Automating Regulatory → Internal Requirement Mapping

A technical deep-dive into retrieval experiments, their limitations, and a proposed agentic full-space reasoning approach

Executive Summary

- Over the past few weeks, I’ve been working on analysing and improving the automation of mapping external regulatory citations (NIST, DORA, ISO, etc.) to our internal cybersecurity requirements. This document summarizes (1) the core problem we face, (2) the retrieval-based approaches tried previously by the team and the enhancements I experimented with, (3) why retrieval—even in advanced forms—does not fully solve this use case, and (4) a new proposed architecture using domain-specialist agents who evaluate the entire requirement space.
- Note: The earlier experiments made it clear that retrieval alone had precision and recall constraints in this scenario. I used those insights as a starting point for further exploration and to evaluate alternative approaches.

The Core Problem

- Our organization maintains a large set of internal cybersecurity requirements (around 4,000). These internal requirements are usually **very concrete, operational, and implementation-focused**. They describe *what exactly needs to happen*, *who must do it*, and *how often it must be done*.
- External regulatory citations—whether from NIST, DORA, ISO, or other bodies—tend to be **high-level, principle-driven, and abstract**. They express the *intent* or *outcome* regulators expect but rarely specify the detailed mechanisms through which organizations must achieve those outcomes.
- This creates a structural challenge:
The language, abstraction level, and intent of regulatory citations rarely match how our internal controls are written.
- To illustrate why this makes the mapping inherently difficult, the next slide contains detailed examples that consistently appear in real evaluations.

Example 1:

- **External Citation (Regulator):**
  “The organization shall ensure appropriate oversight and alignment of cybersecurity activities with its strategic objectives.”
- **Internal Requirement (Ours):**
  “Risk owners must be assigned to each business process, and review meetings must be held quarterly.”
- **Why Retrieval Fails:**
  - The citation uses concepts like “oversight”, “alignment”, “strategic objectives”.
  - The requirement focuses on “risk owners”, “processes”, “review meetings”.
  - There is almost no lexical overlap.
  - Dense embeddings won’t see them as semantically close.
  - Sparse retrieval won’t match any meaningful tokens.
- **But logically:**
  Assigning risk owners + scheduled governance reviews *is exactly how* cybersecurity oversight aligns with strategic objectives.
- This is a **mechanistic implementation** of a **governance-level intention**, but the language is mismatched.

Example 2:

- **External Citation:**
  “Organizations must be prepared to detect and respond to cybersecurity incidents in a timely manner.”
- **Internal Requirement:**
  “The SOC shall escalate incidents to Tier-2 analysts within 15 minutes based on predefined severity thresholds.”
- **Retrieval Problems:**
  - The citation uses broad terms: “prepared”, “detect”, “respond”, “timely”.
  - Our requirement uses highly operational terms: “SOC”, “Tier-2”, “15 minutes”, “severity thresholds”.
  - Even a powerful hybrid retrieval pipeline cannot reliably connect these unless the exact tokens overlap.
- **Human interpretation:**
  This internal control *is* the operational enforcement of “timely detection and response”.
  But no embedding model will reliably map “Tier-2 escalation in 15 minutes” → “timely response” across 5,000 documents.
- This is a **conceptual match**, not a **semantic match**.

What we have Built so far (and What I Learned From It)

- Before I joined this effort, the team implemented a solid RAG-based model. They used dense embeddings over internal requirements, retrieval over a vector store, and an LLM to evaluate and rank candidates. Their work helped me get a grounded understanding of:
  - how internal requirements are structured
  - how citations vary in tone and detail
  - the kinds of mismatches retrieval struggles with
  - what the precision and recall looked like in practice
- The team also experimented with enhancing retrieval by adding contextual information like Domain, Topic, etc into the requirement chunks, RAG Fusion, breaking citations into LLM-generated sub-citations to increase recall. Through their evaluations, it became clear that recall did improve, but precision remained low and many valid mappings were still missed. This set the stage for me to explore additional retrieval techniques and understand their behaviour on this use case.

