

EMPOWERING LLM TOOL INVOCATION WITH TOOL-CALL REWARD MODEL

Anonymous authors

Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) have recently alleviated limitations in outdated internal knowledge and computational inaccuracies by invoking external tools such as search engines and code generation. While reinforcement learning (RL) has substantially enhanced tool usage in LLMs, most existing agentic RL approaches rely solely on outcome-only reward signals, which assign credit at a coarse granularity and often induce gradient conflict (e.g., correct tool calls may be penalized due to incorrect final answers). To address this, we propose the *Tool-call Reward Model* (TRM), a specialized process reward model meticulously designed to evaluate and reward each tool invocation. Since previous PRM research has predominantly focused on traditional reasoning tasks such as step-wise mathematical reasoning, the introduction of TRM brings two unique challenges: (1) limited understanding of how to construct effective TRMs, including data requirements and model size; and (2) difficulties integrating TRM with classical RL algorithms such as PPO and GRPO, where naive adaptation may lead to reward hacking (minimizing tool calls to avoid penalties). To tackle these challenges, we establish a systematic TRM construction workflow and propose refined credit assignment and turn-level advantage estimation for effective integration with PPO and GRPO. Experiments show that a 3B TRM trained on 10K samples achieves robust performance. On search-based QA and Python code-based math tasks, integrating TRM consistently outperforms outcome-only reward RL methods across models of different sizes.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated sophisticated proficiency in addressing complex tasks, profoundly impacting a broad spectrum of domains (OpenAI, 2023; Guo et al., 2025; Yang et al., 2025). However, LLMs are fundamentally limited by the static nature of their internal knowledge and their propensity to make computational errors (Schick et al., 2023; Qian et al., 2025a). To overcome these challenges, LLMs increasingly invoke external tools, such as search engines for accessing up-to-date information (Jin et al., 2025; Chen et al., 2025b) and code generation for solving complex mathematical problems (Liao et al., 2024; Feng et al., 2025).

With tool invocation playing an increasingly important role in overcoming LLM limitations, reinforcement learning (RL), proven effective in traditional reasoning tasks (Guo et al., 2025; Team, 2025; Team et al., 2025; Wang et al., 2024), has been widely used to enhance tool usage. In practice, most RL-based approaches (Jin et al., 2025; Song et al., 2025; Feng et al., 2025; Li et al., 2025b) for tool invocation rely solely on outcome reward signals, evaluating only the correctness of the final output (e.g., math answer correctness) while overlooking the quality of intermediate tool calls. Consequently, credit for each tool call in a trajectory is assigned solely based on the final outcome, irrespective of its individual quality or usefulness. With uniform treatment of tool calls, this approach limits the ability of the model to learn effective tool usage, potentially resulting in unstable or suboptimal performance. For example, if the final answer is incorrect, a trajectory with correct intermediate tool usage is still penalized (Figure 1-a and Figure 1-b)¹. This discourages learning of

¹A reasonable way to determine the paternal grandfather of a person is to first determine the father of the person, then the father of that father.

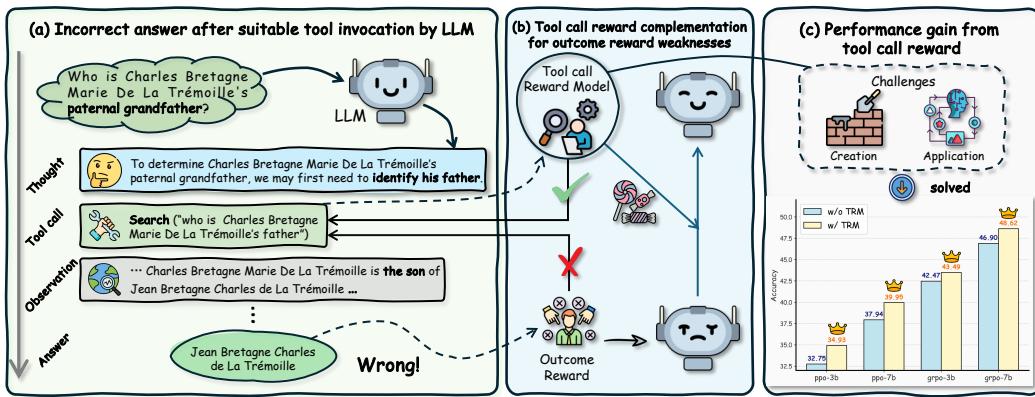


Figure 1: Overview of TRM for improving tool invocation in LLMs: (a) limitations of outcome-only reward, (b) benefits of tool call reward, and (c) performance gains from integrating tool call reward.

effective tool invocation strategies and causes *gradient conflict* (Lightman et al., 2024; Deng et al., 2025b), often leading to unstable tool usage and degraded performance.

To this end, we propose the *Tool-call Reward Model* (TRM), which quantitatively measures the utility of each tool invocation and assigns corresponding rewards. Although TRM can be viewed as a specific type of process reward model (PRM), prior PRM research (Lightman et al., 2024; Zhang et al., 2025b; Setlur et al., 2025) has predominantly focused on traditional reasoning tasks, leaving tool invocation underexplored. TRM fills this gap by enabling fine-grained monitoring of tool usage, thereby facilitating more appropriate tool invocation (Figure 1-b). However, introducing TRM raises two new key challenges (Figure 1-c): 1) *TRM creation*: how to construct an effective TRM, and 2) *TRM application*: how to integrate it with classical RL algorithms.

For the first challenge, the main difficulty lies in the limited understanding of TRMs, including how to construct training data, the required data volume, and the suitable model size. To address this, we develop a workflow to distill training data from frontier LLMs (§ 2.2) and systematically analyze the impact of data volume and model size on TRM performance (§ 3.1). Beyond this, integrating TRM with classical RL algorithms such as PPO Schulman et al. (2017) and GRPO Shao et al. (2024) remains an open challenge, as directly transferring approaches that combine standard PRM and RL algorithms may not work well for TRM. For instance, in GRPO, our experiments demonstrate that group-level advantage estimation (Shao et al., 2024) of tool call reward can result in reward hacking, where the model prefers fewer tool calls over effective usage (see Appendix E.1). To address these issues, we refine the credit assignment strategy by allocating tool call rewards to the end of each tool invocation, and introduce turn-level advantage estimation in GRPO (§ 2.3). Ultimately, our experiments show that the proposed methods yield better overall model performance (Figure 1-c, § 3.2). Furthermore, we observe that TRM enhances generalization in tool invocation, enabling the model to flexibly adapt to unseen tools (§ 3.3).

In summary, this work makes the following three contributions: 1) We propose the *Tool-call Reward Model* (TRM) and conduct a thorough investigation into its construction. 2) We develop and analyze new algorithms for integrating TRM with classical RL methods, including refined credit assignment strategies (PPO) and step-wise advantage estimation (GRPO). 3) We validate our approaches through extensive experiments, demonstrating significant improvements in model performance. We plan to make our data and code publicly available to facilitate future research.

2 METHODOLOGY

We introduce a Tool-call Reward Model (TRM) to resolve gradient conflict from outcome-only rewards by supplying fine-grained, per-call utility signals that stabilize the tool invocation. In this section, we (i) formalize the multi-turn RL framework for tool invocation in LLMs, (ii) detail the construction of TRM, including training data distillation and model optimization, and (iii) integrate

108 TRM with classic RL algorithms by proposing turn-level credit assignment and enhancing GRPO
 109 with turn-level advantage estimation.
 110

111 **2.1 PROBLEM FORMULATION**
 112

113 We formalize multi-turn tool invocation in LLMs as a sequential decision-making process under
 114 the reinforcement learning framework. Following the ReAct paradigm (Yao et al., 2023), the LLM
 115 alternates between reasoning steps and tool invocations, enabling dynamic planning and external
 116 information gathering for more robust and interpretable task-solving. Formally, consider a prompt
 117 p and an LLM π parameterized by θ . Given p , the LLM π engages in multiple rounds of tool
 118 invocation, where at each round, the model reasons over the current information and decides on the
 119 next tool action. This iterative process continues until the model is ready to produce the final answer.
 120 Finally, the LLM π generates a trajectory

$$\tau = (p, t_1, a_1, o_1, \dots, t_{n_\tau}, a_{n_\tau}, o_{n_\tau}, t_{n_\tau+1}, y), \quad (1)$$

121 where t_i ($1 \leq i \leq n_\tau + 1$) denotes the reasoning thought, a_i and o_i ($1 \leq i \leq n_\tau$) is the tool invoked
 122 and its corresponding output at turn i , n_τ is the total number of tool invocation rounds, and y is the
 123 final answer produced by the LLM π . Here, we refer to each triplet (t_i, a_i, o_i) as a single *turn* in the
 124 interaction.²

125 Given this formulation, our objective is to optimize the policy π_θ to maximize the likelihood of
 126 producing the correct final answer y at the end of the trajectory. Formally, the learning objective is
 127 to maximize the expected correctness of the final answer y over trajectories generated by the policy
 128 π_θ :

$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} [\mathbb{I}(y = y^*)], \quad (2)$$

131 where y^* is the ground-truth answer and $\mathbb{I}(\cdot)$ is the indicator function.

133 **2.2 CONSTRUCTION OF TRM**
 134

135 **Data Distillation** We first describe the process of distilling high-quality training data for TRM
 136 from frontier LLMs (Figure 2-a). This process consists of two main steps: 1) *rollout collection* and
 137 2) *tool call evaluation*. In the rollout collection step, the model is provided with a set of prompts and
 138 a tool-enabled environment, and generates multi-turn trajectories by autonomously invoking tools
 139 to complete the task. For each collected rollout, we further evaluate every tool call a_i by re-feeding
 140 the whole trajectory into the model to assess its utility. Specifically, we assign two binary scores for
 141 each tool call a_i :

- 142 • *necessity* s_{ne}^i : whether the tool call contributes substantive progress toward task completion
- 143 • *quality* s_q^i : whether the tool is invoked with reasonable parameters or used correctly

144 Hence, a tool call is assigned a score of 1 only when it is both necessary for task progress and
 145 executed with high quality; if either criterion is not met, the score is 0. Formally, for a tool call a_i ,
 146 the final score is defined as:

$$s^i = s_{\text{ne}}^i \cdot s_q^i, \quad (3)$$

147 where $s_{\text{ne}}^i, s_q^i \in \{0, 1\}$. The detailed design of prompts are illustrated in Appendix A.1.

148 **TRM Training** The TRM adopts a transformer-based (Vaswani et al., 2017) LLM as its back-
 149 bone. To adapt the model for tool-call utility prediction, we replace the original language modeling
 150 head (used for next-token prediction) with a binary classification head consisting of a single linear
 151 layer. Specifically, for each tool call a_i , the model produces a probability $\tilde{s}^i \in [0, 1]$ based on the
 152 hidden state of the last token of the tool call output o_i . This score indicates the predicted utility of
 153 the tool call. During training, the TRM is optimized using a binary cross-entropy loss³:

$$\mathcal{L}_{\text{BCE}} = \mathbb{E}_\tau \left[-\frac{1}{n_\tau} \sum_{i=1}^{n_\tau} (s^i \log \tilde{s}^i + (1 - s^i) \log (1 - \tilde{s}^i)) \right]. \quad (4)$$

157 ²The final turn consists of both reasoning and the generation of the final answer, without involving any tool
 158 call.

159 ³In practice, a score is also produced at the last token of the answer to indicate its correctness.

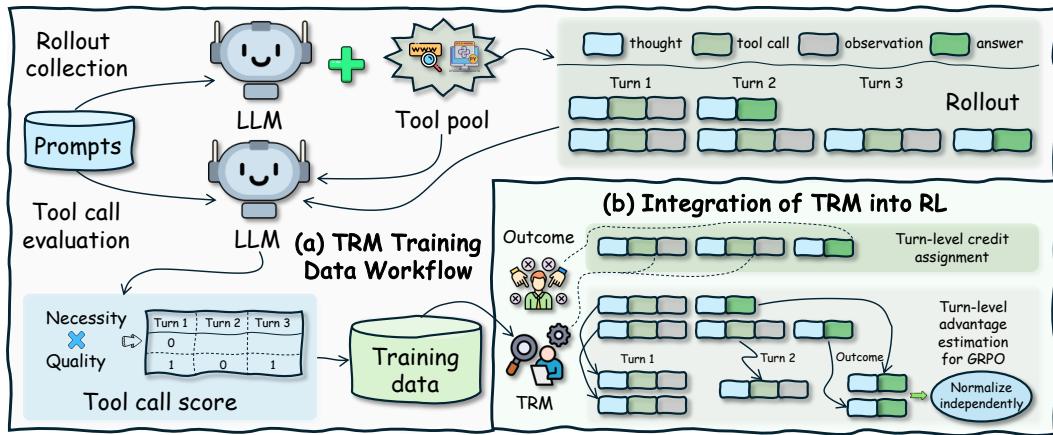


Figure 2: TRM-guided LLM tool invocation. (a) Generation of tool invocation trajectories and turn-level utility labels for TRM training. (b) Turn-level credit assignment and GRPO adaptation via turn-level advantage estimation.

