Structure-Aware Retrieval for Parametric Construction Catalogs: Overcoming Parametric Collapse Using LLM-Based Extraction

Recuperación Estructurada en Catálogos Paramétricos de Construcción: Superando el Colapso Paramétrico Mediante Extracción Basada en LLM

Cesáreo González-Alvarez[1,*], Laura Fernández-Robles[1], Enrique Alegre[1] and Manuel Castejón-Limas[1]

[1]Group for Vision and Intelligent Systems, I4 Institute, Universidad de León, Campus de Vegazana s/n, 24071, León, Spain

Abstract

Parametric catalogues—where items share a concept but differ in discrete parameter values—are common in construction, manufacturing and e-commerce. In this work, we identify and formalise what we call *parametric collapse* as a structural limitation whereby dense models achieve high accuracy at the concept level (> 90%) but very low accuracy at the item level (< 45% in our case), since parametric variants generate almost identical representations. To address this limitation, we propose a solution based on a three-stage pipeline that separates concept identification, parameter extraction and deterministic resolution of the catalogue item. On the BC3CAT catalogue, the rule-based pipeline achieves a 90.3% Acc@1, outperforming neural methods and standard BM25, and approaching domain-tuned BM25. Ablation experiments with LLMs show that scaling primarily improves closed-set classification tasks.

Keywords

structure-aware retrieval, parametric catalogs, parametric collapse, large language models, slot-filling, construction NLP

Resumen

Los catálogos paramétricos —donde los ítems comparten un concepto pero difieren en valores discretos de parámetros— son habituales en construcción, fabricación y comercio electrónico. En este trabajo identificamos y formalizamos lo que denominamos *colapso paramétrico*, una limitación estructural por la cual los modelos densos alcanzan una precisión alta a nivel de concepto (>90%) pero muy baja a nivel de ítem (<45% en nuestro caso), ya que las variantes paramétricas generan representaciones casi idénticas. Para abordar esta limitación, proponemos una solución basada en una arquitectura de tres etapas que separa la identificación del concepto, la extracción de parámetros y la resolución determinista del ítem del catálogo. En el catálogo BC3CAT, la arquitectura basada en reglas alcanza un 90,3% de Acc@1, superando a los métodos neuronales y a BM25 estándar, y aproximándose a BM25 ajustado al dominio. Los experimentos de ablación con LLM muestran que el escalado mejora principalmente las tareas de clasificación de conjunto cerrado.

Palabras clave: recuperación estructurada, catálogos paramétricos, colapso paramétrico, modelos de lenguaje de gran tamaño, relleno de huecos (slot-filling), PLN en construcción.

1. Introduction

Large infrastructure projects rely on parametric construction catalogs that contain tens of thousands of work items, where each item shares a common concept but differs in discrete *parameter values* such as material, dimensions, or execution conditions. Accurately retrieving the correct item from natural-language descriptions is critical for cost estimation, procurement, and budget control [1, 2]. These catalogs follow a structured pattern: each item is defined by a *concept* (a base template describing a type of work) combined with a set of *parameter values*. A single concept may expand into hundreds or thousands of parametric variants, all sharing nearly identical textual descriptions but differing in small, but cost–critical, parameter values.

This paper addresses the retrieval problem on such catalogs: given a natural-language description of a work item, retrieve the exact matching entry from a catalog of nearly 48,000 candidates. The challenge is compounded by terminology variability between stakeholders—as field workers, designers, and cost estimators often use different terms for identical concepts [3, 4]—and by the critical role of numeric parameters, where even minor discrepancies lead to costly errors in project estimation [2]. To the best of our knowledge, this work is the first to explicitly identify and formalize parametric collapse as a structural limitation of dense retrieval in parametric catalogs.

Benchmark characteristics and implications. A key aspect of this work concerns the nature of the evaluation queries. The BC3CAT benchmark, derived from ADIF's parametric price catalog for Spanish railway infrastructure, uses catalog-generated short descriptions (*resumen*) as queries to retrieve catalog-generated long descriptions (*texto*). Both are generated from the same parametric templates, resulting in high lexical overlap between queries and targets. This design provides the closest possible approximation to a perfect retrieval scenario—an upper-bound evaluation setting where every query has an unambiguous correct answer and the query text is maximally informative. Nevertheless, neural models still fail to disambiguate between parametric variants, highlighting that the issue is structural, not just linguistic.

This benchmark characteristic has a direct consequence: lexical methods such as BM25 can exploit exact token matching against the high lexical overlap, achieving up to 97.4% item-level accuracy with parameter-aware tokenization (Section 3). However, in real-world deployment—cost estimators drafting specifications with company-specific word-

SEPLN 2026: 42nd International Conference of the Spanish Society for Natural Language Processing, León, Spain, 22–25 September 2026.
*Corresponding author.
cgonza06@estudiantes.unileon.es (C. González-Alvarez); l.fernandez@unileon.es (L. Fernández-Robles); enrique.alegre@unileon.es (E. Alegre); manuel.castejon@unileon.es (M. Castejón-Limas)
0009-0007-2481-4008 (C. González-Alvarez); 0000-0001-6573-8477 (L. Fernández-Robles); 0000-0003-2081-774X (E. Alegre); 0000-0002-5152-4555 (M. Castejón-Limas)

ing, field engineers writing abbreviated progress reports—this lexical overlap is substantially reduced. Even on this favorable benchmark, standard BM25 without parameter-aware tokens already drops to 86.9%, suggesting that performance would degrade further under realistic phrasing variation.

