PRAM: PARTIAL-CUE RETRIEVAL WITH ASSOCIATIVE MEMORY BEYOND TEXTUAL RECORDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term language agents need to recover related experiences from partial cues, compose them across relations, and selectively retain information as new experiences arrive. We investigate whether fixed-size linear associative maps over frozen language-model representations can support these functions. Our framework, Partial-cue Retrieval with Associative Memory (PRAM), binds whitened entity representations with relation and time codes and organizes facts into complementary maps for target, set, and temporal access. Successive association reads construct a query-specific tree. A learned bridge converts its target representations into soft tokens, which are interleaved with textual relation structure and supplied to a frozen language model for answering. We further study a dynamic extension in which fast acquisition, decay, and error-driven consolidation regulate the persistence of associations. Through controlled retrieval, neural readout, conversational question answering, and streaming-memory experiments, we characterize the roles of representation geometry, structured access, and selective retention. The framework connects graph-like memory operations with a continuous language interface and provides a basis for studying both how associations are recalled and how they are maintained.

1 INTRODUCTION

Language agents engaged in continuing interactions need to remember experiences, preferences, and changing facts. Many external-memory systems preserve this information as conversation excerpts, summaries, or textual facts, retrieve relevant records, and return them to the language model's context. Systems such as Mem0 further organize interactions through fact extraction and memory management (Chhikara et al., 2025). Text provides an inspectable representation, but long-term memory also raises questions about access and persistence: how can incomplete cues evoke related experiences, how can relations connect dispersed information, and which associations should remain accessible as new experiences accumulate?

Associative recollection offers a useful perspective. A person, place, or event fragment can cue other elements of an experience. Research on episodic memory links this ability to the binding of event elements and their reinstatement through hippocampal pattern completion (Horner et al., 2015). Computationally, this suggests organizing memory around associations: jointly encode entities, relations, and context so that a partial cue becomes an address, and a recalled target can in turn become the cue for a subsequent association.

Knowledge graphs provide an explicit representation of these relationships. Graph-based memory systems support relational access and temporal organization through entities and edges; Zep, for example, organizes agent memory with a temporal knowledge graph (Rasmussen et al., 2025). This motivates our central question: *can fixed-size linear associative maps support graph-like access through distributed representations?* If entity–relation combinations form distinguishable addresses, a neighborhood lookup can be implemented by a vector read, and successive reads can unfold a path or branching structure from an initial cue.

Two connected challenges arise. First, structured associations must remain recoverable. Frozen language-model representations are not automatically well suited to linear superposition, queries may omit fields, and one address may correspond to several targets. Second, recovered representations must provide usable evidence to the language model. A multi-hop answer depends on both target content and the relations linking intermediate targets; isolated vectors do not expose this structure by themselves.

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

Partial-cue Retrieval with Associative Memory (PRAM), a framework that connects associative addressing to a soft-token language interface. Facts are represented by a subject, relation, target, and optional time. Whitening and multiplicative binding construct addresses, while complementary linear maps support target lookup, set recovery, and temporal inversion. Starting from an entity and relation cues, successive reads build a query-specific association tree. A trainable neural bridge converts target vectors to soft tokens, interleaved with textual relation and branch structure. The frozen language model receives the resulting tree as evidence and generates an answer using its intermediate and terminal targets.

Associations must also evolve with experience. Under bounded capacity, indiscriminate accumulation increases interference, while uniform decay can erase recurring information along with transient events. Complementary learning systems motivate separating rapid acquisition from slower integration (McClelland et al., 1995). We therefore also study a dynamic write extension: new associations enter a fast layer, and subsequent exposures support error-driven consolidation into a slower layer. Decay and consolidation jointly determine retention as a function of elapsed time, repetition, and reactivation.

Our evaluation follows the construction, use, and maintenance of associations. Structured retrieval and conversational QA isolate geometry, partial-cue access, and routing; neural experiments evaluate individual targets and complete retrieved trees; independent streaming experiments examine retention under continuing background writes. Together, they characterize the conditions under which a linear carrier supports graph-like access, a frozen language model can use its outputs, and repeated experiences receive selective long-term retention.

Our contributions are:

- **Associative representations for graph-like access.** We characterize key geometry over frozen language-model codes and combine multiplicative binding with complementary maps for partial-cue, set, temporal, and multi-hop access.
- **A soft-token interface for association trees.** A learned bridge exposes target content together with path and branch structure, allowing a frozen language model to use an entire retrieved tree for answering.
- **An analysis of access and dynamic retention.** Controlled retrieval, language, and streaming experiments identify the roles of encoding, addressing, and readout, and show how decay and fast–slow consolidation support selective forgetting and reinforcement.

2 Related Work

External and structured language-model memory. Retrieval-augmented generation supplies language models with selected external evidence (Lewis et al., 2020). Mem0 extracts and manages conversational facts, MemTree organizes memory hierarchically, and Zep maintains temporal graph structure (Chhikara et al., 2025; Rezazadeh et al., 2025; Rasmussen et al., 2025). Continuous-memory approaches include retrieved hidden-state key–value stores, episodic memory control, and updatable latent memory pools (Wu et al., 2022; Das et al., 2024; Wang et al., 2024). Our focus is the connection between partial-cue graph access in associative maps and a language interface that preserves the retrieved relational structure. Learned KGQA also uses graph embeddings for multi-hop answering (Saxena et al., 2020); our shared-routing controls instead isolate the memory carrier under a common query plan.

Associative representations and memory dynamics. Sparse distributed memory, holographic representations, and compositional graph embeddings establish distributed addressing, binding, and superposition as tools for associative storage (Kanerva, 1988; Plate, 1995; Nickel et al., 2016). Fast-weight formulations connect key–value writes to recurrent matrix states (Schlag et al., 2021). We study how the geometry of frozen language-model codes governs these operations under entity reuse and set-valued access. Episodic binding and complementary learning systems motivate partial-cue recall and separate timescales of acquisition and integration (Horner et al., 2015; McClelland et al., 1995). Spacing research further motivates evaluating exposure schedules rather than repetition counts alone (Cepeda et al., 2008). Our streaming experiments examine these behaviors through explicit write-rule comparisons.

Reading continuous representations. BLIP-2 uses a query-based bridge between frozen representations and a language model (Li et al., 2023). Patchscopes examines hidden-state interpretation,

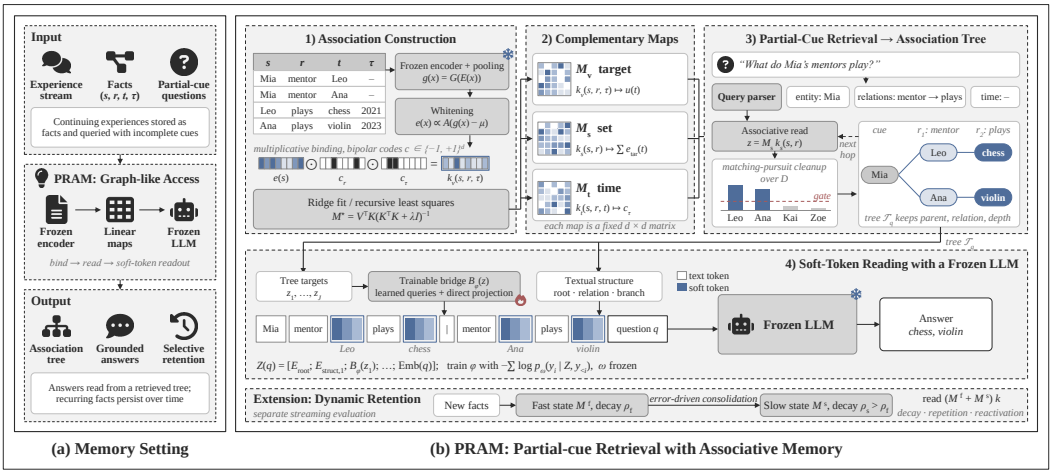

Figure 1: Overview of PRAM. (a) Overall framework for storing experiences and answering partial-cue questions. (b) Detailed architecture for associative memory construction, tree-structured retrieval, and soft-token readout with a frozen LLM, with a dynamic-retention extension shown below.

while embedding inversion reconstructs text from vectors (Ghandeharioun et al., 2024; Morris et al., 2023). We apply a learned bridge to associative readouts and compose member-level soft tokens with relation and branch structure. Individual-target transcription measures the interface's readability; tree-level answering tests whether that interface preserves usable relational evidence. Re-encoding reranking is evaluated as an auxiliary transcription technique, separately from the core tree-input mechanism.

3 Method

Figure 1 summarizes the framework. PRAM has a write stage and a query stage. Structured facts define complementary linear associations. At query time, partial cues recover targets that become cues for subsequent reads, producing an association tree. A neural bridge converts its targets into soft tokens; textual relations and branch structure organize them for a frozen language model. We describe the representations, maps, write mechanisms, tree construction, and neural interface in turn.

