

000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 STEPWISER: STEPWISE GENERATIVE JUDGES FOR WISER REASONING

Anonymous authors

Paper under double-blind review

ABSTRACT

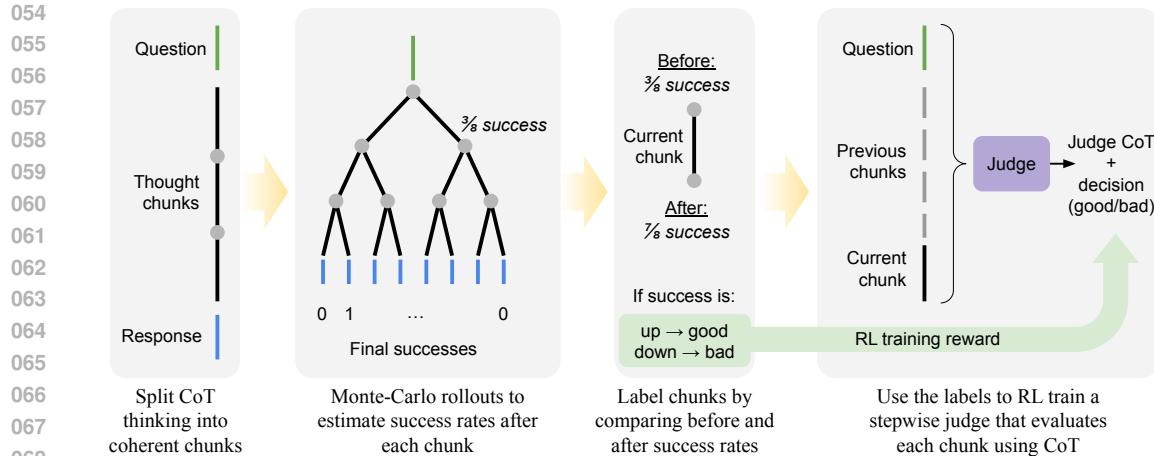
As models increasingly leverage multi-step reasoning strategies to solve complex problems, supervising the logical validity of these intermediate steps has become a critical research challenge. Process reward models address this by providing step-by-step feedback, but current approaches have two major drawbacks: they typically function as classifiers without providing explanations, and their reliance on supervised fine-tuning with static datasets limits generalization. Inspired by recent advances, we reframe stepwise reward modeling from a classification task to a reasoning task itself. We thus propose a generative judge that *reasons about the policy model’s reasoning steps* (i.e., meta-reasons), outputting thinking tokens before delivering a final verdict. Our model, STEPWISER, is trained by reinforcement learning using relative outcomes of rollouts. We show it provides (i) better judgment accuracy on intermediate steps than existing methods; (ii) can be used to improve the policy model at training time; and (iii) improves inference-time search.

1 INTRODUCTION

As large language models (LLMs) increasingly tackle complex problems, they rely on multi-step reasoning strategies like Chain-of-Thought (CoT) (Wei et al., 2022) and ReAct (Yao et al., 2022) to decompose tasks and formulate better solutions. Consequently, ensuring these intermediate reasoning steps possess logical validity has become a critical research challenge. Process Reward Models (PRMs) have emerged as a potential tool to meet this need, providing step-by-step feedback for supervising learning, instead of relying on a single, often sparse, outcome-based reward (Lightman et al., 2023; Wang et al., 2023). However, this approach suffers from two major drawbacks. First, current PRMs typically function as “black-box” classifiers, providing a score or label without explaining why a step is correct or flawed. Second, their reliance on supervised fine-tuning (SFT) with static datasets can limit their ability to generalize to new reasoning patterns (Lightman et al., 2023; Luo et al., 2024; Wang et al., 2023; Xiong et al., 2024b; Zhang et al., 2024a). In contrast, reasoning models themselves are trained to produce CoTs with reinforcement learning (RL) for best performance (DeepSeek-AI et al., 2025).

In this paper we propose to reward intermediate reasoning steps by first **reasoning about those reasoning steps**, before making a judgment – a meta-reasoning process which itself is trained by RL. Our overall method (as shown in Figure 1) to build such a *stepwise generative judge* involves 3 components: (1) a new self-segmentation technique to equip the base policy model with the ability to produce coherent and informative reasoning chunks (chunks-of-thought); (2) assignment of target rewards to chunks via relative outcomes of rollouts; and (3) online training of judgment reasoning chains (i.e., reasoning about reasoning) and final reward judgments via RL. Our stepwise judge, termed STEPWISER, can then be used to provide rewards either at training time or inference time in order to improve the reasoning ability of the policy model.

We conduct a comprehensive evaluation of our method across three key dimensions: (i) the judge’s classification accuracy on intermediate steps, e.g., via its score on ProcessBench (Zheng et al., 2024); (ii) its performance in a new inference-time search paradigm where the judge cleans up the reasoning history and re-samples – a method we propose for efficiently scaling sequential computation while maintaining the original generation length; and (iii) its utility in data selection for downstream model training. Our experiments demonstrate that our RL-trained generative stepwise judge significantly



108 2.2 JUDGE ARCHITECTURES
109110 The process labels and signals described above can be used to train judges with different distinct
111 architectures and training paradigms.112 **Discriminative PRMs** The most straightforward approach is to treat the task as a *classification*
113 *problem*. This involves replacing the language model’s final layer with a linear head and fine-tuning
114 it to predict a binary label for each step using a cross-entropy loss (Lightman et al., 2023). A
115 more recent method formulates the task as next-token prediction, prompting the LLM to generate a
116 pre-defined token (e.g., + or -) as its judgment (Wang et al., 2023; Xiong et al., 2024b). This approach
117 further dates back to preference reward model training (Dong et al., 2024; Liu et al., 2023). Although
118 this method uses a generative mechanism, its function remains purely discriminative, as it outputs a
119 simple judgment without justification. We therefore group both under the discriminative category.120 **Generative judges with CoT reasoning** In sharp contrast, the second and most recent paradigm
121 is the generative reasoning judge. Here, the evaluation itself is framed as a reasoning task. The
122 judge first generates an explicit CoT to explain its rationale before outputting its final judgment.
123 This approach was initially explored for preference learning and ORMs (Zhang et al., 2024b; Chen
124 et al., 2025). There are also a few very recent works studying this paradigm shift in the context of
125 stepwise judges, including Zhao et al. (2025); Zha et al. (2025); Khalifa et al. (2025). Though we
126 share similar spirit of leveraging the inherent reasoning ability of the LLMs to train a stepwise judge,
127 the algorithmic designs are distinctly different.128 **Comparison to Recent Work** First, in contrast to works focusing on offline rejection sampling
129 fine-tuning (Zhao et al., 2025; Khalifa et al., 2025), we identified critical scalability issues with such
130 static methods. While offline fine-tuning provides dramatic initial performance gains, we observed
131 that learning quickly plateaus. Specifically, the model’s loss stagnates after training on a relatively
132 small dataset (e.g., 10k samples), preventing further improvement. In contrast, we observe that
133 online RL framework allows for continuous learning and scalability, successfully showing sustained
134 improvement on over 800k samples.135 Second, our approach fundamentally diverges from RL methods that rely on sparse, trajectory-level
136 supervision, such as Zha et al. (2025). Specifically, they prompt the LLMs to evaluate each individual
137 step and final answer but only the final verification is supervised. Their approach assumes that to get
138 an accurate evaluation of the final answer, models *implicitly become a stepwise judge*. Our divergence
139 is rooted in strong theoretical evidence from preference learning, where an exponential gap in sample
140 complexity can exist between learning from sparse final outcomes versus dense intermediate rewards
141 (Zhong et al., 2024, c.f. Proposition 3.2).142 This motivated our central hypothesis that a similar principle governs the training of generative
143 judges. However, applying RL to decoupled, individual steps is non-trivial. We discovered that this
144 fine-grained approach introduces unique challenges, namely (1) a susceptibility to majority class bias,
145 and (2) noisy signals arising from naive trajectory segmentation. To address these, our framework
146 incorporates two targeted solutions: self-segmentation fine-tuning to generate coherent reasoning
147 steps, and prompt set balancing to mitigate bias. As our experiments will demonstrate, it is this
148 complete framework—harnessing the power of explicit, stepwise signals while actively correcting for
149 their inherent challenges—that is essential for training state-of-the-art generative judges.

150

151 3 METHOD: TRAINING STEPWISE GENERATIVE JUDGES WITH RL
152153 As depicted in Figure 1, our overall method STEPWISER consists of three components:
154155 1. We equip the base policy model with the ability to self-segment Chain-of-Thoughts into
156 coherent and informative reasoning chunks, called Chunks-of-Thought. This is done by
157 creating SFT data with informative segments, so that the model can be trained to self-
158 segment. We show that this causes no loss in performance for the base model and is critical
159 to reduce the training noises during RL by removing the meaningless chunk.
160 2. Given the chunks generated by the policy model, we annotate each chunk to create training
161 data for our generative stepwise judge with binary target labels. This is done by comparing
outcomes of rollouts starting before and after the given chunk using the outcome rewards.

162 3. We perform online RL training using GRPO which trains our stepwise judge model to
 163 produce judgment reasoning chains (i.e., reasoning about reasoning) and reward final
 164 judgments that match the chunk labels from the previous step.

165
 166 We describe the three components in detail in the following three subsections.

167
 168 **3.1 COT GENERATION WITH SELF-SEGMENTATION (CHUNKS-OF-THOUGHT)**

169 A core challenge in evaluating reasoning processes is defining what a “step” is. Simple heuristics,
 170 like splitting on double line breaks or predefined tokens like “Step 1, Step 2”, often create fragmented
 171 or logically incomplete steps, making them difficult for a judge to evaluate. We present representative
 172 examples in appendix (Table 11-13), where the model tends to insert double line breaks before and
 173 after a mathematical equation. This breaks an intuitively unified logical step into multiple different
 174 chunks, where one chunk contains a textual explanation, and the next with the corresponding equation.

175 **Achieving better step definition via self-segmentation.** To mitigate this issue, we propose a method
 176 to teach the model to generate and simultaneously self-segment its own reasoning chains into more
 177 meaningful steps. First, we define the criteria for a high-quality reasoning step. The core idea is that
 178 each step should represent a complete logical leap or a self-contained part of the problem-solving
 179 process. Our definitions are given in Table 5. We then create our training data by:

180
 181 1. Generating a set of initial reasoning trajectories from the base model.
 182 2. Using an LLM prompted with our rules, to automatically segment these trajectories into
 183 logically coherent steps.

184 We fine-tune our base model on this data, thus teaching it to generate and simultaneously self-segment
 185 its own reasoning chains automatically. This self-segmentation ability is crucial for two main reasons.
 186 First, it produces more informative and logically complete steps, which provides better context for
 187 our judge model and improves its evaluation accuracy. Second, this method significantly reduces the
 188 total number of steps per trajectory. This reduction is also important because, as we will show, the
 189 process of annotating each step with a quality label is computationally expensive.

190
 191 **3.2 STEPWISE DATA ANNOTATION**

192 **Stepwise data annotation via Q value estimation** Previous work has used human labelers to
 193 annotate correctness of each reasoning step (Lightman et al., 2023), although most such data is
 194 collected for proprietary models that we cannot access. Other works annotate steps automatically
 195 using methods like Monte Carlo estimation (Wang et al., 2023). We follow this second approach,
 196 using an estimated Q-value to measure the quality of each step.

197 For a given training prompt x with verifiable outcome rewards, we generate a response from our
 198 policy model π which segments its CoT into chunks $a = [a_1, a_2, \dots, a_H]$, where a_i is the i -th
 199 reasoning chunk. Then, the Q value of an individual step a_i and its history is the expected final
 200 reward starting from that point:

201
$$Q^\pi([x, a_{1:i-1}], a_i) := Q^\pi(s_{i-1}, a_i) = \mathbb{E}_{a_{i+1:H} \sim \pi(\cdot|x, a_{1:i})} r^*(x, a_{1:H}), \quad (1)$$

202 where $s_i := [x, a_{1:i-1}]$ is the history, and r^* is a final reward, which can be 1 for correct answers and
 203 0 otherwise. We estimate this Q-value by generating M full completions $a_{i+1:H}^j$ from that step a_i
 204 and calculating the average final reward, i.e. the ratio of correct final answers:

205
$$\widehat{Q}^\pi(s_{i-1}, a_i) = \frac{1}{M} \sum_{j=1}^M r^*(x, a_{1:i}, a_{i+1:H}^j). \quad (2)$$

206 Following prior work (Wang et al., 2023; Xiong et al., 2024b), we can then assign a binary label to
 207 the step based on this Q-value:

208
$$y_i = \begin{cases} + & \text{if } \widehat{Q}^\pi(s_{i-1}, a_i) > 0, \\ - & \text{if } \widehat{Q}^\pi(s_{i-1}, a_i) = 0. \end{cases}$$

209 For convenience, we refer to this labeling approach as *Absolute Q value thresholding* (Abs-Q).

216

217

Table 1: Prompt Template for our STEPWISER judge.

218

219

220

Prompt Template for STEPWISER Judge

221

Instruction:

222

You are a reasoning validator for mathematical problems. Your task is to think step by step and determine if the “New Reasoning Chunk” contains **any explicit errors** based on the problem description and historical context.

223

First, you must **always** perform a step-by-step chain of thought analysis to justify your final judgment. Then, based on your analysis, you will make a definitive judgment. It is OK that the chunk does not contain any numerical calculation.

224

Based on your evaluation, provide your final judgment:

225

- Use **Positive** if the reasoning chunk is free of mistakes.
- Use **Negative** if the reasoning chunk contains one or more mistakes.

226

Input:

227

Mathematical Problem: `{problem}`

228

Historical Reasoning Path: `{history}`

229

New Reasoning Chunk: `{chunk}`

230

231

Output format:

232

1. **Analysis:** [Always provide a step-by-step analysis here. First, briefly state the goal of the current reasoning chunk. Second, verify the logic, method, and any calculations against the problem’s requirements and the historical path. If an error is found, clearly explain the error and why it’s wrong. If the reasoning is correct, explain why it is a valid and logical step forward.]
2. **Final Judgment:** [Provide the final judgment within `\boxed{}`. Examples: `\boxed{Positive}` or `\boxed{Negative}`.]

233

234

235

236

Rewarding the progress One drawback of $\text{Abs-}\mathcal{Q}$ is its insensitivity to the dynamics of the reasoning process. For instance, it does not differentiate between a step that raises the success probability from 10% to 50% and one that drops it from 60% to 55%. To reward progress, we also explore methods that consider the change in \mathcal{Q} -value.

