determinism.

Figure 5: Progressive dual-graph growth on **LoCoMo** and **LongMemEval**, read left to right as a flow: the trace user's graph after session 1, after session 2, and the full dataset-scale graph. Each stage shows its Episodic graph $\mathcal{G}_E^i$ (events, milestones, top) and Preference graph $\mathcal{G}_I^i$ (preferences, values, goals, bottom), co-evolving and joined by the event-as-edge bridge. Per-session node counts are per graph; edge counts are dual-graph totals; the full-graph counts are dataset totals across all users.



(a) **LoCoMo** Query — *"When did Caroline go to the LGBTQ support group?"*



(b) **LongMemEval** (QUERY) — *"What degree did I graduate with?"*

Figure 6: Retrieval traced per dataset, left to right: dataset-scale memory graph → user graph $\mathcal{M}_i$ → retrieved 20-node subgraph; each column shows its episodic (top) and preference (bottom) graphs.

Dataset	Session	Nodes (Ep/Pref)	Edges	Representative new signals
LoCoMo 438 turns	S1 (t. 1–19)	13 (8/5)	19	*LGBTQ Support Group Attendance* (Event); *Caroline* (User); *Caroline's Transgender Transition* (Milestone); *LGBTQ Community Belonging* (Pref); *Transgender Advocacy* (Value)
	S2 (t. 20–37)	16 (9/7)	19	*Charity Race* $\xrightarrow{\text{LED_TO}}$ *Mental Health Awareness* (Value); *Adoption Application* reinforces *Adopting Children and Building a Family* (Goal)
	Full graph	650	827	Dataset-scale memory across all 7 users (totals; not a single user's per-session graph)
LongMemEval 565 turns	S1 (t. 1–13)	19 (10/9)	26	*Thrift and Second-Hand Shopping* (Pref); *Vintage Band T-Shirt Styling* (Event); *Sustainable Fashion* (Value)
	S2 (t. 14–16)	16 (8/8)	21	Node count fluctuates under LLM extraction non-determinism (the agent consolidates entities differently given more context; see Section 3)
	Full graph	610	719	Dataset-scale memory across all 6 users (totals; not a single user's per-session graph)

Table 13: Progressive dual-graph ingestion for **LoCoMo** and **LongMemEval** (episodic and preference node counts shown as Ep/Pref). For each dataset the first two rows trace one user's graph after sessions 1 and 2, and the third row reports the full dataset-scale graph (totals across all users), matching Figure 5. LongMemEval's per-session count fluctuates under LLM extraction non-determinism (S2).

```
("episodic"), name, description, paths
Properties MUST include: outcome
(success/failure/partial/pending/unknown)
Properties SHOULD include: session_date,
event_date, sequence_order
LINK - Create an edge between two
existing nodes Required: from_id,
to_id, edge_label (UPPER_SNAKE_CASE)
Optional: properties (dict with user_id,
graph_type)
UPDATE_NODE - Delta-update properties
on an existing node Required: node_id,
updates (dict of key:value to change)
DELETE_NODE - Remove a node (for forget
requests or corrections) Required:
node_id
UNLINK - Remove an edge Required:
from_id, to_id, edge_label
NOOP - Explicitly skip when nothing is
worth storing Optional: reason (string)
== GUIDELINES ==
- Node types are DYNAMIC: Event,
Activity, Person, Pet, Location,
Organization, Decision, Milestone, Item,
Book, Game, etc. - Edge labels are
DYNAMIC: FOLLOWED_BY, CAUSED, LED_TO,
EXPERIENCED, OCCURRED_AT, OWNS, KNOWS,
PLAYS, READS, etc. - Create a User node
with node_id = the provided user_id -
Every Event/Activity node MUST have an
"outcome" property - Every node MUST have
a "paths" property (list of path strings
from User root) - Every node MUST have
a "description" property (1-2 sentences
capturing specific details) - Focus on
WHAT HAPPENED - not preferences (those go
in the preference graph) - Use NOOP when
a conversation chunk has no memory-worthy
events
```

```
== CRITICAL: EXTRACT FINE-GRAINED DETAILS
==
You MUST extract ALL specific names,
titles, quantities, and dates: - Named
entities: Pet names (e.g., "Susie
the snake"), book titles, game names,
restaurant names - Specific quantities:
"3 years", "10 people", "$500", "2
kids" - Specific dates: "19 January
2023", "last March", "two weeks ago" -
Relationships: "close friend", "married 5
years", "colleague"
Create SEPARATE NODES for named entities
and LINK them to the events they appear
in.
== TIMESTAMP RULES ==
Two types of timestamps - keep them
separate:
1. session_date: When this conversation
session took place (from metadata). Store
on EVERY node.
2. event_date: When the event ACTUALLY
occurred, resolved from context. - "I
went to Tokyo last March" + session_date
"May 2023" -> event_date = "March 2023"
- "yesterday" + session_date "20 October
2023" -> event_date = "19 October 2023"
- "3 years ago" + session_date "2023" ->
event_date = "2020" Store ONLY on nodes
where a date is mentioned or inferable.
== OUTPUT FORMAT ==
{"operations": [ {"operation": "ADD_NODE",
"node_id": "str", "node_type": "str",
"properties": {...}}, {"operation":
"LINK", "from_id": "str", "to_id":
"str", "edge_label": "str", "properties":
{...}}, ... ]}
```