3.1 Facts and Representations

A fact is $f = (s, r, t, \tau)$, with subject s, relation r, target t, and optional time τ. Targets may themselves be entities and serve as subjects in later queries. For text x, a frozen encoder E and a content-token aggregation function G produce

$$g(x) = G(E(x)). \tag{1}$$

To improve addressing geometry, we center, whiten, and normalize this representation:

$$e(x) = \frac{A(g(x) - \mu)}{\|A(g(x) - \mu)\|_2}, \tag{2}$$

where μ and A are the center and whitening transform. Subject addressing and target cleanup use their corresponding transforms; the latter representation is denoted $e_{\mathrm{tar}}(t)$. We denote the target representation used for neural reading by $u(t)$, in the coordinate system expected by the bridge. Memory construction and querying use consistent encodings and transforms.

3.2 Bound Addresses and Complementary Maps

Relations and times receive fixed bipolar codes $c_r, c_\tau \in \{-1, +1\}^d$. Multiplicative binding combines the subject representation with these codes to form the full bound address

$$k_v(s, r, \tau) = \mathrm{normalize}(e(s) \odot c_r \odot c_\tau)\,, \tag{3}$$

where $\odot$ is elementwise multiplication and an absent time cue contributes the identity code. Binding modulates a subject by its relation and context to distinguish field combinations.

Different questions supply different subsets of a fact. We construct three complementary maps:

$$\begin{aligned} M_{\mathrm{v}} k_v &\approx u(t), & k_v(s,r,\tau) &= \mathrm{normalize}(e(s) \odot c_r \odot c_\tau)\,, \\ M_{\mathrm{s}} k_s &\approx \textstyle\sum_{t \in \mathcal{N}(s,r)} e_{\mathrm{tar}}(t), & k_s(s,r) &= \mathrm{normalize}(e(s) \odot c_r)\,, \\ M_{\mathrm{t}} k_t &\approx c_\tau, & k_t(s,r,t) &= \mathrm{normalize}(e(s) \odot c_r \odot \mathrm{sign}(e(t)))\,. \end{aligned} \tag{4}$$

Here $\mathcal{N}(s,r) = \{t : \exists \tau, (s,r,t,\tau) \in \mathcal{F}\}$ contains distinct targets. The target map supports full-cue lookup, the set map recovers a neighborhood without a time cue, and the time map attaches temporal information to a known target. A designated output code represents a missing timestamp.

3.3 Writing and dynamic retention

Association construction. For any map, stack its keys and desired outputs as rows of K and V. Regularized least squares fits the associations:

$$M^\star = \arg\min_M \|KM^\top - V\|_F^2 + \lambda \|M\|_F^2 = V^\top K (K^\top K + \lambda I)^{-1}. \tag{5}$$

The same objective admits recursive least-squares updates. For the set map, distinct targets sharing an address are aggregated before writing. The memory fit is separate from language-model training, and a query reads $\hat{v} = Mk$.

Dynamic extension. For continuing fact streams, we additionally study fast and slow associative states. At update step n, their evolution has the form

$$\begin{aligned} M_n^{\mathrm{f}} &= \rho_{\mathrm{f}} M_{n-1}^{\mathrm{f}} + \Delta_n^{\mathrm{f}}, \\ M_n^{\mathrm{s}} &= \rho_{\mathrm{s}} M_{n-1}^{\mathrm{s}} + \Delta_n^{\mathrm{s}}, \qquad 0 < \rho_{\mathrm{f}} < \rho_{\mathrm{s}} \le 1. \end{aligned} \tag{6}$$

New associations first enter the fast state. Subsequent exposure or reactivation invokes consolidation into the slow state, with the update determined by prediction error. The states jointly support the readout

$$\hat{v}_n = (M_n^{\mathrm{f}} + M_n^{\mathrm{s}})k. \tag{7}$$

Decay reduces the persistent contribution of past writes; consolidation selectively supports recurring associations over longer intervals. Reactivation-based reinforcement is implemented through an update. Static construction evaluates structured access, while the dynamic extension evaluates retention under continuing writes using the same associative-addressing principle.

3.4 Constructing an association tree from partial cues

A parser extracts an initial entity, candidate relations or a relation chain, and optional temporal constraints from question q. These cues determine where to query; the memory supplies the targets. For (s,r), the set map returns $z_{s,r} = M_{\mathrm{s}} k_s(s,r)$. To separate its superposed targets, let $D = [d_1, \ldots, d_N]$ contain candidate vectors $d_j = e_{\mathrm{tar}}(t_j)$. Matching pursuit begins with residual $\xi = z_{s,r}$ and an empty selected set S. Each iteration chooses

$$j^\star = \arg\max_{j \notin S} \cos(d_j, \xi), \qquad S \leftarrow S \cup \{j^\star\}, \tag{8}$$

then refits all selected coefficients and updates the residual:

$$a^\star = \arg\min_a \|z_{s,r} - D_S a\|_2^2, \qquad \xi = z_{s,r} - D_S a^\star. \tag{9}$$

Residual similarity, coefficient strength, and a member cap determine when to stop. An initial similarity gate can reject weakly supported queries. The candidate table provides both cleanup vectors and the entity identities needed for subsequent addressing.

For relations $(r_1, \ldots, r_L)$, successive reads expand a frontier:

$$\mathcal{S}_\ell = \bigcup_{s \in \mathcal{S}_{\ell-1}} \widehat{\mathcal{N}}(s, r_\ell), \tag{10}$$

where $\widehat{\mathcal{N}}$ is the recovered neighborhood. We preserve each target's parent, incoming relation, and depth to construct the association tree $\mathcal{T}_q$. Path provenance is retained when the same entity is reached

through different branches. For example, a mentor query first recovers the mentor and then uses that entity to retrieve an activity; multiple mentors produce separate branches. Temporal queries use M_{t} to attach and filter times, with M_{v} available for explicit-time lookup. The parser supplies relation cues, and associative reads execute the expansion.

3.5 Representing the tree with soft tokens

A trainable bridge maps target readout z_j, expressed in the bridge's input coordinates, to a sequence of language-model embeddings:

$$B_\phi(z_j) = [b_{j,1}, \ldots, b_{j,m}], \qquad b_{j,i} \in \mathbb{R}^{d_{\mathrm{LM}}}. \tag{11}$$

Learnable queries interact with target features, together with a direct target projection. Normalization and scale alignment make the resulting soft tokens compatible with ordinary token embeddings.

Target content is carried by soft tokens. Root cues, relation names, and branch boundaries use textual tokens that preserve parentage during serialization. The tree input is

$$Z_{\mathcal{T}_q} = [E_{\mathrm{root}}; E_{\mathrm{struct},1}; B_\phi(z_1); \ldots; E_{\mathrm{struct},J}; B_\phi(z_J)], \tag{12}$$

where semicolons concatenate along the sequence dimension and $E_{\mathrm{struct},j}$ encodes the relevant relation and branch context. Intermediate and terminal targets are both available to the reader. The structure specifies how each target was reached, allowing the language model to use the entire path or branching tree as evidence.

3.6 Training and answer generation

Target-level vector–text supervision trains the bridge to make individual representations readable. Structure-level supervision uses association trees with varied depths and widths, so the interface learns to preserve content in the presence of multiple related targets. For input Z and target text $y = (y_1, \ldots, y_T)$, training minimizes

$$\mathcal{L}_{\mathrm{read}} = -\sum_{i=1}^{T} \log p_\omega(y_i \mid Z, y_{<i}). \tag{13}$$

Language-model parameters ω remain frozen. Training updates bridge parameters ϕ and the aggregation parameters when a learned aggregator is used. Memory matrices follow the separate write rules above. Structure supervision can require transcription of retrieved targets or direct answer generation when a question is appended.

At inference, the system builds $\mathcal{T}_q$ from the parsed cues and supplies

$$Z(q) = [Z_{\mathcal{T}_q}; \mathrm{Emb}(q)], \qquad p_\omega(y \mid q, \mathcal{T}_q) = \prod_{i=1}^{T} p_\omega(y_i \mid Z(q), y_{<i}). \tag{14}$$

Relation and branch tokens provide provenance, soft tokens carry target content, and the question identifies which information is needed. Answer generation can therefore draw jointly on intermediate entities, separate branches, and terminal targets.

4 Experiments

We need to answer research questions underlying PRAM: (i) can fixed-size associative matrices reliably store and retrieve structured associations, including partial-cue and multi-hop access; (ii) can a frozen LLM read the retrieved representations and use the resulting association trees; and (iii) can these associations be selectively retained under continuing updates?

