Is Hybrid Retrieval Worth It?
An Evaluation-Driven Comparison of Dense, Sparse and Hybrid RAG on Multi-Hop News Questions

Ahmad Tawil
Software Engineering, Braude College of Engineering, Karmiel
ahmadtawil.se@gmail.com

Technical report, September 2026

Abstract. A RAG system answers questions by first searching a document collection and then letting a language model write an answer from what was found. I built such a system over 609 news articles and compared three ways of doing the search step: *dense* (search by meaning), *sparse* (search by keywords, BM25) and *hybrid* (both, merged with Reciprocal Rank Fusion). All three were tested on the same 50 questions from the MultiHop-RAG benchmark, and the answers were scored automatically with RAGAS, a separate "judge" language model, plus a check on 10 trick questions that have no answer. Hybrid search found the most of the needed evidence (context recall 0.625 vs. 0.525 for dense), but with 40 scored questions the gain is not statistically conclusive. The bigger finding is that *search, not the language model, is the weak point*: the top 5 search results held only 17–26% of the evidence a question needed, so the model correctly said "Insufficient information." on more than half of the answerable questions. The report also shows where the time goes (about 0.05 s for search vs. about 10 s for the model) and lists concrete next experiments.

1. Introduction

Large language models (LLMs) can answer questions fluently, but they can also invent facts, and their knowledge stops at their training date. **Retrieval-Augmented Generation (RAG)** [1] is the common fix: before answering, the system searches a trusted document collection, gives the best matching passages to the LLM, and tells it to answer *only* from them.

A RAG system is only as good as its search step. If the right passage is not found, the LLM cannot use it. There are two classic ways to search text:

- **Sparse (keyword) search**, such as BM25, ranks passages by shared words. It is strong on names, numbers and dates, but misses paraphrases.
- **Dense (semantic) search** turns text into vectors of numbers ("embeddings") so that passages with similar *meaning* end up close together. It handles paraphrases, but can miss exact names or numbers.

Hybrid search runs both and merges the two ranked lists. It is often recommended, but it adds complexity. So the question of this project is simple:

> **Research question.** On multi-hop news questions, does hybrid search give the LLM better evidence and better answers than dense or sparse search alone?

A question is **multi-hop** when the answer needs facts from two or more different articles (for example, comparing what two newspapers said). This is the hard case for search.

What this report contributes:

1. A small but complete evaluation setup: a 50-question test set with known evidence, automatic scoring with RAGAS, and a separate test for trick questions.
2. A side-by-side comparison of dense, sparse and hybrid search with everything else fixed, including a statistical check.
3. An analysis showing that missing evidence, not the LLM, explains most of the low scores, with concrete examples and a list of next experiments.

2. Background

RAG. Lewis et al. [1] introduced the name and the idea of pairing a retriever with a generator. Gao et al. [2] survey the field and describe "advanced RAG" improvements such as better chunking, hybrid search and reranking. This project sits in that category.

BM25 [4, 3] scores a passage higher when it contains the query words, especially rare words, while not over-rewarding long passages that repeat a word many times.

Dense retrieval [5] uses a neural model to embed questions and passages. I used `bge-small-en-v1.5` [6], a small, free embedding model that produces 384 numbers per text. The BEIR benchmark [7] showed that neither dense nor BM25 wins everywhere, which is the main reason hybrid search is interesting.

Reciprocal Rank Fusion (RRF) [8] merges ranked lists using only positions, not scores. This matters because BM25 scores (e.g. 45.8) and dense similarity scores (e.g. 0.75) are on different scales. Each passage gets

$$\mathrm{RRF}(d) = \sum_{\text{lists}} \frac{1}{k + \mathrm{rank}(d)}, \qquad k = 60,$$

so a passage near the top of *both* lists wins. A passage missing from a list gets nothing from it.

RAGAS [9] scores RAG answers automatically by asking a judge LLM specific questions (defined in Section 4.2).

MultiHop-RAG [10] is a public benchmark of 609 news articles (late 2023) and 2,556 questions. Each question comes with the exact facts ("evidence") needed to answer it, which lets me check whether search found them.

3. The System

Figure 1 shows the whole pipeline. The top row answers questions; the bottom row runs the evaluation offline.