237

238

239

240

241

242

Setlur et al. (2024) proposes to consider the change in value. Specifically, they define the notion of *effective reward* as a combination of \mathcal{Q} value and advantage function of the best-of- n policy induced by r^* :

$$Q^\pi(s_{i-1}, a_i) + \alpha \cdot A^\mu(s_{i-1}, a_i), \quad (3)$$

243

where $\alpha > 0$ is a hyperparameter, and $A^\mu(s_{i-1}, a_i) := Q^\mu(s_{i-1}, a_i) - Q^\mu(s_{i-2}, a_{i-1})$. Here μ is taken as the best-of- n policy with r^* . In other words, we generate n responses from π and use r^* to select the best one. In this case, μ satisfies that $Q^\mu(s_{i-1}, a_i) = 1 - (1 - Q^\pi(s_{i-1}, a_i))^{n/2}$. Therefore, the effective reward can also be estimated via \mathcal{Q} value estimation. Accordingly, we consider an alternative approach of data annotation:

244

245

246

247

$$y_i = \begin{cases} + & \text{if } \widehat{Q}^\pi(s_{i-1}, a_i) + \alpha \cdot \widehat{A}^\mu(s_{i-1}, a_i) > 0, \\ - & \text{if } \widehat{Q}^\pi(s_{i-1}, a_i) + \alpha \cdot \widehat{A}^\mu(s_{i-1}, a_i) = 0, \end{cases}$$

248

249

250

where \widehat{A}^μ is the estimated advantage through the Monte-Carlo estimation of the \mathcal{Q} value. We refer to this labeling approach as *Relative Effective Reward Thresholding* (Rel-Effective).

251

252

253

254

255

As a simpler alternative to capture relative improvement, we also consider a method based on the value ratio, where the label is determined by if $\widehat{Q}^\pi(s_{i-1}, a_i) / \widehat{Q}^\pi(s_{i-2}, a_{i-1}) > \gamma$. We refer this labeling approach as Rel-Ratio.

256

257

258

259

Using one of these methods, we can assign binary label y_i to every step a_i in a reasoning trajectory. Since these labels come from unbiased estimates of the actual \mathcal{Q} -values, they are likely to be more reliable compared to more ad-hoc methods. For example, if a step a_i is the first step with a mistake, rollouts starting after a_i are more likely to fail compared to ones that start before the flawed step a_i .

¹Assuming binary {0, 1} outcome rewards where $(1 - Q^\pi(s_{i-1}, a_i))^n$ is the probability of n rollouts failing.

270
271

3.3 TRAINING THE JUDGE VIA REINFORCEMENT LEARNING

272

273 Our goal is to train a stepwise judge using the segmented (chunked) reasoning chains and stepwise
 274 target labels. While a standard approach is to train a discriminative judge via SFT (Wang et al., 2023;
 275 Xiong et al., 2024b), we adopt a generative formulation inspired by recent studies (Zhang et al.,
 276 2024b; Chen et al., 2025; Whitehouse et al., 2025). We frame the evaluation as a reasoning task
 277 where the judge first generates a CoT analysis of the step in question, then outputs a final judgment.
 278 This generative process is compelling because it forces the judge to “show its work,” leading to a
 279 more transparent and potentially more accurate evaluation.

280

The Insufficiency of Offline Fine-tuning. We first considered a simpler offline approach, rejection sampling fine-tuning (RFT), but found it has a critical limitation. On a static dataset, the training loss plateaus very quickly after very few optimization steps (Figure 2). This demonstrates that static methods are insufficient for this complex task, motivating our adoption of an online RL framework to leverage a continuous and more diverse set of learning signals.

281

RL Task Formulation and Training. We formulate the training as a step-level judgment task. For each step, the judge model is given the original problem x , the reasoning history $a_{1:i-1}$, and the current chunk a_i to evaluate. It is then prompted to generate a CoT rationale explaining its analysis, followed by a final verdict (see prompt template in Table 1).

282

Prompt balancing. A critical challenge we identified in this stepwise setting is the severe label imbalance produced by our data annotation process (e.g., over 70% positive labels with $\text{Abs}-\mathbb{Q}$). Our early experiments showed this leads to degenerate judges that simply learn to always predict majority class. To address this, we incorporate **prompt dataset balancing** by down-sampling the majority class to ensure a 1:1 ratio of positive and negative examples. This simple technique proved essential for stable training and robust performance, an impact we quantify in our ablation study in Section 4.3.

283

The RL training itself is straightforward. The judge receives a reward of 1 if its verdict matches the target label y_i and 0 otherwise. We use the GRPO algorithm (Shao et al., 2024) for optimization.

284

285

4 EXPERIMENTS

286

287 We use the Qwen2.5-1.5B-it and 7B-it models (Yang et al., 2024) as our base models. The prompts
 288 come from a subset of Numinath-CoT dataset (Beeching et al., 2024), which we preprocess by
 289 removing duplicates and filtering for problems verifiable by Math-Verify. This section highlights a
 290 key result that enables our pipeline’s feasibility; full implementation details for all training stages,
 291 hyperparameters, and additional ablations are deferred to Appendix B.

292

293

4.1 SELF-SEGMENTATION FINE-TUNING

294

295 We first fine-tune the base policy to segment its own reasoning into meaningful “chunks”. This is
 296 achieved by generating solutions to 20k problems, using a powerful teacher model (Llama-3.1-70B-it)
 297 to segment them into high-quality steps, and then fine-tuning our policy on this data. As shown in
 298 Table 6, this crucial pre-training step significantly reduces the number of steps per solution (e.g., from
 299 9.6 to 6.0) without harming task performance.

300

301

302 This reduction in complexity is vital for two reasons: it (1) substantially lowers the computational
 303 cost of our subsequent data annotation stage, and (2) provides a cleaner, less noisy signal for the
 304 final RL training by filtering out trivial or meaningless steps. We provide detailed ablation studies in
 305 Appendix C.1 that further verify the effectiveness of this approach.

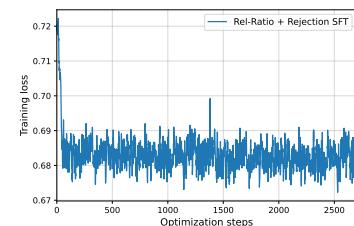


Figure 2: The training loss of offline RFT on a static dataset. The loss saturates quickly, indicating that learning has stagnated.

324
 325 **Table 2: ProcessBench results.** Average accuracy (Avg) of our method STEPWISER is better than all
 326 variants of our discriminative baselines, and existing baselines in the literature (first rows). Further
 327 comparisons are given in Appendix Table 9.

| Method | Learning signal | GSM8K | MATH | Olympiad | Omni-MATH | Avg ↑ |
|----------------------------------|-----------------|-------------|-------------|-------------|-------------|--------------------|
| <i>Existing Reference Models</i> | | | | | | |
| Math-Shepherd-PRM-7B | Abs-Q | 47.9 | 29.5 | 24.8 | 23.8 | 31.5 |
| RLHflow-Llama3-8B-it | Abs-Q | 50.4 | 33.4 | 13.8 | 15.8 | 28.4 |
| Skywork-Qwen2.5-Math-7B-it | Abs-Q | 70.8 | 53.6 | 22.9 | 21.0 | 42.1 |
| Eurus-Qwen2.5-Math-7B-it (DPO) | Outcome | 56.6 | 43.0 | 27.3 | 26.8 | 35.1 |
| RL-TANGO-Qwen2.5-7B-it | Outcome | 53.1 | 48.2 | 37.8 | 36.3 | 43.9 |
| <i>Qwen2.5-1.5B-chunk</i> | | | | | | |
| Discriminative + SFT | Abs-Q | 39.3 | 32.1 | 19.3 | 18.9 | 27.2 |
| Discriminative + SFT | Rel-Effective | 40.8 | 37.2 | 18.7 | 20.1 | 29.2 |
| Discriminative + SFT | Rel-Ratio | 32.1 | 32.0 | 14.2 | 18.0 | 24.1 |
| Generative CoT + RL (STEPWISER) | Abs-Q | 49.2 | 40.5 | 23.8 | 31.0 | 36.1 |
| Generative CoT + RL (STEPWISER) | Rel-Effective | 48.2 | 43.6 | 22.1 | 25.3 | 34.8 |
| Generative CoT + RL (STEPWISER) | Rel-Ratio | 46.9 | 43.4 | 26.3 | 28.4 | 36.2 |
| <i>Qwen2.5-7B-chunk</i> | | | | | | |
| Discriminative + SFT | Abs-Q | 54.8 | 45.9 | 28.0 | 26.9 | 38.9 |
| Discriminative + SFT | Rel-Effective | 55.6 | 48.7 | 26.4 | 28.3 | 39.7 |
| Discriminative + SFT | Rel-Ratio | 48.6 | 46.9 | 21.9 | 25.4 | 35.7 |
| Generative CoT + RL (STEPWISER) | Abs-Q | 61.9 | 61.0 | 48.4 | 43.9 | 53.8 |
| + Maj@8 | Abs-Q | 65.5 | 62.1 | 49.7 | 45.7 | 55.8 (+2.0) |
| Generative CoT + RL (STEPWISER) | Rel-Effective | 72.4 | 68.3 | 54.4 | 52.4 | 61.9 |
| + Maj@8 | Rel-Effective | 72.9 | 72.1 | 57.3 | 54.0 | 64.1 (+2.2) |
| Generative CoT + RL (STEPWISER) | Rel-Ratio | 72.6 | 67.2 | 52.3 | 49.8 | 60.5 |
| + Maj@8 | Rel-Ratio | 74.3 | 69.0 | 53.8 | 50.2 | 61.8 (+1.3) |

4.2 EVALUATION ON PROCESSBENCH

We first evaluate STEPWISER on ProcessBench (Zheng et al., 2024), a benchmark for identifying the first incorrect step in a reasoning process. Performance is measured by the harmonic mean of accuracy on problems with correct (acc_1) and incorrect (acc_2) final answers, calculated as $2 \times (acc_1 \times acc_2) / (acc_1 + acc_2)$.

STEPWISER significantly outperforms SFT and RL baselines. Our primary results in Table 2 show that our RL-trained generative judge, STEPWISER, consistently and substantially outperforms baselines. First, it is far superior to SFT-trained discriminative judges across all signals and model scales. For instance, our 7B model with the Rel-Effective signal scores 61.9, beating the SFT baseline’s 39.7. Its performance also surpasses that of similar community-trained PRMs.

Our RL-trained STEPWISER judge significantly outperforms existing RL-trained judges. Furthermore, we benchmark STEPWISER against other models trained with online methods like online DPO (Xiong et al., 2023; Xu et al., 2023) or GRPO (Shao et al., 2024) (e.g., Eurus-7B, RL-TANGO-7B). Unlike our method, these models are supervised at the *trajectory level*, using only the final answer’s correctness as a reward signal, denoted by “Outcome” in Table 2. For instance, our best 7B model’s score of 61.9 is well ahead of RL-TANGO’s 43.9. This result provides strong evidence for our core hypothesis: that direct, dense, step-level supervision provides a much richer and more effective learning signal for training process reward models.

4.3 ANALYSIS OF KEY COMPONENTS TO THE PERFORMANCE GAP

To understand the source of this performance gap, we conduct a series of ablation studies comparing our full STEPWISER method against baselines that remove one key component at a time: (1) *Ablate RL* by using offline rejection sampling (RFT) (Dong et al., 2023); (2) *Ablate CoT* by training a discriminative judge with RL where the model directly outputs a token to indicate the judgment; and (3) *Ablate prompt dataset balancing*. For brevity, we focus on the Rel-Ratio signal here, as the trends are consistent across others (see Appendix for more).

Online RL contributes to the performance improvement. The importance of online learning is evident when comparing our full STEPWISER model to the RFT baseline. On ProcessBench using

378
 379 Table 3: Ablation study results on ProcessBench. The results show that both the generative CoT
 380 reasoning and RL components of our STEPWISER method are important for overall results.
 381

| Method | GSM8K | MATH | Olympiad | Omni-MATH | Avg ↑ |
|---|-------------|-------------|-------------|-------------|-------------|
| <i>Qwen2.5-1.5B-chunk</i> | | | | | |
| Discriminative + SFT (Baseline) | 32.1 | 32.0 | 14.2 | 18.0 | 24.1 |
| STEPWISER (Generative Reasoning + RL) | 46.9 | 43.4 | 26.3 | 28.4 | 36.2 |
| – Ablate RL (use RFT) | 32.8 | 23.9 | 16.3 | 19.6 | 23.1 |
| – Ablate CoT (use Discriminative format + RL) | 42.0 | 43.2 | 23.6 | 28.7 | 34.3 |
| <i>Qwen2.5-7B-chunk</i> | | | | | |
| Discriminative + SFT (Baseline) | 48.6 | 46.9 | 21.9 | 25.4 | 35.7 |
| STEPWISER (Generative Reasoning + RL) | 72.6 | 67.2 | 52.3 | 49.8 | 60.5 |
| – Ablate CoT (use Discriminative format + RL) | 58.7 | 49.4 | 40.8 | 42.7 | 47.9 |
| – Ablate Prompt Balancing (Generative Reasoning + RL) | 58.8 | 54.8 | 41.0 | 36.9 | 47.9 |

392
 393 Qwen2.5-1.5B-chunk, the RFT model achieves an average score of only 23.1, which is substantially
 394 lower than STEPWISER’s score of 36.2 and is even worse than the standard discriminative SFT
 395 baseline (24.1). From Figure 2, we notice that its training loss on a large, static dataset plateaus
 396 quickly. This trend is consistent across other learning signals and the larger 7B model, indicating that
 397 offline methods are insufficient to capture the complexity of CoT reasoning and reward modeling,
 398 making online RL a critical component.

399 **STEPWISER judge with CoT leverages intrinsic reasoning ability to obtain better evaluation** The
 400 benefit of the generative CoT format is illustrated by the “Ablate CoT” baseline. With the Qwen2.5-
 401 1.5B-chunk model, augmenting a discriminative-style judge with RL boosts the ProcessBench score
 402 from 24.1 (SFT) to 34.3 (RL), but it still falls short of the STEPWISER model’s 36.2. Moreover, the
 403 in-distribution accuracy results in Figure 5 show that the STEPWISER model with CoT reasoning
 404 achieves higher accuracy on the held-out data. This suggests that generating explicit rationales
 405 provides a more expressive and informative structure for learning and modeling the stepwise reward
 406 signal. The gap between the generative CoT model and the discriminative model becomes much
 407 larger with the stronger Qwen2.5-7B-chunk. Specifically, the generative STEPWISER model reaches
 408 an average score of 60.5, while the discriminative model only achieves 47.9. This is because we
 409 are leveraging the intrinsic reasoning ability of the base model through CoT in the judgment so the
 410 stronger model offers more advantages.

