AgentPRM: Scalable Process Reward Models for Language Model Agents via Step-Wise Promise and Progress

Anonymous ACL submission

Abstract

Despite advancements in large language models (LLMs), they still face challenges in multi-turn decision-making tasks (i.e., agent tasks), where models need to make a sequence of intelligent decisions based on environment feedback. In this work, we explore training process reward models (PRMs) to evaluate each decision and guide model's search in agent tasks. Unlike LLM reasoning, where each step is scored based on correctness, actions in agent tasks do not have a clear-cut correctness. Instead, they should be evaluated based on their proximity to the goal and the progress they have made. Building on this insight, we redefine the PRM for agent tasks, and introduce AgentPRM to capture both the interdependence between sequential decisions and their contribution to the final goal. This allows for better progress tracking and exploration-exploitation balance. To scalably obtain labeled data for training AgentPRM, we employ a TD-based estimation method combined with Generalized Advantage Estimation (GAE). Experiments across sampling strategies, models and tasks demonstrate that our method consistently outperforms baselines, is more compute-efficient, and it exhibits a more stable and robust improvement trend as inference compute scales. Furthermore, it generalizes well on mathematical reasoning. [1]

1 Introduction

With the development of large language models (LLMs), they have achieved significant advancement in text completion and generation tasks, such as summarization, translation, and reasoning (QwenTeam, 2024; OpenAI, 2024; Anil et al., 2023). However, they still face challenges in multi-turn decision-making tasks (i.e., agent tasks), where the models need to make a sequence of intelligent decisions based on feedback from the environment (Xi et al., 2023; Zeng et al., 2024). In

[1] We will release our codes and data for further research.

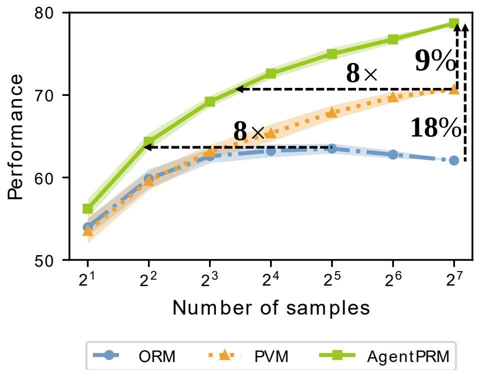


Figure 1: Average Best-of-N performance across three agent tasks. AgentPRM outperforms other baselines. It is more than $8\times$ compute-efficient than ORM and PVM baselines. Moreover, it demonstrates a more stable and robust improvement trend as inference compute scaling.

agent tasks, the decisions made at each step are not isolated, but rather form a chain of dependencies, where each decision influences subsequent decisions and ultimately the outcome (Liu et al., 2024b; Chen et al., 2024b; Xi et al., 2024). Hence, to achieve excellent performance, models are required to possess task knowledge, understand environmental information, and engage in forward-looking planning (Xi et al., 2023; Wang et al., 2024b).

Previous work primarily focuses on fine-tuning-based (Zeng et al., 2024; Chen et al., 2024b) or prompt engineering-based approaches (Yao et al., 2023; Liu et al., 2023; Shinn et al., 2023). The former involves collecting expert data to have the model imitate decisions, which lacks sufficient exploration by the model itself and an understanding of the value and reason behind each decision. The latter leverages state-of-the-art commercial models like GPT-4 to perform self-reflection, which is limited by APIs, making it costly and difficult to train or customize for deployment. As a result, the performance meets bottlenecks (Yang et al., 2023; Koh et al., 2024).

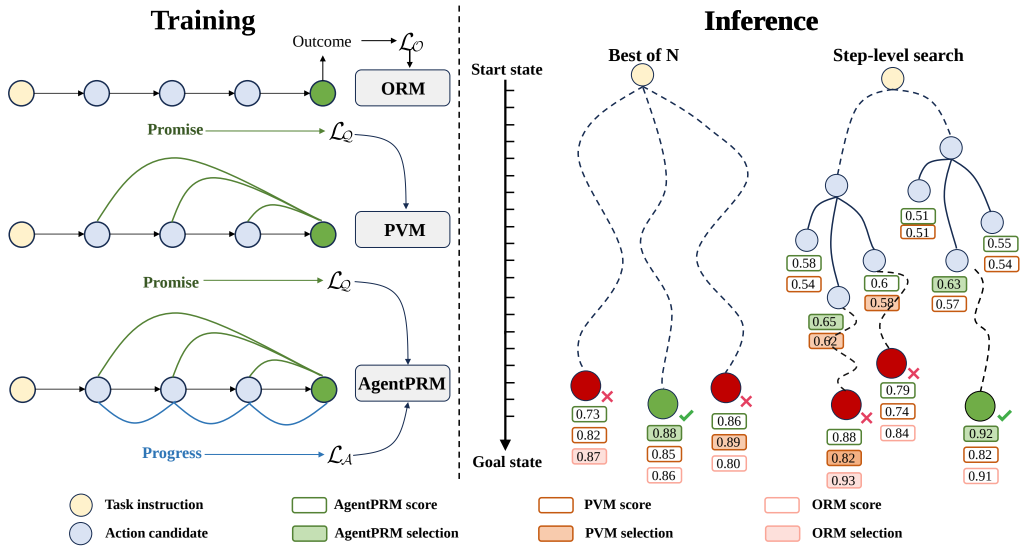


Figure 2: Overview of AgentPRM and other baselines. In training, AgentPRM takes into account both the probability of each step achieving the goal (promise) and the interdependence between sequential steps (progress). This boosts AgentPRM's performance at inference-time.

To this end, we draw inspiration from process supervision in LLM reasoning and explore training process reward models (PRMs) to guide the search and exploration of LLMs on agent tasks (Uesato et al., 2022; Wang et al., 2024c; Lightman et al., 2024; Setlur et al., 2024; Li and Li, 2024). In LLM reasoning, PRMs are used to rate each step based on its correctness and guide the decoding process. However, in agent tasks, they face three key challenges: (1) actions in agent tasks do not have a definitive correctness, making evaluation non-trivial. For example, in web navigation, even if the model makes a poor decision, it can still correct it by backtracking (Yao et al., 2022; Zhou et al., 2024). (2) Previous methods treat each step independently and do not account for the dependencies and nuances among steps within a trajectory (Li and Li, 2024), which is inconsistent with the nature of agent tasks. (3) Previous methods for training PRMs often require expert-level annotations or a large amount of sampling (Wang et al., 2024c; Luo et al., 2024), which can be very costly for real-world agent tasks.

In this work, we propose AgentPRM to address the challenges. It measures the proximity of steps to the goal state and the progress made by LLMs. Specifically, AgentPRM predicts the contribution of each decision to the final goal and captures the dependencies between sequential decisions, enabling better progress tracking and a balance between exploration and exploitation. To scale the acquisition of training data, we employ an automated TD-based method with GAE (Schulman et al., 2016), which is more efficient than previous Monte-Carlo-based methods (Wang et al., 2024c). Extensive experiments across various models and tasks show that AgentPRM consistently outperforms baselines while being more compute-efficient. For example, across three agent tasks and multiple sampling strategies, it achieves an average compute efficiency that is more than $8\times$ greater than baseline RMs with Qwen2.5-3B. Additionally, AgentPRM exhibits a more stable and robust improvement trend as inference compute scales, highlighting its promising potential for training more policy LMs through self-improvement or reinforcement learning (RL).

