AgentPro: Enhancing LLM Agents with Automated Process Supervision

Anonymous ACL submission

Abstract

Large language model (LLM) agents have demonstrated significant potential for addressing complex tasks through mechanisms such as chain-of-thought reasoning and tool invocation. However, current frameworks lack explicit supervision during the reasoning process, which may lead to error propagation across reasoning chains and hinder the optimization of intermediate decision-making stages. This paper introduces a novel framework, **AgentPro**, which enhances LLM **agent** performance by automated **pro**cess supervision. AgentPro employs Monte Carlo Tree Search to automatically generate step-level annotations, and develops a process reward model based on these annotations to facilitate fine-grained quality assessment of reasoning. By employing a rejection sampling strategy, the LLM agent dynamically adjusts generation probability distributions to prevent the continuation of erroneous paths, thereby improving reasoning capabilities. Extensive experiments on four datasets indicate that our method significantly outperforms existing agent-based LLM methods (e.g., achieving a 6.32% increase in accuracy on the HotpotQA dataset), underscoring its proficiency in managing intricate reasoning chains.

1 Introduction

Recent advancements in large language models (LLMs), such as GPT-4 (OpenAI, 2023), PaLM (Anil et al., 2023), and LLaMA (Dubey et al., 2024), have showcased impressive capabilities in semantic understanding, knowledge reasoning, and cross-task generalization. Meanwhile, research indicates that LLM-based agents can effectively decompose complex, multi-step tasks through chain-of-thought prompting and tool invocation mechanisms (Schick et al., 2024). This has led to significant advancements in mathematical problem-solving (Lightman et al., 2023) and in scenarios involving embodied interaction (Shridhar et al., 2020; Wei et al., 2022; Yao et al., 2023).

Despite the demonstrated potential of LLM agents in managing complex tasks, existing frameworks face significant challenges in utilizing process supervision to train and improve these agents. Conventional approaches, such as ReAct (Yao et al., 2022b), which decomposes tasks through alternating reasoning trajectory generation and actions; and Reflexion (Shinn et al., 2024), which uses self-reflection mechanisms for policy iteration, primarily concentrate on the accuracy of final task outcomes while neglecting explicit supervision of intermediate reasoning steps. This design leads to two major issues. First, the lack of real-time error detection and correction mechanisms allows local errors to propagate along reasoning chains during extended inferences (Lightman et al., 2023). Second, the absence of mechanisms in output feedback optimization to discern subtle differences in reasoning quality hinders the identification of optimal decision paths (Wang et al., 2024). For example, QueryAgent (Huang et al., 2024) utilizes environmental feedback for self-correction. However, this feedback primarily relates to task outcomes and fails to pinpoint specific errors in the process.

In mathematical reasoning tasks, process supervision techniques, such as the use of Process Reward Models (PRMs), have proven to be highly effective (Lightman et al., 2023). However, integrating PRMs into LLM agents remains underexplored, primarily due to the substantial costs associated with manual, step-wise supervision. Specifically, creating training datasets for PRMs necessitates human labeling to assess the correctness of each step. Therefore, developing cost-effective, automated process supervision methods is essential for improving the reasoning capabilities and robustness of LLM agents in handling complex tasks.

In this paper, we present AgentPro, a novel framework for LLM **Agent**s that incorporates an Automated **Pro**cess Supervision mechanism to address complex tasks such as reasoning and decision

making. Our approach employs Monte Carlo Tree Search (MCTS) (Wang et al., 2024; Świechowski et al., 2023) to generate step-level labels, which facilitate the training of a process reward model (PRM). The PRM improves its comprehension by learning from automatically annotated data, thereby providing detailed evaluations of each reasoning step. Our goal is to improve the reasoning abilities of an LLM agent by applying the Rejection Sampling (RS) (Liu et al., 2023; Yuan et al., 2023) strategy within the framework of reinforcement learning from human feedback (Bai et al., 2022). This approach adjusts the generation probability distributions based on feedback from the PRM, thereby minimizing error propagation along reasoning paths. We evaluated AgentPro's performance across four distinct datasets in various task domains, demonstrating its substantial enhancements in both the accuracy and robustness of LLM agents when addressing complex tasks. Specifically, our method achieved a 6.32% increase in accuracy on the HotpotQA dataset within multi-hop question answering scenarios, highlighting its proficiency in handling intricate reasoning chains.

