



AgentPRM: Process Reward Models for LLM Agents via Step-Wise Promise and Progress

Zhiheng Xi^{1*†}, Chenyang Liao^{1*}, Guanyu Li¹, Yajie Yang¹, Wenxiang Chen¹,
 Zhihao Zhang¹, Binghai Wang¹, Senjie Jin¹, Yuhao Zhou¹, Jian Guan²,
 Wei Wu², Tao Ji¹, Tao Gui^{1†}, Qi Zhang^{1†}, Xuanjing Huang^{1†}

¹Fudan University ²Ant Group

zhxi22@m.fudan.edu.cn, {tgui, xjhuang}@fudan.edu.cn

Despite rapid development, large language models (LLMs) still encounter challenges in multi-turn decision-making tasks (i.e., agent tasks) like web shopping and browser navigation, which require making a sequence of intelligent decisions based on environmental feedback. Previous work for LLM agents typically relies on elaborate prompt engineering or fine-tuning with expert trajectories to improve performance. In this work, we take a different perspective: we explore constructing process reward models (PRMs) to evaluate each decision and guide the agent’s decision-making process. 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 propose a re-defined PRM for agent tasks, named AgentPRM, to capture both the interdependence between sequential decisions and their contribution to the final goal. This enables better progress tracking and exploration-exploitation balance. To scalably obtain labeled data for training AgentPRM, we employ a Temporal Difference-based (TD-based) estimation method combined with Generalized Advantage Estimation (GAE), which proves more sample-efficient than prior methods. Extensive experiments across different agentic tasks show that AgentPRM is over 8× more compute-efficient than baselines, and it demonstrates robust improvement when scaling up test-time compute. Moreover, we perform detailed analyses to show how our method works and offer more insights, e.g., applying AgentPRM to the reinforcement learning of LLM agents.

1. Introduction

The advent of large language models (LLMs) has resulted in significant advances in a variety of natural language processing tasks, including text generation (Dathathri et al., 2020; Keskar et al., 2019; Li et al., 2024; Zhang et al., 2023b), summarization (Tang et al., 2023; Van Veen et al., 2024; Zhang et al., 2024b, 2025a), translation (He et al., 2024; Moslem et al., 2023; Wang et al., 2023; Xu et al., 2024; Zhang et al., 2023a), and reasoning (Anil et al., 2023; OpenAI, 2024a; QwenTeam, 2024). Despite these advancements, LLMs still encounter considerable challenges in multi-turn decision-making tasks (i.e., agent tasks) such as web shopping (Yao et al., 2022; Zhang et al., 2025b), browser navigation (Deng et al., 2023; Koh et al., 2024a; Xu et al., 2021; Zhou et al., 2024), and digital games (Chevalier-Boisvert et al., 2019; Hu et al., 2025; Park et al., 2025; Prasad et al., 2024; Wang et al., 2025a), where models must make a series of intelligent decisions based on feedback from the environment (Xi et al., 2023; Zeng et al., 2024). These models are referred to as LLM agents (Xi et al., 2023).

*Equal contribution. †Corresponding authors.

The agent tasks are inherently dynamic and context-sensitive, setting them apart from traditional static tasks (Chen et al., 2024b; Liu et al., 2024b; Xi et al., 2024b). Achieving effective performance in these tasks requires models not only to comprehend task-related knowledge and interpret environmental cues, but also to engage in forward-looking planning to anticipate future consequences of their decisions (Wang et al., 2024b; Xi et al., 2023).

Prior work has sought to enhance LLMs for agent tasks using approaches such as supervised fine-tuning (Chen et al., 2024b; Zeng et al., 2024) and prompt engineering (Liu et al., 2023; Shinn et al., 2023; Yao et al., 2023). Supervised fine-tuning methods rely on expert-labeled data, which are scarce and hard to collect, limiting scalability. Prompt engineering typically leverages commercial models like GPT-4o (OpenAI, 2024a) to achieve satisfactory performance, but is hindered by API constraints, making it both costly and inflexible for customization (Koh et al., 2024b; Yang et al., 2023). Another promising direction involves self-improvement that trains models by leveraging successful trajectories they explored (Aksitov et al., 2023; Song et al., 2024; Tao et al., 2024; Xi et al., 2024b; Yang et al., 2024). However, it relies on outcome-based feedback, which does not offer sufficient insight into the value of individual decisions made by the model and, in turn, leads to a performance bottleneck (Ding et al., 2025).

To this end, we draw inspiration from process supervision in LLM reasoning and explore the use of process reward models (PRMs) to guide the search and exploration of LLM agents (Li and Li, 2024; Lightman et al., 2024; Setlur et al., 2024; Uesato et al., 2022; Wang et al., 2024c). While PRMs have proven effective in reasoning tasks to evaluate individual steps and guide the decoding process of LLMs, they face unique challenges in agent tasks:

1. Different from LLM reasoning, actions in agent tasks lack a clear-cut "correctness" metric, making the evaluation non-trivial (Yao et al., 2022; Zhou et al., 2024).
2. Existing process supervision methods treat each step independently, overlooking the sequential dependencies between decisions within a trajectory, which is inconsistent with the inherently sequential nature of agent tasks (Li and Li, 2024).
3. Previous methods for training PRMs often depend on either expert annotations or extensive Monte Carlo-based (MC-based) sampling for estimation, both of which are costly in real-world scenarios (Luo et al., 2024; Wang et al., 2024c).

In this work, we propose AgentPRM to address the challenges (Figure 1 and Figure 2). Our core insight is to *evaluate the proximity of each step to the goal state and tracks the progress made by LLMs*. Specifically, AgentPRM predicts the contribution of each decision to the final goal and captures the interdependencies between sequential decisions, thereby enabling more effective progress tracking

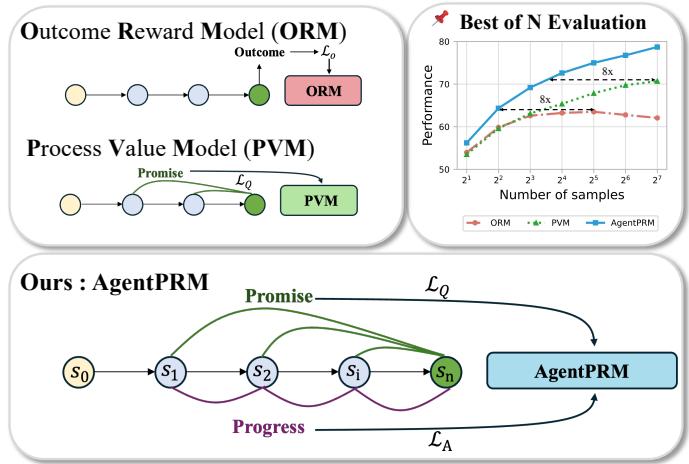


Figure 1 | Comparison of AgentPRM and baselines, and the Best-of-N results. Upper Left: Baseline reward models. ORMs focus on the final outcome reward; PVMs focus on promise of each step only. Bottom: Our AgentPRM that captures both the promise and progress of each step. Upper Right: Average Best-of-N performance of three agent tasks. AgentPRM outperforms other baselines, and it demonstrates a more stable and robust improvement trend as inference compute scaling.

making it both costly and inflexible for customization (Koh et al., 2024b; Yang et al., 2023). Another promising direction involves self-improvement that trains models by leveraging successful trajectories they explored (Aksitov et al., 2023; Song et al., 2024; Tao et al., 2024; Xi et al., 2024b; Yang et al., 2024). However, it relies on outcome-based feedback, which does not offer sufficient insight into the value of individual decisions made by the model and, in turn, leads to a performance bottleneck (Ding et al., 2025).

