Federal Regulation RAG — Chasing Accuracy in a High-Stakes Domain

A product manager's case study in evaluation-driven RAG tuning

Brad Hinkel · April 2026 · regs.bradhinkel.com

Summary

This project built and deployed a production retrieval-augmented generation (RAG) system over the U.S. Code of Federal Regulations — specifically Titles 7 (Agriculture), 21 (Food and Drugs), and 42 (Public Health), totaling 85,351 regulatory sections. Each query returns three outputs: a plain-English explanation, a formal legal-language synthesis with verbatim quotes from the source text, and a structured list of CFR citations (Title / Part / Section). The live system runs at regs.bradhinkel.com.

Unlike the fantasy-lore domain of the prior Sword Coast / D&D projects in this portfolio, regulations are a domain where hallucination is not merely aesthetic — it's a liability issue. The entire product thesis rests on whether the system can reliably ground its claims in the retrieved text. That reframed the work: the question wasn't "does this feel like a good answer," it was "can we prove, at inference time, that this answer is grounded in what the retriever surfaced?"

Eleven evaluation experiments later, three findings carried most of the weight:

1. **Retrieval quality was already strong before the harness showed it.** A metric audit revealed that MRR had been under-reported by roughly 2.5× due to three subtle bugs — a plot twist that reframed every subsequent tuning decision.
2. **Section-level chunking, guided by the CFR's own structural hierarchy (DIV8 sections), beat every embedding and retrieval sophistication stacked against it** — including a law-domain-tuned embedding model, hybrid BM25 retrieval, hierarchical reconstruction, and HyDE query expansion.
3. **An inference-time confidence signal — a linear combination of retrieval similarity and citation-coverage — provides a reliable "I don't know" flag without requiring a judge-LLM call.** That's the PM-distinctive contribution of this project. It's also the feature most worth extending.

The final production configuration reaches MRR 0.858, NDCG 0.880, faithfulness 0.729, legal accuracy 0.734, and citation accuracy 0.602 at a median end-to-end latency around 8 seconds per query on Claude Haiku 4.5. The confidence signal correctly flags 100% of the "not found" cases in the eval set with zero false positives.

The Problem

Federal regulations are the archetype of a domain where an LLM needs help. The CFR runs about 200,000 pages. Natural-language queries like "what are the labeling requirements for organic produce" don't map cleanly to a regulatory text that was written by lawyers for other lawyers, in a controlled vocabulary that rarely aligns with how a non-lawyer would phrase a question.

A user — in practice, a small business owner, a compliance analyst, a journalist, or a curious citizen — has two choices today: wade through multi-level hierarchies of Title → Part → Subpart → Section on ecfr.gov, or ask a general-purpose chatbot and hope it isn't confabulating. Both are bad options. The first is a time sink. The second is a liability: regulatory text is authoritative, and an LLM that paraphrases "may" as "must" has materially changed the meaning of the law.

What the user actually needs is the regulatory text itself, accurately retrieved and verbatim-quoted, with citations precise enough to verify. The plain-English summary is a courtesy. The legal-language output and the structured citations are the product.

A RAG pipeline is the obvious shape for this: retrieve the relevant sections, condition the LLM on them, emit citations that map back to the source. But "obvious shape" is where the easy part ends. The design space — chunking strategy, embedding model, retrieval mode, top-k, reranking, generation strategy — is wide, and the consequences of each decision interact. Without a rigorous evaluation framework, you're guessing.