2.3 INTEGRATION OF TRM WITH RL

With TRM in place, we proceed to integrate it into established RL algorithms to optimize tool invocation in LLMs. Specifically, we focus on two representative policy optimization methods⁴: Proximal Policy Optimization (PPO) and Group Relative Policy Optimization (GRPO).

Turn-level Credit Assignment To achieve appropriate credit assignment throughout the trajectory, we combine TRM scores for intermediate tool invocations with the outcome reward for the final answer (Figure 2-b). In particular, for each turn i ($1 \leq i \leq n_\tau$) of trajectory τ , the reward is given by the TRM score \tilde{s}^i , and for the final reasoning step ($i = n_\tau + 1$), the reward is determined by the correctness of the final answer. Mathematically, the turn-level reward \tilde{r}^i is defined as

$$\tilde{r}^i = \begin{cases} \tilde{s}^i, & 1 \leq i \leq n_\tau \\ \mathbb{I}(y = y^*), & i = n_\tau + 1 \end{cases}. \quad (5)$$

Both PPO and GRPO perform policy optimization at the token level, whereas our reward signals are defined at the turn level. To bridge this granularity gap, we also represent each trajectory as a sequence of tokens, $\tau = (x_1, x_2, \dots, x_L)$, where x_j is the j -th token. For each turn i ($1 \leq i \leq n_\tau$), we identify e_i as the index of the last token of the tool call a_i . The set $\mathcal{E} = \{e_1, \dots, e_{n_\tau}\}$ thus marks all tool-call-ending tokens. We further define a mapping $\mathcal{I}(j)$ that returns the corresponding turn index for any $j \in \mathcal{E}$, and set $\mathcal{I}(L) = n_\tau + 1$ for the final answer. To specify which tokens participate in policy optimization, we define $\mathcal{M} \subseteq \{1, \dots, L\}$ as the set of indices of thought, tool call, and answer tokens that are not masked during RL training. These notations facilitate our subsequent discussion on the integration of TRM with RL.

Integration with PPO To enable token-level policy optimization, we map turn-level rewards to the corresponding tokens by assigning the reward for each tool call to the last token of the associated action, and the outcome reward to the last token of the answer. Formally, the reward r^j of token x_j ($1 \leq j \leq L$) is defined as

$$r^j = \begin{cases} \alpha \cdot \tilde{r}^{\mathcal{I}(j)}, & j \in \mathcal{E} \\ \tilde{r}^{\mathcal{I}(j)}, & j = L \\ 0, & \text{otherwise} \end{cases}, \quad (6)$$

where $\alpha \in (0, 1]$ is a hyperparameter controlling the weight of the TRM score. Advantage A^j is then computed from r^j (e.g., Generalized Advantage Estimation (Schulman et al., 2016)). With this

⁴For clarity, KL regularization is omitted in our discussion.

216 token-level advantage, the PPO objective is formulated as
 217

$$218 \quad \mathcal{L}_{\text{PPO}} = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\frac{1}{|\mathcal{M}|} \sum_{j \in \mathcal{M}} \min \left(w^j(\theta) \cdot A^j, \text{clip} \left(w^j(\theta), 1 - \epsilon, 1 + \epsilon \right) A^j \right) \right], \quad (7)$$

221 where $w^j(\theta) = \frac{\pi_{\theta}(x_j | x_{<j})}{\pi_{\theta_{\text{old}}}(x_j | x_{<j})}$ and ϵ is the clipping parameter.
 222

224 **Integration with GRPO** GRPO is a policy optimization method that compares and normalizes
 225 rewards across a batch of trajectories to increase training efficiency. In GRPO, a group refers to a
 226 batch of G trajectories $\{\tau_1, \dots, \tau_G\}$. For each trajectory τ_g ($1 \leq g \leq G$), variables such as n_{τ_g} , \mathcal{E}_g ,
 227 \mathcal{M}_g , and other notations follow the same definitions as in previous sections, with the addition of the
 228 trajectory index g . Across the group, we collect the TRM rewards $\tilde{\mathcal{R}}_{\text{trm}}^i$ (for any turn i) and outcome
 229 rewards $\tilde{\mathcal{R}}_{\text{out}}$ via

$$230 \quad \tilde{\mathcal{R}}_{\text{trm}}^i = \{\tilde{r}_g^i \mid 1 \leq g \leq G, i \leq n_{\tau_g}\}, \quad \tilde{\mathcal{R}}_{\text{out}} = \left\{ \tilde{r}_g^{n_{\tau_g}+1} \mid 1 \leq g \leq G \right\}. \quad (8)$$

232 We then perform turn-level advantage estimation, where rewards for each turn are normalized inde-
 233 pendently across trajectories (Figure 2-b). In detail, for each turn i and trajectory τ_g , the normalized
 234 rewards are computed as
 235

$$236 \quad \hat{r}_g^i = \frac{\tilde{r}_g^i - \text{mean}(\tilde{\mathcal{R}}_{\text{trm}}^i)}{\text{std}(\tilde{\mathcal{R}}_{\text{trm}}^i)} \quad (1 \leq i \leq n_{\tau_g}), \quad \hat{r}_g^{n_{\tau_g}+1} = \frac{\tilde{r}_g^{n_{\tau_g}+1} - \text{mean}(\tilde{\mathcal{R}}_{\text{out}})}{\text{std}(\tilde{\mathcal{R}}_{\text{out}})}. \quad (9)$$

240 These normalized rewards are then assigned to the corresponding tokens, and token-level advantages
 241 are computed via discounted aggregation:
 242

$$243 \quad r_g^j = \begin{cases} \alpha \cdot \hat{r}_g^{\mathcal{I}(j)}, & j \in \mathcal{E}_g \\ \hat{r}_g^{n_{\tau_g}+1}, & j = L_g \\ 0, & \text{otherwise} \end{cases}, \quad A_g^j = r_g^{L_g} + \sum_{m=j}^{L_g-1} \gamma^{m-j} r_g^m, \quad (10)$$

246 where α is a weighting hyperparameter and γ is the discount factor⁵. With this token-level advan-
 247 tage, the GRPO objective is formulated as
 248

$$249 \quad \mathcal{L}_{\text{GRPO}} = \mathbb{E}_{\{\tau_g\} \sim \pi_{\theta}} \left[\frac{1}{G} \sum_{g=1}^G \frac{1}{|\mathcal{M}|} \sum_{j \in \mathcal{M}} \min \left(w_g^j(\theta) \cdot A_g^j, \text{clip} \left(w_g^j(\theta), 1 - \epsilon, 1 + \epsilon \right) A_g^j \right) \right]. \quad (11)$$

253 3 EXPERIMENTS

255 In this section, we focus on two key aspects:

- 257 • *TRM exploration*: How can we obtain an effective TRM?
- 258 • *TRM exploitation*: Does introducing TRM improve the tool-use capabilities of LLMs?

261 3.1 EXPLORATION OF TRM

262 **Training Model and Data** We use the Qwen2.5 (Qwen et al., 2025) series as the backbone archi-
 263 tecture for TRM. For training data, we sample 15K prompts each from the HotpotQA (Yang et al.,
 264 2018) and NQ (Kočiský et al., 2018) training sets. Rollouts are generated using DeepSeek-R1 (Guo
 265 et al., 2025), which interacts with a search environment (Jin et al., 2025) to produce multi-turn tra-
 266 jectories. Each trajectory is annotated with turn-level utility labels based on necessity and quality
 267 by DeepSeek-R1. Finally, we randomly sample 10K labeled trajectories for TRM training. More
 268 training details are in Appendix B.1.

269 ⁵Masked tokens are skipped when computing the discounted sum of normalized rewards.

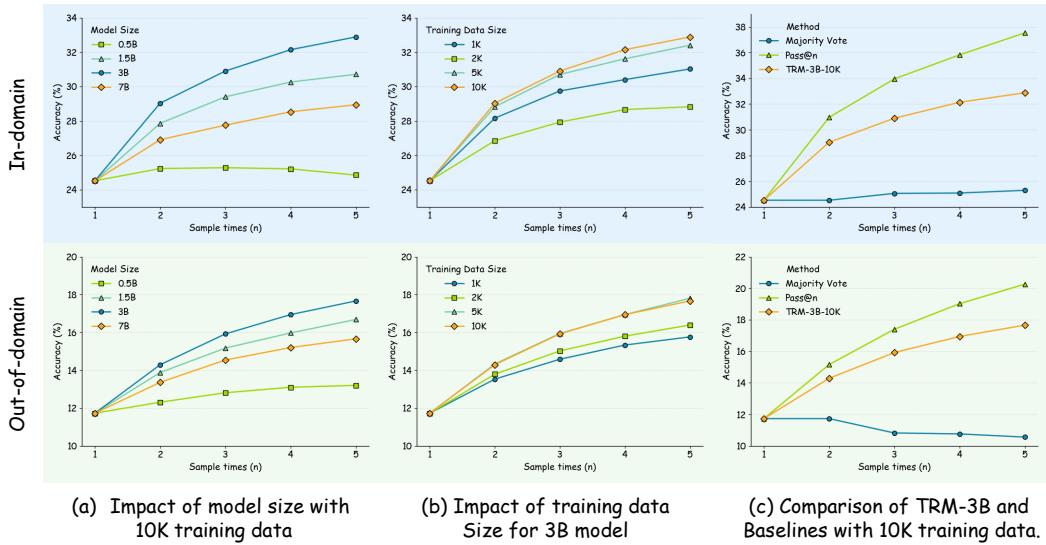


Figure 3: TRM performance comparison under different settings

Evaluation To evaluate TRM effectiveness, we use a best-of-n strategy (Lightman et al., 2024; Luo et al., 2025): for each prompt, n candidate trajectories are generated, and the one with the highest TRM score is selected. The score for a trajectory τ is computed as the product of all tool call scores, i.e., $S(\tau) = \prod_{i=1}^{n_\tau+1} \tilde{s}^i$.⁶ Evaluation is conducted in both in-domain (HotpotQA validation prompts) and out-of-domain (2WikiMultiHopQA (Ho et al., 2020) validation prompts) settings. All candidate trajectories are generated by the Search-R1 (Jin et al., 2025) model, which is PPO-trained based on Qwen2.5-7B. More details are in Appendix C.1.

Results and Analysis According to the results in Figure 3, we observe following key trends:

Key Takeaways for TRM Exploration

- Mid-sized TRMs (1.5B/3B) deliver optimal performance with 10K training samples, while larger models (e.g., 7B) may be prone to overfitting given the same data scale.
- 10K labeled trajectories are sufficient to achieve robust TRM training and stable results.
- TRM consistently outperforms the majority vote baseline, though there remains a gap to the upper bound established by pass@ n .

3.2 EXPLOITATION OF TRM

Setup We conduct experiments in two distinct scenarios: (1) answering questions using a search tool, and (2) solving math problems by writing Python code. Following prior works (Jin et al., 2025; Li et al., 2025b), for the search-based QA task, we evaluate on both Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct models; for the code-based math task, we utilize Qwen2.5-Math-1.5B and Qwen2.5-Math-7B (Yang et al., 2024). The training data for each scenario are also sourced from the corresponding prior works to ensure consistency and comparability. The search tool is allowed up to 5 rounds per query, while the code tool can be invoked up to 3 times per problem. All implementations are based on the Verl (Sheng et al., 2025; Zhang et al., 2024) framework. We set $\alpha = 0.05$ for PPO and $\alpha = 0.01$ for GRPO. Additional training details are provided in Appendix B.2.

