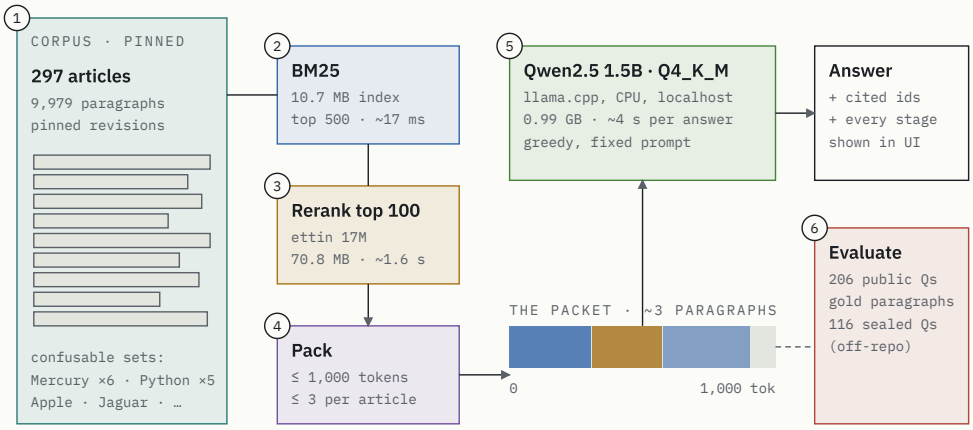
PLATE I · ANATOMY OF ONE ANSWER — MEASURED · DESKTOP CPU, 6 THREADS



KEY

1. **CORPUS** — *rebuilt from pinned revisions, sha-checked*
2. **FIND** — *already holds 96% of the evidence*
3. **REORDER** — *the one model that earned its weight*
4. **FIT** — *where the evidence used to get lost*
5. **WRITE** — *the weakest link today*
6. **MEASURE** — *every claim has a split and a threshold*

Fig. 0 — One question, four stages: about five seconds once warm on a desktop CPU (11.6 s for a cold one-off run), with no network. The interesting stage is ④: the evidence was usually found; it just didn't fit.

lmr

Technical Manual

An offline research assistant, and the experiment that decided how to build it. It answers questions from 297 Wikipedia articles using only what runs on this machine: a keyword index, one 17-million-parameter reranker and a 1.5-billion-parameter language model. The research behind it asked whether better retrieval matters on a device. The answer was *no, but better packing does*.

● CORPUS ● RETRIEVE ● RERANK ● PACK ● GENERATE ● EVALUATE

Code: `lmr/` (app) · `substrate/`, `eval/`, `experiments/` (research) · 518 tests
Runs on: Linux x86-64, Python ≥ 3.11, ~2.3 GB disk, no GPU · Built for poidh bounty #31 (see §1.2)

PART I · ORIENTATION

1 What it is for
2 Running it
3 The corpus

PART II · THE FINDING

4 Found is not shown
5 The reranker is an allocator
6 What each model costs
7 The generator ceiling

PART III · METHOD

8 How the experiment was run
9 What the reviews caught

PART IV · REFERENCE

10 Limits and next steps
A Who did what
B Sources
C Changelog

PART I

Orientation

What the tool does, how to run it, and what it knows.

1

What it is for

A lookup tool for when there is no network, and a measured answer to which parts of one are worth building.

1.1 The tool

Ask a question in a terminal (`lmr ask`) or a local web page (`lmr serve`). The assistant searches its corpus, picks the paragraphs that fit a fixed budget, and has a local model write an answer that cites them. Nothing leaves the machine: the language model is a `llama-server` bound to `127.0.0.1`, and the reranker runs on the CPU in-process.

The web page shows every stage of an answer, not just the result: which paragraphs keyword search found, how the reranker reordered them, which ones fit the 1,000-token budget, and which the model cited (Fig. 2.2).

1.2 Where it came from

The project started as a response to a public bounty: “Build the Best Offline AI Research App for Android” (poidh #31, about 1.5 ETH, closing 2026-10-31). Its bar is a phone app under 12 GB of RAM that is “more than 50% as good as internet plus frontier models”, with no network at all.