Preference Extraction Prompt

Preference Extraction Prompt

```
You are an expert preference-memory
extraction agent for personal
conversation analysis.
YOU MUST RESPOND WITH **ONLY** VALID JSON
- ABSOLUTELY NO OTHER TEXT.
TASK: Extract a PREFERENCE MEMORY
GRAPH from a conversation using memory
operations.
You will receive: 1. A user_id
identifying the person 2. A conversation
(and optionally persona context) 3. A
list of EPISODIC OPERATIONS already
extracted (episodes/events from the same
conversation)
== CORE DESIGN PRINCIPLE ==
Behavioral signals (preferences, values,
goals, constraints) are the only nodes;
the preference graph has NO Event nodes.
Edges between signals carry one of two
label kinds: (1) a structural relation
from the closed set {PREFERS, WANTS,
VALUES, TRIED, AVOIDS, LEARNED} - e.g.
a Preference LEARNED a Constraint; or (2)
an episodic event entity_id, recording
that the relationship was observed during
that event (the event-as-edge bridge).
Chain structure: User -[PREFERS]-> P1
-[LEARNED]-> C1 (structural relation)
User -[PREFERS]-> P1 -[event_entity_id]->
P2 (event-as-edge bridge)
== OPERATIONS ==
ADD_NODE - Create a new
preference/value/goal/constraint node
Properties MUST include: user_id,
graph_type ("preference"), name,
paths, category For grounding, emit
ONLY evidence_list: the distinct
episodes supporting this signal, each
{"episode_id", "outcome"}. The evidence
vector ev(v) = (strength s, evidence
count n, outcome weight w, confidence
c) is computed deterministically from
evidence_list at aggregation time
(UPDATE_NODE); do NOT score s, w, or c
by hand.
LINK - Create an edge between two nodes
For User-to-signal: edge_label is one
of the closed structural set {PREFERS,
WANTS, VALUES, TRIED, AVOIDS, LEARNED}.
For signal-to-signal: edge_label is
either a structural relation from the
same set (e.g. LEARNED) or an episodic
event entity_id.
UPDATE_NODE - Delta-update an existing
signal (re-triggers evidence aggregation)
DELETE_NODE - Remove a signal UNLINK
- Remove an edge NOOP - Nothing
preference-worthy in this chunk
== EVIDENCE RULES ==
The agent emits only the raw grounding;
all scores are computed downstream,
not by the LLM: - evidence_list:
[{"episode_id": "str", "outcome":
"success|failure|partial|pending"}] - one
entry per distinct supporting episode.
This is the ONLY evidence field the agent
produces. - The aggregator then derives
the evidence vector deterministically:
evidence count n = |evidence_list|,
outcome weight w = (successful episodes)
/ (resolved episodes), where resolved
means marked success or failure and
partial/pending episodes count toward
n but are excluded from w (w defaults
to 0.5 when no episode is resolved),
and strength s and confidence c via
the deterministic scoring functions
(both continuous in (0,1)). These are
recomputed on every UPDATE_NODE.
NO Event/Activity nodes in the preference
graph. Those belong in episodic only.
```

Context Rendering Format

Context Rendering Template

```
=== {user_name}'s Memory ===
{user_name}'s Preferences: - [Preference]
Contrastive Pretraining - Prefers
contrastive pretraining for low-resource
retrieval (strength=0.75, evidence=2,
outcome_weight=0.50, confidence=0.62)
Evidence: first_paper_rejected
(failure), third_paper_accepted
(success) - [Constraint] Ablation
Coverage - Learned from a rejected
submission (strength=0.67, evidence=1,
outcome_weight=0.00, confidence=0.62)
{user_name}'s Events: - [Event] Tokyo
Tech Conference - Attended a tech
conference in Tokyo [date: March 2023]
[outcome: success] - [Pet] Susie - Ball
python pet snake named Susie, bought
about a year ago [date: October 2022]
[outcome: success]
```

Answer Generation Prompts The MCQ and True/False prompts below emit a discrete answer (a letter or a boolean) scored by exact match against the gold option. The LLM-as-a-judge prompt (provided in prompt library) is used only for open-ended answers, where it returns a 1–5 semantic-fidelity score against the gold reference.

MCQ Answer Prompt

```
System: Answer based on the user's
memory. Think step by step, then give
the letter.
Memory: {context}
Question: {question} Options: A: ... B:
... C: ... D: ...
Think step by step: 1. Identify what
the user's memory tells us. 2. Pick the
option that best matches.
Reasoning: <your reasoning> Answer:
<letter>
```

True/False Answer Prompt

```
System: Answer true/false based on the
user's memory. Think first.
Memory: {context}
True or false: {question}
Think step by step, then answer.
Reasoning: <your reasoning> Answer: <true
or false>
```

Open-Ended Answer Prompt

```
System: Answer based on the user's
memory. Be concise - short phrase only.
Memory: {context}
Question: {question}
Write an answer in the form of a short
phrase. Answer with exact words from the
memory whenever possible. Be concise.
Short answer:
```

Evaluation Judge Prompt

LLM AS A JUDGE – Standard

```
System: You are an expert evaluator.
Score answers 1-5. Respond with only a
number.
You are an expert evaluator assessing the
semantic fidelity of a memory retrieval
system.
Question: {question} Gold Reference:
{gt_answer} Candidate Answer: {answer}
Score the Candidate Answer against the
Gold Reference on a scale of 1-5: 5
= Exact alignment - captures all key
entities, temporal markers, and causal
relationships. Semantically equivalent.
4 = Substantially correct - main point
is accurate but lacks minor nuances or
secondary details. 3 = Partial match -
contains valid information but misses
key constraints (e.g., wrong date
but correct event). 2 = Tangential -
touches on the topic but misses the
core information requirement. 1 =
Contradiction/Hallucination - completely
unrelated or contradicts the ground
truth.
Evaluation constraints: 1. Temporal
flexibility: Accept relative time
references if they resolve to the same
period as the Gold Reference. 2. Semantic
equivalence: Prioritize informational
content over lexical matching.
Respond with ONLY a number 1-5.
```