Recall is a proposal

Kenneth Pernyér, Reflective Labs
Stockholm, Sweden
kenneth.pernyer@reflective.se · reflective.se · ORCID 0009-0006-4543-1156

Retrieval-augmented generation solved the wrong half of the memory problem. It made relevant text available to a model; it didn't make the retrieved text accountable, and in the standard arrangement the retrieved passage is concatenated into a prompt where it's indistinguishable from instruction. This paper describes Mnemos, the memory extension of my convergence substrate, which takes the opposite position: a recalled item is a *proposal*, carrying its provenance, subject to the same promotion gate as any other proposal, and incapable of becoming a fact without passing it. Two consequences follow. The first is a security property that falls out of the architecture rather than being added to it: retrieved text can't acquire authority, so indirect prompt injection reaches the context as an attributed proposal that policy may refuse. The second is a retrieval result. Mnemos ranks vector similarity and BM25 separately and merges the *rankings* by reciprocal rank fusion rather than merging the scores, and I show why this isn't a tuning preference: the two scores live on incomparable scales, and rank fusion is invariant under any monotone rescaling of either, which score fusion isn't. I close on the tension I haven't resolved — an append-only substrate and a memory that must forget aren't obviously compatible, and I state the problem rather than claim it solved.

1. Introduction

An organization's memory isn't a document store. It's the set of things the organization can be held to having known, which is a stronger and more uncomfortable notion. A retrieval system that returns a relevant passage has answered a question about similarity; a memory that an institution can rely on must also answer *where did this come from*, *who may act on it*, and *what happens if it's wrong*.

The dominant pattern doesn't attempt this. Retrieval-augmented generation (Lewis et al., 2020) fetches passages and concatenates them into a prompt, where they sit alongside the system instruction and the user's request in one undifferentiated sequence of tokens. The model can't tell retrieved evidence from retrieved instruction, because at that point there's no difference: everything is text in a context window. This is the mechanism behind indirect prompt injection (Greshake et al., 2023), and the reason it's hard to fix inside the pattern is that the pattern's whole idea is to erase the boundary that would have contained it.

Mnemos takes the position that recall is a proposal. A retrieved entry enters the shared context the same way any Suggestor's output enters it — as a typed proposal carrying the identity and version of its producer, the query it answered, and its provenance — and it becomes a fact only if the promotion gate of my substrate admits it (Pernyér, 2025a). Nothing that memory returns is load-bearing until the engine says so.

Layer	Owns
Converge	Context, proposals, facts, promotion, the Suggestor contract
Mnemos	Storage, recall, ingestion, agentic memory, feedback learning, recall Suggestors
Product	Which stores, tenancy, credentials, retention, deployment

Table 1 | The boundary. Mnemos never promotes and never decides; it proposes, and everything it proposes is attributed. Retention is deliberately a product concern — how long an organization may keep a fact is a legal question, not a retrieval one.

2. What memory owns

Converge owns governed promotion; Mnemos owns memory. The boundary is worth stating in the form the implementation enforces, because a memory layer is exactly the sort of component that accretes authority by accident.

Concretely, Mnemos contributes two Suggestors — one that retrieves and one that stores — a knowledge base with hybrid retrieval, embedding support, markdown and rich-media ingestion, a set of agentic memory modules, and a feedback and replay path. Its proposals carry a typed provenance marker, and every crossing of the Suggestor boundary is traced.

3. Two scales that can't be averaged

The retrieval question Mnemos has to answer is mundane and its resolution is the most transferable thing in this paper. Semantic search finds passages that mean the same thing; lexical search finds passages that use the same words. An organizational memory needs both, because half of what an organization knows is carried in acronyms, product codes, ticket numbers and names that embeddings smooth away.

So run both and combine. The obvious combination — normalize the scores and take a weighted sum — is wrong, and it's wrong for the same reason the previous paper in this series gave for not averaging a solver's certificate with a model's probability. A BM25 score (Robertson and Zaragoza, 2009) is an unbounded term-weighting sum whose scale depends on the corpus statistics of the query terms. A cosine similarity is bounded in $[-1, 1]$ and depends on the embedding model. There's no principled conversion between them, and any normalization imposes one by fiat.

Mnemos therefore fuses ranks, not scores. For a document d retrieved by systems R, with $r_i(d)$ its one-based rank in system i,

$$\mathrm{RRF}(d) = \sum_{i \in R} \frac{1}{k + r_i(d)}, \quad k = 60, \tag{1}$$

following Cormack, Clarke and Buettcher (Cormack et al., 2009), from whom the constant also comes. The property that justifies this isn't that it performs well, though it does; it's that it can't be misled by the scales.

> **Proposition (Scale invariance).** Let φ be any strictly increasing function. Replacing a retriever's score s by $\varphi(s)$ leaves every rank $r_i(d)$ unchanged, and therefore leaves Eq. 1 unchanged. Score-weighted fusion has no such invariance: for a weighted sum $\alpha s_1 + \beta s_2$, a rescaling of s_1 alone can reverse the induced order.