Neural methods, by contrast, are designed to handle this kind of phrasing variation. Yet they exhibit a systematic failure that we term *parametric collapse*: dense retrieval models achieve 96–99% accuracy at the concept (parent) level but only 12–45% at the item level, because embeddings map parametric variants—items that share more than 95% of their text—into near-identical vector representations. Thus, while concept-level retrieval is nearly solved, item-level disambiguation remains unreliable. This collapse is not a limitation of any specific model but reflects a structural mismatch between flat-text retrieval methods and the concept–parameter organization of parametric catalogs.

Our approach. We observe that the retrieval problem decomposes naturally into two sub-problems with very different difficulty profiles: *concept identification* (already solved by neural models at 96–99%) and *parameter resolution* (the actual bottleneck, currently unaddressed by standard methods). Based on this observation, we propose a three-stage structure-aware pipeline that separates concept retrieval from parameter resolution, matching the natural structure of the catalog.

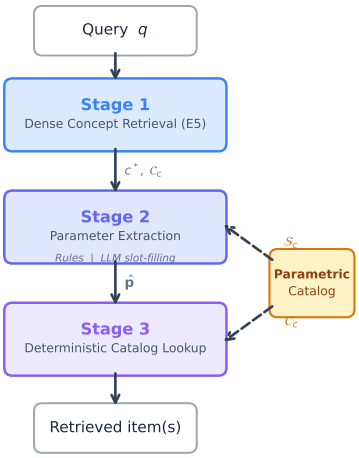

Figure 1: Overview of the three-stage structure-aware retrieval pipeline.

Contributions. Our contributions are fourfold:

1. We identify and empirically characterize *parametric collapse* failure mode, where dense retrieval achieves near-perfect concept-level accuracy but fails at item-level disambiguation.
2. We propose a three-stage structure-aware pipeline—dense concept retrieval, parameter extraction (rule-based or LLM), and deterministic catalog lookup—that explicitly targets the concept–parameter decomposition.
3. We ablate Stage 2 across two LLM models (Llama 3.1 8B, Phi-4 14B) and five prompt strategies, revealing that task framing (closed-set vs. open-ended) determines whether model scaling helps.
4. We provide the first evaluation of structure-aware retrieval for parametric construction catalogs on the BC3CAT benchmark (47,508 items, 16,590 queries).

2. Related Work

Construction catalog retrieval. The application of natural language processing (NLP) to construction document processing has evolved from rule-based and bag-of-words methods [5], through support vector machines [6], to transformer-based approaches [7, 8]. Systematic reviews document this progression [9, 3]. However, most work addresses category-level classification rather than item-level retrieval, and classification accuracy degrades as taxonomic granularity increases: [5] reported 95.9% at top-level divisions but substantially lower for subcategories.

A recent systematic comparison of lexical, neural, and hybrid retrieval methods on a large-scale Spanish construction catalog (under review) established that optimized BM25 achieves 97.4% item-level Acc@1, while the best neural method (BGE-M3 ColBERT) reaches only 44.8%. That work identified numeric mismatches as the dominant error source (approximately 98% of neural failures) but did not propose structure-aware solutions. The present study builds on those findings by targeting the parametric collapse phenomenon with a decomposed pipeline.

Structure-aware retrieval in product search. In the context of e-commerce, several works address the structure of product catalogs. [10] developed structured matching modules encoding multi-field product information into unified representations. [11] demonstrated that sparse TF-IDF with extreme multi-label classification (PECOS) outperformed dense embeddings on catalogs exceeding 100 million items. [12] proposed a two-stage architecture for procurement matching: dual transformer encoders followed by syntactic reranking. While these approaches leverage structured representations, they typically encode all information into a single representation and do not explicitly decompose retrieval into concept identification and parameter resolution. As a result, they do not address the failure mode we identify as parametric collapse.

LLMs for structured information extraction. Large language models have demonstrated strong zero-shot capabilities for structured extraction tasks, including named entity recognition, slot-filling, and form understanding. Recent open-source models in the 7–14B parameter range, such as Llama 3.1 [13] and Phi-4 [14], can perform schema-guided extraction when provided with explicit output constraints. We leverage these capabilities in Stage 2 by framing parameter extraction as constrained slot-filling over a small discrete label space. However, in our setting, extraction is only one component of the retrieval problem, and its effectiveness depends on explicitly modeling the parametric structure of the catalog.

While prior work has addressed structured retrieval and product search, the systematic collapse of parametric vari-

ants into indistinguishable representations has not been explicitly formalized.

3. The Parametric Collapse Problem

3.1. Problem Formulation

In parametric catalogs, each item d can be decomposed as:

$$d = (c, \mathbf{p}), \quad \mathbf{p} = (p_1, p_2, \ldots, p_K) \tag{1}$$

where c is the *concept* (shared base template) and $\mathbf{p}$ is a vector of p_i ($i \in \{0, \ldots, K\}$) parameter values drawn from discrete, finite domains $\mathscr{V}_k$. All items sharing the same concept c form a *concept group*; within a group, items differ only in their parameter combination $\mathbf{p}$.

