Retrieval Unit Mismatch: Why Chunk-Based RAG Cannot Align with Document Structure

Anonymous ACL submission

Abstract

Standard RAG retrieval evaluation measures whether the answer *text* appears in the top-k retrieved units, but not whether the retrieved unit is the structurally appropriate container for that answer. We show this blind spot is not a measurement oversight but a substantive failure mode: a chunk-based retriever can achieve near-perfect recall while its structural alignment is *architecturally zero* — chunks carry no element type, so no chunk-size, BM25 weight, or embedding choice can move it. On a 380-document, 6,383-query multi-domain corpus, three chunk backends score zero on element-type alignment, while a typed-CDU index (*Semantic IR*) achieves `etype_hit@1` = 0.74 and `unit_hit@1` = 0.58. A controlled BM25 ablation moves recall@1 substantially but alignment by less than 1pp, confirming that coverage and structural alignment are orthogonal axes responding to different interventions. Even conditioned on a recall hit, chunk systems still return the wrong element type in 100% of cases, while Semantic IR returns the right type in 84%. Supporting analyses show that the typed-index benefit scales with document structural richness and is sensitive to CDU projection quality. Our contribution names the retrieval-layer metrics that make this structural blind spot visible and shows why no parameter sweep within the chunk paradigm can close it.

1 Introduction

A RAG system that cannot retrieve the right *kind* of evidence cannot answer the right *kind* of question — even if the answer tokens exist somewhere in its top-k results. A user asking "what is the learning-rate schedule used in training?" needs a value from a table or an equation, not a paragraph that mentions "learning rate" in passing. Yet the dominant RAG paradigm — fixed-size chunk indexing — is structurally incapable of distinguishing these cases at retrieval time: a chunk is a character window, not a document element.

Central claim. This paper argues — and empirically demonstrates — that *chunk-based retrieval architecturally cannot achieve structural alignment on typed queries, and standard recall evaluation cannot see this failure*. We make the claim precise through two retrieval-layer metrics (`etype_hit@k` and `unit_hit@k`) on which chunk systems score zero by construction, and show that this zero is not movable by tuning.

The evidence quality gap. Recall measures whether the answer *text* is somewhere in the retrieved units. It does not measure whether the retrieved unit is the structurally appropriate evidence container. On our multi-domain corpus, section-aware chunking achieves `recall@5` $\approx$ `0.96` on every typed query class — the answer is essentially always present — while `etype_hit@5 = 0.000` on all of them: by construction, none of the retrieved units carries an element type, so the top-k delivers answer-containing prose sections in place of the equations, tables, or code blocks the query was about. Current RAG retrieval evaluation is blind to this failure.

Contributions. This paper makes three contributions in support of the central claim:

1. **Two retrieval-layer metrics for structural alignment.** We define `etype_hit@k` (element-type alignment) and `unit_hit@k` (exact CDU identity), both computable from parser metadata at corpus scale without an LLM judge (§3).
2. **A full-corpus comparison showing chunk systems' architectural zero (§6.1, §6.3).** Across four backends, all chunk variants score `etype_hit@1 = 0.000`; Semantic IR achieves `0.737` and `unit_hit@1 = 0.583`. Even conditioned on a recall hit — the condition most favourable to chunking — chunk systems still deliver the correct element type in 0.0% of cases versus 84.4% for Semantic

IR.

3. **An axis-independence ablation (§6.2).** BM25 fusion moves recall@1 by +11.75pp and alignment by +0.9pp, demonstrating that coverage and alignment respond to different interventions and that no retrieval-layer parameter within the chunk paradigm can cross the architectural boundary.

Supporting characterisation (§7, Appendices B–D). Semantic IR's benefit is conditional on document structural richness (Appendix B), is sensitive to CDU projection quality (Appendix C), and is over-reported by LLM-generated benchmark queries relative to human-written ones (Appendix D).

What this paper does not claim. We do not claim that Semantic IR is the complete solution to structured retrieval. Semantic IR's own `unit_hit@1 = 0.583` shows that short typed units (equations, tables, code) lose the rank-1 slot to their parent sections in a shared embedding space — *granularity-induced ranking bias* — and we identify this as the central open problem for follow-up work (§8.3). What we do claim is that no amount of parameter tuning within the chunk paradigm can reach the region of the design space where Semantic IR already sits, and that the field currently lacks the retrieval-layer metrics to see why.

2 Background and Related Work

2.1 Chunking in RAG

The dominant approach to document indexing in RAG (Lewis et al., 2020; Gao et al., 2023) splits text into overlapping fixed-size windows, embeds each window, and retrieves by cosine similarity. Chunk size has been studied as a hyperparameter affecting downstream QA quality (Xu et al., 2023; Edge et al., 2024). The *parent-document retriever* pattern (Liu, 2023) retrieves fine-grained child chunks but expands to parent context for generation — an empirical workaround for the information poverty of short units that we formalise as **granularity-induced ranking bias** (§6.3, §8.3). None of these works introduces element-typed indexing or measures structural alignment as an independent axis.

2.2 Retrieval Granularity

Dense-X Retrieval (Chen et al., 2024) proposes indexing *propositions* — atomic factual claims — rather than passages, showing that finer-grained units improve precision for fact-seeking queries. This is the closest prior work in spirit: both Dense-X and we identify passage-level granularity as a precision bottleneck. The key difference is ontological: Dense-X units are LLM-generated atomic claims (no element type), whereas our CDUs are parser-recovered document elements carrying type semantics that enable our alignment metrics.

2.3 Document Structure in Information Retrieval

Element-level retrieval has a long history in XML IR. The INEX benchmarks (Fuhr et al., 2002; Trotman and Lalmas, 2006; Fuhr et al., 2008) established the core problem: given an XML document with markup-defined element boundaries, retrieve the most *specific* element that answers the query. INEX defined *content-oriented retrieval* tasks requiring systems to return elements at appropriate granularity and introduced element-hierarchy metrics that directly anticipate `etype_hit@k` and `unit_hit@k`.

Our work differs from INEX in three ways. First, INEX operates on documents with explicit XML markup; we recover element boundaries from LaTeX, HTML, and PDF using document parsers. Second, INEX was conducted in the pre-neural era; we operate in the dense-retrieval + LLM context, where element type is not incidental metadata but a *dispatch key* for downstream generation. Third, INEX evaluation relied on human-judged relevance grades; `etype_hit@k` and `unit_hit@k` are automated metrics computed from parser metadata, applicable at any scale.

Modern neural IR has largely abandoned element-typed indexing in favour of dense passage retrieval (Karpukhin et al., 2020; Khattab and Zaharia, 2020; Guu et al., 2020). We argue this abandonment was premature for the structured-document setting: element type is semantically load-bearing for the generator, not merely for the human reader.

2.4 Structured Document Benchmarks

QASPER (Dasigi et al., 2021) provides QA pairs over scientific papers with evidence spans but does not annotate element types. DocVQA (Mathew et al., 2021) requires visual document understanding orthogonal to our setting. CReSt (CReSt Benchmark Authors, 2025) and FreshStack (FreshStack Authors, 2025) target technical documentation and developer knowledge bases; they recognise that

standard passage metrics are insufficient for highly structured corpora, a motivation we share. Our contribution adds the element-type alignment axis, the `unit_hit` metric, and the structural-anchor methodological analysis to this literature.