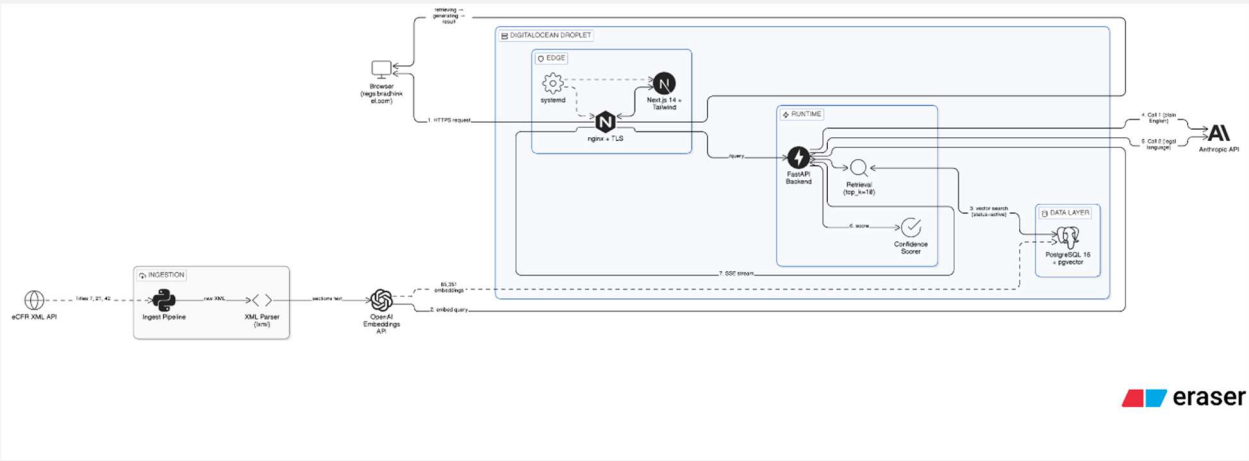
The System in Brief

The architecture is, deliberately, the same unremarkable small-to-mid-scale RAG pattern I used for the Sword Coast and D&D projects in this portfolio. The point of reusing the shape is to let the tuning be the interesting part.

- **Corpus ingest** — The eCFR XML API is fetched per title, parsed into a hierarchy (DIV1 through DIV8), and normalized. Each DIV8 (SECTION) becomes one chunk, with metadata carrying title, part, subpart, section number, section heading, agency, and CFR reference. 85,351 sections total.
- **Embedding** — OpenAI text-embedding-3-small (1536 dimensions). Section-level chunks.
- **Vector store** — PostgreSQL 16 + pgvector on a 4GB DigitalOcean droplet.
- **Retrieval** — Pure vector cosine similarity, top-k = 10. All queries filter status = 'active', enforced in the query layer not per-call-site (this matters for the Phase 8 differential-refresh work not covered here).

- **Generation** — Two-call sequential strategy on Claude Haiku 4.5. Call 1 writes the plain-English answer. Call 2 writes the legal-language answer with verbatim quotes, conditioned on both the retrieved context and the Call-1 summary.
- **API** — FastAPI backend streams status events over Server-Sent Events: retrieving → generating → result. Confidence data is computed inline and returned in the result event.
- **Frontend** — Next.js 14 / Tailwind, a single query form with tabbed outputs for Plain English, Legal Language, and CFR Citations, plus a confidence tier badge.
- **Production** — nginx reverse proxy, Let's Encrypt TLS, HSTS header, systemd units on the same DigitalOcean droplet that hosts the D&D project.

System Architecture Diagram



The interesting decisions were not "which components" but "which configuration of those components actually works for regulations." That question needed a harness.

The Evaluation Framework

I built a Python evaluation harness modeled on the one I used for the D&D project, with significant revisions specific to this domain. It runs a dataset of 60 regulatory questions — each tagged with a ground-truth CFR section reference and a human-written answer — against any given configuration and reports:

- **Retrieval metrics**: Precision@k, Recall@k, MRR, NDCG@k
- **Generation metrics**: faithfulness, answer relevancy, legal accuracy, citation accuracy, answer completeness — scored by a Haiku-based LLM judge with a fixed rubric

- **Operational metrics**: embed latency, retrieve latency, generation latency, end-to-end latency, input/output tokens

Each experiment isolates a single variable so the impact can be attributed cleanly. Each configuration is described in a YAML file and run via python eval/src/evaluate.py --config eval/configs/<name>.yaml. Results persist to JSON so phase-to-phase comparison is a trivial diff.

The discipline of changing one thing at a time is more important than any single result. It's what lets a PM — or a future self, or a reviewer — answer "why did you choose top_k=10?" with a number instead of a shrug.

What Moved the Needle — and What Didn't

Eleven experiments, organized by what they attempted and what they produced.

Wins

Sequential generation (Experiment 1). The first decision was the generation strategy: one LLM call that produces all three outputs as structured JSON, or two calls — plain-English first, then legal-language conditioned on the plain-English summary? Sequential won on every quality metric: faithfulness +4.5 points, legal accuracy +6.7, citation accuracy +12.9. It cost roughly 2× latency (2.5 s → 4.9 s), which for a product that runs at conversational pace was acceptable. Sequential became the standard for every subsequent experiment.