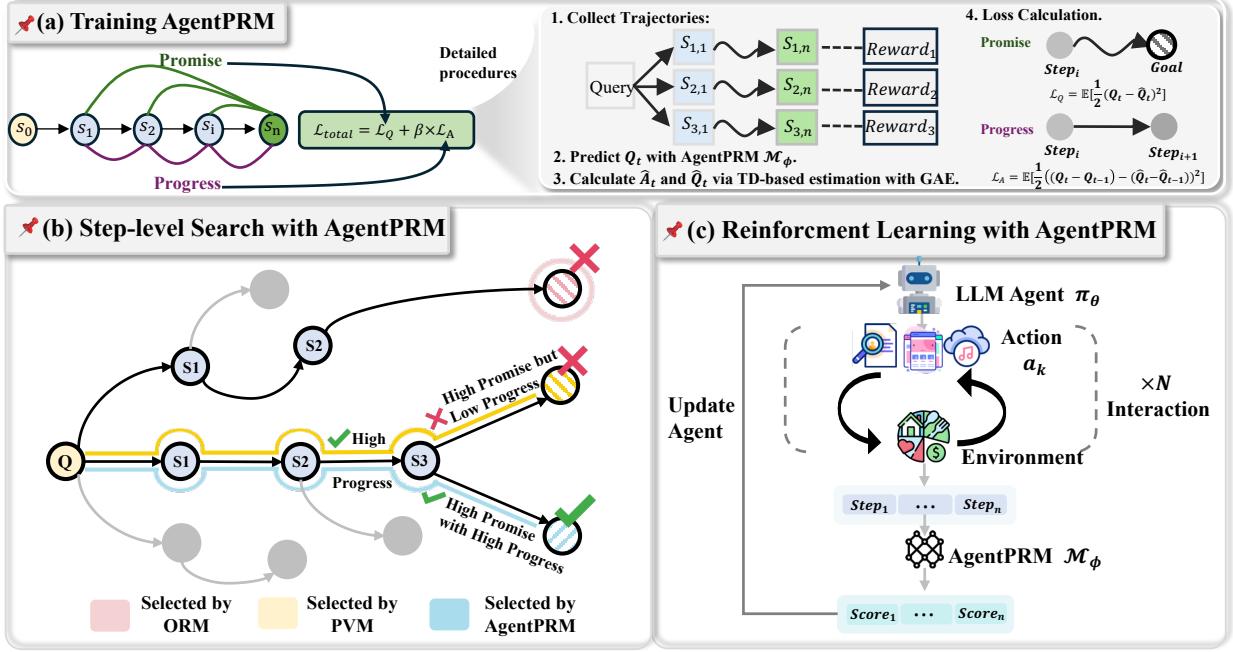


Figure 2 | Overview of the training and the application of AgentPRM. (a): The training objective and the detailed training procedures of AgentPRM. We take into account both the promise (probability of each step achieving the goal) and the progress (the interdependence between sequential steps). (b): With AgentPRM, we perform step-level beam search to guide the LLM agent toward the goal. (c): AgentPRM can be integrated into the reinforcement learning process of LLM agents seamlessly.

and optimizing the balance between exploration and exploitation. To scale the training data acquisition efficiently, we employ an automated Temporal Difference-based (TD-based) method with Generalized Advantage Estimation (GAE) (Schulman et al., 2016), which is more efficient than previous MC-based methods (Wang et al., 2024c), and provides a better trade-off between variance and bias in estimation (Schulman et al., 2016).

Extensive experiments across various models and tasks show that AgentPRM consistently outperforms baselines in both performance and compute efficiency. For instance, with Qwen2.5-3B, AgentPRM achieves over 8× greater compute efficiency compared to baselines across three agent tasks and multiple sampling strategies. Additionally, it demonstrates a more stable and robust improvement trend as inference compute scales. Further analyses, including its application to reinforcement learning (§5.2) and a comparison of sampling efficiency with baselines (§5.4), are also provided to offer more insights.

To summarize, this work makes the following key contributions:

- Drawing inspiration from the nature of agentic tasks, we propose AgentPRM, a novel process reward model for LLM agents that simultaneously captures both the immediate progress and the long-term promise of each decision.
- We propose an automated, scalable method, i.e., TD-based estimation with GAE, for training AgentPRM, which is much more efficient than traditional MC-based methods.
- Through extensive experiments across diverse agentic tasks, we demonstrate that AgentPRM achieves over 8× more compute-efficient than baselines, and it demonstrates robust improvement when scaling up test-time compute. We also perform in-depth ablation and analyses to show how it works and provide more insights.

2. Preliminary and Background

An 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., 2024b), 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 selects an action $a_0 \sim \pi_\theta(\cdot | s_0) \in \mathcal{A}$ under a policy π_θ parameterized by θ . The environment then returns an observation $o_1 \in \mathcal{O}$, yielding the next state $s_1 = \{u, s_0, a_0, o_1\}$ via \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: $\tau = (u, o_0, a_0, o_1, \dots, o_T, a_T)$. Then for a language model, the agent task can be formalized as:

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

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 predicts whether it satisfies the task instruction u (Ouyang et al., 2022; Uesato et al., 2022). ORMs are trained on data sampled from π_θ , where each instruction–trajectory pair is labeled by the corresponding outcome reward $r(u, \tau)$ (Cobbe et al., 2021; Liu et al., 2024a; Ouyang et al., 2022).

Process Reward Model (PRM) A PRM evaluates actions or intermediate states along a trajectory (Lightman et al., 2024; Wang et al., 2024c; Zhang et al., 2024a). In the field of LLM reasoning, the scoring criterion typically involves the correctness of individual steps. However, this is not suitable for the agent tasks we are studying, which will be elaborated in §3 and we will provide appropriate evaluation criteria. PRMs are trained using step-level annotations that assign labels to intermediate actions, and models are optimized to predict these labels (Lightman et al., 2024; Luo et al., 2024; Wang et al., 2024c).

Best-of-N (BoN) with reward models Given a larger inference budget, Best-of-N (BoN) can be applied to improve performance (Touvron et al., 2023). Specifically, the policy π_θ samples N trajectories $\{\tau_i\}_{i=1}^N$, which are then evaluated by a reward model. The highest-scoring trajectory is selected as the final output. Note that BoN can also be executed with PRMs (Lightman et al., 2024). In our setting, the score of the final step is used to represent trajectory quality.

Search with process reward models During the inference phase, we can conduct step-level search against PRMs for agent tasks (Figure 2(b)). Among the various step-level search algorithms, beam search is a widely used due to its balance between performance and efficiency (Chen et al., 2024a; Zhang et al., 2024a). In each iteration, beam search expands M candidate actions per node, scores them with a PRM, and retains the top N candidates. The trajectory ending in the highest-scoring terminal state is returned as the final solution. The algorithm of beam search is summarized in Algorithm 2 of Appendix B.