411 **Prompt dataset balancing stabilizes training and mitigates overfitting.** The practice of balancing
 412 the prompt dataset is also crucial for robust performance. Our ablation study on the Qwen2.5-7B-
 413 chunk model shows that removing this balancing step causes a substantial performance drop, with
 414 the average ProcessBench score dropping from 60.5 to 47.9. A deeper analysis reveals that while
 415 both the “Ablate CoT” ablation and the lack of dataset balancing hurt performance, their underlying
 416 failure modes are different. The “Ablate CoT” model suffers from a general decline in its ability to
 417 recognize correct and incorrect steps. In contrast, without balancing, the prompt dataset is heavily
 418 biased towards positive examples. This trains the model to be overly optimistic, developing a strong
 419 bias towards predicting any given step as correct. This bias is particularly enhanced during online
 420 training, which eventually leads to training instability and model collapse. A detailed analysis of this
 421 phenomenon is provided in the Appendix C.2.

422 4.4 USING THE STEPWISER JUDGE TO OBTAIN BETTER SOLUTIONS

424 In this section, we evaluate the practical utility of our RL-trained STEPWISER judge by using it to
 425 guide an LLM’s reasoning process at inference time. We employ a search strategy called *Chunk-Reset*
 426 *Reasoning*. The base policy model generates a solution “chunk-by-chunk”. After each chunk is
 427 produced, our STEPWISER judge evaluates it. If the chunk is deemed correct, it is accepted, and the
 428 model proceeds. If it is rejected, the flawed chunk is discarded, and the policy model re-generates
 429 a new one from the same point (up to 5 attempts). This allows the model to explore alternative
 430 reasoning paths without committing to an early mistake. This paradigm effectively scales sequential
 431 compute (i.e., compute used to extend a single trajectory) while keeping the final accepted token
 432 count similar.

432
 433 **Table 4: Inference time search via Chunk-Reset Reasoning.** We report results with both Qwen2.5-
 434 1.5B-chunk and Qwen2.5-7B-chunk, using them as both the response generators and the initialization
 435 checkpoints for the STEPWISER judge. We see clear improvements using STEPWISER across both
 436 model sizes, with similar accepted responses lengths (on MATH500). Rejected length is the number
 437 of tokens in removed chunks during inference time search.

| Method | Learning signal | MATH500 | NuminaMath Heldout-1K | Avg ↑ | Accepted length | Rejected length |
|---------------------------------|-----------------|-------------|-----------------------|-------------|-----------------|-----------------|
| <i>Qwen2.5-1.5B-chunk</i> | | | | | | |
| Discriminative + SFT | Abs-Q | 47.7 | 19.1 | 33.4 | 625.2 | 218.7 |
| Discriminative + SFT | Rel-Effective | 47.4 | 19.6 | 33.5 | 612.7 | 302.4 |
| Discriminative + SFT | Rel-Ratio | 50.4 | 20.0 | 35.2 | 596.0 | 475.8 |
| Generative CoT + RL (STEPWISER) | Abs-Q | 51.4 | 19.8 | 35.6 | 599.1 | 1069.2 |
| Generative CoT + RL (STEPWISER) | Rel-Effective | 52.1 | 21.2 | 36.7 | 602.0 | 947.4 |
| Generative CoT + RL (STEPWISER) | Rel-Ratio | 51.9 | 21.8 | 36.9 | 596.4 | 884.7 |
| <i>Qwen2.5-7B-chunk</i> | | | | | | |
| Discriminative + SFT | Abs-Q | 74.8 | 44.4 | 59.6 | 654.0 | 168.2 |
| Discriminative + SFT | Rel-Effective | 76.9 | 46.1 | 61.5 | 654.6 | 186.5 |
| Discriminative + SFT | Rel-Ratio | 76.7 | 45.8 | 61.3 | 641.4 | 219.7 |
| Generative CoT + RL (STEPWISER) | Abs-Q | 77.5 | 46.3 | 61.9 | 658.5 | 345.7 |
| Generative CoT + RL (STEPWISER) | Rel-Effective | 78.3 | 48.1 | 63.2 | 660.8 | 425.8 |
| Generative CoT + RL (STEPWISER) | Rel-Ratio | 79.0 | 47.5 | 63.3 | 653.0 | 295.4 |

452
 453
 454 **Inference-time search consistently improves performance.** As shown in Table 4, using our
 455 STEPWISER judge for guidance leads to superior outcomes. With the Rel-Ratio learning signal,
 456 our approach steers the 1.5B model to an average accuracy of 36.9%, a significant improvement over
 457 the 31.2% of the base model without guidance. We observe a clear trend of our STEPWISER judge
 458 being superior to the discriminative models across all learning signals, and this trend holds for the 7B
 459 model, demonstrating the scalability of our approach.

460 **Superior error detection enables effective self-correction.** The “Accepted Length” column in
 461 Table 4 shows that the final solutions are of a similar length to the baselines. However, the “Rejected
 462 Length” column, which tracks discarded chunks, is significantly higher when using STEPWISER. We
 463 interpret this as direct evidence of STEPWISER’s superior ability to identify incorrect or unproductive
 464 steps. This triggers the reset mechanism more effectively, forcing the model to discard flawed
 465 reasoning and find a better path, which is consistent with its higher accuracy on ProcessBench.

466 **Relative signals prove more effective for guidance.** The inference-time search results also reinforce
 467 a key finding of this paper: training signals that reward relative progress (Rel-Effective,
 468 Rel-Ratio) consistently yield better judges than a signal that only measures a step’s absolute
 469 quality (Abs-Q). For example, the Rel-Effective judge guides the 7B model to 64.3% accuracy,
 470 outperforming the Abs-Q judge (61.9%). This pattern, further confirmed by our data selection
 471 experiments in Appendix C.4, shows that modeling the dynamics of reasoning is a more effective
 472 strategy for training useful judges.

5 CONCLUSION

473
 474 Reasoning models that output internal thought tokens before a final response have proven to outper-
 475 form non-reasoning models. In this paper we have shown that further improvements can be found
 476 by making models ***reason about the reasoning decisions*** made within those internal thoughts. We
 477 provide a recipe to: (1) segment reasoning into chunks-of-thought; (2) assign rewards to chunks via
 478 relative outcomes of rollouts; and (3) train a judge model to reason about the quality of CoT chunks
 479 via reinforcement learning (RL).

480 Our stepwise generative judge STEPWISER is shown to be superior to existing methods on Process-
 481 Bench, to provide improved inference time search, and better training time rewards for building better
 482 response models. We show that both the use of reasoning during judgment, and training with RL in
 483 order to reason about reasoning, are important components to achieve this performance.

486 REFERENCES
487

488 Edward Beeching, Shengyi Costa Huang, Albert Jiang, Jia Li, Benjamin Lipkin, Zihan Qina, Kashif
489 Rasul, Ziju Shen, Roman Soletskyi, and Lewis Tunstall. Numinamath 7b cot. <https://huggingface.co/AI-MO/NuminaMath-7B-CoT>, 2024.

490

491 Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang,
492 Denghui Zhang, Tong Zhang, et al. Rm-r1: Reward modeling as reasoning. *arXiv preprint*
493 *arXiv:2505.02387*, 2025.

494

495 Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu
496 Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang,
497 Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning
498 Ding. Process reinforcement through implicit rewards, 2025. URL <https://arxiv.org/abs/2502.01456>.

499

500 DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu,
501 Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu,
502 Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao
503 Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan,
504 Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao,
505 Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding,
506 Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang
507 Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong,
508 Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao,
509 Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang,
510 Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang,
511 Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L.
512 Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang,
513 Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng
514 Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng
515 Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan
516 Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang,
517 Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen,
518 Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li,
519 Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang,
520 Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan,
521 Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia
522 He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong
523 Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha,
524 Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang,
525 Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li,
526 Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen
527 Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.
528 URL <https://arxiv.org/abs/2501.12948>.

529

530 Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao,
531 Jipeng Zhang, KaShun SHUM, and Tong Zhang. RAFT: Reward ranked finetuning for generative
532 foundation model alignment. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.
533 URL <https://openreview.net/forum?id=m7p507zb1Y>.

534

535 Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen
536 Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf.
537 *arXiv preprint arXiv:2405.07863*, 2024.

538

539 Tao He, Rongchuan Mu, Lizi Liao, Yixin Cao, Ming Liu, and Bing Qin. Good learners think their
540 thinking: Generative prm makes large reasoning model more efficient math learner. *arXiv preprint*
541 *arXiv:2507.23317*, 2025.

542

543 Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moon-
544 tae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *arXiv preprint*
545 *arXiv:2504.16828*, 2025.

540 Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan
 541 Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint*
 542 *arXiv:2305.20050*, 2023.

543 Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalmans, Mohammad Saleh, Peter J Liu, and Jialu Liu.
 544 Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*,
 545 2023.

546 Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun
 547 Zhu, Lei Meng, Jiao Sun, et al. Improve mathematical reasoning in language models by automated
 548 process supervision. *arXiv e-prints*, pp. arXiv–2406, 2024.

549 Meta. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI Blog*, 2024.
 550 <https://ai.meta.com/blog/meta-llama-3/>.

551 Rafael Rafailev, Joey Hejna, Ryan Park, and Chelsea Finn. From r to q*: Your language model is
 552 secretly a q-function. *arXiv preprint arXiv:2404.12358*, 2024.

553 Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal,
 554 Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated
 555 process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

556 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Y Wu,
 557 and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language
 558 models. *arXiv preprint arXiv:2402.03300*, 2024.

559 Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng,
 560 Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint*
 561 *arXiv: 2409.19256*, 2024.

562 Vaishnavi Shrivastava, Ahmed Awadallah, Vidhisha Balachandran, Shivam Garg, Harkirat Behl, and
 563 Dimitris Papailiopoulos. Sample more to think less: Group filtered policy optimization for concise
 564 reasoning. *arXiv preprint arXiv:2508.09726*, 2025.

565 Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. Dart-math: Difficulty-aware
 566 rejection tuning for mathematical problem-solving. *arXiv preprint arXiv:2407.13690*, 2024.

567 Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang
 568 Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *CoRR*,
 569 *abs/2312.08935*, 2023.

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

573 Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep
 574 Saha. J1: Incentivizing thinking in llm-as-a-judge via reinforcement learning. *arXiv preprint*
 575 *arXiv:2505.10320*, 2025.

576 Wei Xiong, Hanze Dong, Chenlu Ye, Han Zhong, Nan Jiang, and Tong Zhang. Gibbs sampling from
 577 human feedback: A provable kl-constrained framework for rlhf. *arXiv preprint arXiv:2312.11456*,
 578 2023.

579 Wei Xiong, Chengshuai Shi, Jiaming Shen, Aviv Rosenberg, Zhen Qin, Daniele Calandriello, Misha
 580 Khalmans, Rishabh Joshi, Bilal Piot, Mohammad Saleh, et al. Building math agents with multi-turn
 581 iterative preference learning. *arXiv preprint arXiv:2409.02392*, 2024a.

582 Wei Xiong, Hanning Zhang, Nan Jiang, and Tong Zhang. An implementation of generative prm,
 583 2024b.

584 Wei Xiong, Jiarui Yao, Yuhui Xu, Bo Pang, Lei Wang, Doyen Sahoo, Junnan Li, Nan Jiang, Tong
 585 Zhang, Caiming Xiong, et al. A minimalist approach to llm reasoning: from rejection sampling to
 586 reinforce. *arXiv preprint arXiv:2504.11343*, 2025.

594 Jing Xu, Andrew Lee, Sainbayar Sukhbaatar, and Jason Weston. Some things are more cringe than
 595 others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682*,
 596 2023.

597 Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Sim-
 598 pletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint*
 599 *arXiv:2509.02479*, 2025.

600

601 An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li,
 602 Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint*
 603 *arXiv:2412.15115*, 2024.

604 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao.
 605 React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*,
 606 2022.

607

608 Chenlu Ye, Zhou Yu, Ziji Zhang, Hao Chen, Narayanan Sadagopan, Jing Huang, Tong Zhang, and
 609 Anurag Beniwal. Beyond correctness: Harmonizing process and outcome rewards through rl
 610 training. *arXiv preprint arXiv:2509.03403*, 2025.

611 Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong
 612 Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale.
 613 *arXiv preprint arXiv:2503.14476*, 2025.

614

615 Kaiwen Zha, Zhengqi Gao, Maohao Shen, Zhang-Wei Hong, Duane S Boning, and Dina Katabi.
 616 Rl tango: Reinforcing generator and verifier together for language reasoning. *arXiv preprint*
 617 *arXiv:2505.15034*, 2025.

618 Hanning Zhang, Pengcheng Wang, Shizhe Diao, Yong Lin, Rui Pan, Hanze Dong, Dylan Zhang,
 619 Pavlo Molchanov, and Tong Zhang. Entropy-regularized process reward model, 2024a. URL
 620 <https://arxiv.org/abs/2412.11006>.

621

622 Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal.
 623 Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*,
 624 2024b.

625

626 Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu,
 627 Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical
 628 reasoning. *arXiv preprint arXiv:2501.07301*, 2025.

629

630 Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian,
 631 Biqing Qi, Xiu Li, et al. Genprm: Scaling test-time compute of process reward models via
 632 generative reasoning. *arXiv preprint arXiv:2504.00891*, 2025.

633

634 Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jin-
 635 gren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning.
 636 *arXiv preprint arXiv:2412.06559*, 2024.

637

638 Han Zhong, Guhao Feng, Wei Xiong, Li Zhao, Di He, Jiang Bian, and Liwei Wang. Dpo meets ppo:
 639 Reinforced token optimization for rlhf. *arXiv preprint arXiv:2404.18922*, 2024.

640

641 Yifei Zhou, Song Jiang, Yuandong Tian, Jason Weston, Sergey Levine, Sainbayar Sukhbaatar, and
 642 Xian Li. Sweet-rl: Training multi-turn llm agents on collaborative reasoning tasks. *arXiv preprint*
 643 *arXiv:2503.15478*, 2025.

644

645 **A LLM USAGE STATEMENT**

646

647 In the final stages of preparing this manuscript, we utilized a large language model (LLM) as a
 648 writing assistant. The scope of its use was strictly limited to proofreading and refining the grammar,
 649 and clarity of the text after all scientific work, experiments, and the initial draft were completed
 650 by the authors. All authors reviewed, edited, and approved the LLM’s suggestions and take full
 651 responsibility for the final content of this paper.

648

649

Table 5: Rules that we provide for an LLM to create segmented Chunks-of-Thought SFT data.

