ToolPRMBench: Evaluating and Advancing Process Reward Models for Tool-using Agents

Dawei Li[♠], **Yuguang Yao**[♦], **Zhen Tan**[♠], **Huan Liu**[♠], **Ruocheng Guo**[♦*]
[♠]Arizona State University, [♦]Intuit AI Research
{daweili5, ztan36, huanliu}@asu.edu
{yuguang_yao,ruocheng_guo}@intuit.com

Abstract

Reward-guided search methods have demonstrated strong potential in enhancing tool-using agents by effectively guiding sampling and exploration over complex action spaces. As a core design, those search methods utilize process reward models (PRMs) to provide step-level rewards, enabling more fine-grained monitoring. However, there is a lack of systematic and reliable evaluation benchmarks for PRMs in tool-use settings. In this paper, we introduce ToolPRMBench, a large-scale benchmark specifically designed to evaluate PRMs for tool-using agents. ToolPRMBench is built on top of several representative tool-use benchmarks and converts agent trajectories into step-level test cases. Each case contains the interaction history, a correct action, a plausible but incorrect alternative, and relevant tool metadata. We respectively utilize offline sampling to isolate local single-step errors and online sampling to capture realistic multi-step failures from full agent rollouts. A multi-LLM verification pipeline is proposed to reduce label noise and ensure data quality. We conduct extensive experiments across large language models, general PRMs, and tool-specialized PRMs on ToolPRMBench. The results reveal clear differences in PRM effectiveness and highlight the potential of specialized PRMs for tool-using. Our code and dataset are available at: https://github.com/David-Li0406/ToolPRMBench[1].

1 Introduction

Large Language Models (LLMs) are increasingly deployed as tool-using agents, where they interact with external tools such as APIs, databases, and execution environments to solve complex, multi-step tasks (Shen, 2024; Huang et al., 2024). This paradigm significantly extends the capability of LLMs beyond pure text generation. However, effective tool-using remains challenging. As highlighted in recent work, even strong models frequently fail due to early mistakes that propagate through long sequences, while final outcome-based evaluation provides limited insight into where the reasoning process goes wrong (Yao et al., 2022; Chae et al., 2025).

Benchmark	Step-level	Agent	Tool Use
Agent-RewardBench (Men et al., 2025)	✗	✓	Web-only
AgentRewardBench (Lù et al., 2025)	✗	✓	Web-only
WebRewardBench (Chae et al., 2025)	✓	✓	Web-only
PRMBench (Song et al., 2025)	✓	✗	✗
ToolPRMBench	✓	✓	**Diverse APIs**

Table 1: ToolPRMBench is the first benchmark that supports step-level evaluation for interactive agents with diverse tool APIs.

Motivated by these challenges, recent research has increasingly explored reward-guided search as a way to improve tool-using agents. Instead of committing to a single trajectory, reward-guided methods perform searching or sampling over multiple candidate actions or plans. In tool-use and web agent settings, methods such as best-of-n or Monte Carlo tree search have shown promising results (Chae et al., 2025; Agarwal et al., 2025). A central component of these approaches is a Process Reward Model (PRM) (Snell et al., 2024; Zhao et al., 2025), which provides step-level feedback to guide exploration and prune incorrect trajectories early. Compared to outcome-only rewards, PRMs offer finer-grained signals that are better aligned with the long-horizon tool-using tasks. However, despite their growing importance, evaluating PRMs for tool use remains a challenging task. Tool-using, as an agent task, is characterized by long interactions and a large, structured action space, where

*Corresponding author.
[1]More resources on **LLM-as-a-judge** are on the website: https://llm-as-a-judge.github.io

Findings of the Association for Computational Linguistics: ACL 2026, pages 12378–12391

errors can emerge at many intermediate steps and propagate over time; as a result, existing PRM benchmarks designed for general reasoning (Song et al., 2025) or web agents (Men et al., 2025; Lù et al., 2025) may not be directly applicable or sufficiently effective in tool-use scenarios. Moreover, existing PRM designs vary widely, ranging from LLM-as-a-judge methods (Li et al., 2025) to general PRMs (Yang et al., 2024) or agent- (Chae et al., 2025) and tool-specialized PRMs. However, there is no unified benchmark to systematically evaluate their effectiveness in tool-using settings.

To address this gap, we introduce **ToolPRMBench**, a large-scale benchmark designed specifically for evaluating process reward models for tool-using agents (Figure 1). ToolPRMBench is constructed on top of several representative tool-using benchmarks, covering diverse environments such as information-seeking, multi-step reasoning, and interactive tool execution. Each sample in ToolPRMBench consists of the interaction history, a correct action, an incorrect but plausible alternative, and associated tool metadata. To build the dataset, we combine offline sampling, which isolates local single-step errors around golden trajectories, and online sampling, which captures realistic multi-step failures from full agent rollouts. These candidate samples are further verified through a multi-LLM filtering pipeline to ensure label reliability. This construction process results in a diverse and challenging benchmark that enables fine-grained evaluation of PRMs at the decision-step level.

Through extensive experiments on ToolPRMBench across a total of 17 large language models, general PRMs, and tool-specialized PRMs, we observe clear and consistent performance differences across model types. Our results further show that scaling model size and general capabilities benefit tool process reward modeling, while reinforcement learning demonstrates strong potential for improving robustness and generalization in tool-use PRMs. In addition, we conduct a series of analyses on ToolPRMBench, including meta-evaluation, data synthesis, cost analysis, and case studies, providing further insights to guide future research on reward modeling for tool-using agents.

In summary, the contribution in this work is threefold:

- First, we propose ToolPRMBench, a large-scale benchmark carefully constructed for systematic evaluation of PRMs in tool-use settings.
- Second, we benchmark a series of LLMs, general PRMs, and tool-using specialized PRMs in ToolPRMBench, building a comprehensive PRMs leaderboard in the tool-using scenario.
- Finally, we conduct further analysis, including meta-evaluation and cost analysis, providing findings and insights for future reward-guided trajectory searching in tool use.