1. Chunking. Articles are too long to search as a whole, so each one is split into **chunks** of at most 512 characters, with 64 characters of overlap between neighbours so that a sentence cut at a boundary still appears complete in one chunk. This gave **17,653 chunks**. Each chunk gets a stable ID (a hash of its URL, position and text), so the same chunk always has the same ID.

2. Two indexes. All chunks are embedded and stored in Qdrant, a vector database (dense index), and also stored in a BM25 index (sparse index). Both are built from the exact same chunks, and the build fails if their ID lists differ, so the three search modes are always comparing the same material.

3. Search modes. *Dense* returns the chunks closest in meaning; *sparse* returns the best BM25 matches; *hybrid* takes the top 50 from each and merges them with RRF ($k = 60$, the standard default, not tuned). Each mode passes its top $k = 5$ chunks to the LLM.

4. Grounded answer. The LLM (`gpt-5-mini`) gets the question and the 5 chunks, each labelled with its ID. Its instructions are strict: use only the given text, cite chunk IDs, and if the answer is not there, reply exactly "**Insufficient information.**" The reply must follow a fixed JSON format (answer + list of cited IDs); a malformed reply raises an error instead of being shown, and any cited ID that was not among the 5 chunks is removed and logged.

5. Serving. A FastAPI web service exposes the pipeline (`/api/v1/query`), the saved benchmark results and a health check. The whole stack starts with one Docker command (Appendix A).

4. Experimental Setup

4.1 Test set

I sampled 50 questions from the 2,556 in MultiHop-RAG with a fixed random seed (42), so the set can be rebuilt exactly:

- **20 "direct"** questions (10 about timing, 10 comparisons between articles). 19 of the 20 have a yes/no-style answer.
- **20 multi-hop** questions that require combining facts, typically to identify a person or company.
- **10 trap** questions whose answer is *not* in the collection. The only correct reply is "Insufficient information."

For every answerable question I located the chunks that contain its evidence facts (exact text match, with a fuzzy-match fallback for 2 facts). All 120 evidence facts were found. Each answerable question needs **2 to 5 chunks**, usually from different articles.

I started with 25 questions but increased this to 50 because, with only 20 scored questions, a single question moves an average by 5 points. I did not go to 100 because of budget (about $9 per full run, from a student budget).

4.2 Metrics

The four RAGAS metrics are computed by a judge LLM, on a 0–1 scale, higher is better:

- **Faithfulness**: are the claims in the answer supported by the retrieved chunks? (Measures made-up content.)
- **Answer relevance**: does the answer actually address the question? A refusal scores 0.
- **Context precision**: are the useful chunks ranked near the top of the 5?
- **Context recall**: does the retrieved text contain the information in the reference answer? (Did search find what was needed?)

Three more measures need no judge:

- **Trap refusal**: out of the 10 trap questions, how many got exactly "Insufficient information."
- **False refusal**: out of the 40 answerable questions, how many were refused anyway.
- **Latency**: time per question, mean and 95th percentile (the time 95% of questions stay under).

Why traps are scored separately. RAGAS gives a refusal 0 for answer relevance, even when refusing is the correct behaviour. Averaging traps in would punish the system for being honest. So RAGAS averages use only the 40 answerable questions, and traps get their own exact-text check. (I considered telling the judge which questions are traps, but that would leak the answer to the judge.)

4.3 Models and cost

Generator: `gpt-5-mini`. Judge: `gpt-5.6-terra`, a different and stronger model, so the generator does not grade its own work. Search is fixed at $k = 5$. Each mode ran all 50 questions one at a time (150 generations, 0 errors), and the judge produced all 480 scores (40 questions $\times$ 4 metrics $\times$ 3 modes) with none missing. The judge cost about **$3.64** (measured from token counts) and generation roughly $0.25 (an estimate).

5. Results

Table 1 and Figure 2 give the main numbers.

Hybrid found the most evidence. Hybrid had the best context recall (0.625 vs. 0.525 dense and 0.550 sparse), the best context precision and the best answer relevance. This is the direction the research question expected.

