RAG over Thinking Traces Can Improve Reasoning Tasks

Anonymous ACL submission

Abstract

Retrieval-augmented generation (RAG) has proven effective for knowledge-intensive tasks, but is widely believed to offer limited benefit for reasoning-intensive problems such as math and code generation. We challenge this assumption by showing that the limitation lies not in RAG itself, but in the choice of corpus. Instead of retrieving documents, we propose retrieving thinking traces, i.e., intermediate thinking trajectories generated during problem solving attempts. We show that thinking traces are already a strong retrieval source, and further introduce $\mathcal{T}^3$, an offline method that transforms them into structured, retrieval-friendly representations, to improve usability. Using these traces as a corpus, a simple retrieve-then-generate pipeline consistently improves reasoning performance across strong models and benchmarks such as AIME 2025–2026, LiveCodeBench, and GPQA-Diamond, outperforming both non-RAG baselines and retrieval over standard web corpora. For instance, on AIME 2025-2026, RAG with traces generated by Gemini-2-thinking achieves relative gains of +56.3%, +8.6%, and +7.6% for Gemini-2.5-Flash, GPT-OSS-120B, and GPT-5, respectively, even though these are more recent models. Overall, our results suggest that thinking traces are an effective retrieval corpus for reasoning tasks, and transforming them into structured, compact, or diagnostic representations unlocks even stronger gains.

1 Introduction

Retrieval-augmented generation (RAG) has become a standard way to improve large language models (LLMs) on knowledge-intensive tasks by retrieving external documents that provide factual grounding (Lewis et al., 2020; Fan et al., 2024). However, its value for reasoning-intensive tasks remains far less clear. Prior work suggests that standard retrieval over general-purpose corpora often provides limited or inconsistent gains for tasks such as mathematical reasoning, that these gains tend to appear mainly for weaker models (Lyu et al., 2025), and that it can even hurt performance when the retrieved context is noisy or poorly aligned with the reasoning process (Li et al., 2025; Shi et al., 2023; Geng et al., 2024; BehnamGhader et al., 2023). This has contributed to a growing belief that retrieval may be less helpful for reasoning than it is for factual question answering (Gao et al., 2023).

In this work, we challenge the assumption that RAG is ineffective for reasoning (Liu et al., 2024; Lyu et al., 2025), arguing that the limitation lies not in retrieval itself but in the choice of the retrieval corpus. While prior RAG predominantly uses knowledge sources or generic web or textbook as retrieval corpora, they have shown to be better suited for factual recall than for reasoning tasks such as math. Instead, we posit that reasoning benefits from access to process-level signals such as how solutions are derived. Motivated by this, we propose using *thinking traces*—intermediate reasoning trajectories generated during problem-solving by state-of-the-art reasoning models—as a retrieval corpus for reasoning-focused RAG. We find that simply replacing standard web corpora with raw thinking traces already yields surprising gains for reasoning tasks.

At the same time, naïvely retrieving raw traces is suboptimal: full thinking traces from the state-of-the-art reasoning models are often lengthy, noisy and redundant, making them difficult for downstream models to use effectively. We therefore propose $\mathcal{T}^3$ (**T**ransformation of **T**hinking **T**races), an offline method that transforms thinking traces into more structured, retrieval- and context-friendly forms. Rather than providing raw reasoning trajectories, $\mathcal{T}^3$ distills them into concise scaffolds that provide a "how-to" for the reasoning process rather than mere factual grounding. More broadly, we treat thinking traces as a reusable resource and rather than distilling them into model parameters

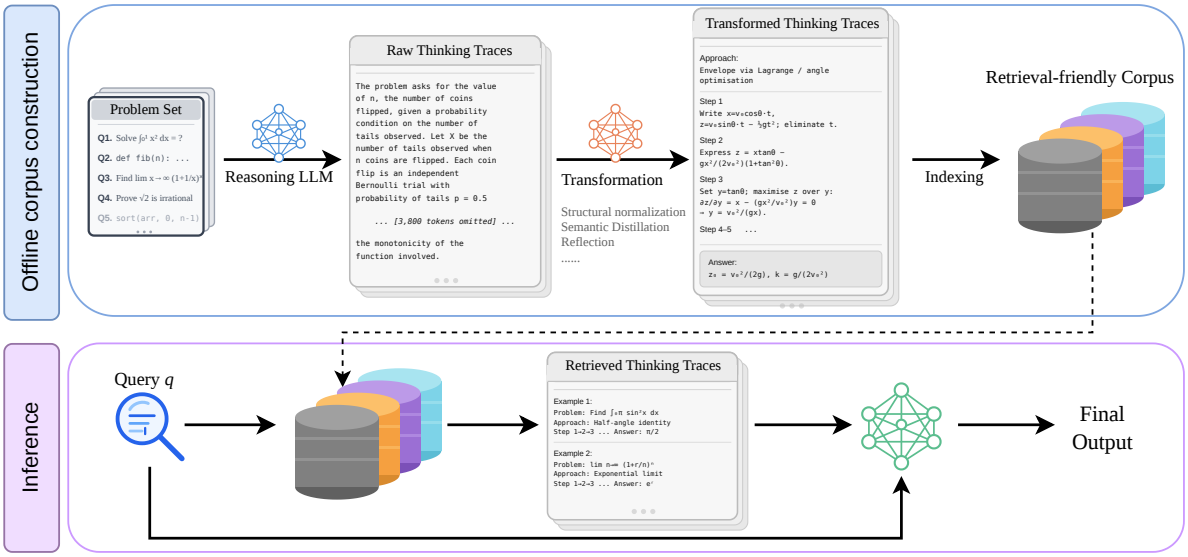


Figure 1: Overview of $\mathcal{T}^3$. Offline, a large reasoning model (e.g., Gemini-2-thinking) solves a set of problems and produces raw thinking traces. A smaller model (e.g., Gemini-2-Flash-Lite) then rewrites them into structured representations, forming a retrieval-friendly corpus. At inference time, a previously unseen query, which is not part of initial problem set, is retrieved against this corpus, and the retrieved context is provided to a downstream LLM to generate the final answer. The inference model may differ from the trace-generation and transformation models.

or discarding them after inference, we transform and retrieve them for future problems. In this sense, our setup is closer to learning from others' prior reasoning attempts and mistakes than to revising a model's own reasoning online. Because the trace corpus is built from a fully separate auxiliary problem set and generated by different models than those used at inference time, it also remains cleanly separated from the evaluation queries and reduces the risk of contamination.

Figure 1 illustrates the overview of the process. In an offline stage, a strong *thinking model* (e.g., Gemini-2-thinking) generates thinking traces over a curated problem set, which a smaller *transformation model* (e.g., Gemini-2-Flash-Lite) rewrites into retrieval-friendly forms using $\mathcal{T}^3$. These transformed traces form the retrieval corpus. At inference time, a standard retrieve-then-generate pipeline retrieves relevant trace segments and conditions a *solver model* on them, replacing conventional web or knowledge corpora with transformed thinking traces. The thinking, transformation, and solver models may be identical or distinct. Notably, our experiments show that even weaker thinking or transformation models can significantly improve stronger solver models.

We run experiments across multiple frontier models, including GPT-OSS-120B (OpenAI Team, 2025a), GPT-5 (OpenAI Team, 2025b), and Gemini-2.5-Flash (Gemini Team, 2023), and across reasoning-intensive benchmarks spanning math(AIME 2025–2026), coding (LiveCodeBench) (Jain et al., 2024), and scientific question answering (GPQA-Diamond) (Rein et al., 2023). Our contributions in this work are as follows:

1. **We show that raw thinking traces are a uniquely effective retrieval source for reasoning-intensive tasks.** On AIME 2025–2026, raw-trace retrieval on Gemini-2-thinking improves Gemini-2.5-Flash from 53.3 to 80.0 (+50.1%), GPT-OSS-120B from 78.3 to 85.0 (+8.6%), and GPT-5 from 86.7 to 91.7 (+5.8%).

2. **We propose $\mathcal{T}^3$, an offline method for transforming thinking traces into more retrieval-friendly representations.** $\mathcal{T}^3$ converts raw traces generated by strong reasoning models, including QwQ-32B and Gemini-2-thinking, into structured and more usable forms using a relatively light LLM (e.g., Gemini-2-Flash-Lite). In several cases where raw traces yield limited gains, transformed traces unlock clear improvements across tasks. For example, on GPQA Diamond, $\mathcal{T}^3$ improves GPT-OSS-120B from 70.7 to 74.7 (+5.7%), and on LiveCodeBench from 57.9 to 61.4 (+6.0%).

3. **We show that RAG on $\mathcal{T}^3$ can improve the cost–accuracy trade-off.** By shifting computation from expensive test-time decoding to cheaper input context, retrieval over thinking traces not only can improve answer quality, but in the best setting, it also makes inference cost up to 15% cheaper per query (e.g., GPT-5).

We intend to release the code and data used in this paper to support future research.

Method	Retrieved object	Reusable artifact	Inference-time mechanism	Vanilla RAG
RAT (Wang et al., 2024)	Documents	×	Iterative CoT revision	×
TRACE (Fang et al., 2024)	Documents/chains	×	Evidence-chain construction	×
BoT (Yang et al., 2024)	Thought templates	Meta-buffer ✓	Template instantiation	×
RoT (Ahmed et al., 2025)	Thought graph nodes	Thought graph ✓	Reward-guided traversal	×
$\mathcal{T}^3$	Transformed thinking traces	Trace-derived corpus ✓	Simple retrieval	✓

Table 1: Comparison with closely related reasoning-RAG systems.

2 Related Work

Reasoning in LLMs. LLMs have recently shown strong performance on reasoning-intensive tasks (Wang et al., 2025; Rozière et al., 2024; Auer et al., 2023). Prior work improves reasoning through prompting and chain-of-thought reasoning (Wei et al., 2022; Wang et al., 2022), distillation from stronger reasoning models (Ho et al., 2023; Magister et al., 2023; Shridhar et al., 2023; Muennighoff et al., 2025), and reinforcement learning with verifiable rewards (Guo et al., 2025; Yu et al., 2025). Recent work also studies how reasoning can interact with retrieval- or memory-like mechanisms (Wang et al., 2026; Du et al., 2025). Our work is complementary: instead of internalizing reasoning through training or distillation, we study whether prior thinking traces can be stored externally and retrieved at inference time to guide reasoning.