2 Related Work

2.1 Tool Use Benchmarks

Tool use agents extend LLMs by enabling them to interact with external tools or APIs rather than only producing text. This capability is necessary because many real-world tasks cannot be solved by language generation alone and require actual execution of external actions to achieve correct outcomes. Several benchmarks have been proposed to evaluate tool use capabilities (Farn and Shin, 2023; Grattafiori et al., 2024; Wang et al., 2024a; Lu et al., 2025). Some focus on whether a model can utilize tools where appropriate (Huang et al., 2024; Ning et al., 2024), while others measure multi-hop or structured tool invocation, such as ToolHop (Ye et al., 2025) and MCP-RADAR (Gao et al., 2025). Additionally, evaluation frameworks like T-Eval and Trajectory-Bench decompose tool utilization into sub-processes (instruction following, planning, reasoning, retrieval, etc.) to provide fine-grained analysis beyond end-to-end task success (Chen et al., 2024; He et al., 2025). Despite progress, many existing benchmarks focus solely on final task success, rather than fine-grained and step-level accuracy. Therefore, we propose ToolPRMBench, a benchmark designed to evaluate PRM across diverse tool use trajectories and scenarios.

2.2 Reward-guided Search and Reward Modeling

Reward-guided search refers to strategies that improve model performance at test time by sampling and searching over multiple candidate actions. In general tasks, reward-guided search relies on learned or heuristic reward models to evaluate candidate outputs, including outcome-based reward models trained from human or AI preferences (Stiennon et al., 2020), classifier-style evaluators or value functions for reranking (Cobbe et al., 2021),

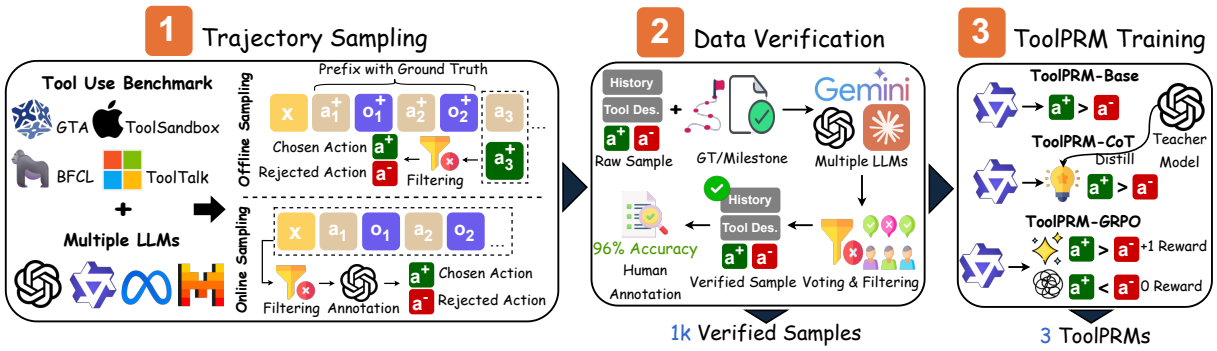

Figure 1: The overview pipeline of ToolPRMBench, including trajectory sampling, data verification & filtering, and ToolPRM training.

and self-consistency or majority-vote signals derived from multiple sampled trajectories (Wang et al.). For long-horizon settings such as agents and tool-using systems, however, sparse outcome rewards are often insufficient, and successful reward-guided search critically depends on process reward modeling and accurate step-level rewards that can guide intermediate decisions and credit assignment (Uesato et al., 2022; Lightman et al., 2023). Such step-wise guidance is especially important when agents must balance reasoning depth, action selection, and external tool costs over extended trajectories (Zhou et al., 2024; Chae et al., 2025). Motivated by recent progress in this direction, we introduce ToolPRMBench to evaluate the effectiveness of PRMs in tool-using.

3 ToolPRMBench

ToolPRMBench is a benchmark for PRMs in tool-using agents. It aggregates trajectories using multiple existing tool-using benchmarks and converts them into step-wise test samples, enabling systematic evaluation of whether a given PRM can distinguish correct actions from incorrect ones at each decision step. Following the design of the previous PRMBench (Chae et al., 2025) for web agents, ToolPRMBench focuses on process-level correctness, rather than just final task success. ToolPRMBench is constructed on top of four representative tool-use benchmarks, including ToolTalk (Farn and Shin, 2023), GTA (Wang et al., 2024a), BFCL (Patil et al.), and ToolSandbox (Lu et al., 2025). These benchmarks cover diverse environments for tool-using agents such as information-seeking, multi-step reasoning, and interactive tool execution.

Each sample in ToolPRMBench corresponds to a single decision step in a sequence of interactions between the agent and the environment. Let x denote the user instruction, and $\tau = ((a_1, o_1), (a_2, o_2), \ldots, (a_T, o_T))$ represent a trajectory of actions of an agent, where a_t denotes the action at step t (i.e., a structured tool call) and o_t is the corresponding observation returned by the environment. The interaction history at step t is defined as $h_t = (x, a_1, o_1, \ldots, a_{t-1}, o_{t-1})$. Then, a ToolPRMBench sample is a tuple (h_t, a_t^+, a_t^-, m_t), where a_t^+ is the chosen (correct) action, a_t^- is the rejected (incorrect) action, and m_t contains metadata such as the available tool description.

3.1 Trajectory Sampling

To collect rejected actions, we adopt two trajectory sampling strategies that complement each other: offline and online sampling. Both strategies leverage the golden trajectory or milestone, but differ in the level of flexibility the model has when generating actions.

Offline sampling constrains the model to follow the golden trajectory prefix and only samples an alternative action at a specific step. This setting focuses on local errors and isolates single-step mistakes. In contrast, online sampling allows the model to generate an entire trajectory from the beginning freely, which naturally leads to multi-step, correlated errors. Online sampling better reflects realistic agent failures but is harder to analyze. In ToolPRMBench, we choose the sampling strategy based on the characteristics of the original benchmark and its evaluation protocol.

Offline Sampling. Let $\tau^\star = ((a_1^\star, o_1^\star), \ldots, (a_T^\star, o_T^\star))$ be the golden trajectory for instruction x. At step t, the golden history is $h_t^\star = (x, a_1^\star, o_1^\star, \ldots, a_{t-1}^\star, o_{t-1}^\star)$. We query a tool-using policy π to sample an action $\tilde{a}_t \sim \pi(\cdot \mid h_t^\star)$. Importantly, the environment is

not updated with $\tilde{a}_t$, and subsequent steps always follow the golden trajectory.