Is the difference real? With only 40 questions, small gaps can be luck. I ran a **paired bootstrap**: resample the 40 questions with replacement 10,000 times, and each time compute the hybrid-minus-dense difference on the same questions. Result: **+0.100, 95% interval [0.000, +0.225]**. Hybrid was ahead in 94% of resamples; per question it scored higher on 5, lower on 1 and the same on 34. Against sparse the interval is [−0.050, +0.200], which includes zero. *In plain words: the evidence leans toward hybrid, but 40 questions are not enough to be sure.*

Faithfulness was lowest for hybrid (0.365). As Section 6.2 shows, averages here are dominated by refusals, so this should not be over-read.

Trick questions. All modes refused 8 or 9 of the 10 traps

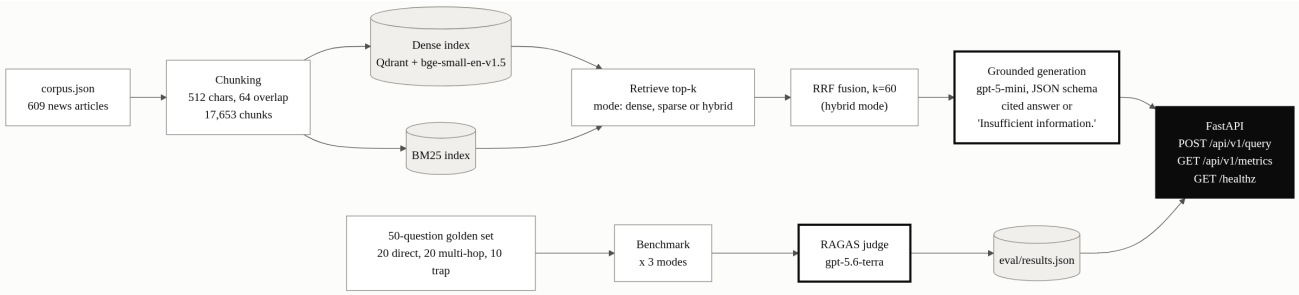

Figure 1: System overview. Top: documents are split into chunks and stored in two indexes; a question is searched in one of three modes and answered by the LLM from the top 5 chunks. Bottom: the 50-question test set is run through all three modes and scored by a separate judge model. Thick borders mark steps that call an LLM.

Mode	Faithfulness	Answer relevance	Context precision	Context recall	Trap refusal	Latency mean / p95
Dense	**0.427**	0.258	0.604	0.525	8/10	10.8 s / 21.1 s
Sparse	0.415	0.211	0.567	0.550	**9/10**	**10.0 s / 17.0 s**
Hybrid	0.365	**0.286**	**0.618**	**0.625**	**9/10**	12.1 s / 21.2 s

Table 1: Main results. RAGAS scores are averages over the 40 answerable questions (0–1, higher is better). Trap refusal is out of the 10 unanswerable questions. Latency covers all 50 questions (lower is better). Bold = best in column. $k = 5$, generator `gpt-5-mini`, judge `gpt-5.6-terra`.

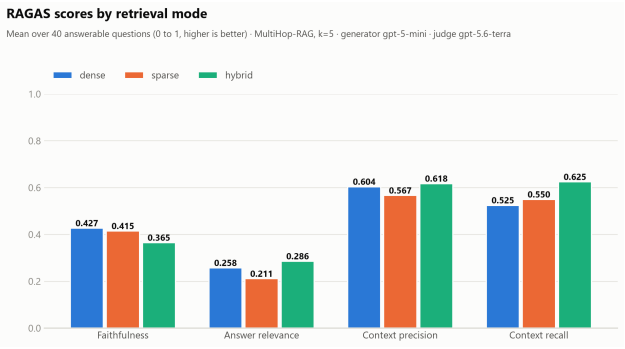

Figure 2: RAGAS scores per search mode (40 answerable questions).

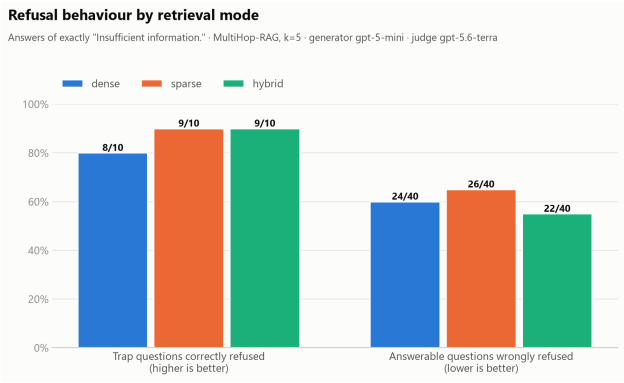

Figure 3: Refusals. Left: trap questions correctly refused (higher is better). Right: answerable questions wrongly refused (lower is better).