Standard retrieval methods encode each item as a flat text representation enc(d), ignoring the $(c, \mathbf{p})$ decomposition and rank candidates according to a similarity function:

$$d^* = \arg\max_d \ sim(enc(q), enc(d)) \tag{2}$$

where q is the query. This formulation ignores the $(c, \mathbf{p})$ structure of the data.

When items within a concept group share more than 95% of their text—differing only in parameter value mentions—flat-text encodings produce near-identical representations that are insufficiently discriminative:

$$\|\text{enc}(d_i) - \text{enc}(d_j)\| \approx 0 \quad \text{for } d_i, d_j \in \text{group}(c) \tag{3}$$

This *parametric collapse* makes item-level discrimination unreliable while leaving concept-level retrieval intact.

3.2. Empirical Evidence

Table 1 presents results from a systematic benchmark of retrieval methods on the BC3CAT dataset (under review), evaluated at both parent level (concept group) and item level (exact parametric variant). The gap is expressed as percentage points (pp).

Method	Parent	Item	Gap (pp)
BM25 (param-aware)	0.985	0.974	−1.1
BM25 (standard)	0.973	0.869	−10.4
BGE-M3 ColBERT	0.997	0.448	−54.9
BGE-M3 Sparse	0.994	0.125	−86.9
BGE-M3 Dense	0.960	0.127	−83.3
Dense E5	0.982	0.134	−84.8

Table 1
Parametric collapse: parent vs. item Acc@1 on BC3CAT (16,590 queries, 47,508 items).

Two patterns emerge. First, all neural methods suffer from a dramatic drop from concept-level to item-level performance: they achieve 96–99.7% accuracy at concept level but only 12–45% at item level. The root cause is the parametric structure: the largest concept groups contain up to 6,336 variants across 5 parameter axes, and semantic models map these variants into overlapping regions of the embedding space.

Second, BM25 with parameter-aware tokenization achieves 97.4% item-level accuracy by exploiting lexical overlap between catalog-generated queries and targets. However, this behavior depends on the benchmark's favorable conditions: queries generated from the same templates as targets. Standard BM25 without parameter-aware tokens already drops 10.5 percentage points, and real-world phrasing variation—with abbreviations, synonyms, and implicit parameters—would further reduce lexical overlap.

Input: Natural-language query q

Stage 1 — Concept Retrieval (Dense E5)
Encode q with E5; retrieve top-1 item from dense index
Derive parent_key from item → concept group mapping
Output: concept group c, candidate set $\mathscr{C}_c$

Stage 2 — Parameter Extraction
Given q and schema $\mathscr{S}_c = \{(k, \mathscr{V}_k)\}_{k=1}^{K}$:
Extract $\hat{\mathbf{p}} = (\hat{p}_1, \ldots, \hat{p}_K)$ where $\hat{p}_k \in \mathscr{V}_k \cup \{\texttt{null}\}$
Methods: rule-based matching *or* LLM slot-filling

Stage 3 — Deterministic Catalog Lookup
Match $\hat{\mathbf{p}}$ against $\mathscr{C}_c$
Full match → single item; partial → ranked sub-group

Output: Ranked list of candidate items

Figure 2: Three-stage structure-aware retrieval pipeline.

3.3. The Decomposition Insight

The parent-level results reveal that concept retrieval is largely solved: neural methods achieve 96–99% accuracy at identifying the correct concept group. The remaining challenge is *parameter resolution*—disambiguating among parametric variants within the identified group. This reveals a fundamental mismatch: dense retrieval models optimize for global semantic similarity, whereas item-level retrieval in parametric catalogs requires precise matching of discrete parameter values. This observation motivates a decomposition of the retrieval task into two steps: concept identification and parameter resolution, each requiring different modeling strategies.

4. Proposed Method

We propose a structure-aware retrieval pipeline that explicitly decomposes retrieval into concept identification and parameter resolution through three stages (Figure 2).

4.1. Stage 1: Dense Concept Retrieval

Stage 1 identifies the concept group associated with the query. Rather than performing direct concept-level retrieval, we retrieve the most similar item and map it to its corresponding concept group using a precomputed mapping. This design leverages the strong performance of dense retrieval models at the concept level, while avoiding the need for explicit concept representations.

We use the multilingual E5 model (`intfloat/multilingual-e5-base`, 278M parameters, 768 dimensions) [15] to encode queries and items, and select the top-1 retrieved item to infer the concept c^*. E5 was selected as the lightest model among those benchmarked while maintaining strong parent-level performance. The existing dense index (47,514 vectors) is reused without modification.

4.2. Stage 2: Parameter Extraction

Given the query q and the concept schema $\mathcal{S}_c$ (axis labels with their discrete value sets), Stage 2 extracts a parameter vector $\hat{\mathbf{p}}$ through structured slot-filling, where each parameter corresponds to a discrete value. We evaluate two approaches.

Rule-based extraction. A deterministic extractor using normalized substring matching with longest-match-wins and subsumption filtering for text axes, and context-aware regex matching for numeric axes. This method exploits the fact that parameter values on the BC3CAT benchmark appear as literal substrings in query text, achieving high accuracy at negligible computational cost (approximately 0.015ms/query).