4.1 Evaluation design

Datasets and evaluation tasks. We use complementary benchmarks to evaluate the three research questions above. For controlled evaluation of structured access, we introduce a synthetic conversational benchmark containing 351 questions over 20 instances and 2,692 facts, covering single-value, set-valued, temporal, and multi-hop queries. We further evaluate graph traversal on MetaQA, which contains 134,741 triples and uses 300 questions per hop (Zhang et al., 2018), and conversational

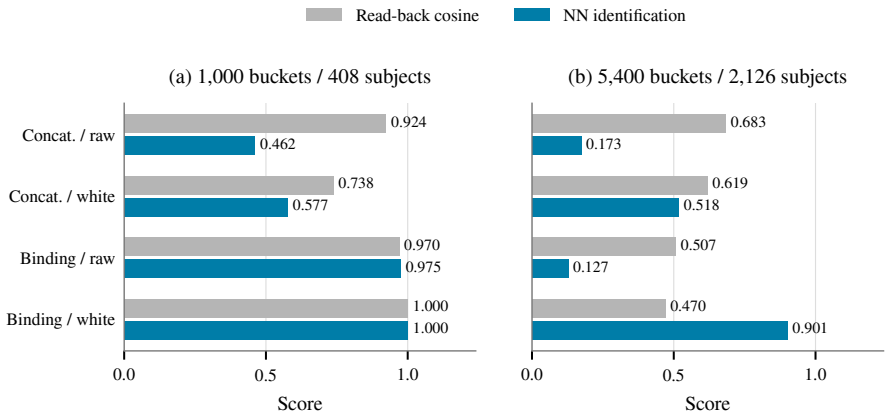

Figure 2: Key geometry under subject reuse on MetaQA. Read-back cosine and nearest-neighbor (NN) identification can diverge: at 5,400 buckets, whitened binding identifies .901 of stored values despite .470 cosine. Duplicate targets count as correct. Exact values and measured ranks are retained in Table 3.

memory on LoCoMo, using 1,802 extracted facts and 1,444 questions (Maharana et al., 2024). To evaluate the soft-token interface, we test target-level string reconstruction on 5,000 held-out Wikidata strings, 2,156 synthetic strings, and 2,000 PopQA strings (Mallen et al., 2023). Finally, to evaluate selective retention, we simulate a 200-day memory stream with continuing background writes and vary exposure frequency, spacing, reactivation, and memory load. Splits and dataset-specific protocols are detailed in Appendix A.

Models, controls, and metrics. Unless otherwise stated, PRAM uses frozen Qwen2.5-1.5B-Instruct layer-20 representations with $d = 1536$, and the static graph-access and QA memories are constructed by ridge regression. Symbolic and growing-KV controls share PRAM's parser and routing, while conversational RAG uses dense retrieval and MetaQA RAG uses TF–IDF. We report candidate-identification accuracy for association recovery, answer-set F1 for graph traversal, transcription exact match for evaluating the soft-token interface, and QA accuracy for conversational tasks. Codebook QA uses Claude Sonnet 4.6 for answer reading and semantic judging. Main codebook results average three code seeds. Additional implementation and scoring details appear in Appendix B.

4.2 Association recovery and graph access

Address geometry. A subject may participate in many different relations, so PRAM must distinguish relation-specific associations that share the same subject representation. A simple concatenation baseline constructs the address as

$$k = [e(s); c_r],$$

where the subject and relation occupy separate blocks. PRAM instead uses multiplicative binding,

$$k = e(s) \odot c_r,$$

so that the relation code modulates the entire subject representation and produces a distinct distributed address for each subject–relation pair. We therefore test whether this binding produces more discriminative addresses under grouped subject reuse on MetaQA.

Figure 2 shows that the advantage becomes pronounced as subject reuse and memory load increase: under the larger setting, whitened binding achieves substantially higher nearest-neighbor identification than whitened concatenation (.901 vs. .518). Moreover, high identification persists despite a read-back cosine of only .470, suggesting that successful associative retrieval depends on preserving separation among candidate targets rather than exact vector reconstruction. Whitening further improves this separation. Rank analysis and random-association load controls are provided in Appendices C and C.1.

Partial-cue and multi-hop access. We next ask whether the improved address discrimination translates into the graph-like operations required by PRAM. Table 1 summarizes both multi-hop

Table 1: Structured access with PRAM. (a) Multi-hop traversal on MetaQA under the same parsed relation path. (b) Partial-cue ablations on controlled conversations.

(a) MetaQA multi-hop traversal

Method	1-hop	2-hop	3-hop
Symbolic	.902	.929	.888
PRAM	.899	.888	.752
Concat.	.597	.466	.247

(b) Partial-cue ablations

Variant	Overall	When	Latest
PRAM	.787	.623	.717
Value map only	.080	–	–
No time map	.610	.000	.094

Symbolic uses the same parsed relation path as PRAM. Full task breakdowns and additional retrieval controls are reported in Appendix F.

Table 2: Target-level transcription EM. Top: direct target representations on held-out Wikidata, Synthetic, and PopQA strings; bottom: transcription from actual $M_{\mathrm{v}}k$ outputs on Synthetic. K denotes the number of generated candidates used for re-encoding reranking. All rows use Qwen2.5-1.5B unless otherwise noted; no test-value consolidation is used.

Configuration	Wiki	Synthetic	PopQA
Direct representations			
Last token (0.2M)	.019	.118	.178
Mean pool + prefix (1M), greedy	.536	.670	.847
Mean pool + prefix (1M), $K = 8$	.744	.830	.942
Compression (3.81M), greedy	.631	.790	.883
Compression (3.81M), $K = 16$	.834	.936	.966
Qwen3-8B, mean pool (1.91M), $K = 8$	.846	.910	.973
After associative retrieval (Synthetic)			
Mean pool (1M), $K = 8$			.737
Compression (3.81M), greedy			.693
Compression (3.81M), $K = 8$			.812
Codebook NN, same memory			.910

traversal and partial-cue ablations. On MetaQA, PRAM remains close to symbolic execution at one and two hops, while the gap widens at three hops. Because the symbolic control follows the same parsed relation path, this degradation reflects errors accumulated during associative retrieval and set cleanup rather than query planning. The substantially weaker concatenation baseline further connects address discrimination to successful traversal.

Controlled-conversation ablations show that discriminative addresses alone are insufficient for partial-cue retrieval. Restricting PRAM to the full-cue value map causes a large drop in overall accuracy, while removing temporal indexing specifically collapses *when* and *latest* queries. These results support the complementary-map design: different partial cues require different access operators, and recovered set members can subsequently serve as the frontier for multi-hop associative reads. Full task breakdowns and additional retrieval controls are provided in Appendix F.

4.3 Soft-token reading of targets and association trees

Target-level reconstruction. We first test whether a frozen LLM can recover target content from the continuous representations used by PRAM. Table 2 shows that representation choice is critical: under the same 0.2M-string training setting, mean pooling substantially improves reconstruction over a last-token representation. With a stronger learned representation and additional training, the interface remains effective after associative retrieval: learned compression reconstructs .812 of held-out synthetic targets from actual $M_{\mathrm{v}}k$ outputs, compared with .910 nearest-neighbor identification from the same memory. Thus, associative retrieval preserves substantial target information, although converting the retrieved representation back into language introduces an additional source of error. Full representation, training-scale, verification, and backbone-transfer ablations are reported in Appendix D.

Reading complete association trees. We next test whether the soft-token interface can use multiple retrieved targets together with their path and branch structure. Figure 3 shows that widening the tree

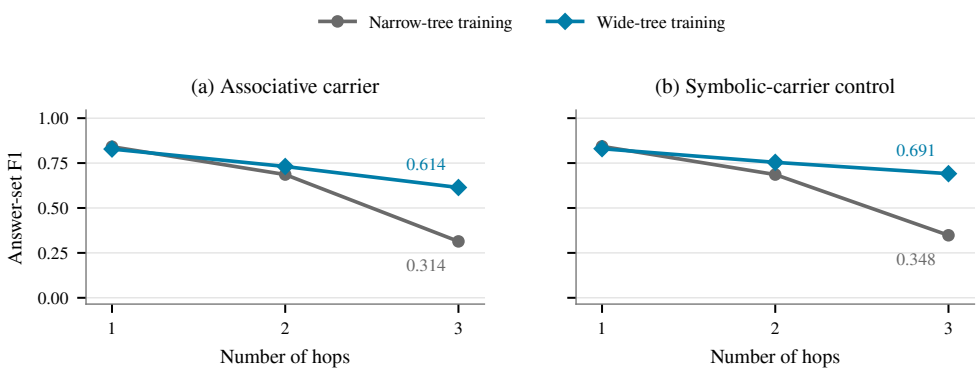

Figure 3: Wider-tree training improves multi-hop use of retrieved association trees. Narrow training uses at most three members, while wide training samples widths up to 12. Both panels report answer-set F1 on 300 MetaQA questions per hop. Full training-exposure and traversal controls are reported in Appendix E.

distribution used for training substantially improves three-hop answering: F1 rises from .314 to .614 with the associative carrier and from .348 to .691 with the symbolic-carrier control, while one-hop performance changes little. Because the intervention changes the bridge's exposure to branching structures rather than the graph-access procedure itself, the improvement indicates better use of the retrieved tree rather than recovery of additional graph edges. We therefore interpret these results as evidence that the soft-token interface can exploit multi-branch relational evidence under the tested training regime. Training-exposure, generation-length, traversal-reference, and replication controls are reported in Appendix E.

4.4 Selective retention under continuing writes

We finally ask whether PRAM can selectively retain recurring associations while allowing transient ones to fade under continuing writes. We simulate a 200-day stream with 400 new background facts per day and compare no decay, single-timescale decay, slow-first routing, and fast-first consolidation. Effective recall requires both correct target identification and sufficient memory strength; results are averaged over three seeds. Figure 4 evaluates forgetting after a single exposure, repeated exposure, spacing, and scheduled reactivation. Full probe schedules, write-rule details, and numerical results are provided in Appendix H.