650

651

652

Rules for CoT Trajectory Segmentation

653

Segmentation Principles

654

1. Unified purpose: A chunk should serve a single, clear objective. For example: setting up an initial equation, executing a self-contained calculation (like integration by parts), or stating a final/intermediate conclusion. All content within the chunk must directly serve this one core goal.
2. Logical Cohesion: All lines within a chunk must form a continuous and uninterrupted logical flow. A new chunk should begin as soon as the focus or purpose of the reasoning shifts.
3. Clear Transition: A new chunk must begin when the problem-solving process enters a new phase. This includes transitioning from "solving for a variable" to "verifying the answer," or inserting an "explanatory side-note" into the main workflow.

655

Format rules.

656

1. Use `<chunk>... </chunk>` to mark the beginning and end of each segment. The text and newlines inside the tags must not be altered.
2. The final output should only contain the tagged content, without any additional text, titles, or blank lines.
3. You **must** preserve all original text and newlines exactly as they appear within the tags.

657

658

659

660

Table 6: Comparison of the base policy with and without self-segmentation fine-tuning. Overall performance is comparable, but self-segmentation results in less chunks than using split by `\n\n`. Here Avg@32 is the test accuracy averaged over 32 trajectories with random seeds.

661

662

663

664

665

666

667

668

669

670

671

672

| Generator | Method | # Steps | # Tokens | Avg@32 on MATH500 |
|--------------------|----------------------------|---------|----------|-------------------|
| Qwen2.5-1.5B-it | Split by <code>\n\n</code> | 9.6 | 686.7 | 44.2 |
| Qwen2.5-1.5B-chunk | Self-segmentation | 6.0 | 714.1 | 44.7 |
| Qwen2.5-7B-it | Split by <code>\n\n</code> | 9.9 | 733.0 | 73.3 |
| Qwen2.5-7B-chunk | Self-segmentation | 6.8 | 768.1 | 73.3 |

673

674

675

676

677

678

679

B EXPERIMENT SETUPS AND IMPLEMENTATION DETAILS

680

681

B.1 BASE MODELS AND DATA PREPROCESSING.

682

683

The base models used in our experiments are `Qwen2.5-1.5B-it` and `Qwen2.5-7B-it` (Yang et al., 2024), both featuring a context window of 8192 tokens. Our training data is sourced from the NuminaMath-CoT dataset (Beeching et al., 2024). Ground-truth verification for mathematical problems is performed using the Math-Verify tool. The preprocessing pipeline for the training data is as follows:

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

- **Deduplication:** Duplicate prompts are removed from the dataset.
- **Verification and Filtering:** We use Math-Verify to extract and score the final answer from each reference solution. Prompts where the answer cannot be reliably verified are discarded.

Unless stated otherwise, the same base model is used to initialize both the policy and the judge.

B.2 IMPLEMENTATION DETAILS OF SELF-SEGMENTATION FINE-TUNING

To enable our models to structure their own reasoning, we first established a set of principles for segmenting CoT trajectories into meaningful "chunks". These rules, detailed in Table 5, are designed to guide a powerful teacher model in creating a high-quality dataset for subsequent self-segmentation fine-tuning.

We then used these rules to generate the self-segmentation dataset via the following multi-step pipeline:

- 702 • **Initial Generation:** From a random subset of 20k NuminaMath-CoT prompts, we generate
703 16 responses per prompt using the base policy model.
- 704 • **Solution Filtering:** We discard incorrect responses, keeping a maximum of 4 correct
705 solutions for each prompt.
- 706 • **Segmentation by Teacher Model:** We prompt a powerful teacher model,
707 Llama-3.1-70B-bit (Meta, 2024), to segment the correct solutions into meaningful
708 chunks based on the rules outlined in Table 5.
- 709 • **Segmentation Filtering:** For each solution, we generate 8 segmented versions and retain
710 only those that perfectly reconstruct the original text and adhere to the specified format.

712 The base model is then fine-tuned on this curated dataset. Fine-tuning is performed using the Axolotl
713 package with the following hyperparameters: a learning rate of $1e-5$, a packing block size of 8192,
714 and a global batch size of 32. The prompt template is provided in Table 10. This process successfully
715 teaches the model to generate more structured reasoning, significantly reducing the number of steps
716 compared to naive splitting methods, as detailed in Table 6. Meanwhile, we observe that for most
717 current open-source thinking models that do long reasoning before answering, the number of steps
718 exceeds 150 when trajectories are segmented using $\backslash n\backslash n$, with each step containing only about 30
719 tokens. While a broader application of our technique is beyond the scope of this work due to resource
720 constraints, we believe it holds particular promise for these long-reasoning scenarios, which we leave
721 for future exploration.

722 B.3 DETAILS OF STEPWISE DATA ANNOTATION

724 To automatically generate supervisory signals for each reasoning step, we adopt a framework based
725 on Monte Carlo estimation of Q-values. In Section 3.2, we described our high-level approach for
726 data annotation. Here, we provide the detailed mathematical formulations.

728 B.3.1 STEPWISE DATA ANNOTATION VIA Q VALUE ESTIMATION

730 B.3.2 IMPLEMENTATION DETAILS OF STEPWISE DATA ANNOTATION

731 We select a subset of 40k prompts from NuminaMath for stepwise data annotation based on a pre-
732 filtering process using the `pass@k` metric. Specifically, for each prompt, we generate 16 responses
733 using our chunk-tuned models (e.g., Qwen2.5-1.5B-chunk). To ensure the selected prompts are of a
734 suitable difficulty, we filter out prompts where the responses were either all correct or all incorrect.
735 During generation, we use a temperature of 1.0 and set the maximum token limit to 8192, or until the
736 model produced a final answer. Then, for each intermediate step in a solution, we sample another
737 $M = 16$ completions starting from that step for estimating Q-values.

738 While we follow the well-established annotation framework from prior literature (Wang et al., 2023;
739 Xiong et al., 2024b), we note that advanced techniques like model ensembles or human verification
740 could further enhance label quality (Zhang et al., 2025). These engineering improvements are
741 orthogonal to our core investigation and could be integrated in future work. The full annotation
742 process is rather computationally intensive, taking approximately 14 days on 8 A100 GPUs for the
743 Qwen2.5-7B-chunk model. Notably, the self-segmentation fine-tuning described previously plays a
744 crucial role here, as it significantly reduces the total number of chunks per trajectory, thereby saving
745 substantial compute and annotation time.

746 B.4 JUDGE MODEL TRAINING DETAILS

748 We train and compare two types of stepwise judges: a discriminative judge trained via SFT, which
749 serves as a strong baseline, and our proposed generative judge trained with RL, which learns to
750 produce CoT reasoning before its final decision.

752 B.4.1 PRELIMINARIES: HYPERPARAMETER SEARCH FOR LABELING SIGNALS

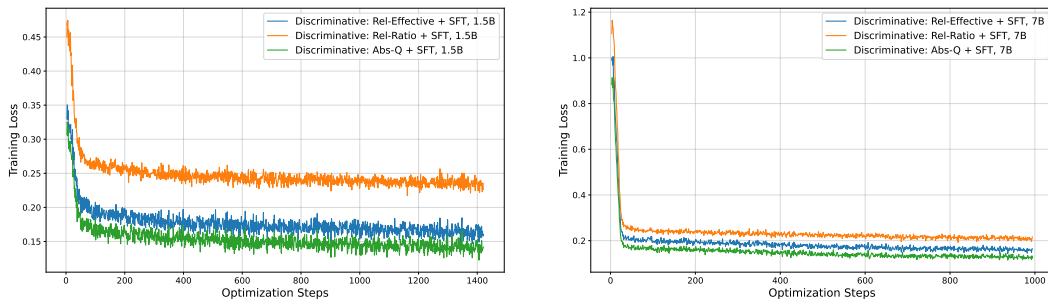
754 We conduct hyperparameter tuning for the learning signals labeling. We mainly search by training
755 discriminative models and SFT training, as this is more computationally efficient than full RL
training. For `Rel-Ratio`, we search over $\gamma \in \{0.6, 0.7, 0.8, 1.0, 1.2\}$, and for `Rel-Effective`,

756 we search over $\alpha \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ with μ set as the best-of-4 policy induced by the base
 757 policy. For Qwen2.5-1.5B-chunk, we choose $\gamma = 0.8$ and $\alpha = 0.4$, while for Qwen2.5-7B-chunk,
 758 we use $\gamma = 0.7$ and $\alpha = 0.8$.
 759

760 B.4.2 BASELINE: DISCRIMINATIVE JUDGE VIA SFT

762 We follow Xiong et al. (2024b) to formulate the discriminative stepwise judge as a multi-turn
 763 conversation task. Specifically, in every user turn, we provide a single step of reasoning, while in the
 764 next assistant turn, the model will decode either “+” or “-” token to indicate its judgment.
 765

766 For training, we use standard SFT code. The data is packed into a block with length 8192 tokens. We
 767 use a learning rate of $1e-5$, a global batch size of 32. We also mask out the user turn’s loss. We
 768 present the representative training loss curves in Figure 3.
 769



779 Figure 3: The training loss curves of discriminative stepwise judge under different learning signals.
 780 Left: 1.5B model, Right: 7B model.
 781

782 B.4.3 OUR METHOD: GENERATIVE JUDGE VIA RL

784 We train the judge model using the GRPO algorithm (Shao et al., 2024), implemented with the verl
 785 library (Sheng et al., 2024).
 786

- 787 • **Hyperparameters:** We use a learning rate of $1e-6$, a per-prompt batch size of 1024, and a
 788 gradient update mini-batch size of 256. The judge generates 4 responses per prompt. The
 789 maximum prompt length is set to 3096 tokens, and the model can generate up to 3096 new
 790 tokens.
- 791 • **Training Stability:** We identified and addressed two primary stability issues. First, to
 792 counteract rapid entropy decay and the resulting zero-gradient problem, we employed the
 793 **clip higher** technique (Yu et al., 2025) with $\epsilon_h = 0.28$ and $\epsilon_l = 0.2$. Second, to mitigate
 794 bias from imbalanced labels, we created a **balanced training set** by down-sampling the
 795 majority class. We also apply a heuristic filtering process to remove prompts that were
 796 overly short or excessively long.

797 The RL training for the 7B model took approximately 5 days on 8 A100 GPUs. Figure 4 illustrates
 798 the training dynamics. The model is Qwen2.5-7B-chunk and the learning signal is Rel-Ratio
 799 with threshold 0.7. We can see that clip higher helps to encourage exploration and leads to a higher
 800 training curve.
 801

802 C ADDITIONAL RESULTS AND ABLATION STUDIES

803 C.1 ABLATION ON SELF-SEGMENTATION

805 To validate the effectiveness of our self-segmentation approach, we compare its performance against
 806 a naive baseline that splits reasoning trajectories by $\backslash n \n$. The results, presented in Table 7, show
 807 that the benefits of self-segmentation are most apparent in the context of RL training.
 808

809 While the average scores for SFT models are comparable (e.g., 27.5 vs. 27.2 for Abs-Q), models
 810 trained with RL show significant improvements. Specifically, the average score for Abs-Q + RL

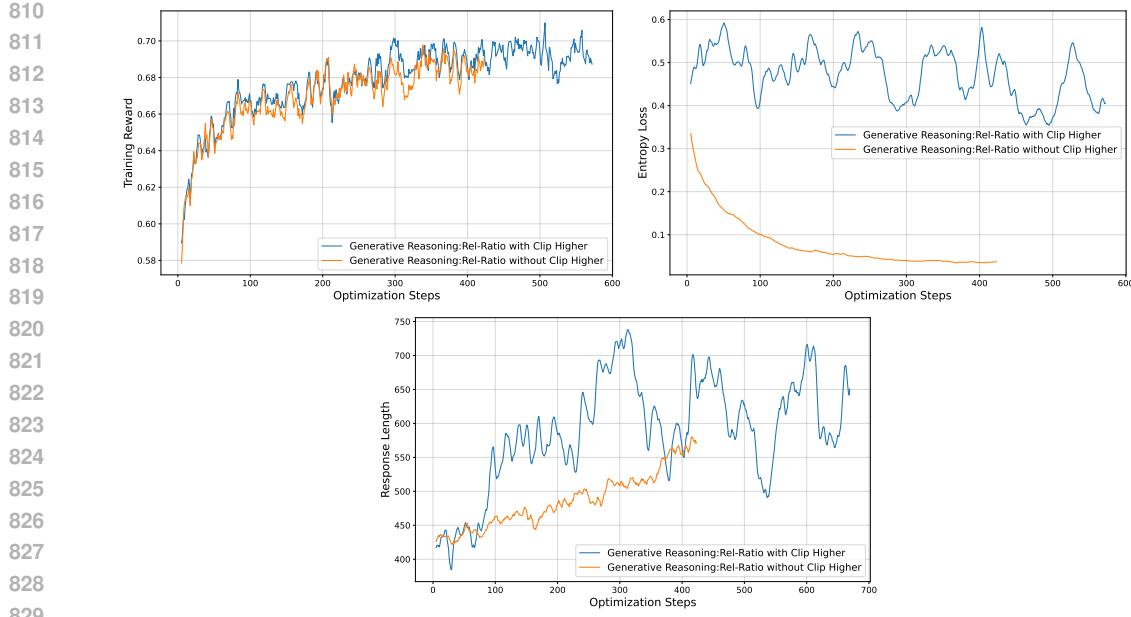


Figure 4: A representative example of training reward, entropy loss, and response length with and without clip higher technique. The model is Qwen2.5-7B-chunk and the learning signal is Rel-Ratio with threshold 0.7.

Table 7: The main ablation results on self-segmentation fine-tuning an chunking.

| Method | Learning signal | # Steps | GSM8K | MATH | Olympiad | Omni-MATH | Ave |
|-------------------|-----------------|---------|-------|------|----------|-----------|------|
| Split by \n\n | Abs-Q + SFT | 5457820 | 33.7 | 37.1 | 20.2 | 18.9 | 27.5 |
| Split by \n\n | Abs-Q + RL | - | 46.3 | 38.4 | 19.0 | 25.8 | 32.4 |
| Split by \n\n | Rel-Ratio + SFT | - | 28.3 | 30.9 | 15.5 | 21.0 | 23.9 |
| Split by \n\n | Rel-Ratio + RL | - | 46.3 | 39.1 | 17.3 | 21.3 | 31.0 |
| Self-segmentation | Abs-Q + SFT | 3463520 | 39.3 | 32.1 | 19.3 | 18.9 | 27.2 |
| Self-segmentation | Abs-Q + RL | - | 49.2 | 40.5 | 23.8 | 31.0 | 36.1 |
| Self-segmentation | Rel-Ratio + SFT | - | 32.1 | 32.0 | 14.2 | 18.0 | 24.1 |
| Self-segmentation | Rel-Ratio + RL | - | 46.9 | 43.4 | 26.3 | 28.4 | 36.2 |

increased from 32.4 to 36.1, and for Rel-Ratio + RL from 31.0 to 36.2. This disparity suggests that the self-segmentation process produces a cleaner, more meaningful step-wise signal by filtering out noisy or trivial intermediate steps. In particular, we refer the interested readers to Table 11-13 for detailed examples. The RL process, being more sensitive to data and reward quality, benefits greatly from this higher-quality signal. Conversely, SFT appears more robust to this type of noise, and thus its performance is less impacted.