Top-k = 10 (Experiment 2). The single highest-impact change in the evaluation. Moving from top_k=6 to top_k=10 improved recall by 22 percentage points (0.683 → 0.833), NDCG by 10%, and every generation metric, at roughly 13% more tokens per query and no MRR cost. The intuition is obvious in retrospect: regulatory questions frequently span multiple sections (a "what are the labeling requirements for organic products" question touches § 205.300, § 205.301, § 205.303 simultaneously), and under-sampling the retrieval set starves the generator. It's the kind of change a "vibe check" loop would never catch — the top-6 answers looked fine, but were missing meaningful grounding.

Section-level chunking at the CFR's own structural boundary (re-ingestion). I'll give this its own section below because it's the largest structural decision of the project.

Neutral — Not Worth the Cost

Sonnet vs. Haiku (Experiment 3). Sonnet produced +0.9 faithfulness at 3.6× latency (5.6 s → 20.1 s) and ~15% more tokens. Within measurement noise. Haiku stayed.

Hybrid retrieval (Experiment 4). BM25 + vector via reciprocal rank fusion gave +6% MRR with identical recall and precision. Neither helped nor hurt meaningfully. Vector retained for simplicity.

Negative Results — Directly Harmful

Query rewriting (Experiment 5). I expected this to help. The intuition is that a plain-English question like "can I sell raw milk" probably doesn't hit the vocabulary the regulatory text uses ("unpasteurized fluid milk products"), and an LLM rewrite would bridge that gap. It didn't. MRR dropped 25% on Haiku, 21% on Sonnet. The honest lesson: regulatory text uses a precise, controlled vocabulary. An expanded query drifts toward approximate synonyms that produce false-positive matches in the vector space. In a domain where the source vocabulary is already carefully chosen, query rewriting is not expansion — it's drift. Disabled.

HyDE query expansion (Experiment 11b, exploratory). A variant of the above — generating a hypothetical regulatory passage before embedding, hoping that the synthetic CFR-style text would better match real CFR text in the embedding space. With section-level chunks already bridging the vocabulary gap, HyDE degraded MRR marginally (0.858 → 0.842). Interesting in other domains; unnecessary here.

The Big Bet That Didn't Pay Off — voyage-law-2 + Paragraph Chunks

This was the experiment I most expected to move the final needle, and it didn't.

voyage-law-2 is a 1024-dimensional embedding model from Voyage AI, fine-tuned on legal text. On paper, this is exactly the domain alignment a federal-regulation RAG should want. Combined with paragraph-level chunks — finer-grained retrieval targets that could surface the specific subparagraph (a)(1)(i) that answers a question — the hypothesis was that domain-specialized embeddings would outperform a general-purpose embedding model, and finer chunks would give the retriever more specific hit targets.

The corpus exploded from 83,448 section-level chunks to 223,918 paragraph chunks. Re-embedding that at Voyage's pricing cost a few dollars in API spend. (This is a normal line item in this kind of work — if you're running three or four serious retrieval experiments, you will spend on the order of low double-digit dollars in API costs, and that's healthy: it's the cost of knowing, not guessing.)

The result was genuinely instructive. Recall went up (0.850). MRR and NDCG went down. At top_k=10, the retriever was finding the right sections — but surfacing three or four paragraph chunks from the same section ahead of the chunks that answered the specific question. This is exactly the failure mode paragraph chunking was supposed to cure, and instead paragraph chunking created it.

Raising top_k to 25 pushed retrieval metrics to their best numbers of the whole study (NDCG 0.653, +25.6% over baseline) — and generation metrics fell. The retriever was excellent. The generator couldn't use what it was given. This is the "lost in the middle" problem from the long-context literature, amplified here by a specific regulatory pattern: a paragraph (a) defining a rule, (b) listing exceptions, and (c)

defining terms are semantically interdependent — separating them into independent retrieval targets fragments the meaning.

The lesson: a domain-specialized embedding model didn't compensate for a chunking strategy that fought the document's natural structure. Voyage-law-2 is a fine model; this wasn't a fair test of it. The failure was the paragraph chunking, which the voyage choice had implicitly locked in.

