

Memory Beyond Recall: A Dual-Process Cognitive Memory System for Self-Evolving LLM Agents

Tianxiang Fei, Mingyang Song, Mao Zheng, and Xiang Yu

Foundation Model Department, Tencent, China

alvinfei@tencent.com

Abstract

Long-term memory for an LLM agent requires more than retrieving a relevant passage. Many existing systems address belief revision, causal connections and cross-domain abstraction primarily through storage and retrieval mechanisms, limiting their ability to support implicit personalisation as a user’s preferences evolve. We present DPCM, which organises memory into a capability hierarchy spanning raw inputs, atomic facts, belief trajectories, identity, domain schemas, prospective intentions and cross-domain patterns. A synchronous daytime writer (SYSTEM 1) records revisions as doubly linked supersedes chains; an asynchronous nighttime engine (SYSTEM 2) induces schemas and intentions, then uses cross-domain collisions to derive core schemas. Dual-process theory motivates this separation between online encoding and offline abstraction without implying that agent memory reproduces human cognition. Across single-run evaluations on LongMemEval, PersonaMem and PersonaMem-v2, SYSTEM 2 yields its largest observed gain on implicit cross-session inference (+5.20 on PersonaMem-v2) and its smallest on the recall-oriented LongMemEval aggregate (+0.33). When the inference budget is limited to 8k tokens and the full history no longer fits, DPCM_{full} exceeds the long-context baseline by 10.5–17.9 points across both backbones.

1 Introduction

After weeks of interaction, an LLM agent may accumulate a detailed record of a user’s statements, actions and preferences yet still answer personalisation queries incorrectly. Existing memory systems provide complementary mechanisms for managing this record: context paging, extracted-memory updates, temporal knowledge graphs and evolving stores (Packer et al., 2023; Chhikara et al., 2025; Rasmussen et al., 2025; Zhang and Wang,

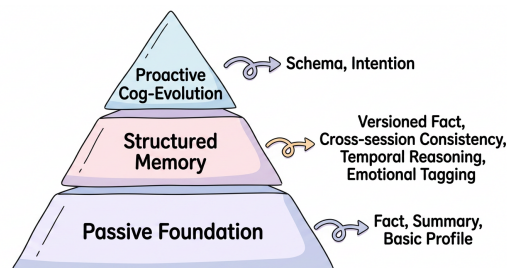


Figure 1: Conceptual hierarchy for LLM-agent memory: a passive foundation of facts, summaries and basic-profile attributes; structured memory for versioned facts and cross-session reasoning; and proactive cognitive evolution through schemas and intentions. DPCM additionally retains raw inputs and induces cross-domain core schemas. “Emotional Tagging” is conceptual rather than a separately evaluated module. Summary nodes were inactive, and proactive initiation was not evaluated.

2026). These mechanisms improve access to prior interactions, but do not by themselves jointly preserve fine-grained belief trajectories and induce behavioural abstractions across otherwise unrelated life domains. We study whether separating these functions can improve long-horizon personalisation.

Figure 1 presents our alternative as a conceptual progression from a passive foundation, through structured memory, to proactive cognitive evolution. DPCM operationalises this hierarchy with retained raw inputs and atomic facts, separately maintained basic-profile attributes, identity items and diachronic supersedes chains, followed by within-domain schemas, prospective intentions and cross-domain core schemas. Work on Theory of Mind describes how representations of others become increasingly structured (Wellman, 2014), while dual-process accounts distinguish rapid responses from more deliberative reasoning (Kahneman, 2011; Evans and Stanovich, 2013). These ideas serve as engineering analogies for separating online encoding from offline abstraction; they do not entail

our hierarchy or imply that agent memory reproduces human cognition.

We instantiate the hierarchy as DPCM, with two processes that inherit only the architectural distinction of dual-process theory. A synchronous *daytime writer* (SYSTEM 1) reconciles each requested write against the vector store and records belief revisions as doubly linked supersedes chains. An asynchronous *nighttime engine* (SYSTEM 2) clusters fresh facts, induces schemas and intentions over a graph, and sweeps for cross-domain collisions whose behavioural similarity is high but semantic similarity is low, abstracting them into higher-level core schemas.

Architectural hypothesis. Separating online encoding from offline abstraction predicts an asymmetric contribution profile: SYSTEM 2 should help less on recall-oriented evaluation than on implicit cross-session inference. The single-run results in Table 2 follow this pattern (+0.33 on LongMemEval versus +5.20 on PersonaMem-v2 under kimi-2.5); Section 3.5 examines it in more detail. The architecture differs from a generic summarisation pipeline in three concrete ways: supersedes chains retain the intermediate states of a revised belief; the dual-space test (Eq. (1)) links schemas that are behaviourally close but semantically distant; and the read path retrieves approximately 0.85k tokens without invoking an LLM, moving abstraction cost offline.

Contributions. We (i) organise long-term agent memory using a *cognitive capability hierarchy* inspired by Theory-of-Mind and dual-process accounts; (ii) introduce a synchronous writer that records belief-evolution trajectories as doubly linked supersedes chains over a flat vector store, with a read path that requires only embedding lookup and pointer traversal; and (iii) design an asynchronous abstraction engine whose induction step and cross-domain sweeper jointly populate schema and intention layers and surface latent cross-domain patterns. We instantiate the design as DPCM_{lite} (SYSTEM 1 only) and DPCM_{full} (with SYSTEM 2), and evaluate them on LongMemEval, PersonaMem and PersonaMem-v2, with stage-wise ablations on PersonaMem-v2.

2 Method

Figure 2 summarises the architecture. The synchronous SYSTEM 1 writer (left) responds to ex-

plicit add_memory requests: it first persists the raw input, then writes reconciled facts and identity items, including bidirectional pointers between revisions. The asynchronous SYSTEM 2 engine (right) runs during idle or nightly windows. It clusters fresh facts without an LLM, uses an induction agent M_a to update the graph $\mathcal{G}$, and derives core schemas from pairs that are close in behavioural space but distant in semantic space. At query time, DPCM traverses the vector store, supersedes pointers and $\mathcal{G}$ without an LLM. Sections 2.1–2.5 describe the hierarchy, both processes, the read path and complexity; Appendix A gives pseudocode.

2.1 Cognitive Capability Hierarchy

DPCM organises memory along the three conceptual tiers in Figure 1; these tiers group capabilities rather than map one-to-one to storage types. The *passive foundation* contains facts, the reserved Summary type and basic-profile attributes. In the implementation, raw inputs are also persisted in a vector store before extraction so they remain available if downstream processing fails, while stable attributes such as name, age, location and occupation are handled by a separate profile updater. *Structured memory* is realised by facts and identity items carrying SUPERSEDES/SUPERSEDED BY pointers, which form doubly linked version histories that support cross-session consistency and temporal reasoning. The figure’s emotional-tagging label denotes a conceptual capability, not a separate node type or evaluated module. The *proactive cognitive-evolution* tier lives in a separate graph $\mathcal{G}$ and contains two co-equal kinds of nodes: *schemas* summarise observed behavioural regularities within a single life domain, while *intentions* encode concrete prospective actions or events and are intended to support proactive use, an affordance we do not evaluate here; intentions sit beside, not above, schemas. A cross-domain sweeper induces higher-level core schemas on top of within-domain schemas. Graph nodes connect to supporting VDB fact IDs through DERIVEDFROM edges. In total the reported system writes facts, identity items, schemas (within-domain basic schemas and cross-domain core schemas), and intentions.

Summary nodes. DPCM reserves a *Summary* type for session-level compression of diffuse inputs. Summary nodes share the vector store and hybrid lookup used by facts and identity items, but this type was inactive in all reported experiments and

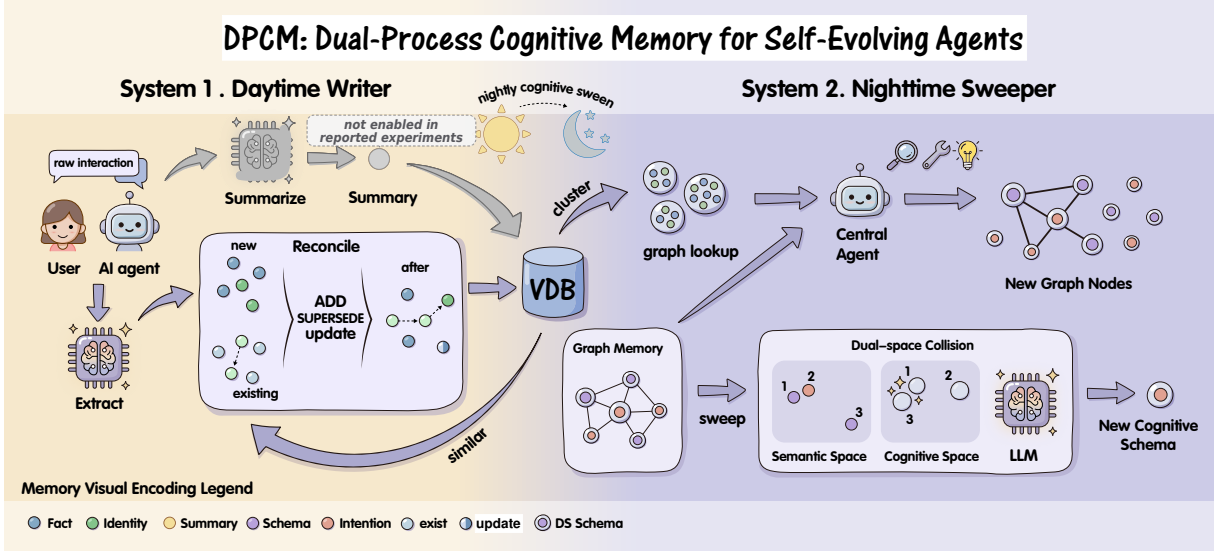


Figure 2: Overview of DPCM. A synchronous SYSTEM 1 daytime writer (left) handles on-demand add_memory requests, while an asynchronous SYSTEM 2 nighttime engine (right) induces schemas, intentions and cross-domain core schemas. The vector database (centre) holds raw inputs, facts and identity items. “Summary” denotes a declared SYSTEM 1 session-level type, not a SYSTEM 2 product; it was not enabled in the reported experiments. See Section 2 for details.

absent from the populated stores. The Summary entries in Figures 1 and 2 therefore describe an architectural extension.

