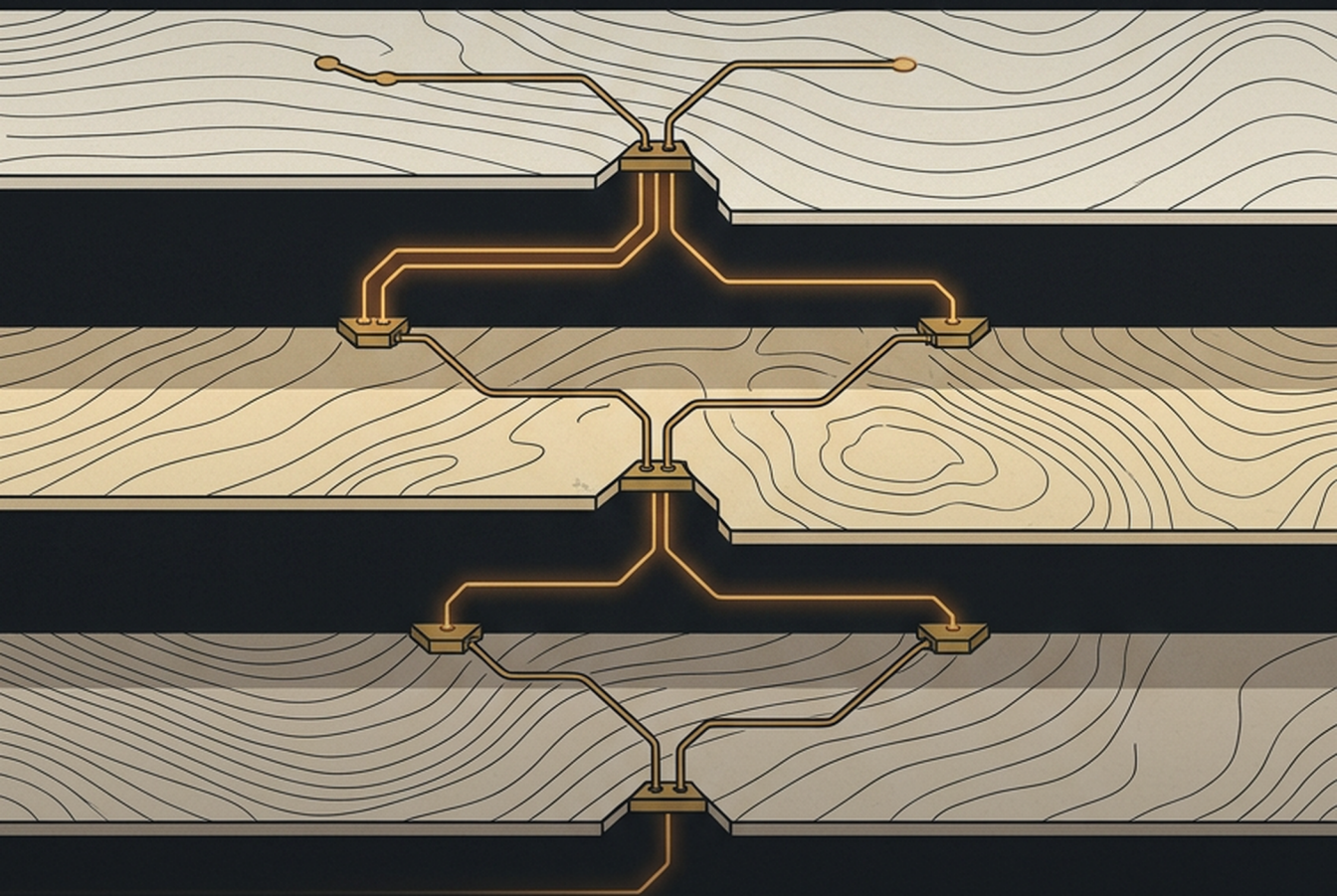


AUDITABLE LLM PIPELINES



A case study in tracing, fallbacks,
and honest evaluation

GROUND TRUTH BOOKS

Auditable LLM Pipelines

A case study in tracing, fallbacks, and honest evaluation

Ground Truth Books

This book is available at <https://leanpub.com/auditable-llm-pipelines>

This version was published on 2026-10-11



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2026 Ground Truth Books

Contents

About This Book	i
Introduction	1
Trusting multi-stage pipelines	1
Who this book is for	1
The bookgen case study	2
The route through the book	3
The auditable pipeline	5
Files as the Source of Truth	6
Why files beat state	6
Per-project artifact layout	6
JSON authoritative, corpus.db derived	6
Content-keyed caches	6
The chapters_done resume, read from files	6
Fork and select versions	6
Where determinism breaks	7
Typed Outputs and Human Gates	8
Pydantic models at every boundary	8
Validation before you trust	8
Stage gate reviews	8
The human gate checkpoint	8
What the gates caught	8
Failing fast at boundaries	8
Fallbacks and Model Provenance	10
Why fallbacks exist	10
pydantic-ai FallbackModel	10

