Detecting the Latest Document Version Counters Context Decay and Raises RAG Accuracy by 46%

Context decay is the quiet failure mode of enterprise RAG. The model's context window fills with outdated versions of the very documents that should answer the question, and answer quality degrades even when the right document was retrieved. In a controlled benchmark on a real enterprise corpus, detecting the latest version of every document and keeping the superseded ones out of the index raised answer accuracy from 27.2% to 39.7% (a 46% relative gain), eliminated answers based on outdated facts, and made every committed answer correct. Nothing in the retrieval or generation stack changed. Only the data did.

By the Deasy Labs team at Collibra

Context decay starts in the corpus

Anthropic's work on *context engineering* describes the phenomenon, which they call context rot. As more tokens fill the context window, the model's ability to use the information in it degrades. Their conclusion is that a model does its best work on the smallest set of high-signal tokens. Most of the effort to get there happens at inference time, curating what goes into the window. This study is about the half that happens earlier, because what is in the corpus determines what can end up in the window at all.

Repositories retain history. A product ships a security patch and a new bulletin is published, but the old bulletin stays in the same folder. Over time the repository holds multiple versions of the same document, and only one is current. The others carry outdated facts such as old severity scores, superseded defect IDs, and wrong fix-pack numbers.

RAG inherits this in a form that feeds context decay directly. The failure is not that the system retrieves the outdated version instead of the current one. It retrieves both. Near-identical documents sit at nearly the same point in embedding space, so the retrieval model has no basis to prefer one over the other. The context window then carries the same content twice, low-signal by definition, and where the versions disagree, a direct contradiction. Even when the current document is retrieved, the outdated copy next to it makes the answer worse.

The only way out is to know, for every document, whether it is the latest version of itself, and to keep the superseded ones out of the index. That knowledge cannot be derived from vector distance. It is metadata, and it has to be established in the data layer, before the index is built. We ran a controlled benchmark to measure what that is worth.

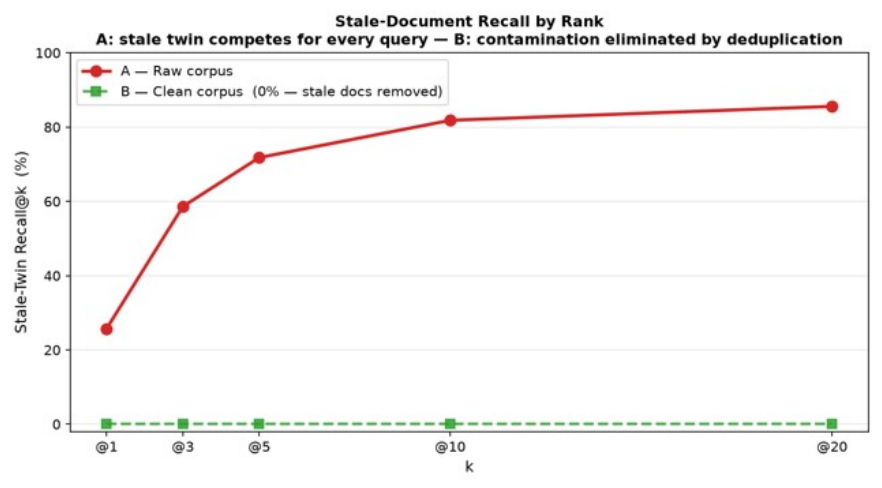
The experiment

We took 905 IBM technical support documents from the public RAGBench-TechQA dataset and added 300 outdated near-duplicates, each a copy of a real document with a few facts deliberately backdated (a defect ID replaced with an older one, a fix-pack number off by one, an earlier CVSS score). Because we controlled the seeding, we have exact ground truth for 804 fact conflicts across 319 questions.

We ran the identical pipeline twice, once on the raw 1,205-document corpus, and once after Deasy's latest-version detection determined the current version of every document and excluded the superseded copies. The cleaned corpus was the pipeline's actual output, not a hand-curated set. Embedding model, retrieval settings, prompts, and scoring were held constant, so the only variable is the data.

The outdated version outranks the current one on 45.5% of queries

On the raw corpus, the superseded version ranked above the current document in 45.5% of queries. In one out of four, it was the first thing the model read.

