Retrieval-Augmented Generation and Multi-Agent Arbitration for Construction Code Compliance

Hao Yin[1], Xichen Chen[2], Yang Zou[3], and Liupengfei Wu[4]

[1]Department of Computing, Faculty of Computer and Mathematical Sciences,
The Hong Kong Polytechnic University, Hong Kong SAR, China
[2]School of Future Environments,
Auckland University of Technology, Auckland, New Zealand
[3]School of Architecture, Building and Civil Engineering,
Loughborough University, Loughborough, UK
[4]Division of Industrial Data Science, School of Data Science,
Lingnan University, Hong Kong SAR, China

hi-bruce.yin@connect.polyu.hk, xichen.chen@aut.ac.nz, y.zou2@lboro.ac.uk, mikewu@ln.edu.hk

Abstract -

Construction code retrieval is a recurring bottleneck in AEC compliance workflows, where decisions require clause-addressable evidence and cross-discipline reference completion under incomplete project attributes. We present a clause-centric retrieval-augmented generation (RAG) pipeline that treats retrieval as auditable evidence discovery and enforces an evidence-bounded selection contract: outputs may cite only clause identifiers from the candidate set C_k with evidence spans anchored to retrieved text for mechanical validation. A trigger-based multi-agent orchestrator invokes an arbiter only under expert disagreement or validation flags. Experiments on a labelled query set (dev200/test100/all300) over a 600-code corpus (141681 clause units) show that, at the original top-10 operating point on dev200, dense retrieval reaches Hit@1=0.550 versus 0.210 for BM25. A retrieval-budget sensitivity analysis over $k \in \{1, 3, 5, 10\}$ further shows that dense gold-in-C_k rises from 0.550/0.500 at $k = 1$ to 0.935/1.000 at $k = 10$ on dev200/test100, while post-hoc feasibility audits of archived selector and arbitration outputs indicate non-linear downstream gains as candidate pools deepen. Under identical evidence inputs, evidence-bounded selection favours cost-efficient models (Qwen-plus Hit@1=0.430; USD 0.896 per 1K queries) over higher-cost alternatives (GPT-4o Hit@1=0.345; USD 5.763 per 1K queries). On held-out tests, arbitration is triggered in 37% of cases and revises the majority decision in 32.4% of triggered cases. The resulting clause packages link design decisions to regulatory clauses with auditable evidence, improving accountability and reducing manual clause hunting. Prototype code and experimental data are released at `https://github.com/bruce0210/RagConstructionAssistant`.

Keywords -

Construction Codes; Clause-Level Retrieval; RAG; Evidence-Bounded Generation; Multi-Agent Arbitration

1 Introduction

Construction code clause retrieval is a routine step in design review and compliance checking [1]. Reviewers typically need clause-addressable evidence, an explicit statement of scope and exceptions, and a compact bundle of cross-referenced clauses that can span multiple disciplines. When project attributes that govern applicability are missing, reliable practice favours an auditable set of candidate clauses with stated assumptions, rather than a single confident answer.

Large language models (LLM) and RAG offer a natural interface for code look-up, but many AEC assistants remain answer-centric: retrieved passages are used as supporting material, and citations are not enforced as mechanically verifiable links to a bounded evidence pool [2, 3, 4]. Moreover, missing scope attributes and cross-discipline coupling can yield multiple plausible clauses; deployment therefore needs governance that surfaces disagreement and evidence gaps for triage instead of hiding them in a single response.

To address these constraints, a clause-centric RAG pipeline is introduced that treats retrieval as evidence discovery and keeps downstream reasoning evidence-bounded. Codes are unitised into clause evidence objects with stable identifiers and provenance; retrieval returns a fixed candidate set C_k; discipline experts produce structured clause packages with applicability labels and evidence spans; and an arbiter is invoked only under explicit disagreement or evidence-quality violations. Figure 1 overviews the end-to-end pipeline.

The pipeline is evaluated on a labelled six-discipline query set with clause-identifier ground truth. Tables 3–5 summarise fixed-point retrieval, selection, and arbitration results at the original $k = 10$ operating point, while Section 5.5 and Figure 4 report retrieval-budget sensitivity and post-hoc feasibility audits for archived downstream outputs.

