Keeping Adaptation Simple: Implicit vs. Explicit Adaptation for LLM-Based CBR

Ravi Regulagedda[1], Nischal Bangalore Krupashankar[2], and David Leake[1]

[1]*Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington IN 47408, USA*

[2]*PamirAI Incorporated, San Francisco CA 94109, USA*

Abstract

The remarkable capabilities of large language models (LLMs) have prompted interest in LLM-based support for the case-based reasoning (CBR) process, including LLM implementations of the CBR cycle. Because case adaptation is an explicit step, it is natural to implement LLM-based CBR with an explicit case adaptation phase. However, CBR research has also shown the benefits of coupling retrieval and adaptation, suggesting potential benefits of a more unified approach. This paper presents a general model for implementing CBR with LLMs, and uses that model to compare the performance of explicitly guiding the model to perform adaptation, versus having adaptation arise implicitly within LLM processing. We present tests of four architectures that vary along two dimensions–retrieval mechanism (multi-step, where separate LLM calls handle retrieval sub-tasks, vs. unified, where a single LLM call performs all retrieval reasoning) and adaptation type (implicit, where the LLM adapts cases internally, vs. explicit multi-stage, where dedicated prompting steps guide adaptation) and evaluate them across six LLM models from three families, on GSM8K and MATH data sets. Performance of the case-based unified retrieval architecture, which presents the retrieved cases and the problem in a single context window, matches published few-shot chain-of-thought benchmarks without requiring curated few-shot examples. In contrast, explicit multi-stage adaptation consistently degrades accuracy, even when retrieval quality is high. A failure analysis identifies eight distinct ways in which multi-stage adaptation may destroy information. Our findings suggest a possible design principle for LLM-CBR systems: adaptation should be implicitly performed by the LLM rather than by explicit multi-step prompting.

1. Introduction

Large Language Models (LLMs) provide remarkable capabilities over a broad range of tasks but remain unreliable at structured, multi-step reasoning, with failure modes including hallucinated intermediate steps and inconsistent rule application [9, 14]. CBR, by bringing to bear specific prior examples, offers a natural complement to the generalization of LLMs, and has prompted much interest in how CBR can support LLMs, and how LLMs can support CBR [3]. One such area is LLM-based implementations of the entire CBR process [15]. A key question for LLM-based CBR is how to implement phases of the CBR process, whether through explicit guidance or implicitly, arising naturally during LLM processing as an effect of prompting the LLM to apply a case to a new problem. Within the CBR cycle, retrieval and case adaptation could be implemented as explicit multi-stage LLM processes, or be left implicit.

This paper presents a general LLM-based architecture for CBR that supports both explicit and implicit adaptation, and describes its application in four testbed systems for reasoning in a mathematical problem-solving domain. The testbed system architectures vary along two dimensions. The first is ***retrieval mechanism***: a ***multi-step retrieval pipeline*** where separate LLM calls handle similarity assessment, relevance grading, and case construction from retrieved documents; versus a ***unified retrieval*** approach in which a single LLM call performs all retrieval reasoning. The second is ***adaptation type***: ***explicit adaptation***, in which dedicated LLM stages implement adaptation by extracting a solution approach from the retrieved case, generating a reasoning chain that applies this approach to the new problem, and performing an error analysis that checks the adapted solution for consistency before producing a final answer; versus ***implicit adaptation***, where the retrieved case is passed directly to the solver and adaptation arises naturally from the LLM's internal processing over the combined context.

We evaluate the system variants on two mathematical reasoning datasets across three LLM families. In these tests, unified retrieval with minimal adaptation achieves high accuracy across all models above 10B parameters, regardless of model family, while explicit multi-stage adaptation consistently degrades performance. The performance effects of architecture design dominate those of model choice, with implicit retrieval resulting in large performance gains on grade-school level problems but not reliably on competition-level problems, revealing limits of surface-similarity retrieval on competition-level problems.

The paper makes the following contributions:

1. It presents a domain-agnostic LLM-based CBR architecture with two retrieval variants (multi-step and unified) and two adaptation variants (minimal and explicit), applicable to any domain with a case base of solved problems. Implementation code will be made publicly available if the paper is accepted.

2. It presents an evaluation across three model families, at multiple scales, and two benchmarks providing evidence that architecture design dominates model choice for CBR-augmented reasoning: mid-size and large models converge to $\sim$92% accuracy on GSM8K under unified retrieval, matching published 8-shot chain-of-thought results without curated examples, regardless of model family.

3. It develops a failure taxonomy for multi-stage case adaptation, identifying eight distinct failure types across 430 failure cases. The types suggest that the issue in the LLM-based CBR process is post-retrieval processing, not retrieval itself, and that multi-step adaptation can degrade quality as each adaptation stage compounds errors by treating its predecessor's output as authoritative without cross-referencing the original problem.

2. Related Work

2.1. Reasoning in Large Language Models

Much recent work on LLM reasoning traces back to chain-of-thought (CoT) prompting [14], which showed that asking a model to produce intermediate reasoning steps leads to large performance gains on arithmetic and commonsense tasks. Follow-up work has refined this idea in several directions: self-consistency decoding samples multiple reasoning paths and takes a majority vote [13], tree-of-thought prompting explores branching solution paths [18], and least-to-most prompting decomposes hard problems into simpler sub-problems [19]. These methods help, but problems remain. Models can produce chains that read well but contain logical errors, and they struggle with problems that require combining familiar reasoning steps in new ways [7]. Our approach connects to this by capturing reasoning as cases: rather than building reasoning chains from scratch, retrieving solved cases whose reasoning steps serve as implicit chain-of-thought examples, for the model to ground its reasoning in demonstrated solutions.

2.2. Retrieval-Augmented Generation

RAG [10], which retrieves related facts to augment queries, is a standard technique for grounding LLM outputs in external knowledge. RAG has demonstrated effectiveness: For example, hybrid methods combining dense retrieval with BM25 keyword matching have been reported to reduce erroneous responses in general-domain settings [2], though it is unclear whether this transfers to reasoning-heavy domains. Even strong embedding models struggle to retrieve structurally similar problems, confirming that standard embeddings match on surface features rather than reasoning structure [12]. Recent work on solution-guided retrieval [6, 11, 17] trains retrievers to match on solution structure; integrating such retrievers into CBR pipelines is a promising direction we discuss in Section 8.

2.3. Integrating Case-Based Reasoning with LLMs

Building case-based reasoning systems can require substantial domain expertise. LLMs are a natural fit for knowledge acquisition and automating parts of this process, since they can aid the knowledge engineering process [4] and draw on their training to assess similarity in flexible ways, as well as perform adaptation through prompting.