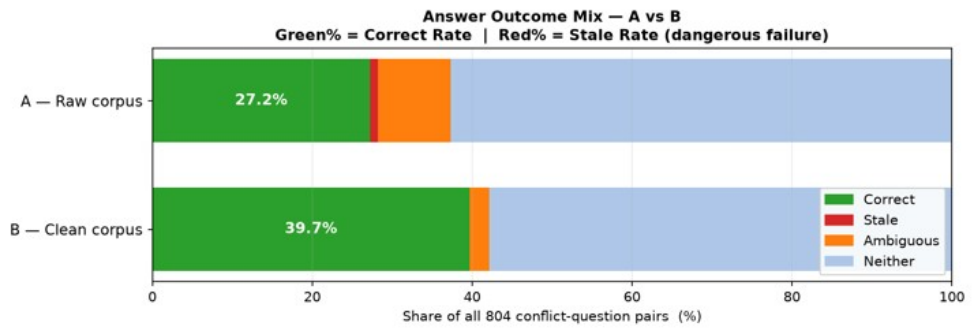


How often the outdated version appears in the top K retrieved results. On the raw corpus it is at rank 1 in 25.7% of queries, in the top 3 in 58.6%, and somewhere in the top 20 in 85.6%. After cleanup it is never retrieved at any K, because it never enters the index.

Retrieval tuning cannot fix this. The two versions' ranks are interleaved, so cutting retrieval depth drops the current document as often as the outdated one. No ranking heuristic can separate near-identical embeddings without knowing which document is the latest version.

Results

	Raw corpus	Latest-version only
Correct answer rate	27.2%	**39.7%**
Answers asserting an outdated fact	2.0%	**0.0%**
Contradictory answers (both values given)	9.1%	2.5%
Reliability when the system commits	93.2%	**100.0%**
Current document ranked first	22.9%	48.9%
Outdated doc outranks current	45.5%	**0.0%**

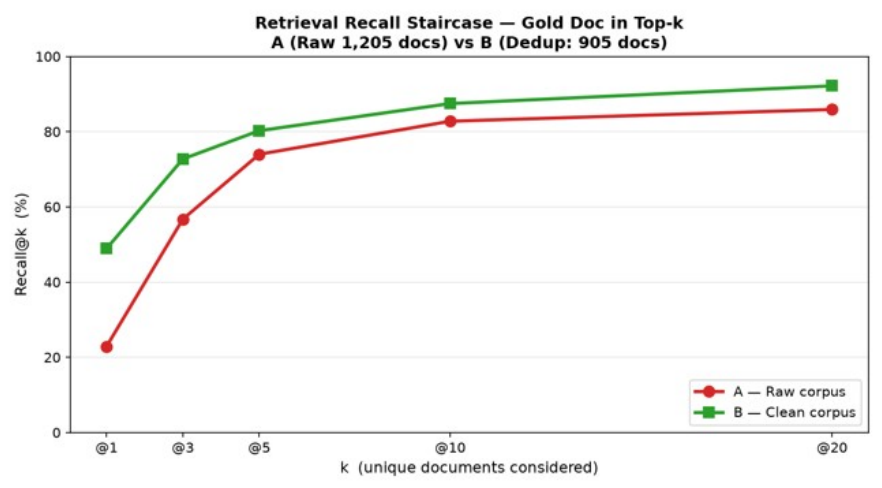


The full outcome breakdown for all 804 fact conflicts. Green is a correct assertion of the current fact, red is an assertion of the outdated fact, orange is a contradictory answer, grey is an abstention. Cleanup converts the red and orange segments almost entirely into green.

The cleanup fixed 83 answers and regressed 12. The confidence intervals of the two conditions do not overlap (McNemar p effectively zero).

Context decay, measured

Ranking. With the outdated competitor gone, the current document reached rank 1 in 48.9% of queries, up from 22.9%. The gap is widest at low retrieval depths, where it matters most.



Recall of the current document by retrieval depth K. The gap is widest at K=1 (+26 points) and narrows at higher K, because the damage from outdated versions is concentrated in the ranking, not in whether the document is found at all.

Focus. This is context decay measured directly. On queries where the correct fact was already in the retrieved context, so retrieval was not the constraint, accuracy was 36.2% on the raw corpus and 50.9% on the cleaned one. Same fact, same availability, 15 points difference. What changed is what else was in the window. With two documents asserting different values, the model hedges or declines to commit, and contradictory answers fell from 9.1% to 2.5% after cleanup. Removing the decayed content from the window is worth as much as retrieving the right document in the first place.