2.5 Structured Document Parsing

Recent work on PDF/LaTeX/HTML parsing (GROBID contributors, 2008–2024; Blecher et al., 2023; IBM Research, 2024) produces equation, table, figure, code, and section boundaries as structured outputs. We use these boundaries directly as retrieval-unit boundaries and treat element type as a first-class index property.

2.6 RAG Evaluation

End-to-end RAG evaluation frameworks such as RAGAS (Es et al., 2023) and ARES (Saad-Falcon et al., 2023) use LLM judges to score generated answers. Our focus is the retrieval layer in isolation, following the tradition of IR benchmarks (Voorhees, 2001; Thakur et al., 2021; Zhang et al., 2023). We introduce three retrieval-layer metrics sensitive to structured evidence: GT-recall (length-independent token-overlap threshold $\theta = 0.2$), `etype_hit@k` (element-type alignment), and `unit_hit@k` (exact CDU identity match). None requires an LLM judge, and all three are computable at corpus scale.

3 Problem Formulation

3.1 Coherent Document Units (CDUs)

Given a parsed document D, we define its *Coherent Document Unit* set:

$$\mathcal{U}(D) = \mathcal{U}_{\text{sec}} \cup \mathcal{U}_{\text{eq}} \cup \mathcal{U}_{\text{tab}} \cup \mathcal{U}_{\text{fig}} \cup \mathcal{U}_{\text{code}}$$

Each CDU $u \in \mathcal{U}(D)$ has an *element type* $\tau(u) \in \{\text{section}, \text{equation}, \text{table}, \text{figure}, \text{code_block}\}$ and a stable identifier $\text{id}(u) = \langle \text{doc_id} : \tau(u) : \text{element_id} \rangle$. Long sections may be sentence-boundary split into sub-units with IDs of the form `sec_N:chunk_j`. We write $\text{id}(r) \sqsubset \text{id}(u)$ when $\text{id}(r)$ is a sub-unit identifier of $\text{id}(u)$ — concretely, when $\text{id}(r) =$ `sec_N:chunk_j` and $\text{id}(u) =$ `sec_N`, the gold-section ID being a prefix of the sub-unit ID under our colon-delimited schema. The `unit_hit@k` metric treats such sub-units as legitimate matches against the gold section.

A *chunk-based* index defines retrieval units through a windowing function $W_c(D)$ parameterised by chunk size c. Chunks have no type and their boundaries are independent of document structure. Throughout the paper, "chunk" refers to this untyped, fixed-window form; a chunked index augmented with explicit element-type metadata is outside the scope of the comparison.

3.2 Metrics

Let q be a query with ground-truth answer text g_q and gold-CDU set $G_q = \{u_1, \ldots, u_m\}$. Let $R_k = (r_1, \ldots, r_k)$ be the top-k retrieved units.

GT-recall@k.

$$\text{recall@}k = \mathbf{1}\left[\max_{r \in R_k} \frac{|\text{tok}(g_q) \cap \text{tok}(r)|}{|\text{tok}(g_q)|} \geq \theta\right], \quad \theta = 0.2$$

Length-independent token overlap; sensitive to short extractive answers.

`etype_hit@k`**.** For queries with a single gold element type $\tau^\star$,

$$\text{etype_hit@}k = \mathbf{1}[\exists\, r \in R_k : \tau(r) = \tau^\star]$$

The metric reads the retrieved *unit's* element type, not the type of any element that happens to lie inside a chunk window. For untyped chunk indexes, units are character windows with no element type field — $\tau(r) \equiv \varnothing$ — so `etype_hit@k` $\equiv 0$ **architecturally**, not merely empirically. By convention we set `etype_hit@k` = 0 for `cross_element_qa` (no single $\tau^\star$); aggregate values across query types therefore weigh structural alignment by the typed-query share, not by an artificial uniform mix.

`unit_hit@k`**.**

$$\text{unit_hit@}k = \mathbf{1}[\exists\, r \in R_k, u \in G_q : \text{id}(r) = \text{id}(u) \vee \text{id}(r) \sqsubset \text{id}(u)]$$

where $\sqsubset$ is the sub-unit relation defined in §3.1. Convention: for non-CDU retrieved units (chunk and section backends), $\text{id}(r)$ lies outside the CDU namespace so the indicator is 0 by construction.

`type_match | hit` (used in §6.3) is the aggregate fraction of *recall hits* whose top-1 unit also has the correct element type:

$$\text{tmH}(Q) = \frac{\sum_{q \in Q} \mathbf{1}[\text{recall@1}(q)=1]\, \mathbf{1}[\tau(r_1)=\tau^\star(q)]}{\sum_{q \in Q} \mathbf{1}[\text{recall@1}(q)=1]}.$$

3.3 The Evidence Quality Gap

Definition. For a query q with gold element type $\tau^\star \neq \varnothing$, the *evidence quality gap* is the event

$$\text{EQ-gap}(q) = \mathbf{1}[\text{recall@}k(q) = 1 \wedge \text{etype_hit@}k(q) = 0].$$

The gap is *structurally inevitable* for chunk-based systems on typed queries: a chunk containing a formula embeds that formula inside surrounding prose, so the retrieval returns an *answer container* rather than the *answer-bearing unit*. We quantify the gap in §6.1.

3.4 The Design Space

We characterise retrieval systems by their position in the 2-D space (`recall@1`, `etype_hit@1`). The central empirical claim is that (1) chunk-based systems are confined to the left strip (`etype_hit@1 = 0`) regardless of parameters, by the architectural argument above; (2) the upper-right quadrant (high coverage *and* high alignment) is unattained in our experiments; and (3) the two axes respond to different interventions — coverage to BM25 and chunk size, alignment to the indexing abstraction itself.

4 Systems

We implement four retrieval backends, all using `bge-small-en-v1.5` embeddings and no reranker.

- **Naive Chunk (`naive`).** Fixed-size overlapping windows ($c = 1000$ chars, overlap $= 100$), vector retrieval only.
- **Hybrid Chunk (`hybrid`).** Same windows as Naive, with BM25 + vector fusion at weight $\lambda = 0.3$: $s = (1 - \lambda)s_{\text{dense}} + \lambda s_{\text{BM25}}$.
- **Section-Aware (`section`).** Heading-boundary splits without further overflow chunking; vector retrieval only. Units respect section boundaries but are *not* typed (no element-type metadata).
- **Semantic IR (`sem`).** Element-typed CDU index. Each equation, table, figure, code block, and section is embedded as a separate typed unit. Retrieval uses BM25 + vector fusion ($\lambda = 0.3$);[1] the BM25 channel is computed over a CDU projection string (element type, section-path prefix, header row / caption / first sentence / code language, content body) designed to surface element-specific keyword signal. Long sections are sentence-boundary split so that `sec_N:chunk_j` sub-units remain coherent.

[1] Semantic IR uses BM25 at the same weight as `hybrid`, making the recall comparison fair: Semantic IR's recall@1 still sits *below* `hybrid` (0.879 vs 0.928 in §6.1), so BM25 is not inflating Semantic IR's coverage. For the alignment metric, BM25 is empirically irrelevant — §6.2 shows that varying λ moves `etype_hit@1` by less than 1pp, and chunk backends return 0 regardless of λ.