To construct samples in offline sampling, we compare $\tilde{a}_t$ with the golden action $a_t^\star$. If the two actions are semantically equivalent, the step is discarded. Otherwise, we create a candidate sample $(h_t^\star, a_t^+ = a_t^\star, a_t^- = \tilde{a}_t)$. The comparison is performed using task-specific rules that examine the tool name and key arguments. Offline sampling, therefore, produces localized deviations around a correct history and provides clean supervision for step-level error detection.

Online Sampling. Offline sampling cannot capture error propagation across multiple steps. To address this limitation, we additionally collect data using online sampling in interactive benchmarks such as BFCL and ToolSandbox. Given instruction x, the policy π generates a full trajectory $\hat{\tau} = ((\hat{a}_1, \hat{o}_1), \ldots, (\hat{a}_{\hat{T}}, \hat{o}_{\hat{T}}))$ by interacting with the environment. Each generated trajectory is evaluated using the benchmark's outcome-based metric, resulting in a binary success signal $s(\hat{\tau}) \in \{0, 1\}$. We retain only failed trajectories with $s(\hat{\tau}) = 0$.

To identify the erroneous step in a failed trajectory, we employ an LLM-based annotation process. The annotator LLM is given the user instruction, the generated trajectory, the metadata, and a golden reference (either the full golden trajectory or task milestones). It is asked to identify the first incorrect step index t_{err} and to propose a corrected action $\bar{a}_{t_{\text{err}}}$. The history is then defined as $\hat{h}_{t_{\text{err}}} = (x, \hat{a}_1, \hat{o}_1, \ldots, \hat{a}_{t_{\text{err}}-1}, \hat{o}_{t_{\text{err}}-1})$, and the resulting preference pair is $(\hat{h}_{t_{\text{err}}}, a_t^+ = \bar{a}_{t_{\text{err}}}, a_t^- = \hat{a}_{t_{\text{err}}})$. This procedure converts trajectory-level failures into step-level supervision, making it suitable for PRM evaluation.

3.2 Data Verification

Both offline and online sampling can introduce noise. In offline sampling, the golden trajectory is not necessarily the only valid solution; therefore, some sampled actions may be incorrectly labeled as rejected. In online sampling, LLM-based annotation may misidentify the error step or propose an incorrect correction. To mitigate these issues, we apply a multi-LLM verification and filtering pipeline.

For each candidate sample (h_t, a_t^+, a_t^-, m_t), we query three powerful LLMs (GPT-5, Gemini-3-flash and Claude-4.5-haiku) to independently judge whether a_t^+ is strictly better than a_t^- given the history. Each model provides a binary judgment, and we aggregate the results using majority voting. Samples that receive unanimous positive votes are retained, while samples unanimously rejected are discarded. For borderline cases with mixed votes, we perform additional human verification on a subset of samples. This multi-judge strategy significantly reduces label noise and improves the reliability of the test set in ToolPRMBench. To further validate the effectiveness of multi-LLM verification, we randomly sample 100 samples from all LLM-verified results and observe a 96% agreement with human judgments.

3.3 ToolPRM Training

We describe the training objectives of different ToolPRM variants. Each training instance is represented as

$$(h_t, \tilde{a}_t^{(1)}, \tilde{a}_t^{(2)}, m_t), \tag{1}$$

where $\{\tilde{a}_t^{(1)}, \tilde{a}_t^{(2)}\}$ is a random permutation of the chosen action a_t^+ and the rejected action a_t^- to avoid position bias (Li et al., 2025). We consider the following three training methods.

ToolPRM-Base. ToolPRM-Base is trained to directly predict which candidate action should be selected. Given the input $(h_t, \tilde{a}_t^{(1)}, \tilde{a}_t^{(2)}, m_t)$, the model outputs an action label $y_t \in \{1, 2\}$, where y_t indicates the position of the chosen action in the permuted candidate list. Specifically, $y_t = 1$ if the chosen action a_t^+ appears at position $\tilde{a}_t^{(1)}$, and $y_t = 2$ otherwise. The model is trained using supervised fine-tuning (SFT) with a cross-entropy loss:

$$\mathcal{L}_{\text{SFT}} = -\log p(y_t \mid h_t, \tilde{a}_t^{(1)}, \tilde{a}_t^{(2)}, m_t). \tag{2}$$

ToolPRM-CoT. ToolPRM-CoT extends the base model by explicitly modeling the reasoning process. For each input $(h_t, \tilde{a}_t^{(1)}, \tilde{a}_t^{(2)}, m_t)$, the model is trained to generate a reasoning sequence r_t followed by the action label y_t. The reasoning sequence r_t used during training is from a larger teacher model. Both r_t and the final action label y_t are optimized jointly using supervised fine-tuning:

$$\mathcal{L}_{\text{SFT}} = -\log p(r_t, y_t \mid h_t, \tilde{a}_t^{(1)}, \tilde{a}_t^{(2)}, m_t). \tag{3}$$

ToolPRM-GRPO. ToolPRM-GRPO further improves the model using reinforcement learning with Group Relative Policy Optimization (GRPO) (Shao et al., 2024). For each case, the policy samples multiple reasoning–action pairs (r_t, y_t) conditioned on

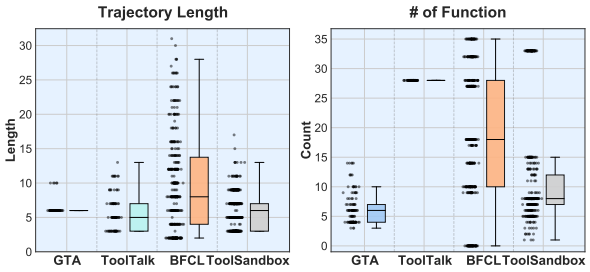

Figure 2: Statistics of trajectory length and number of functions in ToolPRMBench.