Selective persistence. Fast-first consolidation exhibits the desired separation between transient and recurring associations: a single exposure is recalled immediately but fades over longer delays, whereas repeated exposure transfers the association into persistent memory. After four exposures, delayed recall reaches 1.00 compared with .38 under single-timescale decay. Spacing further changes what is retained: with the same three exposures, increasing the inter-exposure gap to 7 and 21 days raises consolidated recall to .79 and 1.00, respectively. Scheduled reactivation produces a similar effect, increasing delayed recall to .77 after two refreshes and 1.00 after four. These results show that persistence depends not only on how often an association is encountered, but also on when and how it is reinforced.

Robustness to continuing interference. The same selective-retention pattern persists in the available probes when the background-write rate is increased fivefold. These high-load experiments are intentionally treated as a robustness check rather than a complete capacity comparison; full results are reported in Appendix H. The retention study is evaluated separately from end-to-end QA, and matched total-state budgets across write rules remain an open comparison.

5 Discussion and Limitations

Conditions for graph-like substitution. The experiments support a conditional answer to the title question. First, reusable entity–relation addresses need sufficient discrimination: binding and whitening address different aspects of this requirement. Second, the query must select the right address family; the value-only and temporal ablations show that full-key recall does not supply

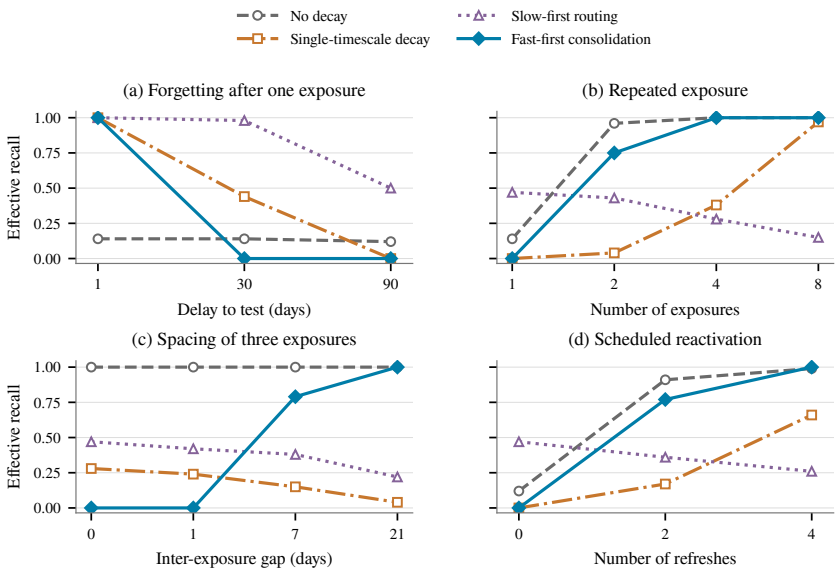

Figure 4: Selective retention under continuing writes. Fast-first consolidation preserves newly written associations at short delays while allowing unrepeated associations to fade. Repetition, wider spacing, and scheduled reactivation progressively support long-term retention. Repetition and spacing are evaluated at a 90-day final-exposure lag. Full values and write-rule controls are reported in Appendix H.

missing-field access. Third, recovered sets need member identities for traversal, so the present system relies on codebook-assisted cleanup. Finally, neural reading must handle the target vocabulary and branching distribution encountered at test time. The tree-width intervention supports this last requirement independently of traversal quality. These conditions explain why controlled shared-routing comparisons can approach symbolic performance while unrestricted conversational access remains harder. They also distinguish replacing a graph's association carrier from replacing its entire storage and query system: parsing, cleanup, provenance, and a readable interface remain necessary components.

Fixed matrices and system cost. "Fixed-size" refers to individual associative maps. Cleanup dictionaries, whitening state, shards, and bridge parameters add storage. Merged synthetic memory scores .735 versus growing KV's .806; sharding trades additional storage for retrieval accuracy. Appendix G reports load sweeps and component accounting. Dual-layer retention also requires accounting for both states when comparing memory budgets.

Evaluation scope. The controlled benchmark shares an ontology with its parser; conversational parsing and judging use a proprietary model; MetaQA uses sampled questions. Adapter exposure controls do not determine base-model pretraining overlap. Set traversal remains codebook-assisted. Matched comparisons with trained neural-memory systems, changing whitening coordinates, reliable deletion, and conflicting temporal events remain open. Dynamic-write settings, exposure schedules, and effective-recall thresholds determine the scope of the retention results.

6 Conclusion

PRAM connects distributed association storage with a language interface that preserves retrieved relations. Bound addresses and complementary maps support graph-like access; soft target tokens and textual structure expose the association tree to a frozen language model. Independent streaming experiments show how fast acquisition, decay, and slow consolidation support selective retention under interference. These findings connect how associations are accessed, used, and maintained, while identifying semantic routing, set recovery, and dynamic end-to-end integration as remaining challenges.

REPRODUCIBILITY STATEMENT

The appendices specify datasets, memory construction, readout variants, overlap regimes, and supplementary static and dynamic experiments. Results distinguish codebook QA, vector transcription, and chain-token answering. Method comparisons share routing where indicated; other differences are stated in table captions.

ETHICS STATEMENT

Reconstructing strings from stored representations can expose information that users may incorrectly assume is hidden by vectorization. These memories should not be interpreted as a privacy-preserving storage mechanism. The experiments use public benchmarks and synthetic facts; deployment with personal conversations requires consent, access control, and reliable deletion beyond the mechanisms studied here.

AI USE STATEMENT

Language models are used within the study for controlled question rewriting, fact extraction, query parsing, answer generation, and semantic judging. Generative AI also assisted manuscript drafting and organization.

REFERENCES

Nicholas J. Cepeda, Edward Vul, Doug Rohrer, John T. Wixted, and Harold Pashler. Spacing effects in learning: A temporal ridgeline of optimal retention. *Psychological Science*, 19(11):1095–1102, 2008. https://doi.org/10.1111/j.1467-9280.2008.02209.x.

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

Payel Das, Subhajit Chaudhury, Elliot Nelson, Igor Melnyk, Sarathkrishna Swaminathan, Sihui Dai, Aurelie Lozano, Georgios Kollias, Vijil Chenthamarakshan, Jiri Navratil, Soham Dan, and Pin-Yu Chen. Larimar: Large language models with episodic memory control. In *ICML*, pp. 10109–10126, 2024. https://proceedings.mlr.press/v235/das24a.html.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *ICML*, 2024. https://arxiv.org/abs/2401.06102.

Aidan J. Horner, James A. Bisby, Daniel Bush, Wen-Jing Lin, and Neil Burgess. Evidence for holistic episodic recollection via hippocampal pattern completion. *Nature Communications*, 6:7462, 2015. https://doi.org/10.1038/ncomms8462.

Pentti Kanerva. *Sparse Distributed Memory*. MIT Press, 1988.

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. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, 2020. https://arxiv.org/abs/2005.11401.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023. https://arxiv.org/abs/2301.12597.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *ACL*, pp. 13851–13870, 2024. https://aclanthology.org/2024.acl-long.747/.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *ACL*, 2023. https://aclanthology.org/2023.acl-long.546/.

James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995. https://doi.org/10.1037/0033-295X.102.3.419.

John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander Rush. Text embeddings reveal (almost) as much as text. In *EMNLP*, pp. 12448–12460, 2023. https://aclanthology.org/2023.emnlp-main.765/.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *AAAI*, 2016. https://ojs.aaai.org/index.php/AAAI/article/view/10314.

Tony A. Plate. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641, 1995.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025. https://arxiv.org/abs/2501.13956.

Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for LLMs. In *ICLR*, 2025. https://arxiv.org/abs/2410.14052.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *ACL*, pp. 4498–4507, 2020. https://aclanthology.org/2020.acl-main.412/.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear Transformers are secretly fast weight programmers. In *ICML*, pp. 9355–9366, 2021. https://proceedings.mlr.press/v139/schlag21a.html.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian McAuley. MEMORYLLM: Towards self-updatable large language models. In *ICML*, 2024. https://arxiv.org/abs/2402.04624.

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing Transformers. In *ICLR*, 2022. https://arxiv.org/abs/2203.08913.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. Variational reasoning for question answering with knowledge graph. In *AAAI*, 2018. https://arxiv.org/abs/1709.04071.

A Evaluation Protocols

A.1 Implementation settings

The structured-memory configuration uses Qwen2.5-1.5B-Instruct layer 20 with $d = 1536$. Inputs receive a fixed newline-and-space prefix, whose tokens are excluded from content pooling. Mean pooling is $g(x) = n^{-1} \sum_j h_j$. The learned readout aggregator is $g(x) = \sum_j W_j \, \mathrm{LN}(h_j)/\sqrt{n}$. These representation variants are identified in each readout experiment. With an eigendecomposition of the empirical covariance, $A = U \, \mathrm{diag}((\sigma + \epsilon)^{-1/2}) U^\top$, the whitening offset is $\epsilon = 10^{-2}$; ridge regularization is $\lambda = 10^{-3}$.