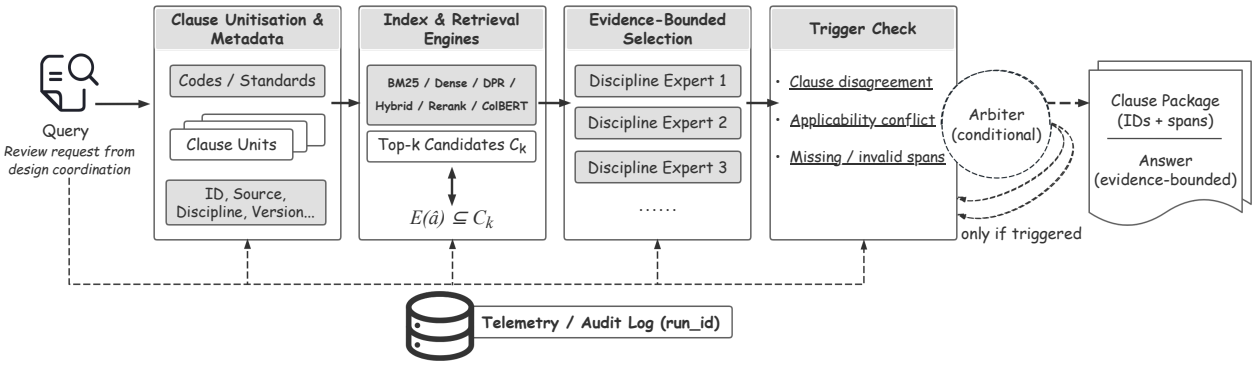

Figure 1. Auditable clause-centric RAG pipeline

The contributions are threefold. First, a clause-centric retrieval unit is defined for construction code retrieval and compliance checking, where each candidate is a citeable clause with stable identifiers and provenance metadata for plan-checking, design coordination, and cross-discipline review. Second, an evidence-bounded clause-selection contract (Formula 1) is enforced through structured validation: returned clause IDs must lie in C_k, and evidence spans must map to retrieved text. Third, a trigger-based multi-agent arbitration mechanism addresses cross-discipline ambiguity by invoking an arbiter only when experts disagree on clause/applicability or evidence is missing; trigger rates and outcomes are reported on dev200/test100/all300.

2 Literature Review

Construction code consultation sits at the intersection of regulatory knowledge management and decision support. Unlike general information seeking, the expected output is reviewable evidence traceable to specific clauses and defensible under audit [5]. This section therefore reviews clause-oriented compliance support, RAG for AEC knowledge services, and reliability and governance for LLM-assisted decision support.

2.1 Clause-Level Evidence Traceability

Digital compliance support in AEC has long recognised that regulatory text is structured by scopes, exceptions, and cross-references that affect applicability [6]. Traditional rule-checking approaches often rely on formalised representations aligned with building models [7], which can be effective when attributes are complete and scope is well specified, but usually require substantial authoring effort and remain brittle to code updates and local interpretations [8].

For practical construction retrieval, engineers need to locate the governing clause, confirm scope and exceptions, and assemble a minimal evidence bundle for review. The main bottleneck is therefore evidence traceability at clause granularity. Document-level retrieval or paragraph-level quoting can obscure the exact normative unit supporting a decision and complicate both evaluation and review workflows [9]. This motivates clause-addressable evidence units with stable identifiers and provenance, making the evidence object explicit, mechanically checkable, and easier to cite and track in review.

2.2 RAG for AEC Knowledge Services

RAG-style systems have been explored for AEC knowledge search over project documents, specifications, standards, and BIM-linked information [2, 3, 10]. They improve topical relevance and reduce reliance on parametric memory, but construction codes expose two limitations of generic RAG patterns: the review-relevant unit is usually a clause or subclause rather than an arbitrary chunk, and retrieval must account for applicability under scope conditions and cross-discipline coupling. Large passages can mask these constraints, while small chunks without stable identifiers struggle to provide review-ready citations.

2.3 LLM Reliability and Governance

Reliability in LLM-assisted decision support is often addressed with citations, structured outputs, and verifier-style checks, but compliance consultation requires a stronger guarantee than merely "having citations": cited evidence must remain restricted to, and traceable within, a bounded retrieved pool. This motivates a mechanically checkable evidence boundary (Formula 1) and span-level evidence tied to concrete text. Multi-agent designs are most useful here as governance rather than blanket ensembling: disagreement and evidence-quality violations should surface as operational risk signals that trigger conditional scrutiny, consistent with engineering review workflows and judge-style arbitration frameworks [11].

2.4 Summary of Gaps

Existing AEC compliance support and RAG systems rarely provide review-ready, clause-addressable evidence packages. Many rely on document or chunk retrieval and allow models to cite beyond a fixed retrieved pool, which weakens verifiability and auditability [4]. Ambiguity from missing scope attributes and cross-discipline coupling is also often absorbed into a single response, obscuring failure modes and governance needs under cost and latency constraints.

These gaps motivate the clause-centric evidence design, evidence-bounded selection contract, and trigger-based arbitration protocol introduced in Section 3.

3 Methodology

This section specifies a clause-centric RAG workflow for construction-code clause retrieval in compliance review, treating retrieval as auditable evidence discovery and constraining clause selection with an evidence-bounded contract. Figure 1 summarises the end-to-end pipeline.

