BEYOND UNCERTAINTY: TURNING MODEL FAILURES INTO REUSABLE LOGIC TEMPLATES FOR ADAPTIVE REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Exemplar-based chain-of-thought (CoT) prompting can effectively improve multi-step reasoning in large language models (LLMs), but its effectiveness depends on the quality and relevance of the demonstrations for each query. Existing approaches face three limitations: demonstration selection fails to identify genuinely difficult problems, generated reasoning paths remain noisy or redundant, and inference-time retrieval often relies on surface-level similarity rather than the underlying solution structure. To address these limitations, we propose Reusable Logic Template Activation (**ReLTA**), a framework for constructing and dynamically selecting reasoning demonstrations. ReLTA combines prediction errors with answer-space entropy to identify difficult examples, reconstructs and simplifies their reasoning traces into concise logic templates, and retrieves reasoning-relevant templates for each test query. Experiments on eight reasoning benchmarks using two reasoning backbones show that ReLTA achieves the highest accuracy on seven of eight datasets for each backbone, with an average improvement of **2.42%** over Active-Prompt under the DeepSeek-Coder-V2-Lite-Base backbone. ReLTA provides a modular framework for constructing and selecting reasoning demonstrations.

1 INTRODUCTION

While large language models (LLMs) excel at general text processing, they consistently struggle with tasks that demand rigorous, multi-step logical deduction (Matarazzo & Torlone, 2025), where intermediate reasoning errors easily propagate across extended deduction chains. Chain-of-thought (CoT) prompting provides an effective approach to elicit and organize such reasoning by augmenting input demonstrations with intermediate steps (Wei et al., 2022b), providing explicit reasoning paths that guide the model through complex deductions. However, the practical construction and application of effective CoT demonstrations face three primary problems:

First, conventional demonstration selection struggles to reliably identify genuinely difficult questions. Existing approaches often rely on coverage- or structure-oriented selection to obtain informative and diverse demonstrations (Gupta et al., 2023; Ye et al., 2023), while sample-efficient methods aim to identify useful exemplars under a limited evaluation budget (Purohit et al., 2025). Other approaches use active sampling to prioritize queries that induce high prediction disagreement (Kuhn et al., 2023; Diao et al., 2024; Che et al., 2026). However, equating difficulty solely with high variance overlooks persistent errors, where a model consistently converges on an identical incorrect answer with low entropy. Demonstration selection should expose genuinely informative reasoning bottlenecks. To achieve this, we introduce a hybrid selection metric that combines empirical prediction error with answer-space entropy, ensuring that both high-variance instability and low-entropy persistent errors are identified.

Second, automatically generated reasoning traces harvested for these demonstrations may contain exploratory trial-and-error, redundant steps, or other forms of verbosity. Existing generation and self-improvement pipelines provide useful reasoning supervision, but do not necessarily remove every non-contributory step from the resulting traces (Zelikman et al., 2022; Huang et al., 2023; Zhang et al., 2023). To prevent biasing the model toward such inefficiencies, raw traces should be distilled

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

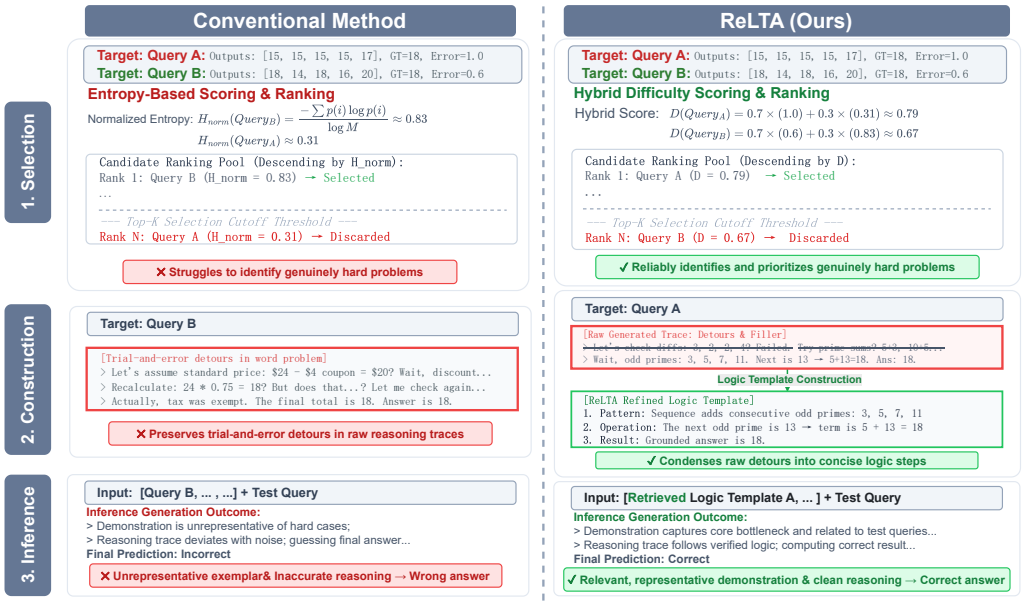

Figure 1: Comparison between a conventional automatic prompting pipeline and ReLTA across three stages. **Selection:** entropy-based scoring can miss low-entropy persistent failures, whereas the error–entropy hybrid score retains them. **Construction:** conventional pipelines preserve trial-and-error detours in raw traces, while the Logic Template Construction stage condenses answer-consistent reasoning into concise logic templates. **Inference:** conventional prompting propagates suboptimal demonstrations and reasoning noise into the context, whereas ReLTA selects clean, structurally relevant templates for the target query.

into correct and concise logical steps. We address this through an offline Logic Template Construction process that reconstructs failed solutions and simplifies raw traces into concise, structured logic templates by removing unnecessary detours.

Third, demonstration retrieval in the inference phase typically relies on surface-level textual similarity rather than the underlying solution structures. Conventional pipelines commonly use dense semantic or learned retrievers, which may pair test queries with source problems that share topical or narrative features but involve different mathematical or logical operations (Karpukhin et al., 2020; Izacard et al., 2022; Liu et al., 2022; Rubin et al., 2022; Li et al., 2023). To avoid negative transfer, instance retrieval should align demonstrations based on deductive structural compatibility. To address this, we use a reasoning-aware retriever to select structurally compatible logic templates for each test query, and order them by structural relevance to the target question (Das et al., 2025; Saad-Falcon et al., 2025).