More generally, CBR-LLM integrations have recently gained much traction; a community paper [3] lays out a research agenda and argues that the relationship is symbiotic: LLMs can help CBR, for example by automating case acquisition, learning similarity measures, and proposing adaptation rules, while CBR can help LLMs, for example, by structuring retrieval, planning reasoning chains, and grounding outputs in concrete precedents.

Empirical research has tested whether LLMs can perform CBR sub-tasks such as case adaptation and similarity assessment [15]. Working with Llama and ChatGPT, they find that providing retrieved cases does improve accuracy, but also that getting consistent CBR-like behavior from a general-purpose LLM requires careful prompt design. Their results suggest that while LLMs have the raw capability for CBR, the interface between the two may be challenging.

Wiratunga et al. [16] take a different approach with CBR-RAG, which applies the CBR retrieval cycle to structure Retrieval-Augmented Generation for legal question answering. They use CBR's indexing vocabulary and similarity knowledge to select contextually relevant cases, showing that this leads to better answer quality than standard RAG. Their work is in the legal domain, but we expect that the principle applies more broadly: CBR can impose useful structure on what would otherwise be a generic retrieval step. Our work applies a similar idea to mathematical reasoning.

3. An Architecture for LLM-Based CBR

We have designed, implemented, and evaluated a general architecture for CBR-augmented reasoning using LLMs. The design is domain-agnostic: It can be applied to any domain with

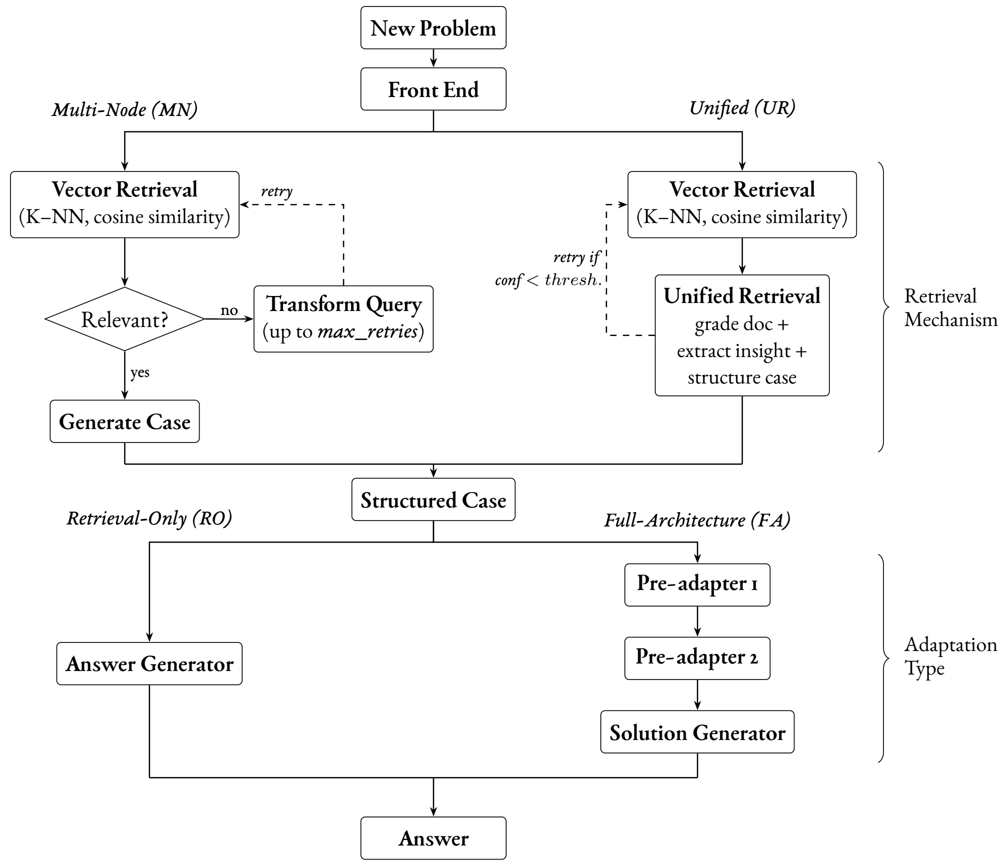

Figure 1: System architecture, varying along two independent dimensions: ***retrieval mechanism*** (Multi-Node vs. Unified) and ***adaptation type*** (Retrieval-Only vs. Full-Architecture).

a case base of solved problems simply by providing them to the system. Code of the implementation, all prompts used for the experiments, and documentation will be made accessible if the paper is accepted.

Based on this architecture, we developed four testbeds varying along two independent dimensions: the ***retrieval mechanism*** – how cases are found and validated, and the ***adaptation type*** – how retrieved cases are processed before answer generation. Crossing these two dimensions yields four retrieval-augmented configurations, plus a zero-shot baseline with no retrieval. Table 1 summarizes the five core configurations. No model fine-tuning is performed, and all reasoning is done through zero-shot prompting.

Central to the architecture is our definition of the *case*: a structured representation derived from retrieved solved problems, containing the problem statement, solution strategies, supporting facts, and step-by-step reasoning. Both retrieval variants construct this case from the case base; they differ in how.

3.1. Retrieval Variants

The retrieval mechanism has two variants:

Multi-Node (MN) Pipeline: The *multi-step retrieval* pipeline consists of the following stages:

1. **Front-end**: Extracts keywords and structures the query for retrieval.
2. **Retrieve**: Performs vector similarity search over the case base.

Table 1: System configurations. MN = Multi-Node, UR = Unified Retrieval, RO = Retrieval-Only, FA = Full-Architecture (explicit adaptation).

Config	Retrieval	Adaptation	LLM Calls	Description
LLM-BL	None	None	1	Direct prompting baseline
MN-RO	Multi-node	Minimal	6–9+	Multi-stage retrieval, case → solver
MN-FA	Multi-node	Explicit 3-stage	9–12+	Multi-stage retrieval + explicit adaptation
UR-RO	Unified	Minimal	2	Single retrieval call, case → solver
UR-FA	Unified	Explicit 3-stage	4	Single retrieval call + explicit adaptation

3. **Grade documents**: An LLM call judges each retrieved document for relevance, filtering out cases that match on surface features but not reasoning structure.

4. **Transform query** (loop, max 3 iterations): If grading rejects all documents, the query is reformulated and retrieval is retried.