3.1 Overview and Evidence-Bounded Formulation

Given a natural-language query q, retrieval returns a bounded candidate set of code clauses $C_k = \{c_1, \ldots, c_k\}$ from a clause-indexed corpus, and a selector produces the final clause set supporting the response. An evidence-bounded contract is enforced: the generated answer $\hat{a}$ may cite only clauses inside C_k. Let $\mathcal{E}(\hat{a})$ be the set of clause identifiers cited as evidence in $\hat{a}$, then

$$\mathcal{E}(\hat{a}) \subseteq C_k. \tag{1}$$

This constraint makes citations verifiable and bounds uncontrolled generation when project attributes are underspecified. Applicability in construction codes is shaped by scope conditions, exceptions, and cross-references; accordingly, clause unitisation provides citeable units with stable identifiers and provenance metadata, expert outputs include applicability labels and evidence spans anchored in C_k, and arbitration is triggered only when clause/applicability disagreement or invalid evidence spans indicates elevated ambiguity.

3.2 Clause Unitisation and Corpus Schema

Instead of treating each standard as an unstructured document, the corpus is represented as clause units [12]. Each clause unit stores (a) its canonical text, (b) a stable internal identifier (clause_id) used for indexing and evaluation, and (c) structured metadata such as clause number (clause_no), source document identifier, and optional media references (e.g., figures extracted from the source). This representation supports clause-addressable citations and allows downstream components to operate on comparable evidence granularity across different codes and disciplines. Clause boundaries are produced by a lightweight rule-based structural parser that detects clause-numbering patterns and common chapter/article/appendix markers, then normalises numbering and merges continuation lines to form a single clause unit. Unitisation can introduce noise in cases of inconsistent numbering, scanned layouts, or table/figure regions; therefore, raw source text is retained as provenance and spot checks are used to validate clause_id–text alignment.

3.3 Candidate Retrieval for Auditable Evidence

The retrieval stage maps q to a bounded candidate set C_k by ranking clause units using one or more retrieval engines. Each candidate $c_i \in C_k$ is packaged as an evidence item with its clause_id, clause_no, source, and a minimal text span sufficient for inspection. This evidence packaging is the only admissible evidence pool for downstream selection under the contract in Formula 1.

Construction-code queries commonly combine domain terminology with scope constraints, exceptions, and cross-discipline coupling; therefore, retrieval is framed as evidence discovery rather than a single best-match search.

3.4 Evidence-Bounded Clause Selection

Given q and C_k, each discipline expert performs clause selection and returns a structured clause package rather than free-form text. The output is defined as $o_j(q) = (S_j, A_j, \Sigma_j)$, where S_j is a short ranked list of selected clause identifiers, A_j is an applicability label for the query as stated, and Σ_j are evidence spans anchored in the retrieved candidate texts. Requiring Σ_j operationalises the evidence-bounded contract in Formula 1 and supports mechanical checks (e.g., whether cited spans fall inside C_k and whether a claimed selection is backed by explicit text), providing governance signals for downstream arbitration and error analysis.

3.5 Trigger-Based Arbitration and Governance

Arbitration serves as a governance step for ambiguity-prone cases rather than a default refinement stage, and it is invoked if the parameter-free trigger $\tau(q) = 1$ in Formula 3 holds: experts disagree on clause IDs or applicability, or schema/evidence checks fail ($\nu_j(q) = 0$). For example, if experts return $c_a \in C_k$ with applicability "yes" and $c_b \in C_k$ with "no", the arbiter stays within C_k, rejects any package whose clause IDs or spans fail validation, and then retains the package whose anchored span more directly supports the queried scope condition or exception. Thus, a scope-aligned clause may replace the majority top clause while keeping the final package

evidence-bounded; see Table 6 for an arbiter-changed example. Given $\{o_j(q)\}$, the arbiter outputs a final clause package by prioritising evidence-supported clauses, resolving cross-discipline references, and discarding selections not anchored to C_k, keeping the decision evidence-bounded and auditable.

$$m(q) = \text{mode}\big(\{s_j(q)\}_{j=1}^{M}\big), \tag{2}$$

$$\tau(q) = \mathrm{I}\Big[(\exists\, j,l:\ s_j(q) \neq s_l(q)) \vee (\exists\, j,l:\ a_j(q) \neq a_l(q)) \vee (\exists\, j:\ \nu_j(q) = 0)\Big], \tag{3}$$

$$\chi(q) = \mathrm{I}[s^*(q) \neq m(q)]\,, \tag{4}$$

where $\mathrm{I}[\cdot]$ is the indicator function and $s^*(q)$ is the arbiter-selected clause when $\tau(q) = 1$. Formulas 2–4 define the majority top clause, trigger indicator, and arbiter-change flag.