(Figure 3). A one-question difference is not a ranking.

The surprise: too many refusals. The model said "Insufficient information." on **22 to 26 of the 40 answerable questions** (55–65%). The next section explains why.

6. Analysis

6.1 Search, not the LLM, is the bottleneck

Because I know which chunks hold each question's evidence, I can check how many of them made it into the top 5, with no LLM involved (Figure 4).

On average the top 5 held only **17% (dense), 21% (sparse) and 26% (hybrid)** of the needed evidence chunks, and only **1 of 40** questions had *all* of its evidence retrieved. Figure 5 links this to the refusals: in hybrid mode, of the 22 refused answerable questions, 11 had none of their evidence, 11 had only part of it, and **none had all of it**. Across all three modes there was only one refusal where everything needed was actually retrieved (sparse).

So the refusals are mostly *correct* behaviour: the LLM followed its rule and did not guess when the evidence was missing. The problem is upstream, in search. Hybrid refused the least (22) because it found the most evidence.

6.2 Why the RAGAS averages look low

Splitting the scores by outcome makes the picture clear. On questions the model *answered*, scores were reasonable: faithfulness 0.59–0.69, answer relevance 0.60–0.64, context precision 0.71–0.88, context recall 0.75–1.00. On refused questions, answer relevance is exactly 0 by definition and faithfulness was only 0.18–0.27. Since more than half the questions were refused, they pull every average down.

One caution: the "answered" questions are a different set in each mode, so answered-only numbers must not be compared across modes. The fair comparison remains the 40-question average in Table 1.

6.3 A concrete failure: question q11

q11 asks whether a *Fortune* article and a *TechCrunch* article describe the Israel conflict in a certain way. In hybrid mode, all 5 retrieved chunks came from the TechCrunch article and none from Fortune. A comparison was impossible, so refusing was right. This is the typical multi-hop failure: **the top 5 fills up with chunks from one article**, and the second source never gets a place.

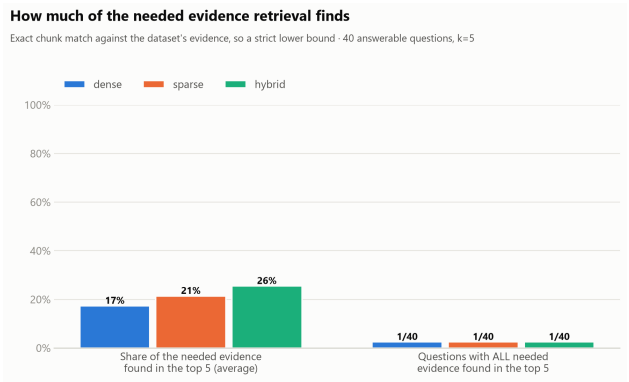

Figure 4: Share of the needed evidence chunks that search placed in the top 5. Only 1 of 40 questions had all its evidence found, in every mode.

6.4 When the model did not refuse a trap

The few non-refused traps are real hallucinations, and they share a pattern. Trap q47 asks which baseball position "represented by a single character" a baseball executive prioritized; all three modes gave an answer built around the number "5", with a citation. Trap q48 asks for the first letter of a new MacBook Pro feature; dense answered "M", taken from an article about Apple's M3 chips. In both cases the question looked answerable, the retrieved text contained something related, and the model produced a confident, cited, but unsupported answer. Citations alone do not prove grounding.

6.5 Where the time goes

Mean latency was 10–12 s per question (Figure 6). To find out why, I ran the real web service with the LLM replaced by a fake: the whole endpoint (search, fusion, building the response) took only **38 ms (dense), 50 ms (sparse) and 83 ms (hybrid)**. So more than 99% of the time is the LLM. `gpt-5-mini` is a "reasoning" model that thinks before answering; with `gpt-4o-mini` the same question took about 1 s. Hybrid's extra cost is real but tiny (about 45 ms more than dense).

6.6 A weakness of RRF seen during testing

For a long paraphrased question about the game *Cocoon*, dense search ranked the correct chunk #1, but BM25 did not have it in its top 50. Its RRF score was only $1/61 = 0.016$. A mediocre chunk ranked #44 by dense and #2 by BM25 scored $1/104 + 1/62 = 0.026$ and pushed the correct chunk out of the hybrid top 5. RRF rewards agreement between the two lists over one confident list. This is one reason hybrid is not always better, and it suggests tuning k or weighting the lists.

7. Limitations

- **Small sample.** 40 scored questions give wide confidence intervals; differences of 0.05–0.10 are within noise.
- **Narrow question types.** 19 of 20 direct questions are yes/no, and 9 of 20 multi-hop answers are the same person (Sam Bankman-Fried), so the set covers fewer topics than its size suggests.
- **No correctness metric.** None of the four RAGAS metrics compares the answer with the reference answer.
- **Judge bias.** Generator and judge are both OpenAI models, so some same-vendor bias is possible.
- **Not exactly repeatable.** `gpt-5-mini` and the judge do not allow setting temperature to 0, so a rerun gives slightly different numbers. The benchmark was run once.
- **Strict evidence check.** Only the exact evidence chunk counts as found; a different chunk with the same fact does not. The 17–26% figures are therefore a lower bound.
- **Default settings.** Chunk size, overlap, RRF k, the 50-candidate pool and the embedding model were all left at defaults.

8. Future Work

Each item below is a hypothesis to test, not a result.

Idea	Why it might help
Larger k (10, 20)	Multi-hop questions need 2–5 chunks from different articles; 5 slots rarely hold them all.
Diversify results (max chunks per article, or MMR [11])	Stops one article from filling the top 5 (the q11 failure).
Query decomposition	Split a two-source question into one search per source, then merge.
Recall@k on known chunks	Measures search alone, with no judge cost.
Cross-encoder reranker, tuned RRF	Re-scores candidates more carefully; fixes cases like Cocoon.
More questions, repeated runs	Tighter intervals and a measure of run-to-run spread.
Answer-correctness metric, cross-vendor judge	Checks the answer itself and reduces judge bias.
Stricter grounding check	A second pass to catch cited-but-unsupported answers (q47, q48).

The first two are the most promising, because the analysis shows that missing evidence is the main problem.

9. Conclusion

I built a complete, measured RAG pipeline and compared dense, sparse and hybrid search on the same multi-hop questions. Hybrid search found the most evidence and refused the fewest answerable questions, but with 40 questions its advantage is suggestive rather than proven. The clearest lesson came from measuring beyond the headline scores: the LLM behaved well, refusing when evidence was missing, while search delivered only about a fifth of the needed evidence. For multi-hop questions, improving *what reaches the model* (more and more varied results) is the next step, and this setup can now measure whether it works.

References

[1] P. Lewis et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *NeurIPS*, 2020. arXiv:2005.11401.

[2] Y. Gao et al. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997, 2023.

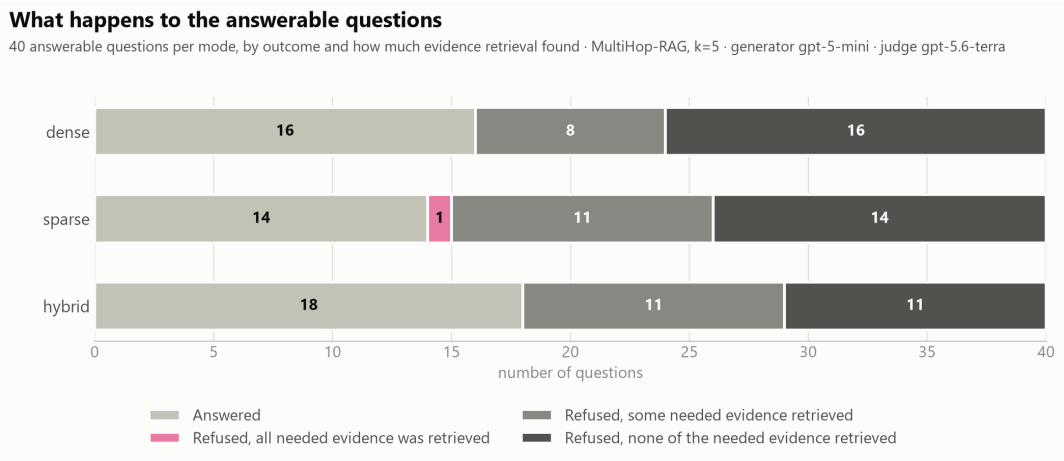

Figure 5: What happened to the 40 answerable questions in each mode, split by how much of the needed evidence search found. Almost every refusal happened when evidence was missing.

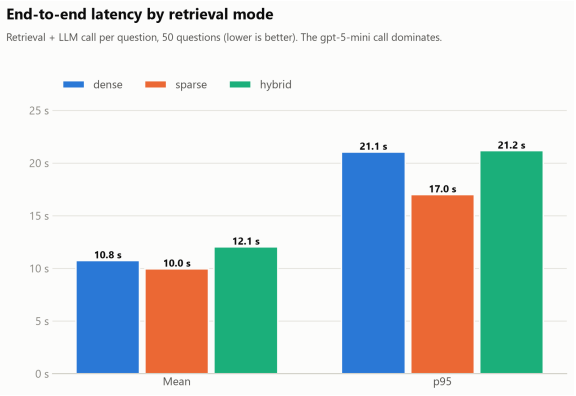

Figure 6: Latency per question, including the LLM call.

[3] C. D. Manning, P. Raghavan, H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[4] S. Robertson, H. Zaragoza. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in IR*, 2009.

[5] V. Karpukhin et al. Dense Passage Retrieval for Open-Domain Question Answering. *EMNLP*, 2020. arXiv:2004.04906.

[6] S. Xiao et al. C-Pack: Packaged Resources To Advance General Chinese Embedding. arXiv:2309.07597, 2023.

[7] N. Thakur et al. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. *NeurIPS Datasets and Benchmarks*, 2021. arXiv:2104.08663.

[8] G. V. Cormack, C. L. A. Clarke, S. Büttcher. Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods. *SIGIR*, 2009.

[9] S. Es et al. RAGAS: Automated Evaluation of Retrieval Augmented Generation. arXiv:2309.15217, 2023.

[10] Y. Tang, Y. Yang. MultiHop-RAG: Benchmarking Retrieval-Augmented Generation for Multi-Hop Queries. *COLM*, 2024. arXiv:2401.15391.

[11] J. Carbonell, J. Goldstein. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. *SIGIR*, 1998.

A. Reproducibility

Code: the `eval-hybrid-rag` repository (Python 3.13, dependencies locked with `uv`). **Data:** MultiHop-RAG (ODC-BY licence). **Test set:** 50 questions, seed 42, saved in `eval/test_dataset.json`. **Summary results:** `eval/results.json`.

- `docker compose up -build`: downloads data, builds both indexes, starts the API (about 20 minutes on an 8-core CPU from an empty start; under a minute afterwards).
- `uv run python -m eval.benchmark`: full benchmark (about $4, 30+ minutes).
- `uv run python -m eval.make_charts`: regenerates all figures.
- `uv run pytest`: 21 unit and API tests, no network or API key needed.

B. Engineering Notes

Real problems met while building the experiment, and how each was solved:

- **Text encoding on Windows.** Saving the dataset crashed on special characters (such as emoji) because Windows used a Hebrew codepage by default. Fixed by forcing UTF-8 everywhere.
- **A silent hang in the judge.** The judge model only accepts its default temperature, but RAGAS always sends a near-zero value. Every call was rejected and retried, so the first test hung for minutes. Fixed with a small wrapper that removes the temperature setting; retries and timeouts were also capped.
- **A broken dependency.** RAGAS failed to import because a newer `langchain-community` removed a module it needs, and it also needed Pillow without declaring it. Found by testing in a throwaway environment first, then pinning the working versions.
- **Keeping cost under control.** The benchmark records a failed item as an error and continues, instead of crashing halfway through a paid run. Judge cost was measured on 3 samples before the full run (about $0.03 per scored answer), which set the test-set size.
- **A smaller Docker image.** The first dependency lock

pulled 43 GPU packages (several GB) that a CPU server never uses. Switching Linux to CPU-only PyTorch and keeping evaluation tools out of the image gave a 2.64 GB image; the running API uses about 590 MB of memory.

- **Clear errors.** Without an API key, the query endpoint returned a bare "500 Internal Server Error". It now returns a 503 with a message that explains what is missing. A test from a completely clean copy of the repository confirmed the one-command start works.