To address these challenges, we propose Reusable Logic Template Activation (ReLTA). Its key innovation is to transform model-specific failures into concise, reusable logic templates and select them at inference phase according to their structural compatibility with the target query. Figure 1 illustrates how ReLTA differs from conventional automatic prompting pipelines. Our key contributions are summarized as follows:

- **Error–Entropy Hybrid Selection:** We introduce a dual-signal difficulty criterion that combines empirical prediction error with answer-space entropy, capturing both high-entropy prediction instability and low-entropy persistent errors.
- **Logic Template Construction:** We develop an offline Generation–Reconstruction–Simplification procedure that converts selected model failures into concise, instance-grounded logic templates, with final-answer consistency checks and fallback mechanisms controlling template acceptance.
- **Modular and Extensible Framework:** We formulate ReLTA as a modular framework that separates failure identification, logic-template construction, and inference-time demonstra-

tion selection. This design allows each component to be independently replaced or extended, providing a flexible foundation for incorporating alternative difficulty measures, trace-refinement strategies, and retrieval methods in other reasoning settings.

2 Related Work

2.1 Chain-of-Thought Reasoning

Chain-of-thought (CoT) prompting improves multi-step reasoning by augmenting in-context demonstrations with intermediate rationales (Wei et al., 2022b). Subsequent work reduces the reliance on manually written rationales by generating and clustering candidate chains (Zhang et al., 2023), bootstrapping successful reasoning through self-training (Zelikman et al., 2022; Huang et al., 2023), or learning from complex explanation traces through parameter updates (Mukherjee et al., 2023). These studies establish model-generated reasoning traces as useful supervision, but generally construct supervision from representative questions or successful solutions rather than systematically converting model-specific failures into verified, reusable demonstrations.

Beyond generating successful chains, another line of work studies which reasoning structures should be preserved and reused. Complexity-based prompting favors demonstrations with richer reasoning steps, showing that necessary reasoning complexity can be beneficial (Fu et al., 2023). At a higher level of abstraction, Buffer of Thoughts extracts recurring solution patterns as reusable thought templates (Yang et al., 2024). Together, these approaches highlight a distinction between useful logical complexity and incidental trace verbosity. ReLTA does not favor shorter chains simply for the sake of brevity; instead, it reconstructs model-specific failures and accepts a simplified trace only when the final-answer consistency check confirms that the required deduction has been preserved. The resulting templates remove non-contributory detours while remaining grounded in their source instances rather than being reduced to fully abstract problem-solving schemas.

2.2 In-Context Demonstration Selection

The effectiveness of in-context learning depends strongly on which examples are included as demonstrations. Pool-level selection methods promote diversity, coverage, or compositional structure among selected demonstrations (Gupta et al., 2023; Purohit et al., 2025; Ye et al., 2023). Uncertainty estimation instead quantifies disagreement across sampled outputs, including disagreement after grouping semantically equivalent answers (Kuhn et al., 2023). Building on such signals, Active-Prompt selects task-specific questions with high predictive uncertainty for rationale annotation, and related methods use uncertainty to guide prompt selection (Diao et al., 2024; Che et al., 2026). However, output dispersion does not indicate whether a concentrated prediction is correct. In ReLTA's answer-labeled candidate pool, correctness provides an observable and complementary signal of model failure; ReLTA therefore combines empirical prediction error with answer-space entropy to capture both unstable predictions and persistent low-entropy errors.

A complementary line of work selects demonstrations conditionally for each test query. Early approaches retrieve semantically similar demonstrations using nearest-neighbor representations (Liu et al., 2022), whereas learned retrievers optimize demonstration utility using language-model feedback (Rubin et al., 2022; Li et al., 2023). More recent reasoning-oriented retrievers further target structural relevance beyond generic semantic similarity (Das et al., 2025; Saad-Falcon et al., 2025). These methods primarily address how to select from an existing demonstration pool. ReLTA additionally determines how that pool is constructed: the error–entropy criterion identifies model-specific failures, and Logic Template Construction converts them into reusable logic templates. Unlike task-level active selection methods that construct a shared exemplar set, ReLTA separates template construction from query-specific retrieval and allows the retrieval interface to be replaced or extended for different reasoning settings.

3 Method

The goal of ReLTA is to provide a general framework for constructing and selecting query-adaptive in-context demonstrations for multi-step reasoning. Let $\mathcal{D}_{\text{cand}} = \{(q_i, y_i^*)\}_{i=1}^{N}$ denote a labeled can-

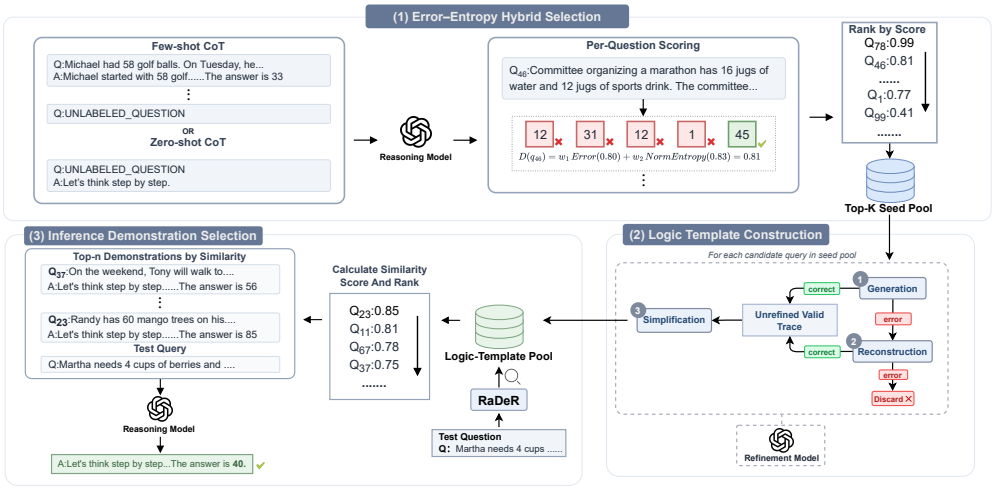

Figure 2: Overview of ReLTA, which coordinates offline construction and dynamic inference across three stages: (1) Error–Entropy Hybrid Selection identifies hard seeds capturing both prediction variance and persistent errors; (2) Logic Template Construction refines and simplifies raw traces into concise, answer-consistent logic templates; and (3) Inference Demonstration Selection retrieves structurally relevant templates at test time using a reasoning-aware retriever interface.