LLM-based extraction. A zero-shot slot-filling approach using local LLMs served via Ollama [16]. The query and concept schema are presented to the LLM, which returns a JSON dictionary mapping axis labels to extracted values. We evaluate two models—Llama 3.1 8B [13] (4.9 GB) and Phi-4 14B [14] (9.1 GB, Microsoft)—across five prompt strategies:

- **Strategy 1: Extract**: open-ended extraction ("extract the value for each axis"). Extracted values are normalized to match catalog vocabulary through unit standardization, numeric formatting, and synonym mapping.
- **Strategy 2: Classify**: closed-set selection with pipe-separated options ("classify the query into one of these values")
- **Strategy 3: Twostep**: free-form extraction followed by schema-guided matching (two LLM calls)
- **Strategy 4: Paraaware**: parameter aware per-axis detection followed by value matching, with full value lists shown (two LLM calls)
- **Strategy 5: Paraaware2**: per-axis detection with value lists shown in a single call per axis, returning bare values (N LLM calls, one per axis)

We distinguish between two task formulations: (i) open-ended extraction, where the model generates free-text parameter values that must later be normalized, and (ii) closed-set selection, where the model selects values directly from a predefined set of allowed options.

All prompts are in Spanish to match the catalog language, with temperature 0.0 for reproducibility.

4.3. Stage 3: Deterministic Catalog Lookup

Stage 3 matches the extracted parameter vector $\hat{\mathbf{p}}$ against the candidate set $\mathcal{C}_c$.

Because the parameter space is discrete and fully enumerated, this stage is exact: if all parameters are correctly extracted, the correct item is uniquely identified without ambiguity. If one or more axes are unresolved (`null`), the stage returns the sub-group of candidate items consistent with the resolved axes, ranked by a three-tier scoring scheme: matched items (score approximately 1.0), remaining concept group items (score approximately 0.5), and E5 fallback results (score below 0.5).

Query (*resumen*): Canalización hormigonada 1 T, polietileno libre de halógenos de 110 mm, bajo vías. (Diurno / i $\geq$ 5 horas / Volumen relevante)

Target (*texto*): Canalización hormigonada de 1 tubo de polietileno libre de halógenos de 110 mm de diámetro en cruce bajo vías, incluso el descerne y la entibación de los costados y la posterior reposición del balasto retirado, el relleno y compactado de la zanja, el suministro y montaje de los tubos y hormigón tipo HE-20 sin vibrar, la prueba de los conductos, el transporte y la retirada de los productos al lugar de empleo. Trabajo: Diurno. Banda de mantenimiento: i $\geq$ 5 horas. Condiciones de ejecución: Volumen relevante.

Gold parameters: Nº TUBOS = 1; TIPO DE TERRENO = Bajo vías; TRABAJO = Diurno; BANDA DE MANTENIMIENTO = i $\geq$ 5 horas; CONDIC. DE EJEC. = Volumen relevante

Figure 3: Example query–target pair from the OEB chapter (`item_key` `OEB030abaaa`, concept group `OEB030$` with 6 336 sibling items). The short query (*resumen*) and the long target (*texto*) differ mainly in surface form while sharing the same parameter values, illustrating the high lexical overlap of the benchmark.

5. Experimental Setup

5.1. Dataset

The BC3CAT dataset derives from ADIF's parametric price catalog for Spanish railway infrastructure. We focus on the OEB subcategory (platform system infrastructure), because it provides a large-scale, internally consistent parametric setting with substantial variation in group size and parameter complexity. It contains 47,508 distinct items organized into 25 concept groups. Group sizes range from 3 items (1 axis) to 6,336 items (5 axes with 3–8 values each). Each item has a short-format description (*resumen*, mean 19.7 words) used as the query and a long-format description (*texto*, mean 84.5 words) used as the retrieval target. Thus, each query is associated with exactly one gold catalog item, and retrieval is performed against the full inventory of 47,508 items.

From these, we sample 16,590 short descriptions as evaluation queries (99% confidence, ±1% margin), stratified by concept group to preserve the catalog's distributional structure.

As noted in Section 1, both descriptions are generated from the same parametric templates, producing high lexical overlap between query and target.

This represents an upper-bound evaluation setting: results on this benchmark are optimistic estimates of real-world performance, particularly for lexical methods.

5.2. Evaluation Protocol

We evaluate 16,590 queries (99% confidence, ±1% margin) against the full corpus of 47,508 items. The primary metric is item-level Acc@1; we also report parent-level Acc@1 to isolate concept identification performance, MRR when full ranked outputs are available.

Pipeline vs. oracle conditions. To isolate the contribution of each stage, we consider two conditions. In the *pipeline* setting, Stage 1 predicts the concept group using E5 dense retrieval. In the *oracle* setting, the ground-truth concept group is provided to Stages 2 and 3. The gap between both conditions quantifies the contribution of Stage

1 errors.