3.6 Evaluation Protocol and Metrics

Evaluation adopts dev200 for calibration and test100 for final reporting. For each query q with a gold set of relevant clause identifiers $G(q)$, retrieval quality is measured by Hit@k on identifiers, together with MRR and nDCG@k. For the multi-expert stage, the majority selector is compared with the trigger-based arbiter output; Hit@1 is reported overall and on the triggered/changed subsets, and the arbitration accuracy delta is defined as $\Delta\text{Hit@1} = \text{Hit@1}_{arb} - \text{Hit@1}_{maj}$ (overall and triggered) under the same evidence pool. Governance rates over a query set Q are defined in Formula 5.

$$\begin{aligned} \text{TrigRate} &= \frac{1}{|Q|}\sum_{q\in Q}\tau(q), \\ \text{ChangeRate} &= \frac{1}{|Q|}\sum_{q\in Q}\tau(q)\chi(q), \\ \text{Change} \mid \text{Trig} &= \frac{\sum_{q\in Q}\tau(q)\chi(q)}{\sum_{q\in Q}\tau(q)}. \end{aligned} \tag{5}$$

4 Prototype Implementation

This section describes a working prototype that operationalises the methodology in Section 3. The implementation emphasises auditability, reproducibility, and modular replacement of retrieval and model components. Figure 2 shows the system architecture and data flow.

4.1 System Architecture and Data Flow

A query q enters the pipeline as a retrieval request. The retrieval layer ranks clause units and returns a bounded candidate set C_k as a packaged evidence bundle (clause identifiers, source references, and minimal inspectable text spans). The orchestration layer applies the evidence-bounded selector under Formula 1; if trigger conditions indicate clause/applicability disagreement or evidence violations, arbitration is invoked with q, C_k, and expert outputs. The system returns a clause-addressable package and a concise explanation bounded to cited evidence, while emitting structured telemetry keyed by a run identifier for replay and error analysis, as shown in Figure 2.

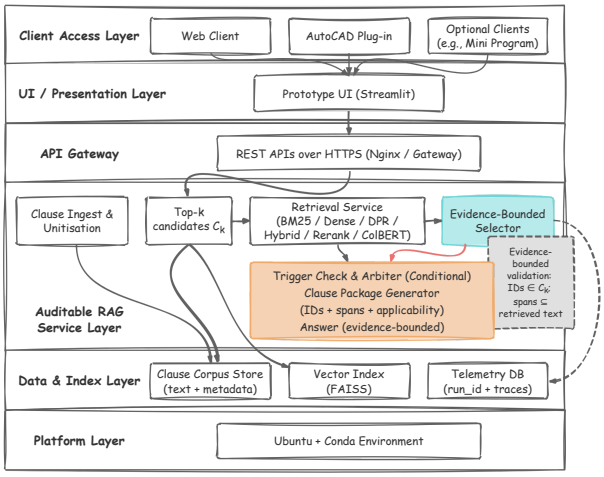

Figure 2. System architecture of the auditable clause-centric RAG prototype. Evidence-bounded validation checks that cited clause IDs belong to C_k and evidence spans are anchored to the retrieved clause text; violations raise governance triggers.

4.2 Web Interface and Telemetry Logging

The web UI prioritises evidence inspection: retrieved candidates are rendered with clause identifiers and provenance, and the final output is a clause package with explicit clause IDs and evidence spans for rapid cross-checking. All stages are recorded in a relational telemetry store for replay and audit, capturing the query, retrieval configuration, candidate-set hashes, model settings, and the final clause package. Table 1 summarises the core telemetry entities captured for reproducible experiments.

4.3 Arbitration Service and Prompt Orchestration

The orchestration layer separates multi-agent expert selection from arbitration under strict input-output schemas. A lightweight validator enforces the evidence-bounded contract by checking that cited clause identifiers belong to C_k and that evidence spans align with the returned candidate texts; arbitration is invoked only when the trigger rules

Table 1. Telemetry artifacts captured for auditable runs

Artifact	Stored in	Audit role
Run record	`trace_logs` (PostgreSQL)	Groups a run and links query, configs, and outputs via `run_id`.
Candidate pool	`*_details.jsonl`	Reproduces retrieval candidates for offline metrics and error analysis.
Expert outputs	`*.agents.jsonl`	Inspects discipline votes, applicability labels, and evidence spans.
Arbitration trace	`*.adjudication.jsonl`	Explains trigger reasons and produces the final clause package.

in Section 3.5 are satisfied. Prompts use a shared schema containing the query, minimal context, bounded candidates C_k (clause id/provenance/spans), and a fixed output contract (selected IDs, applicability, anchored spans). When triggered, the arbiter receives q, C_k, and the expert clause packages, and returns a final clause package with clause identifiers, aligned evidence spans, and an applicability judgement.

