ThoughtTrim: Anchor-Driven RL Modification[1]

Andrew Briand[†]
atb8888@comcast.net

Josh Rauvola[†]
jrauvola@gmail.com

With Apart Research
[†]Equal contribution.

Abstract

Large language models often generate long chain-of-thought (CoT) traces in which only a small subset of sentences materially influences the final answer. We propose **ThoughtTrim**: a simple evaluation framework that ranks CoT chunks by counterfactual-importance KL [1], reconstructs prompts using only the top-ranked chunks, and measures the accuracy-retention trade-off as filtering thresholds rise. Using Qwen2.5-1.5B on a 100-question Biology subset of MMLU-Pro, we find that (i) for some questions, KL-guided trimming preserves accuracy at substantial token savings (60-90% on many items), (ii) "first failure" thresholds are heterogeneous - some problems fail immediately while a long tail remains robust up to aggressive pruning, and (iii) a *KL-shuffled* control that preserves the number of kept chunks but breaks informativeness is consistently worse than the original selection, demonstrating the value of the ranking signal. We release a lightweight pipeline that utilizes the counterfactual-importance KL to understand the thresholds, efficiency frontiers, and failure distributions. This opens up future work in creating systems that are more deterministic, efficient, and robust through fine-tuning approaches leading to safer more deterministic approaches in agentic and LLM-based systems.

Keywords: Reasoning optimization, thought anchors, RL fine-tuning, AI security, model evaluations, safety infrastructure

1. Introduction

Reasoning models love to narrate. That's great when the narrative is doing real work; it's wasteful when the model spins its wheels. **ThoughtTrim** starts from the observation in [1] that only a handful of sentences, called *thought anchors* appear to have a large influence on the model's final answer distribution. If we could compress prompts around the "backbone" formed by these thought anchors while retaining accuracy, we could improve inference efficiency and produce shorter, potentially easier to digest CoTs.

Practically, we (i) detect high-influence sentences in chain-of-thought traces, (ii) attempt to formulate an "anchor" CoT which contains only the most important sentences, (iii) re-run inference to measure accuracy.

2. Methods

2.1. Model pair and why this comparison

We use a clean A/B pair at the ∼1.5B scale: **R1-Distill-Qwen-1.5B** (a *reasoning* model distilled to emit `<think>...</think>` traces) versus **Qwen2.5-1.5B** (a *non-reasoning* baseline that answers directly) [3, 4]. Same backbone family and size; different post-training

[1]Research conducted at the CBRN AI Risks Research Sprint, 2025.

behavior. This lets us isolate the causal effect of explicit CoT training. We generate rollouts with **R1-Distill-Qwen-1.5B** and inject our trimmed CoTs into **Qwen2.5-1.5B**, using the same prompts for both.

2.2. Dataset

We evaluate on the Biology subset of **MMLU-Pro** [2]; see Section 2.3 for details on the benchmark and its differences from the original MMLU.

2.3. Benchmark: MMLU-Pro

Overview. MMLU-Pro is an enhanced successor to the original Massive Multitask Language Understanding benchmark, designed to stress reasoning rather than recall. It comprises ∼12k rigorously curated multiple-choice questions drawn from exams and textbooks across 14 domains (Biology, Business, Chemistry, Computer Science, Economics, Engineering, Health, History, Law, Math, Philosophy, Physics, Psychology, and Others). Problems are formatted with ten answer options (A–J), which lowers the random-guess baseline to 10% and makes evaluation more realistic. During this body of work we take a randomized subset of 100 biology question to evaluate the model's ability to reason and answer the questions correctly using our proposed methodology. This is due to resource constraints and time constraints. A more robust evaluation would evaluate the model on the entire dataset.

Evaluation protocol in this work. Following the dataset's 10-choice format, we constrain model outputs to a single letter ∈ {A,. . . ,J} after any optional `</think>` block. Concretely, we guide decoding with the regex `[^<]*</think>\nThe correct answer is [ABCDEFGHIJ]` to ensure parseable, answer-only scoring. We report accuracy as the primary metric, with secondary analysis token count.