5. **Generate case**: Structures the approved documents into a case representation containing relevance summaries, problem breakdowns, supporting facts, and solution approaches.

6. **Generate answer**: Produces the final answer using the structured case. This is done by the adaptation module.

This pipeline requires 6–9+ LLM calls per question, depending on the number of retrieval retries. When all retrieval attempts fail, a fallback mechanism routes directly to answer generation without case context.

Unified Retrieval (UR) Pipeline: The *unified retrieval* pipeline collapses the multi-node pipeline into a single LLM call. After vector retrieval returns a set of K candidate documents, one LLM call simultaneously performs document grading, relevance assessment, hallucination checking, fact extraction, and case structuring within a single context window. The output is a structured case representation identical to that produced by the multi-node pipeline, but generated in a single pass.

A confidence-based retry mechanism (max 2 retries) handles cases where the model is uncertain about the quality of the generated case or the retrieved documents. The LLM is prompted to rate how well the constructed case can guide a solution to the current problem, producing a confidence score in $[0, 1]$. If confidence is below a set threshold, retrieval and case generation are repeated.

Fundamental to our core experimental question, these two approaches represent a design trade-off: isolated calls could in principle allow each stage to specialize and validate intermediate outputs, while a single integrated context allows the LLM to act on all information at the hidden-representation level. Our experiments test which of these benefits dominates in practice.

3.2. Adaptation Variants

Each retrieval mechanism is paired with either of two adaptation strategies:

Retrieval-Only (RO): The structured case from retrieval is passed directly to a final answer generator. The LLM receives the original question plus the case and produces an answer. This represents *implicit adaptation* where the LLM is prompted to adapt retrieved patterns through its attention mechanism.

Full-Architecture (FA): After case generation, three *explicit adaptation* stages process the case before answer generation:

1. **Pre-adapter 1**: Analyzes the case to identify relevant solution patterns, analogies, and potential pitfalls.

2. **Pre-adapter 2**: Refines the identified patterns and generates specific transformation guidelines for the current problem.

3. **Solution generator**: Produces a step-by-step solution using the adapted case information.

This mirrors the classic CBR cycle [1] "Revise" step: explicit, multi-stage transformation of retrieved solutions before application. Each stage receives the output of its predecessor plus the original question.

4. Experimental Questions

Our experiments are designed to answer four questions:

1. **Implicit vs. explicit adaptation**: When retrieval is held constant, does implicit or explicit adaptation produce better results?

2. **How can adaptation fail?** What mechanisms cause either type of adaptation to break down?

3. **Architecture vs. model scale**: Does architecture design or model choice matter more for CBR-augmented reasoning?

4. **Is retrieval the bottleneck?** Do adaptation failures stem from poor retrieval quality, or from the adaptation process itself?

5. Experimental Setup

We set the number K of documents retrieved initially to 4 and the confidence threshold for the unified retriever (UR) variant to 0.3.
Models: We evaluate six instruction-tuned LLMs from three model families, spanning 7B to 32B parameters (Table 2). No fine-tuning is performed; all models are used with zero-shot prompting.
Datasets: We evaluate on two mathematical reasoning benchmarks. **GSM8K** [5] contains 7,473 training and 1,319 test problems requiring grade-school level mathematical reasoning (arithmetic, fractions, word problems). Answers are integers, enabling exact-match evaluation. **MATH** [8] contains 7,500 training and 5,000 test problems spanning seven subjects

Table 2: Models evaluated are instruction-tuned variants without fine-tuning.

Family	Model	Parameters
Qwen 2.5	Qwen2.5-7B-Instruct	7B
	Qwen2.5-14B-Instruct	14B
	Qwen2.5-32B-Instruct	32B
Gemma 3	gemma-3-12b-it	12B
	gemma-3-27b-it	27B
Llama 3.1	Llama-3.1-8B-Instruct	8B

(algebra, geometry, number theory, counting and probability, intermediate algebra, precalculus, prealgebra) at competition level. Answers are LaTeX expressions (fractions, radicals, symbolic expressions), requiring symbolic equivalence checking.

Our baseline (LLM-BL) uses zero-shot direct prompting, which differs from published benchmarks that typically use 8-shot chain-of-thought [14]. This makes our baselines deliberately weaker to isolate the effect of retrieval augmentation. Published GSM8K results for Qwen 14B and 32B are 94.8% and 95.9% respectively under 8-shot CoT; our zero-shot baselines for the same models are 38.1% and 37.2%. Under unified retrieval, these same models recover to $\sim$92% accuracy, nearly matching 8-shot CoT without needing manually curated few-shot examples. Our method uses the train split as the case base and automatically retrieves relevant cases for each test example. This suggests that much of the benefit of 8-shot CoT can be recovered by retrieving examples from a broad in-domain case base, without manually constructing a fixed few-shot prompt.

5.1. Evaluation Metrics

For GSM8K, we evaluated solution quality by exact numeric match after extracting the final numerical answer from model output. For MATH, we applied a three-tier comparison. The primary strategy parses both the model answer and ground truth as LaTeX expressions and checks whether their symbolic difference simplifies to zero. When symbolic parsing fails, it falls back to numeric float evaluation within tolerance. Its final fallback is to compare normalized string representations. All configurations are evaluated on the full test sets.

5.2. Retrieval Quality

Because performance of later steps depends on upstream performance as well, before attributing downstream performance differences to architecture design, we test whether retrieval quality is adequate. Using an LLM-based document grader (the same grading node used in our multi-node pipeline), we evaluate retrieval quality across both datasets. Dense retrieval with Nomic Embed v1.5 achieves a 78.5% document approval rate on GSM8K (97.6% coverage, which is the proportion of queries with at least one relevant retrieved case) and 71.4% on MATH (93.9% coverage). These rates confirm that the retrieval pipeline provides relevant cases for the vast majority of queries. We analyze retrieval quality and its implications further in Section 7.4.

Table 3: GSM8K accuracy (%) across models and configurations. Best results in **bold**. †Llama 8B results are affected by systematic formatting errors.

Model	LLM-BL	MN-RO	MN-FA	UR-RO	UR-FA
Qwen 7B	80.21	66.57	22.29	**80.06**	23.88
Qwen 14B	38.13	73.92	28.89	**92.04**	27.90
Qwen 32B	37.23	47.08	14.56	**91.89**	15.62
Gemma 12B	22.37	85.75	22.67	**91.28**	37.76
Gemma 27B	30.33	90.45	47.16	**92.95**	58.15
Llama 8B†	**42.76**	29.80	14.48	33.81	18.50

6. Results

6.1. GSM8K Results

Table 3 presents accuracy across all 30 model–configuration combinations on GSM8K. Several patterns emerge:

Unified retrieval dominates for 12B+ models. UR-RO achieves 91–93% for Qwen 14B/32B and Gemma 12B/27B, a remarkably tight range given that these models' baselines span 22–38%. In these tests, architecture design eliminates the 16-percentage-point gap between model families.