The proof is one line and appears in Appendix B; the content is in what it licenses. Because fusion is invariant to monotone rescaling, Mnemos can change embedding models, switch tokenizers, or re-index a corpus with different statistics without re-tuning a fusion weight. The retrieval quality changes; the combination rule doesn't need to.

The cost is equally clear and I don't hide it. Rank fusion discards *magnitude*: a document that's overwhelmingly the best lexical match and one that's marginally the best contribute the same $\frac{1}{k+1}$. Where the score gap is meaningful — a near-exact identifier match, say — RRF is throwing away information that a calibrated system could have used. I accept that trade because I don't believe my scores are calibrated, and a system that pretends otherwise is worse than one that declines to.

4. Recall as a proposal

A retrieval result in Mnemos isn't injected anywhere. It's proposed.

The proposal carries the retrieving Suggestor's identity and version, the query it answered, the entries it found with their fused ranks, and the provenance of each entry — which store, which ingestion, which source document. The promotion gate then applies the three checks of the substrate: is this Suggestor permitted to assert this fact type in this context, does the payload match its declared schema, and is confidence above the required threshold.

Three properties follow, and only the first is about retrieval.

Attribution. A fact derived from memory can always be traced to the entry it came from and the ingestion that created it. "Why did the system believe this?" terminates in a document, not in a context window.

Refusability. Policy can decline a recalled item for reasons that have nothing to do with relevance — tenancy, retention, jurisdiction, clearance. The gate already evaluates authority per proposal, so memory inherits authorization without implementing any.

Injection containment. This is the property I didn't design for and value most. Suppose an ingested document contains text crafted to read as an instruction. Under the RAG pattern it arrives in the prompt with the same standing as the system's own instructions. Under this pattern it arrives as a proposal from a memory Suggestor, attributed to a document, and it can become a fact only if a policy permits *that Suggestor* to assert *that fact type* in *that context*. The attack doesn't disappear — a malicious passage can still mislead a model that reads it — but it can't escalate: text in the store has no path to authority that doesn't pass a gate that knows where the text came from.

> **Proposition (No authority from ingestion).** For any entry e in the knowledge base and any fact f derived from e, $f \in K$ implies $\gamma(p_f, K) = \text{promote}$ for the proposal p_f carrying e's provenance. There's no admission path for stored text that bypasses the gate, so authority over f is determined by policy over e's Suggestor and provenance, never by e's content.

The proposition is a restatement of the substrate's promotion invariant, which is exactly the point: memory needed no new security mechanism, because the one that governs every other proposal already governs this one.

5. Agentic memory

Beyond the knowledge base, Mnemos carries a set of memory modules that record different things about a Formation's history: *causal* memory, linking outcomes to what preceded them; *temporal* memory, ordering what was known when; *reflexion*, retaining what a previous attempt concluded about its own failure; *skill* memory, retaining procedures that worked; *session* memory, holding a working set; and *online* and *meta-learning* paths that adjust retrieval and weighting from feedback.

I am deliberately brief here, because this is the part of Mnemos with the weakest evaluation. The modules exist, they are exercised, and I can't yet say which of them earns its complexity. The honest summary is that causal and temporal memory have paid for themselves in explaining why a Formation reached a conclusion, session memory is plainly useful, and I don't have evidence about the others.

5.1. Feedback must not become authority

Feedback collection, replay and batch learning close a loop: what was recalled, what was used, what turned out to be right. Those signals adjust retrieval priors — which stores to consult, how to weight recency, when to widen a query.

They adjust nothing else. The invariant established for the planning layer (Pernyér, 2025b) applies here unchanged and for the same reason: learning flows backward into recall, authority flows forward from policy, and the two channels never meet. A memory whose learned state could influence what is permitted would make the substrate's fixed point a function of training history, and the determinism the whole stack rests on would quietly become conditional on what the system happened to have read.

6. The tension I haven't resolved

The substrate is append-only: context only grows, which is what makes runs replayable and provenance total. Memory is the component for which that's least tenable. An organization's knowledge base accumulates superseded prices, departed employees, cancelled contracts and documents it's legally obliged to delete, and a store that can only grow isn't a memory but an archive with a search box.

Three mechanisms are in tension and I can state them precisely. *Monotonicity* requires that facts are never removed. *Correctness* requires that superseded knowledge stops being recalled as current. *Retention law* requires that some records are destroyed on request, entirely.

The current implementation resolves this outside the proof: retention and deletion are product concerns per Table 1, superseding is modelled as a new fact rather than an edit, and recall filters by validity interval rather than by deletion. This works and it isn't a theory. In particular, a deletion performed for legal reasons removes an object the substrate's proofs quantify over, and I have no formal account of what a replay means afterwards.

> **Open problem (Forgetting in a monotone substrate).** Give a semantics for deletion in an append-only context under which (i) replay of a run that read the deleted entry is well-defined, (ii) the ledger remains a complete record of what was decided, and (iii) the deleted content is genuinely unrecoverable. State whether the three are simultaneously satisfiable.