4.4 CAD Integration Entry Point

To integrate construction-code clause retrieval into drawing production and coordination, the prototype provides a lightweight CAD entry point for embedded consultation inside AutoCAD. A practitioner can invoke the custom command `"RAGCA"` during drafting to query relevant code clauses without leaving the drawing environment. The server-side pipeline returns the same evidence bundle and clause package as the web interface, keeping results auditable under the evidence-bounded contract and supporting compliance checking during drawing-based coordination. Figure 3 illustrates the within-CAD consultation flow.

5 Experiments and Results

This section reports fixed-point retrieval, selection, and arbitration results at the original $k = 10$ operating point, and then adds a retrieval-budget sensitivity analysis to examine how stricter candidate budgets affect reachability and downstream decision feasibility.

5.1 Datasets and Experimental Setup

Evaluation uses a labelled six-discipline query set (architecture, structure, heating, ventilation, and air conditioning (HVAC), plumbing, electrical, fire), with dev200 for calibration, test100 for held-out assessment, and all300 for their union. Retrieval operates over a clause-indexed corpus of 600 construction codes (141681 clause units); within each comparison, all methods share the same candidate budget k to ensure comparable evidence inputs.

Table 2. Dataset statistics and split configuration. Primary discipline is used for each query. "Cross" denotes integrated multi-discipline scenarios. Single-discipline queries = $\#Q - \#Cross$.

Split	#Q	#Cross	Arch.	Str.	HVAC	WSD	Elec.	Fire
Dev200	200	36 (18.0%)	31	30	35	29	39	36
Test100	100	24 (24.0%)	19	20	15	21	11	14
All300	300	60 (20.0%)	50	50	50	50	50	50

Abbreviations: Arch.=Architecture; Str.=Structure; WSD=Water supply and drainage; Elec.=Electrical; Fire=Fire protection.

Table 3. Clause retrieval results under a fixed top-10 candidate budget (dev200, $n = 200$)

Method	Hit@1	Hit@5	Hit@10	nDCG@10
Dense	**0.550**	**0.850**	**0.935**	**0.691**
Hybrid	0.385	0.725	0.820	0.504
Rerank	0.315	0.600	0.760	0.414
DPR	0.305	0.585	0.685	0.370
ColBERT	0.290	0.535	0.660	0.351
BM25	0.210	0.460	0.565	0.267

Main selector and arbitration comparisons use top-k candidates with $k = 10$ as the original operating point. This supports controlled comparison but does not by itself characterise retrieval-depth sensitivity. Increasing k would generally improve upper-bound recall, but also increases evidence inspection cost and can dilute top-rank precision. Section 5.5 therefore analyses $k \in \{1, 3, 5, 10\}$ using exact dense gold-in-C_k rates on dev200/test100 and post-hoc audits of archived $k = 10$ selector/arbitration outputs under stricter prefix pools C_k. Table 2 summarises split statistics and discipline coverage.

5.2 Retrieval Baseline Comparison

Retrieval baselines rank clause units to form the bounded evidence pool for downstream selection. Stronger retrieval raises the upper bound of evidence-bounded selection, although final correctness still depends on accurate clause selection.

Table 3 reports retrieval effectiveness on dev200 under a fixed top-10 candidate budget. Dense retrieval achieves Hit@1=0.550 and nDCG@10=0.691, versus 0.210 and 0.267 for BM25. Hybrid improves over BM25 on Hit@1 (0.385) and nDCG@10 (0.504) but remains below Dense under the same budget, suggesting that score fusion can still elevate lexically similar yet scope-mismatched clauses when attributes are underspecified.

5.3 Evidence-Bounded Selector Comparison

Evidence-bounded clause selection is evaluated with the same candidate budget ($k = 10$) to test whether a selector can identify the correct clause from a bounded evidence pool; dev200 is used because this stage is sensitive to prompt calibration and output-schema valida-

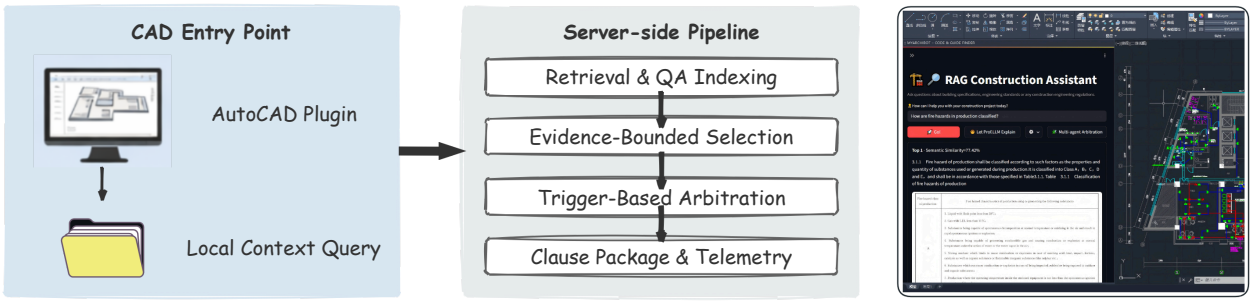

Figure 3. Within-CAD consultation flow reusing the server-side evidence pipeline

Table 4. Evidence-bounded selector results under a fixed top-10 candidate budget (dev200, $n = 200$). Cost is reported as USD per 1K queries.