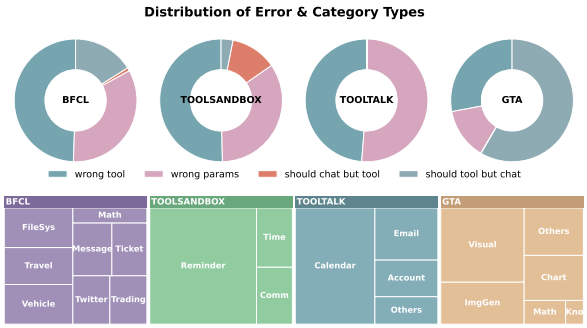

Figure 3: Statistics of error and category distribution in ToolPRMBench.

$(h_t, \tilde{a}_t^{(1)}, \tilde{a}_t^{(2)}, m_t)$. We define a binary reward function:

$$R(y_t) = \begin{cases} 1, & \text{if } y_t \text{ corresponds to the position of } a_t^+ \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The training objective is to maximize the expected reward:

$$\mathbb{E}_{(r_t, y_t) \sim p_\theta(\cdot | h_t, \tilde{a}_t^{(1)}, \tilde{a}_t^{(2)}, m_t)} [R(y_t)]. \tag{5}$$

This reinforcement learning stage encourages the model to refine both the reasoning and action selection behavior beyond supervised fine-tuning.

3.4 Dataset Statistics

There are 984 samples in the ToolPRMBench in total. Figure 2 and 3 present the distribution of error, category, trajectory length, and function number in four subsets of ToolPRMBench. We found that most trajectories have moderate lengths, covering both short interactions and more complex multi-step processes. This distribution allows the test set to probe model performance across varying levels of task difficulty without being dominated by trivial cases. Additionally, the data includes common failure modes, such as incorrect action selection, incorrect arguments, and improper use of tools, as well as natural language responses. These errors appear at different positions along trajectories, supporting a comprehensive assessment of a model's ability to detect and rank incorrect decisions throughout the interaction. Overall, the statistics indicate that ToolPRMBench is diverse and challenging, making it suitable for fine-grained evaluation of step-level decision quality. More statistics and details of the ToolPRMBench collection can be found in Appendix A.

4 Experiment

4.1 Experiment Setting

Models. We benchmark a series of LLMs in our ToolPRMBench, including (1). API-Based LLMs: GPT-5 (OpenAI, 2025), Claude-4.5-haiku (Anthropic, 2025) and Gemini-2.5-flash (Comanici et al., 2025). (2). Open-source LLMs, including models in Qwen3 (Yang et al., 2025) and LLaMA-3 (Grattafiori et al., 2024) families. (3). General PRMs for math and web navigation, including WebShepHerd-8B (Chae et al., 2025), Qwen2.5-Math-7B (Yang et al., 2024), Llemma-7b-prm (Sun et al., 2024) and Math-shepherd (Wang et al., 2024b). (4). Tool use specialized PRMs, including ToolPRM-Base, ToolPRM-CoT, and ToolPRM-GRPO.

Implementation Details. ToolPRM is trained on parts of the BFCL and ToolSandbox subset in ToolPRMBench, with the training and testing ratio to be 7:3. To prevent data contamination, we ensure that all ToolPRMBench samples derived from the same instruction are assigned exclusively to either the training set or the testing set. All ToolPRM variants are trained on top of Qwen-3-4B. For ToolPRM-CoT, the reasoning supervision is distilled from GPT-5-mini. Model training is conducted using LLaMA-Factory (Zheng et al., 2024) and TRL (von Werra et al., 2020), and inference is performed with the vLLM backend engine. More details about experiment implementation can be found in Appendix B.

4.2 Main Result

Overall comparison across model categories. Table 2 and the leaderboard in Figure 5 provide a comprehensive comparison of various model types on ToolPRMBench. A clear performance hierarchy emerges across model families. API-based LLMs consistently achieve the strongest overall results, ranking at the top across almost all subsets. This

Model	GTA	ToolTalk	BFCL	ToolSandbox	AVG
API-Based LLMs					
GPT-5 (OpenAI, 2025)	87.3	82.5	44.1	83.7	74.4
Claude-4.5-haiku (Anthropic, 2025)	91.5	**93.0**	45.9	**70.0**	**75.1**
Gemini-2.5-flash (Comanici et al., 2025)	90.1	86.7	40.8	75.3	73.2
Open-Source LLMs					
Qwen3-1.7B (Yang et al., 2025)	50.8	50.0	36.7	38.1	43.9
Qwen3-4B (Yang et al., 2025)	63.5	66.2	30.1	37.8	49.4
Qwen3-8B (Yang et al., 2025)	66.1	68.6	35.2	45.0	53.7
Qwen3-14B (Yang et al., 2025)	74.6	80.1	35.2	62.1	63.0
LLaMA-3-3B-Instruct (Grattafiori et al., 2024)	40.7	41.9	40.5	50.7	43.4
LLaMA-3-8B-Instruct (Grattafiori et al., 2024)	42.4	50.0	47.2	39.7	44.8
LLaMA-3-70B-Instruct (Grattafiori et al., 2024)	65.3	70.1	43.2	36.0	53.6
General PRMs					
WebShepHerd-8B (Chae et al., 2025)	52.0	64.0	37.5	43.4	49.2
Qwen2.5-Math-7B (Yang et al., 2024)	29.7	69.4	36.9	67.0	50.8
Llemma-7b-prm (Sun et al., 2024)	41.5	64.1	45.0	60.8	52.8
Math-shepherd (Wang et al., 2024b)	59.3	33.6	53.0	57.7	50.9
Tool Use PRMs					
ToolPRM-Base	38.1	65.1	47.7	77.7	57.1
ToolPRM-CoT	55.1	56.9	57.7	**83.0**	63.2
ToolPRM-GRPO	**84.7**	**73.3**	**86.4**	70.0	**78.6**

Table 2: Main experiment result in ToolPRMBench. Best result in each subset is **bold**; second best is underlined.