C.2 ABLATION ON KEY COMPONENTS OF THE GENERATIVE JUDGE: CoT AND PROMPT DATASET BALANCING

To understand the contributions of the core components of our generative judge, we conduct two key ablation studies presented in Table 9: (1) removing the generative CoT reasoning and (2) disabling the dataset balancing mechanism. The results reveal that while the absence of either component degrades the final F1 score, the underlying reasons for the performance drop are fundamentally different.

The Impact of CoT Reasoning In this ablation, we train the judge to directly output a final judgment without generating any intermediate reasoning steps. We apply a format penalty of -1.0 if the model fails to follow this instruction. The results show that removing CoT weakens the model’s

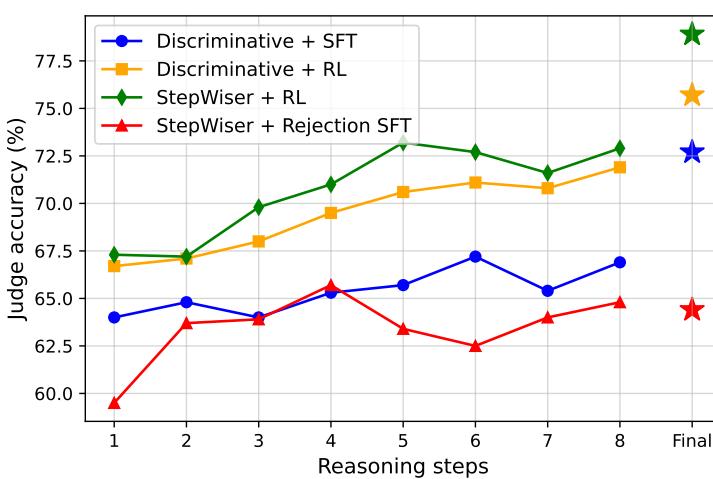


Figure 5: STEPWISER ablation results. **Left:** Test stepwise accuracy of various stepwise judge setups. Both generative CoT and RL training are important for the best stepwise judge. Here we plot the results of Rel-Ratio using Qwen2.5-1.5B-chunk, other results are presented in Figure 6.

overall ability to discriminate between correct and incorrect reasoning. This is evidenced by a general decline in accuracy for both “Correct” and “Error” classifications, suggesting that the act of generating a rationale is integral to the judge’s reasoning capability.

The Impact of Dataset Balancing We also examine the effect of training on the original, imbalanced data, where positive labels can be overrepresented (e.g., 70.2% for Abs-Q). Removing the balancing mechanism introduces a strong class bias. The model learns to over-predict the majority class (“Correct”), leading to a sharp increase in accuracy for correct steps but a catastrophic drop in its ability to detect errors. This trade-off is ultimately detrimental, as a judge that cannot identify mistakes is of little practical use, which is reflected in the significant F1 score degradation.

C.3 ADDITIONAL RESULT ON CLASSIFICATION ACCURACY

Figure 5 and 6 compares the stepwise classification accuracy of our generative judge (STEPWISER) against the discriminative baseline across the different learning signals.

For the relative signals (Rel-Ratio and Rel-Effective), our RL-trained generative judge achieves significantly higher test accuracy on both intermediate steps and final answer evaluation. This suggests that the process of generating CoT reasoning provides the model with greater expressive capacity, enabling it to better capture these nuanced, dynamic signals.

In contrast, the performance gap narrows for the Abs-Q signal. We attribute this to a data distribution shift: the original Abs-Q dataset is highly imbalanced (70.2% positive samples), and the necessary downsampling to stabilize RL training adversely affects the judge’s performance on the original, imbalanced test set. Nevertheless, even under these conditions, our generative judge remains substantially more accurate at the crucial task of verifying the final answer’s correctness.

C.4 APPLICATION: DATA SELECTION VIA REJECTION SAMPLING FINE-TUNING

While PRMs offer more fine-grained supervision, directly using their scores as a reward signal for reinforcement learning can be challenging. These signals are often less reliable than final outcome verification. While this process-level reward usually can improve sample efficiency, it has not consistently resulted in better final performance compared to well-tuned policies trained with outcome-based verifiable feedback.

In contrast, a more robust application for process-level feedback is emerging in data selection. Recent studies consistently demonstrate that using detailed feedback to filter training data is a highly effective

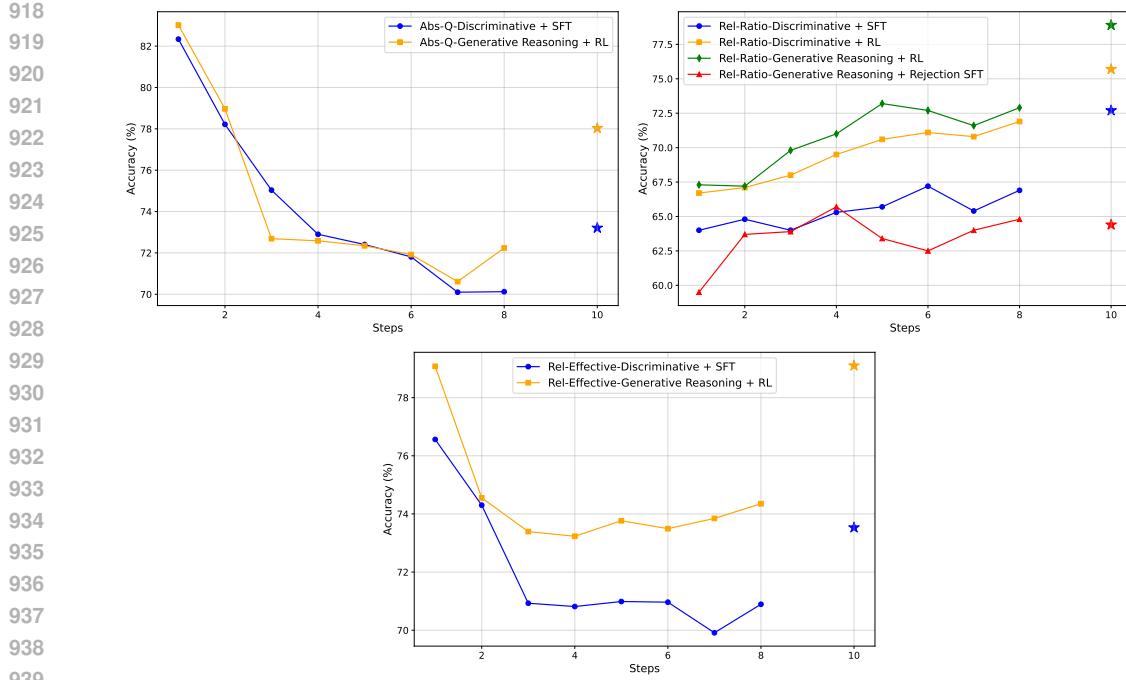


Figure 6: The test stepwise accuracy of different stepwise judges. From left to right, we plot the results of Abs-Q, Rel-Ratio, Rel-Effective, respectively. The stars at step 10 represent the accuracy of recognizing the final answer.

Table 8: **Data selection via Stepwise Rejection Sampling Fine-Tuning.** Our STEPWISER judge trained with RL provides better quality training data, as measured by final average test performance. The evaluation is with greedy decoding and a maximal generated length of 8192.

| Method | Learning signal | MATH500 | NM-Heldout-1K | Average ↑ |
|----------------------------------|-----------------|-------------|---------------|-------------|
| <i>Qwen2.5-7B-chunk (greedy)</i> | - | 75.6 | 44.6 | 60.1 |
| Outcome-based selection | - | 76.6 | 45.2 | 60.9 |
| Discriminative + SFT | Abs-Q | 78.4 | 45.3 | 61.8 |
| Discriminative + SFT | Rel-Effective | 78.2 | 45.2 | 61.7 |
| Discriminative + SFT | Rel-Ratio | 78.2 | 45.7 | 61.9 |
| Generative CoT + RL (STEPWISER) | Abs-Q | 79.0 | 46.1 | 62.5 |
| Generative CoT + RL (STEPWISER) | Rel-Effective | 79.4 | 46.7 | 63.0 |
| Generative CoT + RL (STEPWISER) | Rel-Ratio | 79.0 | 46.8 | 62.9 |

strategy, both for offline data curation (Tong et al., 2024) and online RL training (Xiong et al., 2025; Ye et al., 2025; Shrivastava et al., 2025; Xue et al., 2025). Notably, Ye et al. (2025) show that employing PRMs to select the best trajectories from a candidate pool robustly enhances final model performance.

Motivated by these findings, we evaluate our stepwise judge in a practical data selection application: Rejection Sampling Fine-tuning (RFT) (Dong et al., 2023). The goal of RFT is to improve a base policy by fine-tuning it on its own best-generated outputs. While standard RFT might select trajectories based only on final answer correctness, this coarse signal cannot differentiate between multiple valid reasoning paths that lead to the same answer.

We leverage our generative judge (STEPWISER) to provide a more fine-grained selection criterion. Specifically, from a pool of trajectories that all reach the correct final answer, we use the judge to score each individual reasoning chunk. The trajectory with the highest average chunk score is then selected as the highest-quality reasoning trace for the RFT dataset.

972

973
974
975
Table 9: Judge performance on ProcessBench, broken down by four subsets. Each subset reports Error (%),
Correct (%), and F1 score (%). The final column is the average F1 across all subsets. We remark that the F1
score here is indeed the harmonic mean of the accuracies on two classes.

| Method | Learning Signal | GSM8K | | | MATH | | | Olympiad | | | Omni-MATH | | | Avg. F1 |
|----------------------------------|-----------------|-------|---------|------|-------|---------|------|----------|---------|-------|-----------|---------|------|---------|
| | | Error | Correct | F1 | Error | Correct | F1 | Error | Correct | F1 | Error | Correct | F1 | |
| <i>Qwen2.5-1.5B-chunk</i> | | | | | | | | | | | | | | |
| Discriminative + SFT | Abs-Q | 26.0 | 80.0 | 39.3 | 22.2 | 57.6 | 32.1 | 14.2 | 30.2 | 19.3 | 13.2 | 28.2 | 18.0 | 27.2 |
| Discriminative + SFT | Rel-Effect | 28.5 | 72.0 | 40.8 | 28.6 | 53.0 | 37.2 | 16.4 | 21.8 | 18.7 | 15.8 | 27.6 | 20.1 | 29.2 |
| Discriminative + SFT | Rel-Ratio | 22.5 | 56.0 | 32.1 | 26.2 | 41.0 | 32.0 | 14.0 | 14.4 | 14.2 | 15.2 | 22.0 | 18.0 | 24.1 |
| Generative + CoT + RL | Abs-Q | 42.5 | 58.5 | 49.2 | 36.4 | 45.6 | 40.5 | 31.4 | 19.2 | 23.8 | 32.8 | 29.4 | 31.0 | 36.1 |
| Generative + CoT + RL | Rel-Effect | 38.5 | 64.5 | 48.2 | 37.8 | 51.6 | 43.6 | 23.2 | 21.0 | 22.1 | 24.0 | 26.8 | 25.3 | 34.8 |
| Generative + CoT + RL | Rel-Ratio | 35.0 | 71.0 | 46.9 | 37.8 | 50.8 | 43.4 | 27.0 | 25.6 | 26.3 | 28.0 | 28.8 | 28.4 | 36.2 |
| Gen + RL (no CoT) | Rel-Ratio | 28.5 | 79.5 | 42.0 | 37.0 | 51.8 | 43.2 | 24.4 | 22.8 | 23.6 | 28.6 | 28.8 | 28.7 | 34.3 |
| Gen + CoT + RL (no Chunk) | Rel-Ratio | 36.0 | 65.5 | 46.5 | 37.0 | 39.8 | 38.4 | 25.4 | 15.2 | 19.0 | 29.4 | 23.0 | 25.8 | 32.4 |
| <i>Qwen2.5-7B-chunk</i> | | | | | | | | | | | | | | |
| Discriminative + SFT | Abs-Q | 41.0 | 80.5 | 54.3 | 36.0 | 66.4 | 46.7 | 28.8 | 43.4 | 34.6 | 21.8 | 39.6 | 28.1 | 40.9 |
| Discriminative + SFT | Rel-Effect | 40.5 | 80.0 | 53.8 | 36.8 | 69.6 | 48.1 | 27.0 | 36.2 | 30.93 | 24.6 | 41.0 | 30.8 | 38.7 |
| Discriminative + SFT | Rel-Ratio | 37.5 | 78.5 | 50.8 | 36.6 | 63.8 | 46.5 | 24.0 | 35.8 | 28.7 | 24.2 | 35.6 | 28.8 | 38.7 |
| Generative + CoT + RL | Abs-Q | 59.5 | 64.5 | 61.9 | 63.2 | 59.0 | 61.0 | 53.0 | 44.6 | 48.4 | 44.4 | 43.4 | 43.9 | 53.8 |
| Generative + CoT + RL | Rel-Effect | 70.5 | 74.5 | 72.4 | 69.2 | 67.4 | 68.3 | 61.4 | 48.8 | 54.4 | 54.4 | 50.6 | 52.4 | 61.9 |
| Generative + CoT + RL | Rel-Ratio | 66.5 | 80.0 | 72.6 | 62.6 | 72.6 | 67.2 | 57.2 | 48.2 | 52.3 | 49.4 | 50.2 | 49.8 | 60.5 |
| <i>Qwen2.5-7B-chunk Ablation</i> | | | | | | | | | | | | | | |
| Gen + CoT + RL (no Balancing) | Abs-Q | 31.5 | 94.0 | 47.2 | 34.0 | 79.6 | 47.7 | 25.0 | 58.0 | 34.9 | 23.2 | 43.2 | 30.2 | 40.0 |
| Gen + CoT + RL (no Balancing) | Rel-Effect | 45.0 | 94.0 | 60.9 | 44.8 | 79.0 | 57.2 | 35.8 | 59.2 | 44.6 | 27.0 | 48.8 | 34.8 | 49.4 |
| Gen + CoT + RL (no Balancing) | Rel-Ratio | 42.5 | 95.5 | 58.8 | 41.6 | 80.2 | 54.8 | 29.8 | 65.8 | 41.0 | 29.4 | 49.6 | 36.9 | 47.9 |
| Gen + RL (no CoT) | Rel-Ratio | 45.5 | 82.5 | 58.7 | 37.6 | 72.0 | 49.4 | 36.0 | 47.0 | 40.8 | 40.2 | 45.6 | 42.7 | 47.9 |