Instead of building an app first, this project asked a narrower question that any such app has to answer: **on a phone-sized model, does retrieval quality actually change the answer, and what is the cheapest design that captures the difference?** The generator was held fixed on purpose, so that if evidence quality barely moved the answer, the project could say so.

> SCOPE
>
> Every measurement in this book was taken on a desktop CPU. None of it is a claim about Android or GrapheneOS performance. The Android app is not built (Chapter 10).

2

Running it

Three commands from a fresh clone. The first one downloads 1.2 GB and rebuilds the corpus; after that nothing touches the network.

```
# Python 3.11+, uv. CPU only.
git clone https://github.com/davidghatley/local-mobile-research && cd local-mobile-research
uv venv && uv pip install -e ".[rerank]"
.venv/bin/lmr setup        # llama.cpp b11227 (CPU) + Qwen2.5-1.5B GGUF, sha-checked; corpus
.venv/bin/lmr ask "Which planet is closest to the Sun, and how long is its year?"
.venv/bin/lmr serve        # http://127.0.0.1:8765
```

FIG. 2.1 · WHAT `LMR SETUP` PUTS WHERE — STRUCTURE · SIZES MEASURED

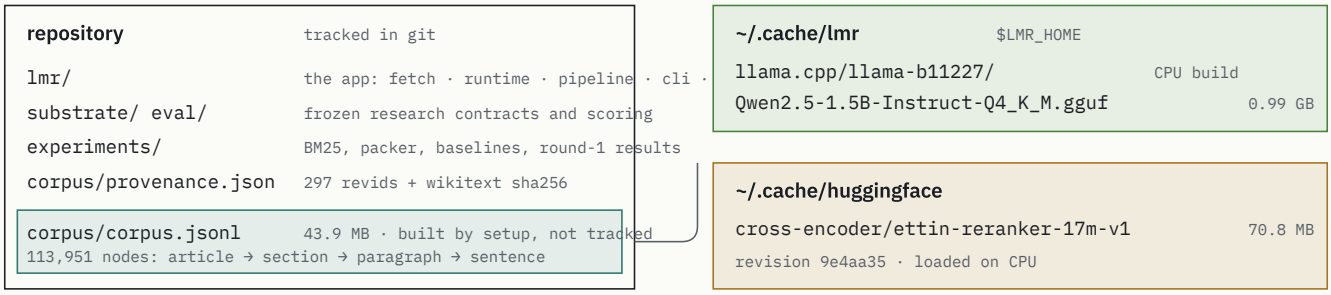


Fig. 2.1 — Code and pins live in git; weights and the built corpus live in caches. Every download is checked against a hash recorded in the repository before it is used.

FIG. 2.2 · ONE REAL ANSWER, EVERY STAGE

LMR ASK --JSON · BM25 18 MS · RERANK 1482 MS · GENERATE 3.3 S

Q. Which planet is closest to the Sun, and how long is its year?
A. Mercury is the closest planet to the Sun, and its sidereal year is 88 Earth days.