The Hierarchical-Retrieval Detour

I spent an experiment budget on trying to rescue the paragraph-chunk approach via hierarchical retrieval — retrieve paragraphs to identify relevant sections, then assemble the full section text as the generation context. Two variants, both failed:

- **Unbounded assembly:** Context tokens ballooned 6.4× over flat retrieval. The LLM drowned. Every generation metric collapsed.
- **Bounded assembly (6 sections cap, 4,000 chars each):** Stayed in budget but truncated answers mid-thought. Citation accuracy rose marginally (0.330 → 0.429) as truncation eased; faithfulness never recovered to flat-paragraph levels.

There was no section-cap value that satisfied both constraints simultaneously. Hierarchical retrieval is intellectually appealing but, for CFR text specifically, it traded a retrieval ceiling for a generation ceiling. This is where I stopped trying to rescue paragraph chunking and went back to the drawing board.

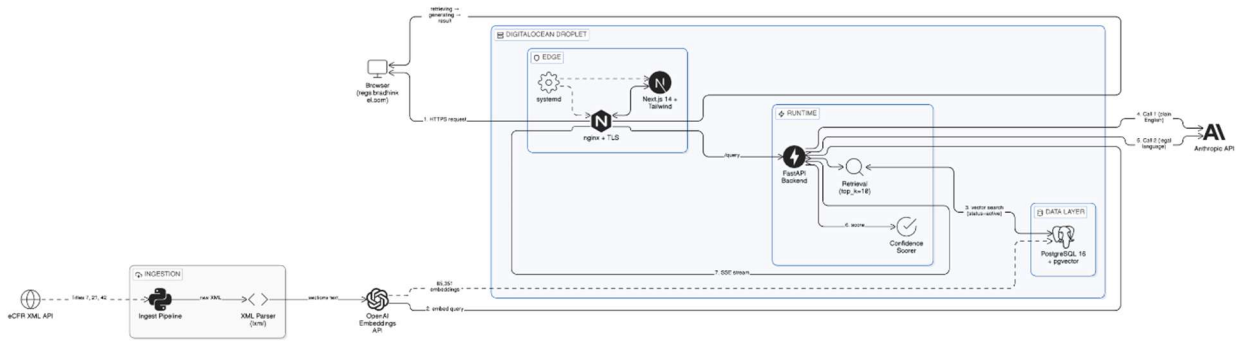
The Plot Twist: Auditing the Metrics

Before spending more API budget on experiments, I stopped and audited the metric code itself. Three bugs surfaced:

4. **Sub-paragraph notation false negatives.** Ground-truth references were written as § 205.301(a)(1); chunk metadata stored only the base section § 205.301. String matching flagged 23 out of 43 apparent retrieval "misses" as perfect hits in disguise.
5. **Chunk-text false positives.** The relevance check was also matching against the full chunk text. Regulatory text is dense with cross-references like "see § 135.110 for definitions," so matching against chunk text credited the retriever for every chunk that mentioned the target section, regardless of whether the retriever had actually found it. Corrected to match only against citation metadata.
6. **Duplicate-section inflation.** In the paragraph corpus, multiple paragraphs from the same ground-truth section all counted as separate hits, pushing NDCG above 1.0 — physically impossible. Corrected by deduplicating: each ground-truth section counts as at most one relevant hit, no matter how many paragraph chunks from it appear.

After the audit, MRR on the baseline configuration jumped from 0.275 to 0.710. NDCG from 0.520 to 0.720. Retrieval quality was already roughly 2.5× better than the pre-fix numbers had suggested.

This is an uncomfortable, honest result, and it's also the most PM-useful single finding in the project. For several weeks I'd been tuning for what I thought was a retrieval problem. It had actually been a measurement problem. The fancy embeddings, the hybrid retrieval, the hierarchical rescue attempts — these were all solutions to a retrieval ceiling that the buggy metrics had invented. Every evaluation harness needs an adversarial sanity pass before tuning decisions get stacked on top of it. I didn't do one early enough; the project cost that mistake several experiment budgets.



eraser

The Re-Ingestion That Actually Fixed Things

With metrics trustworthy, the pattern clarified: the generator wasn't starved for retrieval, it was starved for coherent context. Paragraph chunks had been fragmenting the regulatory meaning that lives at the section level. The natural unit of regulatory meaning is the DIV8 SECTION — the CFR's own drafters had already decided where the meaning boundaries are.

I re-ingested Titles 7, 21, and 42 with text-embedding-3-small (1536 dims, cheaper than voyage, no domain fine-tuning) and section-level chunks. 85,351 sections. Top_k stayed at 10. Sequential generation stayed. Everything else stayed.