didate pool, where q_i is a candidate problem and y_i^* is its reference answer. Given an arbitrary test query q_{test}, ReLTA constructs demonstrations that are structurally compatible with the query while reducing unnecessary exploratory reasoning. The framework separates offline construction from final inference and coordinates three replaceable components: a target reasoning backbone $\mathcal{M}_{\text{R}}$, an offline refinement model $\mathcal{M}_{\text{Re}}$, and a reasoning-aware retriever interface $\mathcal{E}$. During offline construction, ReLTA identifies informative failures in $\mathcal{D}_{\text{cand}}$ and converts them into answer-consistent logic templates. During final inference, Inference Demonstration Selection uses $\mathcal{E}$ to retrieve and select relevant source demonstrations for each test query. The reference answer of the test query is never provided at final inference; the reference answer associated with a retrieved source problem is included in that demonstration together with the source problem and its refined reasoning trace. The following sections describe these components in detail.

3.1 Error–Entropy Hybrid Selection

The goal of this offline stage is to identify candidate problems that expose either unstable predictions or persistent model errors. For each q_i, the target reasoning backbone independently samples M reasoning traces under a sampling temperature T:

$$\tau_{i,m} \sim \mathcal{M}_{\text{R}}(\cdot \mid q_i; T), \qquad m = 1, \ldots, M. \tag{1}$$

We extract and canonicalize the final answer from each trace as $\hat{y}_{i,m} = \text{Parse}(\tau_{i,m})$. If no valid answer can be extracted, we assign the discrete outcome `NO_SOLUTION`. The same canonicalized outcomes are used for accuracy and entropy computation, so semantically equivalent answers are counted as one outcome even when their raw reasoning traces differ substantially. The special outcome `NO_SOLUTION` is treated as an invalid prediction for correctness and remains an outcome when estimating answer uncertainty.

The first signal is empirical answer accuracy:

$$\mathcal{A}(q_i) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{I}\left[\text{Equiv}(\hat{y}_{i,m}, y_i^*)\right], \tag{2}$$

where $\mathbb{I}[\cdot]$ is the indicator function and $\text{Equiv}(\cdot, \cdot)$ is a dataset-specific final-answer equivalence interface. It returns one when the parsed prediction and the reference answer are equivalent and zero otherwise; in particular, `NO_SOLUTION` is not equivalent to a valid reference answer. The corresponding empirical prediction error is $1 - \mathcal{A}(q_i)$.

The second signal is answer-space entropy. Let $\mathcal{Y}_i = \mathrm{Unique}(\{\hat{y}_{i,m}\}_{m=1}^{M})$ be the observed outcome support and define its empirical probability by

$$\hat{p}_i(y) = \frac{1}{M} \sum_{m=1}^{M} \mathbb{I}[\hat{y}_{i,m} = y]. \tag{3}$$

We compute

$$H(q_i) = -\sum_{y \in \mathcal{Y}_i} \hat{p}_i(y) \log \hat{p}_i(y), \tag{4}$$

and normalize it as

$$H_{\mathrm{norm}}(q_i) = \frac{H(q_i)}{\log M}, \qquad M > 1. \tag{5}$$

Because at most M distinct outcomes can be observed from M trials, $\log M$ upper-bounds the empirical entropy and yields $H_{\mathrm{norm}}(q_i) \in [0, 1]$.

The error term measures how often the backbone answers incorrectly, whereas the entropy term measures how dispersed its parsed outcomes are. We combine these two normalized signals into a hybrid difficulty score:

$$D(q_i) = w_1 \left(1 - \mathcal{A}(q_i)\right) + w_2 H_{\mathrm{norm}}(q_i), \qquad w_1, w_2 \geq 0, \quad w_1 + w_2 = 1. \tag{6}$$

The score captures both low-entropy persistent errors and high-entropy prediction disagreement. Let $\mathcal{D}_+ = \{(q_i, y_i^*) \in \mathcal{D}_{\mathrm{cand}} : D(q_i) > 0\}$ denote the eligible candidates. Using a construction budget of K, we retain the eligible candidates with the largest scores:

$$\mathcal{S} = \mathrm{Select}_K(\mathcal{D}_+; D), \qquad |\mathcal{S}| = \min(K, |\mathcal{D}_+|). \tag{7}$$

Here, $\mathrm{Select}_K(\cdot; D)$ returns the top-K candidates ranked by $D(q_i)$, or all eligible candidates when fewer than K have positive scores; it denotes a deterministic ranking operation rather than a separate model component. A zero score corresponds to uniformly correct and deterministic predictions under the sampled trials, which do not reveal an observed failure mode for subsequent refinement.

3.2 Logic Template Construction

The active seed pool contains selected problems, but their raw reasoning traces may still be noisy or unsuitable for use as demonstrations. We therefore use a separate refinement model $\mathcal{M}_{\mathrm{Re}}$ only during offline construction to generate, reconstruct, and simplify reasoning traces. For each $(q, y^*) \in \mathcal{S}$, ReLTA applies three stages: *Generation*, *Reconstruction*, and *Simplification*. Generation first attempts to solve the problem directly; Reconstruction is invoked only when Generation fails; and Simplification condenses the retained trace into a reusable logic template. We use *logic template* to denote an instance-grounded reasoning structure that preserves the entities, values, constraints, principal operations, and logical transitions needed to solve the source problem while removing exploratory branches, repetition, and conversational filler. It is not a fully decontextualized symbolic program. Each refinement call returns a structured pair $(\tau, \hat{y})$ containing a reasoning trace and its parsed final answer; an unparseable output fails the check. We refer to this operation as the *Final-Answer Consistency Check*; formally, all acceptance decisions use

$$\mathrm{FinalAnswerCheck}(\hat{y}, y^*) = \mathbb{I}[\mathrm{Equiv}(\hat{y}, y^*)]\,. \tag{8}$$

This check verifies final-answer consistency only; it does not certify the validity or faithfulness of every intermediate step. The complete procedure is provided in Appendix A.1 and Algorithm 1.