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, three key challenges arise:

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, without considering the dependencies between consecutive decisions (Li and Li, 2024; Setlur et al., 2024). However, in agent tasks, the decisions at each step are interconnected, forming a chain of dependencies, where each decision influences subsequent decisions and ultimately the outcome (Chevalier-Boisvert et al., 2019; Xi et al., 2023, 2024b; Yao et al., 2022).
3. Previous methods for training PRMs often require expert annotations or a large amount of Monte Carlo-based (MC-based) sampling for estimation, both of which are costly (Lightman et al., 2024; Luo et al., 2024; Wang et al., 2024c). Moreover, MC-based estimation may lead to high variance (Sutton and Barto, 2018).

Given these challenges, our work focuses on two critical research questions: **RQ1: how to define appropriate rewards for decisions** and **RQ2: how to efficiently and reliably train process reward models**.

3.2. AgentPRM: Re-Defining Process Rewards for LLM Agents

To answer **RQ1**, we argue that *a good process reward model for agent tasks must consider both the probability that each step advances toward the goal (promise) and the interdependence among sequential steps (progress)*. Based on this insight, we propose the re-defined PRM for LLM agents in this section.

3.2.1. Measuring expected future success probability with value functions.

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

$$Q^\pi(s_t, a_t) = \mathbb{E}_{\tau \sim \pi(\cdot | s_t, a_t)} [r(u, \tau)]. \quad (2)$$

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

$$V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi(\cdot | s_t)} [Q^\pi(s_t, a_t)]. \quad (3)$$

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

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{s_t, a_t \sim \mathcal{D}_Q} \left[\frac{1}{2} (\mathcal{M}_\phi(s_t, a_t) - \hat{Q}(s_t, a_t))^2 \right]. \quad (4)$$

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

3.2.2. Capturing dependencies between steps with advantages.

Nevertheless, the aforementioned \mathcal{M}_ϕ only considers 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 excessive exploitation without a sufficient balance of 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 argue that process rewards should not only measure the promise of success, but also capture the local progress between actions. This perspective aligns with RL (Sutton et al., 1999), where advantage functions quantify the relative improvement in success likelihood resulting from an action:

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t). \quad (5)$$

The value of $A^\pi(s_t, a_t)$ can be either positive or negative. A positive advantage indicates that the current action contributes to progress, whereas a negative advantage suggests the opposite.

Hence, to train our model \mathcal{M}_ϕ to account for both progress and dependencies between actions, we introduce a distinct loss term to fit the advantage:

$$\mathcal{L}_A(\phi) = \mathbb{E}_{s_t, a_t \sim D_Q} \left[(A_\phi(s_t, a_t) - \hat{A}(s_t, a_t))^2 \right], \quad (6)$$

where $\hat{A}(s_t, a_t)$ represents the annotated advantage labels. Next, we show how to integrate the fitting optimization of the advantage into the training of our PRM. For the agent tasks we attempt to solve, they have a sparse reward, which means for any time step $t < T$, $r_t = 0$ where T is the final time step. As the state transition in our setting is deterministic, following previous works (Li and Li, 2024; Setlur et al., 2024), we have:

$$Q(s_t, a_t) - V(s_t) = Q(s_t, a_t) - (r_t + V(s_t)) \quad (7)$$

$$= Q(s_t, a_t) - Q(s_{t-1}, a_{t-1}), \quad (8)$$

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

$$\mathbb{E}_{s_t, a_t \sim D_Q} \left[\left((\mathcal{M}_\phi(s_t, a_t) - \mathcal{M}_\phi(s_{t-1}, a_{t-1})) - (\hat{Q}(s_t, a_t) - \hat{Q}(s_{t-1}, a_{t-1})) \right)^2 \right]. \quad (9)$$

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:

$$\mathcal{L}_{\text{AgentPRM}}(\phi) = \mathcal{L}_Q(\phi) + \beta \times \mathcal{L}_A(\phi), \quad (10)$$

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

3.3. Practical Implementation for Training AgentPRM

In the previous section, we demonstrate how to optimize our AgentPRM based on the estimated or annotated data. Here, we explore how to estimate the action-value function $Q(s_t, a_t)$ in an effective and scalable manner (**RQ2**). We compare the previous Monte Carlo (MC) estimation, and our

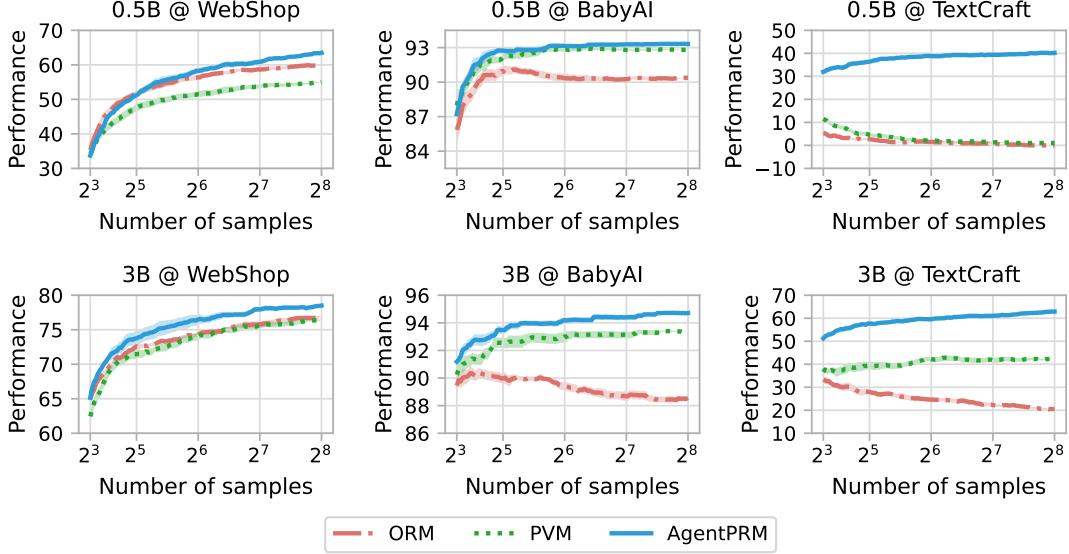


Figure 3 | Performance of Best-of-N evaluation. AgentPRM outperforms other baselines, is more compute-efficient, and demonstrates a more stable and robust improvement trend as inference compute scales.

proposed Temporal Difference-based (TD-based) estimation with Generalized Advantage Estimation (GAE).

3.3.1. MC-based estimation

A common method for automatically estimating the Q-value of an action is based on Monte Carlo (MC) sampling (Luo et al., 2024; Wang et al., 2024c). 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 1:

$$\hat{Q}(s_t, a_t) = \begin{cases} 1 & \exists \tau'_j, \tau'_j \text{ is successful}, \\ 0 & \text{otherwise,} \end{cases} \quad (11)$$

While effective, MC-based estimation is resource-intensive and suffers from high computational cost due to the large number of rollouts required. As such, we explore alternative, more efficient methods.