2.2 System 1: Synchronous Daytime Writer

SYSTEM 1 runs synchronously on demand rather than on every dialogue turn. Short-term context is already available in the model window, so memory is persisted only when the user or application issues an explicit add_memory request. The writer then executes four latency-sensitive steps.

(i) **Raw persist.** Before any LLM call, the raw input is written to the raw-input vector store as the authoritative record, reducing the risk that a downstream extraction failure loses user content. It remains unchanged unless the user requests deletion.

(ii) **Extraction.** In one call, an extraction LLM M_e reads the current turn and a short recent-history window and emits only *identity items* (preferences, attitudes and related identity information) and *atomic facts* (events, experiences, plans and other factual statements). Each item carries topic tags. Stable structured profile fields are handled separately as described in Section 2.1. We use deepseek-v4-flash or kimi-2.5 as M_e , both in the no-think setting; temperature is 0 except for kimi-2.5 ($\tau=0.6$, its default).

(iii) **Reconciliation.** A single batched call to M_r receives all extracted items together with their top-

k neighbours from the vector store and assigns one of three operations where needed: ADD inserts a novel item; UPDATE is instructed to preserve compatible new and old information in a specified existing node, without creating a revision node; and SUPERSEDE inserts a new node for a contradictory or evolved state while retaining the specified predecessor. Exact duplicates receive no operation and are discarded.

(iv) Apply operations and supersedes chain.

The writer batch-embeds new content and applies each operation according to its contract. ADD inserts a new node, whereas UPDATE updates the targeted existing node in place with a synthesis intended to preserve both sources. SUPERSEDE inserts a new node and adds two pointers, $n_{\text{new}}.\text{SUPERSEDES}=n_{\text{old}}$ and $n_{\text{old}}.\text{SUPERSEDED BY}=n_{\text{new}}$, without deleting the predecessor. The resulting chain is a doubly linked list of revisions over a flat vector store. Retaining predecessors distinguishes DPCM from update-in-place systems and preserves diachronic continuity: a read-time hit triggers an $O(\ell)$ traversal of a chain of length ℓ , recovering intermediate states that embedding similarity may miss.

2.3 System 2: Asynchronous Nighttime Engine

SYSTEM 2 runs *asynchronously* on a schedule (idle periods or nightly; once every five writes in our

experiments, see Section 3.1), with a much looser latency budget (tens of seconds). It operates over the stores accumulated by SYSTEM 1 and writes back into the graph $\mathcal{G}$ in three phases.

Phase 1 (Preprocess, no LLM). A *fact triage* selects items not yet incorporated into any schema or intention node, then clusters them with two-stage DBSCAN (an initial pass over semantic embeddings, followed by a tighter pass that subdivides over-sized clusters). Facts left unclustered are retained as singleton inputs rather than dropped. For each clustered or singleton input, we compute its embedding representation and retrieve *graph context* in both directions: the nearest existing schema and intention nodes, and any nodes whose DERIVEDFROM edges already reference the input’s VDB fact IDs. Phase 1 makes no LLM calls.

Phase 2 (LLM-agent induction). An induction agent M_a is invoked once per clustered or singleton input. Four classes of write operation are exposed through three function-calling tools. `create_graph_node` creates either a within-domain behavioural schema or an intention, selected by its layer argument; an intention is a concrete prospective action or event. `add_evidence` links VDB fact IDs to an existing schema or intention through DERIVEDFROM edges. `add_edge` writes only RELATEDTO relationships between graph nodes. All actions persist to $\mathcal{G}$ in emission order, so the graph evolves incrementally across nights rather than being rebuilt.

Phase 3 (Cross-domain sweeper). After per-cluster induction, the sweeper runs in three steps. (a) *Behavioural abstraction*: for each schema s , an LLM produces a short behavioural description (e.g. “*escalates commitment after sunk costs*”). The bge-m3 sentence encoder (Chen et al., 2024) embeds this description in behavioural space E_{beh} and the original schema text in semantic space E_{sem} . (b) *Collision scan*: the sweeper enumerates cross-domain schema pairs (s_i, s_j) and flags a collision when

$$\begin{aligned} \cos(E_{\text{beh}}(s_i), E_{\text{beh}}(s_j)) &> \theta_{\text{beh}}, \\ \cos(E_{\text{sem}}(s_i), E_{\text{sem}}(s_j)) &< \theta_{\text{sem}}. \end{aligned} \quad (1)$$

Collision pairs are aggregated with union-find, and trivially redundant within-domain pairs (high in both spaces) are filtered out. (c) *Core induction*: for each collision cluster, a final LLM call M_c takes the member schemas and either returns a higher-level *core schema* or null. A core schema and its edges to all member schemas are inserted into

$\mathcal{G}$ only when the return value is non-null. Core schemas serve as the most abstract retrieval targets.

2.4 Read Path and Latency

At inference time, a query embedding retrieves candidates from the fact and identity stores and traverses attached supersedes chains. From matched VDB fact IDs, it follows DERIVEDFROM edges to schema and intention nodes, then schema-to-core links to core schemas, stopping at the configured 20-node limit. The memory read path invokes no LLM: vector search dominates its measured latency of 153ms per query, and it supplies an average of $\sim 0.85\text{k}$ tokens to the inference model (Table 1). This payload is set by the retrieval configuration rather than by history length.

2.5 Complexity

Let N be the vector-store size, a the candidates from one write, F the fresh facts in a digest, N_s the schema nodes, K the collision clusters, $\bar{\ell}$ the mean chain length and r the graph edges traversed for a query. Under the usual average-case scaling of an HNSW index, SYSTEM 1 performs a nearest-neighbour lookups in $O(a \log N)$, followed by one extraction and one batched reconcile call. Two-stage DBSCAN is $O(F \log F)$ with a spatial index and $O(F^2)$ in the worst case; induction then makes one M_a call per cluster or unclustered singleton. The cross-domain sweep requires up to $O(N_s)$ behavioural-abstraction calls, $O(N_s^2)$ pairwise comparisons and one M_c call per collision cluster. Read time is $O(\log N + \bar{\ell} + r)$ on average for approximate lookup, chain traversal and evidence-linked graph expansion, with no LLM in the memory read path. Storage is $O(N + N_s + |E|)$ for vector items, graph nodes and edges; supersedes chains add pointers between existing items rather than duplicate nodes.

3 Experiments

3.1 Setup

We follow the no-think evaluation protocol throughout. Extraction and inference models are paired identically across systems for fairness, with $M_e, M_i \in \{\text{deepseek-v4-flash}, \text{kimi-2.5}\}$. The reconcile LLM M_r shares the same backbone as M_e . LongMemEval responses are judged by gemini-3.0-flash using the task-specific judge template of Wu et al. (2025).

Benchmarks. (1) **LongMemEval** (Wu et al., 2025), dataset S, evaluates recall, reasoning over time, knowledge update and abstention. (2) **PersonaMem (32k)** (Jiang et al., 2025a) evaluates persona-consistent response selection across multi-session histories. (3) **PersonaMem-v2 (32k)** (Jiang et al., 2025b) stresses *implicit* preference inference over 32k-token contexts. The original report gives 37–48% accuracy for its evaluated LLMs, making it the most discriminative of the three benchmarks.

Baselines. (B1) **Long-context agent:** under each evaluated backbone, the full conversation history is placed in its context window subject to model truncation. (B2) **Mem0** (Chhikara et al., 2025), evaluated with and without its graph backend. (B3) **Zep/Graphiti** (Rasmussen et al., 2025), the underlying temporal knowledge graph. We discuss concurrent self-evolving writers (Zhang and Wang, 2026; Feng et al., 2026; Liu et al., 2026) as related work in Section 4. We compare two configurations of our system. DPCM_{lite} uses only the SYSTEM 1 daytime writer (raw, fact and identity stores with supersedes chains), and DPCM_{full} additionally enables the SYSTEM 2 nighttime engine (schemas, intentions, and the cross-domain core-schema layer).

Diagnostic alignment. The three benchmarks emphasise different capabilities rather than isolating them perfectly. LongMemEval’s *knowledge-update* subset (Wu et al., 2025) probes whether a system recovers the latest state after revisions, making it relevant to **diachronic continuity**; supersedes chains retain the preceding states as provenance while exposing the latest state. It also tests **causal coupling** when the revising turn is semantically distant from the original topic. PersonaMem-v2’s *implicit-preference* split (Jiang et al., 2025b) probes **inductive abstraction**: answers are not stated directly and must be inferred from behavioural regularities across sessions.

Implementation details. The vector store $\mathcal{V}$ is backed by Qdrant with HNSW indexing ($m=32$, $ef=128$). All embeddings use bge-m3 (Chen et al., 2024) at 1024 dimensions, with a hybrid retrieval score that linearly interpolates dense cosine similarity and sparse BM25 ($\alpha=0.7$). The top- k retrieval budget for the reconcile step is $k=5$. For SYSTEM 2, two-stage DBSCAN uses $\varepsilon_1=0.35$ on semantic embeddings for the initial pass and $\varepsilon_2=0.25$ for the tighter subdivision, with $\text{min_samples}=3$. The behavioural encoder E_{beh} and the semantic en-

Component	LLM calls	Tokens	Latency
S1 write / add_memory	2	4.23k	23.1s
S2 digest (amortised)	0.89	1.28k	5.16s
Write path total	2.89	5.51k	28.3s
Memory retrieval only	0	$\sim 0.85\text{k}$	153ms

Table 1: Measured memory-system cost on PersonaMem-v2 with kimi-2.5. S2 is amortised per write at one digest per five writes (25.8s per digest). Retrieval excludes the shared answer-generation call to M_i .

coder E_{sem} share the same bge-m3 backbone but operate on different inputs: E_{beh} encodes the LLM-generated behavioural description while E_{sem} encodes the original schema text. The collision thresholds are $\theta_{\text{beh}}=0.72$ and $\theta_{\text{sem}}=0.40$, tuned on a held-out split of PersonaMem-v2. Appendix B lists the full hyperparameter set in one place.