Step 1: Generation. The refinement model first attempts the problem directly under structured-output constraints and without access to the reference answer, producing $(\tau^{(1)}, \hat{y}^{(1)})$. If $\mathrm{FinalAnswerCheck}(\hat{y}^{(1)}, y^*)$ passes, we set $\tau_{\mathrm{curr}} \leftarrow \tau^{(1)}$ and advance to Simplification.

Step 2: Reconstruction. If Generation fails the final-answer consistency check, we provide the reference answer y^* to the refinement model and instruct it to derive a reasoning trace that supports this answer. The model returns $(\tau^{(2)}, \hat{y}^{(2)})$, with the output protocol requiring $\hat{y}^{(2)}$ to be parsed from the reconstructed trace rather than inserted as a fixed answer field. Passing $\mathrm{FinalAnswerCheck}(\hat{y}^{(2)}, y^*)$ yields an answer-consistent candidate trace but does not establish

step-level faithfulness. We then set $\tau_{\text{curr}} \leftarrow \tau^{(2)}$; if this check fails, the candidate is discarded and does not proceed to Simplification.

Step 3: Simplification. A retained trace may still contain exploratory detours, repetition, or conversational filler. The refinement model therefore receives the question and τ_{curr}, but not y^*, and returns $(\tau^{(3)}, \hat{y}^{(3)})$. The simplified trace $\tau^{(3)}$ is accepted as the logic template only when $\mathrm{FinalAnswerCheck}(\hat{y}^{(3)}, y^*)$ passes; otherwise, ReLTA retains the answer-consistent trace τ_{curr}.

The resulting pool contains $K' \leq K$ logic templates because candidates that fail both Generation and Reconstruction are discarded. If Simplification fails the final-answer consistency check, ReLTA falls back to the pre-Simplification trace τ_{curr}, preventing a failed simplification attempt from replacing an already answer-consistent trace.

3.3 Inference Demonstration Selection

At inference time, ReLTA selects query-specific demonstrations from the constructed logic-template pool rather than using a fixed prompt. Each stored item is a tuple (q_j, y_j^*, τ_j), where q_j is the source problem used as the retrieval key, y_j^* is its reference answer, and τ_j is the refined logic template. For each selected item, the source problem, its refined reasoning trace, and its source answer are concatenated to form an in-context demonstration; the answer to the test query is never included. For an evaluation query q_{test}, we instantiate the retriever interface with a pre-trained reasoning-aware dense retriever, denoted by $\mathcal{E}$ (RaDeR in our experiments). The interface can be replaced by alternative retrievers without changing the rest of the framework. Its relevance score is

$$s_{\mathcal{E}}(q_{\text{test}}, q_j) = \cos(\mathcal{E}_Q(q_{\text{test}}), \mathcal{E}_P(q_j)), \quad \forall (q_j, y_j^*, \tau_j) \in \mathcal{S}_{\text{refined}}, \tag{9}$$

where $\mathcal{E}_Q$ and $\mathcal{E}_P$ denote the query and passage interfaces of the same fixed retriever. We retain the indices whose scores exceed the selection threshold θ:

$$\mathcal{C}_\theta(q_{\text{test}}) = \{j : s_{\mathcal{E}}(q_{\text{test}}, q_j) \geq \theta\}, \tag{10}$$

We first rank the threshold-qualified candidates and select at most n templates with the highest retrieval scores. If fewer than n templates pass the threshold, all eligible templates are used. We concatenate the selected demonstrations in ascending order of retrieval score, placing the highest-scoring template closest to the test query. If $\mathcal{C}_\theta(q_{\text{test}}) = \emptyset$, inference reduces to zero-shot CoT. The reasoning model then samples N_{sc} outputs, canonicalizes their parsed final answers, and returns the most frequent valid answer:

$$\hat{y}_{\text{test}} = \arg\max_y \sum_{r \in \mathcal{V}} \mathbb{I}[\mathrm{Parse}(r) = y], \tag{11}$$

where $\mathcal{V}$ contains outputs with valid canonicalized answers. Invalid outputs do not participate in voting; if $\mathcal{V} = \emptyset$, the prediction is `NO_SOLUTION`. If multiple answers receive the same maximum vote count, ReLTA selects the answer that appears first among the sampled outputs.

4 Experiments

4.1 Experimental Setup

Datasets and Metrics We evaluate ReLTA on eight benchmarks: GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), ASDiv (Miao et al., 2020), SingleEq (Koncel-Kedziorski et al., 2016), and AQuA (Ling et al., 2017) for arithmetic reasoning; CSQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021) for commonsense and logical reasoning; and Last Letters (Wei et al., 2022a) for symbolic reasoning. We follow the dataset and evaluation protocols adopted by Active-Prompt (Diao et al., 2024). Candidate-pool construction and final evaluation use separate data files: GSM8K, AQuA, CSQA, StrategyQA, and Last Letters use their released training files for candidate construction and their test, development, or task-format evaluation files for final evaluation. For ASDiv, SVAMP, and SingleEq, we use the GSM8K training split as the exemplar candidate pool under the cross-domain transfer setting. Thus, no final-evaluation file is used for difficulty estimation or logic-template construction. For StrategyQA, final evaluation follows the task-format evaluation file used in the public Active-Prompt implementation; the same data configuration is applied to all compared methods. Final-answer accuracy is the primary metric.

Baselines We compare ReLTA with five representative prompting and in-context reasoning baselines: Zero-Shot Chain-of-Thought (Zero-Shot-CoT) (Kojima et al., 2022), Standard Few-Shot CoT, Random CoT (Wei et al., 2022b), cluster-based diversity sampling (Auto-CoT) (Zhang et al., 2023), and uncertainty-based active sampling (Active-Prompt) (Diao et al., 2024). All baseline results are obtained from our reproductions under the corresponding reasoning backbone rather than copied from previously reported result tables. Standard Few-Shot and Active-Prompt use their corresponding fixed exemplar sets, Random CoT uses randomly selected exemplars, and Auto-CoT uses cluster-based automatically generated exemplars. Because the Last Letters questions share essentially the same task structure, clustering does not provide a meaningful diversity signal for this dataset; therefore, Auto-CoT is not applicable and is marked as unavailable in Table 1. All methods use the same reasoning backbone and final-inference sampling budget within each model block.