The numbers on this configuration are the production numbers:

Metric	Paragraph + voyage-law-2	Section + text-3-small
MRR	0.710	**0.858**
NDCG@k	0.720	**0.880**
Faithfulness	0.533	**0.729**
Legal accuracy	0.569	**0.734**
Citation accuracy	0.354	**0.602**

Citation accuracy — the metric that most directly measures "is the answer grounded in real regulations" — moved from 0.354 to 0.602. That is the needle that mattered.

The architectural lesson is worth stating plainly: when the source corpus has been carefully structured by its authors, the chunking strategy should honor that structure. "Honor the document's own structure" sounds like common sense until you have a domain-specialized embedding model in hand and start reaching for it before trying the boring option first. A general-purpose embedding model aligned to the document's natural boundaries outperformed a legal-domain-specialized embedding model operating on contrived paragraph boundaries — by 20 points of faithfulness and 25 points of citation accuracy.

Adding an Inference-Time Confidence Signal

With the retrieval and generation stacks tuned, one problem remained: the system occasionally produced plausible answers that weren't actually grounded in the retrieved context. This is the hallucination-risk tail — low-frequency, high-consequence. The LLM judge in the evaluation framework catches it at test time, but at inference time, the user sees no signal distinguishing a reliable answer from a shaky one.

LLM-as-judge at inference time is expensive (another LLM call per user query) and slow (doubles latency). The product needed something cheaper.

What I Built

A two-component confidence score, computed without any additional LLM call:

- **Retrieval score** = average cosine similarity of the top-3 retrieved chunks. Proxy for "did the retriever find semantically relevant content."

- **Citation coverage** = fraction of CFR § references mentioned in the generated text that actually appear in the retrieved chunk set. Proxy for "did the generator ground its claims in what the retriever gave it."

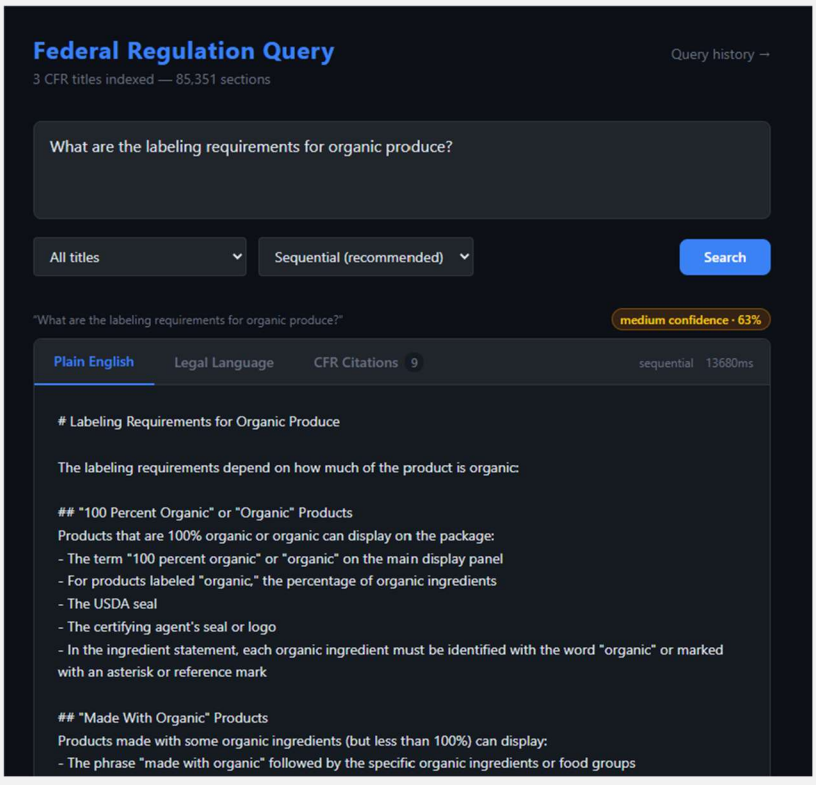
Composite: **0.35 × retrieval_score + 0.65 × citation_coverage**.