SYSTEM 2 scheduling protocol. Section 2.3 describes SYSTEM 2 as running in idle or nightly windows; we state the concrete protocol used in all reported experiments. A SYSTEM 2 digest is triggered **once every five add_memory writes**. The digest executes offline and outside the query path, so query latency excludes digest time. The same schedule applies across benchmarks and backbones, and the number of digests per user is determined by the number of writes.

Cost. Table 1 reports measured memory-system cost on PersonaMem-v2 with kimi-2.5. The online write path issues two LLM calls per add_memory: one extraction call and one batched reconcile call. A SYSTEM 2 digest takes 25.8s; at the 1-in-5 schedule, its measured calls and latency amortise to 0.89 and 5.16s per write. Memory retrieval itself invokes no LLM and supplies an average of $\sim 0.85\text{k}$ tokens to M_i in 153ms. This row excludes the subsequent answer-generation call to M_i , which is shared by all systems. Under this fixed retrieval configuration, the observed payload is bounded by the node cap rather than by the full accumulated history.

3.2 Main Results

Table 2 reports accuracy on the three benchmarks under two matched extraction/inference backbones, deepseek-v4-flash and kimi-2.5, both in no-think mode.

On PersonaMem-v2, DPCM_{full} achieves the best kimi-2.5 result among the evaluated sys-

System	PersonaMem (32k)		LongMemEval (S)		PersonaMem-v2 (32k)	
	deepseek	kimi	deepseek	kimi	deepseek	kimi
B1 Long-Ctx Agent	72.07	77.59	88.43	80.61	50.82	58.46
B2 Mem0	69.07	65.23	56.84	48.87	42.53	41.25
+ graph	68.82	65.82	57.14	51.26	43.28	40.16
B3 Zep / Graphiti	58.93	64.21	67.83	62.35	39.54	38.27
DPCM _{lite} (S1)	69.15	74.32	84.73	85.81	46.85	54.10
DPCM _{full} (S1+S2)	70.46	76.73	85.17	86.14	49.36	59.30

Table 2: Main results. Accuracy (%) on three long-horizon memory benchmarks under two extraction/inference model pairs (deepseek-v4-flash and kimi-2.5, both no-think). Best result per column in bold; our two DPCM configurations are highlighted. PersonaMem spans several persona-consistent response-selection tasks, while PersonaMem-v2 places greater emphasis on implicit persona inference (Jiang et al., 2025a,b).

tems and approaches the long-context baseline under deepseek-v4-flash. Enabling SYSTEM 2 (DPCM_{full} versus DPCM_{lite}) yields the largest observed gain on PersonaMem-v2 (+2.51/+5.20), a smaller gain on PersonaMem (+1.31/+2.41), and little change on LongMemEval (+0.44/+0.33). The long-context agent remains strongest in four of the six columns. Section 3.3 examines how this comparison changes when the available context is reduced.

3.3 Inference Budget and Applicability Boundary

Table 2 gives B1 the full session, its most favourable setting. We therefore vary its inference-time budget. In every condition, DPCM_{full} builds its store from the complete session; B1 receives either the full session or only the most recent N tokens ($N \in \{8k, 16k\}$) at inference time. The “B1 full” column reproduces Table 2. DPCM_{full} always reads the complete structured store and supplies $\sim 0.85k$ retrieved tokens per query (Table 1), so it does not vary with B1’s context budget.

B1 degrades monotonically as N shrinks (PersonaMem-v2 under kimi-2.5: 58.46 $\rightarrow$ 51.7 $\rightarrow$ 46.2), while DPCM_{full} is evaluated with a fixed retrieval setting. In the four conditions where B1 leads at full budget, the ordering reverses by 16k; in the other two, DPCM_{full} already leads and its margin widens. At 8k, DPCM_{full} leads on all three benchmarks under both backbones by a double-digit margin (+10.5 to +17.9).

Applicability boundary. When the full history fits inside the effective context window, B1 is a strong baseline and wins or ties four of the six columns in Table 2. The results instead indicate two regime-specific advantages. When the history

exceeds the usable context budget, a structured store retains information lost to truncation (Table 3); on implicit-inference evaluation, the abstraction layers also contribute at full budget (+5.20 from SYSTEM 2 on PersonaMem-v2 versus +0.33 on LongMemEval). Under kimi-2.5, DPCM_{full} also leads on LongMemEval with the full session available (86.14 versus 80.61), which is consistent with long-context position sensitivity rather than truncation (Section 3.5).

3.4 Ablation

We ablate the SYSTEM 2 engine on PersonaMem-v2 using kimi-2.5 as both M_e and M_i . Figure 3 reports the result alongside the long-context baseline B1 (58.46, the Table 2 value) for the same model pair, obtained by feeding the full session history directly to the inference model. The full DPCM_{full} system exceeds this baseline by 0.84 points, and removing successive cognitive components degrades performance monotonically.

Two patterns emerge from this removal order. Disabling dual-space collision first reduces accuracy by 4.10 points; subsequently removing schema and intention nodes yields a further 1.10-point drop. Removing supersedes chains after all of SYSTEM 2 is disabled costs another 2.90 points. These are conditional marginal drops, not independent additive effects: 4.10 for collision given the remaining pipeline, 1.10 for schema/intention after collision is absent, and 2.90 for supersedes after SYSTEM 2 is absent.

3.5 Analysis

SYSTEM 2 contribution profile. Table 4 gives a coarse benchmark-level comparison rather than an item-level task decomposition; each benchmark mixes multiple capabilities. In the re-

Benchmark	kimi-2.5					deepseek-v4-flash				
	B1@8k	B1@16k	B1 full	DPCM _{full}	Δ @8k	B1@8k	B1@16k	B1 full	DPCM _{full}	Δ @8k
PersonaMem	60.9	73.0	77.59	76.73	+15.8	54.4	64.8	72.07	70.46	+16.1
PersonaMem-v2	46.2	51.7	58.46	59.30	+13.1	38.9	43.5	50.82	49.36	+10.5
LongMemEval	68.2	74.1	80.61	86.14	+17.9	69.7	78.9	88.43	85.17	+15.5

Table 3: Accuracy (%) as the inference-time budget of the long-context baseline B1 shrinks. B1@ N receives the most-recent N tokens; “B1 full” is the untruncated Table 2 setting (32k for the PersonaMem benchmarks, ~ 115 k for LongMemEval-S). DPCM_{full} uses the same fixed retrieval cap in every condition and injects an average of ~ 0.85 k tokens per query. Δ @8k is DPCM_{full} minus B1@8k. Best per backbone-benchmark pair in bold.

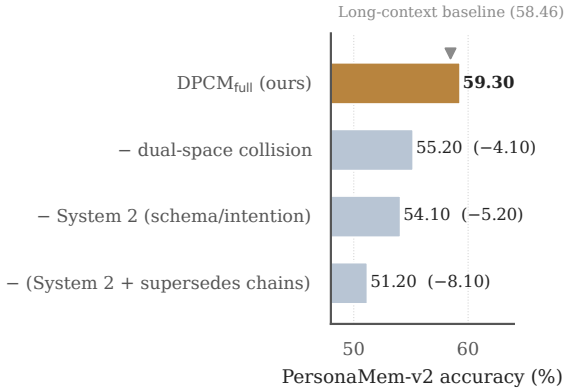


Figure 3: Cumulative ablation on PersonaMem-v2 with kimi-2.5. Mechanisms are removed successively from top to bottom, so adjacent differences are order-dependent conditional drops. The triangle marks B1 (58.46; Table 2); parentheses give cumulative drops from DPCM_{full}.

ported runs, the SYSTEM 2 gain increases across the three benchmark groups under both backbones: $+0.44 \rightarrow +1.31 \rightarrow +2.51$ with deepseek-v4-flash and $+0.33 \rightarrow +2.41 \rightarrow +5.20$ with kimi-2.5. In the cumulative ablation, removing cross-domain collision first lowers PersonaMem-v2 accuracy by 4.10 points (Figure 3). These single-run, benchmark-level observations are consistent with the proposed asymmetry but do not constitute an item-level causal test.

Why can structured memory help at full context? Under kimi-2.5, B1 falls 5.53 points below DPCM_{full} on LongMemEval (80.61 vs. 86.14). This gap is consistent with, but does not by itself establish, lost-in-the-middle degradation (Liu et al., 2024): structured retrieval presents a shorter, focused evidence set, whereas B1 must attend over the full history.

Failure modes. Error analysis on PersonaMem-v2 shows that supersedes chains can over-fire when

Dominant demand	Setting	vs. B1	S2 gain
LongMemEval	LME (ds)	B1 +3.26	+0.44
	LME (ki)	ours +5.53	+0.33
PersonaMem	PM (ds)	B1 +1.61	+1.31
	PM (ki)	B1 +0.86	+2.41
PersonaMem-v2	PM-v2 (ds)	B1 +1.46	+2.51
	PM-v2 (ki)	ours +0.84	+5.20

Table 4: Benchmark-level comparison. “vs. B1” compares DPCM_{full} with the full-budget baseline; “S2 gain” is DPCM_{full} minus DPCM_{lite}. This is not an item-level capability stratification. ds/ki = deepseek-v4-flash/kimi-2.5.

the extraction or reconciliation model treats a transient state as a stable revision; performance is also sensitive to θ_{beh} . In the manual audit of Appendix C, at least 97% of sampled schema nodes passed the content-faithfulness criterion, whereas approximately 82% of audited evidence links were judged correct. Evidence attribution is therefore the larger observed source of graph-construction error.