993
994
Table 10: Prompt template for solver model to produce self-segmented CoTs.

| Prompt Template for Policy Model | | | | | | | | | | | | | |
|---|--|--|--|--|--|--|--|--|--|--|--|--|--|
| You are a helpful assistant designed to solve mathematical problems step-by-step. Your task is to think step-by-step and provide a detailed solution process following a specific format. | | | | | | | | | | | | | |
| You MUST follow the following format: | | | | | | | | | | | | | |
| 1. Enclose each part of your step-by-step reasoning within <chunk> and </chunk> tags. | | | | | | | | | | | | | |
| 2. After completing the chain-of-thought reasoning, provide the final answer within \boxed{ }. | | | | | | | | | | | | | |
| Ensure strict compliance with this format for every response. | | | | | | | | | | | | | |
| Problem: {problem} | | | | | | | | | | | | | |

1008
1009
1010
1011
1012
1013
The results of this approach, presented in Table 8, confirm the benefits of our fine-grained selection
1008 signal. Models fine-tuned on data selected by our generative judge achieve the highest average
1009 performance (63.0%). This surpasses not only the original base model (60.1%) and simple outcome-
1010 based selection (60.9%), but also data selected by the discriminative baseline judge (61.9%). This
1011 result strongly demonstrates that our judge’s nuanced, step-level evaluations provide a superior signal
1012 for identifying high-quality reasoning data for model training.1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025

D TEMPLATE, EXAMPLE, AND ADDITIONAL TABLES

1026
1027
1028
1029
1030
1031
1032
1033

1034 Table 11: A representative example illustrating CoT segmentation methods based on a response from
1035 Qwen2.5-1.5B-it. Standard splitting by newlines (left) produces too many chunks without a clear
1036 logical step. For example, chunk 3 and chunk 4 are logically a whole, but they are separated, resulting
1037 in poor independence of each step and the inability to form a complete thinking unit.

| Split by \n\n | LLM-based self-segmentation (Chunks-of-Thought) |
|---|---|
| chunk 1 To find the value of $f(5)$ for the given function | |
| $f(x) = \begin{cases} x - 3, & x \geq 6 \\ f(f(x + 5)), & x < 6 \end{cases}$ we need to follow the recursive definition of the function for $x < 6$. | chunk 1 (Initial Expansion) To find the value of $f(5)$ for the given function |
| $f(5) = f(f(5 + 5)) = f(f(10)).$ | $f(x) = \begin{cases} x - 3, & x \geq 6 \\ f(f(x + 5)), & x < 6 \end{cases}$ we need to follow the recursive definition of the function for $x < 6$. |
| chunk 2 Since $5 < 6$, we use the second case of the piecewise function: | chunk 2 (Calculate Inner Function and Substitute) Since $5 < 6$, we use the second case of the piecewise function: |
| $f(5) = f(f(5 + 5)) = f(f(10)).$ | $f(5) = f(f(5 + 5)) = f(f(10)).$ |
| chunk 3 Now, we need to evaluate $f(10)$. Since $10 \geq 6$, we use the first case of the piecewise function: | chunk 2 (Calculate Inner Function and Substitute) Now, we need to evaluate $f(10)$. Since $10 \geq 6$, we use the first case of the piecewise function: |
| $f(10) = 10 - 3 = 7.$ | $f(10) = 10 - 3 = 7.$ |
| chunk 4 So, we have: | So, we have: |
| $f(5) = f(7).$ | $f(5) = f(7).$ |
| chunk 5 Next, we need to evaluate $f(7)$. Again, since $7 \geq 6$, we use the first case of the piecewise function: | chunk 3 (Final Calculation) Next, we need to evaluate $f(7)$. Again, since $7 \geq 6$, we use the first case of the piecewise function: |
| $f(7) = 7 - 3 = 4.$ | $f(7) = 7 - 3 = 4.$ |
| chunk 6 Thus, we have: | Thus, we have: |
| $f(5) = 4.$ | $f(5) = 4.$ |
| chunk 7 The final answer is: | The final answer is: |
| $\boxed{4}$ | $\boxed{4}$ |

1071
1072
1073
1074
1075
1076
1077
1078
1079

1080
1081
1082
1083
10841085 Table 12: A representative example illustrating CoT segmentation methods based on a response from
1086 Qwen2.5-1.5B-it. Standard splitting by newlines (left) produces too many chunks without a clear
1087 logical step.

1088

1089

1090

1091

1092

1093

1094

1095

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

1115

1116

1117

1118

1119

1120

1121

1122

1123

1124

1125

1126

1127

1128

1129

1130

1131

1132

1133

| Split by \n\n | LLM-based self-segmentation (Chunks-of-Thought) |
|---|--|
| chunk 1 To solve the given problem, we start by expressing z in its standard form $z = x + yi$, where x and y are real numbers. The given equation is: | chunk 1 (Express the Equation in Terms of Real and Imaginary Parts) To solve the given problem, we start by expressing z in its standard form $z = x + yi$, where x and y are real numbers. The given equation is: $\left \frac{z^2 + 1}{z + i} \right + \left \frac{z^2 + 4i - 3}{z - i + 2} \right = 4.$ |
| chunk 2 $\left \frac{z^2 + 1}{z + i} \right + \left \frac{z^2 + 4i - 3}{z - i + 2} \right = 4.$ | chunk 2 (Simplify the first Term) First, let's express the magnitudes of the complex numbers. For the first term, we have: $\left \frac{z^2 + 1}{z + i} \right = \left \frac{(x + yi)^2 + 1}{x + yi + i} \right .$ |
| chunk 3 First, let's express the magnitudes of the complex numbers. For the first term, we have: | Since $(x + yi)^2 = x^2 - y^2 + 2xyi$, we get $z^2 + 1 = x^2 - y^2 + 2xyi + 1 = (x^2 - y^2 + 1) + 2xyi$. Therefore, the magnitude is: $\left (x^2 - y^2 + 1) + 2xyi + i(x + yi) \right = \left (x^2 - y^2 + 1 + x + y) + (2xy + x + y)i \right .$ |
| chunk 4 $\left \frac{z^2 + 1}{z + i} \right = \left \frac{(x + yi)^2 + 1}{x + yi + i} \right .$ | This magnitude can be simplified to the square root of the sum of the squares of the real and imaginary parts: $\sqrt{(x^2 - y^2 + 1 + x + y)^2 + (2xy + x + y)^2}.$ |
| chunk 5 Since $(x + yi)^2 = x^2 - y^2 + 2xyi$, we get $z^2 + 1 = x^2 - y^2 + 2xyi + 1 = (x^2 - y^2 + 1) + 2xyi$. Therefore, the magnitude is: | chunk 3 (Simplify the Second Term and Formulate Equations) Similarly, for the second term, we have: $\left \frac{z^2 + 4i - 3}{z - i + 2} \right = \left \frac{(x + yi)^2 + 4i - 3}{x + yi - i + 2} \right .$ |
| chunk 6 $\left (x^2 - y^2 + 1) + 2xyi + i(x + yi) \right = \left (x^2 - y^2 + 1 + x + y) + (2xy + x + y)i \right .$ | Since $(x + yi)^2 = x^2 - y^2 + 2xyi$, we get: $z^2 + 4i - 3 = x^2 - y^2 + 2xyi + 4i - 3 = (x^2 - y^2 - 3) + (2xy + 4)i$. |
| chunk 7 This magnitude can be simplified to the square root of the sum of the squares of the real and imaginary parts: | $z - i + 2 = x + (y - 1) = x + yi - i + 2 = (x + 2) + yi.$ |
| chunk 8 $\sqrt{(x^2 - y^2 + 1 + x + y)^2 + (2xy + x + y)^2}.$ | Using the same logic as above, the magnitude is: $\sqrt{(x^2 - y^2 - 3 + x + y + 2)^2 + (2xy + 4 + x + y)^2} = \sqrt{(x^2 - y^2 - 1 + x + y)^2 + (2xy + x + y + 4)^2}.$ |
| chunk 9 | chunk 4 (Solve the System of Equations to Find z) Given that these two magnitudes sum to 4, and recognizing that the equality $a^2 + b^2 = 4$ indicates that each term must separately be 2 (because the maximum magnitude of any complex number is 2), we derive: $(x^2 - y^2 + 1 + x + y)^2 + (2xy + x + y)^2 = 4,$ $(x^2 - y^2 - 3 + x + y + 2)^2 + (2xy + 4 + x + y)^2 = 4.$ |
| Similarly, for the second term, we have: | This leads to $x^2 - y^2 + x = 1$ and $x^2 - y^2 - x = -1$. Solving these equations simultaneously, we get $x = 1$ and $y = 0$, so $z = 1$. |
| chunk 10 $\left \frac{z^2 + 4i - 3}{z - i + 2} \right = \left \frac{(x + yi)^2 + 4i - 3}{x + yi - i + 2} \right .$ | chunk 5 (Calculate the Final Minimum Value) Finally, we need to find the minimum value of $ z - 1 $. This simplifies to: $ z - 1 = 1 - 1 = 0.$ |
| chunk 11 Since $z^2 + 4i - 3 = x^2 - y^2 + 2xyi + 4i - 3 = (x^2 - y^2 - 3) + (2xy + 4)i$, we get: | Thus, the minimum value of $ z - 1 $ is: $0.$ |
| chunk 12 $z - i + 2 = x + (y - 1) = x + yi - i + 2 = (x + 2) + yi.$ | |
| chunk 13 | |
| Using the same logic as above, the magnitude is: | |
| ... | |
| chunk 21 $0.$ | |

1134
1135
1136
1137
1138

Table 13: A representative example illustrating CoT segmentation methods based on a response from Qwen2.5-1.5B-it. Standard splitting by newlines (left) produces too many chunks without a clear logical step.

| | Split by \n\n | LLM-based self-segmentation (Chunks-of-Thought) |
|--|--|---|
| 1139 1140 1141 1142 1143 1144 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183 1184 1185 1186 1187 | <p>chunk 1 Let's start by analyzing the given conditions: 1. When 5655 is divided by N, the remainder is 11. This can be written as:</p> $5655 = qN + 11$ <p>where q is some integer.</p> <p>chunk 2 2. When 5879 is divided by N, the remainder is 14. This can be written as:</p> $5879 = mN + 14$ <p>where m is some integer.</p> <p>chunk 3 We can express these equations as:</p> $5655 - 11 = 5644 = Nq$ $5879 - 14 = 5865 =Nm$ <p>chunk 4 We know that N is a two-digit positive integer. Therefore, we need to find a value of N that satisfies the above conditions for integer q and m.</p> <p>chunk 5 First, let's determine the possible values of N by factoring 5644 and 5865:</p> $5644 = 2^2 \cdot 7 \cdot 11^2$ $5865 = 3 \cdot 5 \cdot 13^2$ <p>chunk 6 We need to find a common divisor for both 5644 and 5865. Let's check the common divisors:</p> <p>Common divisors are: 1, 7, 11, 143</p> <p>chunk 7 Since N is a two-digit positive integer, we consider the possible values:</p> $N = 11 \quad (\text{if } 11 \text{ is a divisor of both 5644 and 5865})$ <p>chunk 8 Let's test $N = 11$: 1. Check if 11 divides 5644 exactly:</p> $5644 = 11 \times 513 \quad \text{Yes}$ <p>2. Check if 11 divides 5865 exactly:</p> $5865 = 11 \times 533 \quad \text{Yes}$ <p>chunk 9 Since $N = 11$ divides 5644 and 5865 exactly, $N = 11$ satisfies the conditions. Now we have:</p> $Nq = 5644 \quad \text{and} \quad Nm = 5865$ <p>chunk 10 Using the Law of Cosines (not needed for this case, but for completeness):</p> $\cos(\theta) = \frac{a^2 + b^2 - c^2}{2ab}$ <p>where $a = 5644$, $b = 5865$, and c is the hypotenuse (since $11 * 11 * 1$ is our N).</p> <p>chunk 11 Sum of the digits of N:</p> $1 + 1 = 2$ <p>chunk 12 Thus, the sum of the digits of N is:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">2</div> | <p>LLM-based self-segmentation (Chunks-of-Thought)</p> <p>chunk 1 (Formulating the Equations) Let's start by analyzing the given conditions: 1. When 5655 is divided by N, the remainder is 11. This can be written as:</p> $5655 = qN + 11$ <p>where q is some integer.</p> <p>chunk 2 (Finding Common Divisors via Factorization) We know that N is a two-digit positive integer. Therefore, we need to find a value of N that satisfies the above conditions for integer q and m. First, let's determine the possible values of N by factoring 5644 and 5865:</p> $5644 = 2^2 \cdot 7 \cdot 11^2$ $5865 = 3 \cdot 5 \cdot 13^2$ <p>We need to find a common divisor for both 5644 and 5865. Let's check the common divisors:</p> <p>Common divisors are: 1, 7, 11, 143</p> <p>chunk 3 (Verifying the Candidate N) Since N is a two-digit positive integer, we consider the possible values:</p> $N = 11 \quad (\text{if } 11 \text{ is a divisor of both 5644 and 5865})$ <p>Let's test $N = 11$: 1. Check if 11 divides 5644 exactly:</p> $5644 = 11 \times 513 \quad \text{Yes}$ <p>2. Check if 11 divides 5865 exactly:</p> $5865 = 11 \times 533 \quad \text{Yes}$ <p>Since $N = 11$ divides 5644 and 5865 exactly, $N = 11$ satisfies the conditions.</p> <p>chunk 4 (Final Calculation) Now we have: $Nq = 5644 \quad \text{and} \quad Nm = 5865$ <p>Using the Law of Cosines (not needed for this case, but for completeness):</p> $\cos(\theta) = \frac{a^2 + b^2 - c^2}{2ab}$ <p>where $a = 5644$, $b = 5865$, and c is the hypotenuse (since $11 * 11 * 1$ is our N). Sum of the digits of N:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">1 + 1 = 2</div> <p>Thus, the sum of the digits of N is:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">2</div> </p> |

1188
1189
1190
1191
1192
1193
1194
1195

Table 14: An illustrative example from the Process-Bench dataset, where the ground-truth label is “correct.” The judge model is STEPWISER with Rel-Ratio and Qwen2.5-7B-chunk.

1196
1197
1198
1199
1200
1201
1202
1203

Example

Mathematical Problem

Cindy’s math and science books weigh 2 pounds each. Her French book weighs 4 pounds and her English book weighs 3 pounds. Her history book weighs twice as much as her English book. If Cindy carries all of her books at once, what will be the total weight of the books she is carrying?