Tiers:

- **high** ≥ 0.75 — strong retrieval and grounded citations; treat as reliable
- **medium** 0.50–0.74 — useful but verify for high-stakes decisions
- **low** < 0.50 — weak retrieval or ungrounded citations; treat with caution
- **not_found** — no relevant content retrieved; explicit coverage gap

The heavier weight on citation_coverage reflects a specific product intuition: a generated answer that cites sections not in the retrieved context is either drawing on LLM training memory (hallucination risk) or quoting cross-references verbatim from regulatory text. Either way, the claim is unverifiable from the retrieved evidence alone — which is the exact thing a regulatory-RAG user needs to know.

Sample Query – Plain English Result



What the Calibration Data Actually Showed (Warts and All)

On the 60-question evaluation set, the distribution came out reasonable: high 71.7%, medium 16.7%, low 5%, not_found 6.7%. Average composite score 0.763.

The calibration data against the LLM judge's faithfulness score, however, was uncomfortable:

Tier	n	Avg faithfulness	Avg legal accuracy
high	43	0.759	0.762
medium	10	0.855	0.867
low	3	0.850	0.867
not_found	4	0.000	0.000

The not_found tier is perfectly calibrated — every answer flagged not_found had faithfulness exactly 0.000 (by construction: the system produced no content to be faithful about). This is the most actionable single piece of the signal, and in practice it's worth deploying on its own even if nothing else shipped.

But the medium and low tiers scored higher on faithfulness than the high tier. That's counterintuitive, and the honest reading is that it's primarily a small-sample problem (n=10 and n=3 don't support strong inference) combined with a specific pathology of the citation_coverage component: it penalizes answers that paraphrase the regulatory content faithfully without dropping inline § 205.301-style references into the text. Those answers are often perfectly grounded — they just don't cite inline. The formula sees no inline citations, counts citation_coverage near zero, and buckets the answer as medium or low. The judge — which evaluates grounding semantically, not by citation pattern — sees a faithful answer.

So: the formula's ordering is wrong for high vs medium in small samples. Its weighting probably over-emphasizes inline citations at the expense of semantic grounding. And its architecture — cheap, inference-time, no judge call — is exactly the right starting point for a signal that wants to live inside every user-facing query response.

What the Signal Reliably Does Today, vs. What It Doesn't

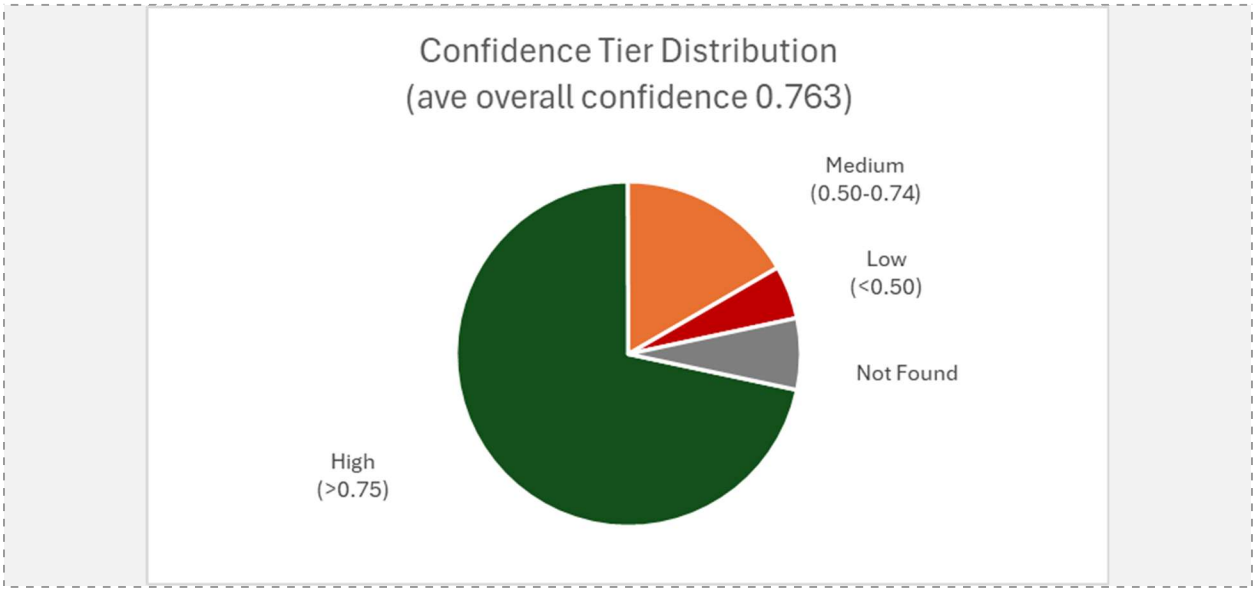
Reliable, ship-as-is:

- **not_found detection.** 100% of not_found flags in the eval set corresponded to answers with zero faithfulness. Users see a clean "no relevant regulations were found for this query" message instead of a confabulated answer.