CONTENTS

OpenRouter as the router	10
Free-tier degraded fallbacks	10
request.model versus response.model	10
The degraded free-tier run	10
Provenance through the stack	11
Tracing as Audit	12
OpenTelemetry GenAI semantics	12
Spans for every model call	12
Tool invocations on the trace	12
Phoenix as the audit pane	12
The first full trace	12
Tool errors hidden from the run	12
pinned_only on two of thirty	13
Retrieved but never cited	13
Evidence you can trust	14
Hybrid Retrieval and Citation IDs	15
Chunking the corpus	15
BM25 and vector branches	15
Reciprocal rank fusion	15
Pinned boost and diversity decay	15
Citation ids tie claims to sources	15
Context assembly and limits	15
Early retrieval failures	16
Measuring Retrieval Honestly	17
Building graded gold sets	17
Hash matching lost 518 judgements	18
Containment matching fix	19
NDCG and recall at K	19
The paired bootstrap test	20
The reranker null result	21
Pooling bias and lower bounds	22
Removing the reranker	23
Reading numbers without lying	25

Closing the loop 27

Critique and Revise Loops 28

 The critique stage 28

 Self-refine feedback loops 28

 Generated books as stress test 28

 Fabrications under valid citations 28

 The deterministic citation validator 28

 Resolving only known ids 28

 When revise makes it worse 29

 Replaying critiques for evaluation 29

 When a deterministic check is wrong 29

Context Rot and Digests 30

 Lost in the middle 30

 How context rots 30

 The measured rot trigger 30

 Chapter digests at 187 tokens 30

 Continuity, not mitigation 30

 Part intros from digests 30

 Stale digests after re-approval 31

 Measure before you mitigate 31

Cost Control and Smoke Runs 32

 Where the money goes 32

 Cheap end-to-end smoke runs 32

 Bounding spend per run 32

 Free tier as cost lever 32

 Real costs of bookgen 32

Stripping Machine Prose Habits 33

 The style lint pass 33

 Machine prose tells 33

 Blind voice evaluation 33

 AUC 0.42 and 12 of 20 33

 The habit tell list 33

 Linting without flattening voice 33

Conclusion 35

 What you can now do 35

The mechanisms that transfer	35
Where bookgen goes next	35
The audit reflex	35
Appendix A: bookgen CLI and stage command reference	36
Command reference	36
Stage pipeline	36
Configuration	36
Project layout	36
Measuring the harness	37
Tracing	37
The critique and revision loop	37
Publishing	37
Appendix B: GenAI semantic convention attribute table	38
Inference span attributes	38
Tool span attributes	38
Operation names	38
Metrics and events	38
Renamed or removed	38
What bookgen reads	38
Appendix C: Gold set and scoring schema reference	40
Files	40
Relevance scale	40
Building a gold	40
Covering rules	40
Score.py reference	40
Comparing systems	40
The two golds	41
Limits	41
Glossary	42
References	43

About This Book

Auditable LLM Pipelines is published by Ground Truth Books. Its case study is bookgen, a multi-stage book-generation harness; every run, cost and measurement it reports comes from that harness's own logs, traces and evaluation sets.

First edition, October 2026. Copyright © 2026 Ground Truth Books. All rights reserved.

Product names such as OpenRouter, Pydantic AI, Arize Phoenix and OpenTelemetry are used descriptively. This book is not affiliated with or endorsed by any of their owners.

Introduction

Trusting multi-stage pipelines

A multi-stage pipeline can fail and still look green. Say a chapter's research stage hits the provider's quota. The fallback chain hands the writing to a different model, the model produces text, and the critique scores it 85, accept. Tool exceptions are caught and fed back to the model, so a run "looks clean from the outside". In the first full book run of the harness this book dissects, the free tier's daily quota decided which model wrote the book, and chapters lost words and citations one after another. Chapter 4 tells that degraded first run in full.

That run was reproducible but unattributable. Calls went through a `Fall-backModel`, so "which model actually served this call" was answerable only by log archaeology, not from any record the run kept. The two properties pulled apart. You could replay each stage, because each stage reads and writes files and keeps no state in memory between runs. You could not say which model wrote which paragraph, which tool returned which source, or which number in the score was real.