Evaluation For the search scenario, we evaluate performance on both general QA datasets (NQ (Kočiský et al., 2018), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023)) and

⁶ $\tilde{s}^{n_\tau+1}$ indicates the correctness score for the final answer as predicted by the TRM.

324
325
326
Table 1: Performance of Qwen2.5 variants with different methods on various QA tasks. **Best** results
are in bold; second best are underlined.

Method	General QA				Multi-Hop QA			Avg.
	NQ	TriviaQA	PopQA	HotpotQA	2wiki	Musique	Bamboogle	
Qwen2.5-3B-Instruct								
Direct Inference	12.08	32.44	13.08	15.98	24.75	2.19	2.40	14.70
IRCOT	26.32	49.47	33.28	24.33	16.19	4.43	19.20	24.75
RAG	37.29	56.05	40.60	26.31	23.08	5.17	6.40	27.84
SFT	27.53	31.37	12.26	20.70	26.28	6.25	11.20	19.37
R1-PPO	19.09	42.16	16.35	19.04	25.91	3.14	8.00	19.10
R1-GRPO	25.10	45.89	18.10	21.55	27.98	5.42	20.00	23.43
Search-R1-PPO	36.93	54.48	35.85	32.65	32.47	12.08	24.80	32.75
Search-R1-PPO-TRM (ours)	39.58	57.78	40.61	34.80	33.22	12.91	25.60	34.93
Search-R1-GRPO	<u>47.01</u>	<u>61.88</u>	<u>45.73</u>	<u>43.34</u>	<u>42.68</u>	<u>18.08</u>	<u>37.60</u>	<u>42.33</u>
Search-R1-GRPO-TRM (ours)	47.89	62.57	47.20	44.47	43.48	19.65	39.20	43.49
Qwen2.5-7B-Instruct								
Direct Inference	14.29	43.69	15.10	19.23	25.54	3.68	10.40	18.85
IRCOT	18.23	50.31	30.33	21.61	8.73	4.05	17.60	21.55
RAG	34.88	58.96	39.45	30.16	23.62	5.50	21.60	30.59
SFT	31.97	34.00	12.36	22.23	26.40	9.72	10.40	21.01
R1-PPO	22.13	49.60	17.51	22.31	28.15	6.95	30.40	25.29
R1-GRPO	31.61	53.69	21.60	24.96	27.47	8.77	32.00	28.59
Search-R1-PPO	40.86	61.42	40.15	37.84	35.27	14.81	35.20	37.94
Search-R1-PPO-TRM (ours)	43.99	61.18	41.56	39.11	37.76	17.63	38.40	39.95
Search-R1-GRPO	<u>49.97</u>	<u>66.81</u>	<u>47.59</u>	<u>49.06</u>	47.80	22.30	44.80	46.90
Search-R1-GRPO-TRM (ours)	52.11	66.90	48.52	51.32	<u>47.67</u>	24.99	48.80	48.62

348
349
350 multi-hop QA datasets (HotpotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020), Musique (Trivedi
351 et al., 2022), Bamboogle (Press et al., 2023)). For the code-writing scenario, evaluation is con-
352 ducted on AIME24, AIME25, MATH500 (Hendrycks et al., 2021), Olympiad (He et al., 2024), and
353 AMC23. More evaluation details are in Appendix C.2.

354
355 **Baselines** For both search and code scenarios, we consider: (1) Direct Inference, which an-
356 swers questions without any tool usage; (2) SFT, supervised fine-tuning without tool usage; and (3)
357 R1-PPO/R1-GRPO, models trained with PPO or GRPO using outcome-only rewards, without tool
358 usage. Additional baselines for the search scenario include: (1) RAG, which retrieves relevant in-
359 formation once before answering; (2) IRCOT, iterative retrieval based on previous results; and (3)
360 Search-R1-PPO/Search-R1-GRPO, trained with PPO or GRPO and allowed to use the search tool.
361 For the code scenario, we further include: (1) Instruct, direct inference with the instruct version
362 of Qwen2.5-Math models; (2) Instruct+PAL (Gao et al., 2023), [generating programs as the inter-](#)
363 [mediate reasoning steps](#); and (3) ToRL-PPO/ToRL-GRPO, trained with PPO or GRPO and allowed to
364 use the code tool. More details are shown in Appendix D.

365 **Results and Analysis** Table 1 and Table 2 summarize the performance of Qwen2.5 variants across
366 different QA and math tasks. Several key observations emerge:

Key Takeaways for TRM Exploitation

- *TRM consistently enhances model performance in both search and code scenarios, across various model sizes (1.5B, 3B, 7B) and training algorithms (PPO, GRPO), indicating that TRM substantially strengthens the ability of LLMs to effectively utilize external tools.*
- *Enabling LLMs to dynamically learn tool use yields notable gains, while reinforcement learning without tool integration leads to much lower performance. Importantly, TRM plays a critical role by helping models utilize tools more effectively.*
- *GRPO generally outperforms PPO in our experiments; however, integrating TRM reliably boosts performance for both approaches.*

378 Table 2: Performance of Qwen2.5-Math variants with different methods on various math problems.
 379 **Best** results are in bold; second best are underlined.

Method	AIME24	AIME25	MATH500	Olympiad	AMC23	Avg.
<i>Qwen2.5-Math-1.5B</i>						
Direct Inference	7.78	1.11	67.80	28.30	35.00	28.00
Instruct	10.67	7.22	72.60	36.59	57.50	36.92
Instruct+PAL	<u>34.44</u>	<u>0.00</u>	<u>21.80</u>	<u>10.07</u>	<u>17.50</u>	<u>16.76</u>
SFT	0.00	0.00	15.40	7.11	27.50	10.00
R1-PPO	11.00	10.00	74.80	33.48	55.00	36.86
R1-GRPO	14.11	3.67	73.40	31.70	57.50	36.08
TorL-PPO	19.11	13.89	75.80	43.56	55.00	41.47
TorL-PPO-TRM (ours)	26.00	<u>19.89</u>	75.80	45.78	50.00	<u>43.49</u>
TorL-GRPO	25.56	19.33	75.80	45.19	50.00	43.18
TorL-GRPO-TRM (ours)	26.00	27.00	75.80	45.78	52.50	45.42
<i>Qwen2.5-Math-7B</i>						
Direct Inference	12.22	6.67	69.80	30.96	40.00	31.93
Instruct	5.11	8.11	79.60	37.33	52.50	36.53
SFT	0.00	0.00	12.80	5.19	42.50	12.10
R1-PPO	28.11	10.11	77.40	37.93	65.00	43.71
R1-GRPO	21.00	9.78	78.00	37.93	67.50	42.84
TorL-PPO	32.56	23.11	82.60	53.04	67.50	51.76
TorL-PPO-TRM (ours)	34.33	26.56	<u>83.40</u>	52.44	<u>70.00</u>	<u>53.35</u>
TorL-GRPO	35.00	21.89	83.80	<u>52.74</u>	67.50	52.19
TorL-GRPO-TRM (ours)	36.56	<u>23.67</u>	83.20	52.59	72.50	53.70

406 3.3 ADDITIONAL ANALYSIS

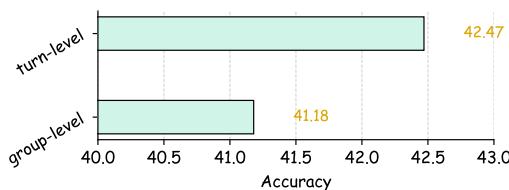
408 In this section, we provide further analysis on several key factors related to TRM exploitation **and**
 409 **some ablation studies**.

411 **Effect of Hyperparameter α** Figure 5-a shows that for PPO, a very small α limits the effect of
 412 TRM, while a very large α overemphasizes tool use. A moderate α balances final performance and
 413 reasonable tool invocation. Figure 5-b shows a similar trend for GRPO. We therefore set $\alpha = 0.05$
 414 for PPO and $\alpha = 0.01$ for GRPO in our experiments.

416 **Improvement of Tool-Use Generalization by TRM** We investigate the generalization ability of
 417 LLMs in tool-use scenarios. Specifically, we evaluate models trained in the search scenario on their
 418 ability to use Python code for solving mathematical problems. As shown in Figure 5-c, introducing
 419 TRM significantly improves generalization in tool invocation across different scenarios.

421 **Effect of Turn-Level Advantage Estimation in GRPO** Unlike turn-level estimation,
 422 which normalizes rewards for each turn individually, group-level estimation normalizes all
 423 tool-call rewards within a group together (Shao
 424 et al., 2024). As shown in Figure 4, turn-
 425 level advantage estimation achieves better per-
 426 formance than group-level estimation.

428 **Comparison with other process-supervised
 429 tool-use methods** We compare our method with two representative process-supervised baselines:
 430 StepSearch (Wang et al., 2025b), which is tailored for search-based QA and evaluates intermediate
 431 search queries for relevance and information gain, and AgentPRM (Choudhury, 2025), a general



432 Figure 4: Comparison of group-level and turn-
 433 level advantage estimation in GRPO

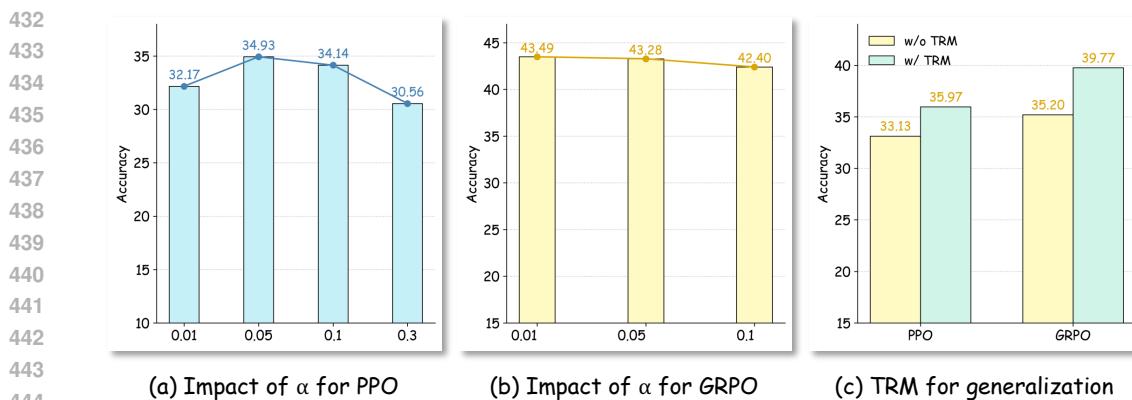


Figure 5: Summary of key analysis results. Subfigures (a) and (b) present the influence of the hyperparameter α on PPO and GRPO in conjunction with TRM. Subfigure (c) demonstrates that TRM improves the generalization capability of LLM for tool-use.

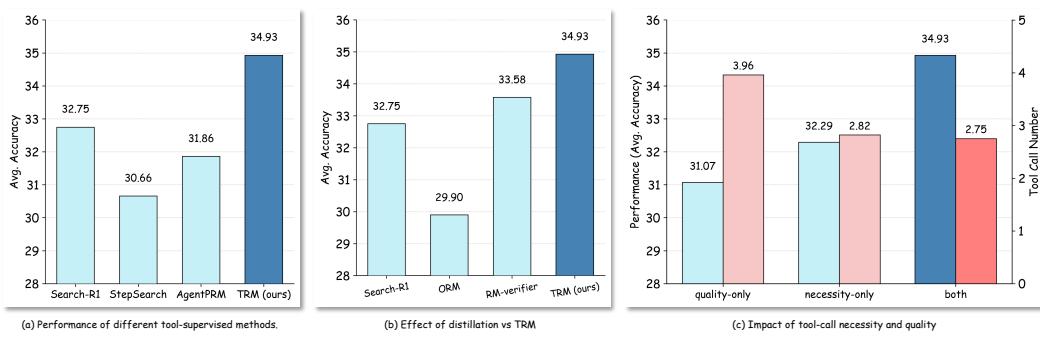


Figure 6: Performance comparisons and ablations for tool-supervised methods.

process-supervised method that labels tool calls based on whether they can eventually lead to a correct answer. Figure 6-a shows our method consistently outperforms both baselines in the search scenario over Qwen2.5-3B-Instruct with PPO, highlighting the advantage of our per-tool-call reward modeling over hand-crafted or generic process supervision signals.

