FROM PROTOTYPE PIPELINES TO PRODUCTION-GRADE RETRIEVAL SYSTEMS

RAG Architecture Patterns for the Enterprise

Executive briefing — CTO & Engineering Leadership | ~20 min

Why Naive RAG Breaks Down at Scale

- Single embed-and-retrieve pipelines are built for demos, not for millions of documents with mixed formats and freshness needs
- Flat top-k retrieval ignores document structure, recency, permissions, and query intent
- Retrieval quality degrades as corpus size and topical overlap grow, surfacing plausible but wrong chunks
- No mechanism for multi-step reasoning, so questions requiring synthesis across documents silently fail
- Naive RAG has no built-in evaluation or observability, so failures surface as user complaints, not metrics

Chunking Strategy: The Foundation Layer

- Fixed-size chunking is simple but fragments tables, code blocks, and multi-step procedures mid-thought
- Semantic and structure-aware chunking (headings, sections, sentence boundaries) preserves meaning at the cost of implementation complexity
- Chunk size is a tradeoff: smaller chunks improve retrieval precision, larger chunks preserve context for generation
- Overlap between chunks reduces boundary-loss errors but increases index size and duplicate retrieval
- Document type should drive strategy — contracts, code, and chat logs each need different chunking logic

Embedding Model Selection

- Domain fit matters more than benchmark leaderboard rank — legal, medical, and code corpora often need specialized or fine-tuned models
- Dimensionality is a storage/latency/accuracy tradeoff, not a free lunch — higher dimensions cost more to store and query
- Managed API embeddings simplify operations but create vendor lock-in and recurring per-token cost exposure
- Self-hosted open-weight models offer data control and cost predictability but require MLOps investment
- Embedding model changes require full re-indexing — this is a migration event, not a config toggle, and should be planned for

Vector Database Landscape

- Managed vector databases reduce operational burden but add a new vendor dependency and data residency questions
- Self-hosted options (open-source vector stores, or vector extensions on existing databases) trade convenience for control and lower marginal cost at scale
- Hybrid search — combining dense vector similarity with sparse keyword/BM25 — consistently outperforms vector-only search on exact-match and rare-term queries
- Metadata filtering (department, document type, date, permission tags) is often more decisive for relevance than embedding quality alone
- Choice should follow existing data infrastructure and team skills, not just feature comparison tables

Advanced Pattern: Re-Ranking

- Initial retrieval optimizes for recall (cast a wide net); re-ranking optimizes for precision (pick the best of that net)
- Cross-encoder or lightweight LLM-based re-rankers reorder the top-N candidates using deeper query-document interaction
- Adds meaningful latency per query, so it is typically applied only to a shortlist (e.g., top 20-50), not the full retrieved set
- Materially improves answer quality in the enterprise-scale corpora where naive top-k retrieval falls short
- Should be treated as a separate tunable stage, not bundled invisibly into the retriever

Advanced Pattern: Query Rewriting & Decomposition

- Raw user queries are often ambiguous, underspecified, or reference prior conversation context that the retriever cannot see
- Query rewriting normalizes phrasing and expands acronyms/synonyms before retrieval, improving match rates
- Query decomposition breaks compound questions into sub-questions, each retrieved and answered independently, then synthesized
- Enables multi-hop reasoning — answering questions that require connecting facts across multiple documents
- Adds an extra LLM call and latency; justified for complex knowledge-work queries, often unnecessary for simple lookups

Advanced Pattern: Hybrid & Agentic RAG

- Hybrid keyword+vector search hedges against the weaknesses of either method alone, particularly for IDs, codes, and proper nouns
- Agentic RAG lets the system decide whether to retrieve, which source to query, and whether to iterate — rather than always doing a single fixed retrieval pass
- Multi-hop agentic patterns can query multiple systems (docs, databases, APIs) and reason across the combined results
- Increases capability but also increases cost, latency variance, and failure surface — each additional step is a place errors can compound
- Best reserved for query classes where single-pass retrieval has demonstrated a measurable quality gap

Evaluation Methodology

- Retrieval precision and recall must be measured against a labeled evaluation set, not judged anecdotally by spot-checking answers
- Answer faithfulness (is the response actually supported by the retrieved context) is a distinct metric from answer correctness
- Hallucination testing requires adversarial and out-of-scope queries specifically designed to probe when the system should say "I don't know"
- LLM-as-judge evaluation can scale review but should be periodically validated against human judgment, not trusted blindly
- Evaluation should run continuously against production query logs, not only at initial launch

Illustrative Deployment Example

- Illustrative scenario, not a verified case study: a mid-size enterprise consolidating policy documents, engineering wikis, and support tickets into one assistant
- Phase 1 (weeks 1-4): narrow pilot on a single well-curated document set with hybrid search and basic re-ranking
- Phase 2 (weeks 5-10): expand corpus, add access-control-aware retrieval, and instrument evaluation dashboards
- Phase 3 (weeks 11+): introduce query decomposition for complex queries and expand to additional business units
- Illustrates a staged rollout, not a specific vendor stack or guaranteed timeline for any given organization

Latency and Cost Tradeoffs

- Each added stage (query rewriting, re-ranking, multi-hop) improves quality but adds latency — end-to-end response time should be a design constraint, not an afterthought
- Embedding and generation costs scale with token volume; chunk size and retrieval breadth directly drive per-query cost
- Managed services shift cost from fixed infrastructure to variable per-call pricing, which can be advantageous at low volume and expensive at high volume
- Caching frequent queries and embeddings reduces both cost and latency for repeat or near-duplicate questions
- Architecture choices should be validated against realistic query volume projections, not idealized demo traffic

Security: Access-Control-Aware Retrieval

- The retriever must enforce the same permissions as the source system — a user should never see, via the assistant, content they could not already access directly
- Naive RAG implementations frequently index everything into one flat store, creating a permission-bypass risk by default
- Access control should be enforced at query time (filtering by user entitlements) and validated with periodic permission audits
- Row-level or document-level metadata tagging is required infrastructure, not an optional enhancement
- This is a common and serious failure mode in early RAG deployments and should be treated as a launch-blocking requirement, not a post-launch fix

Observability and Citation Traceability

- Every generated answer should be traceable back to the specific source chunks that produced it
- Citations let users verify claims and give engineering teams a debugging path when answers are wrong
- Logging retrieved chunks, scores, and re-ranking decisions is essential for diagnosing quality regressions after model or index changes
- Query and retrieval logs support both evaluation and compliance/audit requirements in regulated environments
- Without traceability, hallucinations and retrieval errors are indistinguishable from the user's perspective

Common Failure Modes

- Stale indexes: source documents change but the vector index is not refreshed, so the assistant answers from outdated information
- Wrong-chunk retrieval: a chunk is topically similar but contextually irrelevant, producing a confident but incorrect answer
- Context window overflow: too many retrieved chunks are stuffed into the prompt, degrading generation quality or truncating critical content
- Permission leakage: retrieval bypasses source-system access controls, as covered in the security section
- Silent degradation: without ongoing evaluation, these failures accumulate gradually and are often discovered by users before they are caught internally

Next Steps and Rollout Plan

- Commission a scoped pilot on one high-value, well-bounded document set with clear success metrics before any broad rollout
- Stand up an evaluation harness (retrieval precision/recall, faithfulness, hallucination tests) as a prerequisite, not a nice-to-have
- Require access-control-aware retrieval and full citation traceability as launch-blocking criteria, not post-launch fixes
- Assign clear ownership for index freshness, evaluation monitoring, and cost tracking before scaling beyond the pilot
- Decision needed: approve pilot scope, budget, and timeline, and confirm the initial document set and business sponsor