Selector	Hit@1	MRR	Latency (s/query)	Cost
Qwen-plus	**0.430**	**0.523**	1.418	0.896
Qwen-max	0.395	0.501	2.498	2.944
GPT-4o	0.345	0.470	4.063	5.763
GPT-4o-mini	0.335	0.471	2.145	0.353

tion. Schema checks reject 1.0% of Qwen-plus outputs (2/200) due to out-of-candidate IDs, while other selectors have 0% invalid outputs; rejected cases are counted as misses. Table 4 compares API-based selectors under identical evidence inputs: Qwen-plus achieves Hit@1=0.430 (MRR=0.523) with 1.42 s/query and USD 0.896 per 1K queries, whereas GPT-4o achieves Hit@1=0.345 (MRR=0.470) with 4.06 s/query and USD 5.763 per 1K queries. Section 5.5 complements this fixed-point comparison with a post-hoc budget-feasibility audit of the archived Qwen-plus top-1 outputs under stricter prefix candidate pools.

5.4 Multi-Agent Arbitration Outcomes

Conditional arbitration is assessed on test100 to quantify when multi-agent expert disagreement indicates elevated ambiguity and how often arbitration revises majority selections under the fixed $k = 10$ evidence pool. Governance rates follow Formula 2 and Formula 5: arbitration triggers for 37% of queries and changes the majority top clause for 12% overall (32.4% among triggered cases). On test100, the main benefit concentrates on triggered queries: Hit@1 improves from 0.243 to 0.297 when arbitration is activated, while the overall gain is smaller (0.280 to 0.300) because non-triggered cases are already handled by the majority selector. Clause-ID disagreement accounts for all triggered cases (37/37 cases); applicability disagreement appears in 16 triggered cases as an additional signal, and no span-invalid triggers are observed in this benchmark (Table 5). Section 5.5 further audits how the archived final arbiter decisions remain feasible under stricter retrieval budgets.

Table 5. Arbitration statistics on test100 ($n = 100$). Trigger categories may overlap. Change is measured as whether the final arbiter-selected clause differs from the majority-voted top clause.

Split	Trig. rate	Change rate	Change|Trig.
Test100	37/100 (37.0%)	12/100 (12.0%)	12/37 (32.4%)

Split	Clause disagr.	Appl. disagr.	Span inv.
Test100	37	16	0

Notes: Trig.=arbiter invoked (computed from experts' top-1 clause outputs under the fixed $k = 10$ evidence pool); Change|Trig.=changes among triggered cases; Clause disagr.=experts select different clause IDs; Appl. disagr.=applicability conflict; Span inv.=missing/invalid evidence spans. Categories may overlap.

5.5 Retrieval-Budget Sensitivity and Post-hoc Feasibility Audit

To examine retrieval-depth sensitivity, Dense retrieval is analysed at $k \in \{1, 3, 5, 10\}$ using archived traces on dev200 and test100. Figure 4(a) shows that gold-in-C_k rises monotonically: on dev200, 0.550, 0.765, 0.850, and 0.935; on test100, 0.500, 0.820, 0.890, and 1.000 for $k = 1, 3, 5, 10$, respectively. The largest gain occurs between $k = 1$ and $k = 3$, with smaller gains thereafter, indicating diminishing marginal returns from deeper candidate pools.

Because the archived Qwen-plus selector and multi-agent arbitration runs in Tables 4 and 5 were executed at $k = 10$, downstream sensitivity is assessed through post-hoc feasibility audits under stricter prefix pools C_k. Figure 4(b) shows the results for the archived Qwen-plus run on dev200. Selected top-1 feasibility rises from 0.430 ($k = 1$) to 0.575 ($k = 3$), 0.660 ($k = 5$), and 0.740 ($k = 10$); all 86 correct top-1 decisions already correspond to retrieval rank 1. Figure 4(c) shows the results for the archived arbitration run on test100. The final arbiter-decision feasibility rises from 0.220 to 0.400, 0.620, and 0.880, while feasibility among the 30 correct final arbiter decisions rises from 43.3% to 60.0%, 83.3%, and 100.0%, respectively. Retrieval depth therefore increases reachability, but downstream gains are not linear in k: many correct selector and arbiter decisions are concentrated in highly

ranked candidates, whereas a smaller subset depends on deeper evidence pools.