A silent substitution at the third stage costs more than the model call: it poisons every stage after it. An invented metric protects the change that caused it. You ship the artifact, and the reader who questions the claim finds the run is gone.

Ground truth for an LLM pipeline means three properties, and this book shows how one harness reaches them stage by stage. Stages reproduce the same output from the same inputs and files. Each call is attributable to a model, a tool, and a source. Each number is honest, measured against a gold set graded independently of the system under test (by LLM judges, with the limits Chapter 7 states), with a null result reported as one. With all three properties in place, a challenged artifact produces evidence: the run that made it, the model that served each call, the source behind each claim.

Who this book is for

You build LLM pipelines past the demo stage: agents with tools, hybrid retrieval, a fallback chain, a critique loop. You write prompts and Python, and you have shipped at least one RAG system that mostly works. Right now a pipeline's record lives in your terminal history and your memory, and last month's run cannot be reconstructed. What you lack is the machinery around the model calls: files, gates, caches, traces, gold sets.

The reader this book assumes is the engineer who holds the trace and answers when the artifact is challenged. It quotes real source from a real harness rather than pseudocode, because the claims live in details like how a cache key is built, what a span records, and how a judgment is pooled.

Each chapter names one mechanism, shows it in code, and closes with the number that says whether it worked. Some numbers are small and unglamorous. A reranker that visibly spreads its scores looks like an obvious win until you measure recall the way Chapter 7 describes.

The bookgen case study

bookgen is a multi-stage harness that researches a topic and writes a Leanpub-ready Markua book. Its target is the kind of technical book that carries 64 to 148 references and cites source files at tags and commits. bookgen is not published, so the commands shown throughout this book illustrate its interface and cannot be installed and run. The stage pipeline fits in two lines:

```
1 scope -> research -> [research gate] -> plan -> [plan gate] -> per chapter:  
2 research, write, critique, revise, digest -> back matter -> assemble -> build
```

Both gates are human review points. The research gate lets you review and extend the corpus before anything is planned from it, and the plan gate holds the outline until you approve it.

Each stage reads and writes files under the project directory, so any stage can be re-run in isolation and a crash costs a single chapter, no more. Chapters are written strictly sequentially, and each runs critique and revision until the quality threshold or the round cap.

A book is the stress test because it is the hardest pipeline output we know. It must be planned before any prose, stay consistent across chapters that do not see each other's text, and cite claim by claim to sources a reader can check. The design started from four claims about how such books get made, and each became a design decision: generate in stages rather than one shot, draw knowledge from research and retrieval through tool calls, critique until a quality threshold, and expect context rot past roughly 60 percent context usage.

Scale is part of the audit. The completed eight-chapter run cost \$0.2577, taken from what the provider charged rather than from token arithmetic. A two-chapter smoke run with no critique costs about two cents. At that scale you can afford to run the pipeline until it breaks and rerun it after the fix, which is how each mechanism in this book got its number.

Three books supply most of that evidence. The first test book is that eight-chapter run. The second test book is a fourteen-chapter run over 291 sources that cost \$0.4719. The third is a generated networking book, which Chapter 11 compares against a human-written book on the same subject to test whether the prose still reads as machine-made. The subjects matter less than the runs, so the book refers to them by these names throughout.

The route through the book

Each stage contributes one mechanism. Chapter 2's files make replay deterministic, Chapter 5's spans make each call attributable, and Chapter 7's gold makes the numbers checkable. When one fails, the fix lives in that stage rather than in another prompt.

Chapters 2 through 5 build the foundation. Chapter 2, Files as the Source of Truth, decides what is authoritative and what rebuilds from it. Chapter 3, Typed Outputs and Human Gates, makes each stage boundary a typed contract with a human approval point. Chapter 4, Fallbacks and Model Provenance, keeps the pipeline running through outages while recording which model actually served each call. Chapter 5, Tracing as Audit, exports OpenTelemetry GenAI spans to Phoenix and turns each model call and tool invocation into a queryable trail [1].

Chapters 6 and 7 cover retrieval and its measurement. Chapter 6, Hybrid Retrieval and Citation IDs, binds each generated claim to a specific source the

reader can check. Chapter 7, Measuring Retrieval Honestly, builds the graded gold set and runs the paired bootstrap, then reports the reranker null result.