3.3.2. TD-based estimation with GAE

We draw inspiration from previous work (Ouyang et al., 2022; Schulman et al., 2016; Sutton, 1988; Sutton et al., 1999) and introduce TD-based methods, using GAE to reduce variance and improve stability (Schulman et al., 2016). First, we define the TD residual for an action as follows:

$$\begin{aligned} \delta(s_t, a_t) &= r_t + \gamma Q(s_t, a_t) - Q(s_{t-1}, a_{t-1}) \\ &= r_t + \gamma \mathcal{M}_\phi(s_t, a_t) - \mathcal{M}_\phi(s_{t-1}, a_{t-1}), \end{aligned} \quad (12)$$

where r_t is the instant reward at timestep t , and in our setting, sparse rewards are only assigned when $t = T$. Next, given a trajectory $\tau = (u, o_0, a_0, o_1, \dots, o_T, a_T)$, we estimate the advantage for

Table 1 | Evaluation results of training-based methods and reward-model-based beam search on agent tasks. For RM-based results, @N×M denotes the number of nodes retained and expanded during beam search. The best performance in each column is highlighted in bold.

Model	Method	WebShop			BabyAI			TextCraft		
		@2×2	@4×4	@8×8	@2×2	@4×4	@8×8	@2×2	@4×4	@8×8
Qwen2.5-0.5B	<i>Training-based</i>									
	SFT	–	30.5	–	–	37.6	–	–	27.8	–
	RFT	–	39.0	–	–	54.4	–	–	32.9	–
	<i>Reward-Model-based</i>	@2×2	@4×4	@8×8	@2×2	@4×4	@8×8	@2×2	@4×4	@8×8
	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
Qwen2.5-3B	<i>Training-based</i>									
	SFT	–	46.0	–	–	67.4	–	–	29.8	–
	RFT	–	48.0	–	–	64.5	–	–	36.0	–
	<i>Reward-Model-based</i>	@2×2	@4×4	@8×8	@2×2	@4×4	@8×8	@2×2	@4×4	@8×8
	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

different actions using GAE (Schulman et al., 2016):

$$\hat{A}(s_t, a_t) = \sum_{k=0}^{\infty} (\gamma \lambda)^k \delta(s_{t+k}, a_{t+k}), \quad (13)$$

where λ is the discount factor. Finally, the current estimated $\hat{Q}(s_t, a_t)$ can be represented as:

$$\begin{aligned} \hat{Q}(s_t, a_t) &= \hat{A}(s_t, a_t) + \hat{V}(s_t) \\ &= \hat{A}(s_t, a_t) + \hat{Q}(s_{t-1}, a_{t-1}) \\ &= \hat{A}(s_t, a_t) + \mathcal{M}_\phi(s_{t-1}, a_{t-1}), \end{aligned} \quad (14)$$

where $t < T$. For the terminal step $t = T$, the target action-value $\hat{Q}(s_T, a_T)$ is directly defined as the final outcome reward obtained from the environment, i.e., $\hat{Q}(s_T, a_T) = r(u, \tau)$.

In implementation, we sample N_{TD} trajectories $\{\tau_i\}_{i=1}^{N_{TD}}$ from the policy π_θ for training. Since our estimation process involves prediction of \mathcal{M}_ϕ , we iteratively sample a batch from the trajectory set, conduct estimation based on the current model, and update the model with Equation 10. We summarize the training algorithm of AgentPRM in Algorithm 1, which is also illustrated in Figure 2(a).

Our method provide two main benifits: 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 (See §5.4). 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

4.1.1. Tasks.

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

WebShop (Yao et al., 2022)¹ 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.

The BabyAI task (Chevalier-Boisvert et al., 2019)² involves agents navigating a grid world based on natural language instructions. The environment includes various entities such as the agent, balls, boxes, doors, and keys. Agents perform tasks like moving objects, unlocking doors, and interacting with the world according to textual commands. We set the max interaction rounds to 20.

The TextCraft task (Prasad et al., 2024)³ 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.

Our implementation is based on AgentGym framework (Xi et al., 2024b). We also conducted experiments on mathematical reasoning in §5.3.

4.1.2. Baselines.

We compare our AgentPRM with several training-based and reward model-based methods. For training-based approaches, supervised fine-tuning (SFT) uses expert data to fine-tune the base model, while rejection sampling fine-tuning (RFT), or self-improvement, trains the model by leveraging successful trajectories it explored (Trung et al., 2024; Xi et al., 2024a,b; Yuan et al., 2023; Zelikman et al., 2022). For reward model-based methods, we include ORMs (Outcome Reward Models) (Cobbe et al., 2021; Liu et al., 2024a; Ouyang et al., 2022; Uesato et al., 2022; Yu et al., 2024) and PVMs (Process Value Models) (Chen et al., 2025a; Li and Li, 2024; Lightman et al., 2024; Miao et al., 2025; Setlur et al., 2024; Uesato et al., 2022; Wang et al., 2025b; Xia et al., 2025; Xiong 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 §5.4.

4.1.3. Implementation Details.

All experiments in this work are conducted with A100-80GB GPUs. Our backbone models include Qwen-2.5-0.5B-Instruct, Qwen-2.5-3B-Instruct, Qwen-2.5-7B-Instruct (QwenTeam, 2024) and Llama-3.1-8B-Instruct (Dubey et al., 2024). For agent tasks, we use the ReAct format (Yao et al., 2023) where the model first generates reasoning process and then outputs the action. 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 at most 5 epochs under a learning rate of 1×10^{-6} . For AgentPRM, we set $\beta = 1.0$ and $\lambda = 0.95$. We set the temperature to 1.0 in trajectory collection to maintain diversity in training data.

¹<https://github.com/princeton-nlp/WebShop/blob/master/LICENSE.md>

²<https://github.com/mila-iqia/babyai/blob/master/LICENSE>

³<https://github.com/archiki/ADaPT/blob/main/LICENSE>

4.1.4. Evaluation Settings.

Following AgentGym (Xi et al., 2024b), 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. We perform Best-of-N and beam search to evaluate reward models as in (Chen et al., 2025a; Lightman et al., 2024), setting the sampling temperature to 0.7. For SFT and RFT method, we set the temperature to 0.0 (i.e., greedy decoding).

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, the incorporation of reward models for Best-of-N (BoN) and search strategies significantly enhances model performance, especially as inference compute increases for more sampling. This observation aligns with prior work on test-time scaling (Bansal et al., 2024; Snell et al., 2024).

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 ORMs and PVMs across different tasks, demonstrating its effectiveness. Figure 1 (upper right) shows that, on average, AgentPRM is 8 \times more compute-efficient than PVMs and ORMs. This highlights the potential of AgentPRM in training stronger LLM agents 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 \times 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. (2025c), 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 agents. In contrast, AgentPRM consistently shows stable improvement, highlighting its robustness and broadening its potential for future applications.

5. Discussion and Analysis

5.1. 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 AgentPRM. Here, we conduct an ablation on the advantage term to validate its effect. Results in Figure 4 show that without $\mathcal{L}_A(\phi)$, the performance on agent tasks drops regardless of the sampling strategy, showing that capturing progress is important for training AgentPRM.