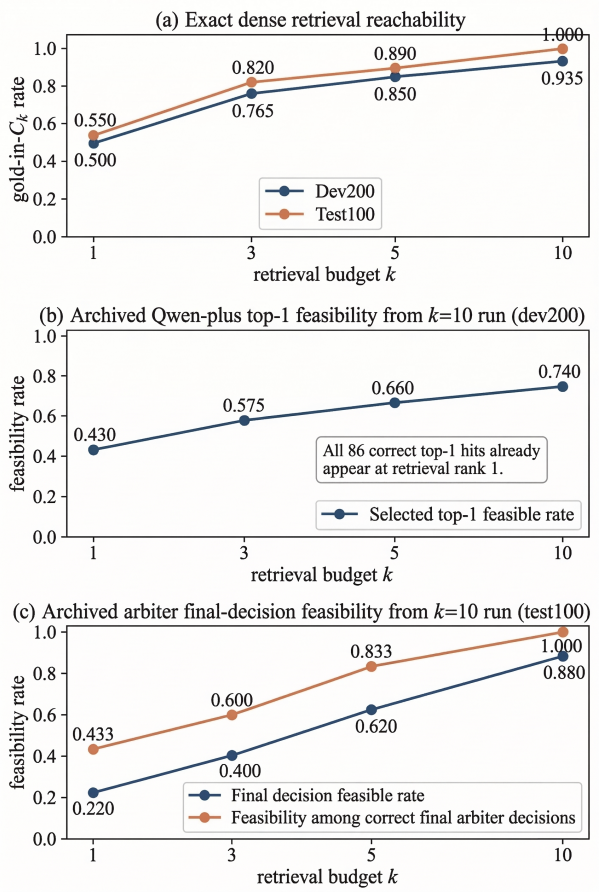

Figure 4. Retrieval-budget sensitivity and post-hoc feasibility audit over $k \in \{1, 3, 5, 10\}$. (a) Exact dense gold-in-C_k rate on dev200 and test100. (b) Post-hoc feasibility of archived Qwen-plus top-1 outputs from the $k = 10$ dev200 run under stricter prefix candidate pools. (c) Post-hoc feasibility of archived final arbiter decisions from the $k = 10$ test100 run under stricter prefix candidate pools.

5.6 Qualitative Case Studies

Two case studies illustrate auditable clause packaging and triggered arbitration under underspecified or cross-discipline conditions. Each reports the query, retrieved top-k clause IDs, the final clause package, and the arbitration trace when triggered; Table 6 provides the compact reporting template. Case B shows that arbitration corrects the majority choice under underspecified hazard-zone attributes by selecting an evidence-aligned clause within the retrieved pool, rather than a topically relevant but scope-mismatched alternative.

Table 6. Compact case study reporting for two representative queries on test100. Top-10 clause IDs are retrieved under a fixed candidate budget; the final clause package records the selected clause and an evidence span (translated excerpt).

Case A (no arbitration)	
Split / qid	Test100 / qid=12
Query	What is the minimum required clear width for a residential building corridor (passage)?
Retrieved top-10 clause IDs	25429, 3683, 125838, 76891, 60520, 60519, 12735, 11011, 76499, 37937
Final clause package (ID + evidence span)	25429: "The clear width of a corridor passage shall not be less than 1.20 m."
Applicability (final)	yes
Trigger flags	–
Arbiter change (vs. majority top clause)	No (not triggered)
Case B (arbiter changed)	
Split / qid	Test100 / qid=230
Query	For Zone 20 in a hazardous (explosive) atmosphere, how should electrical equipment be selected per code requirements?
Retrieved top-10 clause IDs	29097, 71780, 88188, 122388, 39587, 71781, 69799, 101799, 35034, 39588
Final clause package (ID + evidence span)	71781: "The explosion-proof class/group of equipment shall not be lower than that of the explosive gas mixture present in the environment."
Applicability (final)	yes
Trigger flags	clause disagreement; applicability disagreement
Arbiter change (vs. majority top clause)	Yes (71780→71781)

6 Discussion

Section 5 suggests that reliability in construction-code clause retrieval improves when evidence is bounded and disagreement is treated as a governance signal, rather than resolved through unconstrained generation. Dense retrieval raises the evidential ceiling by placing correct clauses earlier in C_k, while clause-level units keep verification tied to citeable clauses with stable identifiers. The retrieval-budget sensitivity analysis further qualifies the fixed-point results: reachability rises sharply from $k = 1$ to $k = 5$ and then slows, so $k = 10$ should be read as an operating point rather than a universal optimum. Post-hoc audits likewise show that downstream gains are not linear in k: many correct selector and arbiter decisions are already supported by highly ranked candidates, whereas a smaller subset depends on deeper evidence pools. The evidence-bounded selection contract keeps grounding mechanically checkable by requiring cited clause identifiers and evidence spans to remain traceable to retrieved text. For deployment, this supports a low-cost default selector with conditional arbitration for elevated-ambiguity cases. Residual failure sources remain missing scope attributes, long cross-references, and cross-edition drift. Reference-aware retrieval and lightweight attribute elicitation remain practical directions for improving robustness on tightly coupled multi-discipline queries.