Our main contributions are as follows:

- We present the first LLM agent framework that integrates automated process supervision to optimize reasoning and decision-making tasks. Our framework conducts real-time quality evaluation at each step, effectively mitigating the issue of error propagation found in traditional approaches.
- We propose an automated label generation algorithm based on Monte Carlo Tree Search (MCTS) that minimizes the cost of step-wise annotation for training PRMs, thereby facilitating the feasibility of large-scale process supervision.
- We conduct extensive experiments on four datasets, considering both reasoning and decision-making scenarios. The results indicate that our method achieves superior performance compared to existing agent-based methods.

2 Related Work

2.1 Large Language Model Agent

LLM agents (Yao et al., 2023; Sumers et al., 2023; Gong et al., 2023) exhibit exceptional reasoning capabilities across a variety of problem domains, including embodied environments and reasoning tasks. CAMEL (Li et al., 2023) introduces a multi-agent role-playing framework that guides agents in task completion by utilizing initial prompts and detailed constraints while ensuring alignment with human intentions. AutoAgents (Chen et al., 2023) emphasize the autonomous generation of language agents to adapt to diverse task requirements. ExpeL (Zhao et al., 2024) achieves continuous performance enhancement and transfer learning in decision-making tasks by autonomously collecting experiences and extracting knowledge through natural language. Existing approaches primarily focus on the final outcomes. In contrast, our approach addresses error propagation in extensive reasoning chains by providing explicit process supervision of the intermediate reasoning steps.

2.2 Reasoning-Action Framework

Integrating actions with reasoning enhances the efficiency and accuracy of LLMs in multi-step, conditional problems (Gong et al., 2023; Chen et al., 2023; Huang et al., 2024; Arora et al., 2024). ReAct (Yao et al., 2022b) integrates reasoning with action generation, decreasing hallucination in chain-of-thought processes. AUTOACT (Qiao et al., 2024) introduces a self-planning framework for question-answering that facilitates agent learning from scratch, addressing limitations of single-model systems. LLM+AL (Ishay and Lee, 2025) merges LLMs with action languages, utilizing their strengths in semantic parsing, commonsense generation, and automated reasoning tasks. However, these methods do not thoroughly evaluate the quality of intermediate steps, and we address this by employing rejection sampling for dynamic error correction during reasoning.

2.3 Process Reward Model

PRM enhances the accuracy of reasoning by supervising and emphasizing intermediate steps, rewarding correct inferences and penalizing errors, in contrast to traditional methods that solely concentrate on final outcomes (Uesato et al., 2022; Lightman et al., 2023; Wang et al., 2024; Zhang et al., 2025). It allows for more precise pattern learning and handles the misalignment between correct conclusions and flawed derivations. For example, Math-Shepherd (Wang et al., 2024) employs automated annotation to provide step-level rewards, and DeepMind's Process Advantage Verifiers assign rewards based on the correctness probability changes at each step (Setlur et al., 2024). However, existing PRM implementations are limited to math problems. Our work expands PRM to a wider range of reasoning and decision-making tasks.