Evaluation conditions. We report six primary conditions:

Condition	Stage 1	Stage 2
Rules (pipeline)	E5 retrieval	Rule-based
Rules (oracle)	Ground truth	Rule-based
Phi-4 cl. (pipeline)	E5 retrieval	Phi-4 14B
Phi-4 cl. (oracle)	Ground truth	Phi-4 14B
Llama ext. (pipeline)	E5 retrieval	Llama 8B
Llama ext. (oracle)	Ground truth	Llama 8B

Table 2
Evaluation conditions used in the main experiments. Pipeline conditions use Stage 1 concept prediction; oracle conditions replace Stage 1 with the gold concept group to isolate Stage 2 performance.

These represent the best-performing configuration for each Stage 2 approach, selected from the full 2 models × 5 prompts exploration (Section 6.4).

5.3. Baselines

We compare against lexical, neural, and hybrid baselines from a systematic retrieval benchmark on the same dataset under the same retrieval setting (under review). These include BM25 with parameter-aware tokens (97.4%), BM25 standard unigram (86.9%), BGE-M3 ColBERT (44.8%), Dense E5 standalone (13.4%), and hybrid fusion methods.

5.4. Implementation

All experiments were developed using Python 3.10. Stage 1 reuses the existing E5 index built with sentence-transformers v2.2.2 [17] and uses top-1 retrieval without reranking. Stage 2 LLM inference uses Ollama 0.17.7 (Docker, GPU-enabled) with pinned model versions for reproducibility and is executed with deterministic decoding (temperature 0.0). Stage 3 lookup uses a precomputed reverse index from the parquet data. Evaluation runs on servers with Intel i9-13900KF CPUs and NVIDIA RTX 4090 GPUs. The full evaluation of 16,590 queries takes 10–11 minutes for the rules pipeline and 3–5 hours for LLM conditions. Reported runtime includes full end-to-end evaluation, including retrieval, parameter extraction, lookup, and result aggregation. Code is available at https://doi.org/10.5281/zenodo.20533039 and maintained at https://github.com/cga-telice/bc3cat-retrieval/releases/tag/sepln2026-submission.

6. Results

6.1. Main Results

We evaluate whether explicitly modeling parametric structure improves item-level retrieval compared to flat-text methods. Table 3 presents item-level performance across all pipeline conditions and baselines.

The rules-based pipeline achieves 90.3% item-level Acc@1, substantially outperforming all standalone neural methods (best: BGE-M3 ColBERT at 44.8%) and the hybrid fusion approach (89.0%). This large gap highlights the inability of dense retrieval models to resolve parametric variations, despite their strong concept-level performance. The pipeline also surpasses standard BM25 (86.9%) by 3.4 percentage points, demonstrating that explicitly separating concept identification from parameter resolution yields measurable gains even in a benchmark with high lexical overlap.

Method	Acc@1	Par.	MRR	Type
			Baselines (flat retrieval)	
BM25 param-aw.	0.974	0.985	0.979	Lex.
BM25 standard	0.869	0.973	0.898	Lex.
3-way fusion	0.890	—	0.908	Hyb.
BGE-M3 ColBERT	0.448	0.997	0.580	Neu.
Dense E5	0.134	0.982	0.251	Neu.
			Structure-aware pipeline (this work)	
Rules (pipe.)	0.903	0.985	—	Pipe.
Rules (orac.)	0.914	1.000	—	Pipe.
Phi-4 cl. (pipe.)	0.877	0.985	—	Pipe.
Phi-4 cl. (orac.)	0.886	1.000	—	Pipe.
Llama ext. (pipe.)	0.809	0.985	—	Pipe.
Llama ext. (orac.)	0.821	1.000	—	Pipe.

Table 3
Item-level performance on BC3CAT (16,590 queries). Baselines from prior work (under review); pipeline conditions from this work.

BM25 with parameter-aware tokenization remains the top performer at 97.4%, as it directly exploits exact lexical matches between queries and catalog entries. The 7.1 percentage point gap between BM25 param-aware and the rules pipeline stems primarily from Stage 2 extraction failures on a single parameter axis (Section 6.5).

6.2. Stage 1 Is Not the Bottleneck

To assess whether concept retrieval is a limiting factor, we compare pipeline and oracle conditions. The gap between both settings is consistently small across all conditions: 1.1% for rules, 0.9% for Phi-4 classify, and 1.2% for Llama extract.

$$\Delta_{Stage1} = Acc_{oracle} - Acc_{pipeline} \tag{4}$$

For the rules configuration, the gap is:

$$\Delta_{Stage1} = 91.4\% - 90.3\% = 1.1\% \tag{5}$$

These results confirm that concept identification is largely solved: E5 achieves 98.5% parent-level Acc@1, and residual errors have only a minor impact on overall performance. Instead, the dominant error source is Stage 2 parameter extraction. Across all configurations, Δ_{Stage1} remains below 1.2%, indicating that Stage 1 errors contribute marginally to overall failure.

All three pipeline conditions share the same 252 Stage 1 errors (1.5% of queries), which arise from confusions between closely related concept groups (e.g., 110mm vs. 160mm polyethylene tube canalizations, or manual vs. machine trench excavation).

6.3. Stage 2: Rules vs. LLMs

To isolate parameter extraction performance, we compare Stage 2 methods under oracle conditions, where the correct

concept group is provided. Table 4 summarizes the results.