Chapters 8 through 11 cover what happens after a chapter is drafted: the critique loop that revises it, the digest that carries it forward, the smoke run that bounds its cost, and the style checks that strip machine prose. Chapter 8, Critique and Revise Loops, makes critique typed, gated, and replayable. A deterministic validator rejects drafts that cite invented ids before the critic sees them, and the critic catches fabricated content under real ids, with the checks' own false-positive rate part of the accounting. Chapter 9, Context Rot and Digests, explains why the rule of thumb that quality degrades past about 60 percent context usage shaped the architecture without ever binding a single prompt. The full corpus sits far above that threshold, so chapters retrieve excerpts instead of stuffing the corpus in. The digests, measured, take about 0.26 percent of the window at fifteen chapters, so they earn their place through cross-chapter relevance rather than compression. Chapter 10, Cost Control and Smoke Runs, keeps spend bounded with a cheap end-to-end run and per-call cost provenance. Chapter 11, Stripping Machine Prose Habits, counts the habits that mark machine prose and verifies the fix with a blind evaluation. A judge model scored per-paragraph AUC 0.42 when asked to tell the generated networking book's prose from the human-written one. That is one judge model, one book pair and 40 paragraphs, measured after a polish pass, and 0.42 sits at the edge of the ± 0.08 noise interval around chance. It shows that this one judge could not reliably tell the two apart, and says nothing about other readers.

Read it straight through once, or jump to the stage that is hurting you: a run that degraded silently, start at Chapters 4 and 5; retrieval that returns plausible nonsense, 6 and 7; a review loop that rubber-stamps, 8. The ground layer comes first. Chapter 2 starts with the filesystem as the source of truth, and each later stage builds on it.

The auditable pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Files as the Source of Truth

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Why files beat state

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Per-project artifact layout

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

JSON authoritative, corpus.db derived

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Content-keyed caches

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

The chapters_done resume, read from files

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Fork and select versions

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Where determinism breaks

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Typed Outputs and Human Gates

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Pydantic models at every boundary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Validation before you trust

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Stage gate reviews

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

The human gate checkpoint

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

What the gates caught

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Failing fast at boundaries

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Fallbacks and Model Provenance

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Why fallbacks exist

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

pydantic-ai FallbackModel

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

OpenRouter as the router

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Free-tier degraded fallbacks

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

request.model versus response.model

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

The degraded free-tier run

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Provenance through the stack

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Tracing as Audit

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

OpenTelemetry GenAI semantics

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Spans for every model call

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Tool invocations on the trace

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Phoenix as the audit pane

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

The first full trace

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Tool errors hidden from the run

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

pinned_only on two of thirty

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Retrieved but never cited

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Evidence you can trust

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Hybrid Retrieval and Citation IDs

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Chunking the corpus

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

BM25 and vector branches

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Reciprocal rank fusion

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Pinned boost and diversity decay

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Citation ids tie claims to sources

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Context assembly and limits

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Early retrieval failures

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Measuring Retrieval Honestly

Building graded gold sets

You change chunking, or fusion weights, or you slot a reranker in front of `search_corpus`. The next run looks better: each call returns eight excerpts, several pinned, and the text reads on-topic. Those signals are exactly the ones that fool you, because they cannot show whether the eight slots hold the passages a reader needed. Only judged (query, passage) pairs, each graded 0 to 3 by judges who did not see your change, can answer that. In bookgen those judges were LLM subagents. What makes their grades usable is that they were produced independently of the system under test: no retriever being scored had a hand in its own answer key. This section builds a set of them the way `eval/pool.py` does, under one rule: no system is scored against a gold it built.

The gold for a corpus lives in `eval/relevance-gold-<slug>.json`, and the file declares its own scale, 0=irrelevant 1=marginal 2=relevant 3=authoritative. bookgen now carries two, one for the first test book and one for the second test book. Together they held 60 judged queries and 1,299 relevance judgements when the second was finished.

The first choice that keeps the gold honest is that the queries come from the runs themselves. `pool.py` reads the book's actual `search_corpus` calls back from Phoenix, `span_kind=TOOL`, so the gold measures the workload that happened rather than one somebody imagined. Phoenix spans carry no project id, so `corpus_queries` keeps only the queries whose returned passages match this corpus's text prefixes, which makes the corpus itself the filter. `spread()` then samples 30 queries evenly across the run, so the judged set covers the whole book rather than its first chapter. The second choice is the pool. For each query, `candidates()` takes the top 10 of each system that will be scored (BM25, vectors, RRF fusion and the shipped pipeline) and unions them. A challenger you did not pool cannot be judged relevant, and a gold the incumbent built for itself flatters the incumbent.

```
1 uv run python eval/pool.py pool my-book -n 30 --batches 6 \  
2   --since 2026-09-19T06:30:00+00:00  # start of the run; needs Phoenix  
3 #   hand each eval/judge/<slug>/batch-N.json to an independent judge;  
4 #   each writes batch-N.grades.json as {sha256: 0-3}  
5 uv run python eval/pool.py merge my-book
```

Treat these commands as a template. `--since` must be the start of the run whose queries you want, since only spans after it are read. `write_batches` splits the pool into six judging batches, and `merge` folds the graded batches into one gold. It rejects any set of grades that does not exactly cover the pool, a check that has already caught a judge who dropped one character from a sha256. Judging is the expensive step. Six LLM subagents, each given one batch and nothing about the systems that produced it, spent about 1.1M tokens grading the first gold for one corpus, which is why re-pooling and re-judging are the steps you postpone.