Supersedes chain statistics. Across the PersonaMem-v2 sessions, SYSTEM 1 writes an average of 1.8 supersedes chains per user (range 0–7), with a mean chain length of $\bar{\ell}=2.4$ revisions and a maximum of 5. On the knowledge-update subset of LongMemEval, 78% of correctly answered queries touch at least one supersedes pointer during read-time traversal, compared with 31% of incorrect ones. This association is consistent with chain traversal helping recover revised belief states, but does not establish causality. Identity items are rarely superseded ($< 3\%$ carry a pointer); most revisions concern factual claims about preferences and habits.

3.6 Qualitative Illustration

We give two excerpts from PersonaMem-v2. **Supersedes chain (fitness):** a user’s stance on group fitness passes through five recorded states across seven months, from n_1 “*finds working out with others a motivator that pushes them to new physical milestones*” through solitary and renewed group-fitness phases to n_5 “*finds hiking more fulfilling than fitness classes, prefers welcoming interactive environments over competitive ones.*” An update-in-place writer retains only n_5 ; DPCM preserves and retrieves the full revision chain. **Cross-domain collision:** a wellness schema and a gardening/teaching schema are distant in E_{sem} but close in E_{beh} (0.18 versus 0.81), so they satisfy Eq. (1). Both describe patient, process-oriented investment despite sharing little surface vocabulary. The resulting core schema summarises the common pattern as patient, consistent investment over time rather than forced acceleration. The example illustrates the abstraction produced by the collision step; it does not establish the correctness of the inferred user trait.

4 Related Work

Existing agent-memory systems differ mainly in how they store and update experience. MemGPT (Packer et al., 2023) manages context as an OS-like page cache; Mem0 (Chhikara et al., 2025) updates extracted memories in place and optionally links them in a graph; Zep (Rasmussen et al., 2025) represents temporal knowledge as a graph. A-MEM (Xu et al., 2025), Memory-Bank (Zhong et al., 2024) and HippoRAG (Gutiérrez et al., 2024) instead draw on Zettelkasten, forgetting curves and hippocampal indexing. Recent work extends different parts of the memory pipeline: TSUBASA uses dynamic memory writing (Zhang and Wang, 2026), SEARL jointly optimises a policy and tool-graph memory (Feng et al., 2026), FileGram organises file-system behavioural traces (Liu et al., 2026), and the Experience Compression Spectrum analyses memory, skills and rules as forms of compressed experience (Zhang et al., 2026). DPCM focuses on preserving revision paths while inducing abstractions above the fact layer.

Retrieval-augmented generation (Lewis et al., 2020; Borgeaud et al., 2022; Gao et al., 2023) and long-context models (Grattafiori et al., 2024) address the related problem of selecting evidence from a largely static corpus. Long-horizon mem-

ory is evaluated by benchmarks such as LongMemEval (Wu et al., 2025), LoCoMo (Maharana et al., 2024), PersonaMem (Jiang et al., 2025a,b), PerLTQA (Du et al., 2024), MSC (Xu et al., 2022) and LMEB (Zhao et al., 2026).

Agent architectures use several forms of retained experience: Generative Agents combine a memory stream with reflection and planning (Park et al., 2023), Voyager stores executable skills (Wang et al., 2024), Reflexion keeps verbal reflections in episodic memory (Shinn et al., 2023), and CoALA provides an explicit taxonomy of working and long-term memory types (Sumers et al., 2024). We instead study levels of abstraction within long-term personal memory. DPCM shares a wake-sleep separation with Xie (2026), but its offline process operates on consolidated facts and explicitly searches for cross-domain behavioural correspondences. The hierarchy is informed by Theory-of-Mind (Wellman, 2014; Apperly, 2012), dual-process theory (Kahneman, 2011; Evans and Stanovich, 2013), reconstructive memory (Bartlett, 1932; Schacter et al., 2012) and the episodic-semantic distinction (Tulving, 1972).

5 Conclusion

We presented DPCM, a long-term memory system that separates online recording from offline abstraction. SYSTEM 1 preserves revisions to facts and identity items through bidirectional supercedes pointers. SYSTEM 2 induces within-domain schemas, prospective intentions and cross-domain core schemas, while the query-time read path remains LLM-free. In the reported runs, SYSTEM 2 contributes most on implicit personalisation and least on the recall-oriented LongMemEval aggregate; DPCM_{full} gains a larger advantage over full-history context when the inference budget falls below the history length. These results motivate treating abstraction as a distinct offline memory process.

Limitations

Evaluation scope. All three benchmarks are English-language and consist of *passive* question answering or response selection: a query arrives and the system answers it. None involves tool use, multi-step action, or environment interaction. We therefore position DPCM as a memory *substrate* rather than as an agent architecture. It can be combined with tool-using agents, but we present no

evidence about how the hierarchy behaves when memory is consumed by an action policy instead of an answer generator. Cross-lingual transfer, embodied settings, and settings in which the user’s identity is itself uncertain are likewise unstudied.

Proactive care and intentions are not evaluated end to end. The intention layer represents events a user may face, but all three benchmarks are passive QA or response-selection tasks and therefore do not test proactive initiation. Moreover, removing schemas and intentions together after disabling collision yields a conditional ≈ 1.1 -point drop (Figure 3); because both node types are removed at once, this experiment does not isolate the contribution of intentions. The proactive use of intention nodes should therefore be understood as an architectural affordance rather than a demonstrated capability.

Evidence-linking accuracy. The ADD_EVIDENCE tool attributes supporting facts to graph nodes at $\sim 82\%$ accuracy in our manual audit (Appendix C), against $\geq 97\%$ for schema content. These errors weaken provenance and may degrade retrieval because graph nodes are reached through matched fact IDs. The single-annotator audit covers 150 nodes from PersonaMem-v2 and reports no agreement statistic, so its rates are descriptive.

Backbone coverage and variance. Results cover two no-think backbones, deepseek-v4-flash and kimi-2.5, which is insufficient to separate model-scale from model-family effects. We report point estimates; because kimi-2.5 uses temperature 0.6, sub-point differences should not be interpreted as statistically significant without repeated runs. Whether the asymmetric SYSTEM 2 profile persists across broader model families is left to future work.

Design and representation. The asynchronous SYSTEM 2 engine introduces an offline cost that grows with the size of the schema store, and the dual-space thresholds (θ_{beh} , θ_{sem}) are set by held-out tuning rather than learned. The hierarchy encodes our inductive bias that some behavioural patterns recur across a user’s life domains; for users whose history is dominated by a single domain the cross-domain core schemas may give limited benefit, and in settings with high domain diversity the quadratic dual-space sweep becomes the binding cost. Schema and intention nodes are written in natural language, which inherits the verbosity

and inconsistency of the induction model’s output. While interpretable, natural-language nodes are not amenable to algebraic operations such as set intersection over preference vectors. Pairing natural-language summaries with structured slots could address this limitation without changing the hierarchy.

Ethics Statement

The proposed architecture stores progressively deeper inferred information about a user, including induced cognitive patterns the user may not have explicitly disclosed. While this enables richer personalisation, it also raises privacy and consent considerations. Any deployment should expose the schema, intention and core-schema nodes to the user for inspection and removal, and should provide a hard-delete path that cascades through the supersedes chains. The benchmarks used in this work are public. LongMemEval-S mixes synthetic sessions with filtered public ShareGPT and UltraChat conversations; PersonaMem and PersonaMem-v2 use benchmark-generated persona histories. No new user data was collected for this study.

A second consideration concerns the asymmetry of inferred knowledge. Because core schemas can synthesise information across domains the user never explicitly linked, the system may surface inferences the user finds invasive even though every individual input was voluntarily disclosed. We recommend that production deployments offer per-layer opt-in granularity, with the cross-domain core-schema layer disabled by default and enabled only after the user reviews the currently inferred patterns and gives explicit consent. We also caution against using core-schema patterns for any inferential task with high-stakes consequences (e.g. access decisions, eligibility scoring), since the patterns reflect statistical regularities in conversational behaviour rather than verified ground truth.

References

- Ian A. Apperly. 2012. What is “theory of mind”? concepts, cognitive processes and individual differences. *Quarterly Journal of Experimental Psychology*, 65(5):825–839.
- Frederic C. Bartlett. 1932. *Remembering: A Study in Experimental and Social Psychology*. Cambridge University Press.
- Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George