Ablation study to disambiguate distillation and TRM To separate the effects of distillation from TRM, we introduce two baselines: ORM, which scores entire trajectories, and TRM used as a verifier, which aggregates per-tool-call scores. Figure 6-b shows that trajectory-level ORM underperforms the answer-only baseline in the search scenario over Qwen2.5-3B-Instruct with PPO, likely because scoring entire trajectories introduces additional noise. TRM as a verifier improves slightly but still lags behind full TRM, suggesting that fine-grained per-tool-call evaluation is essential for guiding the model effectively and fully leveraging the distillation data.

Ablation study on the necessity and quality of tool calls We evaluate the impact of tool-call necessity and quality on model performance and tool usage. Figure 6-c shows that using quality-only yields the lowest performance, likely due to excessive tool usage that introduces noise. Necessity-only reduces the number of tool calls but may compromise the quality of each call, limiting overall effectiveness. Combining both necessity and quality achieves the best performance while maintaining a relatively stable number of tool calls across datasets, suggesting that balancing necessity and quality is important for efficient and effective tool use.

4 RELATED WORK

Process Reward Model Reward models have been widely adopted in various reasoning tasks to supervise output quality, such as mathematical problem-solving (Uesato et al., 2023; Shao et al., 2024; Zhang et al., 2025a). These models are generally divided into outcome reward models (ORMs), which provide holistic evaluations, and process reward models (PRMs), which offer fine-grained, step-level assessments. PRMs have shown strong effectiveness (Lightman et al., 2024; Wang et al., 2024; Luo et al., 2024; Cheng et al., 2025), especially in mathematical problem-solving, and have been used both for guiding inference (e.g., best-of-n selection) and for supervising post-training. By providing more granular feedback, process reward models enable models to learn more interpretable and robust reasoning strategies. However, most existing work on PRMs focuses on traditional reasoning tasks, with limited exploration in tool-use scenarios. In this work, we introduce the Tool-call Reward Model (TRM), specifically designed for tool-invocation of LLMs, and conduct a comprehensive study on both the exploration and exploitation of TRM. Our approach aims to extend process-level supervision to agentic tasks, enabling more effective and flexible tool usage in LLMs.

Agentic RL for LLM Tool Invocation Recent advances in outcome-based RL have enabled LLMs to achieve impressive performance in agentic reasoning tasks (Guo et al., 2025; Hu et al., 2025). This paradigm has spurred active research in tool invocation for LLMs, with works such as Search-R1 (Jin et al., 2025), ReSearch (Chen et al., 2025a), R1-Searcher (Song et al., 2025), DeepResearcher (Zheng et al., 2025), WebRL (Qi et al., 2024), WebThinker (Li et al., 2025a), ZeroSearch (Sun et al., 2025), ToRL (Li et al., 2025b), and ToolRL (Qian et al., 2025b) extending outcome-supervised RL to scenarios where LLMs autonomously utilize search engines or code execution for complex reasoning and problem-solving. While these methods have improved agentic capabilities, the reward signals are typically coarse-grained, focusing only on final outcomes and providing limited guidance for efficient tool-use or search strategies. Atom-Searcher (Deng et al., 2025a) and StepSearch (Wang et al., 2025a) further consider intermediate tool-use steps by leveraging existing large models or rule-based approaches. In contrast, our work designs and develops a dedicated TRM to explicitly monitor and supervise intermediate tool invocations, and validates its effectiveness on both search and code-generation scenarios.

5 CONCLUSION

We present the Tool-call Reward Model (TRM), a special process reward model that provides fine-grained supervision for tool invocation in large language models. TRM enables more precise credit assignment for each tool call, mitigating issues with outcome-only reward signals such as gradient conflict. We systematically study TRM construction and propose effective integration strategies with classical RL algorithms, including turn-level credit assignment and advantage estimation. Experiments on search-based QA and code-based math tasks show that TRM consistently improves tool usage and generalization across various model sizes and RL methods. Our findings demonstrate that robust TRM performance can be achieved with moderate model sizes and limited training data. We believe TRM offers a promising direction for advancing agentic capabilities in LLMs.

REFERENCES

- Bytedance-Seed-Foundation-Code-Team, :, Yao Cheng, Jianfeng Chen, Jie Chen, Li Chen, Liyu Chen, Wentao Chen, Zhengyu Chen, Shijie Geng, Aoyan Li, Bo Li, Bowen Li, Linyi Li, Boyi Liu, Jiaheng Liu, Kaibo Liu, Qi Liu, Shukai Liu, Siyao Liu, Tianyi Liu, Tingkai Liu, Yongfei Liu, Rui Long, Jing Mai, Guanghan Ning, Z. Y. Peng, Kai Shen, Jiahao Su, Jing Su, Tao Sun, Yifan Sun, Yunzhe Tao, Guoyin Wang, Siwei Wang, Xuwu Wang, Yite Wang, Zihan Wang, Jinxiang Xia, Liang Xiang, Xia Xiao, Yongsheng Xiao, Chenguang Xi, Shulin Xin, Jingjing Xu, Shikun Xu, Hongxia Yang, Jack Yang, Yingxiang Yang, Jianbo Yuan, Jun Zhang, Yufeng Zhang, Yuyu Zhang, Shen Zheng, He Zhu, and Ming Zhu. Fullstack bench: Evaluating llms as full stack coders, 2025. URL <https://arxiv.org/abs/2412.00535>.
- Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. Research: Learning to

- 540 reason with search for llms via reinforcement learning, 2025a. URL <https://arxiv.org/abs/2503.19470>.
- 541
- 542
- 543 Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan,
544 Wen Zhang, Huajun Chen, Fan Yang, et al. Learning to reason with search for llms via reinforce-
545 ment learning. *arXiv preprint arXiv:2503.19470*, 2025b.
- 546
- 547 Jie Cheng, Ruixi Qiao, Lijun Li, Chao Guo, Junle Wang, Gang Xiong, Yisheng Lv, and Fei-Yue
548 Wang. Stop summation: Min-form credit assignment is all process reward model needs for rea-
549 soning, 2025. URL <https://arxiv.org/abs/2504.15275>.
- 550
- 551 Sanjiban Choudhury. Process reward models for llm agents: Practical framework and directions,
552 2025. URL <https://arxiv.org/abs/2502.10325>.
- 553
- 554 Yong Deng, Guoqing Wang, Zhenzhe Ying, Xiaofeng Wu, Jinzhen Lin, Wenwen Xiong, Yuqin Dai,
555 Shuo Yang, Zhanwei Zhang, Qiwen Wang, Yang Qin, Yuan Wang, Quanxing Zha, Sunhao Dai,
556 and Changhua Meng. Atom-searcher: Enhancing agentic deep research via fine-grained atomic
557 thought reward, 2025a. URL <https://arxiv.org/abs/2508.12800>.
- 558
- 559 Yong Deng, Guoqing Wang, Zhenzhe Ying, Xiaofeng Wu, Jinzhen Lin, Wenwen Xiong, Yuqin Dai,
560 Shuo Yang, Zhanwei Zhang, Qiwen Wang, et al. Atom-searcher: Enhancing agentic deep research
561 via fine-grained atomic thought reward. *arXiv preprint arXiv:2508.12800*, 2025b.
- 562
- 563 Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang,
564 Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms,
565 2025. URL <https://arxiv.org/abs/2504.11536>.
- 566
- 567 Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and
568 Graham Neubig. Pal: program-aided language models. In *Proceedings of the 40th International
569 Conference on Machine Learning*, ICML’23. JMLR.org, 2023.
- 570
- 571 Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu,
572 Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms
573 via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- 574
- 575 Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong
576 Sun, and Yang Liu. StableToolBench: Towards stable large-scale benchmarking on tool learning
577 of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of
578 the Association for Computational Linguistics: ACL 2024*, pp. 11143–11156, Bangkok, Thailand,
579 August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.
580 664. URL <https://aclanthology.org/2024.findings-acl.664>.
- 581
- 582 Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han,
583 Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench:
584 A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific
585 problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd
586 Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp.
587 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi:
588 10.18653/v1/2024.acl-long.211. URL <https://aclanthology.org/2024.acl-long.211>.
- 589
- 590 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn
591 Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset.
592 In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks
593 Track (Round 2)*, 2021. URL <https://openreview.net/forum?id=7Bywt2mQsCe>.
- 594
- 595 Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-
596 hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel,
597 and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational
598 Linguistics*, pp. 6609–6625, Barcelona, Spain (Online), December 2020. International Com-
599 mittee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL <https://aclanthology.org/2020.coling-main.580>.

- 594 Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. Reinforce++: An efficient rlhf algorithm with
 595 robustness to both prompt and reward models. *arXiv preprint arXiv:2501.03262*, 2025.
- 596
- 597 Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Za-
 598 mani, and Jiawei Han. Search-rl: Training LLMs to reason and leverage search engines with
 599 reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL <https://openreview.net/forum?id=Rwhi91ideu>.
- 600
- 601 Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly
 602 supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan
 603 (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for
 604 Computational Linguistics. doi: 10.18653/v1/P17-1147. URL <https://aclanthology.org/P17-1147>.
- 605
- 606
- 607 Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi
 608 Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie
 609 Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on
 610 Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November
 611 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550.
 612 URL <https://aclanthology.org/2020.emnlp-main.550>.
- 613
- 614 Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis,
 615 and Edward Grefenstette. The NarrativeQA reading comprehension challenge. *Transactions of
 616 the Association for Computational Linguistics*, 6:317–328, 2018. doi: 10.1162/tacl_a_00023.
 617 URL <https://aclanthology.org/Q18-1023>.
- 618
- 619 Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and
 620 Zhicheng Dou. Webthinker: Empowering large reasoning models with deep research capability,
 621 2025a. URL <https://arxiv.org/abs/2504.21776>.
- 622
- 623 Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint
 arXiv:2503.23383*, 2025b.
- 624
- 625 Minpeng Liao, Chengxi Li, Wei Luo, Wu Jing, and Kai Fan. MARIO: MATH reasoning with code
 626 interpreter output - a reproducible pipeline. In Lun-Wei Ku, Andre Martins, and Vivek Sriku-
 627 mar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 905–924,
 628 Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/
 2024.findings-acl.53. URL <https://aclanthology.org/2024.findings-acl.53>.
- 629
- 630 Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan
 631 Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The Twelfth
 632 International Conference on Learning Representations*, 2024. URL [https://openreview.net/forum?id=KwPUQOQIKt](https://openreview.net/

 633 forum?id=v8L0pN6E0i.</p>
<p>634</p>
<p>635 Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun

 636 Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language

 637 models by automated process supervision. <i>arXiv preprint arXiv:2406.06592</i>, 2024.</p>
<p>638</p>
<p>639 Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li,

 640 Lei Shu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language

 641 models with automated process supervision, 2025. URL <a href=).
- 642
- 643 Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi.
 644 When not to trust language models: Investigating effectiveness of parametric and non-parametric
 645 memories. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the
 646 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*,
 647 pp. 9802–9822, Toronto, Canada, July 2023. Association for Computational Linguistics. doi:
 10.18653/v1/2023.acl-long.546. URL <https://aclanthology.org/2023.acl-long.546>.
- 648
- 649 OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

- 648 Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring
 649 and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino,
 650 and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP*
 651 2023, pp. 5687–5711, Singapore, December 2023. Association for Computational Linguis-
 652 tics. doi: 10.18653/v1/2023.findings-emnlp.378. URL <https://aclanthology.org/2023.findings-emnlp.378/>.
- 653
- 654 Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue
 655 Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. Webrl:
 656 Training llm web agents via self-evolving online curriculum reinforcement learning. *CoRR*,
 657 abs/2411.02337, 2024. URL <http://dblp.uni-trier.de/db/journals/corr/corr2411.html#abs-2411-02337>.
- 658
- 659 Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan
 660 Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*,
 661 2025a.
- 662
- 663 Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan
 664 Tur, and Heng Ji. Toolrl: Reward is all tool learning needs, 2025b. URL <https://arxiv.org/abs/2504.13958>.
- 665
- 666 Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan
 667 Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang,
 668 Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin
 669 Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li,
 670 Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang,
 671 Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.
 672 URL <https://arxiv.org/abs/2412.15115>.
- 673
- 674 Timo Schick, Jane Dwivedi-Yu, Roberto Dessí, Roberta Raileanu, Maria Lomeli, Eric Hambro,
 675 Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: language models can
 676 teach themselves to use tools. In *Proceedings of the 37th International Conference on Neural*
 677 *Information Processing Systems*, NIPS ’23, Red Hook, NY, USA, 2023. Curran Associates Inc.
- 678
- 679 John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-
 680 dimensional continuous control using generalized advantage estimation. In Yoshua Bengio and
 681 Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016,*
 682 *San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL <http://arxiv.org/abs/1506.02438>.
- 683
- 684 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy
 685 optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- 686
- 687 Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal,
 688 Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated
 689 process verifiers for LLM reasoning. In *The Thirteenth International Conference on Learning*
 690 *Representations*, 2025. URL <https://openreview.net/forum?id=A6Y7AqlzLW>.
- 691
- 692 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,
 693 Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical
 694 reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>, 2(3):5, 2024.
- 695
- 696 Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng,
 697 Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlf framework. In *Proceedings*
 698 *of the Twentieth European Conference on Computer Systems*, EuroSys ’25, pp. 1279–1297, New
 699 York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400711961. doi: 10.
 1145/3689031.3696075. URL <https://doi.org/10.1145/3689031.3696075>.
- 700
- 701 Huatong Song, Jinshao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang,
 702 and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement
 703 learning. *arXiv preprint arXiv:2503.05592*, 2025.