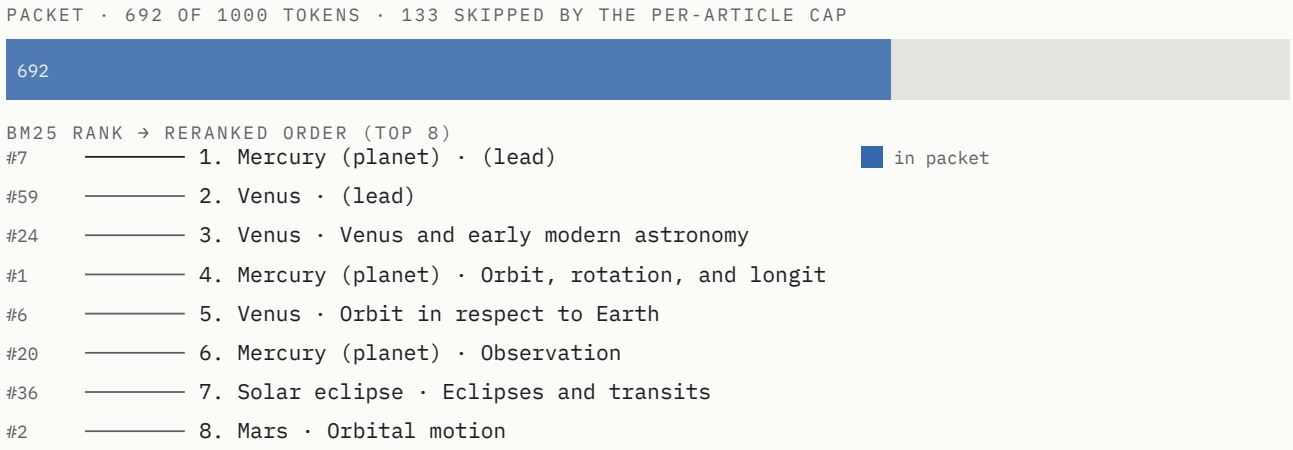


Fig. 2.2 — The web UI draws this live for every question. Source: `docs/manual/data/example_answer.json`.

PITFALL · FIRST RUN

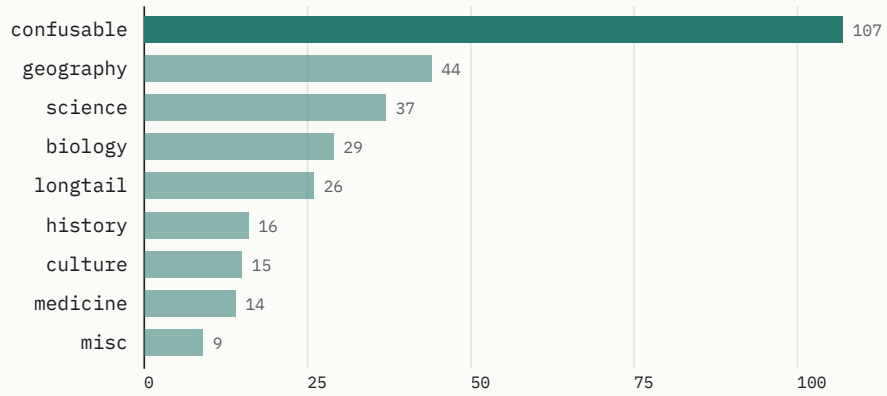
The reranker is fetched from Hugging Face the first time the pipeline loads (about 70 MB). Run `lmr ask` once while online after `lmr setup`. Pass `--no-rerank` to run on BM25 alone with no Python dependencies beyond the standard library.

3

The corpus

297 articles chosen before any question was written, weighted towards names that mean several things at once.

FIG. 3.1 · ARTICLES PER TOPIC CLUSTER — COUNTED FROM CORPUS/CORPUS.JSONL



WHY "CONFUSABLE"

1. **SAME NAME** *Mercury the planet, element, god, car, label and singer*
2. **SAME WORDS** *keyword search scores them alike*
3. **SAME GROUP** *kept in one split, so no answer leaks across*

Fig. 3.1 — More than a third of the corpus is deliberately ambiguous, which is where a reranker should earn its keep.

3.1 Pinned, then rebuilt

`corpus/provenance.json` records, for each article, the Wikipedia revision id and the sha256 of its wikitext. `lmr setup` fetches those exact revisions (not today's pages), refuses any whose bytes differ, and runs the original offline converter. All 297 matched on 2026-10-15.

> **VERIFY**
>
> The rebuild reproduces the research numbers exactly: BM25 packet completion is 54/73 on dev and 80/122 on shadow_dev, the same as the frozen round-1 results, and all 235 gold evidence ids resolve. One known difference: article-level cross-links came from a live API list at the original fetch and are re-derived from wikitext here, so the file hash differs. Retrieval does not read them.

3.2 The questions

SPLIT	QUESTIONS	USE	WHERE
train	60	Reference only; never used to fit anything here	`corpus/questions.json`
dev	59	Tuning and quick checks	same
shadow_dev	87	One frozen confirmation run per system	same
private test	116	Sealed; never scored against	outside the repo, hash only

Types across the public splits: factoid 128, multihop 33, comparison 17, ambiguous 15, unanswerable 13. Each answerable question records the exact paragraphs that are sufficient to answer it; that is what “required evidence” means in Part II.

PART II

The finding

Retrieval was already good. The evidence got lost on the way to the model.