Stage 2	Acc@1	Errors	Time
Rule-based	0.914	1,434	0.015ms
Phi-4 (classify)	0.886	1,887	~886ms
Llama (extract)	0.821	2,976	~585ms

Table 4
Stage 2 comparison under oracle conditions (16,590 queries).

The rule-based extractor outperforms both LLM-based approaches while being approximately 40,000–60,000× faster. This is expected on the current benchmark: since queries are generated from catalog templates, parameter values appear as literal substrings in the query text, making rule-based matching highly effective. In contrast, LLM extraction introduces errors through value normalization (e.g., returning `1.10` instead of the catalog value `1,10 m`) and omissions.

Critically, the rule-based extractor exhibits a predictable failure mode: it either matches correctly or returns `null`, but does not produce incorrect values. This makes it a reliable and interpretable baseline under controlled conditions.

However, this advantage is tied to the benchmark design. In more realistic scenarios with paraphrased or implicit parameter mentions, rule-based matching would be expected to degrade, whereas LLM-based extraction may better handle lexical variability and implicit information—a hypothesis we leave for future evaluation.

These results highlight that parameter extraction—not concept retrieval—is the core challenge in parametric catalogs, and that its optimal solution depends on the level of lexical variability in the input queries.

6.4. Prompt Strategy and Model Scaling

We analyze how prompt formulation and model size affect parameter extraction performance. Table 5 summarizes results across five prompt strategies and two LLMs (20-query Stage 2 comparison).

Prompt	Llama 8B	Phi-4 14B
Rule-based	100.0%	—
Extract	87.8%	77.0%
Classify	86.5%	100.0%
Paraaware2	82.4%	93.2%
Twostep	64.9%	58.1%
Paraaware	48.6%	78.4%

Table 5
Per-axis accuracy across prompt strategies and models (20-query sample, 74 axis extractions).

The most striking finding is the highly non-linear interaction between prompt strategy and model size. For *classify* (closed-set selection from pipe-separated options), Phi-4 achieves perfect accuracy while Llama reaches only 86.5%—a 13.5 percentage point improvement from model scaling, indicating that larger models better exploit constrained decision spaces. In contrast, for *extract* (open-ended value extraction), Phi-4 actually *degrades* compared to Llama (77.0% vs. 87.8%), as larger models tend to produce more selective outputs, increasing omission errors.

The *paraaware2* strategy—which makes N separate LLM calls (one per axis), each showing the full list of possible values—dramatically improves over the original *paraaware* for both models: Llama improves from 48.6% to 82.4% (+33.8pp) and Phi-4 from 78.4% to 93.2% (+14.8pp). This suggests that showing possible values upfront eliminates the abstract detection failures present in the original two-step approaches. However, paraaware2 still does not surpass the simpler single-call *classify* mode, which remains more accurate (especially for Phi-4) while requiring only one LLM call regardless of the number of axes.

This pattern generalizes: closed-set tasks (classify, paraaware2) benefit from model scaling because they require selecting among discrete alternatives, while open-ended extraction tasks are limited by normalization and alignment with catalog values rather than reasoning capacity.

The two-step approaches (twostep, paraaware) consistently underperform single-call methods, as Step 1 produces verbose compound descriptions introducing additional sources of error that propagate to Step 2.

6.5. Error Analysis

To better understand the sources of error in the proposed pipeline, we analyze all failed queries across the six primary conditions. Errors fall into two main categories:

E1 — Wrong concept (Stage 1 error). 252 queries (15.7% of rules pipeline errors) where E5 retrieves the wrong concept group. These are exclusively confusions between closely related concept groups (e.g., polyethylene 50mm vs. 160mm, or manual vs. machine excavation). Notably, these errors are themselves a manifestation of parametric collapse at the concept level: the concept descriptions differ by a single word or number (e.g., “110mm” vs. “160mm”, “a mano” vs. “a máquina”), causing dense embeddings to map distinct concepts into overlapping regions of the vector space. However, their limited frequency confirms that concept retrieval is not the primary bottleneck.

E2 — Parameter extraction failure (Stage 2 error). 1,434 queries (100% of rules oracle errors) where at least one axis is unresolved. For the rule-based extractor, all errors are partial extractions (`null` on one or more axes); zero wrong-value errors were observed.

Per-axis analysis. A per-axis analysis (Table 6) reveals that errors are concentrated on a single axis.

The TRABAJO (work shift) axis accounts for approximately 70% of all rules oracle errors (1,017 out of 1,434 null extractions). This axis encodes compound values combining multiple conditions (shift type, maintenance window, execution conditions) into a single textual string, making substring matching unreliable.

The remaining axes—including numeric axes like Nº TUBOS, TIPO DE TERRENO, and DIÁMETROS—achieve perfect extraction accuracy.

Axis	Occ.	Null	Acc.
TRABAJO	16,571	1,017	93.9%
CONDIC. DE EJEC.	16,580	34	99.8%
BANDA DE MANT.	16,571	28	99.8%

Table 6
Per-axis extraction accuracy (rules, oracle, 16,590 queries). Only axes below 100% shown; 10 other axes achieve perfect accuracy. Axis names translated: TRABAJO = work shift; CONDIC. DE EJEC. = execution conditions; BANDA DE MANT. = maintenance window.