To summarize, our main contributions are:

1. We propose AgentPRM, a new process reward model to evaluate LLMs' actions in agent tasks and guide their search, which captures the dependency between sequential steps and their contribution to the final goal.

2. We propose an automated, scalable method for training AgentPRM. We perform extensive experiments to demonstrate the efficiency and effectiveness of AgentPRM.

3. We conduct in-depth ablation and analysis to show how it works. We hope AgentPRM provides insights for developing better language model agents for the community.

2 Preliminary and Background

The agent task can be formalized as a Partially Observable Markov Decision Process (POMDP) $(\mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, r)$ (Hausknecht and Stone, 2015; Xi et al., 2024), where $\mathcal{U}$ is the instruction space, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{O}$ is the observation space, $\mathcal{T}: \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the deterministic state transition function, and $r: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function. Given a task instruction $u \in \mathcal{U}$, the initial observation $o_0 \in \mathcal{O}$, and the initial state $s_0 = \{u, o_0\}$, the agent task requires the language model to generate an action $a_0 \sim \pi_\theta(\cdot|s_0)$ based on its policy π_θ parameterized by θ, where $a_0 \in \mathcal{A}$. The agent receives an observation $o_1 \in O$ from environment, and the state is then transitioned to $s_1 = \{u, s_0, a_0, o_1\}$ according to $\mathcal{T}$. Following the process, the agent proposes a sequence of actions $\{a_t\}_{t=0}^{T}$, where T is the number of steps, to interact with the environment until the task is completed or the maximum number of steps is reached: $\tau = (u, o_0, a_0, o_1, \cdots, o_T, a_T)$. Then for a language model, the agent task can be formalized as:

$$\pi_\theta(\tau|s_0) = \prod_{t=0}^{T} \pi_\theta(a_t|s_t), \tag{1}$$

where s_t represents the interaction history up to timestep t. Finally, the environment e provides an outcome reward $r(u, \tau) \in [0, 1]$ to describe the completion of the agent task.

Outcome Reward Model (ORM). An ORM r_{orm} takes a trajectory τ as input and outputs a score to predict whether this trajectory has completed the instruction u (Uesato et al., 2022; Ouyang et al., 2022). The ORMs are trained using data sampled from the policy model π_θ, which generates instruction-trajectory pairs. The model is trained to fit the output of the outcome reward function r (Cobbe et al., 2021; Ouyang et al., 2022; Liu et al., 2024a).

Process Reward Model (PRM). A PRM scores the actions or states at each step of a trajectory (Lightman et al., 2024; Wang et al., 2024c; Zhang et al., 2024). In the field of LLM reasoning, the scoring criterion is usually the correctness of the steps (Lightman et al., 2024; Wang et al., 2024c; Zhang et al., 2024). However, this is not suitable for the agent tasks we are studying, which will be elaborated in Section 3 and we will provide appropriate evaluation criteria. During training, annotated labels for each step are collected, and PRMs are trained to fit these labels (Lightman et al., 2024; Wang et al., 2024c; Luo et al., 2024).

Best-of-N (BoN) with reward models. With increased inference compute, we can apply BoN (Best of N) for improved performance (Touvron et al., 2023). The process involves first sampling N trajectories $\tau_{i=1}^{N}$ using the policy π_θ, then scoring these trajectories using reward models. The trajectory with the highest score is selected as the final output. Note that BoN can also be executed with PRMs (Lightman et al., 2024). In this work, we use the score of the last step as the score of a trajectory.

Search with process reward models. During the inference phase, we can conduct step-level search against PRMs for agent tasks. Among the various step-level search algorithms, beam search is a widely used method that achieves a balance between performance and efficiency (Zhang et al., 2024; Chen et al., 2024a). In each iteration, beam search expands M candidate action nodes for each node and uses PRM to score the candidates. Only the top-N scored nodes are retained in each iteration, and we select the terminal state with the highest score as the final response.

3 Methodology

3.1 Motivation

In LLM reasoning, researchers train PRMs by collecting annotated data to score each step based on its correctness. However, for agent tasks, we face three major challenges: (1) Decisions in agent tasks do not have a clear-cut correctness, making evaluation non-trivial. For example, in web navigation, if the model makes a poor decision by clicking a button and navigating to a new page, it can immediately correct this by using the back button to return to the previous state (Yao et al., 2022; Zhou et al., 2024). (2) Previous PRMs typically treat each state independently and do not account for the dependencies between different decisions (Li and Li, 2024; Setlur et al., 2024). However, in agent tasks, the decisions made at each step are not isolated, but rather form a chain of dependencies, where each decision influences subsequent decisions and ultimately the outcome (Yao et al., 2022; Xi et al., 2023, 2024; Chevalier-Boisvert et al., 2019). (3) Previous methods for training PRMs often require expert annotations or a large amount of sampling, making them expensive (Lightman et al., 2024; Wang et al., 2024c; Luo et al., 2024).

Therefore, our research question includes: how to define appropriate rewards for decisions and efficiently train such process reward models.

3.2 AgentPRM: Re-Defining Process Rewards for Decisions in Agent Tasks

Measuring expected future success probability with value functions. An agent task typically requires making a sequence of intelligent decisions to reach the goal state. Conceptually, this means we need to evaluate whether a decision brings the state closer to the goal (Yao et al., 2022; Xi et al., 2023; Liu et al., 2024b). In RL, this is often defined as the action-value function $Q^\pi(s_t, a_t)$ (Sutton and Barto, 2018), which measures the likelihood of future success after taking a particular action a_t based on state s_t:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{\tau \sim \pi(\cdot|s_t, a_t)} \left[r(u, \tau) \right].$$

Similarly, we can define the state-value function $V(s_t)$ (Sutton and Barto, 2018) with:

$$V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t)} \left[Q^\pi(s_t, a_t) \right].$$

Now, given annotated labels for each state-action tuple, $\mathcal{D}_Q = \{s_t, a_t, \hat{V}(s_t, a_t)\}$, we train our PRM $\mathcal{M}_\phi$ parameterized by ϕ to predict the action value with mean squared error (MSE) loss (Chen et al., 2024a):

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{s_t, a_t \sim \mathcal{D}_Q} \left[\frac{1}{2} (\mathcal{M}_\phi(s_t, a_t) - \hat{Q}(s_t, a_t))^2 \right]. \tag{2}$$

After training, Based on the predictions of $\mathcal{M}_\phi$, we can perform inference-time search or BoN.