7 Conclusion

This paper presented a clause-centric RAG pipeline for building construction code retrieval, where retrieval is treated as auditable evidence discovery and downstream models operate under an evidence-bounded clause-selection contract. By representing standards as clause-level evidence units with stable identifiers and provenance metadata, the system produces clause-addressable evidence bundles for rapid review and mechanical validation.

Experiments across multiple disciplines show that improving top-ranked evidence raises the ceiling for evidence-bounded selection, but that ceiling grows non-linearly with retrieval budget. Exact dense-retrieval sensitivity over $k \in \{1, 3, 5, 10\}$ shows strong reachability gains from $k = 1$ to $k = 5$, with smaller gains thereafter; post-hoc audits of archived $k = 10$ selector and arbitration outputs further show that many correct downstream decisions are already concentrated in highly ranked candidates, while a smaller subset depends on deeper evidence pools. Conditional multi-agent arbitration further serves as a governance layer by concentrating additional reasoning effort on high-risk cases identified through disagreement and evidence-quality triggers.

Together, these design choices support practical deployment under latency and cost constraints while keeping outputs traceable to explicit clause identifiers. Future work will focus on structured elicitation of missing project attributes and systematic handling of cross-standard cross-references for tightly coupled multi-discipline queries.

References

[1] Nicholas Nisbet, Zijing Zhang, and Selçuk Çidik. Real-time assessment of regulatory compliance of construction sites. In *EC3 Conference 2024*, volume 5, pages 0–0. European Council on Computing in Construction, 2024.

[2] Xiaorui Xue, Jiansong Zhang, and Yunfeng Chen. Question-answering framework for building codes using fine-tuned and distilled pre-trained transformer models. *Automation in Construction*, 168:105730, 2024.

[3] Jack Wei Lun Shi, Wawan Solihin, and Justin KW Yeoh. Fine-tuning a large language model for automated code compliance of building regulations. *Advanced Engineering Informatics*, 68:103676, 2025.

[4] Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. Evaluation of retrieval-augmented generation: A survey. In *CCF Conference on Big Data*, pages 102–120. Springer, 2024.

[5] Alsaffar Alhadi, Beach Dr Tom, and Rezgui Yacine. Enhancing asset management: Integrating digital twins for continuous permitting and compliance-a systematic literature review. *Journal of Building Engineering*, 99:111515, 2025.

[6] Thomas Beach, Jonathan Yeung, Nicholas Nisbet, and Yacine Rezgui. Digital approaches to construction compliance checking: Validating the suitability of an ecosystem approach to compliance checking. *Advanced Engineering Informatics*, 59:102288, 2024.

[7] Philipp Hagedorn, Judith Fauth, Sven Zentgraf, Sebastian Seiß, Markus König, and Ioannis Brilakis. Ontobpr: An ontology-based framework for performing building permit reviews using standardized information containers. *Advanced Engineering Informatics*, 66:103369, 2025.

[8] Stefan Fuchs, Michael Witbrock, Johannes Dimyadi, and Robert Amor. Using large language models for the interpretation of building regulations. *arXiv preprint arXiv:2407.21060*, 2024.

[9] Thomas Beach, Jonathan Yeung, Ali Ghoroghi, and Yacine Rezgui. Moving automated compliance checking to the operational phase of the building life-cycle: analysis and feasibility study in the uk. *International Journal of Construction Management*, 25(7):840–849, 2025.

[10] Odin Iversen and Lizhen Huang. Leveraging large language models for bim-based automated compliance checking. *Automation in Construction*, 182: 106707, 2026.

[11] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024. URL `https://aclanthology.org/2024.emnlp-main.992/`.

[12] Hansi Hettiarachchi, Amna Dridi, Mohamed Medhat Gaber, Pouyan Parsafard, Nicoleta Bocaneala, Katja Breitenfelder, Gonçal Costa, Maria Hedblom, Mihaela Juganaru-Mathieu, Thamer Mecharnia, et al. Code-accord: A corpus of building regulatory data for rule generation towards automatic compliance checking. *Scientific data*, 12(1):170, 2025.