2.4. Evaluation pipeline (ThoughtTrim)

Our toolchain runs three steps per problem and threshold: (1) **filtering**—rank chunks by counterfactual-importance KL (min–max normalized per problem to $[0, 1]$); keep chunks with score $\geq$ a threshold t and reconstruct the prompt in the original order; (2) **model runs**—embed the filtered CoT in the original prompt and query Qwen2.5-1.5B via Ollama with answer-only scoring; (3) **aggregation**—compute per-problem accuracy vs. threshold and then average across problems at each threshold.

2.5. Randomized baseline and normalization

To test whether KL-based selection beats non-informative selection, we add a **KL shuffle** control: shuffle the (per-problem normalized) KL scores across chunks, apply the same thresholds, and repeat the evaluation. This preserves how much text is kept in terms of number of sentences but breaks the informativeness of the ranking. Throughout, KL scores are min–max normalized per problem to make thresholds comparable across items.

3. Anchor discovery

3.1. What are thought anchors?

Following Bogdan et al. [1], we use the term *thought anchors* for sentences in a chain-of-thought that exert outsized causal influence on subsequent reasoning and the final answer. In [1], evidence comes from three complementary lenses: (i) a black-box resampling test of counterfactual importance that compares outcome distributions across 100 rollouts conditioned on including a sentence vs. replacing it with a semantically different variant; (ii) a white-box aggregation of attention patterns that reveals "broadcasting" sentences attracting receiver-head attention from many future tokens; and (iii) a causal attention-suppression probe that dampens attention to a sentence and measures downstream effects token-by-token. In this work we focus on (i).

3.2. Setup and metrics for anchor discovery

We represent the model's reasoning trace for question q as tokens $y_{1:T}$ from policy π_θ, segmented into sentences $S_1, \ldots, S_M$ with token index sets $\mathcal{S}_j \subseteq \{1, \ldots, T\}$. The final answer is a categorical variable $Z \in \mathcal{A}$ with $|\mathcal{A}| = 10$ for MMLU-Pro.

Black-box counterfactual importance via rollouts (used). For each sentence S_j, we estimate two answer distributions using rollouts:

- **Present:** condition on the prefix up to S_{j-1} and include the exact sentence S_j; then sample R rollouts for the suffix.
- **Absent:** condition on the prefix up to S_{j-1} only (do not include S_j); then sample R rollouts for the suffix.

For a given rollout r in the **Absent** rollouts, let T_j be the sentence that replaced S_j. If T_j is semantically similar to S_j, defined as the dot product of the sentences' embeddings by `all-MiniLM-L6-v2` being greater than 0.8, then we move r to the **Present** distribution. Let $n_{j,a}^{(c)}$ be the number of times answer $a \in \mathcal{A}$ appears under condition $c \in \{\text{pres}, \text{abs}\}$. With Laplace smoothing $\alpha > 0$,

$$\hat{P}_j^{(c)}(a) = \frac{n_{j,a}^{(c)} + \alpha}{R + \alpha\,|\mathcal{A}|} \quad (c \in \{\text{pres}, \text{abs}\}).$$

In our experiments we use $R = 100$ and $\alpha = 1$. Sentences are ranked by I_j^{KL} and we select the top-K as anchors.

See appendix 7.1 for details about our setup for rollouts.

4. Results

We evaluate on 100 Biology questions from MMLU-Pro using Qwen2.5-1.5B. Unless stated, thresholds sweep $t \in [0, 1]$ with step 0.01 over KL (normalized per problem). Model decoding uses greedy decoding and answer-only scoring to minimize variance.

4.1. Key empirical findings

Across 100 Biology questions:

- **Accuracy vs. threshold.** The original curve decays gradually from ∼95% at $t = 0$ to ∼46% near $t = 1.0$. The KL shuffle baseline is consistently below the original after $t0.05$, indicating that KL ranking retains more task-relevant chunks than chance.
- **First-failure distribution.** Most problems fail between $t \in [0.6, 0.9]$; a long tail remains correct up to $t \approx 1.0$, suggesting some questions are robust to heavy trimming.
- **Token savings.** Substantial token reductions (60–90%) are possible while maintaining ≥80% accuracy for many questions.

4.2. Why Original and KL shuffle converge only at the end

Since the model is scored on reconstructed CoTs, correctness differs even when only few sentences remain in the CoT. Once all chunks have been removed, we see in Figure 1 that accuracy converges.