Capturing dependencies between steps with advantages. Nevertheless, the aforementioned $\mathcal{M}_\phi$ considers only the actions' contribution to the final goal, and fails to effectively capture the relationships and dependencies between consecutive states or decisions (Li and Li, 2024; Setlur et al., 2024). In other words, it primarily measures promise but not progress. This often leads to high exploitation while lacking a balance with exploration (Setlur et al., 2024; Snell et al., 2024). However, in many agent tasks, models need sufficient exploration to successfully achieve the final goal (Chevalier-Boisvert et al., 2019; Yao et al., 2022; Zhou et al., 2024). For example, in a web navigation task, the model needs to first navigate to the login page to log in, and then return to the current page to post a comment. Although the action of entering the login page may temporarily move the model away from the target page, it is still crucial because it is a necessary step to log in before posting.

Therefore, we point out that in addition to promise, process rewards for agent decisions should capture the dependencies between steps to measure local progress. This can be measured by advantage in RL (Sutton et al., 1999), which quantifies the change in the likelihood of success before and after a given action:

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t).$$

The value of $A^\pi(s_t, a_t)$ can be either positive or negative. If it is positive, it indicates that the current action has contributed to positive progress; otherwise, it indicates negative progress.

Hence, to train our model $\mathcal{M}_\phi$ to capture progress and dependencies between actions, we introduce the loss for fitting advantage:

$$\mathcal{L}_A(\phi) = \mathbb{E}_{s_t, a_t \sim D_Q} \left[(A_\phi(s_t, a_t) - \hat{A}(s_t, a_t))^2 \right], \tag{3}$$

where $\hat{A}(s_t, a_t)$ is the annotated labels for advantage. Next, we show how to integrate the fitting optimization of the advantage into the training of our PRM. As the state transition in our setting is deterministic, we have (Li and Li, 2024; Setlur et al., 2024):

$$Q(s_t, a_t) - V(s_t) = Q(s_t, a_t) - Q(s_{t-1}, a_{t-1}) \tag{4}$$

So the loss term for advantage $\mathcal{L}_A(\phi)$ becomes:

$$\mathbb{E}_{s_t, a_t \sim D_Q} \Big[\Big(((\mathcal{M}_\phi(s_t, a_t) - \mathcal{M}_\phi(s_{t-1}, a_{t-1})) - (\hat{Q}(s_t, a_t) - \hat{Q}(s_{t-1}, a_{t-1})) \Big)^2 \Big]. \tag{5}$$

In this way, we can optimize our PRMs for considering not only the contribution to the final outcome but also the dependencies between adjacent actions, and our final loss for AgentPRM becomes:

$$\mathcal{L}_{\text{AgentPRM}}(\phi) = \mathcal{L}_Q(\phi) + \beta \times \mathcal{L}_A(\phi), \tag{6}$$

where β is a scaling factor that balances the two loss terms.

3.3 Practical Implementation for Training AgentPRMs

In the previous section, we demonstrate how to optimize our AgentPRM based on the estimated or

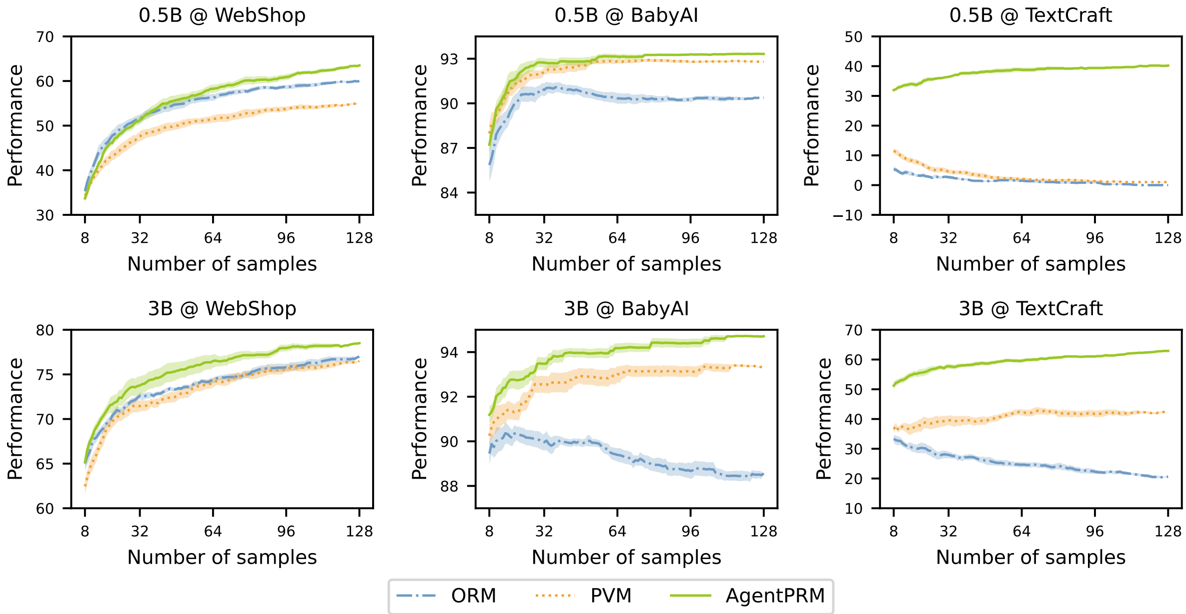


Figure 3: Performance of Best-of-N evaluation. AgentPRM outperforms other baselines, is more compute-efficient, and demonstrates a more stable and robust improvement trend as inference compute scales.

annotated data. Next, we explore how to obtain the data in an effective and scalable way.

MC-based estimation. A common method for automatically estimating the Q-value of an action is based on Monte Carlo (MC) sampling (Wang et al., 2024c; Luo et al., 2024). Specifically, it first samples N_{Traj} seed trajectories $\{\tau_i\}_{i=1}^{N_{\text{Traj}}}$ from the policy π_θ. Then, for each action $a_{i,t}$ in each trajectory τ_i, we start from the next state $s_{i,t+1}$ derived from the action and perform N_{mc} rollouts $\{\tau'_j\}_{j=1}^{N_{\text{mc}}}$ with π_θ. As in previous work, if any of the rollouts reaches the goal state, the value of this action is set to be 1:

$$\hat{Q}(s_t, a_t) = \begin{cases} 1 & \exists \tau'_j, \tau'_j \text{ is successful}, \\ 0 & \text{otherwise}, \end{cases} \tag{7}$$

Though MC-based estimation is effective, it still suffers from efficiency and cost issues because it requires a large number of rollouts for estimating different actions. Therefore, we explore using other more efficient methods.