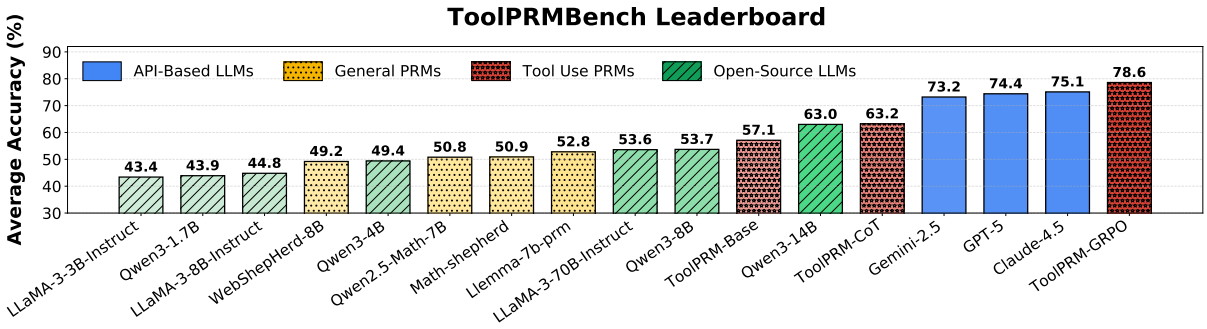

Figure 4: ToolPRM leaderboard on 17 LLMs.

suggests that large-scale training and strong general reasoning abilities continue to be highly effective for process-level evaluation in tool-use scenarios.

Tool-specialized PRMs also demonstrate strong performance. In particular, ToolPRM-GRPO achieves the best average accuracy among all non-API models, outperforming even the API-based LLMs. ToolPRM-CoT and ToolPRM-Base further show that reward models explicitly trained for tool use substantially outperform both open-source LLMs and general-purpose PRMs. In contrast, open-source LLMs and general PRMs exhibit noticeably weaker performance. Many of these models struggle to exceed 55% average accuracy, suggesting that PRMs trained for math reasoning or web navigation do not directly transfer to tool-use process evaluation. Overall, these results highlight the importance of tool-specific supervision when designing effective process reward models.

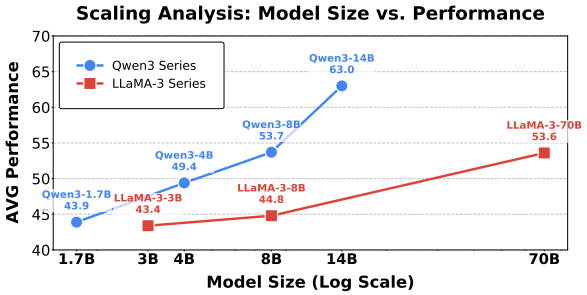

Figure 5: Scaling analysis result on Qwen3 and LLaMA-3.

Scaling behavior of base models. Figure 5 reports the scaling analysis over the Qwen3 and LLaMA-3 model families. A clear positive trend can be observed between model size and performance on ToolPRMBench. This result suggests that improvements in general model capacity, such as reasoning ability and instruction following, are beneficial for tool process reward modeling. How-

ever, scaling alone is not sufficient. Even the largest open-source models still lag behind ToolPRMs by a significant margin. This gap suggests that while model size and general performance are beneficial, specialized training remains crucial for achieving strong performance in tool-using PRMs.

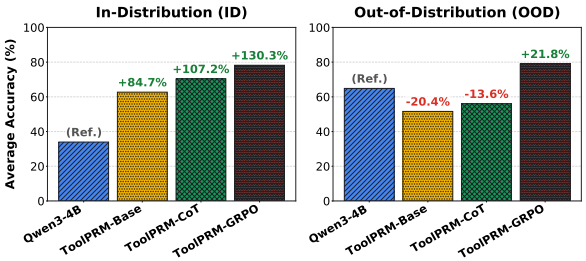

Figure 6: ToolPRMs' results in ID and OOD settings.

In-distribution vs. out-of-distribution generalization. Figure 6 compares three ToolPRM variants under in-distribution (ID) and out-of-distribution (OOD) evaluation settings. ToolPRM-Base and ToolPRM-CoT, both trained using supervised fine-tuning, show clear improvements in the ID setting. However, their performance drops substantially in the OOD setting, with relative decreases of 20.4% and 13.6%, respectively. This behavior suggests that SFT-based methods are prone to overfitting and may rely on distribution-specific patterns that do not generalize well.

In contrast, ToolPRM-GRPO achieves consistent gains in both ID and OOD evaluations, with a 21.8% improvement in the OOD setting. This result indicates that reinforcement learning encourages more robust decision boundaries and reduces reliance on spurious correlations. Overall, these findings suggest that RL-based training is a promising direction for ToolPRM learning, particularly when generalization is required beyond the training distribution.

5 Further Analysis

5.1 Meta-Evaluation

To examine whether ToolPRMBench reflects PRM performance under realistic and dynamic conditions, we conduct a meta-evaluation on GTA and BFCL. Specifically, we use different models as reward functions to guide best-of-n search with $n = 8$, and measure the resulting performance gains. Figure 7 illustrates the correlation between ToolPRMBench accuracy and the effectiveness of reward-guided search.

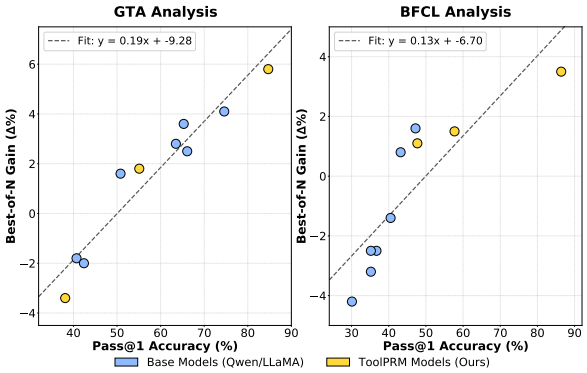

Figure 7: Meta-evaluation of ToolPRMBench on GTA and BFCL.

We observe a strong positive correlation on both benchmarks. Models that perform well on ToolPRMBench consistently yield larger gains during reward-guided search, indicating that ToolPRMBench is a reliable proxy for PRM effectiveness in inference-time decision making. At the same time, models with poor ToolPRMBench performance (accuracy below 50%) often yield negative gains. In these cases, using such models as reward functions actively harms performance, as they tend to misguide exploration and amplify incorrect trajectories.

5.2 Can Synthetic Data Improve ToolPRMs?

Collecting high-quality pairwise data for ToolPRM training is costly and challenging. To alleviate this issue, we explore a simple data synthesis strategy that constructs preference pairs by directly inserting incorrect actions into ground-truth trajectories following (Wang et al., 2024c). This approach avoids additional rollouts and reduces annotation cost while preserving the overall task structure.