5. Discussion and Conclusion

KL-guided trimming is a simple, data-driven and black-box way to compress CoT while preserving utility. Relative to a shuffled baseline that keeps the same *amount* of text but discards informativeness, KL-based selection consistently wins—especially in the mid-threshold regime where most practical budgets lie. First-failure and efficiency-frontier views reveal heterogeneous robustness across items, arguing for per-domain or per-problem thresholds rather than a single global value. For future work, we would aim to choose thresholds on the frontier that meet accuracy requirements (e.g., ≥80%) while minimizing retention.

Future work includes (i) multi-sample rather than greedy evaluation (ii) anchor-aware fine-tuning so models learn to follow high-impact steps without explicit filtering, and (iii) domain-adapted safety checks for CBRN-adjacent applications.

5.1. Policy objective for anchor-following RL (Future Work)

Let π_θ generate tokens $y_{1:T}$ and let $\mathcal{P}$ denote a compact *anchor plan* extracted from the selected anchors. We shape a reward

$$R = R_{\text{task}} + \lambda_{\text{plan}} R_{\text{plan}} - \mu_{\text{fluff}} R_{\text{fluff}} - \beta T,$$

where $R_{\text{task}} = 1[Z = a^*]$, R_{plan} rewards adherence to plan checkpoints (e.g., emitting required subgoal statements or checks), R_{fluff} penalizes off-plan narration, and T is token count. We optimize $J(\theta) = E[R]$ with PPO, constraining KL to a reference policy. In practice we implement $R_{\text{plan}}/R_{\text{fluff}}$ via simple regex/critic signals keyed to anchor semantics.

6. References

References

[1] P. C. Bogdan, U. Macar, N. Nanda, and A. Conmy. Thought Anchors: Which LLM Reasoning Steps Matter? 2025.

[2] Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, *et al.* "MMLU-Pro: A more robust and challenging multi-task language understanding benchmark," *arXiv preprint arXiv:2406.01574*, 2024.

[3] Qwen Team. Qwen2.5: A Party of Foundation Models. September 2024. `https://qwenlm.github.io/blog/qwen2.5/`.

[4] DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL], 2025. `https://arxiv.org/abs/2501.12948`.

7. Appendix

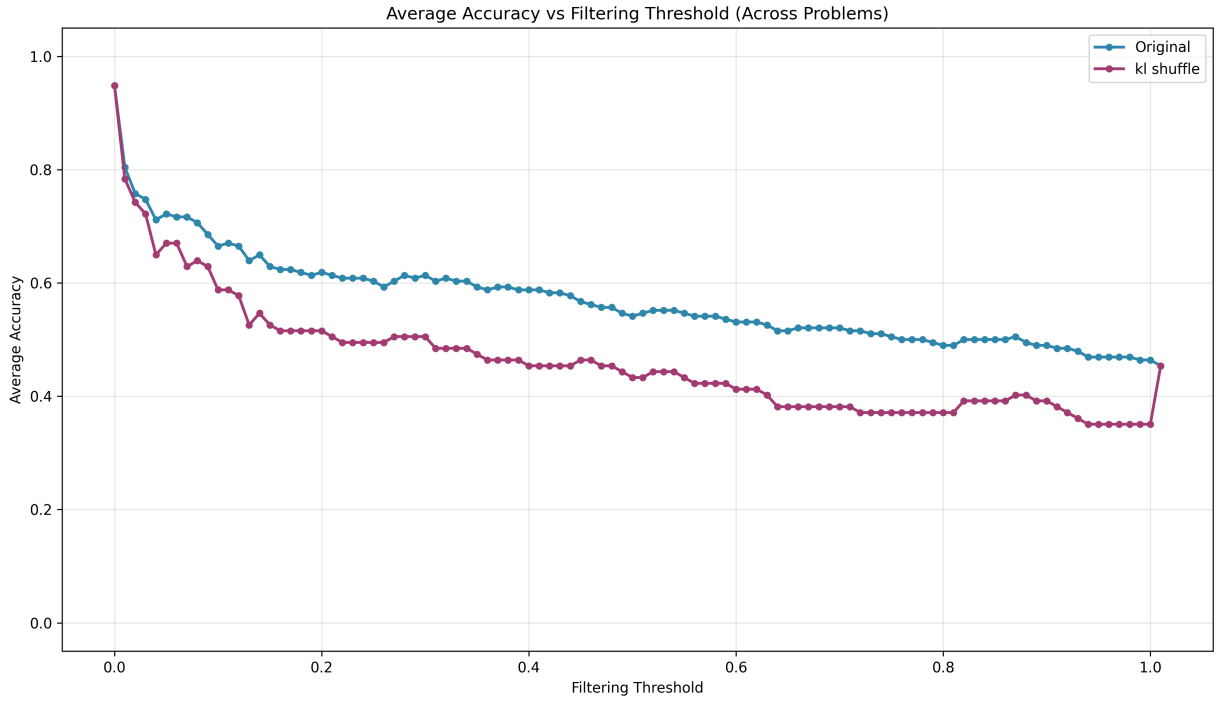


*Figure 1: Average Accuracy vs. Filtering Threshold across 100 Biology problems. The **Original** (blue) uses KL-ranked chunk selection; the **KL shuffle** (magenta) randomly permutes KL per problem before thresholding, preserving keep-count but breaking informativeness. KL-based selection consistently outperforms the shuffled control.*

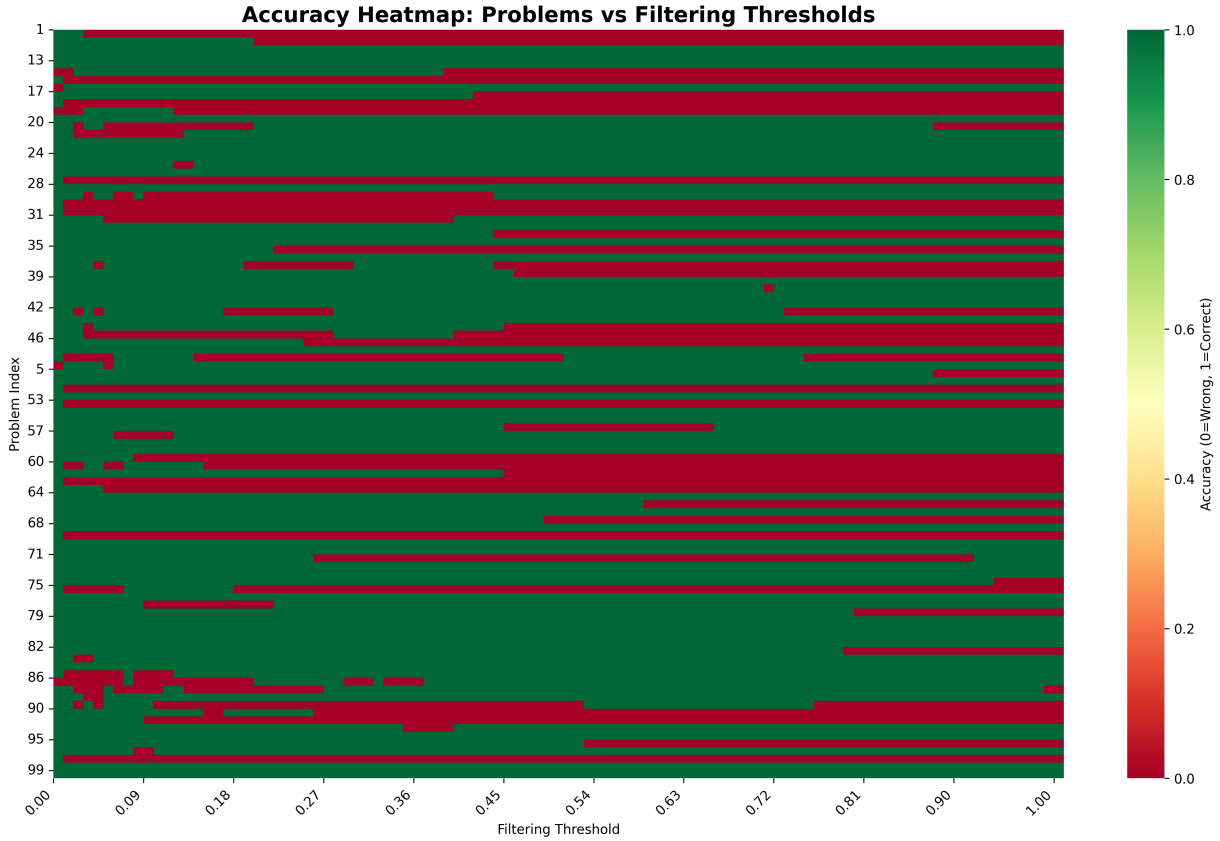


Figure 2: Accuracy heatmap (problems × thresholds). Green indicates correctness and red indicates failure. Columns correspond to KL thresholds (normalized per problem). Many items are robust (green through $t \approx 0.6$–0.9), while fragile items fail almost immediately (red emerging near $t \approx 0.01$–0.03). This view is useful for diagnosing which questions drive the tails and for identifying domain-specific cohorts.

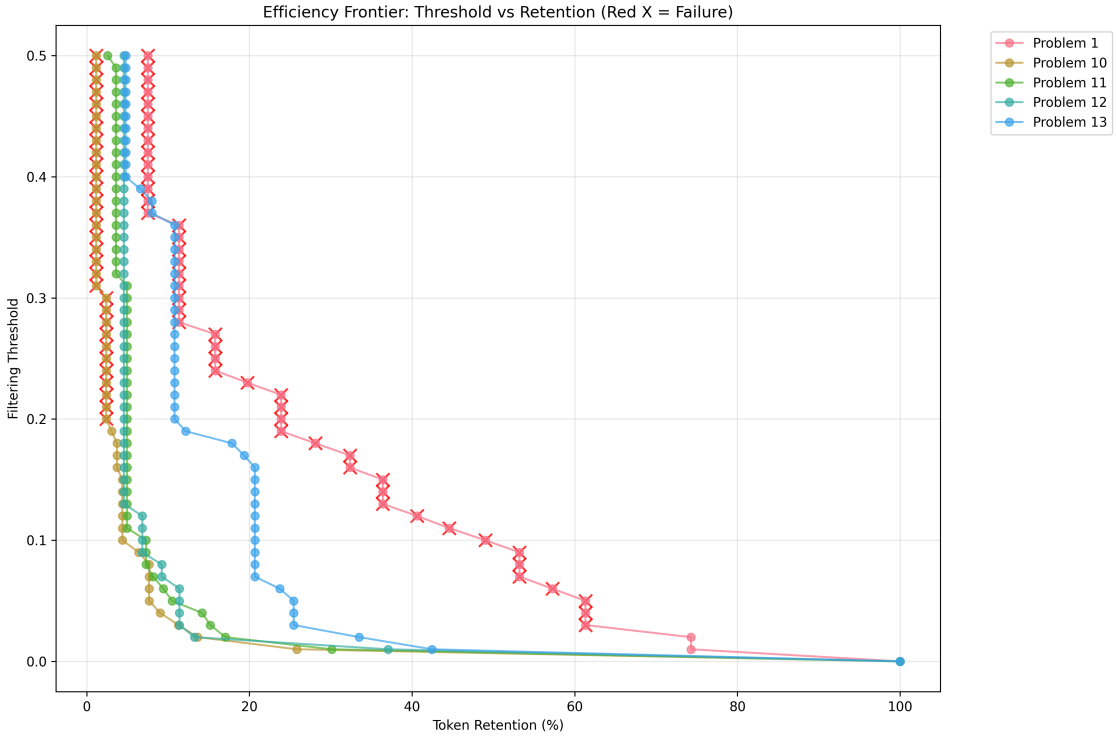
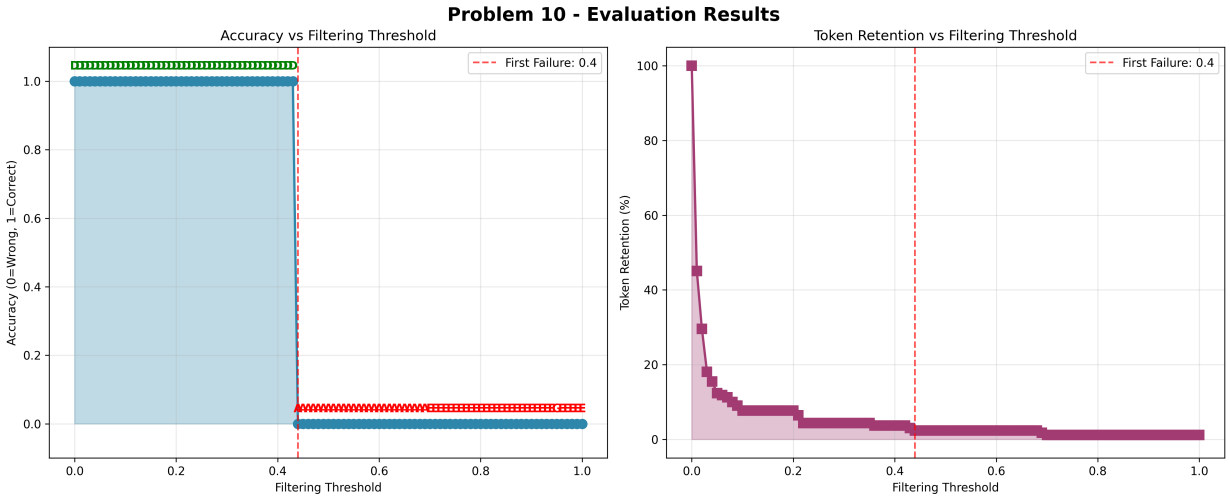
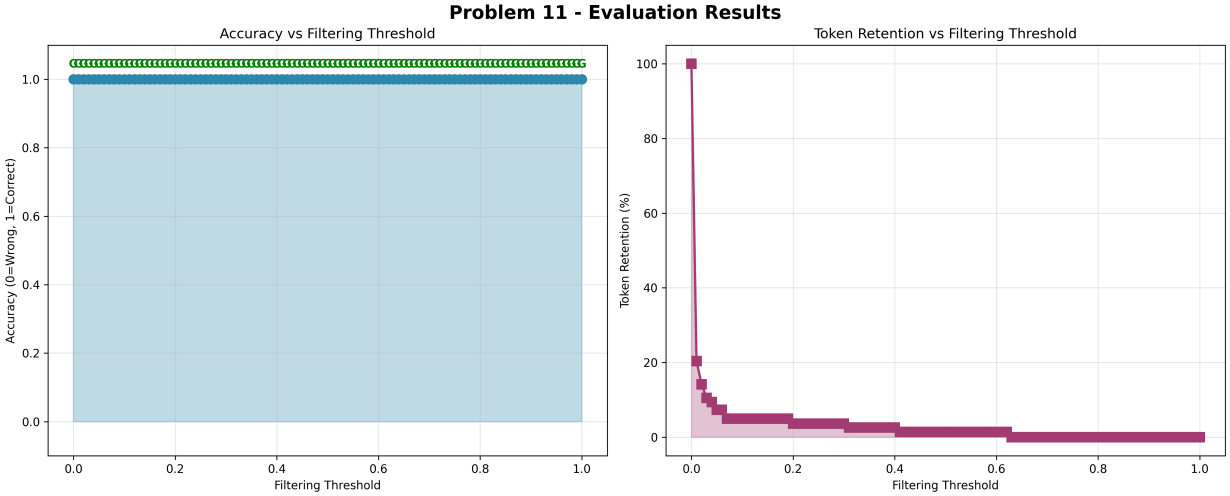


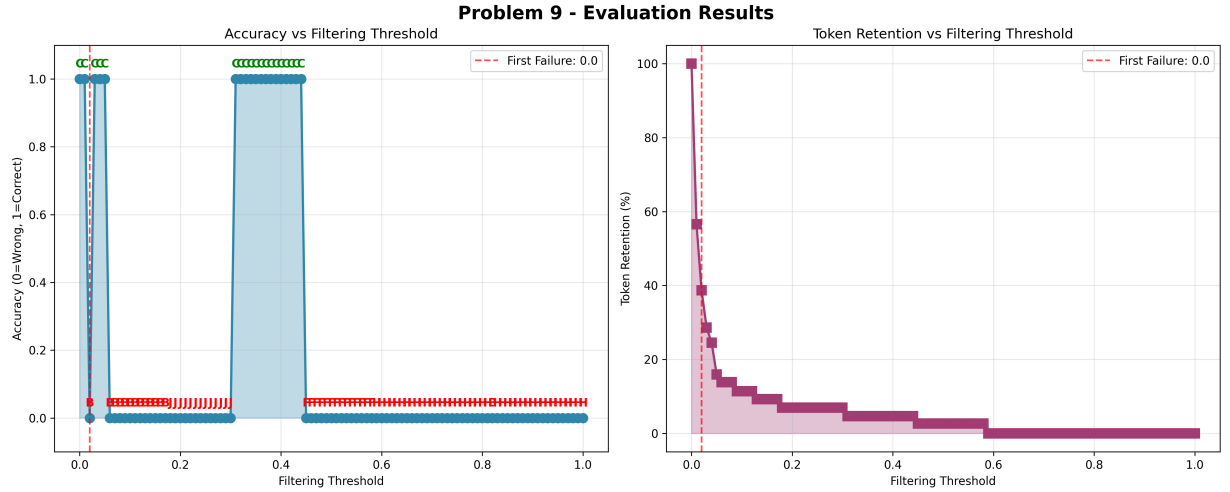
Figure 3: Efficiency frontier for representative items: threshold vs. token retention with failure points (red X). Robust items allow large reductions; fragile ones fail early. This would motivate per-item thresholding in an RL setting to obtain the efficiency gains of thought-trimming without destroying model performance.