TD-based estimation with GAE. To explore more efficient estimation methods, we are inspired by previous work (Sutton, 1988; Schulman et al., 2016; Ouyang et al., 2022; Sutton et al., 1999) and introduce TD-based methods, using GAE (Generalized Advantage Estimation) to reduce variance and improve stability (Schulman et al., 2016). First, we define the TD residual for a state as follows:

$$\delta(s_t, a_t) = r_t + \mathcal{M}(s_t, a_t) - \mathcal{M}(s_{t-1}, a_{t-1}), \tag{8}$$

where r_t is the instant reward at timestep t, and in our setting, sparse rewards are only assigned when $t = T$. Next, we estimate the advantage for different actions using Generalized Advantage Estimation (GAE):

$$\hat{A}(s_t, a_t) = \sum_{k=t+1}^{T} \lambda^{k-t} \hat{\delta}(s_t), \tag{9}$$

where λ is the discount factor. Finally, the current estimated $\hat{Q}(s_t, a_t)$ can be considered as the return of the next state $\hat{G}(s_{t+1})$:

$$\hat{Q}(s_t, a_t) = \hat{G}(s_{t+1}) = \hat{A}(s_t, a_t) + \mathcal{M}_\phi(s_t, a_t). \tag{10}$$

In implementation, we sample N_{TD} trajectories $\{\tau_i\}_{i=1}^{N_{\text{TD}}}$ from the policy π_θ for training. Since our estimation process involves prediction of $\mathcal{M}_\phi$, we iteratively sample a batch from the trajectory set, conduct estimation based on the current model, and update the model with Equation 5. We summarize the training algorithm of AgentPRM in Algorithm 1 of Appendix A.

From the efficiency perspective, TD-based estimation with GAE does not require additional rollouts from each state like MC-based method, saving a significant amount of computational resources

Model	Method	WebShop			BabyAI			TextCraft		
					Fine-tuning-based-methods					
Qwen2.5-0.5B	SFT	30.5			37.6			27.8		
	RFT	39.0			54.4			32.9		
					Reward-Model-based Beam Search					
		$@2 \times 2$	$@4 \times 4$	$@8 \times 8$	$@2 \times 2$	$@4 \times 4$	$@8 \times 8$	$@2 \times 2$	$@4 \times 4$	$@8 \times 8$
	ORM	19.5	18.5	8.0	73.8	74.9	78.8	25.7	24.7	27.8
	PVM	30.0	50.0	57.5	**84.6**	86.5	88.1	25.7	26.8	26.8
	AgentPRM	**30.5**	**51.5**	**62.5**	82.9	**87.7**	**90.4**	**28.8**	**29.9**	**32.9**
					Fine-tuning-based-methods					
Qwen2.5-3B	SFT	46.0			67.4			29.8		
	RFT	48.0			64.5			36.0		
					Reward-Model-based Beam Search					
		$@2 \times 2$	$@4 \times 4$	$@8 \times 8$	$@2 \times 2$	$@4 \times 4$	$@8 \times 8$	$@2 \times 2$	$@4 \times 4$	$@8 \times 8$
	ORM	51.0	59.0	57.0	83.9	83.5	83.7	38.1	41.2	43.3
	PVM	50.5	59.0	54.5	72.7	84.9	89.1	39.1	40.2	44.3
	AgentPRM	**61.0**	**72.5**	**76.0**	**84.4**	**89.6**	**89.8**	**47.4**	**51.5**	**56.7**

Table 1: Evaluation results of fine-tuning-based methods and beam search on agent tasks. The best performance is in **bold**. Our method is marked in blue. We set the $@N \times M$ in each beam search setting.

(See Section 5.3). From the performance perspective, though TD-based methods have concerns regarding high variance, we introduce GAE to reduce variance and improve stability, ultimately achieving better performance.

4 Experiments

4.1 Experimental Setup

Tasks. We conduct our experiments on three challenging agent environments: WebShop (Yao et al., 2022), BabyAI (Chevalier-Boisvert et al., 2019), and TextCraft (Prasad et al., 2024). The detailed description is in Appendix B. To show the comprehensiveness of our method, we include reasoning dataset GSM8K (Cobbe et al., 2021) Our implementation is based on the AgentGym framework (Xi et al., 2024).

Baselines. We compare our AgentPRM with several fine-tuning and reward model-based methods. For fine-tuning-based approaches, supervised fine-tuning (SFT) uses expert data to fine-tune the base model, while rejection sampling fine-tuning (RFT) refines the model by leveraging successful trajectories generated by the base model. For reward model-based methods, we include ORM (Cobbe et al., 2021) and PVM (Process Value Models) (Lightman et al., 2024). ORM estimates the reward for the outcome, while PVM estimates step-level values by assigning the reward of a trajectory to individual steps. We also include MC-based method Math-Shepherd (Wang et al., 2024c) to estimate Q-Value of each step in Section 5.3.

Implementation Details. All experiments are conducted with A100-80GB GPUs. Our backbone models include Qwen-2.5-0.5B-Instruct and Qwen-2.5-3B-Instruct. For agent tasks, we use the ReAct format (Yao et al., 2023) for model outputs. To initialize the models, we randomly select 300 trajectories from the AgentGym training set. For SFT, we set the learning rate to 1×10^{-5}. We report the success rate for WebShop and TextCraft, and the reward for BabyAI. For MC-based estimation, we set $N_{\text{Traj}} = 1$ for each query, and $N_{\text{mc}} = 16$ for each step; for TD-based estimation, we set $N_{\text{TD}} = 16$ for each query. We train reward models for 5 epochs under a learning rate of 1×10^{-6}. For AgentPRM, we set $\beta = 1.0$ and $\lambda = 0.95$. See more details in Appendix C.

4.2 Main Results

Result 1: Compared to greedy decoding, introducing RMs for BoN and search can improve LM performance on agent tasks. The experimental results are shown in Figure 3 and Table 1. Compared to the greedy decoding of SFT and RFT methods, using reward functions for BoN and search can significantly improve model performance, especially when increasing inference compute for more sampling. This is consistent with previous work on test-time scaling (Snell et al., 2024; Bansal et al., 2024).

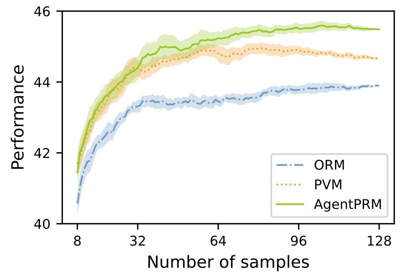


Figure 4: Evaluation results of Best-of-N on GSM8K.

Model	Method	GSM8K		
		@2 × 2	@4 × 4	@8 × 8
Qwen2.5-0.5B	ORM	39.5	42.9	44.2
	PVM	38.9	41.3	42.9
	AgentPRM	**41.5**	**44.7**	**45.7**
Qwen2.5-3B	ORM	64.1	70.3	72.9
	PVM	63.6	69.6	71.2
	AgentPRM	**65.1**	**70.5**	**73.4**

Table 2: Evaluation results of beam search on GSM8K.

Result 2: AgentPRM is more compute-efficient than other reward models, and outperforms them consistently in both Best-of-N and test-time search. As shown in Figure 3, under different sampling budgets in Best-of-N evaluation, our method consistently outperforms ORM and PVM across different tasks, demonstrating its effectiveness. Figure 1 shows that, on average, AgentPRM is 8× more compute-efficient than PVMs and ORMs. This highlights the potential of AgentPRM in training stronger agentic LLMs with methods like reinforcement learning, which we leave for future work.