- 702 Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang,
 703 Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without
 704 searching, 2025. URL <https://arxiv.org/abs/2505.04588>.
- 705 Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen,
 706 Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv*
 707 preprint arXiv:2507.20534, 2025.
- 708 Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL <https:////qwenlm.github.io/blog/qwq-32b/>.
- 709 Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-
 710 hop questions via single-hop question composition. *Transactions of the Association for Compu-
 711 tational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL <https://aclanthology.org/2022.tacl-1.31/>.
- 712 Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Yamamoto Siegel, Lisa
 713 Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with
 714 process-based and outcome-based feedback, 2023. URL <https://openreview.net/forum?id=MND1kmmNy00>.
- 715 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N
 716 Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon,
 717 U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Gar-
 718 nett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Asso-
 719 ciates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fdb053c1c4a845aa-Paper.pdf.
- 720 Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Dixin Jiang, Rangan Ma-
 721 jumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv*
 722 preprint arXiv:2212.03533, 2022.
- 723 Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and
 724 Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human anno-
 725 tations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd*
 726 *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp.
 727 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi:
 728 10.18653/v1/2024.acl-long.510. URL <https://aclanthology.org/2024.acl-long.510/>.
- 729 Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu.
 Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. *arXiv preprint*
 730 arXiv:2505.15107, 2025a.
- 731 Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu.
 Stepsearch: Igniting llms search ability via step-wise proximal policy optimization, 2025b. URL
 732 <https://arxiv.org/abs/2505.15107>.
- 733 An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jian-
 734 hong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical
 735 expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.
- 736 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu,
 737 Chang Gao, Chengan Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint*
 738 arXiv:2505.09388, 2025.
- 739 Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov,
 740 and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question
 741 answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun’ichi Tsujii (eds.), *Proce-
 742 dings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–
 743 2380, Brussels, Belgium, October–November 2018. Association for Computational Linguistics.
 744 doi: 10.18653/v1/D18-1259. URL <https://aclanthology.org/D18-1259/>.

- 756 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan
 757 Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International*
 758 *Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.
- 760 Chi Zhang, Guangming Sheng, Siyao Liu, Jiahao Li, Ziyuan Feng, Zherui Liu, Xin Liu, Xiaoying
 761 Jia, Yanghua Peng, Haibin Lin, et al. A framework for training large language models for code
 762 generation via proximal policy optimization. In *NL2Code Workshop of ACM KDD*, 2024.
- 763 Yuxin Zhang, Meihao Fan, Ju Fan, Mingyang Yi, Yuyu Luo, Jian Tan, and Guoliang Li. Reward-
 764 sql: Boosting text-to-sql via stepwise reasoning and process-supervised rewards, 2025a. URL
 765 <https://arxiv.org/abs/2505.04671>.
- 766 Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu,
 767 Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathe-
 768 matical reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher
 769 Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–
 770 10516, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-
 771 8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.547. URL <https://aclanthology.org/2025.findings-acl.547/>.
- 772 Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyuan Luo, Zhangchi Feng, and
 773 Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Pro-
 ceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume*
 774 *3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguis-
 775 tics. URL <http://arxiv.org/abs/2403.13372>.
- 776 Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei
 777 Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environ-
 778 ments, 2025. URL <https://arxiv.org/abs/2504.03160>.
- 779
- 780
- 781
- 782
- 783
- 784
- 785
- 786
- 787
- 788
- 789
- 790
- 791
- 792
- 793
- 794
- 795
- 796
- 797
- 798
- 799
- 800
- 801
- 802
- 803
- 804
- 805
- 806
- 807
- 808
- 809

810 THE USE OF LARGE LANGUAGE MODELS 811

812 In this work, large language models were used solely for language polishing and improving the
813 clarity of the manuscript. The LLMs did not participate in any substantive aspects of the research,
814 including problem definition, research motivation, methodology, experimental design, or analysis.
815 All scientific contributions, conceptual developments, and experimental results were conducted and
816 validated by the authors without the involvement of LLMs in the core research process.

818 ETHICS STATEMENT 819

820 This work does not involve sensitive personal data, or practices that raise privacy or security con-
821 cerns. All datasets used are publicly available and do not contain personally identifiable information.
822 The research does not present potentially harmful methodologies, applications, or insights, and does
823 not raise issues related to discrimination, bias, or fairness. The authors have adhered to the ICLR
824 Code of Ethics throughout the research and submission process.

826 REPRODUCIBILITY STATEMENT 827

828 All experimental details are provided in the main text (§ 3) and Appendix to ensure reproducibility.
829 Key code components have been submitted with this paper, and the complete codebase will be
830 released publicly at an appropriate time.

832 LIMITATIONS AND IMPACTS 833

834 While the Tool-call Reward Model (TRM) demonstrates significant improvements in tool-use super-
835 vision for large language models, several limitations remain. First, our study is scoped to tasks with
836 verifiable final outcomes (e.g., factual QA and code generation), as our primary focus is to address
837 the limitations of outcome-only reward RL in such settings. Extending TRM to open-ended (Guo
838 et al., 2024) reinforcement learning, where correctness is difficult to assess, would require additional
839 mechanisms for outcome evaluation and is left for future work. Second, to keep rollouts manage-
840 able and reduce judge bias, we truncate trajectories to a moderate length, and our current framework
841 does not fully resolve the challenge of providing reliable process supervision for very long tool-use
842 trajectories. Finally, TRM models tool utility via a simple binary necessity-quality decomposition,
843 which may be insufficient to capture more nuanced, multi-objective notions of tool usefulness in
844 complex domains.

845 Despite these areas for improvement, TRM provides fine-grained supervision that enables more
846 interpretable and robust tool usage, advancing the agentic capabilities of large language models. This
847 approach can facilitate safer and more reliable deployment of LLMs in real-world tasks requiring
848 external tool invocation, and we hope our work inspires further research in process-level reward
849 modeling.

850
851
852
853
854
855
856
857
858
859
860
861
862
863

864
865

A PROMPTS

866
867

A.1 PROMPTS FOR TRM TRAINING DATA DISTILLATION

868
869

Prompt of Tool Call Evaluation for Search Scenario

870
871

```
## TASK
You are a professional Tool Call Evaluator for AI agent trajectories. For a given
    ↪ user question and its complete step-by-step trajectory, review every tool
    ↪ call (all are of type 'search') and assess each using the following
    ↪ evaluation dimensions:
- Tool Selection Accuracy
- correct (1): It is appropriate to use the 'search' tool for this subtask;
    ↪ this call is necessary to make progress.
- incorrect (0): Using 'search' is not appropriate here (the information is
    ↪ already available, the call is unnecessary, or it does not help answer
    ↪ the user's question).
- Query Quality
- perfect (1): The 'query' is clear, directly addresses the user's need, and
    ↪ uses precise wording.
- minor or major error (0):
    - minor error: There is some ambiguity or slight irrelevance, but the search
        ↪ will likely still provide useful results.
    - major error: The query is unclear or unrelated to the user's actual need.

## INSTRUCTIONS
- Evaluate every tool call (all are 'search') on both dimensions.
- Briefly justify each score you assign.

## INPUT FORMAT
You will receive:
- 'user_question': The original user question.
- 'trajectory': The full step-by-step trajectory as a list of steps.
- Each step includes:
    - 'step_id'
    - 'thought': The agent's reasoning or intention before making the search.
    - 'query': The search query issued.
    - 'response': The information returned from the search.

Example:
```
{
 "user_question": "What is the capital of France and the population of Germany
 ↪ in 2023?",
 "trajectory": [
 {
 "step_id": 0,
 "thought": "I need to find the capital of France.",
 "query": "capital of France",
 "response": "Paris is the capital of France."
 },
 {
 "step_id": 1,
 "thought": "Now I should get the population figure for Germany in 2023.",
 "query": "population of Germany 2023",
 "response": "The population of Germany in 2023 is estimated to be about 84
 ↪ million."
 }
]
}
```
... (continued in next page)
```

916

917

```

918
919             Prompt of Tool Call Evaluation for Search Scenario (continued)
920
921
922     ## OUTPUT FORMAT
923     Provide your evaluations as a JSON list.
924     For each step, output an object with:
925     - 'step_id'
926     - 'tool_selection_accuracy': 1 or 0
927     - 'tool_selection_justification': your brief justification
928     - 'query_quality': 1 or 0
929     - 'query_quality_justification': your brief justification
930     Example:
931     """
932     [
933         {
934             "step_id": 0,
935             "tool_selection_accuracy": 1,
936             "tool_selection_justification": "The user asked for the capital of France,  

937                                         ↪which is factual information requiring a search.",
938             "query_quality": 1,
939             "query_quality_justification": "The query is clear and directly requests the  

940                                         ↪needed information."
941         },
942         {
943             "step_id": 1,
944             "tool_selection_accuracy": 1,
945             "tool_selection_justification": "The user needs the population of Germany in  

946                                         ↪2023, which requires a search.",
947             "query_quality": 1,
948             "query_quality_justification": "The query is specific and unambiguous."
949         }
950     ]
951     """
952
953
954     ## INPUT
955     """
956     {input}
957     """
958
959
960
961
962
963
964
965
966
967
968
969
970
971

```

972
 973
 974
 975
 976
 977
 978
 979
 980
 981
 982
 983
 984
 985
 986
 987
 988
 989
 990
 991
 992
 993
 994
 995
 996
 997
 998
 999
 1000
 1001
 1002
 1003
 1004
 1005
 1006
 1007
 1008
 1009
 1010
 1011
 1012
 1013
 1014
 1015
 1016
 1017
 1018
 1019
 1020
 1021
 1022
 1023
 1024
 1025

Prompt of Tool Call Evaluation for Code Scenario

TASK
 You are a professional Tool Call Evaluator for AI agent trajectories. For a given user question and its complete step-by-step trajectory, review every tool call (all are of type ‘program’) and assess each using the following evaluation dimensions:
 - Tool Selection Accuracy
 - correct (1): It is appropriate to use the ‘program’ tool for this subtask;
 ↳ writing and executing a program is necessary or clearly helpful for making progress (e.g., for calculation, verification, or complex reasoning).
 - incorrect (0): Using ‘program’ is not appropriate here (the calculation or reasoning can be done easily by hand, the program is unnecessary, or it does not help answer the user’s question).
 - Code Quality
 - perfect (1): The code is complete, correct, and directly serves the intended purpose (e.g., correct imports, clear logic, no errors, and directly answers the subtask).
 - minor or major error (0):
 - minor error: The program has small issues (e.g., missing imports, minor inefficiency), but will likely still work as intended.
 - major error: The code is incomplete, incorrect, or does not address the intended purpose.
 Note:
 In this evaluation, it is acceptable for the program to be used for verifying or checking results that were derived by hand in previous reasoning steps. The code does not need to independently derive all intermediate parameters or replicate the full logical chain, as long as it correctly verifies or computes the intended result. This use of code for auxiliary verification is considered sufficient for a perfect score, provided the code is correct and complete.

INSTRUCTIONS
 - Evaluate every tool call (all are ‘program’) on both dimensions.
 - Briefly justify each score you assign.