(a) Problem 10. Left: accuracy vs. threshold with first-failure marker; Right: token retention vs. threshold (area under curve). This item is robust until $t \approx 0.4$, then fails sharply.



(b) Problem 11. Accuracy remains ≈1.0 across the sweep; retention falls quickly—a highly compressible case.



(c) Problem 9. Early failure followed by a secondary robust plateau: illustrates non-monotone item behaviour that motivates per-problem thresholding.

Figure 4: Representative per-problem behaviour. These panels complement Fig. 2 by showing the mechanics behind robust (11), late-failure (10), and fragile/non-monotone (9) cases.

Security Considerations

Our study focuses on the mechanics of trimming chain-of-thought for efficiency; nevertheless, we consider security and generalization risks that matter for deployment, especially in CBRN-adjacent settings.

This study should be interpreted within a weekend constrained scope. We evaluate a *small* 1.5B Qwen model on a *100-question* Biology slice of MMLU-Pro; robustness could shift for larger models, other architectures, or different domains. Our setup is *multiple-choice*; applying ThoughtTrim to free-form tasks will require re-running the analysis with open-ended prompts, reliable answer extraction, and explicit safety content checks. We do not fine-tune in this work. If future iterations add anchor-following or compression fine-tuning, safety should be part of both the objective and the evaluation (e.g., guardrail/refusal rewards and gates) so that efficiency gains do not introduce unsafe shortcuts or overconfident errors. Trimming can also remove policy or safety statements; in high-stakes settings one should either exclude such spans from pruning or monitor truthfulness, refusal correctness, and calibration alongside accuracy. Finally, to reduce overfitting of conclusions, subsequent iterations should broaden to multiple model sizes and architectures, extend beyond Biology to additional domains, and incorporate multi-seed sampling to quantify variance.

7.1. Rollout parameters

Following the methodology of [1], we regenerate a CoT and answer for each question until the model answers the question correctly. If the model generates an incorrect answer 100 times in a row, we give up and skip the question. This only happened once in our 100 questions. We then split up the CoT that led to the correct answer into chunks roughly corresponding to sentences. For each chunk in the CoT, we create a prompt that includes the CoT up to but not including that chunk, and then generate 100 CoTs (rollouts) from that point, recording the answer that the model produced at the end of each rollout.

For both initial question generation and rollouts, we use a temperature of 0.6, top-p of 0.95, and a repetition penalty of 1.1. Additionally, we limited output tokens to 4096 to ensure that the rollouts completed in a reasonable amount of time. Since we found that the small models we tested had difficulty consistently boxing their answers, we constrained generation using the regular expression `[^<]*</think>\nThe correct answer is [ABCDEFGHIJ]` to ensure easily parsable output.

[1]:

$$I_j^{\mathrm{KL}} = \mathrm{KL}\Big(\hat{P}_j^{\mathrm{pres}} \Big\| \hat{P}_j^{\mathrm{abs}}\Big) = \sum_{a \in \mathcal{A}} \hat{P}_j^{\mathrm{pres}}(a)\, \log \frac{\hat{P}_j^{\mathrm{pres}}(a)}{\hat{P}_j^{\mathrm{abs}}(a)}.$$

We also report the accuracy lift for the gold answer a^*:

$$\Delta_j = \hat{P}_j^{\mathrm{pres}}(a^*) \; - \; \hat{P}_j^{\mathrm{abs}}(a^*).$$