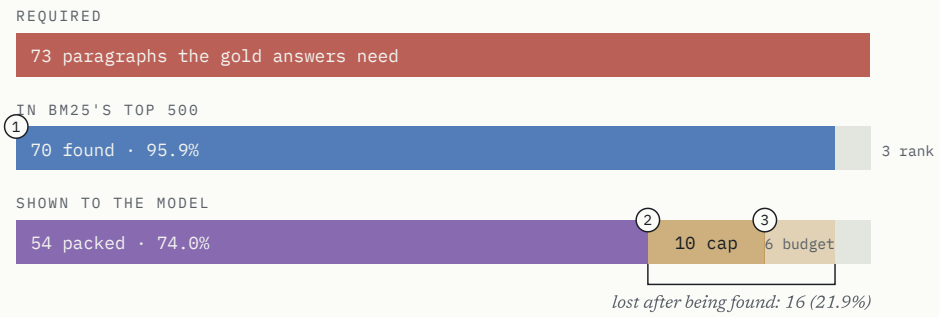
4

Found is not shown

BM25 found almost every paragraph a question needed. A quarter of them never reached the model.

FIG. 4.1 · WHERE REQUIRED EVIDENCE GOES · DEV, BM25

73 REQUIRED PARAGRAPHS · LOSS_ATTRIBUTION_B1_BM25_DEV_CORE.JSON



KEY

1. **SEARCH** *headroom here is 4.1%*
2. **PER-ARTICLE CAP** *three paragraphs per article, max*
3. **TOKEN BUDGET** *1,000 tokens ≈ 3 paragraphs*

Fig. 4.1 — The problem is allocation, not recall. A paragraph averages about 280 tokens, so the budget holds roughly three; which three is the whole game.

Lifting the per-article cap only moved completion from 54 to 56 of 73: the loss migrated from the cap to the budget. The two limits are substitutes. What helps is putting the *right* three paragraphs first.

5

The reranker is an allocator

A 17M cross-encoder found nothing new. It reordered what BM25 already had, and 13 points more of the needed evidence reached the model.

FIG. 5.1 · PACKET COMPLETION, BM25 VS THE 17M RERANKER

FROZEN ROUND-1 RUNS · WILSON 95% INTERVALS

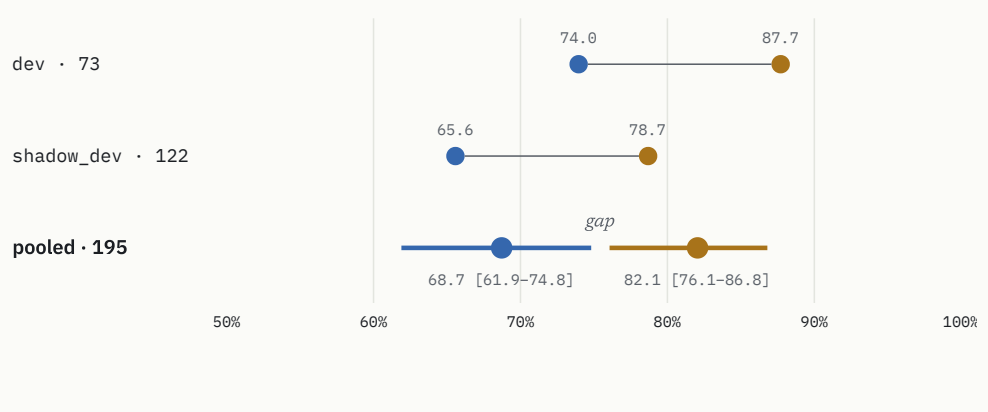


Fig. 5.1 — The only result in round 1 to clear its preregistered threshold. On dev alone (+10 vs a 13-item threshold) it would have been recorded as a null.

Every reranking stage read exactly the same search survival as BM25 (70/73): a reranker cannot add a paragraph BM25 did not return. The gain came entirely from the order, and it arrived using *fewer* tokens (740 vs 798 per packet on shadow_dev).