5.2. Applying AgentPRM to Reinforcement Learning

Reinforcement learning (RL) has become a core method for training LLMs (DeepSeek-AI, 2025; OpenAI, 2024b; Ouyang et al., 2022). To assess the effectiveness of our reward model, we inte-

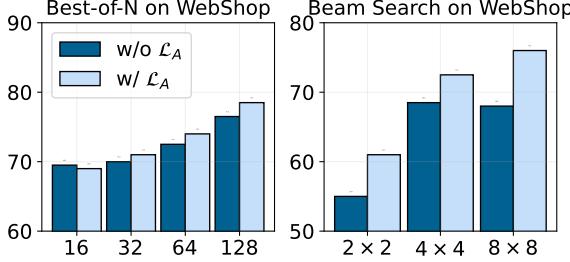


Figure 4 | Ablation study on \mathcal{L}_A with Qwen2.5-3B.

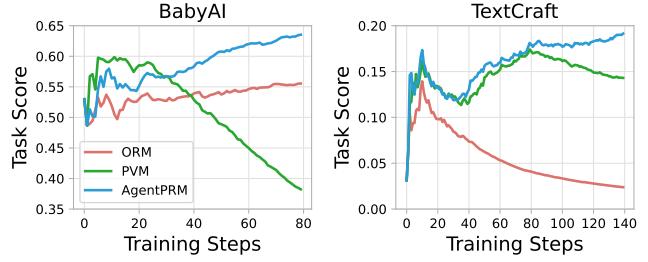


Figure 5 | Task score of RL optimization.

grate it into the RL optimization process. We conduct experiments with Qwen2.5-3B on BabyAI and TextCraft, using Proximal Policy Optimization (PPO) as the algorithm as it is widely adopted. More implementation details are in Appendix C.

As shown in Figure 5, while other baseline reward models face optimization instability or slower performance improvements, AgentPRM delivers a more stable and effective optimization, outperforming the baselines and highlighting its advantages.

5.3. Performance on Mathematical Reasoning

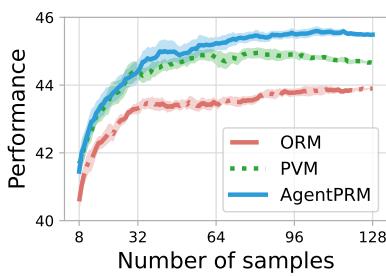


Figure 6 | Best-of-N results on mathematical reasoning tasks.

Table 2 | Evaluation results of beam search on mathematical reasoning tasks.

Model	Method	Math Reasoning		
		@2 x 2	@4 x 4	@8 x 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

To demonstrate the versatility of our method, we also conducted experiments on mathematical reasoning tasks. Following Math-Shepherd (Wang et al., 2024c), we formulate math reasoning as multi-turn decision making: at step t , the partial solution is defined as the state s_t and the next reasoning step is the action a_t . The model emits stepwise text with a delimiter token after each step to segment steps. Since there is no external environmental feedback in this setting, generation simply resumes from the end of the previous step. This yields a well-defined multi-step generation protocol for mathematical reasoning.

In experiments, we employ the GSM8K dataset (Cobbe et al., 2021), with results shown in Figure 6 and Table 2. As we can see, our method still performs exceptionally well on mathematical reasoning 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.

5.4. Comparing Sampling Efficiency of Our Method with MC-based Estimation

In §3.3, we introduced TD-based estimation with GAE for the automated labeling process and stated its benefits. Here, we empirically compare it with the previously commonly used MC-based estima-

Table 3 | Sampling cost and performance of our method and MC-based estimation for PRMs. “Tokens” denotes the amount of sampled tokens used to train PRMs.

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
Math	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

tion. The results are shown in Table 3. We observe that our method requires fewer tokens for labeling the data (i.e., more data-efficient) compared to other methods, yet achieves better performance on Best-of-N and beam search, demonstrating the higher efficiency and effectiveness of our approach.

5.5. Evaluating Value Distributions of Actions with AgentPRM

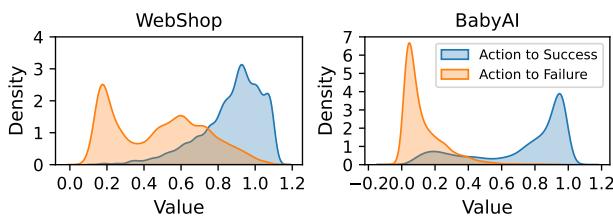


Figure 7 | Visualization of value distribution of Actions with AgentPRM.

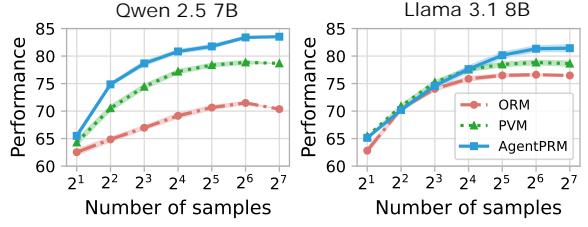


Figure 8 | Average BoN performance on Qwen2.5-7B and Llama3.1-8B across three tasks.

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 7, 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.

5.6. Experiments on Models of Larger Size and Different Series

We also validate the effectiveness of AgentPRM on larger Qwen-2.5-7B-Instruct and the Llama-series (i.e., Llama3.1-8B-Instruct). Figure 8 illustrates the average results across three agentic tasks (i.e., WebShop, BabyAI, and TextCraft). We can find that AgentPRM consistently outperforms the baseline methods on the two models, demonstrating its generalization across model sizes and architectures.

We also perform a 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 training-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. Prompt-engineering-based methods leverage SOTA commercial models like GPT-4o for developing agents, which is limited by APIs, making it difficult to customize (Koh et al., 2024b; Yang et al., 2023). Another line of work adopts self-improvement methods (Aksitov et al., 2023; Song et al., 2024; Tao et al., 2024; Xi et al., 2024b; Yang et al., 2024), allowing the model to explore and learn within the environment. However, these approaches typically rely solely on outcome-based feedback and fail to assess the value and impact of each individual decision (Chen et al., 2025b; Lin et al., 2024). In this paper, we explore training PRMs to guide the exploration of LMs, decoupling it from the optimization of the agent, 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 (Li and Li, 2024; Lightman et al., 2024; Wang et al., 2024c; Yu et al., 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 (Li and Li, 2024; Wang et al., 2024c). 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 (Chevalier-Boisvert et al., 2019; Xi et al., 2023, 2024b; Yao et al., 2022). Our AgentPRM captures both of these aspects and we perform data labeling more efficiently by using the method of TD-estimation with GAE.

See Appendix A for more detailed discussion of related work.

7. Conclusion

In this paper, we introduce AgentPRM, a process supervision model designed for LLM agents in multi-step decision-making tasks. It captures both the probability of each step achieving the goal (promise) and the interdependence between sequential steps (progress). Extensive experiments demonstrate that our method outperforms other baselines across various sampling strategies, models, and tasks. Additionally, it is more compute-efficient, and its performance shows robust improvement as inference compute increases, highlighting its potential for training stronger agents in the future. Moreover, our method generalizes well to mathematical tasks, showcasing its versatility. We also conducted extensive additional analyses and ablation to demonstrate how our method works, its data efficiency, and its adaptability to different model architectures and sizes. We hope our work can provide valuable insights and contributions for the LLM agent community.