We apply this data synthesis strategy to GTA and ToolTalk, and train both ToolPRM-Base and ToolPRM-GRPO using the synthesized data. As shown in Figure 8, synthetic data leads to substantial improvements on GTA, with both models achieving over 22% relative gains. However, the effect on ToolTalk is much weaker. ToolPRM-Base-Syn slightly degrades performance, whereas ToolPRM-GRPO-Syn shows only marginal improvement.

These results suggest that synthetic data is a promising direction. However, its effectiveness strongly depends on the task and environment. Designing more realistic and diverse synthetic errors remains a significant challenge, and more advanced

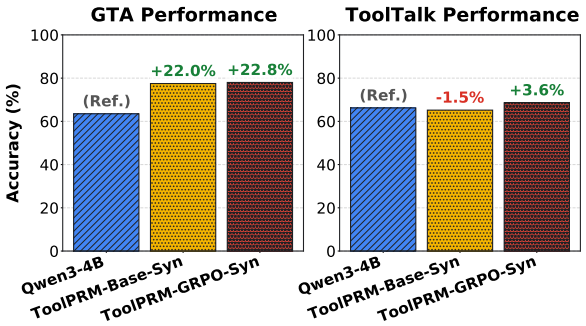

Figure 8: ToolPRMs' results with synthetic data on GTA and ToolTalk.

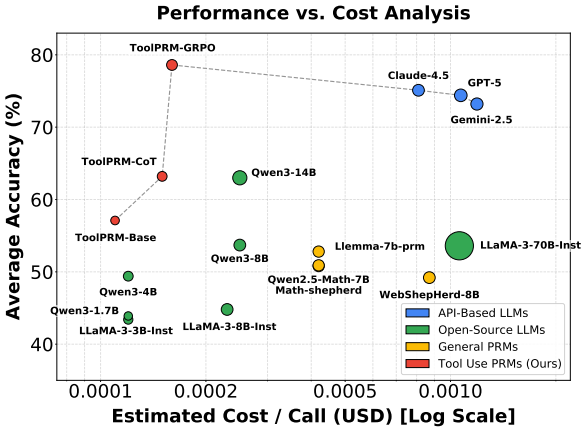

Figure 9: Cost analysis result across 4 types of models.

synthesis strategies are needed for broader applicability.

5.3 Cost Analysis

We further analyze the trade-off between performance and inference cost across different model categories, as shown in Figure 9. For API-based LLMs, we estimate the per-call cost using the official API pricing provided by the respective service. For open-source LLMs, we adopt the pricing provided by Together.ai[2] under a unified inference setup. For ToolPRMs and other general PRMs, we use the cost of their corresponding base model.

The results reveal a clear performance–cost trade-off. API-based LLMs achieve strong performance on ToolPRMBench, but their inference costs are significantly higher than those of other models. In contrast, ToolPRMs operate at a much lower cost while still delivering competitive accuracy, substantially outperforming open-source LLMs and general PRMs under similar or even lower budgets. This finding further supports the practicality and promise of tool-specific PRMs for reward-guided inference-time scaling in real-world agent systems.

[2]https://www.together.ai/

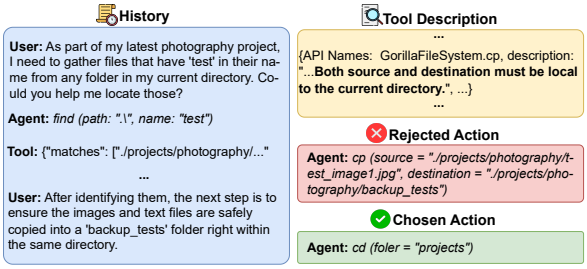

Figure 10: A case study from the BFCL subset.

5.4 Case Study

Figure 10 presents a representative example from the BFCL subset in ToolPRMBench, illustrating a common process-level error in tool-using agents. The user first requests to locate files containing "test" in their name, which the agent correctly accomplishes using the `find` tool. After the relevant files and directories are identified, the user asks to copy the images and text files into the `backup_tests` directory. Although the rejected action correctly reflects the high-level intent of copying files, it violates the tool specification of `cp`, which requires both the source and destination to be local to the current working directory. By directly providing file paths, the agent ignores the implicit state constraint imposed by the file system tools. In contrast, the chosen action first changes the working directory using `cd`, which is a necessary precondition for performing valid copy operations under the given tool interface. This example highlights that errors in tool-using agents often arise not from misunderstanding user intent, but from failing to satisfy low-level tool constraints and state transitions.

6 Conclusion

We present ToolPRMBench, a large-scale benchmark for evaluating process reward models in tool-using agent settings. It's collected across diverse Tool-using benchmarks, combining both offline and online trajectory sampling. Through extensive evaluation on ToolPRMBench, we observe clear and consistent differences in tool process reward modeling across model families. The results indicate that increased model scale and stronger general capabilities are beneficial, but they are not sufficient without specialized training. Reinforcement learning plays a key role in improving ro-

bustness and generalization. We further conduct a set of complementary analyses on ToolPRM-Bench, including meta-evaluation with reward-guided search, studies on distribution shift, synthetic data, and efficiency, offering practical insights for future research on reliable and scalable reward modeling in tool-using agents.

Limitations

Despite the promising results demonstrated by ToolPRMBench, our study has several limitations that provide directions for future research. First, although recent studies have highlighted the potential of inference-time scaling methods with RL (Qian et al., 2025), we did not conduct extensive evaluations of these search-based strategies on our benchmark. Given the constraints on time and available computing resources, our experiments primarily focus on the intrinsic discriminative ability of PRMs rather than their end-to-end impact under heavy training budgets. Future work could explore more efficient RL algorithms to bridge this gap. Second, the current construction of ToolPRMBench is based on a selected set of representative tool-using benchmarks. Recently, new datasets and protocols based on the Model Context Protocol (MCP) have emerged, offering more standardized ways for agents to interact with diverse tools. However, incorporating these MCP-based environments often involves high data collection costs and complex environment setups. Due to budget considerations during the data collection phase, we did not include these specific datasets in the current version of our benchmark. Expanding the scope to include MCP-compatible tools would likely enhance the diversity and real-world applicability of the evaluation.