The parts are the standard test-collection ones: a document collection, a set of topics and relevance judgements, which is what lets the apparatus lean on TREC-style practice [39]. Pooling each system under test at depth 10, before any judging, is what keeps a single model from writing its own answer key.

Hash matching lost 518 judgements

The first gold was graded as 809 judgements over 30 queries. After a later fix removed 145 duplicate chunks from the first test corpus's index, the gold was saved to disk as 719 judgements keyed by `content_sha256` instead of chunk id, on the assumption that this key would let judgements survive a reindex that moves chunk boundaries. The mapping showed 0 unmapped across two full rebuilds. The claim held for one input only, a corpus chunked exactly as the judges saw it. A hash matches exact text, and a chunking change moves the boundaries that decide what text a chunk holds.

The chunk-size sweep made the failure obvious. Re-chunking the corpus at 100/600 tokens left 235 of the 719 judgements mappable. At 100/400 it left 201, so the worst case silently lost 518 of 719. Each candidate was then scored against a different and smaller gold, and the incumbent 100/800 config, the one that kept all 719, came out ahead for the single reason that it was the config that had produced the gold.

A shrinking answer key lies. An eval that drops its own judgements in proportion to how far a candidate departs from the incumbent will re-elect the

incumbent. The hash did nothing wrong. The fixture was at fault, and it pointed toward whichever configuration produced it, the one direction a fixture must not point.

Containment matching fix

The fix was to judge passages rather than chunk ids. Each judgement now carries its passage text alongside the grade, and `score.py` decides coverage by containment. A retrieved chunk counts as covering a judged passage if one of the two texts contains the other, or if the 200 characters at the centre of the shorter text occur somewhere in the longer one. That relationship holds however the boundaries were drawn, because the test compares text and a chunking change moves only boundaries. `grade_hits` and `per_query` normalize whitespace before calling `covers()`, so a line-break difference does not defeat the test.

Re-scored against the whole gold for each candidate, the sweep stopped being a foregone conclusion. The incumbent 100/800 chunking (1,092 chunks) held its place at 0.649 nDCG@8, 0.660 found@8 and 7.3 distinct sources per eight excerpts. The 100/400 config (1,753 chunks) scored 0.629 nDCG and 0.574 found@8, and 100/600 (1,312 chunks) came in at 0.611 and 0.609. The 100/1200 config (908 chunks) landed at 0.590 and 0.610, and 50/800 (1,222 chunks) at 0.630 and 0.621. Only 200/800 (912 chunks) came near the incumbent, at 0.648 nDCG and the sweep's best found@8 of 0.686. It does not survive the same bootstrap that was applied to the reranker: nDCG -0.001 with a 95% CI of [-0.020, +0.020], found +0.026 with [-0.014, +0.067]. The defaults stay. `chunk_min_tokens` stays at 100 and `chunk_max_tokens` at 800.

The fix also had to stay invisible to the rest of the pipeline. `content_sha256` continues to hash the raw text alone, so when contextual chunks later entered the index, the 719 judgements kept mapping. Had the hash covered the contextualized string, all of them would have stopped mapping in one reindex.

NDCG and recall at K

`eval/score.py` ranks each gold query with whichever system you name, then reduces the ranking to numbers. `--system` picks `shipped`, `fused`, `bm25` or `vec`. Only `shipped` calls the live `HybridRetriever`, with pinned boost and

diversity included. The other three are built from the same corpus database's BM25 and vector searches and the same `r rf_fuse`, so the comparison isolates ranking choices inside one harness. A grade counts as relevant at 2 or above, and `grade_hits` scores each hit with the maximum grade among the judged passages it covers, 0 where it covers none.

The two reported metrics answer different questions. `nDCG@k` is rank-weighted. Ranks are counted from 0, so a hit at rank r with grade g contributes $(2^g - 1) / \log_2(r + 2)$, and the top hit is divided by $\log_2(2) = 1$; the ideal is the gold's grades sorted descending and truncated to k , and the query score is the ratio [40]. That is the graded form of the standard measure, and it pays its way when judgements are graded and systems differ mainly in the order of what they return [41, 40]. `bookgen`'s writer reads all eight excerpts, though, so what the eight slots contain matters more to it than their order, and `found@8` is the metric this consumer actually uses.