My present belief, which I'd like to be argued out of, is that (i) and (iii) are in genuine conflict and that the best available answer is a *tombstone with an attested hash*: replay can verify that something was read and what it wasn't, without being able to reconstruct what it was.

7. Limitations

Retrieval quality isn't evaluated in this report. I give a fusion rule and an invariance argument for it; I don't give recall or precision numbers on a standard collection, and the reader should treat Section 3 as an argument about *which combination rule to use* rather than evidence about how well the system retrieves.

The default embedding path is a hash embedding, with a hosted model available. Hash embeddings are cheap, local and considerably worse; conclusions drawn on one don't transfer to the other, and the fusion argument is the only claim here that's indifferent to which is in use.

The constant $k = 60$ in Eq. 1 is inherited from the literature (Cormack et al., 2009) and hasn't been tuned for organizational corpora, which are smaller and far more acronym-dense than the collections it was measured on.

Finally, the injection containment argument bounds *authority*, not *influence*. A malicious passage that reaches a language model's input can still distort what that model proposes. What it can't do is become a fact without an attributed proposal crossing a gate. I regard that as the containment worth having and not as a solution to the problem.

8. Conclusion

Memory in an agentic system is usually treated as a retrieval problem with an accountability afterthought. Inverting that — making recall a proposal, with provenance, subject to the same gate as everything else — costs very little and buys attribution, refusability and a containment property for injected content that no amount of prompt hardening provides.

The retrieval result is smaller and more portable: when two rankers disagree on scales that admit no conversion, fuse the ranks. It's invariant to every monotone rescaling of either input, which means it survives changing your embedding model on a Tuesday.

What remains open is the thing an archive never has to face and a memory always does. The substrate says nothing is ever removed. The law, and the truth, both say some things must be. I have an implementation that's defensible and a theory that's missing.

> **Code and correspondence.** The work described here is implemented, not sketched. The source behind every claim in this paper — the engine, the extension, the fixtures and the tests that hold the invariants — can be shared on request, including the parts that did not work. Write to kenneth.pernyer@reflective.se. We are equally interested in being told where the argument is wrong.

References

Cormack, G.V., Clarke, C.L.A., Buettcher, S., 2009. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval 758–759.

Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., Fritz, M., 2023. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec).

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.- tau, Rocktäschel, T., Riedel, S., Kiela, D., 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems (NeurIPS).

Pernyér, K., 2025a. Proposal, gate, commitment: deterministic convergence as a substrate for organizational decisions. Reflective Labs technical report.

Pernyér, K., 2025b. Institutionalized disagreement: how a plan earns the right to be run. Reflective Labs technical report.

Robertson, S., Zaragoza, H., 2009. The probabilistic relevance framework: BM25 and beyond. Foundations and Trends in Information Retrieval 3, 333–389.

A. Retrieval, operationally

M1 Store `KnowledgeStoreSuggestor` proposes an entry for ingestion. The entry carries source, ingestion identity and time. Storing is itself a proposal and may be refused.

M2 Retrieve `KnowledgeRetrievalSuggestor` runs the vector and lexical retrievers, fuses ranks per Eq. 1, and proposes the top entries with their provenance attached.

M3 Hybrid weight `SearchOptions::hybrid(w)` selects the mix; the fusion itself is rank-based regardless of w, which controls how many candidates each retriever contributes rather than how their scores are combined.

M4 Provenance Every proposal carries the Mnemos provenance marker, and every Suggestor-boundary crossing emits a trace span. A proposal without provenance is malformed and fails the schema check at the gate.

M5 Feedback Usage and outcome signals update retrieval priors only. No feedback path writes to policy or to any input of the authority computation.

B. Two short proofs

B.1. Scale invariance of rank fusion

Let retriever i produce scores $s_i(d)$ and let φ be strictly increasing. Ranks are determined by the order of the scores, and a strictly increasing map preserves order: $s_i(d) > s_i(d')$ if and only if $\varphi(s_i(d)) > \varphi(s_i(d'))$. Hence $r_i(d)$ is unchanged for every d, and Eq. 1 depends on the scores only through the ranks.

For the converse, take two documents with $s_1 = (10, 9)$ and $s_2 = (0, 8)$ and weights $\alpha = \beta = 1$. The weighted sums are 10 and 17, so the second document wins. Rescale s_1 by $\varphi(x) = 10x$, which is strictly increasing and leaves both rankings untouched: the sums become 100 and 98, and the first document wins. The order induced by score fusion therefore depends on a choice that carries no information.

B.2. No authority from ingestion

Let e be an entry and f a fact whose content derives from e. By the substrate's promotion invariant, $f \in K$ only if some proposal p_f with $f_{p_f} = f$ satisfied $\gamma(p_f, K) = \text{promote}$ (Pernyér, 2025a). Mnemos constructs no other path into the context: its only write-side interface is the proposal, and the store isn't itself part of the context. The gate evaluates γ on the proposal's declared Suggestor, version and provenance. Therefore the admissibility of f is a function of policy applied to e's provenance and its proposing Suggestor, and is independent of e's content except insofar as the schema check constrains its shape.