Acknowledgement

This work was partially supported by a grant from the Intuit University Collaboration Program.

References

Mayank Agarwal, Ibrahim Abdelaziz, Kinjal Basu, Merve Unuvar, Luis A Lastras, Yara Rizk, and Pavan Kapanipathi. 2025. Toolrm: Outcome reward models for tool-calling large language models. *arXiv preprint arXiv:2509.11963*.

Anthropic. 2025. Introducing claude sonnet 4.5. https://www.anthropic.com/news/claude-sonnet-4-5. Official announcement of Claude Sonnet 4.5, a next-generation AI model optimized for coding, agents, and sustained reasoning tasks.

Hyungjoo Chae, Sunghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, and 1 others. 2025. Web-shepherd: Advancing prms for reinforcing web agents. *arXiv preprint arXiv:2505.15277*.

Zehui Chen, Weihua Du, Wenwei Zhang, Kuikun Liu, Jiangning Liu, Miao Zheng, Jingming Zhuo, Songyang Zhang, Dahua Lin, Kai Chen, and 1 others. 2024. T-eval: Evaluating the tool utilization capability of large language models step by step. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9510–9529.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Nicholas Farn and Richard Shin. 2023. Tooltalk: Evaluating tool-usage in a conversational setting. *arXiv preprint arXiv:2311.10775*.

Xuanqi Gao, Siyi Xie, Juan Zhai, Shiqing Ma, and Chao Shen. 2025. Mcp-radar: A multi-dimensional benchmark for evaluating tool use capabilities in large language models. *arXiv preprint arXiv:2505.16700*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Pengfei He, Zhenwei Dai, Bing He, Hui Liu, Xianfeng Tang, Hanqing Lu, Juanhui Li, Jiayuan Ding, Subhabrata Mukherjee, Suhang Wang, and 1 others. 2025. Traject-bench: A trajectory-aware benchmark for evaluating agentic tool use. *arXiv preprint arXiv:2510.04550*.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and 1 others. 2024. Metatool benchmark for large language models: Deciding whether to use tools and which to use. In *ICLR*.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu,

and 1 others. 2025. From generation to judgment: Opportunities and challenges of llm-as-a-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2757–2791.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, and 1 others. 2025. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stańczak, Peter Shaw, Christopher J Pal, and Siva Reddy. 2025. Agentrewardbench: Evaluating automatic evaluations of web agent trajectories. *arXiv preprint arXiv:2504.08942*.

Tianyi Men, Zhuoran Jin, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2025. Agent-RewardBench: Towards a unified benchmark for reward modeling across perception, planning, and safety in real-world multimodal agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17521–17541, Vienna, Austria. Association for Computational Linguistics.

Kangyun Ning, Yisong Su, Xueqiang Lv, Yuanzhe Zhang, Jian Liu, Kang Liu, and Jinan Xu. 2024. Wtu-eval: A whether-or-not tool usage evaluation benchmark for large language models. *arXiv preprint arXiv:2407.12823*.

OpenAI. 2025. Introducing gpt-5. https://openai.com/index/introducing-gpt-5/. Official announcement of OpenAI's GPT-5, a unified and state-of-the-art AI system available August 7, 2025.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Zhuocheng Shen. 2024. Llm with tools: A survey. *arXiv preprint arXiv:2409.18807*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. PRMBench: A fine-grained and challenging benchmark for process-level reward models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25299–25346, Vienna, Austria. Association for Computational Linguistics.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021.

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. 2024. Easy-to-hard generalization: Scalable alignment beyond human supervision. *arXiv preprint arXiv:2403.09472*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

Jize Wang, Ma Zerun, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. 2024a. Gta: a benchmark for general tool agents. *Advances in Neural Information Processing Systems*, 37:75749–75790.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Tianlu Wang, Ilia Kulikov, Olga Golovneva, Ping Yu, Weizhe Yuan, Jane Dwivedi-Yu, Richard Yuanzhe Pang, Maryam Fazel-Zarandi, Jason Weston, and Xian Li. 2024c. Self-taught evaluators. *arXiv preprint arXiv:2408.02666*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Junjie Ye, Zhengyin Du, Xuesong Yao, Weijian Lin, Yufei Xu, Zehui Chen, Zaiyuan Wang, Sining Zhu, Zhiheng Xi, Siyu Yuan, and 1 others. 2025. Toolhop: A query-driven benchmark for evaluating large language models in multi-hop tool use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2995–3021.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and 1 others. 2025. Genprm: Scaling test-time compute of process reward models via generative reasoning. *arXiv preprint arXiv:2504.00891*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, pages 62138–62160.

A More Details of ToolPRMBench

Table 3 presents the details of the training-testing split in ToolPRMBench. We also show the prompt template we use in LLM-based annotation and multi-LLM verification in the tables below.

Dataset	#train	#test	#all
BFCL	243	111	354
ToolSandbox	299	130	429
GTA	0	118	118
ToolTalk	0	86	86
all	**542**	**445**	**987**

Table 3: Details statistics of ToolPRMBench.

PRM Annotation Prompt for ToolSandbox

```
You are an annotator LLM for building a process reward model benchmark.

Given an INPUT_JSON with: sample_id, trajectory, available_tools, milestones, milestone_edges, and raw tool feedback.

Do: Identify the first incorrect agent step. A step is incorrect if it violates ANY of:

  • wrong tool choice,
  • wrong tool parameters,
  • incorrect user-facing response,
  • skipped or unordered milestone,
  • performing a later milestone before an earlier required one.

Output a single JSON object (no extra text) with this exact schema:
{
"sample_id": "<string>",
"history": [...],
"action_rejected": <the incorrect agent action>,
"action_chosen": <the corrected action>,
"rationale": "<short, concrete explanation>",
"error type": "<short phrase>"
}
```

PRM Annotation Prompt for BFCL

```
You are an annotator LLM for building process reward model benchmark samples.

Each input is a JSON object with keys: id, model_name, test_category, valid, error, execution_result, prompt, possible_answer, history, and function.

Your Task: Return one JSON object (no extra text) with this schema:
{
"sample_id": "<same as input>",
"history": [...],
"action_rejected": { "name": "<...>", "parameters": { ... } },
"action_chosen": { "name": "<...>", "parameters": { ... } },
"rationale": "1-2 sentences explaining the judgment.",
"error type": "<short phrase>"
}
```

B More Details of Experiment Implementation

B.1 Training Environment and Infrastructure

All experiments are conducted on a single machine equipped with 8 × NVIDIA H20 (96GB) GPUs. We utilize the DeepSpeed library to optimize training efficiency. Specifically, **DeepSpeed ZeRO-3** is employed to shard model states, gradients, and optimizer states across all 8 GPUs, enabling full-parameter fine-tuning of the backbone model within the available memory budget.