References

- Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, Manzil Zaheer, Felix Yu, and Sanjiv Kumar. Rest meets react: Self-improvement for multi-step reasoning llm agent, 2023. URL <https://arxiv.org/abs/2312.10003>.
- 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. Lil-

licrap, 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. Gemini: A family of highly capable multimodal models, 2023. URL <https://doi.org/10.48550/arXiv.2312.11805>.

Hritik Bansal, Arian Hosseini, Rishabh Agarwal, Vinh Q. Tran, and Mehran Kazemi. Smaller, weaker, yet better: Training LLM reasoners via compute-optimal sampling, 2024. URL <https://doi.org/10.48550/arXiv.2408.16737>.

EN Barron and H Ishii. The bellman equation for minimizing the maximum cost., 1989.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning, 2023. URL <https://doi.org/10.48550/arXiv.2310.05915>.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Alphamath almost zero: Process supervision without process, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/30dfe47a3ccbee68cffa0c19ccb1bc00-Abstract-Conference.html.

Wenxiang Chen, Wei He, Zhiheng Xi, Honglin Guo, Boyang Hong, Jiazheng Zhang, Rui Zheng, Nijun Li, Tao Gui, Yun Li, Qi Zhang, and Xuanjing Huang. Better process supervision with bi-directional rewarding signals, 2025a. URL <https://arxiv.org/abs/2503.04618>.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-flan: Designing data and methods of effective agent tuning for large language models, 2024b. URL <https://doi.org/10.18653/v1/2024.findings-acl.557>.

Zhenfang Chen, Delin Chen, Rui Sun, Wenjun Liu, and Chuang Gan. Inference-time scaling of autonomous agents from automatic reward modeling and planning, 1 2025b. URL <https://github.com/heaplax/ARMAP>.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning, 2019. URL <https://openreview.net/forum?id=rJeXCo0cYX>.

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

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation, 2020. URL <https://arxiv.org/abs/1912.02164>.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023. URL <https://arxiv.org/abs/2306.06070>.

Yiwen Ding, Zhiheng Xi, Wei He, Zhuoyuan Li, Yitao Zhai, Xiaowei Shi, Xunliang Cai, Tao Gui, Qi Zhang, and Xuanjing Huang. Mitigating tail narrowing in llm self-improvement via socratic-guided sampling, 2025. URL <https://arxiv.org/abs/2411.00750>.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Bin Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vraneš, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models, 2024. URL <https://doi.org/10.48550/arXiv.2407.21783>.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model, 2023. URL <https://doi.org/10.18653/v1/2023.emnlp-main.507>.

Matthew J. Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps, 2015. URL <http://www.aaai.org/ocs/index.php/FSS/FSS15/paper/view/11673>.

Zhiwei He, Tian Liang, Wenxiang Jiao, Zhuosheng Zhang, Yujiu Yang, Rui Wang, Zhaopeng Tu, Shuming Shi, and Xing Wang. Exploring human-like translation strategy with large language models, 03 2024. ISSN 2307-387X.

Lanxiang Hu, Qiyu Li, Anze Xie, Nan Jiang, Ion Stoica, Haojian Jin, and Hao Zhang. Gamearena: Evaluating llm reasoning through live computer games, 2025. URL <https://arxiv.org/abs/2412.06394>.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation, 2019. URL <https://arxiv.org/abs/1909.05858>.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multi-modal agents on realistic visual web tasks, 2024a. URL <https://arxiv.org/abs/2401.13649>.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents, 2024b. URL <https://doi.org/10.48550/arXiv.2407.01476>.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. Pre-trained language models for text generation: A survey, April 2024. ISSN 0360-0300. URL <https://doi.org/10.1145/3649449>.

Wendi Li and Yixuan Li. Process reward model with q-value rankings, 2024. URL <https://doi.org/10.48550/arXiv.2410.11287>.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2024. URL <https://openreview.net/forum?id=v8L0pN6E0i>.

Zongyu Lin, Yao Tang, Da Yin, Stuart X. Yao, Ziniu Hu, Yizhou Sun, and Kai-Wei Chang. Q* agent: Optimizing language agents with q-guided exploration, 12 2024. URL <https://openreview.net/forum?id=rxUz2DaulF>.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms, 2024a. URL <https://doi.org/10.48550/arXiv.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. Agentbench: Evaluating llms as agents, 2024b. URL <https://openreview.net/forum?id=zAdUB0aCTQ>.

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. BOLAA: benchmarking and orchestrating llm-augmented autonomous agents, 2023. URL <https://doi.org/10.48550/arXiv.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. Improve mathematical reasoning in language models by automated process supervision, 2024. URL <https://doi.org/10.48550/arXiv.2406.06592>.

Bingchen Miao, Yang Wu, Minghe Gao, Qifan Yu, Wendong Bu, Wenqiao Zhang, Yunfei Li, Siliang Tang, Tat-Seng Chua, and Juncheng Li. Boosting virtual agent learning and reasoning: A step-wise, multi-dimensional, and generalist reward model with benchmark, 2025. URL <https://arxiv.org/abs/2503.18665>.

Yasmin Moslem, Rejwanul Haque, John D. Kelleher, and Andy Way. Adaptive machine translation with large language models, 2023. URL <https://arxiv.org/abs/2301.13294>.

OpenAI. Hello gpt-4o, 5 2024a. URL <https://openai.com/index/hello-gpt-4o/>.

OpenAI. Introducing openai o1-preview, 9 2024b. URL <https://openai.com/index/introducing-openai-o1-preview/>.

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. Training language models to follow instructions with human feedback, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Dongmin Park, Minkyu Kim, Beongjun Choi, Junhyuck Kim, Keon Lee, Jonghyun Lee, Inkyu Park, Byeong-Uk Lee, Jaeyoung Hwang, Jaewoo Ahn, Ameya S. Mahabaleshwarkar, Bilal Kartal, Pritam Biswas, Yoshi Suhara, Kangwook Lee, and Jaewoong Cho. Orak: A foundational benchmark for

training and evaluating llm agents on diverse video games, 2025. URL <https://arxiv.org/abs/2506.03610>.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. Adapt: As-needed decomposition and planning with language models, 2024. URL <https://doi.org/10.18653/v1/2024.findings-naacl.264>.

QwenTeam. Qwen2.5: A party of foundation models, September 2024. URL <https://qwenlm.github.io/blog/qwen2.5/>.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2016. URL <http://arxiv.org/abs/1506.02438>.

Amrit Sethuraman, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning, 2024. URL <https://doi.org/10.48550/arXiv.2410.08146>.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters, 2024. URL <https://doi.org/10.48550/arXiv.2408.03314>.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization for llm agents, 2024. URL <https://arxiv.org/abs/2403.02502>.

Richard S Sutton. Learning to predict by the methods of temporal differences, 1988.

Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation, 1999.