5 Corpus and Benchmark

Our primary corpus is `main_paper_380`: 380 documents spanning 7 domains (scientific, developer, finance, legal, medical, encyclopedia, reports) with an accompanying query set `typed_queries_380` of 6,383 queries across 6 types (`factual_qa`, `section_qa`, `table_qa`, `equation_qa`, `code_qa`, `cross_element_qa`). Gold `unit_ids` are assigned for 99.8% of queries. Per-domain corpus statistics and query distributions are given in Appendix A (Table A.1, Table A.2).

Corpus sources. Documents are drawn from established public collections rather than self-collected: scientific papers from arXiv (92 docs, LaTeX source) plus 3 non-arXiv scientific reference texts (the annotated Transformer; an RL textbook chapter; a scientific ML chapter); biomedical literature from PubMed Central (44 docs); financial filings from SEC EDGAR (103 docs, including documents from the FinanceBench benchmark (Islam et al., 2023)); legal contracts and regulations from public legislative sources; developer documentation from Python, MDN, FastAPI, Rust, Kubernetes, and related official sources; and encyclopedic content from Wikipedia and the Stanford Encyclopedia of Philosophy. The multi-source construction means corpus quality and formatting diversity are not artefacts of a single pipeline.

Queries are generated by GPT-4o-mini from typed-element seeds and pass a verification step that rejects unanswerable or under-grounded items. We discuss benchmark validity, including a comparison to human-written queries on a multi-domain pilot, in Appendix D.

6 Main Result: The Chunk Alignment Ceiling

This section presents three pieces of evidence for the paper's central claim: chunk-based retrieval architecturally cannot achieve structural alignment, and no parameter within the chunk paradigm can move it.

6.1 Full-Corpus Backend Comparison

Table 1 reports aggregate retrieval metrics for all four backends on `main_paper_380`.

The evidence quality gap in one number. On `equation_qa` (n = 254), the `section`-aware

Table 1: Aggregate retrieval metrics (n = 6,383 queries, 380 docs). For untyped chunk/section backends, `etype_hit@k` and `unit_hit@k` are 0 by construction (no element type and no CDU identifier on the retrieved unit, respectively); these are architectural ceilings rather than tuned values.

Backend	n_{units}	recall@1	recall@5	MRR	`etype_hit@1`	`unit_hit@1`	`unit_hit@5`
`naive`	36,403	0.8751	0.9649	0.9120	0.0000	0.0000	0.0000
`hybrid`	36,403	**0.9276**	**0.9781**	**0.9475**	0.0000	0.0000	0.0000
`section`	44,198	0.8473	0.9561	0.8910	0.0000	0.0000	0.0000
`sem`	56,966	0.8794	0.9693	0.9168	**0.7371**	**0.5827**	**0.8002**

Table 2: `etype_hit@1` by query type (Semantic IR; all chunk backends 0.0000). `cross_element_qa` has no single expected element type, so `etype_hit@1` is 0 by construction for all backends.

Query type	n	`sem etype_hit@1`	`sem recall@5`
`section_qa`	2,068	**0.9458**	0.9836
`factual_qa`	2,226	**0.9380**	0.9695
`table_qa`	905	**0.5525**	0.9260
`equation_qa`	254	**0.3150**	0.9843
`code_qa`	277	**0.2924**	0.9567
`cross_element_qa`	653	0.0000	0.9832

chunk backend achieves `recall@5 = 0.982` — the answer text is in the top-5 for ∼98% of queries — while its `etype_hit@5 = 0.000` by construction: every retrieved unit is a section (the only unit type `section` indexes), so even when an equation lies inside that section, no equation CDU is returned. The answer text is present; the evidence structure is not. This is not an implementation defect; it is the consequence of indexing without element types.

Finding 1. All chunk variants score `etype_hit@1 = 0.000` across all query types. Semantic IR achieves `etype_hit@1 = 0.737` and `unit_hit@1 = 0.583` on the same corpus. Recall remains competitive: Semantic IR is 4.8pp below the best chunk baseline at rank 1 and 0.9pp below at rank 5 — the cost of typed retrieval is a ranking cost (§8.3), not a candidate-set cost.

6.2 Axis Independence: Can Tuning Close the Gap?

A natural objection is that retrieval-layer tuning (BM25 fusion, chunk size, embedding model) might raise chunk `etype_hit@1` from its observed zero. For untyped chunk indexes this is architecturally precluded — no retrieval-layer score function can populate an element-type field that the indexed unit does not have. For Semantic IR, we run a controlled BM25 ablation (1,890 queries; same query-type mix as Exp 008, including all six types) to test the same axis: BM25 is the largest recall-affecting retrieval-layer intervention we measured in our corpus.

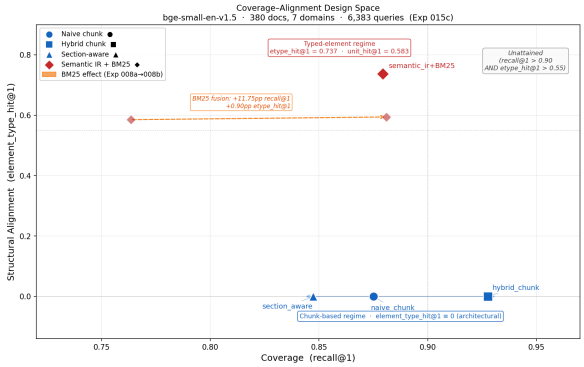

Figure 1: Coverage–alignment design space on `main_paper_380`. The three chunk backends cluster on the horizontal axis (`etype_hit@1`=0, architectural). Semantic IR at (0.879, 0.737). An amber arrow shows the BM25 ablation trajectory (+11.75pp recall, +0.90pp `etype_hit`). The upper-right quadrant remains unattained.

Finding 2 — axis independence on the BM25 axis. BM25 moves recall@1 by +11.75pp and alignment by +0.9pp on Semantic IR: the two axes respond on different scales to the same lever. For the untyped chunk baselines the conclusion is stronger by construction: BM25 cannot move `etype_hit@k` *at all*, because the indexed unit carries no element-type field. The result rules out BM25 as a way to recover alignment within the untyped-chunk paradigm; interventions that would move alignment (type-aware routing, parser-derived metadata) already step outside that paradigm.

Table 3: BM25 intervention on Semantic IR.

λ	recall@1	`etype_hit@1`	Δ recall@1	Δ `etype_hit@1`
0.0	0.7635	0.5852	—	—
0.3	**0.8810**	**0.5942**	**+11.75pp**	**+0.90pp**
0.5	0.871	0.591	+10.8pp	+0.6pp

6.3 Two Stricter Measurements: Type-Precision and Identity-Match

This paper evaluates the retrieval layer in isolation and does not study end-to-end generation. A reasonable counter-position is that a downstream LLM can infer element type from chunk content, making the retrieval-layer alignment metric inconsequential. We address this position directly with two stricter measurements derived from the same retrieval data — both demonstrate that the retrieval-layer gap cannot be closed by what a generator does at read time, because the relevant signal is not present in the retrieved unit to begin with.

Measurement 1 — `type_match | hit` (type precision conditioned on coverage). For each query where the top-1 result is a recall hit, what fraction have the correct element type? This isolates type alignment from coverage: rather than "did we retrieve anything relevant?" the question becomes "given that we did, is it the right *kind* of thing?"

The 0.0% column is not an empirical measurement but an architectural ceiling: chunks have no element-type field to check, regardless of retrieval parameters.