As listed in Table 1, in beam search, our method also outperforms ORMs and PVMs across different tasks significantly, validating its ability to guide model search and achieve a good exploration-exploitation balance. For example, using Qwen2.5-3B on the WebShop task, with an 8×8 sampling search setting, our method surpasses PVM by more than 20.0 points.

Result 3: As inference compute scaling, AgentPRM demonstrates a more robust and stable scaling trend. In Figure 1 and Figure 3, we observe that as the sampling budget increases, PVMs and ORMs tend to experience performance bottleneck or even degradation. This aligns with the findings of Wang et al. (2025), and may be attributed to issues such as false positives or reward hacking (Wang et al., 2024a), which could limit their effectiveness in future RL and self-improvement-based methods for training better policy models. In contrast, AgentPRM consistently shows stable improvement, highlighting its robustness and broadening its potential for future applications.

5 Discussion and Analysis

5.1 Performance on Mathematical Reasoning

To demonstrate the versatility of our method, we also conducted experiments on mathematical tasks,

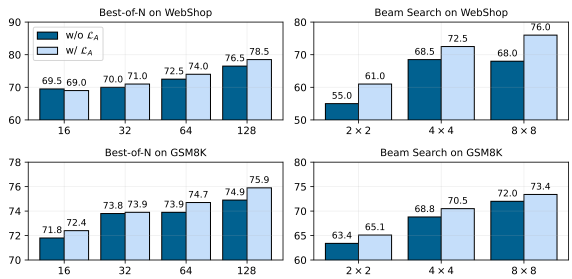


Figure 5: Ablation study on $\mathcal{L}_A$ with Qwen2.5-3B.

with results shown in Table 2 and Figure 4. As we can see, our method still performs exceptionally well on mathematical tasks, surpassing other baselines. This also highlights the generalizability and adaptability of our AgentPRM. We expect to extend it to more tasks in future work, such as coding or logical reasoning.

5.2 Ablation Study on $\mathcal{L}_A(\phi)$

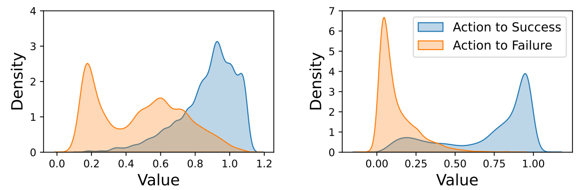
To capture the dependency between steps and evaluate their progress, we add $\mathcal{L}_A(\phi)$ for training AgentPRMs. Here, we conduct an ablation on the advantage term to validate its effect. Results in Figure 5 show that without $\mathcal{L}_A(\phi)$, the performance on both agent and mathematical tasks drops regardless of the sampling strategy, showing that capturing progress is important for training AgentPRMs.

5.3 Comparing Sampling Efficiency of Our Method with MC-based Estimation

In Section 3.3, we introduced TD-based estimation with GAE for the automated labeling process. Here, we compare it with the previously commonly used MC-based estimation. The experimental results are shown in Figure 5. We observe that our method requires fewer tokens for labeling the data compared to other methods, yet achieves better performance on Best-of-N and beam search, demonstrating the higher efficiency and effectiveness of our approach.

Task	Method	Tokens	Best-of-N				Beam Search	
			@8	@16	@32	@64	@4 × 4	@8 × 8
WebShop	MC-based	1.9×	63.5	67.5	69.0	72.0	67.5	70.5
	TD-based	1.0×	**64.5**	**69.0**	**71.0**	**74.0**	**72.5**	**76.0**
BabyAI	MC-based	2.8×	90.5	90.5	91.6	93.1	87.6	88.3
	TD-based	1.0×	**91.4**	**91.4**	**92.4**	**94.4**	**89.6**	**89.8**
GSM8K	MC-based	1.5×	61.8	63.3	63.9	65.0	66.1	70.1
	TD-based	1.0×	**68.9**	**72.4**	**73.9**	**74.7**	**70.5**	**73.4**

Table 3: Comparing sampling efficiency and performance of our method with MC-based estimation.



(a) WebShop

(b) BabyAI

Figure 6: Visualization of value distribution of Actions with AgentPRM.

5.4 Evaluating Value Distributions of Actions with AgentPRM

To further demonstrate the working mechanism of AgentPRM, we visualize the value estimates of the actions predicted by AgentPRM in WebShop and BabyAI across successful and unsuccessful trajectories. From the distribution in Figure 6, we observe that the model assigns higher scores to the actions that lead to positive goals, and lower scores to the actions that lead to negative goals, revealing that our method is effective in credit assignment.

We also perform qualitative analysis in Appendix D to show how AgentPRM works.

6 Related Work

Developing LLMs for agent tasks. To enable language models to perform well in multi-turn decision-making tasks (Chevalier-Boisvert et al., 2019; Yao et al., 2022; Zhou et al., 2024), previous work has proposed fine-tuning-based methods, where expert-labeled trajectories are collected, and the learner imitates them step by step (Chen et al., 2023, 2024b). However, this approach is often difficult to scale and lacks sufficient exploration of the environment by the model, as well as an understanding of the value and reasoning behind each decision (Chen et al., 2025; Lin et al., 2024). Another line of methods is based on state-of-the-art commercial models like GPT-4o for prompt engineering, which is limited by APIs, making it difficult to customize and the performance meets bottleneck (Yang et al., 2023; Koh et al., 2024). In this paper, we explore training PRMs to guide the exploration of LMs, decoupling it from the optimization of the policy model, and the resulted PRMs can also be used as verifiers for re-ranking and search.

PRMs for LLMs. PRMs can provide dense reward signals to help LLMs in RL and test-time search or re-ranking (Snell et al., 2024), and are widely used in LLM reasoning (Lightman et al., 2024; Wang et al., 2024c; Yu et al., 2024; Li and Li, 2024). However, the data labeling required for this approach is expensive and not scalable (Lightman et al., 2024). Therefore, recent work has explored automated annotating methods based on Monte Carlo sampling to reduce the cost (Wang et al., 2024c; Li and Li, 2024). In agent tasks, some works have also used similar MC sampling methods to label the Q-values of actions (Hao et al., 2023; Lin et al., 2024; Zhai et al., 2024). However, they only consider the future success probability of a step, without accounting for the dependencies and progress between steps (Yao et al., 2022; Xi et al., 2023, 2024; Chevalier-Boisvert et al., 2019). Our AgentPRM captures both of these aspects and we perform data labeling more efficiently by using the method of TD-estimation with GAE.

More detailed discussion of related work can be found in Appendix E.

7 Conclusion

In this paper, we present AgentPRM, a process supervision model designed for language model agents. It captures both the probability of each step achieving the goal (promise) and the interdependence between sequential steps (progress). Extensive experiments and analysis demonstrate that our method outperforms other baselines across various sampling strategies, models, and tasks. Additionally, our approach is more compute-efficient, and its performance shows stable and robust improvement as inference compute increases, highlighting its potential for training stronger policy models in the future (e.g., via reinforcement learning). Moreover, our method generalizes well to mathematical tasks, showcasing its versatility. We hope our work provides valuable insights for the language model agent field.