Liyan Tang, Zhaoyi Sun, Betina Idnay, Jordan G Nestor, Ali Soroush, Pierre A Elias, Ziyang Xu, Ying Ding, Greg Durrett, Justin F Rousseau, et al. Evaluating large language models on medical evidence summarization, 2023.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. A survey on self-evolution of large language models, 2024. URL <https://arxiv.org/abs/2404.14387>.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Christian 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, Bin 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. Llama 2: Open foundation and fine-tuned chat models, 2023. URL <https://doi.org/10.48550/arXiv.2307.09288>.

Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. ReFT: Reasoning with reinforced fine-tuning, August 2024. URL <https://aclanthology.org/2024.acl-long.410/>.

Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Y. Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022. URL <https://doi.org/10.48550/arXiv.2211.14275>.

Dave Van Veen, Cara Van Uden, Louis Blankemeier, Jean-Benoit Delbrouck, Asad Aali, Christian Bluethgen, Anuj Pareek, Małgorzata Polacin, Eduardo Pontes Reis, Anna Seehofnerová, et al. Adapted large language models can outperform medical experts in clinical text summarization, 2024.

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. Secrets of RLHF in large language models part II: reward modeling, 2024a. URL <https://doi.org/10.48550/arXiv.2401.06080>.

Jiawei Wang, Kai Wang, Shaojie Lin, Runze Wu, Bihan Xu, Lingeng Jiang, Shiwei Zhao, Renyu Zhu, Haoyu Liu, Zhipeng Hu, Zhong Fan, Le Li, Tangjie Lyu, and Changjie Fan. Digital player: Evaluating large language models based human-like agent in games, 2025a. URL <https://arxiv.org/abs/2502.20807>.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jia-kai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents, 2024b. URL <https://doi.org/10.1007/s11704-024-40231-1>.

Longyue Wang, Chenyang Lyu, Tianbo Ji, Zhirui Zhang, Dian Yu, Shuming Shi, and Zhaopeng Tu. Document-level machine translation with large language models, 2023. URL <https://arxiv.org/abs/2304.02210>.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024c. URL <https://doi.org/10.18653/v1/2024.acl-long.510>.

Weiyun Wang, Zhangwei Gao, Lianjie Chen, Zhe Chen, Jinguo Zhu, Xiangyu Zhao, Yangzhou Liu, Yue Cao, Shenglong Ye, Xizhou Zhu, Lewei Lu, Haodong Duan, Yu Qiao, Jifeng Dai, and Wenhui Wang. Visualprm: An effective process reward model for multimodal reasoning, 2025b. URL <https://arxiv.org/abs/2503.10291>.

Yu Wang, Nan Yang, Liang Wang, and Furu Wei. Examining false positives under inference scaling for mathematical reasoning, 2025c.

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. The rise and potential of large language model based agents: A survey, 2023. URL <https://doi.org/10.48550/arXiv.2309.07864>.

Zhiheng Xi, Wenxiang Chen, Boyang Hong, Senjie Jin, Rui Zheng, Wei He, Yiwen Ding, Shichun Liu, Xin Guo, Junzhe Wang, Honglin Guo, Wei Shen, Xiaoran Fan, Yuhao Zhou, Shihan Dou, Xiao Wang, Xinbo Zhang, Peng Sun, Tao Gui, Qi Zhang, and Xuanjing Huang. Training large language models for reasoning through reverse curriculum reinforcement learning, 2024a. URL <https://arxiv.org/abs/2402.05808>.

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. Agentgym: Evolving large language model-based agents across diverse environments, 2024b. URL <https://doi.org/10.48550/arXiv.2406.04151>.

Yu Xia, Jingru Fan, Weize Chen, Siyu Yan, Xin Cong, Zhong Zhang, Yaxi Lu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Agentrm: Enhancing agent generalization with reward modeling, 2025. URL <https://arxiv.org/abs/2502.18407>.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Watch every step! llm agent learning via iterative step-level process refinement, 2024. URL <https://arxiv.org/abs/2406.11176>.

Haoran Xu, Young Jin Kim, Amr Sharaf, and Hany Hassan Awadalla. A paradigm shift in machine translation: Boosting translation performance of large language models, 2024. URL <https://arxiv.org/abs/2309.11674>.

Nancy Xu, Sam Masling, Michael Du, Giovanni Campagna, Larry Heck, James Landay, and Monica S Lam. Grounding open-domain instructions to automate web support tasks, 2021. URL <https://arxiv.org/abs/2103.16057>.

Hui Yang, Sifu Yue, and Yunzhong He. Auto-gpt for online decision making: Benchmarks and additional opinions, 2023. URL <https://doi.org/10.48550/arXiv.2306.02224>.

Zonghan Yang, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. React meets actre: When language agents enjoy training data autonomy, 2024. URL <https://arxiv.org/abs/2403.14589>.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Fei Yu, Anningzhe Gao, and Benyou Wang. Ovm, outcome-supervised value models for planning in mathematical reasoning, 2024. URL <https://doi.org/10.18653/v1/2024.findings-naacl.55>.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023. URL <https://arxiv.org/abs/2308.01825>.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STar: Bootstrapping reasoning with reasoning, 2022. URL https://openreview.net/forum?id=_3ELRdg2sgI.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms, 2024. URL <https://doi.org/10.18653/v1/2024.findings-acl.181>.

Yuanzhao Zhai, Tingkai Yang, Kele Xu, Dawei Feng, Cheng Yang, Bo Ding, and Huaimin Wang. Enhancing decision-making for LLM agents via step-level q-value models, 2024. URL <https://doi.org/10.48550/arXiv.2409.09345>.

Biao Zhang, Barry Haddow, and Alexandra Birch. Prompting large language model for machine translation: A case study, 23–29 Jul 2023a. URL <https://proceedings.mlr.press/v202/zhang23m.html>.

Dan Zhang, Sining Zhoubian, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: LLM self-training via process reward guided tree search, 2024a. URL <https://doi.org/10.48550/arXiv.2406.03816>.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. A survey of controllable text generation using transformer-based pre-trained language models, October 2023b. ISSN 0360-0300. URL <https://doi.org/10.1145/3617680>.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B. Hashimoto. Benchmarking large language models for news summarization, 01 2024b. ISSN 2307-387X.

Yang Zhang, Hanlei Jin, Dan Meng, Jun Wang, and Jinghua Tan. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods, 2025a. URL <https://arxiv.org/abs/2403.02901>.

Yimeng Zhang, Tian Wang, Jiri Gesi, Ziyi Wang, Yuxuan Lu, Jiacheng Lin, Sinong Zhan, Vianne Gao, Ruochen Jiao, Junze Liu, Kun Qian, Yuxin Tang, Ran Xue, Houyu Zhang, Qingjun Cui, Yufan Guo, and Dakuo Wang. Shop-r1: Rewarding llms to simulate human behavior in online shopping via reinforcement learning, 2025b. URL <https://arxiv.org/abs/2507.17842>.

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. Webarena: A realistic web environment for building autonomous agents, 2024. URL <https://openreview.net/forum?id=oKn9c6ytLx>.