Our primary reasoning model is Qwen2.5-Coder-7B, with DeepSeek-Coder-V2-Lite-Base providing a cross-backbone evaluation. For each backbone, ReLTA reruns hybrid difficulty scoring, seed selection, and logic-template construction to obtain a backbone-specific template pool, while fixed-prompt baselines use their corresponding reproduced exemplar sets. The final-inference decoding protocol is held fixed within each backbone block.

Implementation Details Within ReLTA, Qwen3-235B-A22B-Instruct serves as the offline refinement model for Generation, Reconstruction, and Simplification, and it is not queried during final evaluation. Error–Entropy Hybrid Selection and Logic Template Construction construct a fixed, backbone-specific logic-template pool before final inference. ReLTA therefore incurs additional model calls during offline template construction, while the final-inference model and decoding budget remain controlled across methods. During final inference, each method samples ten reasoning paths for each evaluation question at temperature 0.7, and the most frequent parsed answer is selected as the final prediction. We report final-answer exact-match accuracy as the evaluation metric. All compared methods use the same final-inference sampling budget, answer-parsing procedure, and evaluation protocol within each backbone block. In our experiments, we instantiate the retriever interface with pre-trained RaDeR (Das et al., 2025) and set the cosine-similarity threshold to 0.35 for all datasets and backbones. If fewer than the requested number of templates pass the threshold, all eligible templates are used; if none pass, inference reduces to zero-shot CoT. Selected demonstrations are arranged in ascending retrieval-score order so that the highest-scoring template appears closest to the test query.

For difficulty estimation, we follow the subsampling strategy of Active-Prompt (Diao et al., 2024). We randomly sample $1{,}000$ instances from the candidate pool, or use the entire pool when it contains fewer instances, and generate $M = 10$ reasoning outputs per candidate at temperature 0.7. We retain up to $K = 100$ candidates according to the hybrid difficulty score. Generation and Simplification use temperature 0.0, whereas Reconstruction uses temperature 0.2; all three refinement calls use structured outputs containing a trace and a final-answer field. The trace-simplification prompt instructs the refinement model to produce at most 200 words. We use 8 in-context exemplars for GSM8K, SVAMP, ASDiv, and SingleEq; 7 for CSQA; 6 for StrategyQA; and 4 for AQuA and Last Letters. The hybrid difficulty weights are fixed to $w_1 = 0.7$ and $w_2 = 0.3$ across all experiments. To compute $\mathrm{Equiv}(\cdot)$, dataset-specific regular expressions extract final numerical, option-label, Boolean, or symbolic answers, followed by normalized matching appropriate to each benchmark.

4.2 Main Results

Table 1 shows that ReLTA achieves the best reported accuracy on seven of eight datasets under each backbone. The reported results are averaged over three runs. The comparison is made among methods with available results; Auto-CoT is not applicable to Last Letters because its cluster-based selection assumes meaningful structural or semantic diversity, which is absent when the questions share essentially the same task type. With Qwen2.5-Coder-7B, the largest gain over Active-Prompt is 4.75 percentage points on AQuA, with improvements also observed across arithmetic, commonsense, and symbolic reasoning tasks. With DeepSeek-Coder-V2-Lite-Base, ReLTA remains best on seven datasets, while Active-Prompt is slightly ahead on CSQA.

All methods share the same reasoning backbone and final-inference sampling budget within each model block, while ReLTA's refinement model is used only during offline logic-template construc-

Table 1: Overall performance of ReLTA and reproduced baselines on eight reasoning datasets under two backbone models. CoT denotes chain-of-thought prompting, and all methods use the same final-inference sampling budget within each backbone block. Auto-CoT is marked as unavailable on Last Letters because the dataset contains essentially one shared task structure, for which cluster-based diversity sampling is not applicable. The best result in each column is shown in bold, and the second-best result is underlined.

Method	GSM8K[a]	ASDiv[b]	SVAMP[c]	AQuA[d]	SingleEq[e]	CSQA[f]	StrategyQA[g]	Last Letters[h]
Qwen2.5-Coder-7B								
Zero-Shot-CoT	86.43	83.59	79.90	68.90	82.48	76.25	68.31	48.50
Standard Few-Shot	88.78	87.88	84.90	69.29	89.96	79.61	73.57	61.10
Random CoT	89.01	92.32	90.80	<u>74.02</u>	91.93	79.52	73.54	82.50
Auto-CoT	89.37	91.48	88.90	71.26	92.91	78.87	74.63	–
Active-Prompt	<u>90.06</u>	**94.58**	<u>92.20</u>	73.20	<u>94.49</u>	<u>81.49</u>	<u>76.16</u>	<u>83.90</u>
ReLTA (ours)	**91.58**	<u>93.89</u>	**93.50**	**77.95**	**95.28**	**82.64**	**79.69**	**85.60**
DeepSeek-Coder-V2-Lite-Base								
Zero-Shot-CoT	68.61	83.97	77.00	41.73	89.96	40.38	55.59	33.00
Standard Few-Shot	<u>77.71</u>	91.22	82.90	67.32	95.17	68.14	62.93	36.90
Random CoT	76.95	90.70	<u>87.50</u>	65.75	<u>95.28</u>	70.17	63.84	70.80
Auto-CoT	75.51	90.60	79.00	63.39	93.90	69.94	59.74	–
Active-Prompt	76.57	<u>91.30</u>	85.00	<u>68.11</u>	94.29	**71.01**	<u>65.63</u>	<u>73.60</u>
ReLTA (ours)	**80.31**	**92.37**	**87.60**	**70.08**	**96.06**	<u>70.76</u>	**67.47**	**80.20**

tion. The comparison therefore controls the final-inference model and decoding budget, but ReLTA additionally incurs offline computation to construct its logic-template pool.

4.3 Ablation Studies

Table 2 examines which design choices account for the gains. We modify one principal design choice at a time and evaluate the resulting variants on all eight benchmarks using Qwen2.5-Coder-7B.

The four variants examine the principal design choices in ReLTA: (1) *Entropy-only* removes the hybrid score's empirical accuracy term; (2) *Error-only* removes its entropy term; (3) *w/o Simplification* retains the answer-checked but uncompressed and verbose rationales; and (4) *w/o Inference Selection* replaces query-dependent retrieval with the first n templates in the constructed logic-template pool. This last variant is deterministic rather than random. Seed selection is part of the hybrid construction stage rather than a separate component in the ablation.

Table 2: Ablation results on eight datasets using Qwen2.5-Coder-7B. Each variant modifies one ReLTA design choice while keeping the same backbone, shot setting, and final-inference sampling budget.