Measurement 2 — `unit_hit@1` by query type. Even when the type is correct, is the *exact* gold CDU at rank 1 — not just *a* table, but *the* table the query refers to? Aggregated across all typed queries, Semantic IR achieves `unit_hit@1` = 0.583 (3,713 / 6,372 eligible queries). Per-type breakdown appears in Appendix C, Table C.2. Chunk backends produce no CDU identifier and are undefined on this metric.

(arch) = architectural ceiling, not a parameter-tunable quantity.

Finding 3. Conditioned on the condition most favourable to chunking — the top-1 is already a relevant hit — chunk systems still deliver the correct element type in 0.0% of cases and the exact right unit in 0.0% of cases. Semantic IR delivers 84.4% and 58.3% respectively. Both zeros are architectural ceilings on what the retrieval layer can supply as structured metadata. Whether a downstream generator can reconstruct element type from chunk content at read time is a separate question; at the retrieval layer, no such metadata is produced, so attribution and type-routing operations (§8.2) have nothing to consume.

Semantic IR's own non-unity `type_match | hit` and `unit_hit@1`. The 84.4% aggregate hides a gradient: `section` / `factual` queries score 97.2% / 96.5%, while `table` / `equation` / `code` queries score 47.6% / 27.7% / 29.5%. The same gradient appears in `unit_hit@1`. We call this *granularity-induced ranking bias*: a parent section outranks its child typed CDU at rank 1 even though the child is the gold unit, because in a shared embedding space the longer, prose-rich parent produces a richer embedding than a short equation (~30 chars), table (~80 chars), or code block. This is a real limitation of Semantic IR, but it is a *ranking* failure within a correct abstraction, not an architectural zero: the correct CDU is in the top-5 for 80% of queries (`unit_hit@5 = 0.800`) and the correct parent section is in the top-1 for 88% (`recall@1 = 0.879`). Addressing this rank-1 displacement is discussed in §8.3.

7 Supporting Characterisation

This section summarises three analyses that characterise when the main-text claim applies, what controls its magnitude, and how robust the measurement is to query-source choice. Each is developed in full in the appendix indicated.

7.1 When Does It Matter — Structural Richness (Appendix B)

Semantic IR's benefit is not uniform across documents: it scales with *CDU density*, the fraction of indexable content belonging to typed elements. On a 100-document subset of the corpus selected for balanced tier coverage (queries LLM-generated as in §6), the application-boundary metric `structural_coverage = typed_query_share` $\times$ `typed_etype_hit@1` — the fraction of *all* user queries that receive the correct element type — is monotonic in

Table 4: `type_match | hit` by query type (n = 5,730 typed queries; chunk backends are 0.0% by architectural construction).

Query type	n	chunk `tmH%`	**sem** `tmH%`
`section_qa`	2,068	0.0%	**97.2%**
`factual_qa`	2,226	0.0%	**96.5%**
`table_qa`	905	0.0%	**47.6%**
`equation_qa`	254	0.0%	**27.7%**
`code_qa`	277	0.0%	**29.5%**
aggregate	**5,730**	**0.0%**	**84.4%**

Table 5: Summary of what top-1 retrieval delivers.

Property of the top-1 retrieved unit	`hybrid`	`sem`
Right content in top-1 (`recall@1`)	**0.928**	0.879
Right type in top-1 (`etype_hit@1`)	0.000 (arch)	**0.737**
Right type given relevant (`type_match | hit`)	0.0% (arch)	**84.4%**
Exact right CDU (`unit_hit@1`)	0.000 (arch)	**0.583**
Exact right CDU in top-5 (`unit_hit@5`)	0.000 (arch)	**0.800**

structural richness: strong = 0.130, balanced = 0.091, weak = 0.021 (Appendix B, Table B.1). Raw per-tier `etype_hit@1` reverses this ordering due to query-mix composition (weak-tier documents have mostly `factual_qa` queries on which `etype_hit@1` $\approx$ 0.94), so `structural_coverage` is the correct application-boundary metric. Full per-domain and domain $\times$ tier breakdowns appear in Appendix B. *Practical takeaway*: deployment on a structure-poor corpus exercises the typed path on only $\sim$2% of queries; on a structure-rich corpus it reaches $\sim$13%.

7.2 Within a Correct Abstraction: Representation Fidelity (Appendix C)

Within a fixed typed abstraction, the quality of CDU extraction and projection matters. A controlled repair study on a 100-document probe (replacing raw parent-prose injection with a short synthetic caption in the table projection, removing parallel injection from code CDUs, and switching section chunking to sentence-boundary split) raises aggregate `etype_hit@1` by **+0.7pp** and table-specific `etype_hit@1` by **+1.7pp** with no algorithmic change. This separates the remaining rank-1 error into *granularity-induced ranking bias* (a ranking problem, §8.3) and *representation fidelity* (a projection-quality problem, addressed at the engineering layer). Full repair details appear in **Appendix C**.

7.3 Benchmark Validity: LLM-Query Structural Anchors (Appendix D)

In two paired pilots, LLM-generated benchmark queries over-report `etype_hit@k` relative to human-written queries, because LLMs echo structural vocabulary ("In the equation…", "According to Table…") absent in natural user questions: a 21-query single-paper pilot shows a -17.6pp gap, and a 75-query 5-document multi-domain pilot shows a -9.0pp gap. We adopt a write-question-first human benchmark protocol that prohibits structural hint words. The LLM-query alignment numbers in §6 should therefore be interpreted as upper bounds. The pilots are small-scale and the gap magnitude is corpus-dependent, but the *direction* — LLM queries over-aligning — is consistent across both pilots and across domains, and chunk-backend `etype_hit` is unaffected by query source. The protocol, pilot data, and validation checks appear in **Appendix D**.

8 Discussion

8.1 Parameter vs Architecture

`hybrid` achieves the highest aggregate recall@1 in our main result (0.928). Its `etype_hit@1` is nonetheless exactly 0, and no BM25 weight, chunk size, or embedding swap will change this. BM25 adds +11.75pp to recall@1 and +0.9pp to alignment (§6.2) — doing exactly what it should, on the wrong axis for this problem. Alignment and recall belong to different intervention classes: alignment lives at the indexing layer, recall at the ranking layer. The practical implication is that improving

RAG retrieval on structured documents is not a hyper-parameter problem.

8.2 Element Type as a Dispatch Key and Attribution Primitive

The retrieval-layer alignment metric matters beyond what a downstream generator can recover from raw text, for two reasons.

First, **element type as a dispatch key.** In a type-aware generation pipeline, element type is a routing signal: a table retrieval can be answered by a structured-cell extraction prompt, an equation retrieval by a LaTeX-interpretation prompt, a code retrieval by a code-execution-aware prompt. A chunk system with `element_type = null` cannot participate in such a pipeline because it has no key to dispatch on.

Second, `unit_hit@k` **as an attribution primitive.** A downstream model can infer type from text, but it cannot invent a stable unit identifier. Citation, attribution, and downstream update operations (e.g., "re-read the exact table the user last asked about") require a unit ID. Chunk identifiers are tied to character offsets and to chunking parameters (window size, overlap, tokenisation); they shift whenever any of these changes, or when the source document is edited. Typed CDU identifiers are tied to document elements (section index, equation number, table number) and remain stable under re-chunking and under document edits that do not change the element boundaries.