Retrieval Techniques I Explored (High-Level Overview)

To build on the earlier work and see how far retrieval could be pushed, I experimented with several additional approaches:

- **Sparse neural retrieval** (SPLADE/ColBERT style modelling)
- **Hybrid retrieval** combining sparse + dense signals
- **Reciprocal Rank Fusion (RRF)** to merge ranked lists from multiple retrieval sources
- **Deeper analysis of RAG Fusion** and its sub-citation generation behaviour

All of these techniques are strong for general search/QnA tasks, and each contributed valuable insights. However, the evaluations confirmed the same fundamental issue: retrieval alone is not sufficient for this mapping task.

Dense Embedding Retrieval: Why It Fell Short

Dense embeddings are excellent for capturing semantic similarity across natural language, but our use case often requires mapping texts that have *no natural semantic proximity*.

Example:

- Citation: "*Cybersecurity must align with organizational vision and leadership*."
- Requirement: "*Risk owners must be assigned to each business process*."

These two sentences will never be close in embedding space, yet logically the requirement is exactly one of the mechanisms that satisfies the citation.

Dense retrieval consistently struggled here because:

- It assumes similarity = relevance.
- It cannot infer "*role assignment → governance → oversight → alignment*."
- It misses requirements that are *mechanistically* relevant but *textually* unrelated.

Sparse Neural Retrieval: What It Adds and Why It Still Misses

What Sparse Models Do
Sparse neural retrieval (e.g., SPLADE) is essentially a **smarter BM25**: instead of dense embeddings, it produces sparse vectors weighted by learned token importance. It captures important terms, expands related tokens, and improves keyword-based matching.

Simple intuition:
If BM25 "counts exact keyword matches," sparse neural models behave like a **smarter BM25** that can expand important terms, down-weight irrelevant ones, and highlight words that truly matter in context.

Why I Tried It
I experimented with sparse models hoping they would catch matches that dense embeddings miss. Specifically:

- Dense models were sometimes *too fuzzy*, losing domain-specific terminology.
- Sparse models excel when terminology matters (legal search, code search, e-commerce, etc.).
- They offer interpretability and transparent term-level reasoning.
- I expected them to find cases where requirements and citations share even partial vocabulary or domain cues.

Where It Breaks
Sparse models still rely on **token overlap or token-level relatedness**. In our setting, regulatory text uses **abstract governance language**, while internal requirements use **operational/technical language** — almost no shared vocabulary.

Sparse retrieval cannot infer mappings like:
"risk owner assignment" → "oversight" or
"15-min escalation" → "timely incident response."

Core Limitation
They operate in **lexical space**, not **reasoning space**. They can expand terms and match synonyms, but **cannot bridge the abstraction gap** between high-level regulatory intent and low-level control implementation.

Hybrid Retrieval + RRF: Strong Retrieval, Still Insufficient

Since dense models alone weren't capturing the signals we needed, and sparse models weren't bridging the abstraction gap, the next logical step was to combine them.

Hybrid retrieval blends:

- **Dense embeddings** → capture general meaning and conceptual similarity
- **Sparse models** → capture keyword-level precision

This approach often gives the *best of both worlds*, especially in traditional search scenarios like: *document search, FAQ retrieval, question answering, customer support automation, legal clause lookup.*

In many of these domains, hybrid retrieval is the current best practice because dense and sparse signals complement each other's weaknesses.

Why I Tried Hybrid Retrieval + RRF

My goal was to see whether combining different similarity signals could surface more potential candidates so that the LLM reranker could work on a richer pool. This is a standard approach in IR, and it often boosts both recall and precision.

What Worked (Where It Helped)

- The candidate pool became more diverse.
- Some requirements that dense retrieval alone missed were recovered through sparse signals.
- RRF stabilized the top-ranked candidates and avoided random fluctuations.
- It improved the overall retrieval quality for QnA-like tasks.

However, even the best hybrid retrieval still depends on *some form* of textual/semantic similarity.

Why Retrieval (In Any Form) Will Always Underperform Here

- This is the fundamental insight that emerged after building on the previous team's work and trying multiple retrieval enhancements:
- Retrieval methods rely on textual and semantic similarity. But regulatory → requirement mapping requires **abductive reasoning**, **organizational context**, and **multi-hop inference**, such as:
    - "If the citation requires governance alignment..."
    - "...and governance alignment is typically achieved by assigning accountable owners..."
    - "...and requirement R-001 assigns risk owners..." → therefore R-001 partially or fully satisfies the citation.
- This reasoning chain cannot be encoded into embeddings or sparse vectors. Retrieval can only "see" what is near in the language space; it cannot infer relationships that are implicit or conceptual.

Proposed Direction: Full-Space, Domain-Specialist Agents

Given all the above, the next logical step is to shift from retrieval-driven filtering to **exhaustive reasoning** within each domain.

The idea is:

- Build specialist agents (Risk, Access Control, Incident Response, etc.).
- Give each agent the *entire* set of requirements belonging to its domain (in batches).
- Let agents reason through each requirement logically.
- Use LLMs to perform controlled, stepwise evaluation, not similarity lookup.
- Parallelize agents to maintain throughput.
- This ensures **every requirement is evaluated**, not just the ones that appear semantically similar.

Why “Brute Force” Is Acceptable (And Even Preferred)

Even though this approach is more computationally intensive, it has major advantages:

- *The business prefers accuracy and completeness over raw speed.*
- *Compliance work is high-stakes; missing one requirement is worse than taking longer.*
- *We get full auditability: every requirement was reviewed.*
- *Eventually, optimizations (caching, heuristics, smaller models) can reduce latency while maintaining coverage.*

Orchestration & Concurrency to keep throughput manageable:

- *Batch requirements (e.g., 50–200 per batch).*
- *Run multiple specialist agents in parallel.*
- *Use rate limiting and checkpoints to prevent runaway cost.*
- *Cache previous evaluations to avoid redundant work.*
- *Concurrency allows us to evaluate large requirement sets without sacrificing completeness.*

Conclusion

Based on everything explored so far — dense embeddings, sparse neural retrieval, hybrid models, RRF-based fusion, advanced reranking, and the earlier RAG experiments done by the team — the conclusion I've reached is not just an empirical one, but a conceptual one as well.

Even if we continue improving retrieval, the nature of this specific problem places an inherent ceiling on what retrieval-based approaches can achieve. The issue isn't only a matter of "better embeddings" or "better ranking." It is rooted in how the two sides of the mapping problem are structured:

- regulatory citations express **intent at a very high level**,
- internal requirements express **mechanisms at a very operational level**,
  and the relationship between the two is often **implicit, multi-hop, and organizational**, not textual or semantic.

Because of this abstraction gap, a retrieval-first approach — no matter how advanced — will always focus on *similarity*, while the task requires *reasoning*. Retrieval cannot reliably detect mappings that only exist after interpreting intent, understanding internal processes, and connecting multiple controls together.

After trying multiple retrieval architectures, and seeing where each of them hit similar limitations, the broader conclusion becomes clear:
RAG (or any retrieval-centric pipeline) is not a suitable core solution for this use case.
This is true both from the theoretical standpoint (the task is not similarity-grounded) and from practical experiments (recall/precision issues consistently surfaced).

Given the nature of the use case, the most consistent way to reduce false negatives — which are often more costly than time or compute — is to expand the search scope instead of restricting it. In other words: "We need to let the llm models reason and evaluate *more requirements*, not less."

This leads naturally to an agentic, full-space, domain-specialist approach where LLMs are allowed to systematically examine larger portions of the requirement corpus and perform structured reasoning rather than rely solely on retrieval filtering.

This approach may evolve and be optimized over time — with batching, concurrency, caching, and heuristic shortcuts — but directionally it addresses the key challenge:
exhaustive evaluation is fundamentally more aligned with the risk and accuracy expectations of compliance work than similarity-based retrieval.