The bridge has eight learned query outputs plus one projected target slot, for nine soft tokens and approximately 33M parameters. Layer normalization and a learned gain align their scale with the frozen reader. No LoRA or base-model updates are used. Tree training samples one to three hops and per-hop widths from $\{1, 2, 3, 4, 6, 8, 12\}$. The narrow-tree control uses at most three members. Qwen3-1.7B transfer uses layer 20; Qwen3-8B uses layer 26, dimension 4096, and the 1.91M-string mean-pooling readout configuration reported in Table 2. These model-transfer experiments do not change the backbone of every graph or QA experiment.

MetaQA uses subject-hash sharding. The standard configuration has 256 shards; sensitivity experiments use 64, 128, 256, and 512. Set cleanup uses residual similarity and relative fitted-coefficient criteria, with a development-selected residual similarity threshold of .2. The controlled conversational existence gate uses .5.

A.2 Datasets and scoring

Controlled conversational memory. The 20 test instances contain 2,692 facts and 351 questions: 120 single-value, 57 set, 53 when, 53 latest, and 68 multi-hop questions. Instances contain 4–9 people, temporal recurrences, one-to-many relations, interpersonal chains, and distractors. The relation inventory has 136 labels. Questions are rewritten into natural language without reusing the literal relation phrase. Development instances are disjoint from test instances and are used for threshold selection. The shared generator/parser ontology and small entity populations favor reliable structured parsing; this setting is not intended to approximate unrestricted conversational deployment.

MetaQA. The KB contains 134,741 triples and nine relations. Forward and inverse relation codes allow traversal in either direction. Standard codebook QA samples 300 vanilla test questions per hop and uses 60 development questions per hop for thresholds. The bracketed entity gives the starting cue; the parser predicts a directed relation chain. Matrices are sharded by subject, with 256 shards in the main configuration. The chain-token experiments use their own recorded traversal protocol and reference outputs. Their symbolic reference (.961/.874/.823) differs from standard codebook QA (.902/.929/.888); each neural score is interpreted against its within-protocol reference.

LoCoMo. The extracted temporal KG contains 1,802 facts from ten conversations. Evaluation uses 1,444 questions: category 1 (282), category 2 (321), and category 4 (841). These are the selected multi-hop, temporal, and single-hop categories, not all non-adversarial questions in the dataset. Category 3 and adversarial questions are outside this evaluation. Dense retrieval and structured stores access the same extracted facts. A 444-question diagnostic subset is not used as the headline evaluation because the reported KG answer coverage is .685 on that subset and .389 on the remaining questions.

Readout data. Training sets contain 0.2M, 1M, or 3.81M Wikidata5M-derived names and aliases for the 1.5B configurations, and 1.91M strings for the larger Qwen3-8B mean-pooling configuration. Some runs additionally mix approximately 9.7k single-token aliases. Evaluation uses 5,000 held-out Wikidata strings, 2,156 synthetic strings, and 2,000 PopQA entity strings. PopQA is used here as a source of strings for reconstruction, not as an end-to-end QA leaderboard. The single-value tests are string-disjoint from adapter training. They do not establish absence from base-model pretraining or absence of semantic overlap through aliases.

Explicit-store controls. The symbolic baseline performs exact fact lookup with the same query parser, entity resolution, and relation fallback as the associative store. Growing vector KV retains the constructed key and original fact and uses a .5 similarity threshold. It can miss or include records under that threshold definition. Exact hash lookup is a different control, with perfect retrieval given a valid complete key; a matrix advantage over threshold KV is not an advantage over an exact dictionary.

Retrieval controls. Conversational RAG embeds naturalized fact strings using Titan Embed v2. The two-step variant retrieves initial evidence and uses an LLM-written follow-up query to obtain additional evidence. Budgets of 12, 24, and 40 facts are reported separately. These budgets differ from PRAM's 14-candidate cap and do not equate token usage, parser cost, or number of generation calls. MetaQA baselines index triple text using word unigram/bigram TF–IDF and use Qwen or Claude as reader. They should not be described as dense retrieval.

Scoring and randomness. Candidate identification uses nearest-neighbor recovery of the intended target; repeated target values are counted correct. Transcription EM requires exact output reconstruction under the experiment's string normalization. Graph QA uses programmatic answer-set F1 and exact match. Conversational QA uses a temperature-zero semantic judge. Three code seeds are used for the main synthetic and MetaQA matrix results; capacity experiments use two seeds; neural readout does not have a full training-seed replication. Paired bootstrap intervals use 5,000 question-level resamples. Such resampling conditions on the sampled conversations and does not estimate uncertainty over new conversations or over judge/model revisions.

B Memory Construction and Query Procedures

Coordinate systems. A memory build fixes encoder preprocessing, representation type, whitening statistics, and random-code seeds. Keys and queries must use the same transforms. The target representation for scalar neural readout must match the bridge's training coordinates. For set cleanup, targets are normalized whitened vectors; for single-value readout they are the corresponding pooled or compressed vectors. Changing a whitening transform or compressor after writing changes the coordinate system and generally requires rebuilding the affected memory. Therefore, a batch ridge solution does not by itself demonstrate a continual system with stable preprocessing.

Ridge and RLS. Equation 5 can equivalently be written as

$$M = V^\top (KK^\top + \lambda I_n)^{-1} K. \tag{15}$$

The primal and dual forms are algebraically equivalent for $\lambda > 0$. A streaming implementation with fixed codes maintains an inverse covariance P. With $P_0 = \lambda^{-1} I$ and $M_0 = 0$, one update for key k and value v is

$$a = \frac{Pk}{1 + k^\top Pk}, \tag{16}$$

$$M \leftarrow M + (v - Mk)a^\top, \qquad P \leftarrow P - ak^\top P. \tag{17}$$

These recursions specify the ridge-associated write rule used for structured-memory construction. The decay and consolidation experiments evaluate the separate dynamic write conditions described in Section 4.4. Maintaining P also requires an additional d^2 state during online updates.

Set cleanup. Let D contain unit target codes as rows and $z = M_s k$. Initialize selected indices $S = \varnothing$ and residual $u = z$.

1. If an existence gate is enabled, reject the query when $\max_j \cos(D_j, z)$ is below its threshold.
2. Select an unchosen candidate j maximizing $\cos(D_j, u)$. Stop if this similarity is below the peeling threshold, or if the algorithm would repeat a selected member.
3. Let B contain rows indexed by $S \cup \{j\}$. Fit $\alpha = \arg\min_a \|z - B^\top a\|_2^2$.
4. If the new coefficient is too small relative to the largest fitted coefficient, stop. Otherwise accept j, set $u = z - B^\top \alpha$, and repeat until the member cap or a numerical residual stopping condition is reached.

The coefficients are refitted against the original readout z, rather than subtracting unit vectors blindly. Set outputs are deduplicated. The existence gate is a confidence heuristic; a large true bucket may also have a low maximum member similarity. The method therefore does not treat this gate as an exact test of whether a fact exists.

Query routing. The parser returns a head, up to three candidate final relations, an optional interpersonal chain, a value hint, an optional time, and the requested answer type. Entity resolution combines exact, lexical, and vector matching. Relation fallback selects plausible alternatives among supported relations, with interpersonal-chain fallbacks restricted to interpersonal relations. For each intermediate frontier, the set map provides the next entities. Final members are associated with time through M_{t}; explicit-time queries can additionally use M_{v}. Candidate facts are merged and capped before textual reading. The same broad routing logic is used by the explicit-store controls.

Temporal multiplicity. A single-valued map cannot represent arbitrary conflicting values at an identical key. In particular, $(s, r, v) \mapsto \tau$ is ambiguous if the same event/value recurs at different times without additional distinguishing information. Set buckets recover distinct members rather than event multiplicity. Results on the selected temporal tasks should not be interpreted as a general solution to event identity, temporal normalization, or contradiction resolution.

Implementation settings. The default encoder is Qwen2.5-1.5B-Instruct at layer 20 with $d = 1536$; ZCA uses $\epsilon = 10^{-2}$ and matrix fitting uses $\lambda = 10^{-3}$. Synthetic existence similarity is .5 and the reported peeling threshold is .2. QA candidates are capped at 14. MetaQA uses 256 subject-hash shards, a frontier cap of 40, and intermediate/final member caps of 20/40. The neural bridge produces eight query slots and one raw projected slot, with approximately 33M parameters. Dataset-specific gates and coefficient thresholds must be distinguished from the shared memory parameters.

C Rank, Interference, and Association Capacity

Table 3: Grouped-subject MetaQA buckets. NN identifies the correct stored bucket value (duplicate targets count as correct). Raw and white refer to the representation settings. Cosine alone need not track identification.

Buckets / subjects	Key	Rank	Cosine	NN
1,000 / 408	Concatenation, raw	418	.924	.462
	Concatenation, white	418	.738	.577
	Binding, raw	1,000	.970	.975
	Binding, white	1,000	1.000	1.000
5,400 / 2,126	Concatenation, raw	1,551	.683	.173
	Concatenation, white	1,551	.619	.518
	Binding, raw	1,536	.507	.127
	Binding, white	1,536	.470	.901

Proposition 1 (Additive rank restriction)**.** *Suppose each key concatenates fixed subject, relation, and time codes drawn from n_E, n_R, and n_T distinct fields. Its key matrix satisfies*

$$\mathrm{rank}(K) \le \min\{n, d_{\mathrm{cat}}, \mathrm{rank}(HE) + \mathrm{rank}(RC) + \mathrm{rank}(TF)\} \le \min\{n, d_{\mathrm{cat}}, n_E + n_R + n_T\},$$

where T and F are the time-incidence and time-code matrices. Dropping time removes that term. Multiplying nonzero rows by normalization scalars does not increase rank.