RAG for Reasoning. RAG is widely used for knowledge-intensive tasks by retrieving documents that provide factual grounding (Lewis et al., 2020; Gao et al., 2023; Siriwardhana et al., 2023). Most work focuses on retrieving, selecting, and incorporating textual evidence from large corpora (Fan et al., 2024; Singal et al., 2024; Huo et al., 2023), with recent work also studying retrieval as an inference-time scaling mechanism where larger datastores can improve retrieval-based language models (Shao et al., 2024). While effective for factual question answering, its benefits for reasoning-intensive tasks remain less clear and unreliable.This motivates methods that couple retrieval with intermediate reasoning traces.

Several recent systems reuse "thoughts" or reasoning artifacts, but these usually refer to generated chains, templates, or graph nodes rather than the native long-form thinking traces produced by reasoning models.

Retrieval-Augmented Thoughts (RAT) (Wang et al., 2024) revises an initial chain of thought step by step using retrieved information, showing retrieval can help when tightly coupled with generation; TRACE (Fang et al., 2024) constructs knowledge-grounded reasoning chains from retrieved documents using additional LLM calls before answer generation; Buffer of Thoughts (BoT) (Yang et al., 2024) retrieves high-level thought templates from a meta-buffer and instantiates them for the current task. Retrieval-of-Thought (RoT) (Ahmed et al., 2025) organizes reusable reasoning steps into a thought graph and uses reward-guided traversal to assemble problem-specific templates at inference time. Other work improves reasoning retrieval by training specialized retrievers (Shao et al., 2025) or scaling general-purpose datastores (Lyu et al., 2025). Table 1 summarizes these differences.

Our work studies a different axis. Rather than proposing a new online reasoning controller, we ask whether reasoning gains can be obtained by changing the retrieval corpus while keeping the inference pipeline fixed. This preserves the simplicity, cost profile, and deployability of standard retrieve-then-generate RAG, requiring no additional inference-time model, controller, graph traversal, template instantiation, or iterative revision loop. In contrast to the closest prior work summarized in Table 1, our new component is offline corpus construction. To our knowledge, we are the first to study how existing thinking traces can be transformed into a reusable retrieval corpus for reasoning. At inference time, the resulting corpus can be integrated into existing RAG pipelines with no additional reasoning-time machinery beyond standard retrieval. We provide an expanded discussion of prior work in Appendix A.

3 Methodology

We study how reasoning trajectories can serve as effective retrieval units for reasoning-intensive tasks. The key idea is to view trajectory retrieval as a representation problem, where the same trace can be transformed into different retrieval-friendly forms.

3.1 Thinking Trajectory-Based Corpus Design

Let $q \in \mathcal{Q}$ denote a test query and let L be the target model used for inference. We assume access to an auxiliary collection of problems and their associated reasoning trajectories, from which we construct a set of reasoning trajectories $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$, where each τ_i is a raw reasoning trace generated for an auxiliary problem by a strong model. These traces form the starting point of our corpus construction pipeline.

Problem (AIME 2026): Call finite sets of integers S and T cousins if they have the same number of elements, are disjoint, and their elements can be paired so that each pair differs by exactly 1. Suppose that S has exactly 4040 cousins. Find the least number of elements S can have.

Without retrieval 0/8

The model starts correctly – it identifies that S cannot contain three consecutive integers and partitions S into isolated and paired blocks – but never completes the argument. All 8 samples terminate mid-sentence before reaching a final answer.

Retrieved context from $\mathcal{T}^3$- Reflect

Example 1. Set theory/Real Analysis – Existence proof failure due to oversimplification of set properties.

× Assuming a simple 'building block" approach for unique representation. × Ignoring structural constraints that prevent closure.

✓ Verify closure properties; test uniqueness implications on concrete examples.

Correct approach: Recognise that uniqueness constraints lead to unavoidable contradictions.

Example 2. . . .

Example 3. Set Theory and Number Theory – Incorrect assumptions about consecutive pairs in infinite sets.

× Transferring properties of special numbers to arbitrary set elements. × Flawed casework that does not exploit the defining relationship.

✓ Ensure arguments rely only on defining relationships, not assumed structure.

Correct approach: Assume both sets have finitely many consecutive pairs and derive a contradiction.

Response with RAG 7/8

Based on the provided examples, the model recognizes that the number of cousins factors multiplicatively over independent arithmetic-progression blocks: if S is a union of progressions of sizes $k_1, \ldots, k_s$, then $N_C(S) = (k_1+1)(k_2+1)\cdots(k_s+1)$. Setting this equal to $4040 = 2^3 \cdot 5 \cdot 101$ and minimising $\sum k_i$ yields factors $(k_j+1) \in \{2, 2, 2, 5, 101\}$, giving $n = 1+1+1+4+100 = \boxed{107}$.

Figure 2: **A case study of $\mathcal{T}^3$- Reflect.** Without retrieval, Gemini-2.5-Flash fails to reach a correct answer in 8 attempts. Retrieval over full traces is also insufficient and does not lead to a correct solution. In contrast, retrieval over $\mathcal{T}^3$ provides targeted reasoning guidance that enables the model to solve 7 out of 8 attempts correctly. Retrieved examples and solutions are shortened for brevity. Our comments on model behavior are in dark blue.

From this set of trajectories, we derive a trajectory-based retrieval corpus $\mathcal{C}_\tau$, where each retrieval unit corresponds either to a full or chunked raw trajectory $\tau_i \in \mathcal{T}$. Given $\mathcal{C}_\tau$, a retriever R returns the top-k units $D(q; \mathcal{C}_\tau, k) = \{\tau_1, \ldots, \tau_k\}$. The retrieved units are then concatenated ($\oplus$) with the query and provided to the model: $y \sim L(D(q; \mathcal{C}_\tau, k) \oplus q)$. We study whether thinking traces are an effective retrieval corpus for reasoning-intensive RAG, and whether transforming them into more structured forms can make them even more useful.

3.2 T3: Transformation of Thinking Traces

We model reasoning transformation as a family of offline functions applied to a corpus of raw reasoning trajectories. Each transformation maps a trajectory τ into one or more retrieval-oriented representations, $f : \tau \mapsto \{\tilde{\tau}_1, \ldots, \tilde{\tau}_m\}$. Applying such a transformation to the full trajectory set $\mathcal{T}$ yields a transformed trajectory corpus $\tilde{\mathcal{C}}_\tau = \bigcup_{\tau \in \mathcal{T}} f(\tau)$.

In general, each raw trajectory may produce one or more transformed representations, and these transformed units are often shorter than the original trajectory, i.e., $|\tilde{\tau}_i| \ll |\tau|$, reflecting different degrees of compression and abstraction.

This formulation has two advantages. First, all transformations are *query-independent* and can therefore be applied fully offline, incurring only a one-time cost while enabling reuse of the transformed corpus across future queries. Second, because all retrieval variants are derived from the same base trajectory set, we can isolate the effect of representation design while keeping the retrieval and generation pipeline fixed.

We present three query-independent strategies for reconstructing raw reasoning traces, each one capturing a distinct perspective on what to preserve from the original trajectory. Prompts for each transformation are available in Appendix B.

Structural Normalization Struct This strategy rewrites raw traces into concise step-by-step procedures. It removes detours, noise, and inconsistent formatting of raw thinking traces while preserving the main reasoning structure, producing cleaner scaffolds that are easier to retrieve and reuse as inference-time guidance.

Semantic Distillation Semantic. This strategy keeps the core idea of traces while removing lower-level details. Rather than preserving every intermediate step, it emphasizes the key decisions and central insight of the solution. This lets us test whether retrieval benefits more from concise semantic guidance than from fuller procedural detail.

Reflection Reflect. This strategy rewrites a trace into a contrastive form of likely mistakes and how to avoid them. It highlights common errors, misleading intuitions, and critical checks, along with a brief statement of the right approach. This provides diagnostic context that can help the model avoid unproductive paths and recover from likely errors.

Figure 2 shows an example where Gemini-2.5-Flash fails in both the LLM-only (No RAG) setup and with RAG over full traces, but succeeds once

the retrieved traces are transformed with Reflect. This illustrates that, while raw or unstructured retrieved reasoning may be insufficient and noisy, transformed traces can provide the right guidance to help the model complete the solution.

4 Experimental Setup

4.1 Thinking Trace Sources

We construct thinking-trajectory corpora from different LLMs and problem collections. We focus on a *shared-corpus* setting, where previously generated traces are reused across inference models, allowing us to test the *transferability* of reasoning traces i.e., whether traces from one "thinker" can guide a different "solver". We consider two large-scale sources of reasoning-intensive questions; additional details are provided in Appendix C:

$\mathcal{T}^3$-59k: A collection of 59K reasoning-intensive problems paired with Gemini-2-thinking traces , drawn primarily from math (53k) with additional science and general reasoning domains. We use the S1 data pipeline to curate this collection (Muennighoff et al., 2025).

$\mathcal{T}^3$-114K: A collection of 114K reasoning problems paired with QwQ-32B thinking traces. It spans mathematics (89K), code (20K), science (4K), and puzzles (1K), and is obtained from the OpenThoughts data recipe (Guha et al., 2025).

Decontamination. We take several steps to reduce contamination. First, we evaluate on recent benchmarks, including AIME 2025–2026, whose problems were released after the source problem collections used to construct our trace corpora. Second, following prior work (Borgeaud et al., 2022; Lyu et al., 2025), we decontaminate both trace collections against the evaluation benchmarks by removing samples whose similarity to an evaluation query exceeds a 13-gram Jaccard threshold. This removes approximately 1.8% of the data. In Appendix E, we further analyze the choice of this threshold and provide additional lexical and semantic contamination checks.

Transformation. All transformed variants are generated by applying the prompts in Appendix B with Gemini-2-Flash-Lite. We use a smaller model because transformation is applied once to the entire corpus. On the 59K corpus, each transformation costs at most $26.74, and all three transformed variants cost only $64.74 in total. Even with a substantially more expensive model, this remains a modest one-time preprocessing cost. Unlike trace generation, which requires expensive long-form reasoning from strong models, transformation is a lightweight rewrite of existing traces. This makes it practical to construct shared reasoning corpora that can be reused across inference models. We provide a cost and token breakdown in Appendix C.3.