Variant	GSM8K	ASDiv	SVAMP	AQuA	SingleEq	CSQA	StrategyQA	Last Letters
ReLTA	**91.58**	**93.89**	**93.50**	**77.95**	**95.28**	**82.64**	**79.69**	**85.60**
Entropy-only	90.07	93.23	91.60	74.80	94.29	79.52	75.41	83.30
Error-only	89.23	92.56	91.70	72.44	93.62	80.27	77.47	80.90
w/o Simplification	86.66	90.51	90.80	66.54	93.70	80.10	77.82	76.20
w/o Inference Selection	90.60	92.08	90.70	74.41	93.11	78.87	75.55	84.00

Removing Simplification causes the largest drops on AQuA (11.41 points) and Last Letters (9.40 points), showing that simplified traces are more effective than the corresponding verbose traces under the evaluated prompt budget. This experiment does not isolate whether the gain comes from removing irrelevant reasoning, reducing context consumption, or both. Replacing Inference Demonstration Selection with the fixed first-n selection also degrades performance, including 4.14 points on StrategyQA and 3.54 points on AQuA. Finally, both entropy-only and error-only selection underperform the hybrid score: prediction error captures consistently wrong predictions, while entropy captures unstable outputs. These results support the hybrid difficulty criterion, Simplification, and query-specific template retrieval under the evaluated setup; the isolated contribution of Reconstruction is not evaluated by the current ablations.

4.4 Threshold Sensitivity

We examine the sensitivity of the retrieval gate by varying the cosine-similarity threshold θ on GSM8K, AQuA, and StrategyQA under Qwen2.5-Coder-7B. Across the evaluated settings, $\theta = 0.35$ provides a reasonable common operating point rather than requiring a dataset-specific threshold. Detailed results are reported in Appendix A.2 and Table 4.

4.5 Retriever Interface Analysis

We compare the lexical BM25 retriever, the general semantic BGE-small-en-v1.5 retriever, and the reasoning-aware RaDeR interface using the same Qwen2.5-Coder-7B backbone, logic-template pool, maximum number of demonstrations, and decoding budget. Each interface ranks the same pool with its native scoring function. BM25 directly returns its top-k templates, whereas BGE and RaDeR first retain templates above the cosine-similarity threshold and then select up to k templates. Because these scores are not calibrated to a shared scale, the comparison evaluates the downstream utility of each native retrieval protocol rather than a common absolute threshold.

Table 3: Retriever interface comparison using the same logic-template pool, Qwen2.5-Coder-7B backbone, and maximum number of demonstrations. BM25 uses native top-k ranking, whereas BGE and RaDeR apply cosine-threshold filtering followed by top-k selection.

Retriever Interface	GSM8K	AQuA	StrategyQA
BM25	88.70	73.44	78.30
BGE-small-en-v1.5	89.92	74.80	76.51
RaDeR	**91.58**	**77.95**	**79.69**

RaDeR yields the highest downstream accuracy on all three datasets, exceeding BM25 by 2.88 points on GSM8K and 4.51 points on AQuA. This comparison indicates that the reasoning-aware retrieval interface is more effective than the lexical and general semantic alternatives under the evaluated setup.

A qualitative GSM8K case further illustrates the difference between surface similarity and solution-procedure compatibility. For a query involving "fruity scents," the general semantic retriever selects a problem about "bowls of fruit." Although the two questions share surface vocabulary, the retrieved problem follows a linear-equation structure rather than the target two-group rate-multiplication structure. RaDeR instead retrieves an exemplar with less lexical overlap but a closer operational pattern: two independent groups, group-specific multipliers, and final aggregation. This contrast illustrates the intended interaction between refinement and retrieval: refinement exposes the useful procedure, while the reasoning-aware interface selects a procedure compatible with the current query.

Additional analyses of threshold sensitivity, logic-template construction, and representative failure cases are provided in Appendix A.

5 Conclusion

We introduced ReLTA, a general and extensible framework for constructing and selecting query-adaptive reasoning demonstrations. By combining empirical prediction error with answer-space entropy, ReLTA identifies model-specific failures, distills them into compact, instance-grounded logic templates, and selects structurally compatible demonstrations at inference time. Across eight reasoning benchmarks and two reasoning backbones, ReLTA achieves the highest accuracy among the evaluated methods on seven of eight benchmarks for each backbone, supporting the effectiveness of failure-driven template construction and structure-aware demonstration selection. Because larger-parameter reasoning models are costly to deploy locally and expensive to access through APIs, this study does not evaluate ReLTA on such models. Future work can examine its scalability across larger backbones and broader model families. More broadly, the modular design of ReLTA allows its selection, construction, verification, and retrieval components to be independently extended, providing a flexible foundation for incorporating step-level verification, dynamic template expansion, and alternative retrieval strategies across broader reasoning settings.

AI USE STATEMENT

We used generative AI tools to assist with language editing and manuscript organization. The authors reviewed and verified the final manuscript, including all technical claims, equations, tables, and experimental descriptions, and take responsibility for the submitted content.

REPRODUCIBILITY STATEMENT

The paper specifies the datasets, backbone models, retrieval interfaces, decoding settings, exemplar-construction stages, and final-inference evaluation protocol used in the experiments.

REFERENCES

Xun Che, Wenjia Wu, Yadang Chen, Luanjuan Jiang, and Qianmu Li. Select prompting with chain-of-thought paired with large language models. *Expert Syst. Appl.*, 312:131511, 2026. doi: 10.1016/J.ESWA.2026.131511. URL https://doi.org/10.1016/j.eswa.2026.131511.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems. *arXiv e-prints*, art. arXiv:2110.14168, October 2021. doi: 10.48550/arXiv.2110.14168.

Debrup Das, Seán Ó Nualláin, and Razieh Rahimi. RaDeR: Reasoning-aware dense retrieval models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 19970–19997. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.1011. URL https://doi.org/10.18653/v1/2025.emnlp-main.1011.

Shizhe Diao, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu, and Tong Zhang. Active prompting with chain-of-thought for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 1330–1350. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.73. URL https://doi.org/10.18653/v1/2024.acl-long.73.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1–5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=yf1icZHC-l9.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:607–623, 2021. doi: 10.1162/tacl_a_00370. URL https://aclanthology.org/2021.tacl-1.35/.