- van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2022. [Improving language models by retrieving from trillions of tokens](#). In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.
- Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. [M3-Embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.
- Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. [Mem0: Building production-ready AI agents with scalable long-term memory](#). In *ECAI 2025*, pages 2993–3000. IOS Press.
- Yiming Du, Hongru Wang, Zhengyi Zhao, Bin Liang, Baojun Wang, Wanjun Zhong, Zezhong Wang, and Kam-Fai Wong. 2024. [PerLTQA: A personal long-term memory dataset for memory classification, retrieval, and fusion in question answering](#). In *Proceedings of the 10th SIGHAN Workshop on Chinese Language Processing (SIGHAN-10)*, pages 152–164, Bangkok, Thailand. Association for Computational Linguistics.
- Jonathan St. B. T. Evans and Keith E. Stanovich. 2013. [Dual-process theories of higher cognition: Advancing the debate](#). *Perspectives on Psychological Science*, 8(3):223–241.
- Xinshun Feng, Xinhao Song, Lijun Li, Gongshen Liu, and Jing Shao. 2026. [SEARL: Joint optimization of policy and tool graph memory for self-evolving agents](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24518–24535. Association for Computational Linguistics.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. [Retrieval-augmented generation for large language models: A survey](#). *arXiv preprint arXiv:2312.10997*.
- Aaron Grattafiori et al. 2024. [The Llama 3 herd of models](#). *arXiv preprint arXiv:2407.21783*.
- Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. [HippoRAG: Neurobiologically inspired long-term memory for large language models](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 59532–59569.
- Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J. Taylor, and Dan Roth. 2025a. [Know me, respond to me: Benchmarking LLMs for dynamic user profiling and personalized responses at scale](#). In *Second Conference on Language Modeling*.
- Bowen Jiang, Yuan Yuan, Maohao Shen, Zhuoqun Hao, Zhangchen Xu, Zichen Chen, Ziyi Liu, Anvesh Rao Vijjini, Jiashu He, Hanchao Yu, Radha Poovendran, Gregory Wornell, Lyle Ungar, Dan Roth, Sihao Chen, and Camillo Jose Taylor. 2025b. [PersonaMem-v2: Towards personalized intelligence via learning implicit user personas and agentic memory](#). *arXiv preprint arXiv:2512.06688*.
- Daniel Kahneman. 2011. *Thinking, Fast and Slow*. Farrar, Straus and Giroux.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-augmented generation for knowledge-intensive NLP tasks](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. [Lost in the middle: How language models use long contexts](#). *Transactions of the Association for Computational Linguistics*, 12:157–173.
- Shuai Liu, Shulin Tian, Kairui Hu, Yuhao Dong, Zhe Yang, Bo Li, Jingkan Yang, Chen Change Loy, and Ziwei Liu. 2026. [FileGram: Grounding agent personalization in file-system behavioral traces](#). *arXiv preprint arXiv:2604.04901*.
- 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 (Volume 1: Long Papers)*, pages 13851–13870, Bangkok, Thailand. Association for Computational Linguistics.
- 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*.
- Joon Sung Park, Joseph O’Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. [Generative agents: Interactive simulacra of human behavior](#). In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22. Association for Computing Machinery.
- Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. [Zep: A temporal knowledge graph architecture for agent memory](#). *arXiv preprint arXiv:2501.13956*.

- Daniel L. Schacter, Donna R. Addis, Demis Hassabis, Victoria C. Martin, R. Nathan Spreng, and Karl K. Szpunar. 2012. [The future of memory: Remembering, imagining, and the brain](#). *Neuron*, 76(4):677–694.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. [Re-flexion: Language agents with verbal reinforcement learning](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652.
- Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. 2024. [Cognitive architectures for language agents](#). *Transactions on Machine Learning Research*.
- Endel Tulving. 1972. Episodic and semantic memory. In Endel Tulving and Wayne Donaldson, editors, *Organization of Memory*, pages 381–403. Academic Press.
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. [Voyager: An open-ended embodied agent with large language models](#). *Transactions on Machine Learning Research*.
- Henry M. Wellman. 2014. [Making Minds: How Theory of Mind Develops](#). Oxford University Press.
- Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. [LongMemEval: Benchmarking chat assistants on long-term interactive memory](#). In *International Conference on Learning Representations*, volume 2025, pages 86809–86836.
- Ying Xie. 2026. [Learning to forget: Sleep-inspired memory consolidation for resolving proactive interference in large language models](#). *arXiv preprint arXiv:2603.14517*.
- Jing Xu, Arthur Szlam, and Jason Weston. 2022. [Beyond goldfish memory: Long-term open-domain conversation](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5180–5197, Dublin, Ireland. Association for Computational Linguistics.
- 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*, volume 38, pages 20004–20031.
- Xing Zhang, Guanghui Wang, Yanwei Cui, Wei Qiu, Ziyuan Li, Bing Zhu, and Peiyang He. 2026. [Experience compression spectrum: Unifying memory, skills, and rules in LLM agents](#). *arXiv preprint arXiv:2604.15877*.
- Xinliang Frederick Zhang and Lu Wang. 2026. [TSUB-ASA: Improving long-horizon personalization via evolving memory and self-learning with context distillation](#). *arXiv preprint arXiv:2604.07894*.
- Xinping Zhao, Xinshuo Hu, Jiabin Xu, Danyu Tang, Xin Zhang, Mengjia Zhou, Yan Zhong, Yao Zhou, Zifei Shan, Meishan Zhang, Baotian Hu, and Min Zhang. 2026. [LMEB: Long-horizon memory embedding benchmark](#). *arXiv preprint arXiv:2603.12572*.
- Wanjuan Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. [MemoryBank: Enhancing large language models with long-term memory](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731.

A Algorithms

Algorithms 1 and 2 specify the two processes described in Sections 2.2 and 2.3. Algorithm 1 omits the Summary type because it was inactive in the reported experiments.

Algorithm 1 SYSTEM 1 synchronous write pipeline

Require: add_memory request for turn u , history window w , LLMs M_e, M_r , vector store $\mathcal{V}$ holding raw inputs, facts and identity items

- 1: INSERTRAW($\mathcal{V}, u$) ▷ (i) raw persist, zero-loss
- 2: $(I, F) \leftarrow M_e(u, w)$ ▷ (ii) identity items and facts only
- 3: $A \leftarrow I \cup F$
- 4: **for** each item $a \in A$ **do**
- 5: $\mathcal{N}_a \leftarrow \text{TOPK}(\mathcal{V}, a)$ ▷ hybrid nearest neighbours + topic key
- 6: **end for**
- 7: $\mathcal{O} \leftarrow M_r(A, \{\mathcal{N}_a\}_{a \in A})$ ▷ (iii) one batched reconcile call
- 8: **for all** $o \in \mathcal{O}$ **do** ▷ duplicates produce no operation
- 9: **if** $o.OP = \text{ADD}$ **then**
- 10: $n_{\text{new}} \leftarrow \text{INSERT}(\mathcal{V}, o.CONTENT)$
- 11: **else if** $o.OP = \text{UPDATE}$ **then**
- 12: $\text{UPDATE}(\mathcal{V}, o.MEMORY_ID, o.CONTENT)$ ▷ merge into target; no revision node
- 13: **else if** $o.OP = \text{SUPERSEDE}$ **then**
- 14: $n_{\text{new}} \leftarrow \text{INSERT}(\mathcal{V}, o.CONTENT)$
- 15: $n_{\text{old}} \leftarrow \mathcal{V}[o.MEMORY_ID]$
- 16: $n_{\text{new}}.SUPERSEDES \leftarrow n_{\text{old}};$
 $n_{\text{old}}.SUPERSEDEDBY \leftarrow n_{\text{new}}$
- 17: **end if**
- 18: **end for**

Algorithm 2 SYSTEM 2 asynchronous three-phase engine

Require: vector store $\mathcal{V}$, graph $\mathcal{G}$ (schema and intention nodes), induction agent M_a with four write-operation classes exposed by three tools (create_graph_node, add_evidence, add_edge), behaviour-abstractor LLM M_b , core-schema LLM M_c , encoders $E_{\text{sem}}, E_{\text{beh}}$

Phase 1: Preprocess (no LLM)

- 1: $\mathcal{F}_{\text{fresh}} \leftarrow \text{UNREFERENCEDFACTS}(\mathcal{V})$ $\triangleright$ fact triage
- 2: $(\mathcal{C}, \mathcal{U}) \leftarrow \text{TWOSTAGEDBSCAN}(\mathcal{F}_{\text{fresh}})$ $\triangleright$ clusters and unclustered facts
- 3: $\mathcal{X} \leftarrow \mathcal{C} \cup \{f : f \in \mathcal{U}\}$ $\triangleright$ singleton inputs preserve unclustered facts
- 4: **for all** $x \in \mathcal{X}$ **do**
- 5: $\mathcal{R}_x \leftarrow \text{FORWARDNEAREST}(\mathcal{G}, \text{CENTROID}(x)) \cup \text{BACKWARDBYEVIDENCE}(\mathcal{G}, x)$
- 6: **end for**

Phase 2: LLM Induction (single call per input)

- 7: **for all** $x \in \mathcal{X}$ **do**
- 8: $O \leftarrow M_a(x, \mathcal{R}_x)$ $\triangleright$ emit calls to the three function tools
- 9: **for** $o \in O$ **do**
- 10: $\text{APPLY}(\mathcal{G}, o)$ $\triangleright$ create by layer; evidence = DERIVEDFROM; edges = RELATEDTO
- 11: **end for**
- 12: **end for**

Phase 3: Cross-domain sweeper

- 13: **for all** schema node $s \in \mathcal{G}$ **do**
- 14: $b_s \leftarrow M_b(s); E_{\text{beh}}(s) \leftarrow \text{ENCODE}(b_s)$
- 15: **end for**
- 16: $\mathcal{P} \leftarrow \{(s_i, s_j) \mid \text{Eq. (1) holds, different domains}\}$
- 17: $\mathcal{K} \leftarrow \text{UNIONFIND}(\mathcal{P})$ $\triangleright$ aggregate collision pairs into clusters
- 18: **for all** $K \in \mathcal{K}$ **do**
- 19: $s_{\text{core}} \leftarrow M_c(K)$
- 20: **if** $s_{\text{core}} \neq \text{null}$ **then**
- 21: $\text{INSERTCORESCHEMA}(\mathcal{G}, s_{\text{core}}, K)$
- 22: **end if**
- 23: **end for**

B Hyperparameters

Table 5 consolidates the reported retrieval, clustering and collision hyperparameters. The two collision thresholds are tuned once on a held-out PersonaMem-v2 split and then fixed across benchmarks and backbones.

C Schema Faithfulness Audit

C.1 Scope

We manually audit the outputs of SYSTEM 2 to answer two separate questions: whether an induced schema *faithfully summarises* the user behaviour it claims to describe, and whether the evidence links written by ADD_EVIDENCE *correctly support* that schema. The two are scored independently and are never merged into a single “schema accuracy” figure, because the results below show that they behave differently.