4.2 Inference Setup

We consider a diverse set of frontier LLMs, GPT-5, GPT-OSS-120B, and Gemini-2.5-Flash, deliberately spanning different scales, reasoning capabilities, and open- and closed-source families to study how retrieval interacts with different deployment regimes and model generations while still focusing on strong contemporary reasoners.

For retrieval, we use `e5-base-v2` as our primary encoder for both queries and thinking traces, retrieving the top-3 documents by default. For each problem, we augment the original question with the retrieved examples, formatted as a hint-augmented prompt shown in Figure 7. We compare retrieval over full trajectories, which treat each thinking trace as a single retrieval unit, with chunked trajectories, where traces are split into fixed-length 512-token segments. For transformed traces, we use 2048-token retrieval units and do not apply additional chunking, since they are already substantially shorter on average, typically below 1000 words. Further analysis of trace lengths is provided in Appendix C. We also provide ablations on the number of retrieved documents and on using Qwen3-Embedding-0.6B in Appendix D. These ablations follow the same pattern as our main results i.e., top-3 retrieval is a stable default, and the choice of encoder does not change our main conclusions.

4.3 Baselines

We compare against No RAG LLM-only setup, and RAG over several general-purpose corpora. These include major CompactDS subsets (Lyu et al., 2025): OpenWebMath (6.4M documents), StackExchange (29.8M), Wikipedia-DPR (21.0M), Wikipedia-RPJ (29.8M), GitHub (28.8M), and ArXiv (1.6M). All are chunked into 512-token passages and indexed with the same `e5-base-v2` retriever. We also include random retrieval, where passages are sampled uniformly across all general-purpose corpora, to verify that gains come from relevant retrieval. Indexing the full CompactDS corpus (639M+ documents) with our setup is computationally expensive, so we additionally report DS-Serve results (Liu et al., 2026), which serve the

	AIME 2025–2026						GPQA-Diamond						LiveCodeBench					
	GPT-5		GPT-OSS 120B		Gemini-2.5 Flash		GPT-5		GPT-OSS 120B		Gemini-2.5 Flash		GPT-5		GPT-OSS 120B		Gemini-2.5 Flash	
Corpus	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%
Baseline																		
No RAG	86.7		78.3		53.3		83.8		70.7		77.3		57.4		57.9		45.1	
General-purpose corpora																		
Random	86.7	(0.0%)	73.3	(-6.4%)	53.3	(0.0%)	83.3	(-0.6%)	70.7	(0.0%)	76.3	(-1.3%)	58.4	(+1.7%)	55.0	(-5.0%)	46.0	(+2.0%)
OpenWebMath	85.0	(-2.0%)	63.3	(-19.2%)	45.0	(-15.6%)	82.8	(-1.2%)	69.2	(-2.1%)	75.8	(-1.9%)	58.9	(+2.6%)	37.6	(-35.1%)	39.6	(-12.2%)
StackExchange	83.3	(-3.9%)	76.7	(-2.0%)	46.7	(-12.4%)	83.3	(-0.6%)	34.3	(-51.5%)	76.8	(-0.6%)	57.4	(0.0%)	55.0	(-5.0%)	42.1	(-6.7%)
Wikipedia-DPR	88.3	(+1.8%)	71.7	(-8.4%)	56.7	(+6.4%)	84.3	(+0.6%)	71.7	(+1.4%)	80.3	(+3.9%)	59.9	(+4.4%)	59.4	(+2.6%)	46.5	(+3.1%)
Wikipedia-RPJ	90.0	(+3.8%)	76.7	(-2.0%)	41.7	(-21.8%)	85.9	(+2.5%)	72.2	(+2.1%)	78.3	(+1.3%)	58.9	(+2.6%)	58.9	(+1.7%)	45.0	(-0.2%)
GitHub	91.7	(+5.8%)	76.7	(-2.0%)	60.0	(+12.6%)	84.8	(+1.2%)	68.2	(-3.5%)	80.8	(+4.5%)	56.9	(-0.9%)	54.5	(-5.9%)	42.1	(-6.7%)
Arxiv	85.0	(-2.0%)	78.3	(0.0%)	51.7	(-3.0%)	84.8	(+1.2%)	69.7	(-1.4%)	77.3	(0.0%)	57.9	(+0.9%)	57.4	(-0.9%)	46.9	(+4.0%)
CompactDS	88.3	(+1.8%)	80.0	(+2.2%)	58.3	(+9.4%)	82.8	(-1.2%)	67.7	(-4.2%)	77.3	(0.0%)	60.9	(+6.1%)	57.4	(-0.9%)	45.4	(+0.7%)
Tavily Search API	83.3	(-3.9%)	75.0	(-4.2%)	60.0	(+12.6%)	84.8	(+1.2%)	59.6	(-15.7%)	79.8	(+3.2%)	58.4	(+1.7%)	59.9	(+3.5%)	47.9	(+6.2%)
Thinking traces-based corpora																		
Full traj.	86.7	(0.0%)	73.3	(-6.4%)	73.3	(+37.5%)	80.8	(-3.6%)	69.2	(-2.1%)	76.3	(-1.3%)	57.9	(+0.9%)	58.9	(+1.7%)	46.5	(+3.1%)
Chunked traj.	91.7	(+5.8%)	**85.0**	**(+8.6%)**	80.0	(+50.1%)	84.8	(+1.2%)	71.7	(+1.4%)	79.3	(+2.6%)	**60.9**	**(+6.1%)**	58.9	(+1.7%)	**48.0**	**(+6.4%)**
$\mathcal{T}^3$-59k-Struct	91.7	(+5.8%)	81.7	(+4.3%)	73.3	(+37.5%)	**87.4**	**(+4.3%)**	70.7	(0.0%)	**80.8**	**(+4.5%)**	60.4	(+5.2%)	**61.4**	**(+6.0%)**	47.0	(+4.2%)
$\mathcal{T}^3$-59k-Reflect	**93.3**	**(+7.6%)**	81.7	(+4.3%)	76.7	(+43.9%)	84.3	(+0.6%)	71.7	(+1.4%)	79.3	(+2.6%)	59.9	(+4.4%)	58.4	(+0.9%)	45.0	(-0.2%)
$\mathcal{T}^3$-59k-Semantic	88.3	(+1.8%)	83.3	(+6.4%)	**83.3**	**(+56.3%)**	86.4	(+3.1%)	**74.7**	**(+5.7%)**	78.8	(+1.9%)	58.9	(+2.6%)	60.9	(+5.2%)	45.0	(-0.2%)

Table 2: Results on AIME 2025–2026, GPQA-Diamond, and LiveCodeBench. The best score in each column is shown in bold; green cells mark the three highest accuracies per column and red cells the three lowest (ties included). The relative improvement over the No RAG baseline is reported.

full corpus with Contriever (Izacard et al., 2021). We also include Tavily Search API[1] as a commercial real-time web search baseline. All corpora and retrieved results are decontaminated with respect to the evaluation benchmarks as described in Section 4.1.

4.4 Evaluation

Benchmarks. We evaluate on a diverse set of reasoning benchmarks including 1) AIME (2025–2026): Competition-level math problems, where each year consists of 30 questions. 2) GPQA-Diamond: A benchmark of 198 graduate-level scientific questions across biology, chemistry, and physics (Rein et al., 2023). 3) LiveCodeBench: A subset of 202 programming problems constructed from LCB V3 and V4 (Jain et al., 2024).

Evaluation. We evaluate all pipelines using EleutherAI LM Evaluation Harness (Gao et al., 2024) with custom task definitions. We query the target model through an OpenRouter-hosted OpenAI-compatible interface. Unless otherwise specified, we allow up to 16K generation tokens and use a temperature of 0.6 when applicable.

To reduce variance from stochastic generation, we sample multiple independent responses per problem. For AIME, we use 8 samples per query and report Average@8. For the larger GPQA-Diamond and LiveCodeBench benchmarks, we use 4 samples per query and report Average@4.

Answers are automatically extracted from model outputs and scored against the gold solution. For AIME and GPQA-Diamond, we report exact-match accuracy. For LiveCodeBench, each sampled program is evaluated using the standard pass@1 criterion, and the reported score is averaged over 4 samples. When simple parsing is insufficient, we use GPT-4o-mini only for answer normalization during post-processing.

[1] https://www.tavily.com/

5 Results

We study three research questions:

- **RQ1:** Is RAG over thinking traces helpful for reasoning-intensive tasks?
- **RQ2:** Can thinking traces be transformed to serve as more effective context for RAG?
- **RQ3:** How does retrieval over thinking traces affect the inference cost?

5.1 RQ1: Retrieval over Thinking Traces

We compare three experimental settings of No RAG, retrieval over general-purpose corpora, and retrieval over raw thinking traces in Table 2. The key pattern is that general-purpose retrieval is highly corpus- and task-dependent and it is inconsistent, whereas retrieval over thinking traces provides a more reliable signal for reasoning tasks and improves performance, often substantially.

General-purpose corpora provide mixed results. Some sources help in isolated cases, but none improves all models on benchmarks. For example, OpenWebMath and StackExchange often hurt AIME performance, while Wikipedia, ArXiv, and CompactDS alternate depending on the model and task. This remains true even for CompactDS, despite its much larger scale (639M+ documents).