Query fragment: "...para instalaciones CMS en transiciones entre diferentes plataformas. (**Diurno Excepcional**/*No aplica*/Voumen escaso)"

Catalogue TRABAJO values: "Diurno Excepcional", "*No aplica*", "Nocturno Excepcional"

Catalogue BANDA DE MANTENIMIENTO values: ..., "*No aplica*", "No necesita intervalo", ...

Rule extraction: `null` (two non-subsumed substring matches: **Diurno Excepcional** and *No aplica*)

Gold value: "Diurno Excepcional"

Figure 4: Representative Stage 2 failure on the TRABAJO axis (item `OEB250abaeb`, rules oracle). The catalogue reuses the literal string "*No aplica*" as a legal value for both TRABAJO and BANDA DE MANTENIMIENTO. When the query lists both "Diurno Excepcional" (for TRABAJO) and "No aplica" (for BANDA), longest-substring matching finds two non-subsumed candidates for TRABAJO and resolves the ambiguity by emitting `null`, even though both surface forms occur literally in the query. This cross-axis collision pattern accounts for the structural (non-typographic) component of the 1,017 TRABAJO nulls reported in Table C.

Hardest concept groups. Error rates vary substantially across concept groups: OEB190$ (manual trench excavation, 35.8% error rate) and OEB200$ (machine trench excavation, 33.6%) are the most challenging due to more complex parameter structures, while the large polyethylene tube groups (OEB030$, OEB040$) have error rates below 6.3%.

Rules vs. LLM disagreement. Under oracle conditions, 887 queries fail for rules but succeed for Llama, while 2,429 fail for Llama but succeed for rules, indicating that rule-based and LLM approaches capture different aspects of the extraction problem. For Phi-4 classify, the split is 804 rules-only failures vs. 1,257 Phi-4-only failures. This partial complementarity suggests that hybrid approaches combining deterministic matching with LLM-based interpretation could further improve performance.

7. Discussion

This work demonstrates that flat-text retrieval methods are fundamentally misaligned with the structure of parametric catalogs. By explicitly separating concept identification from parameter resolution, the proposed pipeline addresses the structural failure we term *parametric collapse*, enabling accurate item-level retrieval where dense models fail.

Why rules outperform LLMs on this benchmark. The rule-based extractor's dominance is a direct consequence of the benchmark's catalog-generated queries: parameter values appear as literal substrings, making pattern matching nearly optimal. In contrast, LLM-based extraction introduces errors due to value normalization (e.g., reformatting decimal separators, abbreviating units) and omissions—problems that arise precisely because the LLM attempts to *understand* the text rather than match values exactly.

We hypothesise that, on realistic queries with paraphrased parameters, this relationship would reverse: LLMs' ability to recognize "triple tubing" as "3 tubes" or "nighttime" as "Nocturne" would provide an advantage that rules cannot replicate. We do not test it here; evaluating it on realistic queries is left to future work.

The pipeline vs. BM25. The structure-aware pipeline (90.3%) does not surpass BM25 with parameter-aware tokenization (97.4%) on this benchmark. This is expected: BM25 param-aware tokens exploit the high lexical overlap that characterizes catalog-generated queries, achieving near-perfect performance on this specific evaluation setting. This result reflects the specific conditions of the benchmark rather than a limitation of the approach. However, the proposed pipeline follows a fundamentally different retrieval paradigm: it relies on semantic concept identification and explicit parameter resolution rather than lexical matching.

Its Stage 1 (dense E5) is robust to phrasing variation by design, and its Stage 2 explicitly reasons over the parameter schema rather than relying on lexical overlap. As lexical overlap diminishes—as it is likely to with real-world queries—we expect the pipeline to degrade more gracefully than BM25, although we do not meausre this on the present benchmark.

Practical implications. For deployment on catalog-generated or template-based queries, BM25 with domain-specific tokenization remains the clear choice: it is fast, accurate, and requires no GPU infrastructure.

However, in real-world operational conditions where lexical overlap decreases, the proposed pipeline provides a more robust alternative. For deployment with heterogeneous real-world queries—where phrasing varies across regions, companies, and individual users—the three-stage pipeline offers a principled architecture that combines semantic robustness with explicit parameter handling. The rules-based variant requires no GPU and runs in milliseconds; the LLM variant offers potential for handling paraphrased parameters at the cost of latency and infrastructure.

Limitations. This study has several limitations. First, evaluation is limited to a single catalog chapter (OEB) with a homogeneous parametric structure; extending the analysis to other chapters and catalogs is necessary to assess generality. Second, the use of catalog-generated queries introduces high lexical overlap, favoring lexical methods and limiting conclusions about real-world performance. Third, the evaluation of LLM-based extraction is limited to two models; larger or fine-tuned models may yield different results. Fourth, the pipeline's three-tier ranking for partial matches has not been optimized, leaving room for further improvements. Importantly, these limitations do not affect the central finding of the paper: that *parametric collapse* is a

structural issue that requires explicit modeling of parameter information.