The audit covers PersonaMem-v2 only, and covers only the Phase 2 and Phase 3 products of SYS-

Component	Parameter	Value
Vector store	backend	Qdrant
	HNSW m	32
	HNSW ef	128
	index metric	cosine
Embedding	model	bge-m3
	dimension	1024
	dense/sparse mix α	0.7
Retrieval	reconcile top- k	5
	read-time node limit	20
SYSTEM 1	M_c/M_r temperature	0
	(kimi-2.5 default)	0.6
SYSTEM 2	digest schedule	1 per 5 writes
	DBSCAN ε_1	0.35
	DBSCAN ε_2	0.25
	DBSCAN min_samples	3
	$E_{\text{beh}}/E_{\text{sem}}$ backbone	bge-m3
Collision	θ_{beh}	0.72
	θ_{sem}	0.40

Table 5: Reported retrieval, clustering and collision hyperparameters. E_{beh} and E_{sem} share an encoder backbone but receive different inputs (behavioural description versus original schema text).

TEM 2. It does not assess the quality of SYSTEM 1 fact or identity extraction.

C.2 Sample

We audit 150 schema nodes drawn from the graphs built for the runs reported in Table 2, under both backbones (kimi-2.5 and deepseek-v4-flash): 100 basic schemas together with their evidence links, and 50 advanced/core schemas together with their member schemas and links. Since a schema is induced from a fact cluster rather than from an individual fact, each node carries several evidence links, so the link-level sample is larger than the node-level sample; accuracies reported for ADD_EVIDENCE in §C.5 are computed over links rather than over nodes.

C.3 Criteria

The criteria applied in the audit are two at the schema level and one at the link level.

C1: Scope. A schema describes a regularity that a user exhibits within a bounded class of situations or activities. The schema must state that class, and the class must not be broader than its evidence warrants. A schema that generalises a regularity observed in one setting into an unrestricted claim about the user is scored as over-scoped, even when

Audit target	Accuracy
Schema and core-schema content (C1–C2)	$\geq 97\%$
ADD_EVIDENCE links (E1)	$\sim 82\%$

Table 6: Schema faithfulness audit on PersonaMem-v2. Schema content is scored per node; evidence links are scored per link.

every individual statement in it is traceable to some fact.

C2: Evidential support. Every claim the schema makes must appear in its evidence, and the regularity must be instantiated by at least two evidence items. A single instance does not establish a regularity, with one exception: one item suffices when the user’s own utterance in that item carries explicit habitual language (for example *often*, *always*, *usually*), since the recurrence is then asserted by the user rather than inferred by the model.

For core schemas, the member schemas play the role of evidence: the higher-level regularity must be instantiated by at least two member schemas and must hold for each of them.

E1: Link support, judged in reverse. Evidence links are scored in the opposite direction from schema induction. The annotator reads the schema first, then reads the linked item, and asks whether that item supports the schema as stated. An item that belongs to the same cluster as the schema, or that shares a topic with it, but that does not itself support the asserted regularity, is scored incorrect. Links whose evidence_vdb_ref does not resolve are also scored incorrect.

C.4 Procedure

The audit was carried out by a single annotator, one of the authors. We did not perform a second independent annotation and therefore report no inter-annotator agreement; §C.6 discusses the implications.

C.5 Results

Table 6 reports the outcome. Schema content is faithful in at least 97% of the audited nodes, whereas roughly 82% of the ADD_EVIDENCE links are correct. This gap is the central finding of the audit: the induction agent summarises reliably but attributes less reliably. The failures we observed under E1 are typically links to items that are topically related to the schema without supporting the

Symbol	Stage	Prompt
M_e	SYSTEM 1 extraction	§D.1
M_r	SYSTEM 1 reconcile	§D.2
M_a	SYSTEM 2 Phase 2	§D.3
M_b	SYSTEM 2 Phase 3	§D.4
M_c	SYSTEM 2 Phase 3	§D.5

Table 7: Index of prompts by model symbol and pipeline stage.

regularity it states, which is consistent with attribution operating at the granularity of the cluster supplied to the agent rather than at the granularity of the regularity the agent induced. Evidence linking is accordingly the component of SYSTEM 2 we regard as most in need of improvement.

C.6 Limitations of this audit

Because the audit uses one annotator and covers 150 nodes from PersonaMem-v2, its rates indicate magnitude rather than precise population estimates. We cannot quantify annotation subjectivity through inter-annotator agreement, and the error profile may differ on corpora with a different domain structure.

D Memory-Construction Prompts and Evaluation Templates

This appendix transcribes the memory-construction prompts used by DPCM in the reported system, with two terminology inconsistencies normalised to the contracts enforced at runtime. Table 7 maps each prompt to the model symbol used in Section 2. Evaluation templates follow the official benchmark implementations and are cited in Section D.6, rather than reproduced in full. To preserve pdfLaTeX compatibility, the single Chinese rendering of the subject phrase in the source extraction prompt is replaced by the meaning-equivalent text [Chinese equivalent of “The user”]. Placeholders appear in braces, e.g. {content}; literal braces inside JSON templates are doubled, following the Python string-formatting convention used in our implementation.

D.1 Extraction (M_e)

The extraction model is called once per add_memory request. It receives the system prompt of Listing 1 and the user message of Listing 2, and emits identity items (L4_IDENTITY) and atomic facts (L2_FACT) in a single call.

Listing 1: Extraction system prompt (M_e).

# ROLE You are a precise Memory Extractor - an analytical system responsible for extracting rich, structured, and context-bound memories from conversations. Your core objective is to identify personal information, preferences, and events, converting them into self-contained factual statements. You must extract valuable context from BOTH user and assistant messages: - **User messages**: Reveal personal facts, preferences, future plans, past experiences, beliefs, and implicit attitudes. - **Assistant messages**: Contain specific recommendations provided to the user, agreed-upon plans, or confirmed conclusions. (Frame these from the user's perspective, e.g., "The user accepted a recommendation to..."). Do NOT extract generic conversational filler, assistant meta-commentary, or vague unconfirmed characterizations. # INPUT CONTEXT You will process the conversation based on the following input data components (provided in the prompt context): - **Last k Messages**: The recent conversation history leading up to the New Messages. Use this strictly for coreference resolution (e.g., resolving pronouns like "it", "he", or "that project") and contextual grounding. - **New Messages**: The current conversation turn(s) requiring memory extraction. - **Memory Date**: The actual current time of the conversation. When the conversation actually took place (e.g., "2023-05-24"). This is your ONLY temporal anchor for resolving time references. - **Current Date**: Today's system date. May be years after Memory Date. Do NOT use this to resolve temporal references in messages - only Memory Date grounds user and assistant statements. # CORE PRINCIPLES 1. **User as Subject**: Every memory MUST describe the user in the third person. Use "The user ..." for English input, and "[Chinese equivalent of "The user"]...." for Chinese input. 2. **Contextual Completeness**: Every extracted memory must be a complete, self-contained statement. Never omit circumstances, conditions, coreferences, or reasons that give the memory meaning. Retain ALL specific details (names, brands, numbers, equations like $E=mc^2$, etc.). A reader must fully understand the memory WITHOUT seeing the original conversation. 3. **Language Consistency**: The output language MUST match the input language strictly. 4. **No Information Loss**: Each memory must capture the FULL picture of what was said. Do not drop important context, qualifying statements, or nuances. When in doubt, include more rather than less. ## Language Rules 1. Preserve the exact script and alphabet of the input - never transliterate or convert between writing systems. 2. Technical terms, proper nouns, and brand names must remain in their original form exactly as they appear in the conversation. 3. All quality standards (contextual richness, temporal grounding, self-contained completeness) apply equally regardless of language. 4. For mixed-language input (e.g., code-switching, Hinglish, Spanglish), preserve both the language mixing pattern and the original scripts used. Time Handling Rules - **Time-Sensitive Content** (Plans, specific past events, deadlines): Resolve relative time (e.g., "next week") against the Memory Date and embed the absolute time naturally into the memory content. - **Atemporal Content** (Stable preferences, personality traits): Do NOT include time references. - If the Memory Date is missing but a relative expression exists, rewrite the content without the raw relative expression (make it atemporal). # MEMORY LAYERS	You will categorize extracted information into two distinct layers. **Overlap is acceptable**; a single conversation snippet may produce multiple facts and identity memories. ### 1. Identity (L4_IDENTITY) Captures the user's **personal attributes, preferences, attitudes, opinions, values, and feelings**. It defines what makes the user unique. - Includes tastes, beliefs, emotional dispositions, and behavioral patterns. - Each identity memory must include full context. If the user formed an opinion through a specific experience, include that context. - *Exception*: Stable structured profile attributes (name, age, location, occupation, employer) are handled via the <code>`update_basic_user_profile`</code> tool and should NOT be emitted here. **Implicit preference detection - read between the lines**: Users rarely say "I like X" or "I hate Y" directly. Watch for attitude-laden cues and create explicit identity memories when you spot them: - Positive: resuming/returning to an activity, passion/excitement words, voluntary effort, repeated engagement, enthusiastic detail - Negative: burden/chore words, loss of interest, stepping back, disappointment, avoidance - Shift (critical): "but now ...", "recently ...", "used to ... but ...", "decided to resume/quit", re-engagement after disengagement When such a signal appears - even implicitly - you MUST emit an identity memory that explicitly states the attitude . ### 2. Facts (L2_FACT) Captures **objective events, experiences, and factual occurrences**. Focuses on *what* the user did, is doing, or will do, completely decoupled from how they feel about it. - Includes past events, current ongoing activities, future plans, and objective factual statements. ### Consolidation - prefer whole thoughts over fragments Within each layer (identity, facts), avoid over-splitting. Emit ONE self-contained memory that captures the whole picture rather than several fragmentary ones. # TAGS Each memory must include a <code>`tags`</code> field - a list of 1 to 3 short topic keywords that categorize what the memory is about. Tags help with organizing and retrieving memories by topic. Use lowercase, specific topic words (e.g., "music", "travel", "work", "food").
--	---

Listing 2: Extraction user message (M_e).

```

## Last k Messages:
{last_messages}

Current date: {current_date}

## New Messages
{content}

## Memory date
{memory_at}

## Output contract

Strict formatting rules:
1. Output MUST be a valid JSON object wrapped in a fenced
   code block that starts with ```json and ends with ```
2. NO prose before or after the fenced block.
3. NO explanations, comments, apologies, or chain-of-thought.

4. Field types MUST match exactly (strings / arrays / null).
   Unknown or not-applicable fields use `null`.