Reliability. On the raw corpus, the system was correct 93.2% of the time when it committed to a specific fact. Inverted, that is a 7% confident-error rate on exactly the questions where users need precision. On the cleaned corpus it committed more often and was never wrong. That is the difference between a system people trust and one they re-verify.

How latest-version detection works

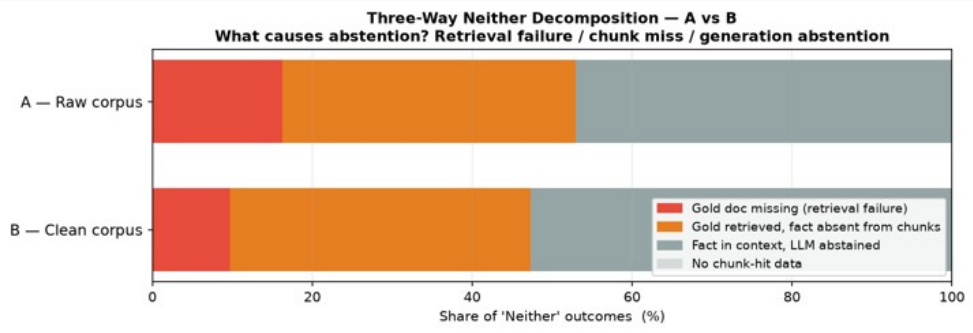
The capability is knowing, for every document in the repository, whether it is the current version of itself. That takes two steps.

First, documents that are versions of the same logical document have to be found and grouped. Exact copies are caught by content hash, versions with differing bytes by an identity fingerprint over extracted metadata (title, author, type, topic), and near-duplicates by content fingerprinting that scales to hundreds of thousands of files. Entity-overlap and embedding checks reject groups of documents that merely look alike.

Second, and this is the decisive step, the latest version within each group is determined from the strongest unambiguous signal, starting with the effective date the document states about itself, then source-system timestamps, then extracted version identifiers. Superseded copies are tagged and excluded from the AI-ready slice, not deleted, so every decision is auditable and reversible.

What is left

39.7% on a strict benchmark leaves an obvious question about the rest. We traced every abstention to its cause. On the cleaned corpus only 9.7% were retrieval failures, and in over half of the remaining cases the correct fact was in the model's context and the model still declined to assert it. The residual gap is a generation and chunking problem, not a data problem, and not a reason to buy a better retriever.



Decomposition of every abstention by cause. On the cleaned corpus, only 9.7% are retrieval failures. The dominant cause, 52.7%, is the generation model declining to assert a fact that was already in its context.

The scoring is deliberately strict, which caps the absolute numbers. "Apply the latest available fix pack" counts as an abstention even when it is sound advice, because it does not assert the exact value asked for. Scoring used a deterministic exact-fact matcher with an independent model for edge cases, separate from the answer model. Vague answers were never credited, and passing mentions of an old value (in a version list or an "and prior versions" clause) were never counted as failures.

Takeaways

The default response to disappointing RAG accuracy is pipeline tuning, meaning a better embedding model, a reranker, or a larger context window. This benchmark shows the ceiling on that. The embedding model retrieved the outdated version because it was, geometrically, an excellent match. The pipeline did its job. The corpus contained superseded documents, and no pipeline component can know which version of a document is current. A larger context window makes it worse, because it admits more of the decayed content.

Countering context decay is therefore a two-sided discipline. The model labs work the inference side, curating what enters the window. The data side is knowing the latest version of every document and indexing only that, and in this study that alone was worth a 46% relative accuracy gain, with the reliability of committed answers going to 100%.

We call the broader failure class RAG Contamination, meaning any case where a model's context contains content it should never have seen, whether outdated, sensitive, or ungoverned. This study covered the outdated variant. The next one measures the sensitive-data variant, where the same metadata approach filters PII-bearing documents from the retrieval pool.

> The benchmark used RAGBench-TechQA (IBM technical support documents), with 905 authoritative documents plus 300 synthetically backdated near-duplicates, and 319 questions targeting 804 seeded fact conflicts with exact ground truth. The stack was identical across conditions (text-embedding-3-small, 1,024-token chunks with 200-token overlap, top-20 retrieval). Answers were generated by GPT-4.1 and scored by a hybrid deterministic-plus-LLM judge with false-positive guards. A McNemar paired test on 645 rows gives chi-squared 51.6 and p effectively zero. Latest-version detection and exclusion were performed by Deasy's Unstructured Data Quality pipeline running on the full collection.