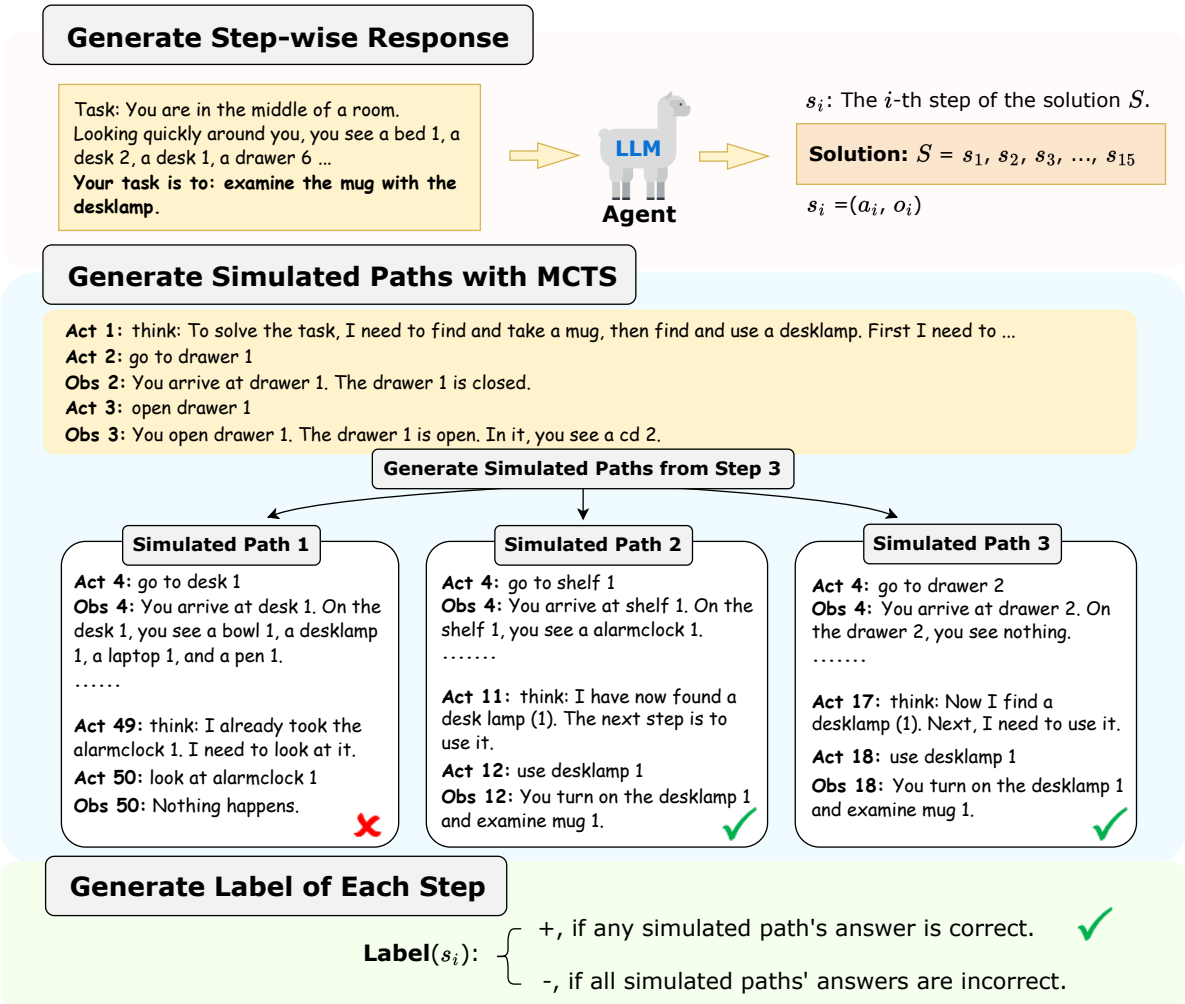

Figure 1: Overview of the Monte Carlo Tree Search-driven automatic labeling framework. First, the LLM Agent generates a solution S for a given task q. Then, MCTS is employed to simulate multiple decision-making paths for a given step s_i (in this case $i = 3$). Finally, the results of these simulations determine if the step s_i is labeled as correct or incorrect, depending on whether any of the simulated trajectories reach the correct final answer.

3 Method

Existing LLM frameworks often lack robust verification mechanisms to ensure the accuracy of intermediate steps in complex problem-solving tasks. Furthermore, manual annotation methods are prohibitively expensive. To address these challenges, we propose a novel framework, AgentPro, designed to enhance the reasoning capabilities of LLM agents while minimizing costs.