8.3 Granularity-Induced Ranking Bias and Routes Forward

The `unit_hit@1` $\rightarrow$ `unit_hit@5` gap of 21.7pp on Semantic IR is the signature of a ranking failure rather than a coverage failure: the correct CDU is in the top-5 for 80% of queries but wins rank 1 for only 58%. The per-type breakdown (Appendix C.2, Table C.2) localises the failure to short typed units (equation 17.1%, code 23.6%, table 43.0%) whose embeddings cannot outscore their parent sections under the same query.

Two complementary interventions are natural next steps: **parent-context enrichment** (augment short CDUs with controlled amounts of parent-section prose to raise their embedding signal without merging them into the parent), and **element-type routing** (classify the query type at runtime and restrict the candidate pool to the relevant typed sub-index, eliminating parent-section distractors). Both operate at the indexing/retrieval interface and require no change to the embedding model.

9 Conclusion

Chunk-based retrieval architecturally cannot achieve structural alignment, and standard recall evaluation cannot see this failure. We support this claim with three pieces of evidence: chunk backends score `etype_hit@1 = 0.000` on a 380-document, 6,383-query multi-domain corpus while a typed-CDU index scores 0.737; the largest retrieval-layer intervention (BM25 fusion) moves recall by +11.75pp and alignment by +0.9pp, demonstrating axis independence; and even conditioned on a recall hit, chunk systems still deliver the correct element type in 0.0% of cases and the exact gold unit in 0.0% of cases, while the typed index delivers 84.4% and 58.3%. The upper-right quadrant of the coverage $\times$ alignment design space (high recall *and* high alignment) is unattained in our experiments. Our contribution is to name the retrieval-layer metrics that make the failure visible and to show why no parameter sweep within the chunk paradigm can close the gap.

Limitations

Embedding model. All experiments use `bge-small-en-v1.5` (33M parameters). A larger model may shift absolute recall and alignment numbers but cannot move untyped chunk indexes off the architectural zero, since that zero is a property of the indexing abstraction, not the embedding.

Human-query scale. The structural-anchor analysis rests on two paired pilots (n=21 and n=75). Larger-scale human-annotated benchmarks for typed-element retrieval would strengthen the validity check; we adopt a write-question-first protocol (Appendix D) to make such scaling reproducible.

Document parsing. Structure recovery depends on parser quality; LaTeX and HTML parsing is clean, while finance/legal/medical PDF parsing is noisier. Representation fidelity (Appendix C) partly absorbs parse noise, but a controlled parser-quality $\times$ `etype_hit@1` study is left to future work.

References

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. 2023. Nougat: Neural optical understanding for academic documents. *arXiv preprint arXiv:2308.13418*.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024. Dense X retrieval: What retrieval granularity should we use? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

CReSt Benchmark Authors. 2025. CReSt: A benchmark for retrieval evaluation over structured documents. Citation placeholder; verify exact venue before camera-ready.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2023. RAGAS: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*.

FreshStack Authors. 2025. FreshStack: A developer knowledge base benchmark for retrieval and qa. Citation placeholder; verify exact venue before camera-ready.

Norbert Fuhr, Norbert Gövert, Gabriella Kazai, and Mounia Lalmas. 2002. INEX: Initiative for the evaluation of XML retrieval. In *Proceedings of the SIGIR 2002 Workshop on XML and Information Retrieval*.

Norbert Fuhr, Jaap Kamps, Mounia Lalmas, and Andrew Trotman. 2008. INEX 2007 evaluation measures. In *Focused Access to XML Documents: INEX 2007*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

GROBID contributors. 2008–2024. GROBID: A machine learning library for extracting, parsing, and re-structuring raw documents into structured TEI-encoded documents. https://github.com/kermitt2/grobid.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

IBM Research. 2024. Docling technical report. https://github.com/DS4SD/docling.

Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. 2023. FinanceBench: A new benchmark for financial question answering. *arXiv preprint arXiv:2311.11944*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jerry Liu. 2023. LlamaIndex: Parent-document retriever. Technical report and documentation, https://docs.llamaindex.ai.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. 2021. DocVQA: A dataset for VQA on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2023. ARES: An automated evaluation framework for retrieval-augmented generation systems. *arXiv preprint arXiv:2311.09476*.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*.

Andrew Trotman and Mounia Lalmas. 2006. Why structural hints in queries do not help XML-retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Ellen M. Voorhees. 2001. The philosophy of information retrieval evaluation. In *Workshop of the Cross-Language Evaluation Forum (CLEF)*.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Retrieval meets long context large language models. *arXiv preprint arXiv:2310.03025*.

Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. 2023. MIRACL: A multilingual retrieval dataset covering 18 diverse languages. *Transactions of the Association for Computational Linguistics*.

A Experimental Details

Embedding model. `BAAI/bge-small-en-v1.5` (33M params, 512 max tokens, 384-dim).

BM25. BM25S implementation, stemming enabled. Fusion: linear interpolation $s = (1 - \lambda)s_{\text{dense}} + \lambda s_{\text{BM25}}$, $\lambda = 0.3$.

GT-recall threshold. $\theta = 0.2$.

Chunk parameters. chunk_size = 1000 chars, chunk_overlap = 100 chars (unless varied in the granularity sweep, Appendix E).

CDU projection templates. Each CDU is projected to a synthetic retrieval string with a type-prefix tag, the section path, and a type-specific body. Newlines (\n) separate fields within a projection:

- Section: `[section | {heading path}]\n{content}`
- Equation: `[equation | {section path}]\n{LaTeX}`
- Table: `[table | {section path}]\n(caption or ‘‘table in: heading’’)\nHeader: col1 | col2 | ...\n{text rendering}`
- Code block: `[code | {language} | {section path}]\n{code content}`
- Figure: `[figure | {section path}]\n{caption}`

Corpus parsing. Scientific: arXiv LaTeX source via custom parser. Encyclopedia and web developer docs: HTML via BeautifulSoup. Finance/legal/medical: HTML where available, PDF via marker/nougat otherwise.

A.1 A.1 Query Generation

Queries are generated per-document by GPT-4o-mini from parsed CDU seeds. For each document, the generator is given the section list, the typed-element inventory (equations / tables / figures / code blocks with their captions or headers), and prompt templates per query type. Per-document caps prevent over-concentration on element-rich documents: `factual_qa` ≤ 5, `section_qa` ≤ 5, `table_qa` ≤ 3, `equation_qa` ≤ 2, `code_qa` ≤ 2, `cross_element_qa` ≤ 3, `adversarial_qa` ≤ 2 (the adversarial generator inherits a primary type and sets `is_adversarial=true`). `figure_qa` is not generated for this paper because our backends index only figure captions, not image content.

After generation, two filters run sequentially:

1. **Ground-truth validation.** Each query's `ground_truth` string is matched against the parsed CDU it claims as its gold source. Items where the token-overlap recall against the gold CDU falls below 0.40 are rejected (this is a generator-quality filter, distinct from the retrieval-evaluation $\theta = 0.20$).
2. **Leakage audit.** A two-stage filter (regex prefilter + LLM semantic review) flags structural cues in question text — direct mentions of `table`, `figure`, `equation`, `section`, `code`, anchor phrases like *“according to”* or *“as shown”*, and document-meta phrases like *“from the paper”* or *“the authors”*. Items flagged as `structural_leakage` are dropped; items flagged as `benign_reference` or `clean` are kept.