8. Conclusions and Future Work

We identified *parametric collapse*—the systematic failure of dense retrieval models to disambiguate parametric variants in structured catalogs—and proposed a three-stage pipeline that decomposes retrieval into concept identification and parameter resolution. On the BC3CAT benchmark, the rule-based pipeline achieved 90.3% item-level Acc@1, outperforming all neural baselines and surpassing standard BM25 by 3.4 percentage points, while approaching the domain-tuned BM25 upper bound (97.4%). Error analysis reveals that 70% of residual failures stem from a single parameter axis, indicating that improvements can be achieved by refining extraction for this specific parameter type.

These findings suggest that accurate retrieval in structured domains requires explicitly modeling parameter information rather than relying on flat-text similarity. Future work will focus on evaluating the approach on real-world queries with lower lexical overlap, exploring hybrid methods that combine rule-based and LLM-based parameter extraction, lightweight classifiers (e.g., per-axis BERT heads) as efficient alternatives to LLM-based extraction, cross-domain evaluation on other parametric catalogs (e-commerce, industrial parts), and the integration of retrieval pipelines into BIM and estimation software for semi-automated construction workflows.

Acknowledgments

The authors acknowledge the use of generative AI tools to assist in drafting sections of the manuscript and producing code. All scientific interpretations, data analyses, and final revisions were performed by the authors.

References

[1] M. Martínez-Rojas, N. Marín, M. A. Vila, An Approach for the Automatic Classification of Work Descriptions in Construction Projects, Computer-Aided Civil and Infrastructure Engineering 30 (2015) 919–934. doi:10.1111/mice.12179.

[2] C. Wu, X. Li, Y. Guo, J. Wang, Z. Ren, M. Wang, Z. Yang, Natural language processing for smart construction: Current status and future directions, Automation in Construction (2022). doi:10.1016/J.AUTCON.2021.104059.

[3] S. Chung, S. Moon, J. Kim, J. Kim, S. Lim, S. Chi, Comparing natural language processing (NLP) applications in construction and computer science using preferred reporting items for systematic reviews (PRISMA), Automation in Construction 154 (2023) 105020. doi:10.1016/j.autcon.2023.105020.

[4] L. Rampini, F. R. Cecconi, Artificial intelligence in construction asset management: A review of present status, challenges and future opportunities, Journal of Information Technology in Construction 27 (2022) 884–913. doi:10.36680/j.itcon.2022.043.

[5] C. H. Caldas, L. Soibelman, Automating hierarchical document classification for construction management information systems, Automation in Construction 12 (2003) 395–406. doi:10.1016/S0926-5805(03)00004-9.

[6] M. Martínez-Rojas, J. M. Soto-Hidalgo, N. Marín, M. A. Vila, Using Classification Techniques for Assigning Work Descriptions to Task Groups on the Basis of Construction Vocabulary, Computer-Aided Civil and Infrastructure Engineering 33 (2018) 966–981. doi:10.1111/mice.12382.

[7] S. Tang, H. Liu, M. Almatared, O. Abudayyeh, Z. Lei, A. Fong, Towards Automated Construction Quantity Take-Off: An Integrated Approach to Information Extraction from Work Descriptions, Buildings 12 (2022) 354. doi:10.3390/buildings12030354.

[8] J. I. Deza, H. Ihshaish, L. Mahdjoubi, A Machine Learning Approach to Classifying Construction Cost Documents into the International Construction Measurement Standard, 2022. arXiv:2211.07705.

[9] C. Wu, X. Li, Y. Guo, J. Wang, Z. Ren, M. Wang, Z. Yang, Natural language processing for smart construction: Current status and future directions, Automation in Construction 134 (2022) 104059. doi:10.1016/j.autcon.2021.104059.

[10] J. I. Choi, S. Kallumadi, B. Mitra, E. Agichtein, F. Javed, Semantic Product Search for Matching Structured Product Catalogs in E-Commerce, 2020. doi:10.48550/arXiv.2008.08180. arXiv:2008.08180.

[11] W.-C. Chang, D. Jiang, H.-F. Yu, C. H. Teo, J. Zhang, K. Zhong, K. Kolluri, Q. Hu, N. Shandilya, V. Ievgrafov, J. Singh, I. S. Dhillon, Extreme Multi-label Learning for Semantic Matching in Product Search, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, ACM, Virtual Event Singapore, 2021, pp. 2643–2651. doi:10.1145/3447548.3467092.

[12] W. Cunha, C. França, L. Rocha, M. A. Gonçalves, TPDR: A Novel Two-Step Transformer-based Product and Class Description Match and Retrieval Method, 2023. doi:10.48550/arXiv.2310.03491. arXiv:2310.03491.

[13] A. Grattafiori, A. Dubey, A. Jauhri, et al., The Llama 3 herd of models, arXiv preprint arXiv:2407.21783 (2024). Meta AI.

[14] M. Abdin, J. Aneja, H. Awadalla, et al., Phi-4 technical report, arXiv preprint arXiv:2412.08905 (2024). Microsoft Research.

[15] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, F. Wei, Text Embeddings by Weakly-Supervised Contrastive Pre-training, 2024. doi:10.48550/arXiv.2212.03533. arXiv:2212.03533.

[16] Ollama, Ollama: Run large language models locally, https://ollama.com, 2024. Version 0.17.7.

[17] N. Reimers, I. Gurevych, Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, 2019. doi:10.48550/arXiv.1908.10084. arXiv:1908.10084.