Proof. Each block is the product of a field-incidence matrix and its field-code matrix. The column space of their concatenation is contained in the sum of their column spaces, so its dimension is at most the sum of block ranks. Each block rank is at most its number of field types. Row normalization is multiplication by an invertible diagonal matrix when all row norms are nonzero, and therefore preserves rank. □

The field-count sum is an upper bound, not a formula for exact rank. The 418 rank observed with 408 subjects is a measurement, not a universal equality. Binding has no corresponding additive block

factorization, but its rank is still at most $\min(n, d)$. At 5,400 buckets and $d = 1536$, "full rank" means full column rank, not 5,400 independent addresses.

A threshold model for set cleanup. For mutually orthogonal unit codes $e_1, \ldots, e_k$ and residual $u_m = \sum_{i=m+1}^{k} e_i$ after exact removal,

$$\|u_m\|_2 = \sqrt{k-m}, \qquad \cos(e_j, u_m) = \frac{1}{\sqrt{k-m}} \quad (j > m). \tag{18}$$

With threshold η, initial acceptance in this idealized model requires $k \leq \eta^{-2}$. At $\eta = .3$, a 20-member orthogonal sum has initial similarity about .224 and fails the gate; it does not support the claim that precisely nine members can be recovered. After successful correct removals the signal increases. Real codes are correlated and readouts contain error, so the expression is a diagnostic scaling law, not a general recovery guarantee.

Identification beyond rank. If keys are approximately orthogonal, a Hebbian readout can be written as $Mk_i = v_i + \sum_{j \neq i} v_j (k_j^\top k_i)$ for unit self-keys. Retrieval depends on whether interference changes the nearest candidate, not on reconstructing every coordinate exactly. Whitening targets can improve this decision margin. Thus, near-perfect finite-codebook recognition at $n \gg d$ is compatible with low vector reconstruction fidelity, and does not violate the rank constraint on exact interpolation.

C.1 Random-association capacity

Table 4: Random-association nearest-neighbor recall using one 1536^2 map and Qwen codes. The two first columns whiten both bound keys and targets; the final column retains raw targets. Approximate high-load Hebbian parity is reported qualitatively because exact values were not provided.

Facts	Whitened Hebbian	Whitened RLS	RLS, raw targets
500	1.000	1.000	—
2,000	1.000	1.000	—
4,000	1.000	1.000	.87
8,000	1.000	1.000	.47
12,000	1.000	1.000	.26
16,000	≈ RLS	1.000	.15
24,000	≈ RLS	1.000	.07
32,000	≈ RLS	.996–.999	.03

These random facts differ from the grouped-subject rank experiment and from BFS neighborhoods. The reported concatenated-RLS recall is 1.000 at 4,000, .999 at 8,000, .989 at 12,000, .969 at 16,000, .906 at 24,000, and .819 at 32,000. This does not contradict grouped-key failure: the incidence structure and distinct-subject counts determine the effective rank. Capacity claims should specify the sampling process and the candidate dictionary together with n and d.

D Supporting the Neural Readout Analysis

For individual-target transcription, we compare greedy decoding with an auxiliary re-encoding verifier. Given generated candidates $\mathcal{C}_K(z)$, it selects

$$\hat{x} = \arg \max_{x \in \mathcal{C}_K(z)} \cos(g_*(x), z), \tag{19}$$

where g_* matches the tested representation. Candidates are generated by the reader; this step adds generation and encoding work. It is evaluated for transcription separately from the tree-input mechanism.

The learned aggregation variant uses content states h_j and position-dependent projections:

$$g_\theta(x) = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} W_j \, \mathrm{LN}(h_j). \tag{20}$$

The frozen encoder remains unchanged while the compressor and bridge are learned. Mean pooling provides the parameter-free counterpart. The comparisons below test the representation and readout choices needed to interpret Table 2.

Table 5: Representation and adapter ablations. Entries are direct-string EM on held-out Wikidata / synthetic / PopQA strings, with greedy decoding. Fixed-prefix status and training-string count are explicit.

Configuration	Wiki	Synthetic	PopQA
Last token, unprefixed, 0.2M	.019	.118	.178
Last token, deeper bridge, 0.2M	.033	.152	.205
Last token, layer 12, 0.2M	.032	.158	.203
Last token, 1M	.060	.228	.256
Last token plus prefix, 0.2M	.012	.106	.256
Mean, skip first position, unprefixed	.210	.317	.372
Mean plus prefix / single-token training, 0.2M	.435	.593	.796
Compressor plus prefix / single-token training, 0.2M	.458	.609	.785
Mean plus prefix, 1M	.536	.670	.847
Compressor plus prefix, 1M	.588	.721	.863
Compressor plus prefix, 3.81M	.631	.790	.883
Full token sequence, uncompressed reference	.873	.949	.971

First-position diagnostic. In the tested unprefixed extraction setting, single-token strings have nearly identical codes (pairwise cosine 1.000) and norms around 11,000 rather than the typical 60–90. Qwen3-8B exhibits a similar large-norm first-position effect (12,511 in the diagnostic). A fixed prefix moves content away from that position and restores variation, with pairwise cosine approximately .50–.60. This is an observed encoder/preprocessing pathology; assigning all of its cause to an attention-sink mechanism would require further intervention. Single-token values constitute approximately 2.1% of the synthetic set, 7.7% of LoCoMo, and 4.0% of MetaQA entities.

Why aggregate tokens? Prefixing a last-token code does not by itself give good reconstruction. The data support aggregation as a way to preserve more recoverable string information, rather than the stronger claim that final-token states contain no information about the string. Learned compression provides an additional gain at larger training scales, while an uncompressed sequence serves as a less constrained representation reference.

Candidate count and training scale. With 1M training strings, $K = 8$ reranking gives EM of .744, .830, and .942 for mean pooling on Wikidata, synthetic, and PopQA strings, respectively; learned compression gives .768, .880, and .950. With 3.81M strings, learned compression gives .807, .924, and .959 at $K = 8$, and .834, .936, and .966 at $K = 16$. These results supplement the greedy-versus-reranked contrast in Table 2; they also expose the additional generation and re-encoding work.

Table 6: Length effects on synthetic transcription. Direct compression uses $K = 16$; matrix-conditioned compression uses $K = 8$, so the rows isolate different protocols rather than a matched corruption gap.

Protocol	1 token	2 tokens	3 tokens	4 tokens	5+
Direct code	1.000 (pooled)		.995	.953	.821
Matrix output	.930	.885	.859	.822	.707

One- and two-token strings form a pooled bin in the direct-code length analysis. Matrix readout has .970 mean read-back cosine and .910 codebook identification in the corresponding comparison. Date-valued and temporal associations are prominent among reconstruction errors, consistent with interference between related facts.

Backbone transfer. At 0.2M training strings, prefix-and-pooling gives Wiki/synthetic/PopQA EM .372/.494/.747 with Qwen3-1.7B and .508/.596/.829 with Qwen3-8B, compared with .435/.593/.796 for Qwen2.5-1.5B. The 8B last-token control gives .129/.278/.368. The representation effect thus extends beyond one tested backbone, although the 8B model also changes the state dimension to 4096 and is not a matched storage-cost comparison.

Readout failures. A tested training-free hidden-state reinjection configuration inspired by Patchscopes gives near-zero transcription EM; this is a result for that configuration, not the Patchscopes framework as a whole. Binding token positions with random bipolar carriers and summing also gives near-zero reconstruction in the tested setting. Direct list generation from sums of compressed member vectors has set F1 at most .08. These results motivate explicit set cleanup before member-level neural generation. A deeper bridge alone does not compensate for the information bottleneck of the tested last-token representation. A matched vec2text baseline and multiple adapter-training seeds have not been evaluated.

E Supporting the Chain-Token Analysis

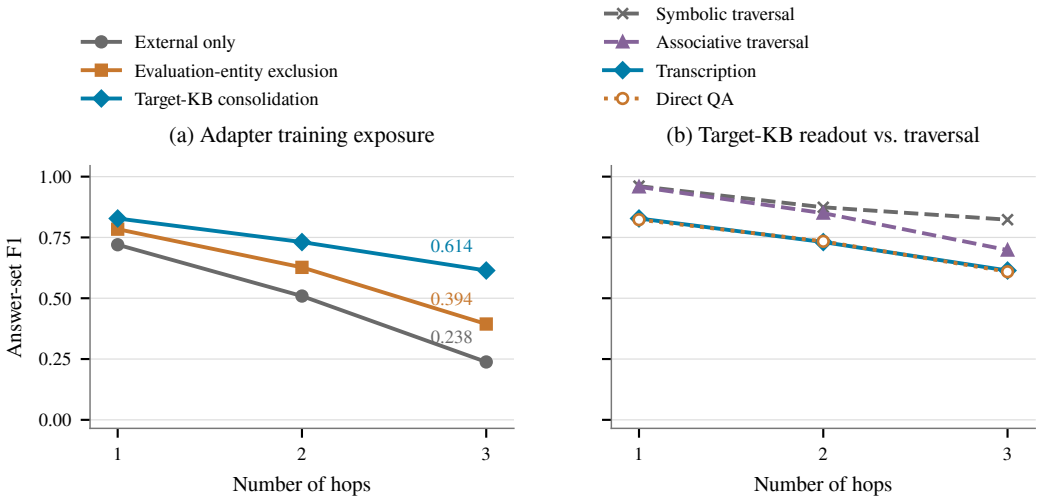

Figure 5: MetaQA chain-token answer-set F1 over 300 questions per hop. (a) Adapter-training exposure affects multi-hop transcription. (b) Under target-KB consolidation, transcription and direct QA are similar; dashed traversal references score retrieved leaves before generation. All neural curves use associative traversal. These references belong to the chain protocol, separately from codebook QA. Exact scores appear in Table 7.

