# Deterministic Recall Without Embeddings

kgai maintainers

kgai.dev · team@kgai.dev · August 2026

**ABSTRACT**

Retrieval layers for AI coding agents are dominated by embedding-based semantic search, which buys tolerance to paraphrase at the price of nondeterminism, model dependence, and operational weight. We describe the read path of kgai, a local-first memory of engineering decisions for AI-assisted teams, which contains no embeddings and no model by design. Recall combines lexical matching over element names and decision texts with traversal of a small graph of domain elements and the immutable decisions that shaped them. Because the store replays deterministically from an append-only log, the same store and the same query yield the same context on every machine, a property we argue is the correct default when an autonomous agent assembles the context it will act on. Context is bounded to the head set, the decisions currently in force for the elements in play. We treat honestly what lexical recall loses relative to semantic search, how agent-driven rephrasing compensates, and where it does not, and we discuss a contextual-search index on the roadmap without making commitments about it.

## 1. Introduction

Before an AI coding agent touches an area of a system, it assembles context: what exists there, what was decided about it, and why. The retrieval machinery behind that step is, in most current designs, borrowed from open-domain question answering. Text is embedded into vectors, queries are embedded the same way, and nearest neighbors are returned. Dense retrieval of this kind is effective and well studied [1], and the retrieval-augmented generation pattern built on it is now standard [2].

We took a different position for kgai, a decision record kept local-first for development teams. Its read path contains no embeddings and no model call of any kind. Recall is lexical matching plus graph traversal over a small, curated store of engineering decisions. This report describes that read path, argues why determinism is the property worth optimizing when the reader is an autonomous agent, bounds what the approach loses relative to semantic search, and describes how the surrounding system compensates. We also state what is on the roadmap and deliberately promise nothing about it.

## 2. The Store Being Read

The store is an append-only, content-addressed log of decisions, each an immutable event that mutates a small graph of domain elements, meaning the features, services, and business objects that make up the system under development. This is the standard event sourcing arrangement [3]. Replay is deterministic. Sorted by Lamport clock value and, on ties, by content hash [4], the log projects into a read model of two planes, a live element graph for the current shape and a decision plane keeping every decision beside the elements it shaped.

Two design choices matter for retrieval. First, element identity is a deterministic hash of the element's normalized kind and name, so independent writers who record the same concept converge on the same node without coordination. Second, the projection is idempotent and totally ordered, so two stores that have replayed the same events hold identical graphs, a property each pair of stores can check by comparing the digests of their canonical exports. The corpus under retrieval is therefore not an arbitrary document pile but a small, stable graph with an exact, reproducible state. The replay machinery behind that guarantee is the subject of the projection report in this series (TR-2026-03).

## 3. The Read Path

The system exposes a small set of read operations. Recall returns the decisions in force for the elements relevant to a task, selected by element name, topic, or recorded path patterns. Free-text search matches the words of a query against element names and decision texts by lexical overlap. History returns the ordered decision chain for one element. Point-in-time replay reconstructs the whole graph as of a past date, and a conflicts query lists elements with competing current decisions.

Mechanically, every one of these operations is lexical matching, graph traversal, or replay. A query term either matches recorded text or it does not, and from a matched element the engine walks links to related elements and into the decision plane for provenance. Lexical retrieval is among the most mature technologies in information retrieval [5, 6], and the variant here is modest by those standards, because the corpus is a curated record rather than the open web, one in which names have been normalized at write time and text is dense with domain terms.

What the read path does not contain is any learned component. There is no embedding index to build or refresh, no vector store to operate, no similarity threshold to tune, and no model whose version changes what a query returns. This is a design decision, not a missing feature, and the remainder of this report is an argument about its consequences.

## 4. Why Determinism Matters for Agent Context

The property the design buys is easy to state. Same store, same query, same context. Any machine that has replayed the same log answers any read identically, byte for byte. We argue this is the correct default for agent memory on three grounds.

Reproducibility. An agent's action is conditioned on the context it retrieved. When retrieval is a pure function of the log and the query, a surprising action can be investigated by replaying the exact context that produced it. When retrieval depends on an embedding model, an index build, and an approximate nearest neighbor search, the retrieved set is difficult to reproduce after any component changes, and the investigation loses its footing.

Team symmetry. A shared decision memory exists so that every teammate's agent reasons from the same record. Under deterministic recall, two agents that answer differently must hold different logs, which synchronization state can explain and canonical digests can confirm. Retrieval variance is eliminated as a hypothesis, and that elimination has diagnostic value in itself.

Stability over time. Embedding-based systems inherit the lifecycle of their models. Reindexing with a new model reorders neighbors, and yesterday's recall is not today's. A lexical and structural read path has no such dependency. The answer changes only when someone records a decision, which is exactly when it should change.