Historical Reasoning Path

- To determine the total weight of all Cindy’s books, we need to calculate the weight of each book individually and then sum these weights.
- **Math and Science Books:**
 - Each math book weighs 2 pounds.
 - Each science book weighs 2 pounds.
 - Cindy has 2 math books and 2 science books.
 - Total weight of math books: $2 \times 2 = 4$ pounds.
 - Total weight of science books: $2 \times 2 = 4$ pounds.
 - Combined weight: $4 + 4 = 8$ pounds.
- **French Book:**
 - The French book weighs 4 pounds.

New Reasoning Chunk

Third, for the English book:

- The English book weighs 3 pounds.

Judgment

To determine if the “New Reasoning_chunk” contains any explicit errors, we need to analyze the information given in the problem and the reasoning presented.

The new reasoning chunk states: Third, for the English book:

- The English book weighs 3 pounds.

From the problem statement, we know:

- The French book weighs 4 pounds.
- The English book weighs 3 pounds.
- The history book weighs twice as much as the English book, so the history book weighs $2 \times 3 = 6$ pounds.
- The math and science books each weigh 2 pounds, and Cindy has 2 math books and 2 science books.

The new reasoning chunk simply states that the English book weighs 3 pounds, which is directly given in the problem statement. Therefore, this statement is correct and logical. Thus, the final judgment is:

Positive

1234
1235
1236
1237
1238
1239
1240
1241

1242

1243 Table 15: An illustrative example from the Process-Bench dataset, where the ground-truth label is
 1244 “correct.” The judge model is STEPWISER with `Rel-Ratio` and `Qwen2.5-7B-chunk`.

1245

1246

1247

Example

1248

Mathematical Problem

1249

1250 An n -label is a permutation of the numbers 1 through n . For example, $J = 35214$ is a 5-label and
 1251 $K = 132$ is a 3-label. For a fixed positive integer p , where $p \leq n$, consider consecutive blocks of p
 1252 numbers in an n -label. For example, when $p = 3$ and $L = 263415$, the blocks are 263, 634, 341, and
 1253 415. We can associate to each of these blocks a p -label that corresponds to the relative order of the
 1254 numbers in that block. For $L = 263415$, we get the following:

1255

1256

1257

1258

$$263415 \rightarrow 132; \quad 263415 \rightarrow 312; \quad 263415 \rightarrow 231; \quad 263415 \rightarrow 213$$

1259

1260 Moving from left to right in the n -label, there are $n - p + 1$ such blocks, which means we obtain an
 1261 $(n - p + 1)$ -tuple of p -labels. For $L = 263415$, we get the 4-tuple $(132, 312, 231, 213)$. We will call
 1262 this $(n - p + 1)$ -tuple the p -signature of L (or signature, if p is clear from the context) and denote it by
 1263 $S_p[L]$; the p -labels in the signature are called windows. For $L = 263415$, the windows are 132, 312,
 1264 231, and 213, and we write

$$S_3[263415] = (132, 312, 231, 213)$$

1265

1266 More generally, we will call any $(n - p + 1)$ -tuple of p -labels a p -signature, even if we do not know of
 1267 an n -label to which it corresponds (and even if no such label exists). A signature that occurs for exactly
 1268 one n -label is called *unique*, and a signature that doesn’t occur for any n -labels is called *impossible*. A
 1269 *possible* signature is one that occurs for at least one n -label.

1270

1271 In this power question, you will be asked to analyze some of the properties of labels and signatures.

1272 **Compute the 3-signature for 52341.**

1273

Historical Reasoning Path

1274

1275 To compute the 3-signature of the 5-label $L = 52341$, we need to identify all the consecutive blocks of
 1276 3 numbers in the label and then convert each block into its corresponding 3-label based on the relative
 1277 order of the numbers.

1278

1279 First, identify the first block: 523. The relative order of the numbers 5, 2, and 3 is 3, 1, and 2
 1280 respectively. Therefore, the 3-label for this block is 312.

1281

1282 Second, identify the second block: 234. The relative order of the numbers 2, 3, and 4 is 1, 2, and 3
 1283 respectively. Therefore, the 3-label for this block is 123.

1284

New Reasoning Chunk

1285

1286 Third, identify the third block: 341. The relative order of the numbers 3, 4, and 1 is 2, 3, and 1
 1287 respectively. Therefore, the 3-label for this block is 231.

1288

Judgment

1289

1290 To determine if the “New Reasoning.chunk” contains any explicit errors, we need to verify the logic for
 1291 identifying the 3-label of the block 341 in the 5-label 52341.

1292

1293 The new reasoning chunk states:

1294

1295

1296 Third, identify the third block: 341. The relative order of the numbers 3, 4, and 1 is
 1297 2, 3, and 1 respectively. Therefore, the 3-label for this block is 231.

1298

1299 Let’s check the relative order of the numbers in the block 341:

1300

1301

1302

1303

1304

- The number 3 is greater than 1 but less than 4.
- The number 4 is greater than both 3 and 1.
- The number 1 is the smallest.

1305

1306 Thus, the relative order of the numbers 3, 4, and 1 is indeed 2, 3, and 1 respectively. This means the
 1307 3-label for the block 341 is 231.

1308

1309 Since the reasoning follows correctly from the definition of a 3-label, the “New Reasoning.chunk” does
 1310 not contain any explicit errors.

1311

1312 Therefore, the final judgment is:

1313

1314

1315 **Positive**

1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349

Table 16: An illustrative example from the Process-Bench dataset, where the ground-truth label is “incorrect.” The judge model is STEPWISER with Rel-Ratio and Qwen2.5-7B-chunk. We highlight the key parts of the reasoning trajectory in red.

| Example |
|--|
| <p>Mathematical Problem</p> <p>There are 60 empty boxes B_1, \dots, B_{60} in a row on a table and an unlimited supply of pebbles. Given a positive integer n, Alice and Bob play the following game.</p> <p>In the first round, Alice takes n pebbles and distributes them into the 60 boxes as she wishes. Each subsequent round consists of two steps:</p> <p>(a) Bob chooses an integer k with $1 \leq k \leq 59$ and splits the boxes into the two groups B_1, \dots, B_k and B_{k+1}, \dots, B_{60}.</p> <p>(b) Alice picks one of these two groups, adds one pebble to each box in that group, and removes one pebble from each box in the other group.</p> <p>Bob wins if, at the end of any round, some box contains no pebbles. Find the smallest n such that Alice can prevent Bob from winning.</p> |
| <p>Historical Reasoning Path</p> <p>To find the smallest n such that Alice can prevent Bob from winning, we need to understand the constraints imposed by Bob’s moves and how Alice can counter them. Let’s consider the minimum number of pebbles each box must have after the first round. Since there are 60 boxes and Alice distributes n pebbles among them, the minimum number of pebbles each box will get is $\lfloor \frac{n}{60} \rfloor$, where $\lfloor x \rfloor$ represents the greatest integer less than or equal to x.</p> |
| <p>New Reasoning Chunk</p> <p>Now, let’s analyze Bob’s move in each round. In step (a), he splits the boxes into two groups of size k and $60 - k$. In step (b), Alice adds a pebble to each box in the chosen group and removes a pebble from each box in the other group. For Alice to prevent Bob from winning, each box must have at least one pebble after her move. This means that the difference between the total number of pebbles in the chosen group and the other group should be at most 1.</p> |
| <p>Judgment</p> <p>Let’s break this down step by step:</p> <ol style="list-style-type: none"> 1. Understanding Bob’s Move: In step (a), Bob splits the boxes into two groups of size k and $60 - k$. In step (b), Alice adds one pebble to each box in the chosen group and removes one pebble from each box in the other group. This means that the total number of pebbles in the chosen group increases by k and the total number of pebbles in the other group decreases by $60 - k$. 2. Condition for Alice to Prevent Bob from Winning: For Alice to prevent Bob from winning, each box must have at least one pebble after her move. However, the statement “the difference between the total number of pebbles in the chosen group and the other group should be at most 1” is not necessarily true. The correct condition would be that the difference in the total number of pebbles between the two groups after Alice’s move should not cause any box to have zero pebbles. <p>The reasoning in the new reasoning chunk assumes that the difference in the number of pebbles between the two groups should be at most 1, which is not necessarily the case. The correct condition would be that after Alice’s move, the number of pebbles in each box should be at least 1, not necessarily that the difference in the total number of pebbles between the two groups should be at most 1.</p> <p>Therefore, the “New Reasoning_chunk” contains explicit errors. The correct interpretation of the condition for Alice to prevent Bob from winning is not correctly stated in the new reasoning chunk.</p> <p>The final judgment is:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">Negative</div> |

1350

1351 Table 17: An illustrative example from the Process-Bench dataset, where the ground-truth label
 1352 is “incorrect.” The judge model is STEPWISER with Rel-Ratio and Qwen2.5-7B-chunk. We
 1353 highlight the key parts of the reasoning trajectory in red.

1354

Example

1355

Mathematical Problem

1356

1357 Let n and k be fixed positive integers of the same parity, $k \geq n$. We are given $2n$ lamps numbered 1
 1358 through $2n$; each of them can be on or off. At the beginning all lamps are off. We consider sequences of
 1359 k steps. At each step one of the lamps is switched (from off to on or from on to off).

1360

1361 Let N be the number of k -step sequences ending in the state: lamps $1, \dots, n$ on, lamps $n+1, \dots, 2n$ off.

1362

1363 Let M be the number of k -step sequences leading to the same state and not touching lamps
 1364 $n+1, \dots, 2n$ at all.

1365

1364 Find the ratio N/M .

1365

Historical Reasoning Path

1366

1367 To solve the problem, we need to determine the number of k -step sequences that end in the state where
 1368 lamps $1, \dots, n$ are on and lamps $n+1, \dots, 2n$ are off, and compare it to the number of such sequences
 1369 that do not touch lamps $n+1, \dots, 2n$ at all.

1370

New Reasoning Chunk

1371

1372 First, we determine N . The total number of lamps is $2n$, and we need exactly n of them to be on
 1373 and n to be off. Since the sequence ends with lamps $1, \dots, n$ on and $n+1, \dots, 2n$ off, we need to
 1374 switch exactly n times among the first n lamps and the remaining $k-n$ times among the last n lamps.
 1375 However, since the last n lamps are always off, these $k-n$ switches do not affect their state. Therefore,
 1376 we only need to count the number of ways to choose n steps out of k to switch the first n lamps. **This is**
 1377 **given by the binomial coefficient:**

1378

$$N = \binom{k}{n}.$$

1379

Judgment

1380

1381 To determine if the “New Reasoning_chunk” contains any explicit errors, we need to carefully analyze
 1382 the statement and check its logic step by step.

1383

1384 The reasoning chunk suggests that N , the number of k -step sequences ending in the state where lamps
 1385 $1, \dots, n$ are on and lamps $n+1, \dots, 2n$ are off, **is given by the binomial coefficient** $\binom{k}{n}$. **However,**
 1386 **this conclusion is not necessarily correct because it does not take into account the specific sequence of**
 1387 **switching the lamps.**

1388

1389 The correct approach to find N is to consider that we need exactly n switches among the first n lamps
 1390 to turn them on, and the remaining $k-n$ switches can be any combination of the n lamps (including
 1391 possibly switching some lamps on and off multiple times, but ensuring that at the end, each of the first
 1392 n lamps is on). The number of such sequences is not simply $\binom{k}{n}$, because it does not account for the
 1393 specific order in which the switches occur.

1394

1395 In fact, the correct way to calculate N is more complex and involves considering the parity of k and
 1396 n (which are the same) and the specific sequences of switching. The reasoning chunk simplifies the
 1397 problem incorrectly by assuming that any n switches among the first n lamps will result in all of them
 1398 being on, without considering the constraints of the problem.

1399

1400 Therefore, the “New Reasoning_chunk” contains explicit errors. The correct reasoning would involve a
 1401 more detailed analysis of the sequences of switches, taking into account the parity and the specific
 1402 requirements of the problem.

1403

1404 The final judgment is:

Negative

1404

1405

1406

1407 Table 18: An illustrative example from the Process-Bench dataset, where the ground-truth label
 1408 is “incorrect.” The judge model is STEPWISER with Rel-Ratio and Qwen2.5-7B-chunk. We
 1409 highlight the key parts of the reasoning trajectory in red.

1410

1411

1412

1413

1414

1415

1416

1417

1418

1419

1420

1421

1422

1423

1424

1425

1426

1427

1428

1429

1430

1431

1432

1433

1434

1435

1436

1437

1438

1439

1440

1441

1442

1443

1444

1445

1446

1447

1448

1449

1450

1451

1452

1453

1454

1455

1456

1457

Example

Mathematical Problem

Determine all functions f from the set of positive integers into the set of positive integers such that for all x and y there exists a non degenerated triangle with sides of lengths

$$x, f(y) \text{ and } f(y + f(x) - 1).$$

Historical Reasoning Path

To determine all functions f from the set of positive integers into the set of positive integers such that for all x and y there exists a non-degenerate triangle with sides of lengths x , $f(y)$, and $f(y + f(x) - 1)$, we need to use the triangle inequality. The triangle inequality states that the sum of the lengths of any two sides of a triangle must be greater than the length of the remaining side. Therefore, we have the following three inequalities: First, $x + f(y) > f(y + f(x) - 1)$ Second, $x + f(y + f(x) - 1) > f(y)$ Third, $f(y) + f(y + f(x) - 1) > x$

We will analyze these inequalities one by one.

Step 1: Analyze the inequality $x + f(y) > f(y + f(x) - 1)$

This inequality can be rewritten as:

$$f(y + f(x) - 1) < x + f(y)$$

This tells us that $f(y + f(x) - 1)$ is strictly less than $x + f(y)$.

New Reasoning Chunk

Step 2: Analyze the inequality $x + f(y + f(x) - 1) > f(y)$

This inequality can be rewritten as:

$$f(y + f(x) - 1) > f(y) - x$$

This tells us that $f(y + f(x) - 1)$ is strictly greater than $f(y) - x$. Since $f(y)$ and x are positive integers, $f(y) - x$ could be negative or zero, but $f(y + f(x) - 1)$ must be a positive integer. Therefore, the smallest possible value for $f(y) - x$ is 0, which means $f(y) \geq x$.

Judgment

To determine if the ”New Reasoning_chunk” contains any explicit errors, we need to analyze the given inequality $x + f(y + f(x) - 1) > f(y)$ and see if it leads to the conclusion that $f(y) \geq x$.

Starting with the inequality:

$$x + f(y + f(x) - 1) > f(y)$$

we can rearrange it to:

$$f(y + f(x) - 1) > f(y) - x$$

Since $f(y)$ and x are positive integers, $f(y) - x$ could be negative or zero. However, $f(y + f(x) - 1)$ must be a positive integer. This means that $f(y) - x$ must be less than $f(y + f(x) - 1)$, but it does not necessarily mean that $f(y) \geq x$. The inequality $f(y + f(x) - 1) > f(y) - x$ only tells us that $f(y) - x$ is less than a positive integer, which does not imply that $f(y)$ itself must be at least x .