Limitations

In this paper, we introduce AgentPRM to guide language models' search and perform trajectory reranking for agent tasks, enhancing the performance. However, our work still has some limitations: (1) Our approach primarily focuses on performance improvement and does not address safety concerns. Safety in language models and agents has become a critical issue (Bai et al., 2022; Zhiheng et al., 2023; Xi et al., 2023, 2024; Zheng et al., 2024; Yang et al., 2024), especially for digital agents capable of manipulating web pages or embodied agents operating in the real world. Therefore, future work should consider reward models aligned with safety for agents and integrate them with our approach to ensure development within safe boundaries. (2) Our experiments were mainly conducted on a series of models discussed earlier, and in the future, we expect to include a broader range of model types to further demonstrate the generalizability of AgentPRM.

References

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. 2023. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosiute, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemí Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional AI: harmlessness from AI feedback. *CoRR*, abs/2212.08073.

Hritik Bansal, Arian Hosseini, Rishabh Agarwal, Vinh Q. Tran, and Mehran Kazemi. 2024. Smaller, weaker, yet better: Training LLM reasoners via compute-optimal sampling. *CoRR*, abs/2408.16737.

EN Barron and H Ishii. 1989. The bellman equation for minimizing the maximum cost. *NONLINEAR ANAL. THEORY METHODS APPLIC.*, 13(9):1067–1090.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning. *CoRR*, abs/2310.05915.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024a. Alphamath almost zero: Process supervision without process. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. 2024b. Agent-flan: Designing data and methods of effective agent tuning for large language models. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 9354–9366. Association for Computational Linguistics.

Zhenfang Chen, Delin Chen, Rui Sun, Wenjun Liu, and Chuang Gan. 2025. Inference-time scaling of autonomous agents from automatic reward modeling and planning.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. Babyai: A platform to study the sample efficiency of grounded language learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 8154–8173. Association for Computational Linguistics.

Matthew J. Hausknecht and Peter Stone. 2015. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposia, Arlington, Virginia, USA, November 12-14, 2015*, pages 29–37. AAAI Press.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024. Tree search for language model agents. *CoRR*, abs/2407.01476.

Wendi Li and Yixuan Li. 2024. Process reward model with q-value rankings. *CoRR*, abs/2410.11287.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zongyu Lin, Yao Tang, Da Yin, Stuart X. Yao, Ziniu Hu, Yizhou Sun, and Kai-Wei Chang. 2024. Q* agent: Optimizing language agents with q-guided exploration.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024a. Skywork-reward: Bag of tricks for reward modeling in llms. *CoRR*, abs/2410.18451.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2024b. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Le Xue, Shelby Heinecke, Rithesh Murthy, Yihao Feng, Zeyuan Chen, Juan Carlos Niebles, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. 2023. BOLAA: benchmarking and orchestrating llm-augmented autonomous agents. *CoRR*, abs/2308.05960.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language models by automated process supervision. *CoRR*, abs/2406.06592.

OpenAI. 2024. Hello gpt-4o.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. 2024. Adapt: As-needed decomposition and planning with language models. In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 4226–4252. Association for Computational Linguistics.

QwenTeam. 2024. Qwen2.5: A party of foundation models.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2016. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024. Rewarding progress: Scaling automated process verifiers for LLM reasoning. *CoRR*, abs/2410.08146.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314.

Richard S Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

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

Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin, Enyu Zhou, Chenyu Shi, Songyang Gao, Nuo Xu, Yuhao Zhou, Xiaoran Fan, Zhiheng Xi, Jun Zhao, Xiao Wang, Tao Ji, Hang Yan, Lixing Shen, Zhan Chen, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. 2024a. Secrets of RLHF in large language models part II: reward modeling. *CoRR*, abs/2401.06080.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024b. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.*, 18(6):186345.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024c. 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), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 9426–9439. Association for Computational Linguistics.

Yu Wang, Nan Yang, Liang Wang, and Furu Wei. 2025. Examining false positives under inference scaling for mathematical reasoning. *Preprint*, arXiv:2502.06217.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. 2023. The rise and potential of large language model based agents: A survey. *CoRR*, abs/2309.07864.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. 2024. Agentgym: Evolving large language model-based agents across diverse environments. *CoRR*, abs/2406.04151.

Hui Yang, Sifu Yue, and Yunzhong He. 2023. Auto-gpt for online decision making: Benchmarks and additional opinions. *CoRR*, abs/2306.02224.

Zonghan Yang, An Liu, Zijun Liu, Kaiming Liu, Fangzhou Xiong, Yile Wang, Zeyuan Yang, Qingyuan Hu, Xinrui Chen, Zhenhe Zhang, Fuwen Luo, Zhicheng Guo, Peng Li, and Yang Liu. 2024. Position: Towards unified alignment between agents, humans, and environment. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Fei Yu, Anningzhe Gao, and Benyou Wang. 2024. Ovm, outcome-supervised value models for planning in mathematical reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 858–875. Association for Computational Linguistics.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024. Agenttuning: Enabling generalized agent abilities for llms. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 3053–3077. Association for Computational Linguistics.

Yuanzhao Zhai, Tingkai Yang, Kele Xu, Dawei Feng, Cheng Yang, Bo Ding, and Huaimin Wang. 2024. Enhancing decision-making for LLM agents via step-level q-value models. *CoRR*, abs/2409.09345.

Dan Zhang, Sining Zhoubian, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Rest-mcts*: LLM self-training via process reward guided tree search. *CoRR*, abs/2406.03816.

Rui Zheng, Wei Shen, Yuan Hua, Wenbin Lai, Shihan Dou, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Haoran Huang, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Improving generalization of alignment with human preferences through group invariant learning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Xi Zhiheng, Zheng Rui, and Gui Tao. 2023. Safety and ethical concerns of large language models. In *Proceedings of the 22nd Chinese National Conference on Computational Linguistics (Volume 4: Tutorial Abstracts)*, pages 9–16, Harbin, China. Chinese Information Processing Society of China.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

A Algorithms

We summarize the training algorithm of AgentPRM in Algorithm 1. We list the process of beam search in Algorithm 2.

B Details of Agent Tasks

We conducted experiments on three agent tasks: WebShop (Yao et al., 2022), BabyAI (Chevalier-Boisvert et al., 2019), and TextCraft (Prasad et al., 2024).

WebShop WebShop (Yao et al., 2022) is a simulated e-commerce website environment with 1.18 million real-world products. In this environment, an agent needs to navigate multiple types of webpages and perform diverse actions to find, customize, and purchase a product given an instruction. We set the max interaction rounds to 6.

BabyAI The BabyAI platform (Chevalier-Boisvert et al., 2019) comprises an extensible suite of 19 levels of increasing difficulty. The levels gradually lead the agent towards acquiring a combinatorially rich synthetic language which is a proper subset of English. The environment is populated with entities of different colors, such as the agent, balls, boxes, doors and keys. Objects can be picked up, dropped and moved around by the agent. Doors can be unlocked with keys matching their color. At each step, the agent receives a 7×7 representation of its field of view (the grid cells in front of it) as well as a Baby Language instruction (textual string). We set the max interaction rounds to 20.