`found@k` counts distinct judged passages with grade 2 or above covered by the k hits, divided by $\min(\text{len}(\text{relevant}), k)$. The normalization matters because a query can have more relevant passages in the gold than the writer has slots. Without it, a perfect ranking could score below 1. `found@8` replaced an older `recall@8` that counted relevant hits per slot. That per-slot recall credited a duplicated chunk twice, so it rewarded an index full of duplicates, and it is the metric behind the earliest numbers in this chapter: the reranker verdict and the first BM25 comparison. Those figures are labelled per-slot `recall@8` below. They cannot be compared with `found@8`. Both metrics are computed and written per query, which is what lets the bootstrap resample queries rather than averages.

The paired bootstrap test

A mean difference over 30 queries is a number with no error bar, and the error bar is the point. Both systems run on the same queries, so a hard query depresses both. A paired test is the standard control for that topic difficulty, because pairing holds each query's difficulty fixed [42]. In IR the bootstrap has long been advocated because it makes no normality or symmetry assumptions [42]. - - `compare` runs a paired bootstrap over the queries instead of comparing two averages.

The implementation in `score.py` is short, and worth reading whole (type hints and docstring removed):

```

1 def bootstrap(a, b):
2     rng = random.Random(0)
3     pairs = list(zip(a, b, strict=True))
4     deltas = []
5     for _ in range(RESAMPLES):
6         sample = [pairs[rng.randrange(len(pairs))] for _ in pairs]
7         deltas.append(mean(x for x, _ in sample) - mean(y for _, y in sample))
8     deltas.sort()
9     lo = deltas[int(0.025 * RESAMPLES)]
10    hi = deltas[int(0.975 * RESAMPLES)]
11    return mean(a) - mean(b), lo, hi

```

Each resample draws n query pairs with replacement, computes the mean of A minus the mean of B on that draw, and stores the delta. Ten thousand draws give a distribution of the paired difference, and its 2.5 and 97.5 percentiles are the 95% CI. The fixed seed makes the test reproducible. `--compare` reads two dumps, keeps the queries both systems scored, and prints the delta, the CI and per-query better/worse counts for nDCG and found.

```

1 uv run python eval/score.py my-book --dump /tmp/shipped.json
2 uv run python eval/score.py my-book --system bm25 --dump /tmp/bm25.json
3 uv run python eval/score.py --compare /tmp/bm25.json /tmp/shipped.json

```

The rule the harness adopted is that a delta whose CI includes zero is not a win. At this query count it is unresolvable. On 30 queries the noise floor is roughly ± 0.05 nDCG, and the candidates tried after the reranker all landed inside it: 200/800 chunking at -0.001 , deduplication at $+0.038$, contextual chunks at $+0.029$. Doubling the gold to 60 queries, pooled over two corpora, separated what 30 could not: fused retrieval beat vectors alone by $+0.079$ nDCG and the shipped pipeline by $+0.046$, both with CIs clear of zero, while fused over BM25 still crossed zero on nDCG and cleared it only on found@8.

The reranker null result

The reranker entered the pipeline on the strength of score spread. RRF reads ranks and ignores scores, so what it returns barely separates. Over the 75 real queries of one chapter run, replayed against one index, the first hit beat the eighth by only 1.26x, which left `pinned_boost` multiplying noise. A cross-encoder reading query and chunk together spread the same eight results 4.47x,

and the harness shipped it as an obvious win, at 7x the latency and 550 MB of VRAM on a card that had just run out of memory.

The graded gold, 809 judgements over 30 of the real queries, was built to check exactly that. Measured against it, reranking moved per-slot recall@8 from 0.548 to 0.547 and lost nDCG@8 from 0.689 to 0.659. The paired bootstrap over the 30 queries, with A – B as the delta:

system A	system B	Δ nDCG@8	95% CI	CI clear of zero?
plain RRF	fusion plus	+0.030	[-0.056,	no
fusion	reranker		+0.120]	
plain RRF	fusion plus	+0.033	[+0.001,	yes
fusion	diversity		+0.067]	
	decay			
RRF	RRF 1:1	+0.013	[-0.008,	no
weighted			+0.035]	
3:1 toward				
BM25				

The first row is the reranker’s verdict. The CI spans zero in both directions, so the gold cannot say the reranker hurt ranking, and it cannot say it helped. The point estimate leans toward the pipeline without it. That was enough to turn it off. The case for it had been the spread, and the gold showed no measurable benefit behind the spread, while the costs (7x latency, 550 MB of VRAM) were certain. The lesson is that spread is confidence, not accuracy. A cross-encoder trained on web prose does not transfer to C headers and other source files, and a confidently wrong ordering looks exactly like a confidently right one until something scores it. The cross-encoder could not have graded itself either. Only 65% of its top-8 were rated relevant by the judges, so a reranker-built gold would have measured family resemblance between transformer rankers.