INPUT FORMAT
 You will receive:
 - ‘user_question’: The original user question.
 - ‘trajectory’: The full step-by-step trajectory as a list of steps.
 - Each step includes:
 - ‘step_id’
 - ‘thought’: The agent’s reasoning or intention before programming.
 - ‘code’: The code issued (if any; otherwise may be empty).
 - ‘output’: The output from executing the code (if any; otherwise may be empty).

... (continued in next page)

Prompt of Tool Call Evaluation for Code Scenario (continued)

```

1026
1027
1028
1029 Example:
1030 """
1031 {{ "user_question": "What is the sum of the first 100 positive integers?", "trajectory": [
1032   {
1033     "step_id": 0,
1034     "thought": "I can use the formula for the sum of the first n integers, but
1035       ↪I'll write a program to verify the result.",
1036     "code": "n = 100\nresult = n * (n + 1) // 2\nprint(result)",
1037     "output": "5050"
1038   },
1039   {
1040     "step_id": 1,
1041     "thought": "Now I will write a program to sum the integers from 1 to 100
1042       ↪directly.",
1043     "code": "print(sum(range(1, 101)))",
1044     "output": "5050"
1045   }
1046 }}
1047 """
1048 ### OUTPUT FORMAT
1049 Provide your evaluations as a JSON list.
1050 For each step, output an object with:
1051 - 'step_id'
1052 - 'tool_selection_accuracy': 1 or 0
1053 - 'tool_selection_justification': your brief justification
1054 - 'code_quality': 1 or 0
1055 - 'code_quality_justification': your brief justification
1056 Example:
1057 """
1058 [
1059   {
1060     "step_id": 0,
1061     "tool_selection_accuracy": 1,
1062     "tool_selection_justification": "Using a program to verify the formula is
1063       ↪reasonable and helps ensure correctness.",
1064     "code_quality": 1,
1065     "code_quality_justification": "The program is correct, complete, and directly
1066       ↪ computes the required sum."
1067   },
1068   {
1069     "step_id": 1,
1070     "tool_selection_accuracy": 0,
1071     "tool_selection_justification": "Writing a second program to do the same
1072       ↪calculation in a different way is redundant and not necessary for
1073       ↪solving the user's question.",
1074     "code_quality": 1,
1075     "code_quality_justification": "The program is correct and concise, but does
1076       ↪not add value beyond the previous step."
1077   }
1078 ]
1079 """
1080 ### INPUT
1081 """
1082 {input}
1083 """
1084 ### OUTPUT
1085
1086
1087
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
1134
1135
1136
1137
1138
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
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
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
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
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
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3047
3047
3048
3049
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3139
3140
3141
3142
3143
3144
3145
3146
3147
3147
3148
3149
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3365
3366
3367
```

Prompt of Tool Call Evaluation for Multi-tool Scenario

TASK
You are a professional Tool Call Evaluator for AI agent trajectories.
For a given user question, a list of available tools (with descriptions and
parameter schemas),
and the complete step-by-step trajectory, review every tool call and assess each
using the
following evaluation dimensions:
- Tool Selection Accuracy
- correct (1):
- The chosen tool matches the intended subtask and is consistent with the
tool's description and schema.
- The call is necessary or clearly helpful for making progress toward
answering the user's question or fulfilling the user's request.
- incorrect (0):
- The chosen tool does **not** match the subtask (e.g., wrong tool given the
intention or user need).
- Or the call is redundant / unnecessary (e.g., the information is already
available from earlier steps, or the call does not help answer the user
's question).
- Argument Quality
- perfect (1):
- The arguments to the tool are correct, complete, and specific.
- They respect the tool's parameter schema (types, required fields) and align
with the user's need or the agent's stated intention.
- minor or major error (0):
- minor error:
- Small mismatch, ambiguity, or slight irrelevance in arguments that still
likely allows the tool to work and return useful results.
- major error:
- Missing required fields, wrong types, wrong values, or arguments that do
not actually reflect the intended subtask or the user request.
- The tool would likely fail, error, or return irrelevant / unusable
results.
INSTRUCTIONS
- Evaluate every tool call on both dimensions.
- Briefly justify each score you assign.
INPUT FORMAT
You will receive:
- 'user_question': The original user question.
- 'trajectory': The full step-by-step trajectory as a list of steps.
- Each step includes:
- 'step_id'
- 'thought': The agent's reasoning or intention before making the search.
- 'tool_calls': The tools invoked.
- 'response': The information returned from the tool calls.
- 'available_tools': A list of available tools with their descriptions and
parameter schemas.
... (continued in next page)

Prompt of Tool Call Evaluation for Multi-tool Scenario (continued)

1136
1137 Example:
1138
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

```
```  
{
 "user_question": "What is the capital of France and the population of Germany
 ↪in 2023?",
 "available_tools": [
 {
 "name": "search",
 "description": "Use this tool to search for factual information from the
 ↪web.",
 "parameters": {
 "type": "object",
 "properties": {
 "query": {
 "type": "string",
 "description": "The search query string."
 }
 },
 "required": ["query"]
 }
 }
],
 "trajectory": [
 {
 "step_id": 0,
 "thought": "I need to find the capital of France.",
 "tool_calls": [{
 "name": "search",
 "arguments": {
 "query": "capital of France"
 }
 }],
 "response": "Paris is the capital of France."
 },
 {
 "step_id": 1,
 "thought": "Now I should get the population figure for Germany in 2023.",
 "tool_calls": [{
 "name": "search",
 "arguments": {
 "query": "population of Germany 2023"
 }
 }],
 "response": "The population of Germany in 2023 is estimated to be about 84
 ↪million."
 }
]
}
```
```

... (continued in next page)

```

1188
1189             Prompt of Tool Call Evaluation for Multi-tool Scenario (continued)
1190
1191     ## OUTPUT FORMAT
1192     Provide your evaluations as a JSON list.
1193     For each step, output an object with:
1194     - 'step_id'
1195     - 'tool_selection_accuracy': 1 or 0
1196     - 'tool_selection_justification': your brief justification
1197     - 'argument_quality': 1 or 0
1198     - 'argument_quality_justification': your brief justification
1199     Example:
1200     """
1201     [
1202         {
1203             "step_id": 0,
1204             "tool_selection_accuracy": 1,
1205             "tool_selection_justification": "The user asked for the capital of France,
1206                 ↪which is factual information requiring a search.",
1207             "argument_quality": 1,
1208             "argument_quality_justification": "The query is clear and directly requests
1209                 ↪the needed information."
1210         },
1211         {
1212             "step_id": 1,
1213             "tool_selection_accuracy": 1,
1214             "tool_selection_justification": "The user needs the population of Germany in
1215                 ↪2023, which requires a search.",
1216             "argument_quality": 1,
1217             "argument_quality_justification": "The query is specific and unambiguous."
1218         }
1219     ]
1220     """
1221
1222     ## INPUT
1223     """
1224     {input}
1225     """
1226
1227
1228     ## OUTPUT
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241

```

A.2 SYSTEM PROMPTS FOR TASKS

```

1232             System Prompt for QA Tasks with Search Tool
1233
1234             Answer the given question. You must conduct reasoning inside <think> and </think>
1235                 ↪ first every time you get new information. After reasoning, if you find
1236                 ↪you lack some knowledge, you can call a search engine by <search> query </
1237                 ↪search> and it will return the top searched results between <tool_response
1238                 ↪> and </tool_response>. You can search as many times as you want. If you
1239                 ↪find no further external knowledge needed, you can directly provide the
1240                 ↪answer inside <answer> and </answer>, without detailed illustrations. For
1241                 ↪example, <answer> Beijing </answer>.

```

1242

1243

1244

1245 Answer the given question. You should first have a reasoning process in mind and
 1246 ↳ then provides the answer. Show your reasoning in <think> </think> tags and
 1247 ↳ return the final answer in <answer> </answer> tags, for example <answer>
 1248 ↳ Beijing </answer>.

1249

1250

1251

1252

1253

1254

1255 Solve the following problem step by step. You now have the ability to selectively
 1256 ↳ write executable Python code to enhance your reasoning process. The
 1257 ↳ Python code will be executed by an external sandbox, and the output (‘
 1258 ↳ wrapped in ‘<tool_response>output_str</tool_response>’) can be returned to
 1259 ↳ aid your reasoning and help you arrive at the final answer. The Python
 1260 ↳ code should be complete scripts, including necessary imports. Put your
 1261 ↳ final answer within \\boxed{ }.

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275 In this environment you have access to a set of tools you can use to assist with
 1276 ↳ the user query. You may perform multiple rounds of function calls. In each
 1277 ↳ round, you can call one or more functions.

1278

1279 Here are available functions in JSONSchema format:

```
1280   ````json
1281   {func_schemas}
1282   ````
```

1283

1284 In your response, you need to first think about the reasoning process in the mind
 1285 ↳ and then conduct function calling to get the information or perform the
 1286 ↳ actions if needed. The reasoning process and function calling are enclosed
 1287 ↳ within <think> </think> and <tool_call> </tool_call> tags. The results of
 1288 ↳ the function calls will be given back to you after execution, and you can
 1289 ↳ continue to call functions until you can provide the final answer
 1290 ↳ enclosed within <answer> </answer> tags for the user’s question.

1291

1292 Tool call example:
 1293 <tool_call>
 1294 {{"name": <function-name>, "arguments": <args-json-object>}}
 1295 ...
 1296 </tool_call>

1297 Final answer example:
 1298 <answer> ... </answer>

System Prompt for QA Tasks without Search Tool

Answer the given question. You should first have a reasoning process in mind and
 ↳ then provides the answer. Show your reasoning in <think> </think> tags and
 ↳ return the final answer in <answer> </answer> tags, for example <answer>
 ↳ Beijing </answer>.

System Prompt for Mathematical Problems with Code Tool

Solve the following problem step by step. You now have the ability to selectively
 ↳ write executable Python code to enhance your reasoning process. The
 ↳ Python code will be executed by an external sandbox, and the output (‘
 ↳ wrapped in ‘<tool_response>output_str</tool_response>’) can be returned to
 ↳ aid your reasoning and help you arrive at the final answer. The Python
 ↳ code should be complete scripts, including necessary imports. Put your
 ↳ final answer within \\boxed{ }.

System Prompt for Mathematical Problems without Code Tool

Please reason step by step, and put your final answer within \\boxed{ }.

System Prompt for Real-world Problems with Multi Tools

In this environment you have access to a set of tools you can use to assist with
 ↳ the user query. You may perform multiple rounds of function calls. In each
 ↳ round, you can call one or more functions.

Here are available functions in JSONSchema format:

```
   ````json
{func_schemas}
````
```

In your response, you need to first think about the reasoning process in the mind
 ↳ and then conduct function calling to get the information or perform the
 ↳ actions if needed. The reasoning process and function calling are enclosed
 ↳ within <think> </think> and <tool_call> </tool_call> tags. The results of
 ↳ the function calls will be given back to you after execution, and you can
 ↳ continue to call functions until you can provide the final answer
 ↳ enclosed within <answer> </answer> tags for the user’s question.

Tool call example:
 <tool_call>
{{"name": <function-name>, "arguments": <args-json-object>}}
...
</tool_call>

Final answer example:
<answer> ... </answer>

1296 **B TRAINING DETAILS**

1297 **B.1 TRM TRAINING DETAILS**

1300 For the search scenario, we use Qwen2.5-3B-Instruct as the backbone model and train with 10K
 1301 examples. For the code scenario, we adopt Qwen2.5-Math-1.5B as the backbone and utilize 20K
 1302 training samples. The training process follows standard supervised fine-tuning procedures, and the
 1303 key hyperparameters are summarized as follows. We set the number of training epochs to 10, with
 1304 a learning rate of 1e-6. The global batch size 128. The maximum sequence length is 8192, and the
 1305 maximum prompt length is 1024. All experiments are conducted using Huggingface implementa-
 1306 tion⁷.