Appendix

A. More Detailed Discussion of Related Work

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

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., 2025b). 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. IPR (Xiong et al., 2024) acquires step-level reward by exploring from every step and calculate the mean reward of the explorations. AgentRM (Xia et al., 2025) deploys a MCTS-inspired method to collect the trajectories for training and estimate the values of each step with information stored in the search tree. Similar (Miao et al., 2025) also proposes a MCTS-based dataset collecting method. To better adapt to its real-world tasks, it evaluates the step-level value from 5 dimensions based on the outcome reward as well as information like length of the trajectories. However, 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).

Table 4 | Comparison of different process supervision paradigms. “SG” means Supervision Granularity, and “P” means Progress.

Method	Labeling	SG	P	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., 2025b)	MC-based	Outcome	✗	Agent
Q* Agent (Lin et al., 2024)	TD-based	Process	✗	Agent
DPO-Q (Zhai et al., 2024)	MC-based	Process	✗	Agent
IPR (Xiong et al., 2024)	MC-based	Process	✗	Agent
AgentRM (Xia et al., 2025)	MC-based	Process	✗	Agent
Similar (Miao et al., 2025)	MC-based	Process	✗	Agent
AgentPRM (Ours)	TD-based	Process	✓	Agent, Reasoning

B. Algorithm

We demonstrate the training algorithm of AgentPRM in 1, and the process of beam search in Algorithm 2.

Algorithm 1: Training of AgentPRM

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

```

1 Procedure Trajectories collection
2    $\mathcal{D}_{train} \leftarrow [\ ]$                                  $\triangleright$  Initialize AgentPRM Train set  $\mathcal{D}_{train}$ 
3   for  $s_0^i$  in  $\{s_0^i\}_{i=1}^{N_{Task}}$  do
4     for  $n = 1$  to  $N_{TD}$  do
5        $\tau \leftarrow \pi_\theta(s_0^i);$ 
6       Add  $\tau$  to  $\mathcal{D}_{train};$ 
7     end
8   end
9 Procedure AgentPRM model training
10  for  $n = 1$  to  $m$  do
11    for batch in  $\mathcal{D}_{train}$  do
12      for trajectory  $\tau$  in batch do
13         $Q \leftarrow [\ ];$                                       $\triangleright$  AgentPRM model estimated value list  $Q$ 
14        for  $(s_t, a_t)$  in  $\tau$  do
15           $Q_t \leftarrow \mathcal{M}_\phi(s_t, a_t)$ 
16          Add  $Q_t$  to  $Q;$ 
17        end
18         $\hat{A} \leftarrow GAE(Q, r(\tau));$ 
19         $\hat{Q} \leftarrow TD(\mathcal{A}, Q)$ 
20         $\mathcal{L}_Q = \mathbb{E} \left[ \frac{1}{2}(Q_t - \hat{Q}_t)^2 \right]$ 
21         $\mathcal{L}_A = \mathbb{E} \left[ \frac{1}{2}((Q_t - Q_{t-1}) - (\hat{Q}_t - \hat{Q}_{t-1}))^2 \right]$ 
22         $\mathcal{M}_\phi \leftarrow \text{Back\_Propagation}(\mathcal{L}_Q + \beta \mathcal{L}_A)$ 
23      end
24    end
25  end
```

C. More Implementation Details for RL

We train LLM agents with PPO using a batch size of 16, a learning rate of 1×10^{-6} , a KL coefficient of 1×10^{-3} , and a sampling temperature of 1.0. For both BabyAI and TextCraft, the maximum interaction horizon is set to 20. In the PRM-based RL, we use the PRM’s predicted score at the final step as the reward for the trajectory.

D. Qualitative Analysis

We perform a qualitative analysis to show how AgentPRM works. The case shown in Figure 9 demonstrates the process of beam search guided by AgentPRM. The LLM agent successfully solves this task under the guidance of AgentPRM. 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).

Algorithm 2: Beam search with PRM.

```

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

1 Procedure Step-level beam search with PRM
2    $C = [\mathbf{s}_0] * M, t = 0$                                      ▷ Initialize candidates
3   while  $t < T$  and non-terminal path in  $C$  do
4      $C_{t+1} \leftarrow []$                                          ▷ Initialize priority queue
5     for  $\mathbf{s}_t$  in  $C$  do
6       Sample  $\{\mathbf{a}^{(b)}\}_{b=1}^{B_2} \sim \pi_\theta(\mathbf{s}_t)$ 
7       for  $b = 1$  to  $M$  do
8          $\mathbf{s}_{t+1} = \text{Concat}[\mathbf{s}_t, \mathbf{a}^{(b)}]$ 
9         Add  $(\mathbf{s}_{t+1}, \mathcal{M}_\phi(\mathbf{s}_{t+1}))$  to  $C_{t+1}$ 
10      end
11       $C \leftarrow \text{Top-}N$  of  $C_{t+1}$ 
12    end
13  end
14  return Top-1 of  $C$                                          ▷ Return top-1 as the final solution path

```

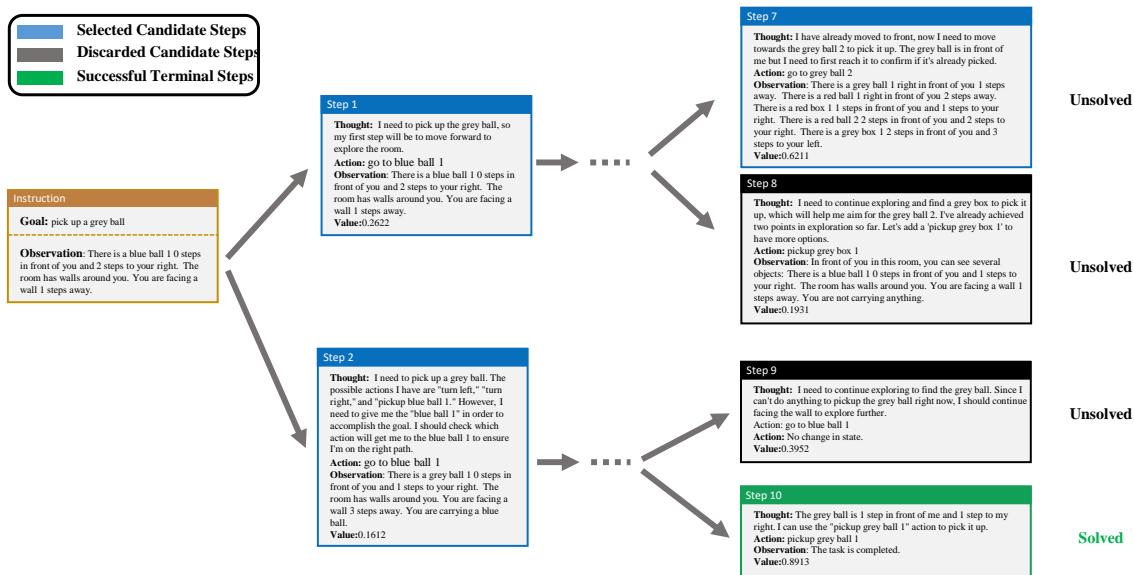


Figure 9 | Example of qualitative analysis on beam search. This figure demonstrates a successful solution with beam search guided by AgentPRM. The policy model solves this task in 10 steps under the guidance of AgentPRM.