- **Aggregate monitoring.** Average confidence across rolling windows of production traffic is a reasonable proxy for system health. A drop in average confidence is a signal worth paging on, even if no individual score is trustworthy at per-answer granularity.

Not reliable yet, don't present as calibrated probabilities:

- **Per-answer high/medium/low user labels.** The tier ordering doesn't hold up in the current sample. Showing a user "this answer scored 72%" would be a false guarantee.



What I'd Do Next With It

Three moves, in increasing effort:

7. **Expand the test set to 200+ questions.** The biggest source of noise in the calibration table is sample size. A dataset at 3–4× the current size would tell us whether the counterintuitive tier ordering is a sampling artifact or a genuine formula flaw. This is by far the highest-leverage next step.
8. **Reweight the composite.** The 0.35 / 0.65 split was an engineering guess, not a learned value. With a larger validation set, a small grid search over weightings — including adding a third component for retrieval concentration (how tightly clustered are the top similarity scores) — would likely produce a meaningfully better ordering.
9. **Replace the citation-coverage regex with a semantic grounding check.** The current implementation rewards inline CFR references in the generated text. A better signal would check whether each factual claim in the answer is semantically supported by at least one

retrieved chunk — which is a small additional prompt, not a full judge call. More expensive than the current signal, still cheaper than a full judge.

The broader product point: inference-time quality signals are a real and underused product feature in LLM applications. The conventional framing is that quality is a test-time concern — eval harness, LLM judge, etc. But users care about "is this answer good," not "is the average answer good." A cheap, honest per-query quality signal — even one that only reliably surfaces the not_found case — is more valuable than a better aggregate eval score.

Sample Query — Legal Result

Results Snapshot

Production Configuration

Component	Value
Corpus	CFR Titles 7, 21, 42 — 85,351 sections (DIV8)
Embedding	OpenAI text-embedding-3-small, 1536 dims
Vector store	PostgreSQL 16 + pgvector (cosine), 796 MB on disk
Retrieval	Pure vector, top_k=10, status = 'active' filter
Generation	Claude Haiku 4.5, sequential two-call strategy
Confidence	Retrieval score + citation coverage, 4 tiers
Frontend	Next.js 14, SSE streaming
Backend	FastAPI, uvicorn, asyncpg
Infrastructure	DigitalOcean 2 vCPU / 4GB droplet, nginx, Let's Encrypt, systemd

Final Metrics

- MRR: **0.858**
- NDCG@10: **0.880**
- Faithfulness: **0.729**
- Legal accuracy: **0.734**
- Citation accuracy: **0.602**
- Average confidence: **0.763** (71.7% high, 16.7% medium, 5% low, 6.7% not_found)
- Median end-to-end latency: ~8 seconds per query
- Approximate cost: fractions of a cent per query at current Haiku pricing

Cost to Build

The API-spend line item across the whole evaluation process was roughly a low double-digit dollar amount — most of it in re-embedding the corpus across the voyage and OpenAI experiments. That's a healthy number for this kind of work. The cost of not running those experiments would have been not knowing which configuration to ship, which is a much worse problem.

Lessons Learned

Audit the harness before tuning on it. The metric audit was the most consequential finding in the project, and the one I should have done earliest. It retroactively invalidated several tuning decisions and a non-trivial amount of experiment budget. Every evaluation framework needs an adversarial sanity pass — construct cases where the expected answer is obvious and confirm the metrics reward them — before real tuning starts. I didn't, and I paid for it.

Honor the document's natural structure. Before reaching for a domain-specialized embedding model, check whether the document's authors have already solved the chunking problem. The CFR is organized in a strict hierarchy (Title → Subtitle → Chapter → Subchapter → Part → Subpart → Section), and the SECTION (DIV8) is the natural unit of regulatory meaning. A general-purpose embedding model operating on that natural boundary beat a legal-domain-specialized model operating on engineered paragraph chunks — by every metric that mattered. The corollary: a domain-specialized component can't compensate for a structural mismatch upstream of it.