The main-text experiments use the `llm_native` profile (current generator without post-hoc rewriting). Appendix D pilots use the `human` profile (write-question-first; see §D.3). A third `llm_debiased` profile (regenerate-on-leak) exists in our pipeline but is not used in this paper.

A.2 A.2 Gold Unit ID Assignment

Each query carries a `gold_unit_ids` field listing the CDU identifiers that contain the answer. Assignment resolves the generator's `source_hint` string against the parsed JSON via a four-strategy cascade:

1. **Typed ID match.** Direct lookup if `source_hint` matches the schema `(eq|tab|cod|fig|sec)_NNN`.
2. **Legacy index match.** Hints of the form `TABLE_0`, `FIGURE_3` resolve by zero-indexed position in the parsed element list (legacy generator output).
3. **Heading fuzzy match.** For section queries, fuzzy-substring match against parsed section headings; accept if the ratio ≥ 0.85.
4. **Cross-section split.** Hints of the form `"A + B"` split on the separator and resolve each half independently.

Across `main_paper_380`, 99.8% of queries (6,372 of 6,383) receive a non-empty `gold_unit_ids`. The 11 unresolved queries are excluded from the `unit_hit@k` denominator (column `eligible` in Table 5).

A.3 A.3 Reproducibility

The four backends, the CDU schema, the metric implementations, the query generation pipeline, and the gold-ID assignment script are all implemented in a single research repository released alongside the paper. The 380-document parsed corpus, the 6,383-query benchmark with gold unit IDs, and the 75-query human-written multi-domain pilot are released as artefacts. End-to-end experiments are reproducible from a single `run_ablation.py` invocation per backend, with embedding caches keyed on the CDU projection strings to make reruns deterministic. Documents are drawn from public sources whose terms permit research redistribution: arXiv (CC-BY or arXiv non-exclusive license per paper), PubMed Central Open Access subset (CC-BY family), SEC EDGAR (public domain), Wikipedia (CC-BY-SA), Stanford Encyclopedia of Philosophy and official developer documentation (publisher-specific research-use terms), legislative sources (public record), and FinanceBench under its publication terms (Islam et al., 2023). We redistribute parsed JSON only; the released corpus and benchmark are intended for non-commercial research use, matching the most restrictive condition among the sources.

CDU density = typed-element character share of the document. Typed queries = table_qa + equation_qa + code_qa. Tier distribution: scientific S=57 B=34 W=4; developer S=28 B=36 W=11; finance S=3 B=52 W=5; legal S=3 B=35 W=22; medical S=18 B=25 W=7; encyclopedia S=21 B=1 W=3; reports S=0 B=7 W=8.

B Structural Richness as Conditioning Variable (Full Analysis)

B.1 B.1 CDU Density and Structural Tiers

We assign every document in `main_paper_380` to one of three structural tiers — **strong**, **balanced**, **weak** — by CDU density, where density is the typed-element character share. Tier is not a domain proxy: within each of the seven domains we observe documents across all three tiers (Table A.1).

B.2 B.2 Why Raw Per-Tier `etype_hit@1` Is Misleading

An initial approach is to report `etype_hit@1` stratified by tier. Doing so on the full corpus yields a *reversed* ranking:

This is a composition artefact. Weak-tier documents have few typed elements, so their queries are overwhelmingly `factual_qa` / `section_qa` — the two types on which Semantic IR scores ≈ 0.94. Strong-tier documents attract the structural queries (`table_qa`, `equation_qa`, `code_qa`) where the raw score is 0.29–0.55. The raw per-tier aggregate averages over *different query mixtures* and reports the wrong answer to the application-boundary question.

B.3 B.3 `structural_coverage` — The Correct Application-Boundary Metric

We define

$$\text{structural_coverage}(\mathcal{S}) = p_{\text{typed}}(\mathcal{S}) \times \text{etype_hit@1}_{\text{typed}}(\mathcal{S})$$

where p_{typed} is the typed-query share within tier/domain $\mathcal{S}$ and $\text{etype_hit@1}_{\text{typed}}$ is Semantic IR's `etype_hit@1` restricted to those queries. This is the correct application-level figure because it answers: *of all queries a user might send, what fraction does Semantic IR serve with the right kind of evidence?*

strong > balanced > weak: the application boundary is monotonic in structural richness.

B.4 B.4 Per-Domain Decomposition

`etype_hit@1` is highest where the document-element inventory is cleanest (reports, encyclopedia) and lowest where table-dense or equation-dense content creates parent/child competition (scientific, developer). This is consistent with §6.3's per-type analysis: `sci` and `dev` have the largest populations of short typed elements (equations, code blocks), which are exactly the units that lose the rank-1 slot to their parent sections.

B.5 B.5 Domain × Tier Cross-Tabulation

Across all 20 cells, high `etype_hit@1` and negative recall gain co-occur routinely. The two axes are orthogonal — improving structural alignment does not require winning on coverage.

C Representation Fidelity (CDU Repair Study)

Within a fixed typed abstraction, the quality of CDU extraction and projection matters. We ran

Table A.1: Per-domain corpus statistics (`main_paper_380`).

Domain	# docs	Sections	Tables	Equations	Code	Figures	Avg CDU density	# queries	Typed-query %
scientific	95	2,146	424	1,234	56	520	0.420	1,799	29.5%
developer	75	3,008	295	0	2,237	136	0.442	1,324	25.0%
finance	60	2,936	2,155	0	0	83	0.267	1,044	24.7%
legal	60	5,390	3,060	0	0	68	0.305	914	17.4%
medical	50	931	179	16	0	211	0.271	757	10.8%
encyclopedia	25	895	370	0	1	777	0.458	388	18.6%
reports	15	233	4	0	0	35	0.150	157	1.9%
Total	**380**	**15,539**	**6,487**	**1,250**	**2,294**	**1,830**	—	**6,383**	**22.5%**

Table A.2: Query-type distribution (`typed_queries_380`).

Query type	n	Gold element
`factual_qa`	2,226	section (prose passage)
`section_qa`	2,068	section (heading-scoped)
`table_qa`	905	table
`cross_element_qa`	653	multi-CDU
`code_qa`	277	code_block
`equation_qa`	254	equation
Total	**6,383**	—

Table B.1: Raw per-tier `etype_hit@1` reverses the application-boundary ordering (a query-mix composition artefact; see text).

Tier	`sem etype_hit@1` (raw)
strong	0.7106
balanced	0.7413
weak	0.8129

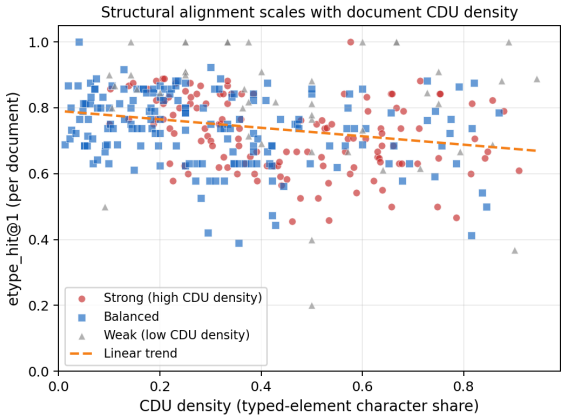

Figure B.1: CDU density scatter. x = CDU density per document; y = Semantic IR `etype_hit@1` per document; colour by tier (strong / balanced / weak). The linear trend confirms a monotonic positive relationship across the 349 documents with valid density values.

a controlled probe on `main_paper_human_100` (n = 1,635 queries) comparing Semantic IR before and after a series of CDU-representation repairs, without changing the retrieval algorithm.