1307 **B.2 RL DETAILS**

1309 **Tool Execution Environment** For the search tool, we follow the setup of Search-R1 (Jin et al.,
 1310 2025) and use the 2018 Wikipedia dump (Karpukhin et al., 2020) as the knowledge source. The
 1311 E5 (Wang et al., 2022) retriever is employed to retrieve relevant passages for each query. For the
 1312 Python code execution environment, we follow the approach in ToRL (Li et al., 2025b) and utilize
 1313 the SandboxFusion environment (Bytedance-Seed-Foundation-Code-Team et al., 2025) to safely
 1314 execute code snippets. This setup ensures both the reliability and security of tool interactions during
 1315 reinforcement learning experiments.

1316 Table 3: Hyperparameters in RL. The notation $3B / 1.5B$ and $7B / 7B$ denote the backbone model
 1317 sizes used for different tasks: the first value corresponds to the search tool for QA, and the second
 1318 value corresponds to the Python code tool for mathematical problem solving.

| Hyperparameter | PPO | | GRPO | |
|---|---------------|---------------|---------------|---------------|
| | $3B / 1.5B$ | $7B / 7B$ | $3B / 1.5B$ | $7B / 7B$ |
| trainer.total_training_steps | 500 | 300 | 500 | 300 |
| algorithm.adv_estimator | gae | gae | grpo | grpo |
| data.train_batch_size | 512 | 512 | $512 / 128$ | $512 / 128$ |
| actor.rollout.ref.actor.ppo_mini_batch_size | 256 | 256 | $256 / 64$ | $256 / 64$ |
| data.max_prompt_length | $8192 / 3072$ | $8192 / 3072$ | $8192 / 3072$ | $8192 / 3072$ |
| data.max_response_length | $512 / 1024$ | $512 / 1024$ | $512 / 1024$ | $512 / 1024$ |
| tools.max_tool_resp_len | 512 | 512 | 512 | 512 |
| actor.rollout.ref.actor.optim.lr | $2e-7 / 1e-6$ | $2e-7 / 1e-6$ | $2e-6$ | $2e-6$ |
| critic.optim.lr | $5e-7 / 5e-6$ | $5e-7 / 5e-6$ | - | - |
| actor.rollout.ref.actor.entropy_coeff | 0.001 | 0.001 | 0 | 0 |
| actor.rollout.ref.rollout.temperature | 1.0 | 1.0 | 1.0 | 1.0 |
| actor.rollout.ref.rollout.n | 1 | 1 | $5 / 8$ | $5 / 8$ |
| tools.max_turns | $5 / 3$ | $5 / 3$ | $5 / 3$ | $5 / 3$ |
| algorithm.kl_ctrl.kl_coef | 0.001 | 0.001 | - | - |
| actor.rollout.ref.actor.kl_loss_coef | - | - | 0.001 | 0.001 |

1336 **Training Hyperparameters** Table 3 presents the key hyperparameters used in our RL experiments.
 1337 All other training configurations follow standard practices as described in the main text.

1340 **C EVALUATION DETAILS**

1341 **C.1 TRM EVALUATION**

1344 For evaluation, we use the checkpoint⁸ from Search-R1 (Jin et al., 2025) to collect rollout candidates
 1345 from the prompts in validation sets of HotpotQA and 2wikimultihopQA. During rollout generation,
 1346 we set the sampling temperature to 0. Additionally, the agent is allowed to perform up to 3 search
 1347 steps per query.

1348 ⁷https://huggingface.co/docs/trl/prm_trainer

1349 ⁸https://huggingface.co/PeterJinGo/SearchR1-nq_hotpotqa_train-qwen2.5-7b-em-ppo-v0.2

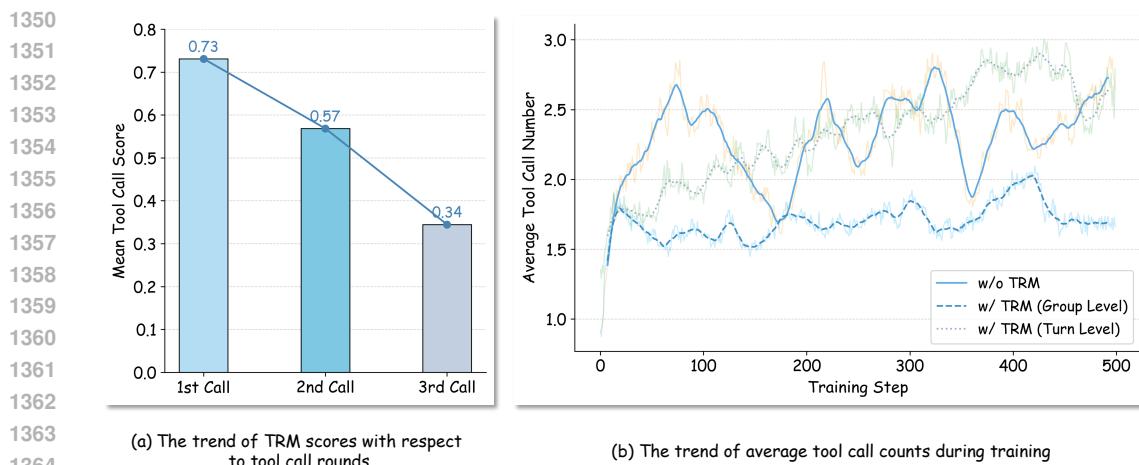


Figure 7: Reward hacking in group-level advantage estimation for GRPO: (a) TRM scores decrease with more tool calls, and (b) turn-level estimation mitigates the penalization of longer tool call sequences.

C.2 LLM EVALUATION

For LLM evaluation, we set the sampling temperature to 0 to encourage deterministic generation.⁹ Other parameters, such as the maximum number of tool calls, are kept consistent with those used during training. Notably, since AIME24 and AIME25 contain very few problems, we report the average results over 30 repeated evaluations for these two datasets to ensure statistical reliability.

D BASELINE DETAILS

Training-free Methods For IRCoT and RAG, we mainly use the implementation¹⁰ of Research (Chen et al., 2025a).

Training Methods For SFT, we adopt LLaMA-Factory (Zheng et al., 2024). For R1, we simply disable tool invocation in our framework.

E ADDITIONAL EXPERIMENTAL RESULTS

E.1 REWARD HACKING CAUSED BY GROUP-LEVEL ADVANTAGE ESTIMATION

Group-level advantage estimation in GRPO can lead to reward hacking, where the model prefers shorter tool call sequences. This is because cascading errors make later tool calls less reliable, resulting in lower scores and penalization for longer sequences (Figure 7-a). In contrast, turn-level advantage estimation alleviates this issue by treating each tool call independently, encouraging more stable tool usage (Figure 7-b). Tool-call numbers on evaluation benchmarks in Table 4 are consistent. Notably, introducing TRM does not significantly increase the number of tool calls compared to outcome-only training methods.

E.2 RESOURCE OVERHEAD INTRODUCED BY TRM

As shown in Table 5, incorporating TRM introduces only an 8.8% overhead per training step, indicating minimal additional compute cost. BoN inference experiences a 50% increase in per-sample time with TRM; however, the absolute time remains small (with the full BoN evaluation taking ~ 20 minutes), which is practically negligible.

⁹Due to the use of the vLLM server, some randomness may still be present during evaluation.

¹⁰<https://github.com/bytedance/SandboxFusion/tree/main>

1404
1405 Table 4: Average tol-call numbers on various QA tasks over Qwen2.5-3B-Instruct with PPO: turn-
1406 level vs. group-level

| Method | <i>General QA</i> | | | <i>Multi-Hop QA</i> | | | Avg. |
|--------------------|-------------------|-----------------|--------------|---------------------|--------------|----------------|-------------|
| | <i>NQ</i> | <i>TriviaQA</i> | <i>PopQA</i> | <i>HotpotQA</i> | <i>2wiki</i> | <i>Musique</i> | |
| Group-level | 1.99 | 1.93 | 2.00 | 2.42 | 2.81 | 3.01 | 2.37 |
| Turn-level | 2.59 | 2.44 | 2.48 | 2.68 | 3.01 | 3.19 | 2.68 |

1411
1412 Table 5: Training and BoN inference speed with vs. without TRM on Qwen2.5-3B-Instruct with
1413 8xA800 GPUs under PPO

| Method | Training (s/step) | BoN Inference (s/sample) |
|---------------|--------------------------|---------------------------------|
| w/o TRM | 56.9 | 0.14 |
| w/ TRM | 61.9 (+8.8%) | 0.21 (+50%) |

E.3 TRM TRAINING DATA QUALITY VERIFICATION

Regarding TRM training data quality, we randomly sampled 100 trajectories and evaluated them using both human annotators and a multi-model judge (DeepSeek-R1, Qwen2.5-72B-Instruct, and GPT-4o-mini). As shown in Table 6, the resulting dataset demonstrates relatively high quality.

Table 6: TRM Training Data Quality Evaluation (Accuracy)

| Evaluation Method | Necessity (%) | Quality (%) |
|--------------------------|----------------------|--------------------|
| Human | 89 | 89 |
| Multi-model | 86 | 75 |

E.4 GENERALITY BEYOND SEARCH AND CODE TOOLS

We follow the ReCall (Chen et al., 2025a) work to evaluate our method in more diverse multi-tool scenarios. These scenarios include tasks such as library management, employee management, and travel planning, which require the use of multiple APIs (e.g., searching and reserving books, calculating salaries, updating flight bookings, etc.). These additional experiments (see Table 7) show that introducing TRM continues to provide consistent improvements in broader multi-tool settings.

Table 7: Performance in more diverse multi-tool scenarios on Qwen2.5-7B-Instruct with GRPO

| Method | Qwen2.5-7B-Instruct | ReCall | ReCall-TRM |
|---------------|----------------------------|---------------|-------------------|
| F1 | 7.37 | 39.71 | 43.28 |

E.5 CASE STUDY

1458
 1459
 1460
 1461
 1462
 1463
 1464
 1465
 1466
 1467
 1468
 1469
 1470
 1471
 1472
 1473
 1474
 1475
 1476
 1477
 1478
 1479
 1480
 1481
 1482
 1483
 1484
 1485
 1486
 1487
 1488
 1489
 1490
 1491
 1492
 1493
 1494
 1495
 1496
 1497
 1498
 1499
 1500
 1501
 1502
 1503
 1504
 1505
 1506
 1507
 1508
 1509
 1510
 1511

? Question: who wrote lord have mercy on the working man?

Model w/o TRM:

Turn 1:
Thought: To determine who wrote "Lord Have Mercy on the Working Man", I'll start with a general search to identify the song.
 I need to find out which song this is referring to. It's possible that the title might be a specific song or a phrase from a song.
Tool-call: who wrote lord have mercy on the working man
Observation: ... Lord Have Mercy on the Working Man "Lord Have Mercy on the Working Man" is a song written by Kostas and recorded by American country music singer [Travis Tritt](#) ...

Turn 2:
Thought: From the search results, I found that "Lord Have Mercy on the Working Man" is a song written by Kostas and recorded by American country music singer Travis Tritt. I need to confirm if this is the correct information.
Tool-call: who is kostas in the context of the song Lord Have Mercy on the Working Man
Observation: ... Kostas Hatzis Kostas Hatzis () (born August 13, 1936) is a Greek singer-songwriter and musician of Romani origin...

Turn 3:
Thought: <None>
Answer: Travis Tritt

Comment:
 Both models (with and without TRM) adopt reasonable tool call strategies, but only the TRM-integrated model produces the correct answer. The error made by the model without TRM highlights the motivation for our approach and shows that TRM helps address cascading errors in tool usage.

Model w/ TRM:

Turn 1:
Thought: To determine who wrote "Lord Have Mercy on the Working Man", I will start with a general search to gather this information.
 If I don't find direct information online, I might need to look into historical records or music archives.
Tool-call: who wrote lord have mercy on the working man
Observation:... Lord Have Mercy on the Working Man "Lord Have Mercy on the Working Man" is a song written by Kostas and recorded by American country music singer [Travis Tritt](#) ...

Turn 2:
Thought: From the search results, I found that "Lord Have Mercy on the Working Man" is a song written by Kostas and recorded by American country music singer [Travis Tritt](#).
Answer: Kostas

Figure 8: Case 1

1480
 1481
 1482
 1483
 1484
 1485
 1486
 1487
 1488
 1489
 1490
 1491
 1492
 1493
 1494
 1495
 1496
 1497
 1498
 1499
 1500
 1501
 1502
 1503
 1504
 1505
 1506
 1507
 1508
 1509
 1510
 1511

? Question: Let $\omega \neq 1$ be a 13th root of unity. Find the remainder when $\prod_{k=0}^{12} (2 - 2\omega^k + \omega^{2k})$ is divided by 1000.