Therefore, the reasoning in the new reasoning chunk contains an explicit error in its conclusion that $f(y) \geq x$. The correct interpretation of the inequality $f(y + f(x) - 1) > f(y) - x$ does not necessarily lead to $f(y) \geq x$.

The final judgment is:

Negative

1458
1459
1460
1461
1462
1463

1464 Table 19: An illustrative example from the Process-Bench dataset, where the ground-truth label
1465 is “incorrect.” The judge model is STEPWISER with Rel-Ratio and Qwen2.5-7B-chunk. We
1466 highlight the key parts of the reasoning trajectory in red.

| 1467 | Example |
|------|--|
| 1468 | Mathematical Problem |
| 1469 | How many ways are there to put 4 balls in 3 boxes if the balls are distinguishable but the boxes are not? |
| 1470 | |
| 1471 | Historical Reasoning Path |
| 1472 | To solve this problem, we need to consider the number of ways to distribute 4 distinguishable balls into 3 indistinguishable boxes. |
| 1473 | Step 1: Counting the total number of distributions if the boxes were distinguishable. If the boxes were distinguishable, each ball 1474 would have 3 choices (3 boxes). Since there are 4 distinguishable balls, the total number of distributions would be $3^4 = 81$. However, since the boxes are indistinguishable, we need to adjust for overcounting. |
| 1475 | Step 2: Identifying the cases where the boxes are indistinguishable. There are three main cases where the boxes are indistinguishable: |
| 1476 | First, all 4 balls in one box (1 way). |
| 1477 | Second, 3 balls in one box and 1 ball in another box (4 ways, since the balls are distinguishable and we can choose any of the 4 balls 1478 to be alone). |
| 1479 | Third, 2 balls in one box and 2 balls in another box (6 ways, since there are 6 pairs of balls that can be in the same box). However, 1480 these cases also include some overcounting since the boxes are indistinguishable. We need to adjust for this overcounting. |
| 1481 | New Reasoning Chunk |
| 1482 | Step 3: Adjusting for overcounting. In case 2, when we have 3 balls in one box and 1 ball in another box, we counted each distribution 1483 3 times (since there are 3 boxes and we could swap the lone ball with the other balls). So, we divide by 3 for this case. In case 3, 1484 when we have 2 balls in one box and 2 balls in another box, we counted each distribution 6/2 = 3 times (since there are 3 pairs of boxes). So, we divide by 3 for this case. Now we have: |
| 1485 | <ul style="list-style-type: none"> • Case 1: 1 way • Case 2: 4 / 3 ways • Case 3: 6 / 3 ways |
| 1486 | |
| 1487 | |
| 1488 | Judgment |
| 1489 | To determine if the “New Reasoning_chunk” contains any explicit errors, we need to carefully analyze the reasoning step by step. 1490 The new reasoning chunk states: In case 2, when we have 3 balls in one box and 1 ball in another box, we counted each distribution 3 1491 times (since there are 3 boxes and we could swap the lone ball with the other balls). So, we divide by 3 for this case. In case 3, when 1492 we have 2 balls in one box and 2 balls in another box, we counted each distribution 6/2 = 3 times (since there are 3 pairs of boxes). 1493 So, we divide by 3 for this case... |
| 1494 | Let’s evaluate each case: |
| 1495 | Case 2: 3 balls in one box and 1 ball in another box |
| 1496 | When we have 3 balls in one box and 1 ball in another box, we do indeed count each distribution 3 times because we can place the 1497 single ball in any one of the 3 boxes. Therefore, the correct number of distinct distributions for this case is 4 (the number of ways 1498 to choose which ball will be the single ball) divided by 3, but this division is incorrect because it assumes that each distribution is 1499 counted exactly 3 times, which is not necessarily true. The correct approach would be to consider the different ways to assign the 1500 balls without overcounting due to the indistinguishability of the boxes. The correct count for this case is actually 4, not 4/3. |
| 1501 | Case 3: 2 balls in one box and 2 balls in another box |
| 1502 | When we have 2 balls in one box and 2 balls in another box, we need to count the number of ways to partition the 4 balls into two 1503 groups of 2. The number of ways to choose 2 balls out of 4 is $\binom{4}{2} = 6$, but since the boxes are indistinguishable, we have counted 1504 each distribution twice (once for each way to assign the groups to the boxes). Therefore, the correct number of distinct distributions 1505 for this case is 6/2 = 3, which the reasoning chunk correctly states. |
| 1506 | Given the above analysis, the reasoning in the new reasoning chunk contains an explicit error in its calculation for Case 2. Therefore, 1507 the final judgment is: |
| 1508 | Negative |
| 1509 | |
| 1510 | |
| 1511 | |

1512
1513
1514
1515
1516

1517 Table 20: An illustrative example of chunk-reset reasoning. The judge model is STEPWISER with
1518 Rel-Effective and Qwen2.5-7B-chunk. In the first attempt of chunk 3, it contains a significant
1519 algebraic mistake in the formula relating λ and r .

1520
1521
1522
1523
1524

Example

Mathematical Problem

8 singers participate in the arts festival, preparing to arrange m performances, each time 4 of them will perform on stage, requiring that any two of the 8 singers perform together the same number of times. Please design a scheme so that the number of performances m is the least.

Chunk 1

To solve this problem, we need to ensure that each pair of singers performs together in the same number of performances. This is a specific type of combinatorial design problem, and the solution can be approached using the concept of a balanced incomplete block design (BIBD).

Here are the steps to find the solution: **1. Identify the parameters of the BIBD:**

- Let ($v = 8$) (the number of singers).
- Let ($k = 4$) (the number of singers performing in each performance).
- Let (r) be the number of performances each singer participates in.
- Let (λ) be the number of performances each pair of singers participates in together.
- Let (b) be the number of performances (the number we need to determine).

Chunk 2

2. Use the BIBD parameters to derive equations: The equation for the number of performances each singer participates in is ($r(k-1) = (v-1)\lambda$). The equation for the number of performances is ($b = \frac{vr}{k}$).

Chunk 3 (first attempt): **This step uses the incorrect formula** $\lambda = \frac{4(r-1)}{7}$ **instead of the correct one** $\lambda = \frac{3r}{7}$.

3. Determine the value of (λ):

- From the first equation, we can express (λ) as ($\lambda = \frac{r(k-1)}{v-1}$).
- Substituting ($v = 8$) and ($k = 4$), we get ($\lambda = \frac{4r-4}{7} = \frac{4(r-1)}{7}$).
- Since (λ) must be an integer, ($4(r-1)$) must be divisible by 7. The smallest positive integer (r) that satisfies this condition is ($r = 2$) (since ($4(2-1) = 4$), and 4 is not divisible by 7, the next is ($r = 8$) which gives ($4(8-1) = 28$), which is divisible by 7).
- Therefore, ($\lambda = \frac{28}{7} = 4$).

Chunk 3 (second attempt)

3. Calculate the parameters:

- First, we need to find (λ). We know that ($r = \frac{(v-1)\lambda}{k-1}$). For (λ) to be an integer, $(v-1)\lambda$ must be divisible by $(k-1 = 3)$. So, λ must be a multiple of 3.
- Let's try ($\lambda = 1$). Then ($r = \frac{(8-1)\cdot 1}{4-1} = \frac{7}{3}$), which is not an integer.
- Let's try ($\lambda = 3$). Then ($r = \frac{(8-1)\cdot 3}{4-1} = \frac{21}{3} = 7$).
- Now, we can calculate (b): ($b = \frac{8\cdot 7}{4} = 14$).

Chunk 4

4. Verify the design:

- We need to check if we can indeed arrange 14 performances with 8 singers, 4 at a time, such that each pair of singers performs together exactly 3 times.
- This can be verified using known designs or by constructing a specific design, but for the sake of this problem, we assume the existence of such a design.

Therefore, the least number of performances m is

14.

1559
1560
1561
1562
1563
1564
1565

1566

1567

1568 Table 21: An illustrative example of chunk-reset reasoning. The judge model is STEPWISER with
 1569 Rel-Effective and Qwen2.5-7B-chunk. In the first attempt of chunk 3, the model fails to verify
 1570 the condition.

1571

1572

1573

Example**Mathematical Problem**

Which is the two-digit number that, when divided by the digit in the units place, gives a quotient of 9 and a remainder of 6?

Chunk 1

To solve the problem, let's break it down step by step.

1. Identify the structure of the number and the division operation:

- Let the two-digit number be represented as $10a + b$, where a is the tens digit and b is the units digit.
- According to the problem, when this number is divided by the digit in the units place (which is b), the quotient is 9 and the remainder is 6. This can be expressed as:

$$10a + b = 9b + 6$$

Chunk 2**2. **Simplify the equation:**

- Rearrange the equation to isolate terms involving a :

$$10a + b - 9b = 6$$

- Simplify: $10a - 8b = 6$
- Further simplify by dividing the entire equation by 2: $5a - 4b = 3$

Chunk 3 (first attempt) This step fails verify the condition and stop with $b = 3$.**3. Solve for a and b :**

- We need to find integer values of a and b that satisfy the equation $5a - 4b = 3$. Since a and b are digits (0-9) and a cannot be 0 (as $10a$ would not be a two-digit number), let's test possible values for b .
- If $b = 2$:

$$5a - 4(2) = 3 \implies 5a - 8 = 3 \implies 5a = 11$$
 $5a = 11$ does not give an integer value for a .
- If $b = 1$:

$$5a - 4(1) = 3 \implies 5a - 4 = 3 \implies 5a = 7$$
 $5a = 7$ does not give an integer value for a .
- If $b = 3$:

$$5a - 4(3) = 3 \implies 5a - 12 = 3 \implies 5a = 15 \implies a = 3$$
This gives an integer value for a .

Chunk 3 (second attempt)****Determine possible values for a and b :**

- Since a and b are digits (0-9), we need to find values that satisfy the equation $5a - 4b = 3$.
- We will test possible values for b (since b is the units digit and must be a digit from 1 to 9, as 0 would make the remainder 6 impossible).

– Testing $b = 1$:

$$5a - 4(1) = 3 \implies 5a - 4 = 3 \implies 5a = 7 \implies a = \frac{7}{5} \text{ (not an integer)}$$

– Testing $b = 3$:

$$5a - 4(3) = 3 \implies 5a - 12 = 3 \implies 5a = 15 \implies a = 3$$

$a = 3$ is an integer, and we need to check if $b = 3$ is feasible. If $a = 3$ and $b = 3$, the number is 33. We need to verify:

$$33 \div 3 = 11 \text{ (quotient = 11, remainder = 0) (not a solution)}$$

- Testing $b = 4$: $5a - 4(4) = 3 \implies 5a - 16 = 3 \implies 5a = 19 \implies a = \frac{19}{5}$ (not an integer)
- Testing $b = 5$: $5a - 4(5) = 3 \implies 5a - 20 = 3 \implies 5a = 23 \implies a = \frac{23}{5}$ (not an integer)
- Testing $b = 6$: $5a - 4(6) = 3 \implies 5a - 24 = 3 \implies 5a = 27 \implies a = \frac{27}{5}$ (not an integer)
- Testing $b = 7$: $5a - 4(7) = 3 \implies 5a - 28 = 3 \implies 5a = 31 \implies a = \frac{31}{5}$ (not an integer)
- Testing $b = 8$: $5a - 4(8) = 3 \implies 5a - 32 = 3 \implies 5a = 35 \implies a = 7$. $a = 7$ is an integer. So, the number is 78. We need to verify:

$$78 \div 8 = 9 \text{ (quotient = 9, remainder = 6)}$$

This satisfies the condition.

Chunk 4**4. Conclusion:** The two-digit number that meets the condition is 78. Therefore, the final answer is:

1620
1621
1622
1623
1624

Table 22: An illustrative example of chunk-reset reasoning. The judge model is STEPWISER with Rel-Effective and Qwen2.5-7B-chunk. The idea is to use the commutative law of multiplication and division to perform large-scale reduction, while the rejected chunks focus on the approximation for simplifying the expression.

1625
1626
1627
1628
1629**Example****Mathematical Problem**Calculate: $241 \times 345 \div 678 \div 345 \times (678 \div 241)$ **Chunk 1 (first attempt)** we don't need the approximation for simplification

Let's break down the given expression step by step:

1630
1631

$$241 \times 345 \div 678 \div 345 \times \left(\frac{678}{241} \right).$$

1632

First, let's simplify the expression inside the parentheses:

1633
1634

$$\left(\frac{678}{241} \right) = 2.812 \approx 2 \text{ (for simplicity, we can approximate it to 2 if dealing with integers)}$$

1635
1636
1637**Chunk 1 (second attempt)**

Let's break down the given expression step by step:

1638
1639
1640

$$241 \times 345 \div 678 \div 345 \times \left(\frac{678}{241} \right).$$

1641
1642

First, let's simplify the expression by following the operations order:

1. Calculate the division inside the parentheses first:

1643
1644

$$\frac{678}{241} = 2.8 \text{ (approximately, but we don't need the approximation for simplification here, just the fraction)}$$

1645
1646

So the expression becomes:

$$241 \times 345 \div 678 \div 345 \times \frac{678}{241}$$

1647
1648**Chunk 2 (first attempt)**

2. Simplify the expression by canceling out common factors:

1649
1650
1651

- The 345 in the numerator and one of the denominators cancel out:

$$241 \times 345 \div 678 \div 345 \times \frac{678}{241} = 241 \times \frac{678}{678} \times \frac{678}{241}$$

1652
1653
1654

- The 678 in the numerator and one of the denominators cancel out:

$$241 \times \frac{678}{678} \times \frac{678}{241} = 241 \times 1 \times \frac{678}{241}$$

1655
1656
1657

- The 241 in the numerator and the denominator cancel out:

$$241 \times 1 \times \frac{678}{241} = 678 \times 1 = 678$$

1658
1659
1660**Chunk 2 (second attempt)**2. Notice that $345 \div 345 = 1$, so we can simplify the expression to:1661
1662
1663

$$\begin{aligned} 241 \times 1 \div 678 \times \frac{678}{241} \\ = 241 \div 678 \times \frac{678}{241} \end{aligned}$$

1664
1665**Chunk 3**

3. Now, we can cancel out 678 in the numerator and the denominator:

1666
1667
1668
1669
1670
1671

$$\begin{aligned} 241 \times \left(\frac{1}{678} \times \frac{678}{241} \right) \\ = 241 \times \left(\frac{1}{241} \right) \\ = 241 \times \frac{1}{241} = 1 \end{aligned}$$

1672
1673**Chunk 4**

Therefore, the final answer is: