# Answer Engine

A Small Reference Implementation for Citation-Grounded AI Answers

Luke F. Walton

Independent Researcher · lukefwalton.com

ORCID: 0009-0005-9263-1954

Technical note · v1.1 · June 2026

Software companion to *Building Answerable AI: Why Automation Needs Owned Error*

https://doi.org/10.5281/zenodo.20682307

## 1. Problem

A conversational assistant pointed at a body of work tends to blur things worth keeping separate. Retrieval, the use of private context, generation, citation, and refusal all happen behind one chat turn, and the seams between them are invisible at the surface. Retrieving and citing doesn't fix this: the citations are asserted rather than guaranteed, the choice to answer rather than decline stays the model's own, and nothing structurally keeps unpublished material out of the response. A chatbot that is right most of the time speaks *for* the author; what an archival setting wants is a system that speaks *from* a bounded record, under a contract held outside the model rather than left to its discretion.

This artifact operationalizes a design pattern, *framed automation*, developed in companion work on answerability and authored frames: hold the junctures at which an evaluative frame enters force, automate what merely executes between them, and make the unauthored move structurally inexpressible at the interface rather than caught after the fact [1–3]; the constructive pattern is stated in [4]. The relevant move is not to forbid the machinery but to keep the frame it must satisfy — what evidence may enter an answer, what must stay out, when to decline — outside the model, where it can be checked rather than trusted.

`answer-engine` is a reference implementation of that pattern: the evidence boundary, the citation contract, the refusal modes, and the regression suite are concrete enough to read, run, and check.

## 2. Artifact

`answer-engine` is a small TypeScript reference implementation for bounded, one-shot answer generation over a corpus of documents. It is a clone-and-run example repository, not an npm package, hosted service, or chatbot UI. Each query is a stateless transaction: a question goes in, and a cited answer or an explicit refusal comes out, with no conversation state, memory, or persona. The data path is `markdown → embed → retrieve → cited JSON`, and the implementation is deliberately small enough to read in one sitting.

The repository ships with a synthetic example corpus (fictional throughout, attributed to a placeholder author rather than a real person), so it runs out of the box. Default models are `text-embedding-3-large` for indexing and `gpt-4o-mini` for answering, configured in one file and swappable for any Responses-API model; retrieval is in-memory brute-force cosine similarity, which is adequate at teaching scale and whose shape does not change when swapped for a vector store. Code is licensed Apache-2.0.

## 3. Design contract

The system is built from four commitments, stated here as the contract the model output must satisfy.

**Two corpus layers.** Documents are split into *records* and *private notes*. A record is a published page: its title, canonical URL, summary, curated themes, and full body are all available to the model, because the body was already published. A private note is material that should be *searchable but never quotable*: it is embedded so retrieval can find it, but its text never reaches the model. Each note declares the public page it routes to and where the relevant moment lives.

**Retrieval finds both; assembly strips prose.** Both layers share one embedding index. Retrieval scores every item by cosine similarity with two conservative boosts (naming a work's title, and using a curated theme verbatim), so that author-maintained metadata can outrank raw similarity, and it drops anything below a score floor. An empty result is where refusal begins, before the model is called at all. Records and private notes are returned as two separate lists, because what happens to each differs: record bodies travel to the model, while private notes are reduced to *routing hints* carrying only a label, a locator, and a URL.

**A type-level no-leak boundary.** The routing-hint structure has no field for the note's text, so along the typed prompt-construction path there is nothing through which private prose could reach the model. This is the design's central instance of structural inexpressibility made runnable: the boundary is a property of the type's shape rather than a runtime guard a maintainer has to remember to write, so on that path the prohibited move is not merely discouraged but absent. The

prompt builder cannot leak a hint's body because the body does not exist in the object it receives.

**Four modes, enforced in schema and validator.** Every answer declares one of four modes that exactly partition the citation mix, which makes honesty checkable: `supported` (records plus hints), `partial` (records only), `related-material` (hints only), and `not-found` (no citations, empty answer). Three layers enforce this rather than trusting the model's self-report. A JSON schema constrains the shape; a validator rejects contract violations such as a `not-found` carrying prose or a sourced mode missing its citations; and a repair step snaps near-miss citations onto the exact retrieved pairs, deduplicates, and *re-derives the mode from the final citation mix*, so the model cannot claim `supported` while citing only hints. A final check confirms every citation is the exact identifier-and-URL pair of something actually retrieved. An invented source is an error, not a footnote.

A corollary the implementation treats as a rule: **retrieved is not cited.** Retrieved neighbors are candidates; the final citation list is the evidence. A user interface should render source cards from the final citations, and render none for a refusal, so that a decline never appears to be backed by the very sources the system declined to use.

## 4. Evaluation

Evaluation treats answerability as a regression target. The repository carries a fixed file of gold queries, each annotated with required behavior, and the set is constructed to exercise the contract rather than only accuracy: questions the system must answer from the public canon, questions it must *refuse* because no evidence clears the floor, and at least one question that must route to a private note without quoting it. The harness runs in two tiers: a cheap retrieval-only pass that uses a single batched embedding call, and a full pass that invokes the answer engine and checks the declared modes. An offline, deterministic test suite needs no API key.

The governing rule of the eval is that when a query fails, the fix goes into the corpus, the scoring, or the prompt — never into a special case for that question. A special-cased question makes the suite pass while making the system worse, and the prohibition is what keeps the gold set meaningful as a regression guard.

The eval harness supports targeted reruns: a full run can be scoped to specific query identifiers or driven from a prior run's report to rerun only the failures. This keeps the expensive answer-generating pass cheap to iterate on while leaving the full set as the release gate.

## 5. Limitations

This is a small teaching implementation, and its scope should not be overstated. It is **not a benchmark**: the gold set is a fixed regression suite over one synthetic corpus, not a graded comparison across systems, and the numbers it produces are pass/fail against a hand-authored contract, not a metric for ranking models. It is **not a production framework**: retrieval is in-memory and unchunked, there is no HTTP layer, rate limiting, caching, or persistence, and long documents are indexed whole rather than as passages. Quality depends on the corpus and the curated metadata; a thin or poorly-themed corpus degrades both retrieval and the honesty of refusals. The boundary the implementation guarantees is narrow but real: private text cannot enter the prompt. Everything above that line, including answer quality and the calibration of the score floor, remains an empirical matter left to the deployer.

What the artifact is meant to be is a reproducible pattern: the smallest version of an answer contract that states what evidence may enter a prompt, what must stay out, how citations are grounded, and when the system must decline — with each of those enforced outside the model, by types, schema, validators, and tests, so the answer can be checked rather than trusted.

## 6. Scope: scale, ownership, and what a bounded implementation establishes

The system is bounded by construction: it answers over a corpus whose edge is authored and owned. That edge is the condition under which an answer engine can be answerable at all. Unbounded synthesis over an unauthored corpus is the case this architecture is defined against, not a larger instance of what it does. Smallness is different. The corpus is small for legibility; the pattern does not require smallness. Size and ownership are independent. A two-document system that concatenates both documents into the prompt and lets the model improvise is unowned, because no contract governs what may enter an answer or when it must decline, even though it is tiny; a large system with a documented admission contract, refusal policy, grounding check, and named maintainer is owned, even though it is large. What grows with scale is the cost of operating the pattern, not its applicability.

The pattern does not require a discrete instant at which ranking or compression begins to set what counts. Authority attaches where synthesis is admitted into force: no answer reaches a reader until it has resolved onto retrieved evidence as a citation or returned as a refusal. At scale, that surface is an admission policy authored in advance and revised after failure, not a human inspecting every output.

Admission owns soundness, not completeness. A gate can certify that an answer is grounded or

honestly refused; it cannot certify that retrieval surfaced everything it should have. A source below the score floor is absent, and absence is what a gate cannot catch. What can be owned upstream is the authored boundary around retrieval: the corpus edge, the score floor, the candidate rules, and the metadata by which some sources are made easier to find. But which in-corpus source the embedding geometry surfaces for a given query is not itself an authored judgment. The gold suite tests enumerated recall: a known-relevant source must surface, and a due refusal must fire. What remains irreducible is recall for cases not yet named by the suite.

The implementation therefore establishes a narrow claim: these junctures can be control surfaces, not labels. The type boundary exposes no private-prose field; the validator rejects ungrounded citations; the floor refuses; the suite catches named regressions. It does not establish that admission-surface ownership discharges the moral account owed to the reader, nor that public-scale operation is economically bearable. The companion papers carry the first argument; the second remains empirical. Future implementations could expose near-misses or below-floor candidates to make recall failures more contestable. This version does not.

## 7. Scaling the pattern: empirical notes from deployment

Section 6 holds that what grows with scale is the cost of operating the pattern, not its applicability, and leaves the economic question empirical. This section reports how that cost was met in one deployment — the production system behind "Ask the Archive," which runs the same pattern over a corpus several orders of magnitude larger than the bundled example — and its subject is not the size of the savings but the way each saving was authorized.

A disclosure belongs first. The production corpus is private — you can read these figures but cannot run it, so they are real, not reproducible. It indexes 10,554 embedding vectors: 9,777 private passages — the 210 podcast episodes (~123 hours), chunked into overlapping windows — and 777 public records, the published pages (380 songs, plus albums, essays, letters, and per-episode notes). By words the split is starker — roughly 186,000 public against 1.75 million private — and those 1.75 million private words are searchable but never quotable, reaching the model only as routing hints: the Section 3 boundary holding at scale. What is public and checkable is the method — every cost-reducing change was admitted or rejected by the same gold suite that governs grounding and refusal (`eval/gold.yaml`), under Section 4's rule against special-casing a failing query.

The cost concentrates almost entirely in one object — the embedding index, in both its in-memory footprint and the latency of making it available to a stateless serving instance — and three levers reduce it, none of which alters the contract of Section 3.

**Embedding dimension.** The embedding's dimensionality is a tunable cost: Matryoshka repre-

sentation learning lets a vector be truncated and renormalized with graceful quality decay, so the index can be rebuilt at a lower dimensionality (roughly threefold smaller at 1024, for a small quality cost) and the query embedded to match. This deployment has not spent that lever — all 10,554 vectors are the full 3072 dimensions — but it belongs here because, were it taken, the choice would be made against the gold suite rather than by intuition, and because the homogeneity it requires is itself a checked invariant: the stored (model, dimensions) pair is the index's identity, the query is embedded at that same dimensionality, and a store that mixed dimensions fails fast rather than comparing incomparable vectors. A cost lever held under the gate's authority is, in that exact sense, answerable.

**Wire format.** This cost is absent from the bounded implementation, where the index is read in process; it appears only once the index must be transported to a stateless serving instance, where its serialized size becomes a cold-start latency paid by every cold instance. Embedding vectors dominate that serialized form, and quantizing the published vectors to one signed byte per dimension — symmetric, with a per-vector scale — reduces each vector by roughly an order of magnitude; here it took the published bundle from ~616 MB to ~53 MB, about twelvefold — a little under the per-vector factor because the routing metadata it also carries does not quantize. Two facts make this admissible, and they differ in kind. The first is exact: cosine similarity normalizes by vector norm, so a positive per-vector scale cancels from the comparison entirely — the ranking is invariant to it as a matter of algebra, and one can score against the quantized bytes without restoring the scale at all. The second is not exact: int8 rounding perturbs each vector's direction and can reorder near-ties, so its harmlessness is not proven but measured — the quantized index is checked against the gold suite and accepted only if it holds (here, rank correlation above 0.99, then a passing eval). That split is the same epistemic move the rest of the note makes: what can be guaranteed is guaranteed, here by algebra rather than by a type, and what cannot is gated by the eval rather than asserted. The full-precision vectors remain the source of truth in memory and on disk, so quantization is a transport encoding rather than a lossy store, and the wire format is versioned so that a mismatch between code and stored blob fails loudly instead of misreading bytes.

**Corpus geometry.** The first lever the repository's own roadmap named — chunking long documents into overlapping windows — proves to be the dominant driver of index size at scale, so this is follow-through on stated future work rather than a fresh claim. Indexing passages rather than whole documents, necessary once a document is long enough that a single embedding dilutes its topical center, makes index size scale with passage count: linearly in the corpus's decomposed size rather than in its document count. The granularity of the passage window trades index size against retrieval precision and is, once again, chosen against the gold suite rather than assumed.

That cut is the least of what matters here. What matters is that the gate which owns soundness also adjudicated every cost choice made in its name: answerability here governs not just what the

system may answer but how it may be tuned. These are notes from a single system, not a general cost model; a controlled study across systems and corpora remains future work.

---

## Artifact statement

Source code, the example corpus, and the gold-query eval are available at https://github.com/lukefwalton/answer-engine under the Apache-2.0 license, archived on Zenodo at https://doi.org/10.5281/zenodo.20676773 (concept DOI; resolves to the latest version). This technical note is archived on Zenodo at https://doi.org/10.5281/zenodo.20686053 (concept DOI; resolves to the latest version). The repository is the teaching-sized version of the engine behind "Ask the Archive" at lukefwalton.com.

## Disclosures

**Competing interests.** The author is the founder of Surmado, Inc., which builds managed AI systems of the kind this pattern describes; the contract this note states binds the author's own work. The note is written in a personal capacity, without employer involvement and without funding.

**Generative AI.** Generative AI systems (ChatGPT 5.5, OpenAI; Claude Opus 4.8 and Claude Fable 5, Anthropic; Gemini 3.5, Google) were used for drafting assistance, editorial review, and reference checking under the author's direction. The design, the implementation, and the final text are the author's, and the author is answerable for the work.



## References

[1] L. F. Walton, "The Decision No One Authored: The Answerability Gap in Generative AI," preprint, v1.4, Zenodo, 2026. https://doi.org/10.5281/zenodo.20622946

[2] L. F. Walton, "The Captured Oracle: Authorship and Agency in the Ethics of Answer-Engine Optimization," preprint, Zenodo, 2026. https://doi.org/10.5281/zenodo.20676328

[3] L. F. Walton, "The Invariant of Answerability," working paper, Zenodo, 2026. https://doi.org/10.5281/zenodo.20606493

[4] L. F. Walton, "Building Answerable AI: Why Automation Needs Owned Error," working paper, Zenodo, 2026. https://doi.org/10.5281/zenodo.20682307