	AIME 2025–2026						GPQA-Diamond						LiveCodeBench					
	GPT-5		GPT-OSS 120B		Gemini-2.5 Flash		GPT-5		GPT-OSS 120B		Gemini-2.5 Flash		GPT-5		GPT-OSS 120B		Gemini-2.5 Flash	
Corpus	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%
No RAG	86.7		78.3		53.3		83.8		70.7		77.3		57.4		57.9		45.1	
Output	81.7	(-5.8%)	83.3	(+6.4%)	68.3	(+28.1%)	84.8	(+1.2%)	70.7	(0.0%)	80.8	(+4.5%)	59.4	(+3.5%)	58.4	(+0.9%)	**56.4**	(+25.1%)
Thinking trajectories	**91.7**	**(+5.8%)**	**85.0**	**(+8.6%)**	**80.0**	**(+50.1%)**	84.8	(+1.2%)	**71.7**	**(+1.4%)**	79.3	(+2.6%)	**60.9**	**(+6.1%)**	**58.9**	**(+1.7%)**	48.0	(+6.4%)
QwQ-32B	86.7	(0.0%)	78.3	(0.0%)	68.3	(+28.1%)	83.3	(-0.6%)	71.2	(+0.7%)	79.3	(+2.6%)	57.4	(0.0%)	**59.9**	**(+3.5%)**	44.1	(-2.2%)
GPT-OSS-120B	90.0	(+3.8%)	80.0	(+2.2%)	45.0	(-15.6%)	52.0	(-37.9%)	70.2	(-0.7%)	77.8	(+0.6%)	57.9	(+0.9%)	55.9	(-3.5%)	42.1	(-6.7%)
Gemini-2-thinking	**91.7**	**(+5.8%)**	**85.0**	**(+8.6%)**	**80.0**	**(+50.1%)**	**84.8**	**(+1.2%)**	**71.7**	**(+1.4%)**	**79.3**	**(+2.6%)**	**60.9**	**(+6.1%)**	58.9	(+1.7%)	**48.0**	**(+6.4%)**

Table 3: Ablations on the retrieval source for RAG on reasoning traces. *Top*: comparing retrieval over corpus of thinking trajectories against retrieval over the same model's final output corpus, showing that the gains stem from access to the reasoning process rather than from exposure to related problems alone. *Bottom*: comparing RAG thinking trajectories generated by three different thinker models over the same 59K-problem set, showing that trace quality from the thinker strongly affects downstream performance.

Similarly, Tavily Search API, a real-time web retrieval baseline, does not yield consistent gains. Overall, these results suggest that the bottleneck is not simply whether the corpus is large or web-scale, but whether the retrieved content is aligned with the reasoning process required by the task.

In contrast, retrieval over thinking traces is substantially more effective. On AIME, Gemini-2.5-Flash improves from 53.3 to 73.3 (+37.5%) with full traces, and further to 80.0 (+50.1%) with simple chunking. These gains are much larger than those from general-purpose retrieval, even though the thinking-trace corpus contains only ∼59K traces, orders of magnitude fewer than the general-purpose corpora. The benefit is not limited to weaker models: GPT-5 improves from 86.7 to 91.7 (+5.8%) with chunked traces, showing that reasoning-oriented retrieval remains useful for frontier models. While our trace corpus is heavily skewed toward mathematics (Appendix C), we still observe improvements on GPQA and LiveCodeBench, though the gains are more modest.

A consistent pattern is that chunked traces outperform full traces, suggesting that long raw traces are often too verbose and noisy to serve as effective retrieval units. This motivates transforming traces into compact, retrieval-friendly representations.

Retrieval on Thinking Traces vs Output. We next compare retrieval over thinking trajectories with retrieval over final outputs in Table 3. While output retrieval is often beneficial, full reasoning trajectories are generally stronger, suggesting that gains come not only from related problems but from intermediate reasoning. The advantage is clearest on AIME, where thinking traces outperform output-only retrieval across all models. On GPQA and LiveCodeBench, the gap is smaller and occasionally mixed; for Gemini-2.5-Flash, output attempts slightly outperform traces on both benchmarks. Overall, RAG on thinking traces provides richer context than final answers alone.

Impact of Thinking Traces. We further analyze the impact of the model generating the thinking traces in the bottom section of Table 3 in the Appendix. To isolate this effect, we generate traces over the same set of 59K problems from (Muennighoff et al., 2025) using three different thinkers: QwQ-32B, GPT-OSS-120B, and Gemini-2-thinking. As shown in Table 3, we observe that the quality of the thinker strongly affects downstream performance. While traces from QwQ-32B and GPT-OSS-120B are often helpful, Gemini-2-thinking consistently produces the most effective retrieval corpus across benchmarks and models. Notably, this holds despite all traces being derived from the same problems, indicating that how the reasoning is expressed matters more than the underlying data itself.

5.2 RQ2: Transforming Thinking Traces

In RQ1, we found that chunked raw traces often outperform full trajectories. We now ask whether transforming those traces can produce even better retrieval corpora. Results in the last section of Table 2 show that RAG with $\mathcal{T}^3$ consistently outperforms both raw-trace retrieval and general-purpose corpora. This indicates that not only the presence of reasoning traces, but also how they are represented, plays a critical role in their usefulness.

The impact of transformation is most pronounced on AIME 2025–2026. For example, using $\mathcal{T}^3$-59k, Reflect reaches 93.3 for GPT-5, outperforming both No RAG (86.7) and the best raw-trace baseline (91.7). For Gemini-2.5-Flash, Semantic reaches 83.3, again improving over No RAG (53.3) and raw full-trace retrieval (73.3), while RAG over general-purpose corpora reaches at most 60.0 in this setting.

The best transformation depends on the task.

On GPQA-Diamond, Struct performs best for GPT-5 and Gemini-2.5-Flash, reaching 87.4 and 80.8, while Semantic performs best for GPT-OSS-120B at 74.7. On LiveCodeBench, transformed traces remain competitive and often outperform raw traces. Across tasks, they consistently outperform general-purpose retrieval, even when domain mismatch limits absolute gains. Additionally, transformation gains are larger for weaker inference models. On AIME, Gemini-2.5-Flash improves by 56.3%, from 53.3 to 83.3, with RAG on $\mathcal{T}^3$, while GPT-5 improves by 7.6%, from 86.7 to 93.3. This suggests that transformation is especially valuable when models rely more on external reasoning signals.

Impact of the problem source. We further study the impact of the underlying problem collection in Table 4 in the Appendix C.2. Although $\mathcal{T}^3$-114K is derived from a larger and broader problem set, $\mathcal{T}^3$-59K often yields stronger downstream performance after transformation. This is especially clear on AIME and GPQA, where transformed traces from $\mathcal{T}^3$-59K outperform those from $\mathcal{T}^3$-114K across most models and transformation types. These results suggest that corpus size alone is not the main driver of reasoning-oriented retrieval; the alignment between the source problem set, its reasoning style, and the target benchmarks is also important.

Interestingly, our shared-corpus setup also lets us test whether reasoning traces can transfer across models, even when they are produced by a different or older thinker. The answer is often yes: For example as shown in Table 3 Gemini-2.5-Flash benefits substantially from Gemini-2-thinking traces, and GPT-OSS-120B also benefits from QwQ-32B traces, showing that the value of reasoning traces can transfer across model generations and families.

Overall, in response to RQ2, we find that **while raw traces are already useful, transforming them into cleaner, more compact, or more diagnostic representations often yields stronger performance across tasks and models.**

5.3 Cost–Accuracy Trade-offs

Figure 3 summarizes the average cost–accuracy trade-off across the three benchmarks. We report average inference cost per question, including both input and output tokens, together with average accuracy. We compare No RAG, RAG over full raw trajectories, and RAG with the best-performing $\mathcal{T}^3$ variant. A clear pattern is that full-trace retrieval is consistently the most expensive setting: it adds long contexts and often increases total inference cost. In contrast, $\mathcal{T}^3$ provides a better cost–accuracy frontier by replacing full trajectories with shorter, more targeted transformed traces. For GPT-5, $\mathcal{T}^3$ improves accuracy from 76.14 to 80.53 while reducing cost from 1.22 to 1.04 cents per query, a 14.8% cost reduction relative to No RAG. For GPT-OSS-120B, accuracy improves from 68.99 to 74.82 while cost decreases slightly from 0.10 to 0.09 cents per query. For Gemini-2.5-Flash, $\mathcal{T}^3$ increases cost relative to No RAG, but still achieves substantially higher accuracy and remains cheaper and more accurate than full-trace retrieval.

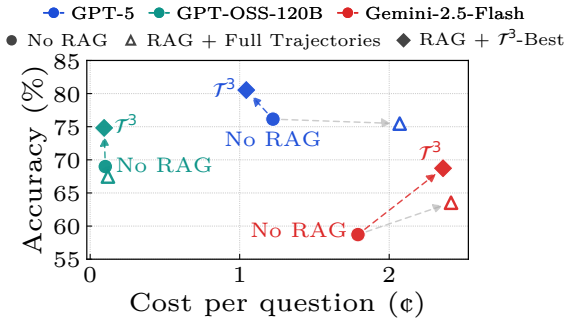


Figure 3: Average cost–accuracy trade-off across the three benchmarks. We compare No RAG, RAG over full thinking traces, and RAG with the best $\mathcal{T}^3$ variant from Table 2. Cost includes total tokens under each model's pricing; $\mathcal{T}^3$ gives the strongest overall trade-off, often improving accuracy at lower or comparable cost.

Overall, $\mathcal{T}^3$ consistently dominates full thinking trace retrieval on the cost–accuracy frontier, and in some cases even improves reasoning at lower cost than No RAG. The effect is not universal, however: whether retrieved traces substitute for generation or stimulate more reasoning is strongly model-dependent.

6 Conclusion

We revisit the role of retrieval in reasoning tasks and show that the limitation of RAG is not retrieval itself, but the choice of retrieval corpus. By shifting from retrieving documents to retrieving thinking traces, we demonstrate that even a simple retrieval-then-generate pipeline can significantly improve reasoning performance. Our results show that raw thinking traces are already a strong retrieval source, and that transforming them with $\mathcal{T}^3$ into more structured, compact, and diagnostic forms yields further gains across models and tasks. More broadly, we argue that thinking traces should be treated as a reusable resource that can be stored, transformed, and retrieved to support future reasoning.

Limitations

This work has several limitations. First, we study only vanilla RAG. This choice is intentional: our goal is to test whether simple retrieval over thinking traces can help reasoning in the first place. We leave more complex retrieval settings, such as iterative, adaptive, or reasoning-time retrieval, to future work.

Second, the trace corpora used in this work are heavily skewed toward mathematics. This likely limits gains on domains such as scientific QA and code, and may restrict generalization to other reasoning-intensive tasks.

Third, although we study traces from different strong reasoning models, we do not fully disentangle the effect of the thinker. In particular, we do not run a fully controlled comparison where the same auxiliary problem set is solved by many different thinker models and then evaluated with the same downstream solver. Such experiments would be informative, but are computationally expensive at the scale required here.

Despite these limitations, our results show that retrieval can be effective for reasoning when the right type of content is used. We hope this motivates future work on richer trace corpora, broader task coverage, and more adaptive retrieval mechanisms for reasoning-intensive RAG.