5.1 How deep does the reranker need to look?

The research reranked all 500 candidates, which cost 40 s per question on one thread. The app reranks only the top *k*. Measured with `lmr bench` on this machine:

FIG. 5.2 · COMPLETION AND LATENCY AGAINST RERANK DEPTH

MEASURED · LMR BENCH · DESKTOP CPU

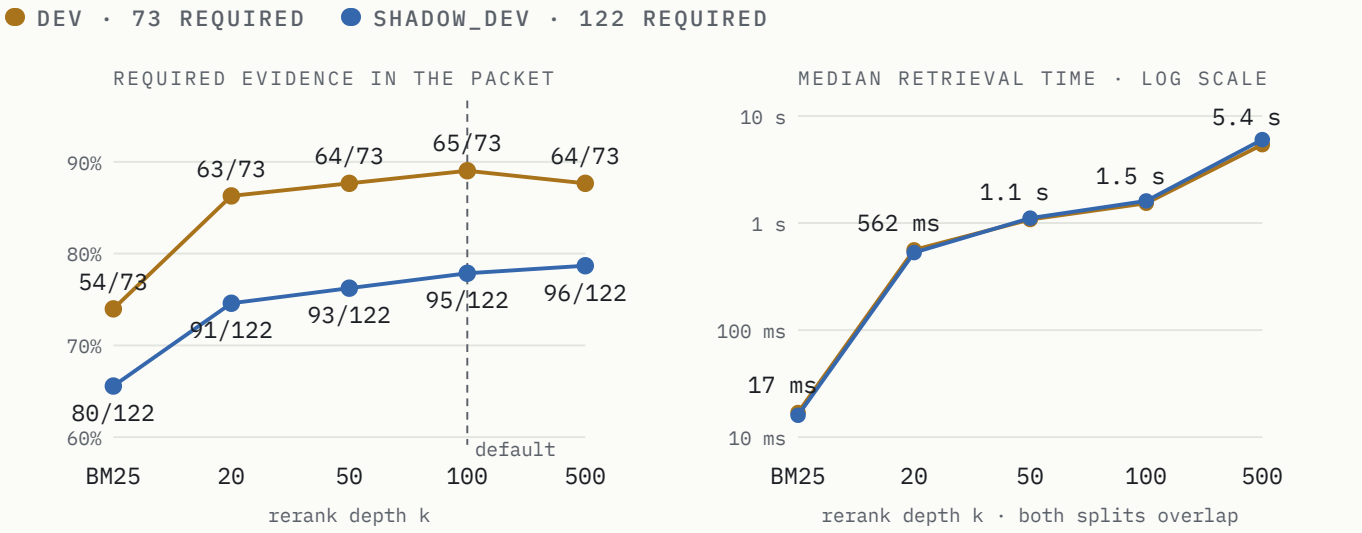


Fig. 5.2 — Most of the gain arrives by k = 20; k = 100 matches the full 500 within one paragraph per split at about a quarter of the time. Retrieval only (BM25 + rerank); generation adds about 4 s. Source: `docs/manual/data/bench_*.json`.

DECISION

The app defaults to `--rerank-k 100`. Pooled over both splits it keeps the full-depth result (160/195 required paragraphs) at about a quarter of the latency; $k = 50$ gives 157/195 in two-thirds of that time. The app and the frozen research system agree exactly at $k = 500$ on both splits (64/73, 96/122), which is a second check that the rebuilt corpus is faithful.

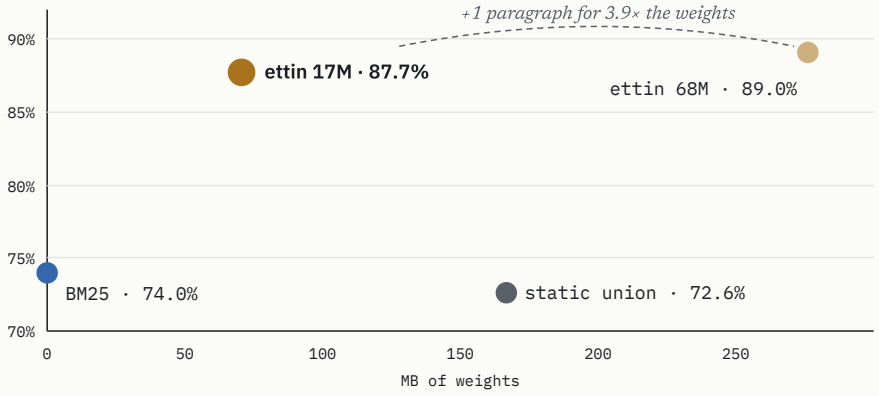
6

What each model costs

Bigger was not better. The 68M reranker beat the 17M by one paragraph at four times the size.