Inference-time quality signals are a PM product feature, not just an eval concern. The conventional LLM-PM instinct is to push quality into test-time: bigger eval sets, better judges, tighter prompts. Those matter. But a cheap, per-query, honest quality signal — one the user can see, even if it's just a not_found flag — is differentiated product value. It turns a question of system reliability into a transparent product affordance. And the "cheap" part matters: a signal that doubles the cost of every query is a signal that will get turned off.

Warts-and-all calibration beats shipped-with-asterisks calibration. My temptation with the confidence-tier data was to bury the high-vs-medium ordering issue behind hedging language and ship. Writing the warts-and-all version forced me to articulate exactly what the signal reliably does (not_found detection, aggregate monitoring) versus what it doesn't (per-answer probability estimates), and that distinction is what drives the roadmap — expand the test set, learn the weights, replace the regex with a semantic check. The honest version is more useful than the polished version.

Token budget and precision are the same variable, not opposing ones. The top_k=10 experiment improved every retrieval and generation metric for a 13% token cost, and the re-ingestion to section-level chunks did the same at higher magnitude. The PM frame that treats tokens as a cost center — "can we prompt-engineer our way to fewer tokens" — misses half the picture. Sometimes more tokens buy more precision, and sometimes fewer tokens are the precision gain because noise is what you dropped. The question is always "is this context actually load-bearing for the answer," not "are we using too many tokens."

What's Next

200+ question evaluation set — the single highest-leverage next step. The current 60-question set produced the confidence calibration uncertainty above, and the generation metrics' confidence intervals aren't tight enough to resolve sub-10-point differences between configurations. Expanding the set to 200–300 questions, with systematic coverage across question types (lookup, comparison, interpretation, cross-section synthesis) and regulatory subdomains, would both tighten the existing tuning decisions and give the confidence signal the calibration data it needs.

Semantic citation coverage. Replace the regex-based citation_coverage check with a lightweight semantic-grounding prompt. Each factual claim in the generated answer gets evaluated against the retrieved chunks for semantic support. More expensive than the regex, still far cheaper than a full LLM-judge call. The expected outcome: a confidence ordering that doesn't penalize paraphrase-without-inline-citation.

Differential corpus refresh (originally Phase 8). The current system ingests once. The CFR updates daily via the eCFR API. The schema was designed for atomic version-swap from the start — every chunk has a status ENUM (active / staged / archived) and a version_id, and retrieval filters status = 'active' at the query layer, not per-call-site. A weekly differential refresh that ingests changed sections into staged, runs sanity checks, then atomically swaps staged → active and the old active → archived is a natural follow-up. The versioned-replacement pattern is likely its own case study.

Corpus expansion. Currently Titles 7, 21, 42. The CFR has 50 titles. Expanding to full coverage is mostly a data-engineering effort, though the 50-title ingest would cross into territory where the existing 4GB droplet starts to feel tight — likely a database-sizing and index-structure (ivfflat vs. hnsw) revisit.

Monitoring and CI/CD. Currently deferred: Sentry for error tracking, a latency dashboard, and GitHub Actions push-to-deploy. None of these are technically interesting — they're plumbing — but they're the difference between a "I deployed this once" and "I operate this continuously." Worth doing before the user count goes past one.

Closing Thought

The most useful thing this project taught me wasn't about RAG at all — it was about the gap between "we built an evaluation harness" and "we trusted our evaluation harness." I had the former from day one. The latter took a painful audit that invalidated weeks of tuning work and reframed the entire project. That's a PM story worth telling: the discipline of treating your own tools as suspect until proven otherwise.

The other durable lesson is about inference-time signals. Every RAG system I've seen ships with an eval harness (good), and most ship with zero per-query quality indicators (less good). The confidence signal

in this project is imperfect and the calibration story is in progress — but it exists, it's computed for free on top of the existing retrieval and generation pipeline, and in its current form it reliably catches the case where the system doesn't know. That's a product primitive, not an engineering concern. The version that learns its weights, swaps in semantic grounding, and ships tiers the user actually sees is the version I'd want to build next.

The live system runs at regs.bradhinkel.com. It is, at the time of writing, the most accurate and most honest version of this product I know how to ship with today's tools. That's the standard the next version will have to beat.

Legal Disclaimer

This tool provides information about federal regulations for educational purposes. It is not legal advice. Consult a qualified attorney for legal guidance.