```

5. If there is nothing to extract for `identity` or `facts`, use an empty array `[]`.
6. Each memory content MUST be complete and self-contained - do NOT omit important information.

JSON shape:

```
```json
{
 "identity": [
 {
 "content": "self-contained memory about user's
 preference / attitude / opinion with full context",
 "tags": ["topic1", "topic2"]
 }
],
 "facts": [
 {
 "content": "self-contained memory about an event /
 experience / plan with full context",
 "tags": ["topic1"]
 }
]
}
```
```

Output the JSON now.

D.2 Reconciliation (M_r)

The reconcile model receives the retrieved existing memories together with the batch of new ones and emits ADD, UPDATE or SUPERSEDE operations; superseded nodes are retained and linked into the evolution chain by the system rather than by the model. The model emits ADD, UPDATE or SUPERSEDE. UPDATE requests a compatible in-place merge intended to preserve both sources and creates no revision node; SUPERSEDE creates a new chain head while retaining its predecessor.

Listing 3: Reconciliation prompt (M_r).

```
You are a memory management system. Your task is to
integrate a batch of new memories into the existing
memory base while keeping it clean, retrievable, and
losslessly informative.

Current date: {current_date}

## Existing memories
{existing_memories}

Each existing memory has these fields:
- `memory_id`: Stable identifier (reference this for
  SUPERSEDE/UPDATE).
- `content`: the stored text.
- `memory_at`: timestamp when the memory occurred (ISO
  format). May be `null` (time unknown).
- `layer`: `l2_fact` or `l4_identity`.
- `tags`: topic keywords.
- `history_versions`: A supersedes chain of previous
  versions.

### New Memories
{new_memories}
- These are raw extractions containing `content`, `layer`,
  and `tags`. They lack IDs and history.

## Understanding supersedes chains

A chain represents state evolution on one specific
dimension - the user's stance on something changed
over time.

Given a chain A -> B -> C (where A is the latest head, C is
the oldest):
- All three nodes talk about the same dimension (e.g., "
```

- preferred drink"), but each states a different claim.
- A is the **current truth** on that dimension.
- B and C are **historical** - their claims on that dimension are outdated.
- However, B and C may contain OTHER information (on different dimensions) that is still valid and useful.

Example:

- C: "The user lives in Beijing and works as a software engineer."
- B: "The user moved to Shanghai for a new job." (supersedes C on residence - cannot live in both cities simultaneously)
- A: "The user relocated to Tokyo." (supersedes B on residence)

Here, only the residence dimension evolves (one can only live in one city at a time). The fact "works as a software engineer" in C is still valid - it was never contradicted.

Counter-example (NOT a valid chain):

- "The user likes coffee." -> "The user also likes tea."
- These are NOT contradictory - preferences can coexist. Do NOT supersede. Just ADD independently.

Key principle: A chain node ONLY states the new/changed claim on the contested dimension. It does NOT repeat information from older nodes. Old nodes remain visible and retrievable - there is no need to copy their content forward.

When you see `history\_versions` on an existing memory, read the full chain to understand the complete history before making decisions.

Operation Primitives

1. ADD (Pure New Memory)

- **When to use**: The information is entirely new and does not overlap with or contradict any existing memories.
- **Fields**: `op: "ADD"`, `content`, `layer`, `tags`.

2. SUPERSEDE (State Evolution / Chaining)

- **When to use**: The new memory represents a change in state or a contradiction to an existing memory (e.g., a change in location, job, or a complete shift in a specific preference).
- **Strict Content Rule**: The `content` MUST ONLY describe the new/changed information. **Do NOT** copy or repeat valid information from the superseded node. The system preserves the old node in the history chain automatically.
- **Fields**: `op: "SUPERSEDE"`, `memory\_id` (of the old head node), `content` (new state only), `supersede\_reason`, `tags`.

3. UPDATE (Consolidation & Refinement)

- **When to use**: The new memory describes the same topic as an existing one without contradiction. Use this to merge fragments or add detail.
- **Merge Logic**: Combine the new information and the existing memory into a single, high-density node. The new `content` must be a lossless synthesis of both sources.
- **Fields**: `op: "UPDATE"`, `memory\_id` (of the existing node), `content` (the merged/refined text), `tags`.

Objectives

Objective 1 - Build contradiction chains for ACTUALLY CONTRADICTIONARY state

When a new memory and existing memories make claims on the **same dimension** that **cannot both be currently true**, form a chain: emit a SUPERSEDE operation whose `memory\_id` identifies the existing head node.

Key test: "If a reader asks 'what does the user currently do/prefer/have on dimension X?', would the old node give the **WRONG** answer?" If yes -> supersede it.

STRICT rules for supersedes:

- ONLY use `supersedes` when there is a genuine factual contradiction (A says X, B says not-X).
- Do NOT supersede for refinement, elaboration, or accumulation - those go to Objective 2.

```

- Do NOT supersede just because the new memory adds more
  detail on the same topic.
- The new head node MUST contain ONLY the new/changed claim.
  Do NOT copy content from the superseded node - it is
  still visible in the chain.

### Objective 2 - Consolidate fragmented memories

When memories describe the same topic compatibly (no
contradiction), merge them:
- Use the `UPDATE` primitive to absorb the new information
  into the existing node.
- The updated content MUST preserve every meaningful fact
  from all sources.

### Objective 3 - Granularity control

Each memory should represent one coherent thought with
complete internal logic. Avoid extremes:
- Too fine-grained: fragmentary sentences that lack context
  on their own
- Too coarse-grained: a single topic that has grown too
  large (e.g. exceeding ~2000 characters), packing too
  many aspects into one node. In this case, split the
  oversized node into multiple finer-grained aspect nodes
  - each covering a distinct sub-aspect - while ensuring
  NO information is lost across the split.

When in doubt, prefer slightly coarser (keep related aspects
together) over splitting too aggressively.

### Objective 4 - Do not touch what does not need changing

Memories you choose not to operate on remain untouched. Do
not emit ops against them.

## Absolute prerequisite - lossless information preservation

**No information from any source - new or existing - may be
lost after your ops execute.** Every meaningful fact
must remain retrievable: as content of a chain node, as
part of a consolidated ADD, as an independent ADD, or
untouched in its original node.

**However, for supersedes chains**: the old node is NOT
deleted - it remains in the chain and is visible to
readers. Therefore you do NOT need to copy old node
content into the new head. Only write the new/changed
claim. Information is preserved because old nodes are
still there.

## Hard constraints

- Every ADD's `content` is self-contained and third-person.
  Language MUST match input memories.
- `memory_at: null` means time unknown. Never fabricate
  chronology; never cite "age" as a reason for
  superseding.
- Prefer same-layer ops (L4<->L4, L2<->L2). Cross-layer only
  when semantic meaning genuinely transfers.
- Output language matches input memory language.

## Acceptance self-check (run mentally before emitting)

1. **Lossless**: every fact from every new memory, and every
  fact from every existing node you are superseding or
  updating, ends up somewhere in the final state.
2. **Fragmentation cleared**: no two non-chain nodes on
  clearly the same topic remain as separate fragments
  where a merge would improve retrievability.

## Output format

Top-level is a flat JSON array of operation objects:

```json
[
 {{"op": "ADD", "content": "The user ...", "layer": "
 L4_IDENTITY", "tags": ["..."]}},
 {{"op": "SUPERSEDE", "memory_id": "<existing_id>", "
 content": "The user relocated to Tokyo.", "
 supersede_reason": "Previously lived in Shanghai, now
 moved to Tokyo - one can only live in one city", "layer":
 "L2_FACT", "tags": ["location"]}},
 {{"op": "UPDATE", "memory_id": "<existing_id>", "content":
 "<merged_content>", "reason": "New memory adds detail
 about the same topic, merging for completeness", "layer

```

```

 "": "L4_IDENTITY", "tags": ["..."]}]
]
...

Output contract (strict)

1. Output MUST be a single JSON array wrapped in a fenced
 code block that starts with ```json and ends with ```
2. NO prose, markdown, apologies, or chain-of-thought
 outside the fenced block.
3. If nothing needs to change, output ``[]` inside the fenced
 block.
4. Field types must match exactly: `tags` is an array (use
 `[]` when empty).

Now produce the JSON array.

```

### D.3 Induction Agent ( $M_a$ )

The induction agent is invoked once per clustered or singleton input produced by the Phase 1 pre-process. It writes schema nodes (L6) and intention nodes (L7) into  $\mathcal{G}$  exclusively through its tools.

Listing 4: Induction agent system prompt ( $M_a$ ).

```

You are a cognitive processing Agent. Your job is to evolve
higher-order cognitive structures from a user's raw
memory data.

Memory Architecture

VDB (Vector Database) - read-only
| Layer | Description |
|-----|-----|
| L0 BASIC_INFO | User attributes (name, age, occupation) |
| L1 RAW | Original conversation transcripts |
| L2 FACT | Atomic facts extracted from conversations |
| L3 SUMMARY | Reserved session-summary type (inactive in
 the reported experiments) |
| L4 IDENTITY | Identity traits, preferences, habits |
| L5 KNOWLEDGE | Optional domain knowledge (not used in the
 reported experiments) |

Graph (Knowledge Graph) - you are the sole writer
| Layer | Description |
|-----|-----|
| L6 SCHEMA | Behavioral patterns - domain-bound, atomic,
 immutable once created |
| L7 INTENTION | Prospective events - concrete plans with
 clear action, time may be fuzzy |

What is an L6 Schema?