C.1 C.1 Repairs Applied

1. **Table CDU — header row extraction.** Parse markdown pipe / LaTeX tabular / plain text headers and render them as `Header: col1 | col2 | ...` in the projection so table embeddings carry column-name signal.
2. **Table CDU — section-path in projection.** Use the full section-path (not just immediate heading) in the `[table][path]` prefix to provide richer topical context.
3. **Table CDU — caption fallback.** When caption is empty, project `(table in: {heading})` so no table unit is semantically anonymous.
4. **Table CDU — synthetic caption.** An earlier attempt injected `Context: {parent_prose[:200]}` into the table projection. This regressed table `etype_hit@1` by -12pp because the parent-prose embedding made the table representation drift into section space, causing section CDUs to displace tables at rank 1 in 29 out of 237 `table_qa` queries. Replacing the injection with `Caption (inferred): {first_sentence[:80]}` — a short caption-shaped string rather than free prose — preserves topical signal while keeping the embedding in table space.
5. **Code CDU — revert context injection.** The same parallel "inject parent context" had been added to code CDUs and produced the same drift. Reverting `context=""` recovers code `etype_hit@1`.
6. **Section CDU — sentence-boundary split.** Replace fixed-offset section chunking with sentence-boundary split, reducing the mid-sentence cut rate from 72.7% to 28.1% and improving section CDU coherence.

Table B.2: Structural coverage by tier on the 100-document human-curated probe.

Tier	p_{typed}	typed `etype_hit@1`	`structural_coverage`
strong	0.281	0.462	**0.1298**
balanced	0.184	0.493	0.0908
weak	0.085	0.251	0.0213

Table B.3: `etype_hit@1` and recall@1 by domain (`sem` vs `hybrid`).

Domain	`sem etype_hit@1`	`sem recall@1`	`hybrid recall@1`
reports	**0.8981**	0.9172	0.9363
encyclopedia	**0.8222**	0.9021	0.9124
medical	**0.8137**	0.8811	0.9538
legal	**0.7637**	0.8676	0.9037
finance	**0.7107**	0.8487	0.8793
developer	**0.7175**	0.9282	0.9411
scientific	**0.6887**	0.8583	0.9494

C.2 C.2 Results

† Per-query audit of the 6 lost `code_qa` hits confirms all 6 are in developer documentation (Go stdlib, FastAPI, Next.js) and one scientific NLP paper. In each case, sentence-boundary split produced a more coherent prose section describing the code concept, which displaced the `code_block` CDU at rank-1. This is granularity-induced ranking bias (§8.3), not a representation-fidelity regression: the code content is unchanged; the surrounding prose section is now a stronger embedding match.

C.3 C.3 What the Repair Study Does and Does Not Show

The study separates the remaining rank-1 error in Semantic IR into two sub-problems:

- **Representation fidelity** (addressed here): +0.7pp aggregate, +1.7pp on tables, +2.2pp on factual. Addressable by projection-quality engineering within the current abstraction.
- **Granularity-induced ranking bias** (§8.3): the negative deltas on equation and code queries. Not addressable by projection repair — the parent section is simply a stronger embedding match than the child CDU once the split makes the section more coherent. Requires routing or enrichment.

The `equation_qa` and `code_qa` columns of Table C.1 are not negative findings about representation fidelity; they are localised manifestations of ranking bias. For the shortest typed units, improving projection is insufficient because the parent section is not being out-competed in projection quality — it is being out-competed in embedding richness after sentence-boundary coherence is applied.

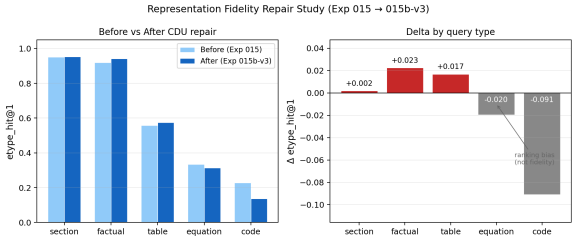

Figure C.1: Representation fidelity repair. Left: grouped before/after bars for `etype_hit@1` by query type. Right: Δ bars; the table $+1.7$pp and factual $+2.2$pp gains are annotated, equation/code negative deltas labelled as ranking bias.

C.4 C.4 `unit_hit@1` by Query Type

Per-type breakdown of the identity-level metric reported aggregate in §6.3. The same granularity-induced ranking bias described in §8.3 explains the gradient: short typed units (equation, code, table) lose rank-1 to their parent sections more often than prose-rich section / factual queries.

D Structural Anchors and the Human QA Benchmark

D.1 D.1 The Structural-Anchor Effect

All queries in the main text are LLM-generated. Inspection reveals that GPT-4o-mini frequently produces structural hints:

> "In the scaled dot-product attention equation, what is the role of $\sqrt{d_k}$?" "According to Table 5, which model achieves the best performance on WMT EN→DE?"

Table B.4: `sem etype_hit@1` and recall@1 gain vs `hybrid` by domain × tier. Domain prefixes: `develo` = developer, `encycl` = encyclopedia, `financ` = finance, `medica` = medical, `report` = reports, `scient` = scientific; tier suffixes S/B/W = strong/balanced/weak.

domain/tier	eH@1	R@1 gain	sem R@1	hybrid R@1	n
develo/B	0.7378	-0.0195	0.9218	0.9414	614
develo/S	0.6661	-0.0105	0.9283	0.9388	572
develo/W	0.8406	+0.0072	0.9565	0.9493	138
encycl/B	0.9231	+0.0000	1.0000	1.0000	13
encycl/S	0.8090	-0.0112	0.8961	0.9073	356
encycl/W	1.0000	+0.0000	0.9474	0.9474	19
financ/B	0.7101	-0.0434	0.8436	0.8871	921
financ/S	0.8246	-0.0526	0.8947	0.9474	57
financ/W	0.6212	+0.1667	0.8788	0.7121	66
legal/B	0.7491	-0.0447	0.8711	0.9158	582
legal/S	0.7255	+0.0000	0.9608	0.9608	51
legal/W	0.8007	-0.0249	0.8434	0.8683	281
medica/B	0.8253	-0.0672	0.9032	0.9704	372
medica/S	0.8098	-0.0820	0.8590	0.9410	305
medica/W	0.7750	-0.0625	0.8625	0.9250	80
report/B	0.8851	-0.0115	0.9425	0.9540	87
report/W	0.9143	-0.0286	0.8857	0.9143	70
scient/B	0.7081	-0.0738	0.8809	0.9547	596
scient/S	0.6698	-0.1011	0.8436	0.9447	1,157
scient/W	0.9130	-0.0652	0.9348	1.0000	46

Table C.1: Representation fidelity repair (`sem` on `main_paper_human_100`, n = 1,635). "Before" = original CDU projection; "after" = projection with the six repairs in §C.1.