Multi-node retrieval helps, but inconsistently. MN-RO produces large gains for Gemma (85–90%) but mixed results for Qwen (47–74%). The multi-node pipeline's 6–9 sequential LLM calls create opportunities for error propagation that the unified approach avoids.

Explicit adaptation is consistently harmful. Both MN-FA and UR-FA degrade accuracy relative to their retrieval-only counterparts, with degradations ranging from 33 to 76 percentage points. The effect is particularly stark for UR-FA: starting from the same high-quality retrieval that produces 92% accuracy under UR-RO, adding three adaptation stages drops accuracy to 15–28% for Qwen and 38–58% for Gemma. This is the central finding of our work.

Small models hit a capacity ceiling. Qwen 7B achieves ∼80% regardless of configuration, suggesting that at 7B parameters, the model's intrinsic reasoning capacity is the binding constraint. Llama 8B generated systematic formatting failures that prevented fair evaluation of retrieval benefits.

6.2. MATH Results

To test whether these patterns generalize beyond grade-school mathematics, we evaluate on the MATH dataset (5,000 competition-level problems). Table 4 presents accuracy across all five configurations. The MATH results suggest the following observations:

Unified retrieval still outperforms multi-node. UR-RO matches or exceeds MN-RO for every model: Qwen 7B (+10.8pp), Qwen 14B (+4.9pp), Qwen 32B (+4.2pp), Gemma 12B (+2.0pp), and Gemma 27B (+1.8pp). The advantage of unified context persists on harder problems.

Retrieval helps Gemma but hurts Qwen. The baseline results reveal a model-family split. Qwen models achieve strong baselines (Qwen 7B: 57.82%, Qwen 32B: 37.55%) that retrieval-augmented configurations cannot match; Qwen 7B's best retrieval result (UR-RO, 29.16%) is half its baseline. In contrast, Gemma models see large gains from retrieval: Gemma 12B jumps from 21.54% to 77.64% under UR-RO (+56pp), and Gemma 27B from 36.58% to 82.92%

Table 4: MATH accuracy (%) across models and configurations. Best results in **bold**.

Model	LLM-BL	MN-RO	MN-FA	UR-RO	UR-FA
Qwen 7B	**57.82**	18.38	6.90	29.16	11.76
Qwen 14B	24.84	7.90	16.92	12.76	**24.86**
Qwen 32B	**37.55**	7.98	24.70	12.22	29.78
Gemma 12B	21.54	75.62	26.40	**77.64**	34.64
Gemma 27B	36.58	81.14	34.94	**82.92**	41.74
Llama 8B	5.52	10.10	6.96	**10.20**	8.70

(+46pp). This divergence was absent on GSM8K, where retrieval helped all models above 10B.

Explicit adaptation remains harmful. MN-FA and UR-FA consistently underperform their retrieval-only counterparts for Gemma models on MATH. Gemma 27B falls from 82.92% to 41.74% (−41pp) and Gemma 12B from 77.64% to 34.64% (−43pp) when adding explicit adaptation to UR-RO. For Qwen 14B and 32B, UR-FA outperforms UR-RO (+12pp and +18pp respectively), but neither retrieval configuration exceeds the zero-shot baseline for Qwen 32B, and Qwen 14B's best retrieval result (UR-FA, 24.86%) only matches its baseline (24.84%).

6.3. Cross-Dataset Comparison

Comparing GSM8K and MATH results reveals which findings are robust across both:

Generalized findings. Two conclusions hold across both datasets. Unified retrieval consistently outperforms multi-node retrieval, confirming that consolidated context is preferable to sequential processing regardless of problem difficulty. Explicit adaptation is harmful across both datasets for models where retrieval helps, with the same failure mechanisms identified in Section 7.2.

Dataset-dependent findings. The tight convergence under UR-RO seen on GSM8K (91–93% for all 12B+ models) breaks down on MATH. Gemma models benefit enormously from retrieval (Gemma 12B: 21.54% to 77.64%, Gemma 27B: 36.58% to 82.92%), while Qwen models are actively harmed by it (Qwen 7B: 57.82% to 29.16%, Qwen 32B: 37.55% to 12.22% under the same UR-RO architecture). On grade-school problems, surface-similar retrieved cases reliably share solution structure, so retrieval helps universally. On competition-level problems, surface similarity no longer guarantees solution relevance and models that cannot filter out misleading cases are worse off than with no retrieval at all. This points to a fundamental limitation of embedding-based retrieval for harder reasoning tasks, which we discuss further in Section 8.

7. Analysis

7.1. Question: Implicit vs. Explicit Adaptation

The contrast between UR-RO and UR-FA provides a clean test of implicit versus explicit adaptation. Both use identical retrieval (same unified retrieval call, same retrieved cases, same structured case). The only difference is what happens next: UR-RO passes the case directly to the solver, while UR-FA routes through three adaptation stages first.

On GSM8K, this difference is dramatic: UR-RO achieves 91–93% for 12B+ models, while UR-FA drops to 15–28% for Qwen and 38–58% for Gemma. The 430 failure cases all involve

Table 5: Failure taxonomy for explicit adaptation on GSM8K, classified by failure mechanism.

Code	Failure Mode	Count	%
MAE	Minor Arithmetic Error	101	23.5
MFT	Missing Final Transformation	99	23.0
TSC	Temporal Semantic Confusion	53	12.3
CCH	Complete Context Hallucination	44	10.2
MC	Magnitude Confusion	24	5.6
APE	Arithmetic Propagation Error	16	3.7
NDS	Numerical Detail Stripping	8	1.9
EAIE	Error Analysis Introduces Error	7	1.6
—	Unclassified	78	18.1

questions where the same retrieval input produced a correct answer under UR-RO but an incorrect one under UR-FA.

We interpret this as evidence that LLMs perform adaptation more effectively through their attention mechanism, operating over a unified context, than through explicit multi-stage token generation. When the model sees the question and retrieved cases together, it can selectively attend to relevant solution patterns, numerical details, and structural cues. When forced to generate an explicit adaptation plan across multiple stages, information is lost at each handoff.