TextCraft The TextCraft task (Prasad et al., 2024) is designed to test the ability of agents to plan and execute complex tasks that require crafting items from available resources. The dataset features a natural compositional structure, with tasks that involve a series of steps of varying complexity. The agent needs to identify and adapt to the varying task complexity. The dataset includes a variety of atomic skills, such as crafting and fetching items, and uses Minecraft's crafting recipes to specify craftable items and their ingredients. The agent's objective is to obtain target Minecraft items by crafting them from available items in the environment. We set the max interaction rounds to 20.

C More Implementation Details

We set the temperature to 1.0 in trajectory collection to maintain diversity in training data. For BoN and beam search, we set the temperature to 0.7. Following AgentGym (Xi et al., 2024), we include 100, 90, 97 queries for evaluation on WebShop (Yao et al., 2022), BabyAI (Chevalier-Boisvert et al., 2019), TextCraft (Prasad et al., 2024), respectively. For GSM8K, we include 1319 evaluation queries as in Cobbe et al. (2021). For math tasks, we initialize the policy model with 5863 trajectories from Math-Shepherd (Wang et al., 2024c).

D Qualitative Analysis

We perform a qualitative analysis to show how AgentPRM works. We provide examples in Figure 7 and Figure 8.

The example shown in Figure 7 demonstrates Best-of-N selection on 4 trajectories by ORM and AgentPRM. ORM fails to select the correct trajectory and shows little difference in the score estimates across the different trajectories. In contrast, AgentPRM successfully identifies the correct trajectory, assigning low scores to negative trajectories and high scores to positive ones.

The second case shown in Figure 8 compares the process of beam search guided by AgentPRM and the baseline PVM. The policy model successfully solves this task under the guidance of AgentPRM, while it fails with PVM. We can also find that AgentPRM effectively distinguishes between good and bad actions (assigning high scores to good actions and low scores to bad ones), whereas PVM does not, with less clear differentiation in scoring between different actions.

E More Detailed Discussion of Related Work

We list the comparison of our method and other related methods in Table 4.

In the LLM agent domain, ARMAP constructs outcome reward models (ORMs) through data labeling to re-rank trajectories, providing better BoN performance (Chen et al., 2025). Q* AGENT uses the Bellman equation (Barron and Ishii, 1989) to estimate the Q-value of each step to train process reward models (Lin et al., 2024). DPO-Q (Zhai et al., 2024) uses a MCTS-based method for building a planning tree and use DPO (Rafailov et al., 2023) to estimate the value of each step. However,

Algorithm 1: Training of AgentPRM.

Input: Initialized AgentPRM model $\mathcal{M}_\phi$; Reward function r; Sample number per query N_{TD}; Agent task query set $\{s_0^i\}_{i=1}^{N_{\text{Task}}}$; Actor π_θ; Number of training iterations m.

Procedure Trajectories collection
- $\mathcal{D}_{train} \leftarrow [\ ]$ ▷ Initialize AgentPRM Train set $\mathcal{D}_{train}$
- **for** s_0^i in $\{s_0^i\}_{i=1}^{N_{\text{Task}}}$ **do**
  - **for** $n = 1$ to N_{TD} **do**
    - $\tau \leftarrow \pi_\theta(s_0^i)$;
    - Add τ to $\mathcal{D}_{train}$;
  - **end**
- **end**

Procedure AgentPRM model training
- **for** $n = 1$ to m **do**
  - **for** batch in $\mathcal{D}_{train}$ **do**
    - **for** trajectory τ in batch **do**
      - $\mathcal{Q} \leftarrow [\ ]$; ▷ AgentPRM model estimated value list $\mathcal{Q}$
      - **for** (s_t, a_t) in τ **do**
        - $Q_t \leftarrow \mathcal{M}_\phi(s_t, a_t)$
        - Add Q_t to $\mathcal{Q}$;
      - **end**
      - $\hat{A} \leftarrow GAE(\mathcal{Q}, r(\tau))$;
      - $\hat{Q} \leftarrow TD(\mathcal{A}, \mathcal{Q})$
      - $\mathcal{L}_Q = \mathbb{E}\left[\frac{1}{2}(\mathcal{Q}_n - \hat{Q}_n)^2\right]$
      - $\mathcal{L}_A = \mathbb{E}\left[\frac{1}{2}((\mathcal{Q}_n - \mathcal{Q}_{n-1}) - (\hat{Q}_n - \hat{Q}_{n-1}))^2\right]$
      - $\mathcal{M}_\phi \leftarrow$ Back_Propagation$(\mathcal{L}_Q + \beta\mathcal{L}_A)$
    - **end**
  - **end**
- **end**

they only consider the promise of each step, without accounting for the dependencies and progress between actions. In contrast, our approach uses TD-based estimation with GAE to estimate the value at different steps, capturing the dependencies between actions.

In the LLM reasoning domain, PQM also considers the relationships between different steps, but unlike us, they use MC-based estimation and introduce a ranking loss to optimize the model (Li and Li, 2024). PAV, on the other hand, estimates the reward of the entire trajectory through ORM and incorporates the advantages of individual steps to assist RL and search (Setlur et al., 2024).

Algorithm 2: Beam search with PRM.

Input: Trained PRM $\mathcal{M}_\phi$; Policy π_θ; Number of actions expanded at each node M; Size of beam search N; Max steps T

Procedure Step-level beam search with PRM

```
    C = [s_0] * M, t = 0                                   ▷ Initialize candidates
    while t < T and non-terminal path in C do
        C_{t+1} ← [ ]                                      ▷ Initialize priority queue
        for s_t in C do
            Sample {a^(b)}_{b=1}^{B_2} ~ π_θ(s_t)
            for b = 1 to M do
                s_{t+1} = Concat [s_t, a^(b)]
                Add (s_{t+1}, M_φ(s_{t+1})) to C_{t+1}
            end
            C ← Top-N of C_{t+1}
        end
    end
    return Top-1 of C                                      ▷ Return top-1 as the final solution path
```

Method	Data Labeling	Supervision Gra.	Progress	Task Type
PRM (Lightman et al., 2024)	Human	Process	×	Reasoning
Math-Shepherd (Wang et al., 2024c)	MC-based	Process	×	Reasoning
PAV (Setlur et al., 2024)	MC-based	Process	✓	Reasoning
PQM (Li and Li, 2024)	MC-based	Process	✓	Reasoning
ARMAP (Chen et al., 2025)	MC-based	Outcome	×	Agent
Q* Agent (Lin et al., 2024)	TD-based	Process	×	Agent
DPO-Q (Zhai et al., 2024)	MC-based	Process	×	Agent
AgentPRM(Ours)	TD-based	Process	✓	Agent, Reasoning

Table 4: Comparison of different process supervision paradigms. "Supervision Gra." means Supervision Granularity.