FIG. 6.1 · EVIDENCE COMPLETION AGAINST WEIGHT · DEV

R1_ANALYSIS.MD §2 · 73 REQUIRED PARAGRAPHS



LATENCY, RESEARCH SETUP

1. **BM25**
   36 ms
2. **STATIC UNION**
   119 ms · 166.6 MB incl. cache
3. **ETTIN 17M**
   39.6 s (all 500, 1 thread)
4. **ETTIN 68M**
   77.4 s · 1.9 GB peak RSS

Fig. 6.1 — The cheap semantic channel (static embeddings) added nothing, as predicted in advance. The 17M reranker captured almost the whole available gain.

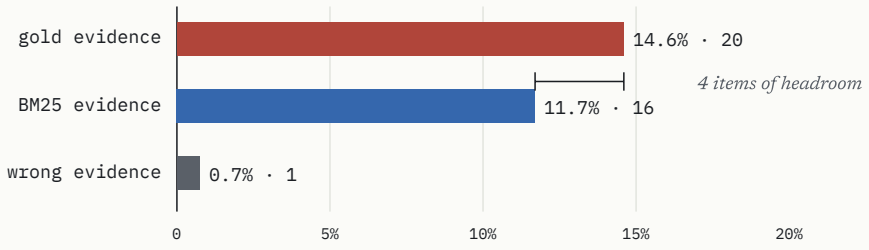
7

The generator ceiling

The 1.5B model can tell good evidence from bad, but even perfect evidence only gets it to about one answer in seven.

FIG. 7.1 · EXACT-MATCH ACCURACY BY EVIDENCE GIVEN

GENERATOR_FLOOR_REPORT.MD · 137 ANALYSABLE QUESTIONS



READING IT

1. **IT LISTENS** *gold beats wrong evidence on all 19 discordant items, $p = 3.8 \times 10^{-6}$*
2. **IT CAPS OUT** *perfect retrieval buys 4 answers; detecting a difference needs 18*

Fig. 7.1 — No retrieval change can show up in final answers on this corpus with this model. That is a property of the generator, found before any architecture was compared.

PITFALL · CITATIONS

The fixed prompt asks for bracketed citations. In practice the 1.5B model rarely emits them (none in six sample questions on 2026-10-15) and adds facts that are not in its evidence. Asked how Freddie Mercury got his stage name, it answered that the stage name was “Farrokh Bulsara”, which is his birth name. Asked to compare Mercury the planet and the element, it claimed both have atomic number 80. The UI shows which packet paragraphs were cited, so this is visible rather than hidden.

This is the clearest next step: fine-tune a small model to write short, cited answers from a packet (Chapter 10).

PART III

Method

How the results above were protected from wishful thinking.

8

How the experiment was run

Thresholds first, systems second, the confirmation split once.

FIG. 8.1 · SMALLEST DIFFERENCE EACH SPLIT CAN DETECT

EXPERIMENTS/COMPARABILITY/CONTRACT.MD §4

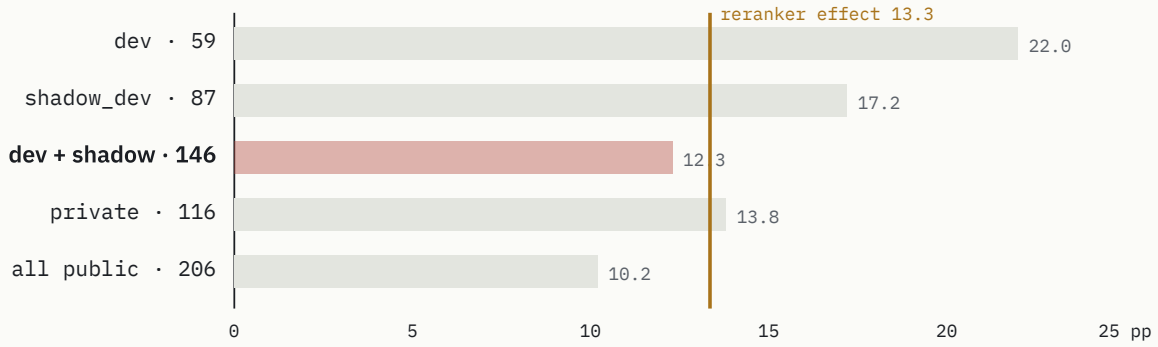


Fig. 8.1 — A gap shorter than its bar is recorded as a null result, never a win. The reranker clears only the pooled bar it was tested against.