A Schema captures ONE behavioral pattern in a SPECIFIC
domain. Three components:

- **Circumstance**: The domain/topic/situation where this
 pattern is observed. A Schema MUST stay within its
 circumstance - do NOT generalize across domains. Cross-
 domain abstraction is handled separately by the system.
- **Pattern**: The user's habitual behavior, thinking style,
 or action tendency in this circumstance.
- **Insight**: The underlying psychological driver or mental
 model.

Content format

Single sentence combining all three:
"When [circumstance], the user [pattern] - reflecting [
 insight]."

Rules

- **Atomic**: One pattern per Schema. Two distinct patterns
 -> two Schemas.
- **Immutable**: Once created, content NEVER changes.
- **Evidence only**: New facts support existing Schema ->
 call `add_evidence`. Do NOT recreate.
- **Domain-bound**: Stay within the observed domain. Cross-
 domain abstraction is handled by a separate system
 process.

Good vs Bad examples

```

```

GOOD "When cooking, the user strictly follows recipe steps
and precisely measures ingredients - reflecting a need
for external structure to manage uncertainty."
GOOD "When gaming, the user always checks walkthroughs to
achieve 100% completion - driven by low tolerance for
incompleteness."
BAD "The user approaches music as a medium for creative
synthesis and cultural bridge-building, consistently
blending diverse musical traditions..." (multiple
patterns merged, too broad)
BAD "The user is passionate about many things and values
quality." (no circumstance, too vague)

What is an L7 Intention?

An Intention is a CONCRETE future event or plan. It requires:

- **A clear action**: Something the user will DO (not just
feel or believe)
- **Temporal boundedness**: Explicit ("next week") or
implicit ("currently preparing for") - the event will
eventually happen or expire

Rules
- No concrete action or event -> NOT an Intention. Consider
Schema instead.
- Life aspirations, values, visions -> Schema, not Intention.
- One event per Intention. Keep concise.

Good vs Bad examples
GOOD "The user is preparing for a job interview."
GOOD "The user plans to attend a Pacific music festival next
month."
GOOD "The user is developing an app for blending cultural
music traditions."
BAD "To use music as a transformative force that transcends
boundaries..." (life vision -> Schema)
BAD "The user wants to be a better person." (no concrete
action)

Your Workflow

Input materials come in two parts:
- **Cluster Results**: Facts pre-grouped by semantic
similarity to help you quickly identify themes
- **Unclustered Facts**: Facts the clustering algorithm
could not group, but they may hold higher-level
abstraction value (cross-topic behavioral patterns,
implicit intentions, etc.) - examine them carefully one
by one

For each group of facts (clustered or unclustered):
1. **Search existing Schemas**: If an existing Schema
already captures this pattern, call `add_evidence` - do
NOT recreate.
2. **Create new Schema**: Only if no existing Schema covers
this pattern. One sentence, three components.
3. **Detect Intentions**: If facts contain a concrete future
plan/event, create an L7 Intention.
4. **Build relationships**: If two Schemas are thematically
related, use `add_edge` (RELATED_TO).

Principles

- All Graph operations MUST go through tools
- Schema content is IMMUTABLE - never recreate what already
exists
- Prefer `add_evidence` over creating duplicates
- Prefer not creating over creating low-quality nodes
- One Schema = one atomic pattern in one domain
- One Intention = one concrete future event
- When tagging, prefer reusing tags from the existing tags
list

```

Through three function-calling tools, the agent can create schema or intention nodes, attach evidence to existing nodes, and add RELATEDTO edges. The two node-creation operations share create\_graph\_node and are selected by its layer argument; Listing 5 gives the complete schemas passed in the tools parameter.

Listing 5: Induction agent tool schemas ( $M_a$ ).

```

[
{
 "type": "function",
 "function": {
 "name": "create_graph_node",
 "description": "Create an L6_SCHEMA or L7_INTENTION
node. Content must be a single sentence for Schema (
circumstance + pattern + insight). Automatically
creates VdbRef + DERIVED_FROM + Tag edges.",
 "parameters": {
 "type": "object",
 "properties": {
 "layer": {
 "type": "string",
 "enum": ["l6_schema", "l7_intention"],
 "description": "Node layer"
 },
 "content": {
 "type": "string",
 "description": "Node content (single sentence
for Schema)"
 },
 "evidence_list": {
 "type": "array",
 "items": {"type": "string"},
 "description": "VDB node_id list as evidence"
 },
 "tags": {
 "type": "array",
 "items": {"type": "string"},
 "description": "Tag list (prefer reusing
existing tags)"
 },
 "confidence": {
 "type": "number",
 "description": "Confidence 0-1, default 0.8"
 }
 },
 "required": ["layer", "content", "evidence_list"]
 }
 },
},
{
 "type": "function",
 "function": {
 "name": "add_evidence",
 "description": "Add evidence to an EXISTING Schema or
Intention. Links new VDB facts to the node via
DERIVED_FROM edges. Does NOT change the node content.",
 "parameters": {
 "type": "object",
 "properties": {
 "node_id": {
 "type": "string",
 "description": "Existing Graph node ID to add
evidence to"
 },
 "evidence_list": {
 "type": "array",
 "items": {"type": "string"},
 "description": "VDB node_id list to add as new
evidence"
 }
 },
 "required": ["node_id", "evidence_list"]
 }
 },
},
{
 "type": "function",
 "function": {
 "name": "add_edge",
 "description": "Create a RELATED_TO relationship
between two Graph nodes.",
 "parameters": {
 "type": "object",
 "properties": {
 "source_id": {"type": "string"},
 "target_id": {"type": "string"},
 "reason": {
 "type": "string",
 "description": "Reason for the relationship"
 }
 },
 "required": ["source_id", "target_id"]
 }
 },
}
]

```

```

 }
 }
]

```

#### D.4 Behavioural Abstraction ( $M_b$ )

Each schema is rewritten into a domain-free behavioural description, which  $E_{\text{beh}}$  then encodes to form the behavioural space used by the dual-space collision test.

Listing 6: Behavioural abstraction prompt ( $M_b$ ).

```

Task: You are a behavioral psychologist. Abstract the
following schema into a pure psychological/behavioral
description. Strip ALL domain-specific nouns and
scenarios - output ONLY the underlying behavioral style
and psychological motivation.

CRITICAL LANGUAGE RULE: Your output MUST be in the SAME
language as the input schema below. If the input is in
English, you MUST output in English. If the input is in
Chinese, you MUST output in Chinese. Violating this
rule invalidates the output.

Input schema:
{content}

Output (strict JSON, same language as input):
{{"abstraction_for_embedding": "...pure behavioral
description, same language as input..."}}

```

#### D.5 Core-Schema Induction ( $M_c$ )

For each Phase 3 collision cluster,  $M_c$  proposes a core schema or returns null when the evidence is insufficient. Only non-null results are inserted into  $\mathcal{G}$ .

Listing 7: Core-schema induction prompt ( $M_c$ ).

```

Task: You are a deep pattern analyst. The system has
detected structural resonance among the following
schemas from different areas of the user's life. On the
surface they appear unrelated, but their underlying
behavioral logic is strikingly similar.

Your job: synthesize a HIGHER-ORDER pattern that explains
WHY these behaviors co-occur - something the user
themselves may not be consciously aware of. Think in
terms of:
- Deep cognitive style (how they process information and
make decisions)
- Core psychological need (what drives them at a fundamental
level)
- Hidden mental model (the implicit belief system behind
these behaviors)

Be insightful. Be precise. Quality over quantity - only
output a synthesis if the connection is genuinely
compelling and logically airtight. If the evidence is
weak or the connection is superficial, output null.

LANGUAGE RULE: Your output (core_pattern, reasoning) MUST be
in the SAME language as the input schemas below. If
the schemas are in Chinese, output in Chinese. If in
English, output in English. Do NOT translate or switch
language.

Input schemas:
{patterns}

Output (strict JSON, or null if no compelling synthesis):
{{"core_pattern": "...one sentence describing the higher-
order pattern, same language as input...", "reasoning":
"...why these schemas are connected at a deep level,
same language as input...", "confidence": 0.85}}

```

#### D.6 Evaluation Templates

The evaluation uses the official implementations without modifying their templates; those templates are referenced rather than reproduced here. LongMemEval answers are generated freely and then judged by gemini-3.0-flash with the judge template of Wu et al. (2025); PersonaMem and PersonaMem-v2 use the response-selection templates of Jiang et al. (2025a) and Jiang et al. (2025b). Because the templates are identical across all systems compared in Table 2, the only difference between conditions is the memory content placed in the context slot: DPCM supplies the tokens returned by its read path, whereas the long-context baseline supplies the session history itself.

#### E Reproducibility

All three benchmarks are publicly available: LongMemEval (Wu et al., 2025), PersonaMem (Jiang et al., 2025a) and PersonaMem-v2 (Jiang et al., 2025b). We collected no new user data; LongMemEval-S includes filtered public conversations in addition to synthetic sessions. The reported configuration specifies the evaluation backbones (Section 3.1), retrieval, clustering and collision parameters (Appendix B), the SYSTEM 2 schedule of one digest per five add\_memory writes, the average retrieval payload (Table 1), pseudocode for both processes (Appendix A), and the memory-construction prompts (Appendix D). Experiments use the no-think setting throughout and require a Qdrant instance with the bge-m3 encoder. We do not report the individual backbone assignments for  $M_a$ ,  $M_b$  and  $M_c$ ; the provided details therefore support architectural reimplementations rather than exact run-level replication.