Acknowledgments

AI assistants were used for editing and proofreading the manuscript. All scientific claims, experimental design, analyses, and final content were verified by the authors.

References

Ammar Ahmed, Azal Ahmad Khan, Ayaan Ahmad, Sheng Di, Zirui Liu, and Ali Anwar. 2025. Retrieval-of-thought: Efficient reasoning via reusing thoughts. *Preprint*, arXiv:2509.21743.

Sören Auer, Dante AC Barone, Cassiano Bartz, Eduardo G Cortes, Mohamad Yaser Jaradeh, Oliver Karras, Manolis Koubarakis, Dmitry Mouromtsev, Dmitrii Pliukhin, Daniil Radyush, and 1 others. 2023. The sciqa scientific question answering benchmark for scholarly knowledge. *Scientific Reports*, 13(1):7240.

Parishad BehnamGhader, Santiago Miret, and Siva Reddy. 2023. Can retriever-augmented language models reason? the blame game between the retriever and the language model. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15492–15509.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, and 1 others. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Xingbo Du, Loka Li, Duzhen Zhang, and Le Song. 2025. Memr^3: Memory retrieval via reflective reasoning for llm agents. *Preprint*, arXiv:2512.20237.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pages 6491–6501.

Jinyuan Fang, Zaiqiao Meng, and Craig MacDonald. 2024. TRACE the evidence: Constructing knowledge-grounded reasoning chains for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8472–8494, Miami, Florida, USA. Association for Computational Linguistics.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, Haofen Wang, and 1 others. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1):32.

Gemini Team. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Shangyi Geng, Wenting Zhao, and Alexander M Rush. 2024. Great memory, shallow reasoning: Limits of $k$nn-lms. *Preprint*, arXiv:2408.11815.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, and 31 others. 2025. Openthoughts: Data recipes for reasoning models. *Preprint*, arXiv:2506.04178.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. *Preprint*, arXiv:2212.10071.

Siqing Huo, Negar Arabzadeh, and Charles Clarke. 2023. Retrieving supporting evidence for generative question answering. In *Proceedings of the annual international acm sigir conference on research and development in information retrieval in the Asia Pacific region*, pages 11–20.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *Preprint*, arXiv:2403.07974.

Zachary Levonian, Chenglu Li, Wangda Zhu, Anoushka Gade, Owen Henkel, Millie-Ellen Postle, and Wanli Xing. 2023. Retrieval-augmented generation to improve math question-answering: Trade-offs between groundedness and human preference. *Preprint*, arXiv:2310.03184.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Xingxuan Li, Weiwen Xu, Ruochen Zhao, Fangkai Jiao, Shafiq Joty, and Lidong Bing. 2025. Can we further elicit reasoning in llms? critic-guided planning with retrieval-augmentation for solving challenging tasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25589–25604.

Jingyu Liu, Jiaen Lin, and Yong Liu. 2024. How much can rag help the reasoning of llm? *arXiv preprint arXiv:2410.02338*.

Jinjian Liu, Yichuan Wang, Xinxi Lyu, Rulin Shao, Joseph E Gonzalez, Matei Zaharia, and Sewon Min. 2026. Ds serve: A framework for efficient and scalable neural retrieval.

Xinxi Lyu, Michael Duan, Rulin Shao, Pang Wei Koh, and Sewon Min. 2025. Frustratingly simple retrieval improves challenging, reasoning-intensive benchmarks. *Preprint*, arXiv:2507.01297.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. *Preprint*, arXiv:2212.08410.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

OpenAI Team. 2025a. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

OpenAI Team. 2025b. Openai gpt-5 system card. *Preprint*, arXiv:2601.03267.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *Preprint*, arXiv:2311.12022.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, and 7 others. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Rulin Shao, Jacqueline He, Akari Asai, Weijia Shi, Tim Dettmers, Sewon Min, Luke Zettlemoyer, and Pang Wei Koh. 2024. Scaling retrieval-based language models with a trillion-token datastore. *Preprint*, arXiv:2407.12854.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen tau Yih, Pang Wei Koh, and Luke Zettlemoyer. 2025. Reasonir: Training retrievers for reasoning tasks. *Preprint*, arXiv:2504.20595.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073.

Ronit Singal, Pransh Patwa, Parth Patwa, Aman Chadha, and Amitava Das. 2024. Evidence-backed fact checking using rag and few-shot in-context learning with llms. In *Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER)*, pages 91–98.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering. *Transactions of the Association for Computational Linguistics*, 11:1–17.

Peng-Yuan Wang, Tian-Shuo Liu, Chenyang Wang, Ziniu Li, Yidi Wang, Shu Yan, Chengxing Jia, Xu-Hui Liu, Xinwei Chen, Jiacheng Xu, and 1 others. 2026. A survey on large language models for mathematical reasoning. *ACM Computing Surveys*, 58(8):1–35.

Peng-Yuan Wang, Tian-Shuo Liu, Chenyang Wang, Yi-Di Wang, Shu Yan, Cheng-Xing Jia, Xu-Hui Liu, Xin-Wei Chen, Jia-Cheng Xu, Ziniu Li, and Yang Yu. 2025. A survey on large language models for mathematical reasoning. *Preprint*, arXiv:2506.08446.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Zihao Wang, Anji Liu, Haowei Lin, Jiaqi Li, Xiaojian Ma, and Yitao Liang. 2024. Rat: Retrieval augmented thoughts elicit context-aware reasoning in long-horizon generation. *Preprint*, arXiv:2403.05313.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. 2024. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems*, 37:113519–113544.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Appendix

A Extended Related Work

Reasoning and Retrieval. LLMs have shown strong performance on reasoning-intensive tasks such as mathematical problem solving, scientific question answering, and code generation (Wang et al., 2025; Rozière et al., 2024; Auer et al., 2023). Prior work improves reasoning through prompting strategies such as chain-of-thought and self-consistency (Wei et al., 2022; Wang et al., 2022), distillation from stronger reasoning models (Ho et al., 2023; Magister et al., 2023; Shridhar et al., 2023; Muennighoff et al., 2025), and reinforcement learning with verifiable rewards (Guo et al., 2025; Yu et al., 2025). These approaches generally aim to improve the model's reasoning behavior directly, either through prompting, training, or distillation. Our work is complementary: we ask whether prior reasoning traces can instead be stored externally and reused through retrieval at inference time.

RAG has become a standard approach for improving LLMs on knowledge-intensive tasks by retrieving external documents that provide factual grounding and reduce hallucinations (Lewis et al., 2020; Gao et al., 2023; Siriwardhana et al., 2023). Most work focuses on retrieving textual evidence from large corpora and improving how it is selected, structured, and incorporated into the model input (Fan et al., 2024; Singal et al., 2024; Huo et al., 2023). Recent work also studies retrieval from a scaling perspective: Shao et al. (2024) show that increasing datastore size can improve retrieval-based language models. However, for reasoning-intensive tasks, retrieved documents may provide relevant facts without exposing the reasoning process needed to use them. This motivates retrieval sources that contain process-level signals rather than only factual evidence.

RAG for Reasoning. A growing body of work explores retrieval in reasoning settings. Levonian et al. (2023) show that retrieval can improve math question answering, but also introduces trade-offs between groundedness and human preference. RAT (Wang et al., 2024) tightly couples retrieval with generation: it first produces an initial chain of thought and then revises each reasoning step using retrieved information. TRACE (Fang et al., 2024) focuses on multi-hop QA and converts retrieved documents into knowledge-grounded reasoning chains using additional LLM calls before answer generation. These methods show that retrieval can help reasoning when retrieved information is structured or integrated into intermediate reasoning, but they still rely on online reasoning-chain construction or revision for each query.

Thought Reuse and Template-Based Reasoning. BoT (Yang et al., 2024) and RoT (Ahmed et al., 2025) are closest in spirit to our work because they also reuse prior reasoning artifacts. BoT maintains a meta-buffer of high-level thought templates distilled from previous problem-solving processes; for a new problem, it retrieves a relevant template and instantiates it through a dedicated reasoning procedure. RoT decomposes prior solutions into reusable reasoning steps, organizes them into a thought graph, and performs reward-guided traversal at inference time to assemble a problem-specific template. These methods demonstrate the value of reusable reasoning artifacts, but their retrieved objects are templates or graph nodes rather than native long-form thinking traces produced by reasoning models, and their use requires specialized inference-time mechanisms such as template instantiation, graph traversal, or thinking intervention.

Comparison to Our Work. Table 1 summarizes the main distinctions. Prior work improves reasoning RAG by modifying how retrieval is used at inference time: revising chains of thought, constructing evidence chains, instantiating templates, or traversing thought graphs. In contrast, our work studies the corpus-design question under a fixed retrieve-then-generate pipeline. We transform existing thinking traces offline into structured, compact, or diagnostic retrieval units, and then use them directly as context in a standard RAG setup.

B Prompts

Here, we present the prompts used for transforming thinking traces and for RAG inference. The prompt used for transformation strategies introduced in Section 3.2, namely Structural Normalization (Struct), Semantic Distillation (Semantic), and Reflection (Reflect) are shown in Figure 4, 5 and 6, respectively. Additionally, we provide our simple RAG inference prompt in Figure 7. These prompts are applied to construct the transformed corpora and to guide the model at inference time.

$\mathcal{T}^3$ - Structural Normalization

Instruction. Convert the reasoning trace into a concise step-by-step cheatsheet.

Guidelines.
- Use at most 7 steps.
- Each step should represent a meaningful action.
- Keep explanations short and clear.
- Focus on reusable reasoning patterns.
- Remove failed or irrelevant attempts.

Output format.
```
Problem: ...
Step 1: ...
...
Step N: ...
Answer: $\boxed{[final answer]}$
```

Given trace: `{trace}`

Figure 4: Prompt for Struct transformation.

$\mathcal{T}^3$ - Semantic Distillation

Instruction. Compress the reasoning trace into progressively more abstract representations.

Guidelines.
- Provide three levels of abstraction.
- Pass 1: structured outline of reasoning steps.
- Pass 2: only key decisions or moves.
- Pass 3: a single core insight.
- Ensure consistency across all passes.
- Avoid redundancy.

Output format.
```
Problem: ...
Pass 1: ...
Pass 2: ...
Pass 3: ...
Answer: $\boxed{[final answer]}$
```

Given trace: `{trace}`

Figure 5: Prompt for Semantic transformation.

$\mathcal{T}^3$ - Reflection

Instruction. Extract failure patterns and negative knowledge from the reasoning trace.

Guidelines.
- Focus on common mistakes and misleading reasoning paths.
- Explain why these mistakes are tempting.
- Highlight how to detect and avoid them.
- Provide contrast with the correct approach.
- Do not reproduce the full solution.

Output format.
```
Problem: ...
Common Mistakes: ...
Misleading Intuitions: ...
Critical Checks: ...
Correct Approach (brief): ...
Answer: $\boxed{[final answer]}$
```

Given trace: `{trace}`

Figure 6: Prompt for Reflect transformation.

RAG Inference

Instruction. Solve the main problem by using useful hints and strategies from the retrieved examples.

```
Example 1: ...
Example 2: ...
Example 3: ...
Main problem: ...
```

Figure 7: Prompt for RAG inference using retrieved examples.

		AIME 2025–2026						GPQA-Diamond						LiveCodeBench					
		GPT-5		GPT-OSS 120B		Gemini-2.5 Flash		GPT-5		GPT-OSS 120B		Gemini-2.5 Flash		GPT-5		GPT-OSS 120B		Gemini-2.5 Flash	
	Method	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%
	No RAG	86.7		78.3		53.3		83.8		70.7		77.3		57.4		57.9		45.1	
$\mathcal{T}^3$-114K	Struct	91.7	(+5.8%)	**88.3**	**(+12.8%)**	65.0	(+22.0%)	81.3	(-3.0%)	69.7	(-1.4%)	76.8	(-0.6%)	59.4	(+3.5%)	57.9	(0.0%)	46.0	(+2.0%)
	Reflect	90.0	(+3.8%)	80.0	(+2.2%)	65.0	(+22.0%)	84.3	(+0.6%)	70.2	(-0.7%)	65.7	(-15.0%)	59.4	(+3.5%)	55.4	(-4.3%)	44.6	(-1.1%)
	Semantic	88.3	(+1.8%)	80.0	(+2.2%)	60.0	(+12.6%)	84.8	(+1.2%)	71.7	(+1.4%)	58.1	(-24.8%)	59.4	(+3.5%)	59.9	(+3.5%)	43.6	(-3.3%)
$\mathcal{T}^3$-59K	Struct	91.7	(+5.8%)	81.7	(+4.3%)	73.3	(+37.5%)	**87.4**	**(+4.3%)**	70.7	(0.0%)	**80.8**	**(+4.5%)**	**60.4**	**(+5.2%)**	**61.4**	**(+6.0%)**	**47.0**	**(+4.2%)**
	Reflect	**93.3**	**(+7.6%)**	81.7	(+4.3%)	76.7	(+43.9%)	84.3	(+0.6%)	71.7	(+1.4%)	79.3	(+2.6%)	59.9	(+4.4%)	58.4	(+0.9%)	45.0	(-0.2%)
	Semantic	88.3	(+1.8%)	83.3	(+6.4%)	**83.3**	**(+56.3%)**	86.4	(+3.1%)	**74.7**	**(+5.7%)**	78.8	(+1.9%)	58.9	(+2.6%)	60.9	(+5.2%)	45.0	(-0.2%)

Table 4: Ablation of $\mathcal{T}^3$ on the problem source set. Although $\mathcal{T}^3$-114K is derived from a larger and broader problem set (114K vs. 59K problems), $\mathcal{T}^3$-59K often yields stronger downstream performance after transformation, suggesting that source-problem alignment and reasoning style can matter more than corpus size alone.

C Thinking Traces

C.1 Corpora Statistics

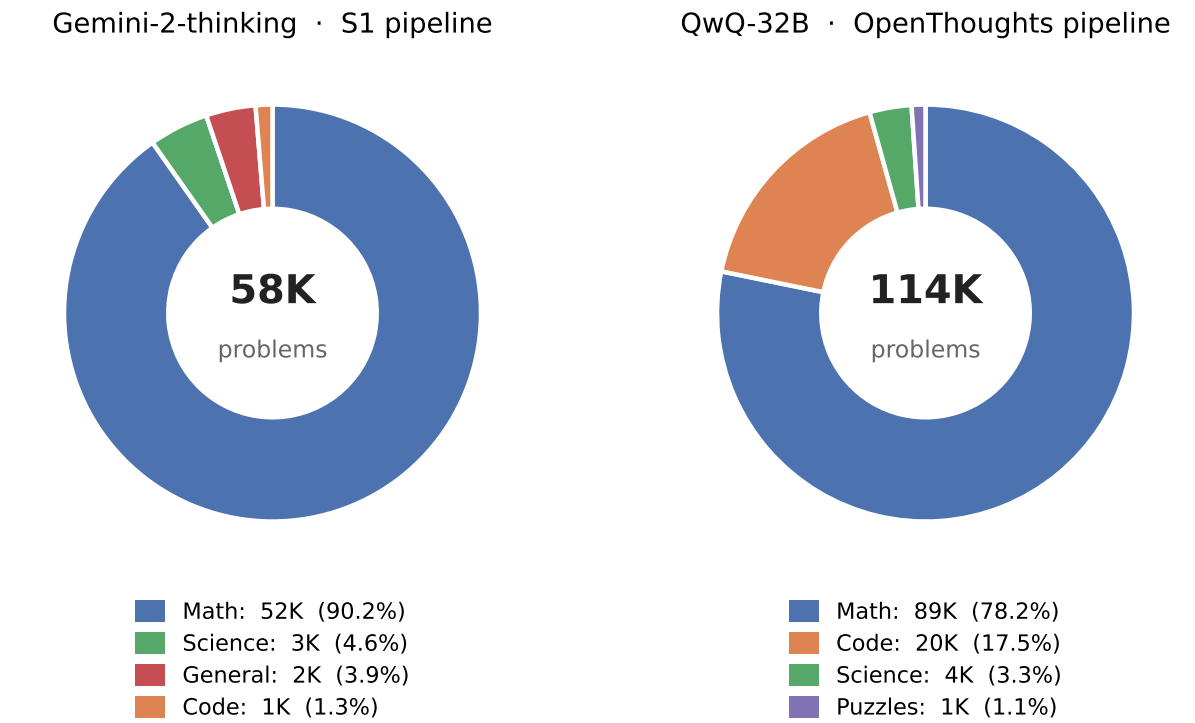
Here we show the domain distribution of the two thinking-trace corpora used in our experiments in Figure 8a. Both corpora are dominated by mathematical reasoning. $\mathcal{T}^3$-59k is more heavily skewed toward math (about 90%) and contains very little code, whereas $\mathcal{T}^3$-114k includes a larger code component (17.5%), reflecting its broader source coverage. Interestingly, despite being smaller and more narrowly focused, $\mathcal{T}^3$-59k often yields stronger RAG performance in our experiments. This suggests that trace quality may matter more than corpus size or breadth alone, potentially because the underlying Gemini-2-thinking model produces more useful reasoning traces for downstream retrieval.

We further analyze how transformation changes both the size and length of the resulting corpora. Structural normalization (Struct) increases the number of passages by 35% (78,522 vs. 58,071), because a single trajectory may be split into multiple procedural units when distinct steps or solution paths are extracted as separate documents. In contrast, semantic distillation (Semantic) and reflection (Reflect) preserve the original number of trajectories, since each trace is rewritten into a single transformed representation.

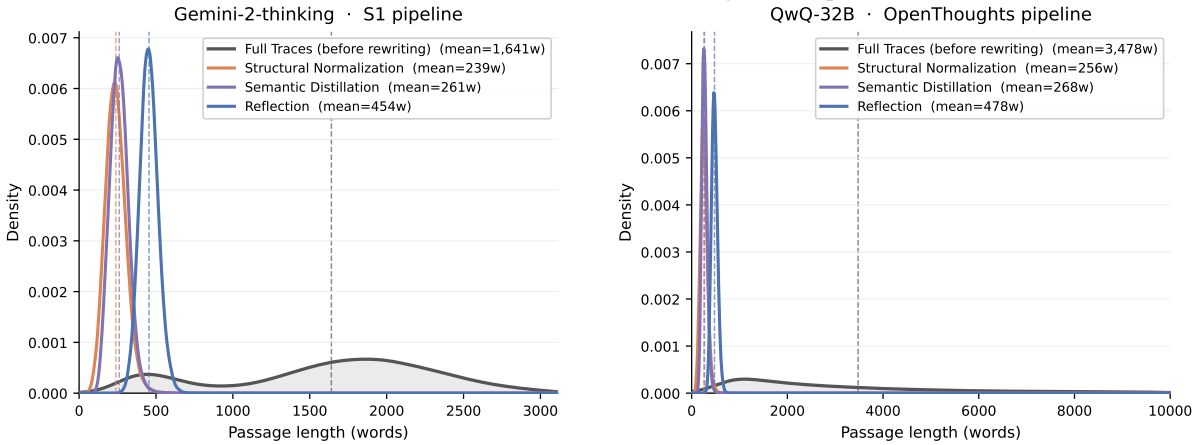
We also examine the distribution of passage lengths before and after transformation. As shown in Figure 8b, full trajectories are substantially longer than all transformed variants. For the Gemini-based corpus, full traces have mean length 1,641 words, compared to 239 for structural normalization, 261 for semantic distillation, and 454 for reflection. The same pattern holds for the QwQ-based corpus, where full traces average 3,478 words, while the transformed variants average 256, 268, and 478 words, respectively. Overall, all three transformations produce much more compact retrieval units, which improves retrieval efficiency and reduces input cost at inference time.

C.2 Problem Source Ablation

Table 4 compares transformed traces from two different problem sources: $\mathcal{T}^3$-114K, a larger and broader corpus, and $\mathcal{T}^3$-59K, a smaller corpus more concentrated on mathematical reasoning. Despite its smaller size, $\mathcal{T}^3$-59K often yields stronger downstream performance, particularly on AIME 2025–2026 and GPQA-Diamond. This suggests that, for reasoning-oriented retrieval, the usefulness



(a) Domain distribution of the two thinking-trace corpora.



(b) Passage length distribution before and after transformation.

Figure 8: Corpus statistics for thinking traces. (Left) Domain distribution of the two corpora. Both are dominated by mathematical reasoning. Despite being smaller (58K after decontamination vs. 114K), $\mathcal{T}^3$-Gemini often yields stronger RAG performance, suggesting that trace quality may matter more than corpus size. (Right) Passage length distributions before and after transformation for both corpora. All transformed variants are substantially shorter than full traces, improving retrieval efficiency and reducing inference cost.

of a trace corpus is not determined by scale alone. Source-problem alignment, reasoning style, and the structure of the generated traces can matter as much as, or more than, the number of available examples. Since the two corpora differ in both source distribution and trace-generation model, this ablation should be interpreted as a problem-source comparison rather than a controlled thinker-only comparison.

C.3 Transformation Cost

We report the estimated cost of generating all transformed variants for the 59K trace corpus in Table 5. Costs are computed using Gemini-2.0-Flash-Lite pricing of \$0.075 per million input tokens and \$0.30 per million output tokens. The three transformations together cost \$64.74 in total. Since this transformation is performed fully offline and only once per corpus, the resulting transformed traces can be reused across all downstream inference models and evaluation settings.

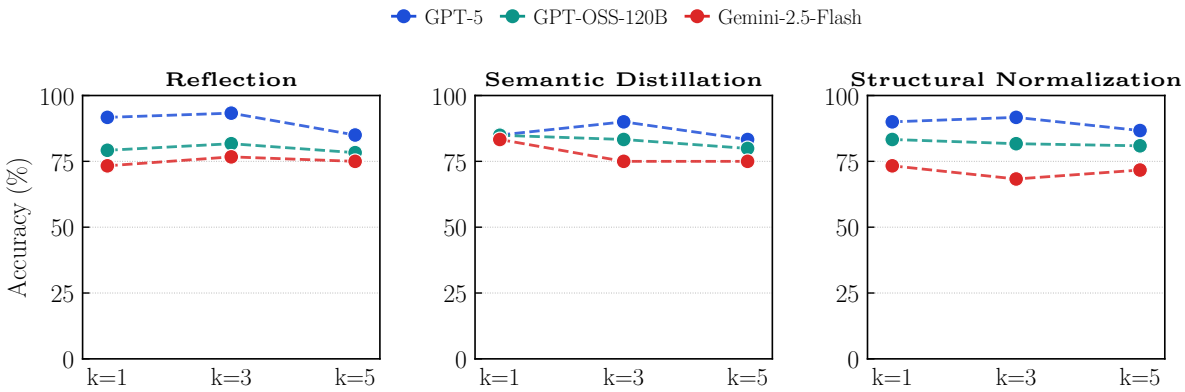


Figure 9: Impact of the number of retrieved documents ($k \in \{1, 3, 5\}$) on accuracy across three transformation strategies and three reader models on AIME 2025–2026. k=3 achieves the best or near-best accuracy most consistently across all settings.

Trans.	Input Tok.	Output Tok.	Input $	Output $
Reflect	184.13M	43.13M	$13.81	$12.94
Semantic	156.93M	22.73M	$11.77	$6.82
Struct	147.60M	27.73M	$11.07	$8.32
Total	488.66M	93.59M	**$36.65**	**$28.08**

Table 5: Cost of generating transformed traces for the 59K corpus using Gemini-2.0-Flash-Lite. Reflect corresponds to reflection, Semantic to semantic distillation, and Struct to structural normalization. The total cost is $64.74.

C.4 Transformation Example

Figures 11, 12, and 13 present representative examples from math, physics, and coding domains, respectively. Each figure shows an example problem and its full reasoning trace (truncated for brevity), along with its transformed variants. As shown in these examples, all transformed versions are significantly shorter compared to full traces, and each of them captures complementary aspects of the reasoning.

D Retrieval Ablation

D.1 Impact of Number of Retrieved Documents

We study how the number of retrieved documents affects downstream performance by evaluating top-k retrieval for $k \in \{1, 3, 5\}$ across all three $\mathcal{T}^3$ transformations and all target models. As shown in Figure 9, k=3 is the most consistent choice across models and methods. While k=1 or k=5 occasionally outperforms k=3 for a specific model–method combination, k=3 yields the best or near-best accuracy on average. Retrieving a single document (k=1) can miss complementary reasoning steps, and k=5 sometimes introduces noisy or redundant context that degrades generation quality. We therefore use k=3 in all main experiments.

D.2 Impact of Retriever

We evaluate whether our results depend on the choice of retriever by replacing `e5-base-v2` with Qwen3-Embedding-0.6B on Gemini-2-thinking traces. As shown in Table 6, the overall pattern remains similar across retrievers. `e5-base-v2` is slightly better in several settings, but Qwen3-Embedding-0.6B also matches or improves over it in some cases, such as AIME with GPT-OSS-120B and GPQA-Diamond with Gemini-2.5-Flash. Importantly, the main conclusion is unchanged: transformed thinking traces consistently provide gains over No RAG across models and benchmarks. This suggests that $\mathcal{T}^3$ is robust to the retriever choice rather than being tied to a single embedding model.

E Decontamination Analysis

We perform additional checks to assess the risk of contamination between our trace corpora and evaluation benchmarks. Our first safeguard is temporal: some evaluation benchmarks, most notably AIME 2025–2026, were released after the source problem collections used to construct our trace corpora. This reduces the likelihood that exact benchmark problems appear in the auxiliary trace-generation sets. We nevertheless apply lexical decontamination following prior work (Borgeaud et al., 2022; Lyu et al., 2025), removing examples whose 13-gram Jaccard overlap with any evaluation query exceeds 0.05.

	AIME 2025–2026						GPQA-Diamond						LiveCodeBench					
	GPT-5		GPT-OSS 120B		Gemini-2.5 Flash		GPT-5		GPT-OSS 120B		Gemini-2.5 Flash		GPT-5		GPT-OSS 120B		Gemini-2.5 Flash	
Corpus	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%	Acc.	Δ%
Baseline																		
No RAG	86.7		78.3		53.3		83.8		70.7		77.3		57.4		57.9		45.1	
e5-base-v2																		
$\mathcal{T}^3$- Struct	91.7	(+5.8%)	81.7	(+4.3%)	73.3	(+37.5%)	**87.4**	**(+4.3%)**	70.7	(0.0%)	**80.8**	**(+4.5%)**	**60.4**	(+5.2%)	**61.4**	**(+6.0%)**	**47.0**	(+4.2%)
$\mathcal{T}^3$-Reflect	**93.3**	**(+7.6%)**	81.7	(+4.3%)	76.7	(+43.9%)	84.3	(+0.6%)	71.7	(+1.4%)	79.3	(+2.6%)	59.9	(+4.4%)	58.4	(+0.9%)	45.0	(-0.2%)
$\mathcal{T}^3$- Semantic	88.3	(+1.8%)	83.3	(+6.4%)	**83.3**	**(+56.3%)**	86.4	(+3.1%)	**74.7**	**(+5.7%)**	78.8	(+1.9%)	58.9	(+2.6%)	60.9	(+5.2%)	45.0	(-0.2%)
Qwen3-Embedding-0.6B																		
$\mathcal{T}^3$- Struct	90.0	(+3.8%)	**88.3**	(+12.8%)	71.7	(+34.5%)	**87.4**	**(+4.3%)**	70.2	(-0.7%)	**80.8**	(+4.5%)	59.9	(+4.4%)	57.9	(0.0%)	46.5	(+3.1%)
$\mathcal{T}^3$-Reflect	90.0	(+3.8%)	81.7	(+4.3%)	76.7	(+43.9%)	84.8	(+1.2%)	71.2	(+0.7%)	77.8	(+0.6%)	58.9	(+2.6%)	57.9	(0.0%)	46.5	(+3.1%)
$\mathcal{T}^3$- Semantic	90.0	(+3.8%)	81.7	(+4.3%)	75.0	(+40.7%)	85.4	(+1.9%)	**74.7**	(+5.7%)	78.8	(+1.9%)	59.4	(+3.5%)	59.4	(+2.6%)	45.5	(+0.9%)

Table 6: Impact of retriever choice. Comparison between Qwen3-Embedding-0.6B and e5-base-v2 on retrieval over $\mathcal{T}^3$-59k.

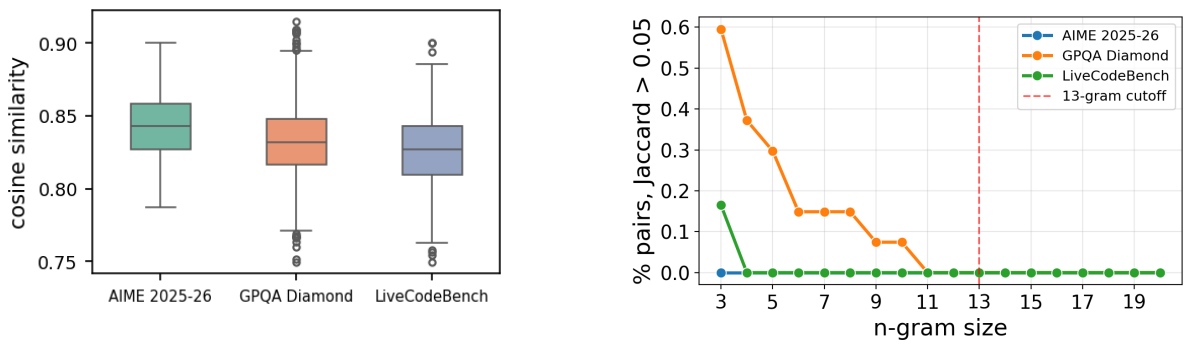


Figure 10: Semantic and lexical contamination diagnostics on the raw trajectory corpus. Left: `e5-base-v2` cosine similarity between benchmark queries and retrieved trajectory documents after 13-gram decontamination. Right: Percentage of retrieved query–passage pairs with n-gram Jaccard similarity above 0.05 as a function of n; the dashed line marks the 13-gram cutoff. High embedding similarity reflects semantic relatedness expected for retrieval, while long n-gram overlap drops to zero at the chosen threshold for all three benchmarks. Since this analysis uses raw trajectories, it is a conservative setting compared to the more compact transformed variants.

Figure 10 summarizes our lexical and semantic contamination checks on the raw Gemini-2-thinking trajectory corpus. We use raw trajectories for this analysis because they are the most surface-similar retrieval units; transformed traces are shorter and more abstract, and since they have been transformed, they form a less lexical contamination-prone setting. Figure 10 (left) shows the distribution of `e5-base-v2` cosine similarities between benchmark queries and retrieved trajectory documents. High embedding similarity is expected in a retrieval setting, since the retriever is designed to find conceptually related reasoning examples. We therefore treat embedding similarity as a diagnostic rather than direct evidence of contamination.

Figure 10(right) shows token n-gram Jaccard overlap between benchmark queries and retrieved trajectory documents, sweeping n from 3 to 20. Lexical overlap drops rapidly as n increases. At the 13-gram cutoff, no retrieved pairs exceed Jaccard > 0.05 for AIME 2025–2026, GPQA-Diamond, or LiveCodeBench. This supports 13-gram Jaccard as a conservative near-duplicate filter: smaller n-grams can capture benign topical overlap or shared technical phrases, while longer n-grams are more indicative of copied surface form.

Overall, the analysis supports our decontamination procedure. The 13-gram filter removes long surface-form overlap, while the remaining high embedding similarity reflects the intended behavior of retrieval: finding semantically related but textually distinct reasoning traces. Since this analysis uses raw trajectories, it is conservative; transformed traces are more compact and abstract, making them less likely to preserve benchmark-specific surface form.

		Tokens
Problem	n coins are simultaneously flipped. The probability that at most one shows tails is $\frac{3}{16}$. Find n.	
Full trace	The problem asks for the value of n, the number of coins flipped, given a probability condition on the number of tails observed. Let X be the number of tails observed when n coins are flipped. Each coin flip is an independent Bernoulli trial with probability of tails $p = 0.5$ ..... *[3,800 tokens omitted]* ..... The reasoning used the properties of the binomial distribution and solving the resulting equation. The uniqueness of the solution was established by analyzing the monotonicity of the function involved.	3,867
Struct	**Approach:** Solving the Probability Equation **Step 1** — Define $P(X \leq 1) = (1+n)(0.5)^n$. **Step 2** — Set $(1+n)(0.5)^n = \frac{3}{16}$; rewrite as $1+n = 3 \cdot 2^{n-4}$. **Step 3** — Test integer values; n=5 gives $1+5 = 3{\cdot}2^1 = 6$. ✓ **Step 4** — $(1+n)2^{-n}$ is strictly decreasing for $n \geq 1$, so the solution is unique. **Answer:** $\boxed{5}$	391
Semantic	**Pass 1 (full):** Let $X \sim \text{Bin}(n, 0.5)$. Then $P(X{\leq}1) = (1+n)(0.5)^n = \frac{3}{16}$, giving $1+n = 3 \cdot 2^{n-4}$. Testing integers yields $n = 5$; monotonicity of $(1+n)2^{-n}$ ensures uniqueness. **Pass 2 (core moves):** (i) Express $P(X{\leq}1)$ via binomial PMF $\Rightarrow (1+n)(0.5)^n = \frac{3}{16}$. (ii) Rewrite and solve by trial. (iii) Confirm uniqueness via monotonicity. **Pass 3 (insight):** Recognise the binomial condition; convert to $1+n = 3 \cdot 2^{n-4}$ and close by integer search. **Answer:** $\boxed{5}$	526
Reflect	**Trap profile:** Binomial probability — overlooking solution uniqueness. × **Assuming uniqueness** without proof: the problem says "find n" but a second solution could exist. × **Misapplying logarithms** to $(1+n)2^{-n}$, introducing or losing solutions. × **Forgetting the binomial coefficient** in $P(X{=}1) = n \cdot (0.5)^n$. ✓ Substitute back into the *original* equation, not a rearranged form. ✓ Prove $(1+n)2^{-n}$ is strictly decreasing to establish uniqueness. **Correct approach:** Reduce to $1+n = 3 \cdot 2^{n-4}$, solve by trial, confirm uniqueness. **Answer:** $\boxed{5}$	630

Figure 11: A single reasoning trace transformed by each strategy, with token counts. All three transformations are substantially more compact than the full trace (3,867 tokens), while preserving the key reasoning content in different ways.

		Tokens
Problem	A ball is launched with speed v_0 in a gravitational field. The boundary of the reachable region satisfies $z \leq z_0 - kx^2$. Find z_0 and k.	139
Full trace	The problem asks for the constants z_0 and k that define the boundary of the region reachable by a projectile launched with a fixed initial speed v_0. We are given the equation of the boundary ..... *[3,000 tokens omitted]* ..... The condition when $z = 0$ gives the maximum range on a flat surface, which also matches the known formula.	3,237
Struct	**Approach:** Envelope via Lagrange / angle optimisation **Step 1** — Write $x = v_0 \cos\theta\, t$, $z = v_0 \sin\theta\, t - \frac{1}{2}gt^2$; eliminate t. **Step 2** — Express $z = x\tan\theta - \frac{gx^2}{2v_0^2}(1+\tan^2\theta)$. **Step 3** — Set $y = \tan\theta$; maximise z over y: $\partial z/\partial y = x - \frac{gx^2}{v_0^2}y = 0 \Rightarrow y = v_0^2/(gx)$. **Step 4** — Substitute back: $z_{\max} = \frac{v_0^2}{2g} - \frac{g}{2v_0^2}x^2$. **Step 5** — Identify $z_0 = v_0^2/(2g)$, $k = g/(2v_0^2)$. **Answer:** $\boxed{z_0 = \frac{v_0^2}{2g},\ k = \frac{g}{2v_0^2}}$	494
Semantic	**Pass 1 (full):** Eliminate t from parametric equations to get $z = x\tan\theta - \frac{gx^2}{2v_0^2}(1+\tan^2\theta)$. Set $y = \tan\theta$, differentiate with respect to y, and set to zero to get $y = v_0^2/(gx)$. Substituting back yields $z_{\max} = v_0^2/(2g) - gx^2/(2v_0^2)$, so $z_0 = v_0^2/(2g)$ and $k = g/(2v_0^2)$. **Pass 2 (core moves):** (i) Derive trajectory $z(x,\theta)$ by eliminating t. (ii) Optimise over launch angle to find the envelope. (iii) Match coefficients to identify z_0 and k. **Pass 3 (insight):** The reachable boundary is the *envelope* of all parabolic trajectories; maximising z over the launch angle at fixed x directly gives it. **Answer:** $\boxed{z_0 = \frac{v_0^2}{2g},\ k = \frac{g}{2v_0^2}}$	648
Reflect	**Trap profile:** Projectile envelope — forgetting to optimise over angle; vertex formula shortcut. × **Fixed-angle fallacy:** Applying kinematic range/height formulas for a single θ instead of finding the envelope over all angles. × **Vertex shortcut:** Using the parabola vertex for a fixed θ as the maximum height — the vertex changes with θ and is not the boundary. × **Algebra error in elimination:** Dropping the $\sec^2\theta = 1 + \tan^2\theta$ factor when substituting t, yielding a wrong trajectory. ✓ Check $\theta = 90°$ (vertical launch): boundary gives $z = v_0^2/(2g)$, matching free-fall max height. ✓ Check $z = 0$: gives $x_{\max} = v_0^2/g$, consistent with the known range formula at $45°$. **Correct approach:** Derive $z(x,\theta)$, treat $\tan\theta$ as a free parameter, maximise at fixed x, match to $z_0 - kx^2$. **Answer:** $\boxed{z_0 = \frac{v_0^2}{2g},\ k = \frac{g}{2v_0^2}}$	724

Figure 12: Example of generated and transformed thinking traces from a physics problem.

		Tokens
Problem	A text editor supports one operation: replace every occurrence of character c_1 with c_2. Given an input string and a target string, find the minimum number of operations to transform one into the other, or return -1 if impossible.	570
Full trace	Let's analyze the input and the problem statement carefully. For each position i where $s_1[i] \neq s_2[i]$, we need a mapping $s_1[i] \to s_2[i]$ ..... *[3,900 tokens omitted]* ..... Consider `ABCD` $\to$ `BACD`. A needs to become B, B needs to become A — this requires 3 steps (an intermediate character is needed to break the cycle).	4,193
Struct	**Approach:** Transformation graph and connected components **Step 1** — Build mapping $s_1[i] \to s_2[i]$; if any character maps to two different targets, return -1. **Step 2** — Construct a directed graph: nodes are characters, edges are required transformations. **Step 3** — Find connected components via DFS/BFS. **Step 4** — Per component: if it contains a cycle, cost $=$ |nodes| $+ 1$ (one extra intermediate step); otherwise cost $=$ |edges|. **Step 5** — Sum costs across all components. **Answer:** minimum keystrokes, or -1 if impossible.	497
Semantic	**Pass 1 (full):** Build a character mapping from s_1 to s_2; any conflict (same source, two targets) is immediately -1. Represent mappings as a directed graph and find connected components. For each component, count cycles: each cycle of length n needs $n + 1$ operations (one intermediate); a simple path needs as many operations as edges. **Pass 2 (core moves):** (i) Detect contradictions while building the mapping. (ii) Model transformations as a directed graph. (iii) Count keystrokes as edges $+$ one extra per cycle. **Pass 3 (insight):** Cycles in the transformation graph are the key difficulty — each requires an extra intermediate step to break; everything else is a simple chain.	389
Reflect	**Trap profile:** String transformation graph — mishandling cycles and global operation scope. $\times$ **Cycle cost off-by-one:** Counting a cycle of length n as $n-1$ operations, missing the extra intermediate step needed to break it. $\times$ **Ignoring global scope:** Applying transformations position-by-position, forgetting that one keystroke affects *every* occurrence of c_1. $\times$ **Missing contradictions:** Not checking whether one input character must map to two different targets at different positions. ✓ Verify contradiction detection: does any source character appear with two different targets? ✓ Trace a small cycle (e.g. A$\to$B$\to$A) manually to confirm the $+1$ intermediate step. **Correct approach:** Build transformation graph, detect conflicts, identify cycles, sum |edges|$+$ |cycles|. Return -1 on contradiction.	620

Figure 13: Example of generated and transformed thinking traces from a coding / optimization example.