Table 7: MetaQA chain-token answer-set F1, 300 questions per hop. All rows use true associative traversal unless explicitly symbolic. Training exposure is part of the setting. The traversal reference scores leaves before neural generation under this chain protocol.

Training / output condition	1 hop	2 hops	3 hops
External-only, transcription	.720	.509	.238
Evaluation-entity exclusion, transcription	.784	.627	.394
Target-KB consolidation, transcription	.828	.731	.614
Target-KB consolidation, QA direct answer	.823	.734	.609
Associative traversal reference	.958	.850	.698
Symbolic traversal reference	.961	.874	.823

For the two wide matrix-transcription runs, mean F1 is .831/.742/.608. Relative to the corresponding traversal reference, these ratios are approximately 87%, 87%, and 87%. They describe the consoli-

Table 8: Tree-width controls under target-KB consolidation. "Readout" scores generated terminal answers; "traversal" scores the retrieved terminal set before neural generation. Width changes the training distribution, not merely the generation limit.

Carrier / training width	Readout F1: 1 / 2 / 3	Traversal F1: 1 / 2 / 3
Symbolic, at most 3 members	.843/.686/.348	.961/.874/.823
Matrix, at most 3 members	.841/.686/.314	.958/.851/.697
Symbolic, wide sampling	.830/.754/.691	.961/.874/.823
Matrix, wide sampling	.828/.731/.614	.958/.850/.698
Matrix, wide sampling, second seed	.834/.752/.601	.958/.850/.698
Matrix, wide sampling, direct QA	.823/.734/.609	.958/.850/.698

dated setting only. External-only and evaluation-entity-excluded scores in Figure 5(a) must remain visible when discussing generalization.

Exposure semantics. Consolidation samples trees from MetaQA and Wikidata, so target entity and answer strings may occur during training. Evaluation-entity exclusion removes the evaluation heads and gold-answer strings from training trees; it does not necessarily remove all intermediate entities, aliases, or overlapping neighborhoods. External-only training uses Wikidata with MetaQA entity strings excluded. These controls probe different transfer questions. Filtering by evaluation entities is an evaluation protocol, not information available to a deployed memory system.

What the reader receives. The structural tokens convey root/context labels and relation/tree organization, while bridge tokens represent members. The reader does not independently discover edges outside that supplied tree. Transcription and QA modes therefore evaluate whether it can use the retrieved structure. The remaining codebook performs cleanup and entity resolution before generation; it must be included in storage even though member strings are not dictionary-decoded during generation.

F Conversational and Graph QA Details

Table 9: Conversational codebook-path QA accuracy over shared fact collections. The synthetic dense-RAG rows use LLM-guided two-step retrieval; LoCoMo uses top-k retrieval. Budgets count facts and do not match token usage or total LLM calls.

Method	Controlled conversations	LoCoMo
PRAM	.787	.380
Symbolic, shared routing	.798	.381
Growing KV, shared routing	.809	.381
Dense RAG, 12 facts	.698	.553
Dense RAG, 40 facts	.761	.584
All-fact context	.852	—

Evidence-selection diagnostic. On LoCoMo, matrix, KV, and symbolic accuracy is nearly identical, with candidate-answer coverage .451/.450/.450. Holding dense retrieval and its reader prompt fixed, original values score .517 and matrix-transformed values .501, whereas structured matrix access scores .380. This comparison locates a larger loss in evidence selection than in value transformation. It is a three-condition diagnostic rather than a complete factorial decomposition.

Partial-cue ablations. On controlled conversations, the full construction scores .787, the value-map-only ablation .080, and the no-time ablation .610 overall. Without temporal indexing, when accuracy is .000 and latest accuracy is .094.

The single-map and no-time ablations in the main text remove the corresponding indices throughout retrieval. Concatenated keys yield .607 overall synthetic accuracy and .386 set accuracy, compared

Table 10: Controlled conversational accuracy. Ranges in the PRAM task columns are reported seed ranges; the overall value is the reported three-seed summary. Blank experiment cells are marked with a dash.

Method	Single	Set	When	Latest	Multi	All
PRAM	.900	.912–.947	.623	.717	.632–.676	.787±.002
Symbolic	.892	.895	.660	.755	.691	.798
Growing KV	.892	.947	.660	.774	.691	.809
RAG two-step, 12 facts	.833	.754	.943	.604	.294	.698
RAG, 24 facts	—	—	—	—	.191	.732
RAG two-step, 40 facts	.883	.930	.943	.642	.353	.761
All-fact context	.900	.912	.925	.642	.824	.852

with .787 and .912–.947 for the complete bound-key construction. This links the grouped-key geometry analysis to the set access needed by conversational queries.

Statistical interpretation. Paired question-level bootstrap comparisons indicate an overall advantage over 24-fact RAG and no significant overall difference from 40-fact RAG; the latter comparison favors PRAM on multi-hop questions and RAG on when questions. The displayed three-seed mean differences are .055 and .026, respectively. These mean differences and code-seed standard deviations should not be interpreted as confidence intervals over new conversations or independent bridge-training runs.

Table 11: MetaQA codebook-path results on 300 questions per hop. Entries for the first three rows are F1 / exact. Retrieval and single-value rows report F1 only. TF–IDF is a lexical retriever.

Method	1 hop	2 hops	3 hops
Symbolic, same parsed path	.902/.840	.929/.640	.888/.457
PRAM, three code seeds	.899/.833	.888/.563	.752/.180
Concatenated keys	.597/.320	.466/.047	.247/.000
Single-value map	.754	.540	.225
TF–IDF two-step, $k = 12$, Qwen	.688	.094	.172
TF–IDF, $k = 12$, Claude	.829	.088	.086
TF–IDF, $k = 40$, Qwen	.706	.091	.188

Standard deviation across the three PRAM code seeds is at most .002 for the reported F1 values. Shard-count sensitivity over 64/128/256/512 shards gives relatively stable one- and two-hop results, with three-hop F1 approximately .73–.78 across the reported sweep. The existence of useful overloaded shards does not eliminate the larger-load breakdown observed in the BFS experiment below.

LoCoMo task detail and relation expansion. PRAM scores .372, .505, and .335 on multi-hop, temporal, and single-hop questions, respectively. Dense top-40 retrieval gives .61, .70, and .53. Expanding the structured relation candidates with two alternatives raises overall accuracy from .380 to .397 and candidate-answer coverage from .451 to .460. This particular expansion strategy does not close the retrieval gap. Coverage failure can reflect extraction omissions, entity resolution, relation choice, time constraints, or candidate truncation; these mechanisms are not individually identified by the aggregate scores.

G Load, Sharding, and Storage

The single matrix has a reported footprint of approximately 4.7 MB. Whitening state and codebooks are additional. At small loads the matrix outperforms threshold vector retrieval on several metrics; an exact dictionary would retrieve exact valid keys perfectly. At larger loads, sharding buys accuracy with growing parameter storage. These BFS results differ from random-association capacity because they include related keys, graph degree variation, and set cleanup.

Table 12: Exact-cue BFS-neighborhood stress test on MetaQA, two seeds. Entries are single-value top-1 / set F1 / two-hop F1. Triple counts refer to sampled triples before bidirectional storage. Memory columns are reported matrix/index footprints, with codebooks separate.

Triples	Single matrix	Sharded matrices	Vector KV	Shards MB	KV MB	Codes MB
100	1.00/1.00/.99	1.00/1.00/.99	.92/.97/.88	4.7	.6	.3
300	1.00/1.00/.98	1.00/1.00/.98	.94/.91/.87	4.7	1.9	.9
1,000	1.00/1.00/.99	1.00/1.00/.98	.96/.88/.88	14	6.2	2.6
3,000	1.00/.95/.87	1.00/1.00/.99	.97/.92/.88	38	18.5	6.9
10,000	.79/.89/.67	1.00/1.00/.98	.94/.96/.86	113	62	19
30,000	.14/.19/.02	1.00/1.00/.98	.92/.94/.82	293	185	45

Merged memories. Merging all 20 synthetic instances gives .735 QA accuracy (.702 on set questions), compared with .806 for growing KV and .792 for symbolic lookup. Merging the ten LoCoMo conversations gives .359 versus .362 for KV. Natural-question performance can therefore degrade before a nominal triple-count threshold suggested by an exact-cue experiment: bucket counts, temporal collisions, gates, and candidate selection differ.