The second row is the one with a CI clear of zero, and the setting it measures stayed on. On nDCG@8 the diversity decay is significantly harmful. On per-slot recall@8 it moves 0.548 to 0.530 with the CI crossing zero, and it buys 1.5 more distinct sources per query. Because the writer reads all eight slots, the decay stayed on as a stated trade, breadth of sources bought at a measured price. The 60-query gold later re-measured this trade, as the section on removing the reranker explains.

Pooling bias and lower bounds

Any score for a system that was not in the pool when the gold was judged is a lower bound, and you should read it as one. The chunking candidates are the clearest case. The pool was built from one chunking, so a different chunking gets no credit for surfacing a passage nobody judged. Those passages exist, and the gold does not know them. The consequence is structural. A pooled gold can only penalize a challenger, which is the right direction for a conservative test and the wrong direction for discovering a genuinely different approach. The pooling rule follows from that: each system whose score will be reported has to be in the pool at pooling time.

The literature documents the same bias at scale. Pooling judges a sample of the collection and treats unjudged documents as non-relevant, and as a fixed-size pool covers a shrinking share of a collection, the judgements drift toward the documents the pooled systems favor [43]. bookgen's pool has the same structure at depth 10, so its errors run the same way: a passage no pooled system returned reads as a miss [43].

Extending the gold made the bias measurable. When the harness later judged a challenger embedder, it was scored on two corpora, each with vector and fused retrieval, which gives four nDCG intervals. On the golds as they stood, the challenger lost by up to 0.107 nDCG, and two of the four intervals excluded zero. Six blind LLM judges then graded the 275 passages in the challenger's top-10 that no prior judgement covered. Afterwards none of the four did. On the second test corpus, 45 of the 149 passages only the challenger surfaced were relevant, and the unextended gold had counted all of them as misses. On the first test corpus the same step cut the other way: only 4 of 126 challenger-only passages were relevant, so the challenger's shortfall there was not an artifact of the pool, and its fused found@8 deficit of 0.083 stayed clear of zero on the extended gold. A gold that has never seen a challenger's results reports the pool, and judging the pool is the only way to tell which case you are in.

Removing the reranker

The null result drove a configuration change and left the code in place. `rerank_model` now defaults to empty, and an empty model turns reranking off.

The reranker code stays in the tree, one setting away, because it is measured and a prose-heavy corpus is a different question. Against 809 relevance judgements it moved per-slot recall@8 0.548 to 0.547 and nDCG@8 0.689 to 0.659, both CIs crossing zero, for 7x the latency.

The removal also reverted a second setting that had been tuned against the reranked pool. `pinned_boost` had gone to 1.5 because a reranker's separated scores could withstand being multiplied. With the reranker gone, 1.15 returns. 1.5 suits a reranked pool, so raise it together with `rerank_model` and not otherwise.

The 60-query gold put a price on what remained. The shipped pipeline sat 0.046 nDCG@8 and 0.053 found@8 below plain fusion, both CIs clear of zero, in exchange for 5.8 to 7.4 distinct sources per eight excerpts. Two settings separate shipped from fused, `diversity_decay` and `pinned_boost`, and sweeping each over the same queries split the bill. Turning the decay off recovered +0.016 nDCG with the CI crossing zero. Turning the boost off recovered +0.020, 43% of the gap, with the CI clear of zero. The boost's share is the only one resolved from zero; the decay's is not. The two intervals overlap, so the gold cannot tell the two settings' costs apart, and they sum to 0.036, so neither alone explains the gap. The decay's +0.016 also differs from the earlier finding that it was significantly harmful on nDCG@8: that was a different baseline, a 30-query gold and an earlier index, and its CI lower bound of +0.001 was marginal.

By relevance alone the boost is monotonically harmful, which would read as "delete it". Relevance is not what it is for. It exists to put immutable, pinned revisions in front of the writer, and measured by that purpose it works: going from 1.0 to 1.15 raises the pinned share of retrieved excerpts from 14.6% to 22.9%, 8.3 points for 0.020 nDCG. The next 0.15 buys 6.1 points for 0.034, and the step after that 3.9 points for a further 0.016. Per point of pinned share, the first step costs about 0.0024 nDCG, the second 0.0056 and the third, a 0.2 step, 0.0041. The first step is by far the cheapest, so 1.15 sits at the knee of the curve and stays. A gold that measures only relevance will recommend removing any feature whose purpose is something else, so score such a setting on the metric it exists for as well.

