RAMR — Retrieval-Augmented Memory Reliability

A contamination-resistant benchmark for agent memory that audits itself. Findings as of v0.6.0.

Rastislav Drahoš (Agora)
7 September 2026

Concept DOI 10.5281/zenodo.20818291 · **Source** github.com/DanceNitra/ramr · **Site** dancenitra.github.io/ramr · MIT

Abstract

Retrieval-backed memory is the load-bearing component of most agent stacks, and most evaluations of it can be gamed by rules that never read the content. RAMR is a small, reproducible, synthetic benchmark that isolates specific failure modes of such memory — a missing hop, a distractor, a compacted summary, a correction that gets echoed back — together with the measurements it produced and the tooling that keeps those measurements honest. Entities are random tokens, so closed-book accuracy is 0.000 by construction. Every cited number is recomputed from a persisted result file by `verify_numbers.py`; a number without a row in the ledger is not citable. The benchmark ships the auditor that measures how much of it can be solved without understanding it, and the first number that auditor produced was about RAMR itself: 97.2%.

1. What RAMR is, and is not

RAMR is a *findings + method* release. It is not a large-scale multi-system leaderboard, and it leads with its limitations on purpose (§7). It claims three things: a reproducible, contamination-resistant method; a robust cross-model chain-fragility result; and a set of honestly caveated findings about where retrieval-backed memory fails.

Design principles:

1. **Contamination-resistant.** Entities are random synthetic tokens. Closed-book accuracy is verified to be ~0 on every run.
2. **Reproducible.** The dataset is frozen to disk with a sha256-pinned manifest; a single runner loads it and never regenerates. Embeddings are cached.
3. **Falsifiable.** Every metric ships with a pre-registered falsifier and bootstrap CIs. Honest negatives and corrections are recorded, not hidden.
4. **Independent baselines.** Claims about RAMR's own reference core (inspeximus) are checked against standard, independent libraries on identical inputs.

2. The eleven metrics

Metric	Question	How
CONVERSION	Does complete retrieval convert to a correct multi-hop answer?	gold-chain accuracy
CHAIN-FRAGILITY	How much does one missing hop cost?	gold − partial (one hop dropped)

DISTRACTION	How much do irrelevant or look-alike facts cost?	gold − noisy
FACT-RETENTION	Does a compiled memory tier drop facts under a fixed budget?	raw − compiled at a hard char budget
OUTCOME-RANKED-RECALL	Does ranking by *was-it-right* beat *was-it-recalled*?	outcome-credit vs relevance-only, vs an independent retriever
FORGET-PRECISION	After an update, does recall return the current value?	fraction current after a supersession pass
ECHO-RESISTANCE	After a correction, does re-stating the old value resurrect it?	fraction whose top-1 stays current after a value-preserving restatement
COMPRESSION-vs-RAW	Does a compiled summary beat raw context, or only lose to it?	acc(compiled) − acc(raw) over distractor load
OPERATIONAL-CONTINUITY	On resume after compaction, does the agent re-run a completed action?	duplicate-rate of a budget-limited resume recall
TEMPORAL-AS-OF	Does supersession resolve by validity time, not ingest order?	reversed-ingest now-accuracy + recall(as_of=T)
INTEGRITY-CONDITIONED RECALL	After a supersession, revert or poison, is the correct current value returned?	acc@1 for naive-cosine, cosine-recency and inspeximus (± warrant gate)

3. Headline findings

All numbers are traceable to a persisted result JSON and recomputed by `verify_numbers.py`.

3.1 One missing hop collapses a multi-hop answer. Dropping a single required hop collapses 3-hop accuracy to near zero for every model tested: seven models across six families (Qwen, Meta, Google, Zhipu, Moonshot, Anthropic), CHAIN-FRAGILITY +0.90 to +1.00. The two anchor models ran at n=200 with paired-bootstrap CIs: qwen3-coder:30b and glm-5.2 both **+1.000, CI [+1.000, +1.000]**. Complete-chain accuracy is 1.000; closed-book accuracy is 0.000 (`ramr_scale_cf_result.json`, `ramr_v0_result.json`).

3.2 Ranking recall by *was-it-right* beats *was-it-recalled*. On a near-duplicate case that relevance cannot solve, outcome-credit reranking beats relevance-only at every ambiguity level: lift **+0.358 / +0.361 / +0.469 / +0.427 at D=1/2/4/8**, n=12 sets, every bootstrap CI excludes zero (minimum lower bound +0.299). A random-credit control is negative (−0.30 → −0.07), so the gain is the outcome signal, not reranking noise. An independent scikit-learn `NearestNeighbors(cosine)` retriever scores identically to the plain baseline (gap 0.000 at every D), so the baseline is not a strawman. This shows the value of an outcome channel; it is not a head-to-head win over shipped products, which were denied the label by design (`outcome_scale_result.json`).

3.3 A correction that sticks can still be undone by re-stating the old value. FORGET-PRECISION shows the correction holds (1.00 for explicit contradiction and for a silent numeric update, both 0.00 without supersession; n=30 topics, 6 seeds). But when the retired

value is re-asserted afterwards — a benign restatement or an attacker re-injecting it — a last-writer-wins store treats the echo as the newest assertion and resurrects the stale value: **echo-resistance 0.00**, verbatim and reworded. An object-keyed superseded-value ledger refuses to revive an already-retired value on a mere restatement: **1.00**, with FORGET-PRECISION unchanged (`echo_resistance_result.json`). Honest scope: value-*preserving* restatements; a value-*obscuring* echo ("go back to the old one") carries no value to key on and is out of scope for any object-level defence.

3.4 Other measured results. DISTRACTION@60 is model-specific, +0.15 (llama3.1:8b) to +0.60 (kimi-k2.6), n=20; lexical near-miss distractors did not bite beyond raw volume (an honest negative). FACT-RETENTION under a 400-char budget at M=48 loses +0.70 (qwen3-coder:30b) and +0.76 (gemma2:9b), n=5. COMPRESSION-vs-RAW: a compiled summary does not beat raw context at any noise level for a capable reader (+0.00 / −0.40 / −0.55 at K=5/20/50). OPERATIONAL-CONTINUITY: with recency, duplicate-rate tracks the recall-budget floor exactly; without it, 1.00 at every budget. TEMPORAL-AS-OF: validity-time supersession serves the current value (1.00) where ingest-order serves the stale one (0.00). INTEGRITY-CONDITIONED RECALL after a revert: inspeximus 1.00, cosine-recency 0.00, naive 0.55, n=100.

4. The benchmark audits itself: the shortcut floor

How much of this benchmark can be solved without understanding it? `memaudit.py` runs a partial-input baseline battery — position, casing, length, token recurrence, query overlap, stray fields, id shape — and reports the score reachable by rules that read only surface form. Every probe is also scored against permuted labels (the label moved to another candidate in the same trace, leaving set size, text and order intact), which is its measured chance level; `REAL` is decided by lift over that null, never by coverage. `test_memaudit.py` asserts both directions in CI: 0 of 8 probes fire on synthetic clean data, and a planted position cue reads 100.0% against a 0.0% null. Without that control, a shortcut battery is a machine for manufacturing alarming numbers about other people's work.

trace version	shortcut floor	coverage
v0.2	**97.2%**	96.0%
v0.3 (structural re-cut)	96.8%	52.7%
v0.5 (connectivity balanced by construction)	**40.0%**	1.7%
LoCoMo, for comparison	36.3%	44.3%

The LoCoMo row is a floor for RAMR's framing (every dialogue turn a candidate, single-evidence questions), not a verdict on that benchmark; LoCoMo shows no positional artifact and no stray label field, clean on both axes where RAMR leaked. No dataset is redistributed.

It caught its author twice. A re-cut that "fixed" the length cue had merely inverted it (73% → 83% the other way, which one-directional scoring reads as *at chance*). Balancing the echo cue injected a record whose distinctive id let an id-shape rule solve 136 to 148 of 300 traces at 95.2% to 100.0%, measured over 100 rebuilds, while every probe reported *at chance* because none looked at ids. Binary probes are now scored in both directions, and `leak:id-outlier` exists because of the

second incident. Both are in `ERRATA.md`. A floor is a lower bound, never a verdict: “failures of partial-input baselines do not mean the dataset is free of artifacts” (Feng, Wallace & Boyd-Graber, ACL 2019).

5. Preflight: was the comparison admissible?

A metric only means something if the arms were comparing the same thing. `ramr_preflight.py` runs four gates before an answerer costs anything, and keeps them separate so an abort names the dead layer:

- **G0 budget parity** (experiment design). Motivating failure: a memory arm at k=20 (1,323 chars) against session-level BM25 (11,941 chars), a 9.03x gap. BM25 appeared to win. Matched (1.00x), accuracy went 0.283 → 0.593 and the ranking flipped.
- **G1 retrieval** (evidence in context). The gold evidence was absent on 96.5% of probes in that run. Per probe a gate; aggregated, the recall ceiling reported beside accuracy, never folded into it.
- **G2 liveness** (store / answerer). A baseline scored 0.000 twice from a truncation bug with clean logs. A missing positive control is a failure, not a skip.
- **G3 parameter efficacy** (harness / API binding). `limit=` passed to an API wanting `top_k=`, silently swallowed. Asserting the knob changed behaviour is the general defence.

`--demo` reproduces a refusal on the frozen dataset (G0 = 20.67x FAIL / 1.00x PASS, evidence ceiling 1.000 on 300 chains) into `preflight_result.json`; the demo itself caught a routing bug when the ceiling read 0.31 instead of 1.0. `--selftest` asserts every gate can still fail: a gate that cannot fail is a demonstration, not a test. The 9.03x / 96.5% / 0.283 → 0.593 figures are from an external run and are not reproducible from this repository; the `--demo` numbers are. The separate-gates design and G3 came from u/jacksonxly in a public thread.

6. Cross-system integrity against real stores

The synthetic metrics are the method; the `integrity/` module is the complementary cut against the memory libraries developers actually run, on their native configurations, through one shared ground-truth-blind judge. It measures value-obscuring REVERT (“go back to what we had”, naming no value), ECHO resurrection, a run-your-own erasure self-check, and a deterministic bi-temporal cell. Results roll into a standing, PR-submittable Agent-Memory Integrity Leaderboard.

Answer-level echo-resistance (recall top-k → judge LLM → is the current value returned; fair to add-based stores that keep both values), n=30:

backend	forget-precision	echo-resistance
inspeximus, echo guard off	1.00	0.00
mem0 2.0.11 (add-based, gpt-4o-mini + text-embedding-3-small + Chroma)	0.87	0.53 (95% CI on resurrection [0.30, 0.63])
Zep/Graphiti (Neo4j + gpt-4o-mini, real runtime)	0.87	0.87 raw; echo-attributable 1.00
inspeximus, echo guard on	1.00	1.00

Graphiti's 0.87 is not an echo failure: in the 26 of 30 cases where the correction registered, the echo flipped 0. The residual 13% is an upstream extraction miss ("Target entity not found") which the echo neither causes nor exploits. A real bi-temporal store and an object-keyed ledger both defend structurally; mem0's resurrection is echo-driven (14/30 flips against 4/30 pre-echo misses). mem0 and Graphiti have no revert operation, a capability gap rather than a tuning gap. On the bi-temporal cell inspeximus scores 4/4 and *matches* the graph-memory leaders' documented design; it does not beat them on that axis and RAMR does not claim it does.

7. Limitations, and what is not claimed

- **Synthetic, not real-world.** Items are random tokens. RAMR does not measure real-document retrieval or real-conversation memory yet.
- **Scale is uneven.** CHAIN-FRAGILITY n=200 with tight CIs; OUTCOME-RANKED n=12 sets; FACT-RETENTION n=5. Small-n magnitudes are directional; the orderings are the signal.
- **One embedder** for OUTCOME-RANKED (local nomic-embed-text), validated against an independent scikit-learn retriever but not against shipped products.
- **Substring answer matching**, exact here because answers are unique tokens; noisy on free-form text.
- **Single covariate construction per metric.** Relative effects are claimed where the CI says so; unchanged transfer to other task shapes is not.
- **The traces leak, and by how much is published** (§4).

RAMR does not claim to be the definitive agent-memory benchmark, that these magnitudes transfer to real corpora, or that shipped products underperform it where they were not run.

8. Reproduce

```
git clone https://github.com/DanceNitra/ramr && cd ramr
python build_dataset.py                       # freeze the sha256-
pinned dataset
python ramr_run.py --model qwen3-coder:30b --n 200 --dist 30
python ramr_outcome_ranked.py && python ramr_external_baseline.py
&& python ramr_factret.py
python memaudit.py --adapter demo             # the shortcut
floor, on demo data
python ramr_preflight.py --demo               # the four gates, on
the frozen dataset
python verify_numbers.py                      # every cited
number, recomputed
```

Models are reached via an OpenAI-compatible endpoint (local Ollama by default). Reference memory core: `pip install inspeximus`.

Cite

Drahoš, R. (2026). *RAMR — Retrieval-Augmented Memory Reliability* (Version 0.6.0) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.20818291