This has implications for how we think about the "Revise" step in LLM-based CBR. Classical CBR assumes explicit adaptation is necessary to transfer solutions between cases. Our results suggest that for LLMs, the Revise step should be *implicit* and performed by the model's internal reasoning over unified context rather than externalized as a multi-stage pipeline.

7.2. Question: How Explicit Adaptation Hurts

To understand why explicit adaptation degrades performance, we conducted a systematic analysis of failure cases. We identified 430 questions on GSM8K for which UR-RO answered correctly but UR-FA answered incorrectly, with identical retrieval and only adaptation different. Using a heuristic rule-based classifier combining NLTK tokenization with numerical analysis, we classified 352 of these (81.9%) into eight failure modes (Table 5).

The top three failure modes—Minor Arithmetic Error, Missing Final Transformation, and Temporal Semantic Confusion—account for 253 of 430 failures (59%). These represent different ways in which information loss can occur:

Minor Arithmetic Error (MAE, 23.5%): The adaptation pipeline produces a solution that is on-topic and uses the right approach, but contains a small calculation error. For example, correctly identifying all prices and quantities but computing 7.5 instead of 5.5 in one intermediate step.

Missing Final Transformation (MFT, 23.0%): The pipeline solves most of the problem correctly but omits a final conversion or transformation step. A typical case: computing 80 legs correctly but failing to convert to "pairs" by dividing by 2. The unified solver, with the full problem visible, catches this; the adapter pipeline, having abstracted the problem across three stages, loses track of what the question actually asks.

Temporal Semantic Confusion (TSC, 12.3%): The fact extraction stage conflates time-related quantities (e.g., interpreting "$300/week" as "monthly $300"), and subsequent stages

Table 6: Retrieval quality evaluation. Approval rate is proportion of retrieved documents judged relevant by LLM grader. Coverage is proportion of queries with at least one relevant document.

	GSM8K		MATH	
Configuration	**Approval**	**Coverage**	**Approval**	**Coverage**
Dense (Nomic, 137M)	78.5%	97.6%	71.4%	93.9%
Dense (Qwen3, 8B)	85.0%	98.6%	86.9%	98.5%
Hybrid+Rerank (Nomic)	77.7%	97.0%	65.8%	92.3%
Hybrid+Rerank (Qwen3)	79.0%	97.0%	69.8%	94.3%

propagate this error without cross-checking the original question.

A notable model-specific pattern: Complete Context Hallucination (CCH) is heavily concentrated in Qwen 14B, which accounts for 39 of 44 cases (89%). In these failures, the adaptation stages solve an unrelated retrieved case problem instead of the target question.

The common thread across all failure modes is that each adaptation stage treats its predecessor's output as authoritative, without cross-referencing the original question, even when it has access to the full information at every step. The unified approach avoids this by keeping all information in a single context upon which the model's attention mechanism freely operates.

7.3. Question: Architecture vs. Model Scale

A striking finding across both datasets is that architecture design dominates model choice. On GSM8K, four models from two different families (Qwen 14B (92.04%), Qwen 32B (91.89%), Gemma 12B (91.28%), and Gemma 27B (92.95%)) converge to a 1.7-percentage-point range under UR-RO, despite baseline performances ranging from 22% to 38%. The 58-percentage-point gap between Gemma 12B's baseline (22.37%) and Qwen 7B's baseline (80.21%) is eliminated: under UR-RO, Gemma 12B (91.28%) substantially outperforms Qwen 7B (80.06%).

Models around 7–8B parameters appear to hit a capacity ceiling. Qwen 7B achieves $\sim$80% regardless of architecture, suggesting its intrinsic reasoning capacity cannot be overcome by retrieval alone. This defines a practical "retrieval breakpoint": below $\sim$10B parameters, the model's internal capabilities are the binding constraint; above it, architecture design determines performance.

On MATH, the convergence is less tight with Gemma models outperforming Qwen models even under UR-RO. This suggests that on harder problems, model-specific capabilities play a larger role. Still, the core pattern holds: a 12B Gemma model with good architecture (77.64%) dramatically outperforms a 32B Qwen model with the same architecture (12.22%).

7.4. Question: Retrieval Quality is Not the Bottleneck

A possible hypothesis is that adaptation failures stem from poor retrieval, due to retrieved cases not being sufficiently relevant to be useful. We test this using a separate retrieval quality evaluation comparing four retrieval configurations across both datasets (Table 6). Several findings emerge:

Retrieval quality is adequate. Our production configuration (Dense Nomic) achieves 78.5% approval on GSM8K and 71.4% on MATH, with 94–98% coverage. The vast majority of queries receive at least one relevant case.

Embedding model quality is the dominant factor for initial retrieval. Upgrading from Nomic (137M parameters) to Qwen3 Embedding (8B parameters) yields +6.5pp on GSM8K and +15.5pp on MATH which exceeds any pipeline-level improvement.

BM25 hurts for mathematical retrieval. Adding BM25 keyword matching (via Reciprocal Rank Fusion) ***degrades*** retrieval quality: −0.8pp on GSM8K and −5.6pp on MATH for Nomic, and −6.0pp/−17.1pp for Qwen3. The penalty increases with dataset complexity. This contradicts findings from general-domain RAG [2], where hybrid retrieval reduces failures. For mathematical reasoning, keyword matching surfaces topically similar but structurally different problems; for example, a "bakery" problem about unit conversion may retrieve other "bakery" problems about quantity counting, not structurally similar unit conversion problems from other domains.

The bottleneck is post-retrieval, not retrieval itself. Dense Nomic achieves 78.5% approval and powers UR-RO to 92% accuracy but the same retrieval under UR-FA produces only 15–28% accuracy. The 430 UR-FA failure cases analyzed in Section 7.2 all received the same high-quality retrieval input. Retrieval is not the bottleneck; what happens ***after*** retrieval determines success.

8. Conclusion

We have evaluated four LLM-based CBR architectures for mathematical reasoning across six models, three model families, and two benchmarks. Our central finding is that explicit multi-stage adaptation consistently and often dramatically harms performance compared to implicit adaptation. An analysis of 430 failure cases identifies eight mechanisms by which adaptation destroys information, with arithmetic errors, missed transformations, and semantic confusion accounting for the majority.

In contrast, unified retrieval with minimal adaptation achieves high accuracy on GSM8K for all models above 10B parameters, regardless of model family. This suggests a possible design principle: Rather than engineering explicit adaptation stages, present the problem and retrieved cases together and let the LLM's attention mechanism handle adaptation implicitly.