The first gold also seemed to settle what the rest of retrieval was worth. On the first test corpus BM25 alone beat each embedder, 0.693 nDCG@8 against the best dense model's 0.616, because the queries are identifier-dense and that is the regime lexical retrieval owns. BM25 even edged plain fusion on nDCG,

0.693 to 0.689, and the conclusion at the time was that fusion earned its place only on per-slot recall, 0.520 alone to 0.548 fused. That conclusion did not survive a better instrument. Scored on the deduplicated index with the 60-query gold, fusion beats BM25 on both corpora, 0.715 against 0.696 nDCG@8 on the first test corpus and 0.747 against 0.696 on the second test corpus. Pooled over all 60 queries it takes found@8 by +0.068 with the CI clear of zero, while its +0.035 on nDCG still crosses zero. The earlier ordering was an artifact of the index it was measured on. BM25 is the strongest single signal on both corpora, and it still loses to being fused with a weaker one.

Reading numbers without lying

These measurements condense into rules you can apply to your own harness. Score against a gold that includes your system in the pool. Never let the answer key shrink. If a fixture drops judgements when inputs change, the comparison is rigged. Report each delta with its CI, and call a change a win or a loss only when the CI excludes zero. Report within-noise changes as measured. The 0.038 nDCG shift from deduplication was published that way because it had not met the bootstrap standard. Label lower bounds as lower bounds. Treat absolute scores across gold versions as incomparable. The golds grew from 719 to 845 judgements on one corpus and from 580 to 729 on the other, and the incumbent's own score moved when relevant passages it had missed entered the gold, so compare deltas within one gold version.

Judge agreement belongs in the accounting too. Per-batch rates of “irrelevant” grades ran from 3% to 62%, which looked like six judges disagreeing about the scale. A seventh judge re-graded 96 pairs and found binary agreement of 0.85 at the ≥ 2 threshold the metric uses. The spread came from query difficulty, and the independent judge ranked the weak batches weakest too.

Keep the command that repeats the whole verdict written down next to your harness. Once both dumps exist, it is one line:

```
1 uv run python eval/score.py --compare /tmp/bm25.json /tmp/shipped.json
```

Run it before and after any change to chunking, fusion or retrieval. When a candidate lands inside the noise floor, the honest verdict is “not resolvable at this query count” rather than “no effect”, and the next useful spend is

more judged queries rather than another candidate. That is the discipline the reranker taught. It looked like the obvious win, and a graded gold, audited for lost judgements and for pooling bias, showed no gain behind the spread. The measurements worth returning to are the ones that can say no.

Closing the loop

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Critique and Revise Loops

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

The critique stage

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Self-refine feedback loops

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Generated books as stress test

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Fabrications under valid citations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

The deterministic citation validator

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Resolving only known ids

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

When revise makes it worse

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Replaying critiques for evaluation

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

When a deterministic check is wrong

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Context Rot and Digests

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Lost in the middle

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

How context rots

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

The measured rot trigger

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Chapter digests at 187 tokens

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Continuity, not mitigation

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Part intros from digests

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Stale digests after re-approval

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Measure before you mitigate

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Cost Control and Smoke Runs

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Where the money goes

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Cheap end-to-end smoke runs

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Bounding spend per run

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Free tier as cost lever

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Real costs of bookgen

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Stripping Machine Prose Habits

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

The style lint pass

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Machine prose tells

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Blind voice evaluation

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

AUC 0.42 and 12 of 20

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

The habit tell list

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Linting without flattening voice

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

What you can now do

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

The mechanisms that transfer

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Where bookgen goes next

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

The audit reflex

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Appendix A: bookgen CLI and stage command reference

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Command reference

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Stage pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Research and retrieval knobs

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Per-chapter spend caps

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Project layout

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Measuring the harness

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Tracing

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

The critique and revision loop

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Publishing

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Appendix B: GenAI semantic convention attribute table

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Inference span attributes

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Tool span attributes

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Operation names

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Metrics and events

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Renamed or removed

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

What bookgen reads

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Appendix C: Gold set and scoring schema reference

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Files

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Relevance scale

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Building a gold

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Covering rules

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Score.py reference

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Comparing systems

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

The two golds

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Limits

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

Glossary

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.

References

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/auditable-llm-pipelines>.