## 5. Bounded Head-Set Context

Recall in kgai does not return the top k passages of an unbounded corpus. It returns the head set, the decisions currently in force for the elements in play, with superseded decisions filtered out. Because the element graph is small and stable by design, and because each element carries few in-force decisions at any moment, the context handed to the agent is bounded and current rather than large and ranked.

This shape serves the consumer as well as the producer. Long-context studies show that models use extended contexts unevenly and privilege the edges of the window [7], so a compact set of decisions that are all in force, all relevant to the elements at hand, and all carrying their rationale is a better payload than a long ranked list with relevance decaying down the page. Boundedness also keeps the cost model trivial. Nothing is embedded at write time, so capturing a decision adds no model cost, and recall does not grow more expensive as the team's history deepens, since history is excluded from the default view and available on demand.

For scale orientation only, the largest archived run of the project's benchmark holds one million decisions across thirty writers' shards and answers decision lookups in roughly 100 milliseconds, with recall and free-text search slower than decision lookups. Measurement methodology and full latency tables are out of scope here and belong to the scale report in this series (TR-2026-09).

## 6. What Is Lost, and How Rephrasing Compensates

The costs of the design should be stated as plainly as its benefits. Dense retrieval outperforms lexical baselines precisely where queries and documents share meaning but not vocabulary [1]. A word-overlap

match cannot find a decision phrased entirely in different terms, and no amount of graph structure changes that when the entry point itself is missed. Systems whose corpus is arbitrary prose, conversation transcripts, or documentation at scale are right to reach for embeddings, and we do not claim otherwise.

Two properties of this setting narrow the gap. The first is structural. Decisions attach to elements, and element names are normalized at write time, so the query needs to reach the element, not the decision text. Once an element matches, traversal returns its decisions even when their texts share no words with the query. The vocabulary problem is thereby reduced from the whole corpus to the element names, a far smaller surface.

The second is behavioral. The caller is a language model in a loop rather than a person typing one query, and retrieval-augmented agents reformulate and retry as a matter of course [2]. An agent that misses with one phrasing rephrases with synonyms, with the element vocabulary it has already seen in the session, or with a path pattern, and converges on the entry point. In our usage this compensation is routine and effective, and it is honest to add that it is not free. Rephrasing spends agent turns, and it fails when the agent lacks the domain vocabulary entirely, the cold-start case in which a semantic index would have bridged the gap and lexical recall cannot.

## 7. Roadmap: a Contextual-Search Index

The project roadmap lists a contextual-search index for stores beyond roughly one hundred thousand decisions, motivated by the observed slowness of free-text search relative to decision lookups at large store sizes. We record its existence on the roadmap and make no commitment in this report about its design, its semantics, or its delivery. The design question it would have to answer is whether recall can be improved without surrendering the property this report is about, that the same store and the same query produce the same context on every machine.

## 8. Limitations

Beyond the vocabulary mismatch treated in Section 6, three limitations deserve statement. First, determinism is a property of a replayed log, not of a team at an instant. Two stores mid-synchronization hold different logs and answer differently until they converge, so recall is deterministic per store, and consistent across a team only as synchronization catches up. Second, free-text search is the weakest read path at scale, as noted above, and teams with very large stores feel this before any other limit. Third, the approach presumes a curated store of structured decisions. It does not extend to arbitrary prose corpora, and a team wanting fuzzy recall over meeting notes or documentation should run a semantic tool beside kgai rather than expect kgai to become one.

## 9. Conclusion

We have described a read path with no embeddings and no model in it, by design. Lexical matching finds entry points in a small normalized graph, traversal supplies related elements and provenance, and in-force filtering bounds the result to the head set an agent actually needs. The determinism this yields, same store, same query, same context, is the property we consider most valuable when the reader is an autonomous agent acting on what it retrieves, because it makes agent behavior reproducible, makes team memory symmetric, and removes a whole class of silent drift. The price is paraphrase blindness at the entry point, which structure narrows and agent rephrasing routinely, though not universally, overcomes. Within its intended domain, a curated decision record read by agents, the exchange is one we would make again, and we prefer stating its edges to blurring them.

## References


[1] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 6769-6781.

[2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W. Yih, T. Rocktaschel, S. Riedel, and D. Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems 33 (NeurIPS 2020).

[3] M. Fowler. 2005. Event Sourcing. martinfowler.com. https://martinfowler.com/eaaDev/EventSourcing.html

[4] L. Lamport. 1978. Time, Clocks, and the Ordering of Events in a Distributed System. Communications of the ACM 21(7), 558-565.

[5] S. Robertson and H. Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval 3(4), 333-389.

[6] J. Zobel and A. Moffat. 2006. Inverted Files for Text Search Engines. ACM Computing Surveys 38(2).

[7] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. Transactions of the Association for Computational Linguistics 12, 157-173.