Our results also show that architecture design dominates model choice on GSM8K. On the harder MATH benchmark, however, retrieval benefits depend on the model family. This suggests that surface-similarity retrieval reaches its limits on competition-level problems. For practitioners building LLM-CBR hybrid systems, the implication is that investing in retrieval architecture yields greater returns than investing in larger models, but only if retrieval quality keeps pace with problem difficulty.

Our evaluation is limited to mathematical reasoning, so solidifying these results as a general principle will require broader testing. The principle that implicit adaptation outperforms explicit adaptation may not hold for domains where the adaptation step involves genuinely novel transformations not present in the retrieved cases.

In addition, the MATH results expose a key limitation of embedding-based retrieval: on competition-level problems, surface-similar cases often do not share solution structure, and retrieval hurts models that cannot filter out structurally irrelevant cases. This suggests the need for a solution-aware retrieval process, in which cases are matched by the reasoning strategies they require rather than by surface features of the problem statement. Our retrieval quality analysis already shows that upgrading embedding models substantially improves retrieval quality; training embeddings to capture solution structure rather than problem surface features might close the gap between model families on harder benchmarks.

9. Acknowledgments and Declaration on Generative AI

This research was supported in part by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute. The authors did not use any generative AI for the preparation of this paper.

References

[1] Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches . AI Commun. 7(1), 39–59 (1994)

[2] Anthropic: Contextual retrieval. Engineering at Anthropic (blog) (2024), https://www.anthropic.com/engineering/contextual-retrieval

[3] Bach, K., Bergmann, R., Brand, F., Caro-Martínez, M., Eisenstadt, V., Floyd, M.W., Jayawardena, L., Leake, D., Lenz, M., Malburg, L., Menager, D.H., Minor, M., Schack, B., Watson, I., Wilkerson, K., Wiratunga, N.: Case-based reasoning meets large language models: A research manifesto for open challenges and research directions (2025), hAL Archives, hal-05006761

[4] Brand, F., Malburg, L., Bergmann, R.: Large Language Models as Knowledge Engineers. In: Proceedings of the Workshops at the 32nd International Conference on Case-Based Reasoning (ICCBR-WS 2024) co-located with the 32nd International Conference on Case-Based Reasoning (ICCBR 2024), Mérida, Mexico, July 1, 2024. CEUR Workshop Proceedings, vol. 3708, pp. 3–18. CEUR-WS.org (2024)

[5] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., Schulman, J.: Training verifiers to solve math word problems (2021), https://arxiv.org/abs/2110.14168

[6] Debnath, D., et al.: RaDeR: Reasoning-aware dense retrieval for mathematical problem solving. In: Proceedings of EMNLP (2025)

[7] Dziri, N., Lu, X., Sclar, M., Li, X.L., Jiang, L., Lin, B.Y., et al.: Faith and fate: Limits of transformers on compositionality. In: NeurIPS (2024)

[8] Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., Steinhardt, J.: Measuring mathematical problem solving with the math dataset. NeurIPS (2021)

[9] Huang, J., Chang, K.C.C.: Towards reasoning in large language models: A survey. In: Findings of ACL (2023)

[10] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: NeurIPS (2020)

[11] Oh, J., et al.: ReasonIR: Training retrievers for reasoning tasks. arXiv preprint arXiv:2504.20595 (2025)

[12] Su, H., et al.: BRIGHT: A realistic and challenging benchmark for reasoning-intensive retrieval. In: ICLR (Spotlight) (2025)

[13] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., et al.: Self-consistency improves chain of thought reasoning in language models. In: ICLR (2023)

[14] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., et al.: Chain-of-thought prompting elicits reasoning in large language models. In: NeurIPS (2022)

[15] Wilkerson, K., Leake, D.: On Implementing Case-Based Reasoning with Large Language Models. In: Case-Based Reasoning Research and Development - 32nd International Conference, ICCBR 2024, Merida, Mexico, July 1-4, 2024, Proceedings. Lecture Notes in Computer Science, vol. 14775, pp. 404–417. Springer (2024)

[16] Wiratunga, N., Abeyratne, R., Jayawardena, L., Martin, K., Massie, S., Nkisi-Orji, I., Weerasinghe, R., Liret, A., Fleisch, B.: CBR-RAG: Case-based reasoning for retrieval augmented generation in LLMs for legal question answering. In: Proceedings of ICCBR. Springer (2024)

[17] Xiao, S., et al.: ReasonEmbed: Reasoning-aware text embeddings. arXiv preprint arXiv:2510.08252 (2025)

[18] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T.L., Cao, Y., Narasimhan, K.: Tree of thoughts: Deliberate problem solving with large language models. In: NeurIPS (2023)

[19] Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., et al.: Least-to-most prompting enables complex reasoning in large language models. In: ICLR (2023)

A. Failure Case Examples

The following are representative examples from the 430 GSM8K failure cases where UR-RO answered correctly and UR-FA answered incorrectly. Each illustrates one of the eight identified failure modes.

MAE --- Minor Arithmetic Error *(Q108, Gemma 12B)*

Question: Frankie watches TV Mon–Fri. Mon and Tue: 1 hour each. Wed: unknown number of 30-minute episodes. Thu: 1 hour and 30 minutes. Fri: two 1-hour episodes. Total: 7 hours. How many 30-minute episodes on Wednesday?

Ground truth: 3. Correct sum: $1 + 1 + 1.5 + 2 = 5.5$ hours, so Wednesday $= 1.5$ hours $= 3$ episodes.

What went wrong: The adapter's Stage 2 computes each day correctly but sums five terms instead of four (double-counting one day), getting $7.5 + 0.5x = 7 \Rightarrow x = -1$. It acknowledges "number of episodes cannot be negative" but blames the problem statement instead of re-checking its own arithmetic. Outputs -1.

Unified solver: Correctly sums $5.5 + 0.5x = 7 \Rightarrow x = 3$.

MFT --- Missing Final Transformation *(Q1268, Gemma 27B)*

Question: A farm has 10 two-legged animals and 15 four-legged animals. How many **pairs** of legs?

Ground truth: 40. Both pipelines correctly compute $10 \times 2 + 15 \times 4 = 80$ total legs.

What went wrong: Neither adapter re-reads the question to notice it asks for ***pairs***, not total legs. Stage 2 notes "no apparent risks." Outputs 80.

Unified solver: After computing 80 legs, adds "The question asks for pairs" $\rightarrow 80/2 = 40$.

Root cause: The adapters followed the retrieved plan faithfully (which said "sum the legs") without verifying it answered the actual question.

TSC --- Temporal Semantic Confusion *(Q305, Qwen 14B)*

Question: Watson earns $10/hr, works 10 hrs/day × 5 days/week, and gets a $300 bonus **each week**. How much does he earn in April (company performed well all month)?

Ground truth: $3,200. Correct: $500/week wages $\times 4 = \$2{,}000$; $300/week bonus $\times 4 = \$1{,}200$; total $3,200.

What went wrong: Fact extraction labels the bonus as "Monthly bonus: $300"—conflating the weekly $300 with monthly. Stage 1 accepts the wrong label without cross-referencing. Stage 2's error analysis worries about "whether April has exactly 4 weeks" (irrelevant) instead of catching the weekly→monthly confusion. Outputs $2,300.

Unified solver: Correctly identifies $300 as weekly, computes $\$300 \times 4 = \$1{,}200$ bonus $\rightarrow \$3{,}200$.

Root cause: Errors introduced during fact extraction become "locked in" across adapter stages.

CCH --- Complete Context Hallucination *(Q452, Qwen 14B)*

Question: 20 balloons cost $900. Price increases $20 per balloon. How much for 170 balloons?

Ground truth: $11,050. Correct: $\$900 \div 20 = \$45$/balloon $\rightarrow +\$20 = \$65 \rightarrow 170 \times \$65 = \$11{,}050$.

What went wrong: Stage 1 completely abandons the target problem and instead solves an unrelated retrieved case: "100 balloons × 3 ounces = 300 ounces, $300 \div 50 = 6$ bottles,

$6 \times \$2.50 = \$15$, $\$20 - \$15 = \$5$." Stage 2 propagates identically, adding error analysis about a concern from the wrong problem. Outputs 5.

Unified solver: Follows the facts step-by-step $\rightarrow$ $\$45 + \$20 = \$65 \rightarrow 170 \times \$65 = \$11{,}050$.

Root cause: The adapter latched onto a coherent narrative in the retrieved case rather than the target question. No mechanism for the solver to detect it received the wrong problem.

EAIE --- Error Analysis Introduces Error *(Q187, Gemma 27B)*

Question: There are 3 cages of rats with 6 rats each, 10 hamsters, and some rabbits. Each rat gets 5 straws, each hamster gets 5 straws, each rabbit gets 20 straws. Total 160 straws. How many rats per cage?

Ground truth: 5. Correct: $18x + 50 + 20 = 160 \Rightarrow 18x = 90 \Rightarrow x = 5$.

What went wrong: Stage 1 **correctly** solves: $18x + 70 = 160 \Rightarrow x = 5$. Stage 2 then "corrects" it: "Rabbits are not rodents [taxonomically true]. The 160 is distributed among small rodents only." Changes the equation to $18x + 50 = 160 \Rightarrow x = 5.56$. Outputs 5.56.

Unified solver: Correctly includes all animals $\rightarrow 160 - 20 - 50 = 90 \rightarrow 90 \div 6 = 15 \rightarrow 15 \div 3 = 5$.

Root cause: The error-prevention mechanism (error analysis) is what causes the error. Stage 1's correct answer is overwritten by Stage 2's spurious "correction."

B. System Prompts

This appendix contains the full text of all prompts used in the system. Prompts are grouped by architecture variant. Shared prompts (Section B.1) are used by both v1.3 and v1.4 architectures. Section B.2 lists the grading and query rewriting prompts specific to the v1.3 multi-node pipeline. Section B.3 lists the three-stage adaptation prompts used in the full architecture variant. Section B.4 contains the single unified retrieval prompt that replaces the entire v1.3 retrieval pipeline. Section B.5 gives the direct prompting baseline.

B.1. Shared Prompts (Used by v1.3 and v1.4)

```
Prompt S1: Query Formatter (Front-End)

You are a question-processing assistant. Your job is to extract
the most relevant keywords from a user's question, rephrase it into
a structured query format that emphasizes its key elements, and
categorize the question based on its intent. Follow these steps:
1. Rephrase the question into a query-friendly structure that
emphasizes clarity and key information.
2. Extract the primary keywords (including proper nouns, concepts,
and important terms).
3. Categorize the question based on its intent (factual or opinion-
based).
```

```
Prompt S2: Document Grader

You are a grader assessing if a retrieved document is relevant to a
user's question AND provides adaptable problem-solving techniques.
Grade "yes" if:
  1. The document shares keywords, semantics, or context with the
question.
  2. It includes methods, formulas, code, or reasoning steps that
could be adapted to solve the question, even if applied in a
different domain.
  3. It offers partial matches with transferable insights or matches
the domain enough to trigger understanding.
Grade "no" only if:
  1. The document is unrelated in content AND lacks problem-solving
value.
  2. It states facts without actionable logic (e.g., raw data,
disconnected theorems).
A "no" can become a "yes" if the problem being solved in the
document can be abstracted to approximate the problem in the user's
question.
Return "yes" or "no".
```

```
Prompt S3: Case Generator

You are a problem solving assistant. Your task is to analyze a
user's question and the content of retrieved documents to create a
structured case for solving the problem.
```

```
Based on the user's question and the provided documents, generate
the following fields:
1. relevance: A concise summary explaining how the techniques or
formulas from the retrieved documents can be adapted to solve the
user's question.
2. breakdown: A step-by-step decomposition of the user's question
into smaller, logical parts.
3. facts: A list of key formulas, definitions, or principles
extracted from the documents that are essential for solving the
problem.
```

Prompt S4: Final Answer Generator

```
You are a reasoner and problem solver. You are given a question and
structured case information to help you solve it.
The structured case contains:
  - Relevance: How retrieved examples relate to the current problem
  - Breakdown: Step-by-step decomposition of the question
  - Facts: Key formulas, definitions, and principles from similar
problems
Your task:
1. Use the case information as guidance to understand the problem
structure
2. Apply the relevant facts and formulas to solve the problem step-
by-step
3. Show your work clearly
4. End your response with 'Final Answer: [numerical value]'
Remember: The case information provides context from similar
problems. Adapt the techniques to solve THIS specific problem.
```