Model w/o TRM:

Turn 1:
Thought: To solve the problem, we need to evaluate the product $\prod_{k=0}^{12} (2 - 2\omega^k + \omega^{2k})$ where ω is a 13th root of unity, and then find the remainder when this product is divided by 1000.
 First, let's understand the properties of the 13th roots of unity. The 13th roots of unity are the solutions to the equation $x^{13} = 1$. These roots can be written as ω^k for $k = 0, 1, 2, \dots, 12$, where $\omega = e^{2\pi i / 13}$.
 The expression we need to evaluate is $\prod_{k=0}^{12} (2 - 2\omega^k + \omega^{2k})$. We can use the fact that the sum of all 13th roots of unity is zero, i.e., $\sum_{k=0}^{12} \omega^k = 0$.
 Let's break down the problem into manageable steps:
 1. Define the 13th roots of unity.
 2. Compute the product $\prod_{k=0}^{12} (2 - 2\omega^k + \omega^{2k})$.
 3. Find the remainder when the product is divided by 1000.
 We will use Python and the sympy library to perform these calculations.

Tool-Call:
 ````python  
 import sympy as sp  
 # Define the 13th root of unity  
 omega = sp.exp(2 \* sp.pi \* sp.I / 13)  
 # Compute the product  
 product = 1  
 for k in range(13):  
 for k in range(13):  
 product \*= (2 - 2 \* omega\*\*k + omega\*\*(2 \* k))  
 # Simplify the product  
 product = sp.simplify(product)  
 # Find the remainder when the product is divided by 1000  
 remainder = sp.Mod(product, 1000)  
 print(remainder)  
 ````

Observation: Run result:<None>

...
 (the Turn 2 generate similar analysis and the same code, skip)

Turn 3:
Thought: The output of the code indicates that the remainder when the product $\prod_{k=0}^{12} (2 - 2\omega^k + \omega^{2k})$ is divided by 1000 is [336](#).
Answer: 336

Comment:
 Without TRM, the model generates overly complex code that fails during execution, resulting in a wrong answer. With TRM, the model produces higher-quality code and answers correctly. This case highlights how TRM improves code generation quality and helps mitigate errors, validating our motivation for its integration.

Model w/ TRM:

Turn 1:
Thought: To solve the given problem, we need to evaluate the product $\prod_{k=0}^{12} (2 - 2\omega^k + \omega^{2k})$ where ω is a 13th root of unity, and then find the remainder when this product is divided by 1000.
 First, let's understand the properties of the 13th roots of unity. The 13th roots of unity are the solutions to the equation $x^{13} = 1$. These roots can be written as $\omega^k = e^{2\pi i / 13}$ for $(k = 0, 1, 2, \dots, 12)$.
 The expression $(2 - 2\omega^k + \omega^{2k})$ can be simplified using the properties of roots of unity. We need to evaluate this product for all k from 0 to 12.
 Let's write a Python script to compute this product and then find the remainder when the product is divided by 1000

Tool-Call:
 ````python  
 import cmath  
 # Define the 13th root of unity  
 n = 13  
 omega = cmath.exp(2j \* cmath.pi / n)  
 # Compute the product  
 product = 1  
 for k in range(n):  
 term = 2 - 2 \* omega\*\*k + omega\*\*(2 \* k)  
 product \*= term  
 # Get the real part of the product (since the imaginary part should be negligible)  
 product\_real = round(product.real)  
 # Find the remainder when the product is divided by 1000  
 remainder = product\_real % 1000  
 print(remainder)  
 ````

Observation: 321

Turn 2:
Thought: The remainder when the product $\prod_{k=0}^{12} (2 - 2\omega^k + \omega^{2k})$ is divided by 1000 is [321](#).
Answer: 321

Figure 9: Case 2

1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565

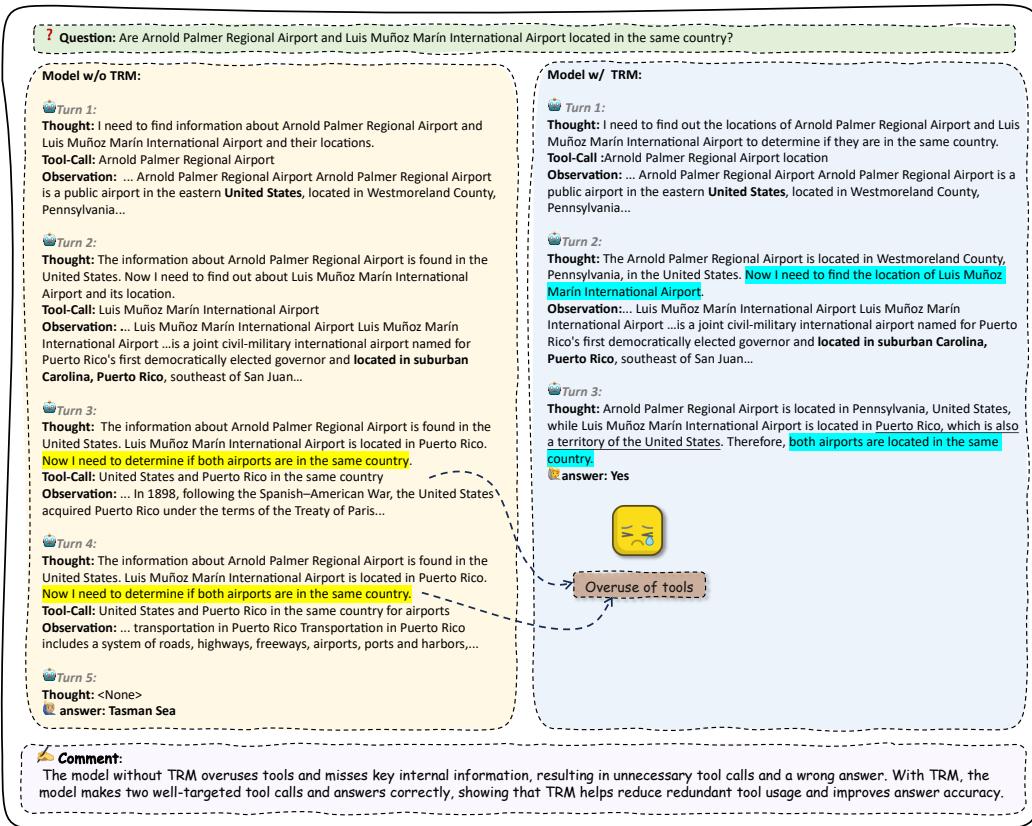


Figure 10: Case 3

1566
1567

E.6 PROCESS-LEVEL TRANSPARENCY OF TRM

1568
1569

Table 8: Examples of failed tool calls

1570
1571
1572
1573
1574
1575

Question	Tool Calls	Comment	Explanation
What was Bill Dvorak's fifth cousin's nationality?	<code>search(Bill Dvorak nationality)</code>	low-quality	The query asks about Bill Dvorak's nationality which is unrelated to the user's need (fifth cousin's nationality). It reflects a major misinterpretation of the question.
Are Arnold Palmer Regional Airport and Luis Muñoz Marín International Airport located in the same country?	<code>search(Arnold Palmer Regional Airport)</code> <code>search(United States and Puerto Rico in the same country)</code> <code>search(United States and Puerto Rico in the same country for airports)</code>	unnecessary	Invoke duplicate tools
The product $\$8 * .25 * 2 * .125 = \$$	# Calculate the product step by step result = 8 * 0.25 * 2 * 0.125 <code>print(result)</code>	unnecessary	The multiplication of four numbers (8, 0.25, 2, 0.125) is simple and can be easily computed by hand; writing a program is unnecessary for this straightforward arithmetic.
Medians \$BD\$ and \$CE\$ of triangle \$ABC\$ are perpendicular, \$BD=8\$, and \$CE=12\$. The area of triangle \$ABC\$ is	Very long code ...	low-quality	The code contains significant unused symbolic logic (Sympy setup/solving) that's irrelevant to the final arithmetic operation, which is inefficient and shows flawed implementation. Though the output is correct, the dead code constitutes a major structural issue.

1607

1608
1609

E.7 DETAILED EXPERIMENTAL RESULTS

1610

1611

1612

1613

1614

1615

1616

1617

1618

1619

Table 9: Detailed results of Figure 3

Method	Model Size	Training Data Size	1	2	3	4	5	Avg
<i>2wikimultihopqa [out-of-domain] (dev split)</i>								
Majority Vote	-	-	11.73	11.73	10.82	10.76	10.56	11.12
Pass@n			11.73	15.16	17.40	19.02	20.26	16.71
		1K	11.73	13.53	14.58	15.33	15.76	14.19
TRM	3B	2K	11.73	13.79	15.02	15.80	16.39	14.55
		5K	11.73	14.33	15.94	16.94	17.80	15.35
		10K	11.73	14.29	15.92	16.94	17.66	15.31
		0.5B	11.73	12.31	12.81	13.10	13.20	12.63
TRM	1.5B	10K	11.73	13.88	15.17	15.97	16.68	14.69
	7B		11.73	13.36	14.54	15.20	15.66	14.10
<i>hotpotqa [in-domain] (dev split)</i>								
Majority Vote	-	-	24.52	24.52	25.05	25.08	25.29	24.89
Pass@n			24.52	30.97	33.96	35.83	37.54	32.56
		1K	24.52	28.16	29.74	30.40	31.02	28.77
TRM	3B	2K	24.52	26.85	27.93	28.66	28.82	27.36
		5K	24.52	28.82	30.70	31.61	32.40	29.61
		10K	24.52	29.03	30.90	32.14	32.88	29.89
		0.5B	24.52	25.23	25.28	25.21	24.85	25.02
TRM	1.5B	10K	24.52	27.86	29.40	30.26	30.71	28.55
	7B		24.52	26.91	27.75	28.53	28.94	27.33

Table 10: Detailed results of Figure 5-a and Figure 5-b

Method	α	NQ	TriviaQA	PopQA	HotpotQA	2wiki	Musique	Bamboogle	Avg.
PPO	0.01	38.14	55.20	36.17	32.64	31.00	11.21	20.80	32.17
	0.05	39.58	57.78	40.61	34.80	33.22	12.91	25.60	34.93
	0.1	40.08	55.82	39.11	32.91	32.73	11.12	27.20	34.14
	0.3	34.52	50.08	35.98	29.41	26.81	9.10	28.00	30.56
GRPO	0.01	47.89	62.57	47.20	44.47	43.48	19.65	39.20	43.49
	0.05	48.09	63.04	46.93	44.66	43.45	19.20	37.60	43.28
	0.1	46.68	62.58	45.93	43.47	42.89	16.88	38.40	42.40

Table 11: Detailed results of Figure 5-c

Method	Search-2 Code			Avg.
	MATH500	Olympiad	AMC23	
ToRL-PPO	50.40	24.00	25.00	33.13
ToRL-PPO-TRM (ours)	54.20	26.22	27.50	35.97
ToRL-GRPO	52.80	22.81	30.00	35.20
ToRL-GRPO-TRM (ours)	56.60	27.70	35.00	39.77

Table 12: Detailed results of Figure 6

Method	NQ	TriviaQA	PopQA	HotpotQA	2wiki	Musique	Bamboogle	Avg.
<i>Performance</i>								
Search-R1 + StepSearch	37.53	55.17	39.20	29.75	27.65	7.74	17.60	30.66
Search-R1 + AgentPRM	38.01	54.62	37.07	32.78	31.15	10.22	19.20	31.86
Search-R1 + ORM	39.47	56.17	40.93	29.63	26.65	6.83	9.60	29.90
Search-R1 + TRM-verifier	35.65	54.10	36.46	33.91	33.90	13.86	27.20	33.58
quality-only	38.95	56.17	38.94	31.06	27.93	8.44	16.00	31.07
necessity-only	37.42	54.06	37.21	32.46	31.80	11.46	21.60	32.29
<i>Tool-call Number</i>								
quality-only	3.96	3.94	3.93	3.96	3.98	3.99	3.93	3.96
necessity-only	2.58	2.68	2.58	2.83	2.95	3.31	2.84	2.82
both	2.37	2.39	2.37	2.85	3.35	3.23	2.71	2.75