Query type	n	`etype_hit@1` (before)	`etype_hit@1` (after)	Δ
`section_qa`	533	0.9512	**0.9531**	+0.2pp
`factual_qa`	579	0.9188	**0.9413**	**+2.2pp**
`table_qa`	237	0.5570	**0.5738**	**+1.7pp**
`equation_qa`	51	0.3333	0.3137	-2.0pp
`code_qa`	66	0.2273	0.1364	-9.1pp †
`cross_element_qa`	169	0.0000	0.0000	—
aggregate	**1,635**	**0.7358**	**0.7425**	**+0.7pp**

These *structural anchors* directly reference the element type the answer lives in and artificially align with Semantic IR's typed units.

D.2 D.2 Pilot Measurements

Chunk-based systems are unaffected (`etype_hit` = `0` regardless of query source; the gap is a Semantic-IR-specific artefact of structural vocabulary cueing typed retrieval).

D.3 D.3 Write-Question-First Protocol

All human queries follow the *write-question-first* protocol: the annotator picks a concept to ask about, writes a natural question as a reader would phrase it, then locates the ground truth in the document. The reverse order — open a table, write a question about that table — is the failure mode that produces structural anchors. Question text is constrained as follows:

Prohibited vocabulary. `table`, `figure`, `equation`, `formula`, `expression`, `mathematically`, `row`, `column`, `code`, `function`, `class`, `method` (when naming a specific code identifier), `section`, `chapter`, `the paper`, `the authors`, anchor phrases such as `according to` / `as shown` / `above` / `below`. The first four phrasings name an element type; the last group leaks document or positional structure.

Per-type rules with examples (excerpted from the annotator guideline; full version released with the benchmark).

- *equation_qa* — never name the equation type.
  - ✓ "How does the model's training loss change as more parameters are added?"
  - × "What is the scaling-law equation for compute-optimal training?"
- *table_qa* — never name the table or its row/column structure.
  - ✓ "Which benchmark does LLaMA-2

Table C.2: `unit_hit@1` by query type for Semantic IR on `main_paper_380`. *Eligible* is the number of queries with a non-empty gold `unit_id`; the small drop from n for table / equation / code reflects the 0.2% of queries whose generator-emitted `source_hint` did not resolve to a CDU.

Query type	n	eligible	`unit_hit@1`	%
`section_qa`	2,068	2,068	1,391	**67.3%**
`factual_qa`	2,226	2,226	1,465	**65.8%**
`table_qa`	905	899	387	**43.0%**
`equation_qa`	254	251	43	**17.1%**
`code_qa`	277	275	65	**23.6%**
`cross_element_qa`	653	653	362	**55.4%**
aggregate	**6,383**	**6,372**	**3,713**	**58.3%**

Table D.1: Structural anchor gap across two pilots.

Query source	Corpus	n	`sem etype_hit@1`
LLM-generated	sci (single paper)	25	0.650
Human-written	sci (single paper)	21	**0.474**
Δ	—	—	**-17.6pp**
LLM-generated	5 docs, 3 domains	110	0.741
Human-written	5 docs, 3 domains	75	**0.652**
Δ	—	—	**-9.0pp**

perform best on compared to other models?"
  - × "According to Table 5, which benchmark shows the highest improvement?"

- *code_qa* — never name the specific function or class.
  - ✓ "How do you implement an iterator that counts down in Python?"
  - × "What does the `Reverse.__iter__` method return?"

- *factual_qa* — single fact, no bundling.
  - ✓ "What dropout rate was used during base model training?"
  - × "What dropout rate and batch size were used during training?" *(two facts)*

Ground-truth format. Verbatim canonical form from the source. For equations, exact LaTeX/notation as it appears (`d_k`, not `dk`). For tables, the exact cell value with enough context to be unambiguous (model name, dataset, units). For code, the verbatim snippet if ≤ 10 lines, otherwise a precise description.

Query-type distribution per document. `equation_qa` ≈ 3, `table_qa` ≈ 3, `code_qa` ≈ 3 (developer docs only; replaces `equation_qa`), `factual_qa` ≈ 3, `section_qa` ≈ 2, `cross_element_qa` $\approx$ 1–2, adversarial $\approx$ 2–3. Total: 14–17 per document.

D.4 D.4 Leakage Audit

Both human-written and LLM-generated query sets pass through the same two-stage leakage audit:

1. **Regex prefilter** (`check_query_leakage.py --mode regex`). High-recall keyword and phrase matcher over the question text; flags any item whose wording contains the prohibited vocabulary listed in §D.3.
2. **LLM semantic review** (`--mode llm`). GPT-4o-mini scores each flagged item into one of `structural_leakage`, `benign_reference`, `adversarial_structural_cue`, `clean`, `uncertain`. Only `structural_leakage` is treated as a hard reject in the human profile; the LLM-generated benchmarks retain leaked items so that the structural-anchor effect remains measurable rather than filtered away.

The audit operates on question text only; it does not modify `source_hint` or `gold_unit_ids`.

D.5 D.5 Implication for the Field

Based on our two pilots, structured-retrieval benchmarks that rely exclusively on LLM-generated queries should be interpreted as upper bounds on alignment rather than ground truth. The multi-domain pilot in Table D.1 suggests the human-query `etype_hit@1` corresponding to our 380-document LLM-query value of 0.737 is approx-

Table E.1: Naive chunk granularity sweep.

c	recall@1	MRR
200	0.632	0.749
500	0.882	0.904
1000	0.882	0.915
1500	**0.897**	**0.923**
2000	0.882	0.907

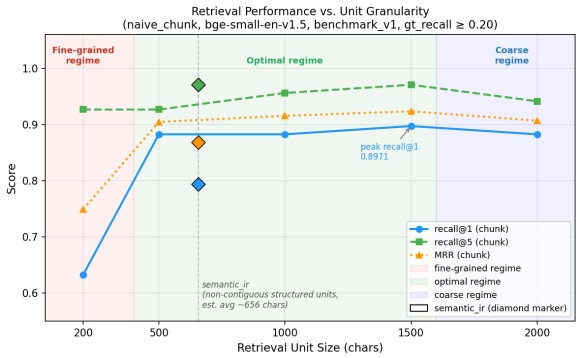

Figure E.1: Granularity curve. x = chunk/section size; left y-axis = `recall@1` (inverted-U on chunk, flat on Semantic IR); right y-axis = `etype_hit@1` (chunk 0 across all sizes, sem flat at 0.47–0.51).

imately 9pp lower (chunk backends are unaffected, since the architectural zero is query-source-independent). Larger-scale paired human/LLM comparisons would strengthen this estimate; as a best practice, we suggest reporting both LLM- and human-query numbers and clearly labelling which is which.

E Granularity Sweep

We evaluate `naive` at chunk sizes $c \in \{200, 500, 1000, 1500, 2000\}$ on an early small-scale benchmark (n = 68); the table is retained for the shape of the curve, not as a main result.

Coverage follows an inverted-U over chunk size, optimum around 1,000–1,500 chars. Smaller chunks fragment evidence; larger chunks over-aggregate.

We additionally sweep Semantic IR over `max_section_chars` $\in \{200, 500, 1000, 1500, 2000\}$: `etype_hit@1` is flat at 0.47–0.51 with no trend. Alignment is invariant to granularity within the typed abstraction, because it is determined by the typed-element inventory, not by section-split size. This reinforces the §6.2 finding that alignment is architectural rather than parametric.