B.2. v1.3 Multi-Node Pipeline Prompts

Prompt M1: Query Rewriter

```
You are an expert at query optimization for vector databases. Your
task is to rewrite a user's question to improve its chances of
retrieving relevant documents.
Analyze the user's question and the provided explanation for why the
initial retrieval failed.
  - Focus on Core Concepts: Identify the fundamental concepts,
theorems, or problem types.
  - Remove Ambiguity: Rephrase conversational or ambiguous parts
into precise, searchable terms.
  - Add Key Terms: If possible, infer and add related keywords that
are likely to be in a relevant document.
  - Use the Explanation: The explanation tells you what went wrong.
Use it to guide your rewrite away from the failed concepts.
Return only the improved question, with no preamble.
```

```
Prompt M2: Hallucination Grader

You are a grader assessing whether the core methods or techniques
described in an LLM generation are grounded in a set of retrieved
facts.
Your task is to give a binary score: 'yes' or 'no'.
  - 'Yes' means that the fundamental methods, processes, or
techniques mentioned in the generation are supported by the provided
facts.
  - 'No' means they are not.
Crucially, you may ignore any specific numbers, statistics,
or quantitative values in favor of the larger problem solving
technique. Discrepancies in numbers alone should not result in a
'no' score if the underlying technique is correct.
```

```
Prompt M3: Answer Grader

You are a grader assessing whether a generated case (Relevance,
Breakdown, Facts) provides a logically coherent foundation to answer
the user's question. Return:
"yes" if:
  1. Thematic Alignment: The case's "Relevance" and "Facts" address
concepts, terms, or methods directly tied to the question.
  2. Logical Breakdown: Sub-questions decompose the problem into
parts that, if answered, would logically lead to solving the
question.
  3. Actionable Facts: "Facts" provide formulas, relationships,
or steps that could plausibly address the sub-questions (even if
abstract).
"no" only if:
  1. Irrelevant Case: The case's content (Relevance/Facts) is
unrelated to the question.
  2. Incoherent Breakdown: Sub-questions are illogical, off-topic,
or unrelated to the case's stated Facts.
  3. Unsupported Facts: Facts include claims that contradict the
case's own logic or lack internal consistency.
Return a binary score: "yes" or "no".
```

B.3. v1.3 Full Architecture: Adaptation Prompts

These three prompts form the adaptation pipeline, applied sequentially after case generation.

```
Prompt A1: First Adaptation Stage --- Solution Approaches &
Analogies

You are an adapter that generates solution strategies using
retrieved documents. For the given [Question] and [Retrieved
Documents], produce:
1. Solution Approaches:
  Outline step-by-step methods to solve the question, directly
derived from techniques in the retrieved documents.
```

```
  Format: "Step 1: [Actionable step inspired by Document X's
method]. Step 2: [Next step, tied to Document Y's logic]."
2. Analogies and Examples:
  Identify parallels between the current question and cases in the
retrieved documents.
  Format as chain-of-thought reasoning: "In Document Z, [problem]
was solved by [method]. Similarly, here, [analogy to current
question] because [shared logic/constraint]."
Rules:
  - Every step/analogy MUST map to explicit content in the retrieved
documents.
  - For coding/math: Use exact formulas, algorithms, or code
patterns from documents.
  - Reject speculative or unsupported connections.
```

Prompt A2: Second Adaptation Stage --- Reasoning Chains & Error Analysis

```
Generate a reasoning chain and optional error analysis using the
Solution Approaches and Analogies/Examples.
1. Step-by-Step Reasoning Chain:
  Synthesize the Solution Approaches into a logical sequence,
directly referencing methods/facts from retrieved documents.
  Format: "Step 1: [Action] (Based on Document X: [Brief
rationale]). Step 2: [Next action] (Using Document Y: [Rationale])."
2. Error Analysis (Optional):
  Identify potential mismatches between the current problem and
retrieved cases.
  Format: "Risk: [Potential error]. Mitigation: [Adjustment inspired
by Document Z's caveat: [Explanation]]."
Rules:
  - Every step/risk MUST reference specific documents.
  - For math/coding: Use exact equations/code patterns from
documents.
  - If no risks are evident, omit error analysis.
```

Prompt A3: Final Solver (Adaptation)

```
You are a solver tasked with answering the user's question by
synthesizing the generated case (Relevance, Breakdown, Facts),
solution approaches, analogies, and reasoning chain.
1. Answer Structure:
  - Restate the question.
  - Summarize critical steps from the reasoning chain, explicitly
referencing:
    -- Solution Approaches (e.g., "Step 1: Apply [method from
Document X]").
    -- Analogies (e.g., "Similar to [Document Y's case], we...").
  - State the final answer.
```

```
2. Rules:
  - The answer MUST derive directly from facts, methods, and
analogies in the retrieved documents.
  - Cite specific documents for key steps.
  - If error analysis identified risks, address them.
  - Avoid unsupported claims, speculative logic, or external
knowledge.
3. Output Format:
  - Concise, logical narrative (1--3 paragraphs).
  - End with "Final Answer: [answer]".
```

B.4. v1.4 Unified Retrieval Prompt

This single prompt replaces the entire v1.3 multi-node retrieval pipeline (Prompts S2, S3, M1, M2, M3). The LLM performs document grading, filtering, case generation, hallucination checking, and confidence assessment in one call.

```
Prompt U1: Unified Implicit Retrieval
You are an expert case-based reasoning assistant. Given a question
and retrieved documents, perform ALL the following steps internally
and return a single structured response.
Your Internal Process (do not output these steps separately):
1. Analyze Question: Extract key concepts, mathematical principles,
and problem type
2. Evaluate Documents: For each document, assess if it shares
concepts/methods that could help solve the question
3. Filter: Keep only documents that are relevant and provide
adaptable problem-solving techniques
4. Generate Case: From relevant documents, create:
  - relevance: How the techniques from documents can be adapted to
solve this problem
  - breakdown: Step-by-step decomposition of the question into sub-
questions
  - facts: Key formulas, definitions, and principles extracted from
the documents
5. Self-Validate: Ensure your case is grounded in the documents (no
hallucination) and provides a foundation to answer the question
6. Assess Confidence: Rate 0.0--1.0 how well this case can guide a
solution
If no relevant documents are found, set confidence to 0.0 and
provide a minimal case based on the question alone.
Output Format (JSON only, no other text):
{
  "relevance": "How the retrieved documents' techniques can be
adapted to solve this problem",
  "breakdown": "Step-by-step decomposition of the question",
  "facts": "Key formulas, definitions, and principles from the
documents",
  "confidence": 0.8
}
```

B.5. LLM Baseline Prompt (No Retrieval)

Used for the direct prompting baseline. No retrieval, no case-based reasoning — a single LLM call with the raw question.

```
Prompt B1: Direct LLM Baseline

Solve this problem and provide only the numerical answer. Your
answer should be only 'Final Answer: [numerical value]'.
Question: {question}
```