- **Preregistration.** Hypotheses, thresholds and kill conditions were committed before code (`experiments/*/HYPOTHESIS.md`), and the BM25 parameters are pinned to the bit (`experiments/common/bm25.py`).
- **One shot at confirmation.** The reranker's configuration was frozen (`FREEZE_RECORD.md`) before its single shadow_dev run.
- **Sealed test.** 116 private questions were written, reviewed and sealed outside the repository; only their counts and hash are tracked. No system was ever scored against them.

- **A freeze gate.** `scripts/verify_round0` checks ten stages (hashes, splits, leakage, custody) and `docs/probes_round0.sh` runs eleven adversarial probes against it.

9

What the reviews caught

Eight independent freeze reviews. Seven said "repair required". Each defect was invisible to the checks written by the same team.

DEFECT	WHY THE EXISTING CHECK PASSED
A partition census that agreed with itself	It was derived from the data it was meant to check
A truncated partition	It passed three consecutive stages because each stage was internally consistent
A fix defeated by editing the data and its census together	Both sides moved; the trust root had to move outside the repo
A stage that compared nothing and printed PASS	Vacuous success: no test asserted the stage had inputs
A leakage gate that could never pass, then one that leaked	Once repaired it worked as a membership oracle and had to be re-scoped
A preregistration anchor pointing at a commit without the policy file	Nobody resolved the anchor
A minimum-difference figure wrong by about 2×	Refuted by its own worked example three lines earlier

> **LESSON**
>
> An internal consistency check is necessary and never sufficient. The same failure shape appeared for a single item, a whole partition and the entire freeze. Every one of them was found by a fresh reader. The full account is in `docs/ROUND0_CLOSURE.md`.

PART IV

Reference

Limits, credits and sources.

10

Limits and next steps

ITEM	STATE	NEXT STEP
Answer quality	WEAK 1.5B model, rare citations	Fine-tune a 1.7–4B model for cited answers from a packet (QLoRA with Soup on a 12 GB GPU) and score it on the same gold/BM25/wrong arms
Android	NOT BUILT	llama.cpp ships Android arm64 builds; the reranker needs an ONNX export and an int8 parity gate first
Private test	LOST ON THIS MACHINE	The sealed store went missing on 2026-10-04 (`docs/CUSTODY_INCIDENT_2026-10-04.md`); it was never scored, so no result depends on it
Corpus size	297 ARTICLES	A 940-article distractor tier exists (`corpus/scale/`); the app does not load it yet
Your own documents	NOT YET	The pipeline only needs paragraphs with ids; a folder importer is the obvious feature

A

Who did what

This project was built with AI agents. Here is where the line sits.

DAVID

- Chose the bounty and reframed it as a falsifiable question
- Set the rules the agents worked under: free models for every subagent, independent reviewers, preregistration before code, nothing tuned on shadow_dev
- Decided what shipped: the app, the default rerank depth, this manual

AGENTS

- Wrote most of the code, the question drafts and the review reports, under that protocol
- Each defect in Chapter 9 was caught by an agent reviewing another agent's work
- Every number in this book is read from a file in the repository, not from an agent's summary

B

Sources

- Round-1 results: `experiments/results/R1_ANALYSIS.md`, `experiments/results/retrieval/`
- Generator floor: `experiments/baselines/GENERATOR_FLOOR_REPORT.md`
- Thresholds: `experiments/comparability/CONTRACT.md` §4
- Depth sweep: `docs/manual/data/bench_dev.json`, `bench_shadow_dev.json` (`lmr bench`)
- Round-0 story: `docs/ROUND0_CLOSURE.md`, `docs/STATUS.md`
- Models: ettin-reranker-17m-v1, Qwen2.5-1.5B-Instruct GGUF, llama.cpp. Text: Wikipedia, CC BY-SA 4.0.

C

Changelog

1.0 · 2026-10-15

First edition: the `lmr` app (corpus rebuild from pinned revisions, top-*k* reranking, CLI and web UI) and the round-0/round-1 findings.