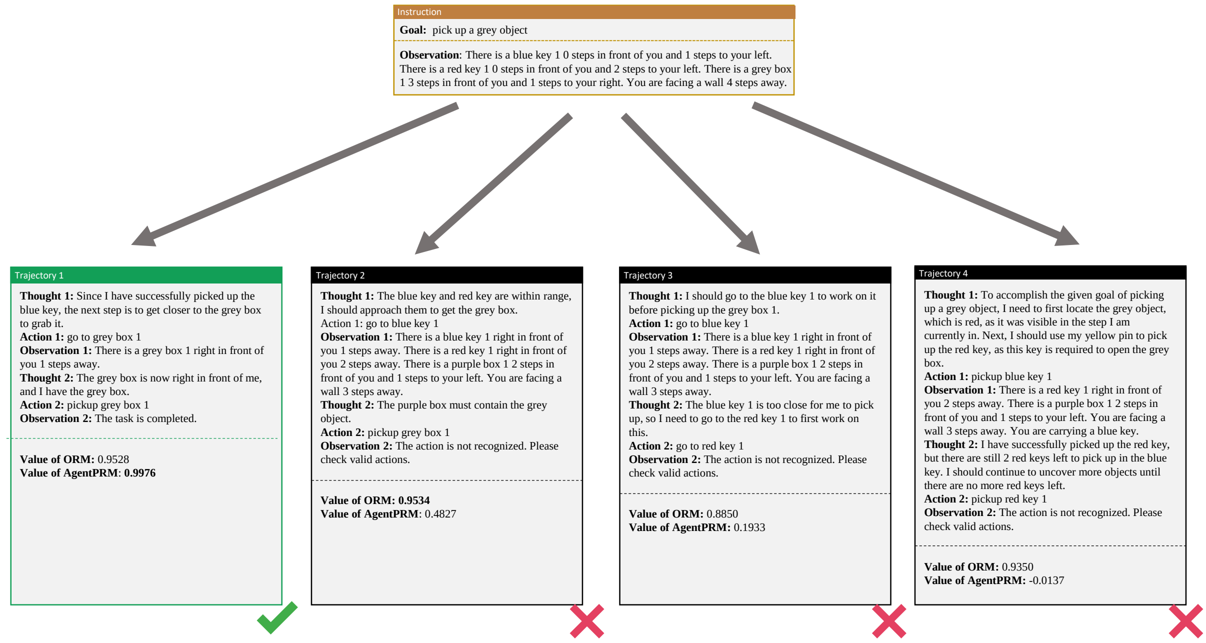


Figure 7: Example of qualitative analysis on BabyAI. ORM fails to select the correct trajectory and shows little difference in the score estimates across the different trajectories. In contrast, AgentPRM successfully identifies the correct trajectory, assigning low scores to negative trajectories and high scores to positive ones.

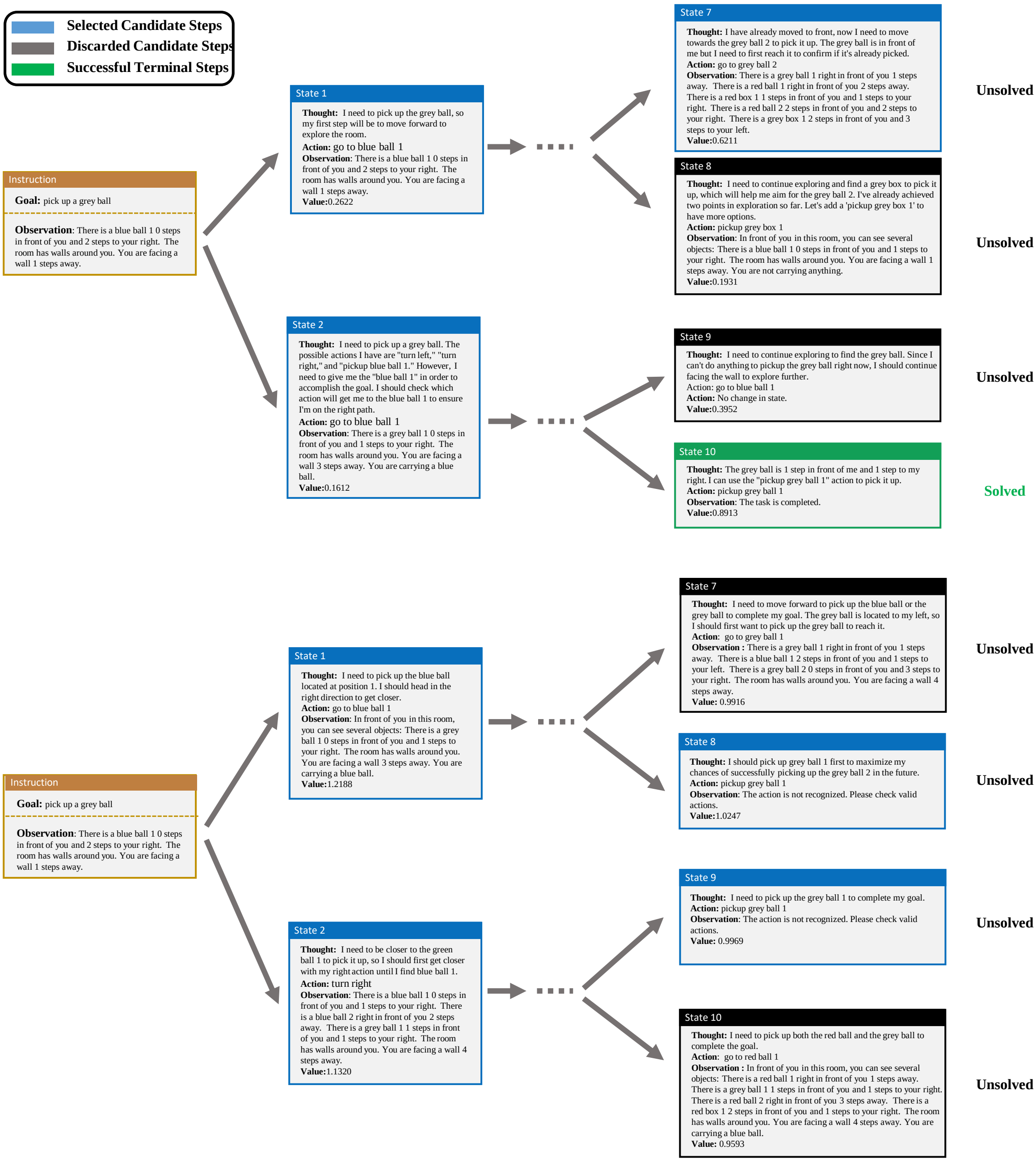


Figure 8: Example of qualitative analysis on beam search. The upper part of this figure demonstrates a successful solution with beam search guided by AgentPRM, and the lower part demonstrates an unsuccessful one guided by PVM. The policy model solves this task in 10 steps under the guidance of AgentPRM, while it fails with PVM. We can also find that AgentPRM effectively distinguishes between good and bad actions (assigning high scores to good actions and low scores to bad ones), whereas PVM does not, with less clear differentiation in scoring between different actions.