Full-state accounting. For $d = 1536$, one matrix occupies 4.72 MB at FP16 or 9.44 MB at FP32; three maps occupy 14.16 or 28.31 MB. These are analytical parameter footprints, not total system sizes. Each retained dense whitening transform and online RLS inverse-covariance matrix adds another d^2 array. A single FP16 codebook vector occupies 3,072 bytes before labels and metadata. Bridge weights add approximately 66 MB at FP16 for 33M parameters, plus any learned compressor; the bridge may be shared across instances. Encoder and reader weights are additional shared costs.

A compact merged-synthetic inventory is estimated at approximately 28 MB versus 33 MB for growing KV, excluding shared language-model/adapter weights and depending on retained whitening state. Consequently, no system-level storage advantage follows from matrix size alone. A reproducible comparison must specify precision, per-instance versus shared state, codebooks, transforms, and whether online-update state is retained.

H Dynamic Retention Experiments

Probe design and metric. The experiment uses Qwen-derived codes of dimension 1536 over a 200-day simulated stream. Background facts appear once. The standard load introduces 400 new background facts per day, totaling approximately 87,000 writes including scheduled probes; the higher load introduces 2,000 background facts per day, totaling approximately 400,000 background facts. A day is an abstract update interval. For probe i, effective recall is

$$R_{\text{eff}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\{\hat{t}_i = t_i\}\, \mathbf{1}\{a_i \geq \theta\}, \tag{21}$$

where a_i is the tested memory-strength score and θ is its threshold. Thus, the metric tests accessible, correctly identified associations. Reported entries average three seeds, with hyperparameters selected before examining these outcomes.

Single-exposure probes vary the interval between exposure and the final test. Repetition probes vary exposure count; spacing probes keep three exposures and vary their gap. The delayed repetition and spacing tests use a 90-day final-exposure lag. The reactivation sweep varies scheduled refreshes. Each panel preserves its reported probe schedule, so the one-exposure and zero-reactivation entries are not pooled across panels.

Write conditions. The no-decay condition accumulates Hebbian associations. The single-timescale condition multiplies its state by a decay factor each day. Slow-first routing directs novel content to a slow layer and repeated content to a fast layer based on surprise. Fast-first consolidation initially writes to the fast layer and uses prediction error at subsequent exposures to consolidate part of the association into a slow layer. The tested fast-layer half-life is approximately 6.6 simulated days.

Higher-load results. Table 14 lists the available scores at 2,000 background facts per day. Missing entries are not inferred from the lower-load condition. No-decay and slow-first routing are reported

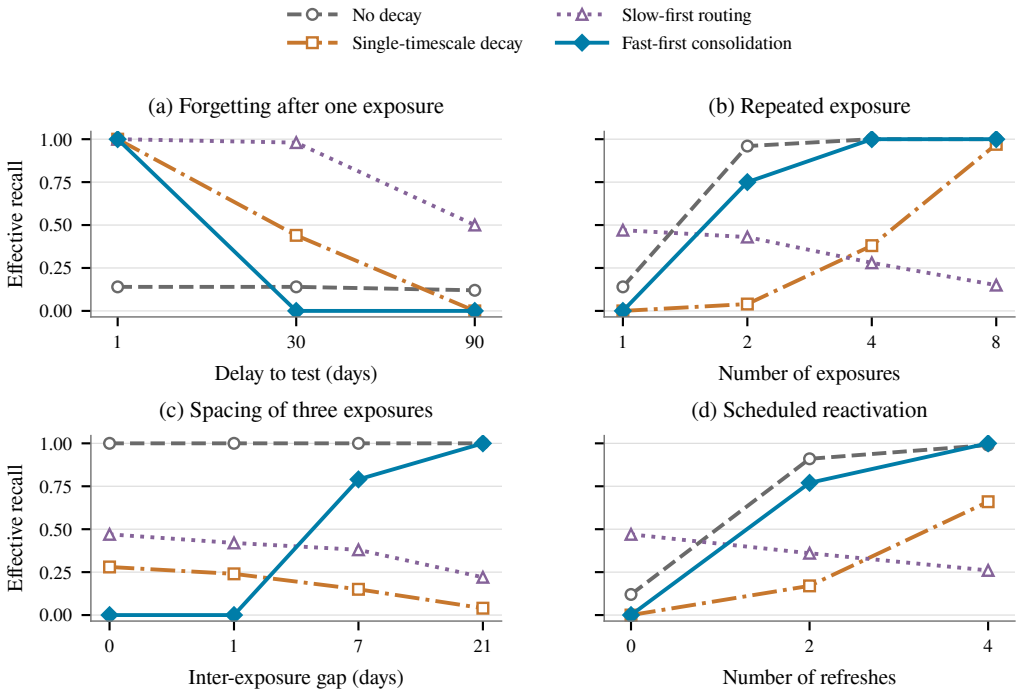

Figure 6: Selective retention with 400 background facts per simulated day. Panels vary (a) single-exposure delay, (b) exposure count, (c) spacing between three exposures, and (d) scheduled refresh count. Repetition and spacing use a 90-day final-exposure lag. Horizontal positions enumerate tested settings at equal spacing; connecting segments guide the eye. Values are three-seed means; per-seed dispersion is unavailable in the manuscript, so no error bars are shown. Full values and panel-specific schedules appear in Appendix H.

Table 13: Complete effective-recall sweeps at 400 background facts per day. All values are reported means over three seeds. Each block is a separate probe family. "Gap" is the interval between three exposures. Reactivations denote scheduled refresh updates.

Single exposure: delay to test				
Write rule	1 day	30 days	90 days	
No decay	.14	.14	.12	
Single-timescale decay	1.00	.44	.00	
Slow-first routing	1.00	.98	.50	
Fast-first consolidation	1.00	.00	.00	
Repetition: number of exposures				
Write rule	1	2	4	8
No decay	.14	.96	1.00	1.00
Single-timescale decay	.00	.04	.38	.97
Slow-first routing	.47	.43	.28	.15
Fast-first consolidation	.00	.75	1.00	1.00
Spacing: gap between three exposures				
Write rule	0 days	1 day	7 days	21 days
No decay	1.00	1.00	1.00	1.00
Single-timescale decay	.28	.24	.15	.04
Slow-first routing	.47	.42	.38	.22
Fast-first consolidation	.00	.00	.79	1.00
Reactivation: number of scheduled refreshes				
Write rule	0	2	4	
No decay	.12	.91	.99	
Single-timescale decay	.00	.17	.66	
Slow-first routing	.47	.36	.26	
Fast-first consolidation	.00	.77	1.00	

near .01 across the tested probes. Under fast-first consolidation, a four-exposure probe remains at .93, a 21-day spacing schedule at .99, and four reactivations at .96.

Table 14: Available higher-load effective-recall results. Dashes indicate unreported individual entries; the near-.01 descriptions for no decay and slow-first routing are aggregate qualitative summaries.

	Exposures		Gap	Reactivations
Write rule	4	8	21 days	4
Single-timescale decay	—	.48	—	—
Fast-first consolidation	.93	—	.99	.96

Interpretation. The non-decaying matrix retains repeated probes under the standard load but fails to preserve recent single-exposure probes. Uniform decay favors recency, with lower delayed recall as exposure spacing increases. Fast-first consolidation combines a short acquisition timescale with long retention for repeated and spaced probes. Slow-first routing exhibits the opposite repetition trend in this test. The findings concern the reported update rules, schedules, and effective-recall metric. Neural answering and association-tree traversal are not part of this retention experiment.

I Supplementary Diary-Stream Diagnostics

A separate synthetic diary benchmark tests frequency aggregation and exact event recall with a frozen Qwen2.5-1.5B reader. Diary entries are written only to associative memory and are absent from the answer prompt. This configuration uses Hebbian fast–slow states, surprise-based routing, and a trained gated bridge; it is distinct from the graph-access and consolidation configurations in the main evaluation. Tests use people excluded from training.

Table 15: Diary-stream accuracy for habit and day-specific recall in the separate gated-bridge configuration. These scores are not pooled with the current graph or conversational benchmarks.

Task	Associative memory	No memory	RAG@5	Long context
Most frequent activity	1.00	.00	.35	.65
Activity on a specified day	.80	.00	1.00	.925

Habit queries benefit from repeated writes to the same association. With fully disjoint training and test vocabularies, the reported habit accuracies are .95 for associative memory, .53 for RAG, and .83 for long context. Exact day-specific recall places different demands on the representation. A separate temporal-key comparison improves accuracy from .50 to .75; it is a distinct configuration from the .80 row in Table 15.

Stream length also affects the neural interface. Training on 20-day diaries and evaluating on 60- or 120-day streams reduces habit accuracy from 1.00 to approximately .13 without scale-invariant readout. With normalized readout, the 120-day score is 1.00. This control addresses readout scale under accumulation rather than time-based forgetting. In a separate representation control, raw BGE sentence codes become difficult to retrieve after approximately 20 superposed items, while contrastively separated representations retain .98–1.00 retrieval at 200 items. These diagnostics motivate distinguishing representation separation, readout scale, and dynamic retention.