B.2 Supervised Fine-Tuning (SFT)

We use **Qwen3-4B** as our base model. The SFT stage is implemented using the LLaMA-Factory framework. We perform full-parameter fine-tuning for 2 epochs using the AdamW optimizer. The learning rate is set to 1.0×10^{-5} with a **cosine** scheduler and a warmup ratio of 0.1. To accommodate long-context reasoning in Chain-of-Thought (CoT) tasks, the **cutoff length** is set to 4,096 tokens. For the standard base tasks, we use a per-device batch size of 4. For CoT-enhanced datasets, we reduce the per-device batch size to 1 to manage memory consumption. All SFT experiments are performed in **bf16** precision.

B.3 Reinforcement Learning (RL)

Following SFT, we perform reinforcement learning using the **Group Relative Policy Optimization (GRPO)** algorithm, implemented via the TRL library. The model is trained for 1 epoch with a peak learning rate of 2.0×10^{-6}. In each training step, GRPO generates a group of $G = 8$ completions per prompt to compute relative rewards.

The maximum prompt length is 2,048 tokens, and the maximum completion length is 4,096 tokens to provide sufficient space for CoT reasoning. We use a **KL divergence coefficient** of $\beta = 0.01$ to regularize the policy. The reward function is

defined based on binary accuracy: a reward of 1.0 is assigned if the ground-truth answer is present in the model's output following the </think> token; otherwise, the reward is 0.0.

Parameter	SFT	RL (GRPO)
Backbone Model	Qwen3-4B	Qwen3-4B (SFT)
Hardware	$1 \times$ 8-H20 GPU Server	
Optimizer	AdamW	AdamW
Learning Rate	1.0×10^{-5}	2.0×10^{-6}
LR Scheduler	Cosine	Cosine
Training Epochs	2.0	1.0
Precision	bf16	bf16
Max Length	4,096	4,096
Batch Size / Dev	1 / 4	4
Group Size (G)	N/A	8
KL Coeff (β)	N/A	0.01
DeepSpeed Stage	ZeRO-3	ZeRO-3

Table 4: Hyperparameters for SFT and RL (GRPO) training stages.

We also present the prompt we used to evaluate various models below, as well as the prompt template we used to distill CoT from the teacher model and to perform data synthesis.

Standard Evaluation Prompt

```
Given the function descriptin, history, and the following two actions, which action is the correct one that could help to finish the task:

## Function Description: {functions}

## History: {history}

## Action1: {action_1}

## Action2: {action_2}

Please generate your annswer in a JSON format:
{
"rationale": <reasoning process for the judgment>,
"chosen action": <"Action1" or "Action2">
}
```

Evaluation Prompt for ToolPRM-Base and ToolPRM-GRPO

```
Given the interaction history, function description and two actions, which action is the correct intermidiate step that could help in finishing the task:

## history: {history}

## function description: {func_desc}

## action_1: {action_1}

## action_2: {action_2}

Please only generate action_1 or action_2 as the final answer.
```

Evaluation Prompt for ToolPRM-CoT

```
You are an expert evaluator. You are given: interaction history, available tool descriptions, and two candidate assistant actions.

Your task is to: Produce a concise chain-of-thought style rationale (2-6 sentences) that explains which of the two actions is better. Also indicate the winning action.

Schema:
{
"rationale": "concise stepwise rationale (2-6 sentences)",
"winning_action": "action_1|action_2"
}

Now INPUT:
available_tools: {tool_desc}
history: {history}
action_1: {action_1}
action_2: {action_2}
```

CoT Distillation Prompt

```
System: You are an expert evaluator. Produce a concise chain-of-thought rationale and a final judgment.

User: You are given:
- an interaction history,
- available tool / function descriptions,
- and two candidate assistant actions (one correct, one incorrect).

Your task is to:
Produce a concise chain-of-thought style rationale (2-6 sentences) that explains, step by step, why one action is better than the other given the context.
Also indicate the winning action as "action_1" or "action_2".

Schema (you MUST output JSON):
{
"rationale": "concise stepwise rationale (2-6 sentences)",
"winning_action": "action_1|action_2"
}

Now INPUT:
available_tools:
{tool_desc}

history:
{history_str}

action_1:
```

```
{action_1}

action_2:
{action_2}
```

Error Injection Prompt

```
Task:
You are given the golden conversation history (an array of objects with keys 'role' and 'content') and available tool descriptions (contained in history/system messages).

1. Identify a single 'assistant' step where a model is likely to make a mistake (e.g., wrong tool, wrong parameters, hallucinated information, or incorrect response format).
2. You MUST produce a mistake of type: {error_type}.
3. Construct the 'action_rejected' (the mistaken version) and 'action_chosen' (the original correct version).
4. Output a valid JSON object only.

Schema:
{
"chosen_index": <int, the 0-based index of the assistant message in history to corrupt>,
"history": <array of messages up to but NOT including the chosen_index>,
"action_chosen": <the original content/tool_call at chosen_index>,
"action_rejected": <the new corrupted/mistaken content or tool_call>,
"rationale": "<short explanation of why this error is plausible>",
"error type": "<short phrase summarizing the error>"
}

Golden History:
{history_str}
```

C The Use of LLMs for Writing

We employed Google's Gemini 2.5 Pro and OpenAI's GPT-5 as writing assistance tools during the preparation of this manuscript. Their role was exclusively for language refinement, such as improving readability and rephrasing for clarity in an academic writing style. This usage aligns with standard academic practices for language polishing.