Shivanshu Gupta, Matt Gardner, and Sameer Singh. Coverage-based example selection for in-context learning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, Findings of ACL, pp. 13924–13950. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.930. URL https://doi.org/10.18653/v1/2023.findings-emnlp.930.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 1051–1068. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.67. URL https://doi.org/10.18653/v1/2023.emnlp-main.67.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=jKN1pXi7b0.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.EMNLP-MAIN.550. URL https://doi.org/10.18653/v1/2020.emnlp-main.550.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/8bb0d291acd4acf06ef112099c16f326-Abstract-Conference.html.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. MAWPS: A math word problem repository. In Kevin Knight, Ani Nenkova, and Owen Rambow (eds.), *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1152–1157, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1136. URL https://aclanthology.org/N16-1136/.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=VD-AYtP0dve.

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. Unified demonstration retriever for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4644–4668. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.256. URL https://aclanthology.org/2023.acl-long.256/.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 158–167. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1015. URL https://aclanthology.org/P17-1015/.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pp. 100–114. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.deelio-1.10. URL https://aclanthology.org/2022.deelio-1.10/.

Andrea Matarazzo and Riccardo Torlone. A survey on large language models with some insights on their capabilities and limitations. *CoRR*, abs/2501.04040, 2025. doi: 10.48550/ARXIV.2501.04040. URL https://doi.org/10.48550/arXiv.2501.04040.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 975–984. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.92. URL https://aclanthology.org/2020.acl-main.92/.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of GPT-4. *CoRR*, abs/2306.02707, 2023. doi: 10.48550/ARXIV.2306.02707. URL https://doi.org/10.48550/arXiv.2306.02707.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168/.

Kiran Purohit, Venktesh V, Sourangshu Bhattacharya, and Avishek Anand. Sample efficient demonstration selection for in-context learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/purohit25a.html.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2655–2671. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.naacl-main.191. URL https://aclanthology.org/2022.naacl-main.191/.

Jon Saad-Falcon, Daniel Y. Fu, Simran Arora, Neel Guha, and Christopher Ré. ReasonIR: Training retrievers for reasoning tasks. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=4tPyDuJw47.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421/.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022a. URL https://arxiv.org/abs/2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022b. URL https://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/20b9e6ec13bb7f1c6e1941c1dd7e15ca-Abstract-Conference.html.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. Compositional exemplars for in-context learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, Proceedings of Machine Learning Research, pp. 39818–39833. PMLR, 2023. URL https://proceedings.mlr.press/v202/ye23c.html.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9,*

2022, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/639a9a172c044fbb64175b5fad42e9a5-Abstract-Conference.html.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=5NTt8GFjUHkr.

A APPENDIX

This appendix provides the complete Logic Template Construction procedure and additional experimental analyses omitted from the main text for space. It includes the construction algorithm, threshold sensitivity, a qualitative construction example, construction statistics, and representative failure cases.

A.1 COMPLETE LOGIC TEMPLATE CONSTRUCTION PROCEDURE

This appendix provides the complete procedure for the Logic Template Construction stage described in the method section. Given the selected seed set $\mathcal{S}$, the offline refinement model $\mathcal{M}_{\mathrm{Re}}$, and the prompts for Generation, Reconstruction, and Simplification, the procedure constructs a validated template pool $\mathcal{S}_{\mathrm{constructed}}$. Algorithm 1 makes explicit the conditional execution order and fallback behavior summarized in the main text.

Algorithm 1 Logic Template Construction

Input: Active seed pool $\mathcal{S} = \{(q_i, y_i^*)\}_{i=1}^{K}$, refinement model $\mathcal{M}_{\mathrm{Re}}$, direct-generation prompt P_{gen}, reconstruction prompt P_{recon}, and simplification prompt P_{simp}.
Output: Constructed logic-template pool $\mathcal{S}_{\text{constructed}}$.

$\mathcal{S}_{\text{constructed}} \leftarrow \emptyset$
for each $(q, y^*) \in \mathcal{S}$ **do**
  // Step 1: Generation
  Generate $(\tau^{(1)}, \hat{y}^{(1)}) \leftarrow \mathcal{M}_{\mathrm{Re}}(q \mid P_{\text{gen}})$
  if $\mathrm{FinalAnswerCheck}(\hat{y}^{(1)}, y^*)$ **then**
    $\tau_{\text{curr}} \leftarrow \tau^{(1)}$
  else
    // Step 2: Reconstruction
    Reconstruct $(\tau^{(2)}, \hat{y}^{(2)}) \leftarrow \mathcal{M}_{\mathrm{Re}}(q, y^* \mid P_{\text{recon}})$
    if $\mathrm{FinalAnswerCheck}(\hat{y}^{(2)}, y^*)$ **then**
      $\tau_{\text{curr}} \leftarrow \tau^{(2)}$
    else
      continue *// Discard candidate*
    end if
  end if
  // Step 3: Simplification
  Simplify $(\tau^{(3)}, \hat{y}^{(3)}) \leftarrow \mathcal{M}_{\mathrm{Re}}(q, \tau_{\text{curr}} \mid P_{\text{simp}})$ *// without y^**
  if $\mathrm{FinalAnswerCheck}(\hat{y}^{(3)}, y^*)$ **then**
    $\mathcal{S}_{\text{constructed}} \leftarrow \mathcal{S}_{\text{constructed}} \cup \{(q, y^*, \tau^{(3)})\}$
  else
    $\mathcal{S}_{\text{constructed}} \leftarrow \mathcal{S}_{\text{constructed}} \cup \{(q, y^*, \tau_{\text{curr}})\}$ *// Fallback*
  end if
end for
return $\mathcal{S}_{\text{constructed}}$

For each selected seed, Generation first produces a reasoning trace without access to the reference answer. If its parsed final answer is inconsistent with y^*, Reconstruction uses y^* to recover an answer-consistent trace; candidates that still fail are discarded. Simplification then removes unnecessary reasoning detours without receiving y^*. A simplified trace is retained only when it preserves the correct final answer; otherwise, the procedure falls back to the answer-consistent trace obtained before simplification. Consequently, every retained template passes the Final-Answer Consistency Check, although this check does not independently verify every intermediate reasoning step.

A.2 THRESHOLD SENSITIVITY

We vary the cosine-similarity threshold θ while keeping the backbone, retriever, candidate templates, and all other inference settings fixed. The reported values are task accuracy in percentage points under Qwen2.5-Coder-7B. A larger θ admits only demonstrations with stronger retriever similarity;

when no candidate satisfies the threshold, ReLTA uses the zero-shot fallback described in the method section.

Table 4: Threshold sensitivity on representative datasets using Qwen2.5-Coder-7B. Larger θ applies stricter similarity filtering.

Dataset	$\boldsymbol{\theta = 0.00}$	$\boldsymbol{\theta = 0.20}$	$\boldsymbol{\theta = 0.35}$	$\boldsymbol{\theta = 0.50}$
GSM8K	90.90	90.84	**91.58**	91.36
AQuA	75.80	75.62	**77.95**	76.41
StrategyQA	77.86	77.77	**79.69**	79.65

As shown in Table 4, $\theta = 0.35$ gives the highest accuracy on all three representative datasets. Relative to disabling similarity filtering ($\theta = 0.00$), the improvement is small on GSM8K but more pronounced on AQuA and StrategyQA. Increasing the threshold further to 0.50 does not provide a consistent additional benefit, indicating that overly strict filtering can remove useful demonstrations. We therefore use $\theta = 0.35$ as a shared operating point rather than tuning a separate threshold for each dataset.

A.3 Logic Template Construction Case Study

Table 5 illustrates how Simplification transforms a valid but exploratory reconstructed trace into a more concise logic template. The left column gives the selected seed question, the middle column shows the reconstructed reasoning before simplification, and the right column shows the retained template. This example clarifies the transformation performed by the construction pipeline rather than establishing its aggregate effectiveness, which is evaluated by the ablation results in the main text.

Table 5: Qualitative comparison between a raw reconstructed trace and the constructed logic template on AQuA.

Seed Query	**Raw Reconstructed CoT**	**Constructed Logic Template**
What should come in place of the question mark in the following number series? 5 8 10 12 16 (?) *Choices:* (A) 15 (B) 18 (C) 17 (D) 21 (E) 23	Let's think step by step. To calculate the next term in the number series, we need to know the pattern governing the sequence. The given series is: 5, 8, 10, 12, 16, (?). First, examine the differences between consecutive terms: $8-5=3$; $10-8=2$; $12-10=2$; $16-12=4$. So the differences are: 3, 2, 2, 4. This does not form a consistent arithmetic progression. Let's look for another pattern... [... *trial-and-error omitted for brevity* ...] Wait—maybe the pattern is based on prime numbers plus something? $5=5$; $8=5+3$; $10=5+5$; $12=5+7$; $16=5+11$. The added values are 3, 5, 7, and 11—the first four odd primes. The next prime is 13, so the next term is $5+13=18$. Therefore, the answer is (B).	Let's think step by step. 1. The sequence adds consecutive odd prime numbers to the first term (5): • $5+3=8$ • $5+5=10$ • $5+7=12$ • $5+11=16$ 2. Added values (3, 5, 7, 11) are the first four odd primes. 3. The next odd prime is 13. 4. The next term is $5+13=18$. **Answer: (B).**

The reconstructed trace initially tests an unproductive consecutive-difference pattern before identifying the relevant relation between the sequence and consecutive odd primes. The constructed template removes this detour while preserving the necessary quantities, the prime-number pattern, and the calculation yielding the final answer. Thus, Simplification reduces surface-level exploration without replacing the instance-grounded reasoning structure with an abstract label or unsupported shortcut.

A.4 Logic Template Construction Statistics

To characterize the offline construction process, we sample 100 candidate instances from each distinct candidate pool and report three quantities in Table 6: the number of retained templates after the consistency checks, the average word count before and after Simplification, and the corresponding relative reduction. SVAMP, ASDiv, and SingleEq reuse the GSM8K candidate pool under the cross-domain protocol and therefore do not require separate construction runs.

Table 6: Logic Template Construction statistics on 100 sampled instances per candidate pool. Accepted reports retained templates over candidate instances (n/N). Average lengths are measured in words, and reduction is computed from the ratio of the two reported averages. The GSM8K pool is also used for SVAMP, ASDiv, and SingleEq under the cross-domain protocol.

Candidate Pool	**Accepted** (n/N)	**Original (words)**	**Simplified (words)**	**Reduction (%)**
GSM8K	100/100	237.75	91.85	61.37%
AQuA	93/100	775.27	112.32	85.51%
CSQA	98/100	174.93	101.86	41.77%
StrategyQA	94/100	55.13	44.29	19.66%
Last Letters	100/100	40.00	35.50	11.25%

Overall, the pipeline retains 485 of 500 candidates, showing that most selected seeds yield an answer-consistent template after Generation or Reconstruction. Compression varies with the verbosity of the original traces: AQuA exhibits the largest reduction (85.51%), whereas Last Letters shows a smaller reduction (11.25%) because its original traces are already short. Across the five construction runs, the procedure uses approximately 855K offline model tokens, excluding the optional process auditor. These costs are incurred once during template-pool construction and are separate from the final inference budget reported in the main experiments.

A.5 Representative Failure Cases

We analyze representative incorrect predictions on GSM8K under Qwen2.5-Coder-7B. The following cases illustrate two remaining failure modes: premise omission and limited structural coverage.

Premise omission. Even when the reasoning-aware retriever provides a relevant template, the reasoning model may fail to incorporate every constraint in the test query. Consider the query: *"Anna had a budget of $16, and she has already spent $4. She also bought 2 books... If Anna has $2 left, how much did each book cost?"* The correct calculation is $(16 - 4 - 2)/2 = 5$. However, the model repeatedly outputs 7 and reasons that Anna spent $16 - 2 = 14$, followed by $14/2 = 7$. The model omits the intermediate constraint that Anna had already spent $4. This error reflects a limitation in query-specific constraint tracking rather than an obviously irrelevant retrieved exemplar.

Limited structural coverage. Inference Demonstration Selection can only select from solution structures represented in the constructed logic-template pool. For idiosyncratic or out-of-distribution mathematical problems, the pool may contain no close structural counterpart. Similarity-threshold filtering can then retain a suboptimal exemplar whose execution graph differs from the required solution, leading to logical misalignment or negative transfer. Logic Template Construction and Inference Demonstration Selection improve the available demonstrations, but they cannot guarantee coverage of every solution structure.