AgentPro comprises two core components:

- **LLM Agent**: The LLM Agent M_{agent} is the core component responsible for generating step-by-step solutions (responses) and is the agent to be fine-tuned in the end. Given a query q, the agent generates a solution $S \leftarrow M_{\text{agent}}(\{q\})$, where $S = \{s_1, s_2, s_3, \ldots, s_K\}$ represents the sequence of steps and K denotes the total number of steps in the solution. For decision-making tasks such as in AlfWorld, each step s_i can defined as $s_i = (a_i, o_i)$, where a_i is an executable action (e.g., "Go to desk 1") and o_i is an optional observation from the environment (e.g., "You arrive at desk 1 and see a bowl 1 on the desk").

- **Process Reward Model**: The process reward model (PRM), denoted as M_{prm}, is utilized to assess the step-wise accuracy of a solution S. It assigns a score to each step s_i to quantify the correctness of that specific step.

Building on these two components, we devised a three-phase optimization framework that utilizes Monte Carlo Tree Search to automatically generate a training dataset for the PRM M_{prm}, which is then employed to improve the reasoning abilities of the

LLM Agent M_{agent} through reinforcement learning. The framework functions as follows:

1. **Automated Process Annotation**: We employ the LLM Agent to produce multi-step trajectories, with each intermediate step s_i being automatically annotated using labels obtained from MCTS simulations. This phase facilitates the creation of a self-supervised dataset, which is subsequently used to train the PRM.
2. **Training of the Process Reward Model**: Utilizing the annotated steps from Phase 1, we fine-tune the PRM M_{prm} to accurately predict correctness scores for each individual step s_i, thereby equipping the PRM with the capability to assess step-wise validity within complex problem-solving scenarios.
3. **Reinforcement Learning**: We adopted the rejection sampling strategy, a technique within the framework of reinforcement learning from human feedback (RLHF), to enhance the capabilities of our LLM Agent M_{agent}, where the PRM M_{prm} functions as the reward model.

3.1 Gathering Step-Wise Trajectories

To enable effective process supervision, it is essential to ensure that the answers from the LLM Agent are generated in a step-by-step manner. A robust base model is critical for generating structured, step-wise responses, leveraging its contextual understanding and reasoning capabilities developed through extensive training on large-scale corpora. Consequently, a good base model ensures the high quality of the generated responses. In our setting, we utilize pre-trained LLMs, such as LLaMA-3.1-8B-Instruct (Dubey et al., 2024) , as the base model for our agent.

Meanwhile, every step of a response must be meticulously documented to facilitate the subsequent automatic labeling process, which is crucial for training the PRM. For each query q, the LLM agent performs a comprehensive semantic analysis to extract crucial information and delineate the inherent reasoning framework. This analysis involves consulting its internal knowledge base to construct logically coherent reasoning or decision-making trajectories. Specifically, the solution $S \leftarrow M_{\text{agent}}(\{q\})$ must be delineated in a step-wise manner and presented as $S = \{s_1, s_2, s_3, \ldots, s_K\}$. Then the trajectory T_q can be represented as:

$$T_q = \{q, s_1, s_2, s_3, \ldots, s_K\} \tag{1}$$

For example, for a given decision-making task (query) q illustrated in Fig. 1 within the AlfWorld scenario, the agent generates a solution S comprising $K = 15$ steps. Thus, the trajectory for task q can be represented as $T_q = \{q, s_1, s_2, \ldots, s_{15}\}$.

3.2 Monte Carlo Tree Search-Driven Automatic Labeling

To systematically evaluate the quality of each step s_i generated by the LLM agent M_{agent}, we define the following criterion: a step s_i is deemed high-quality if it contributes to deriving the ground-truth answer A_{correct}. This criterion aligns with the fundamental objective of multi-step reasoning and decision-making tasks, where intermediate steps should progressively lead to a correct final result. Our goal is to design a method to automatically assess the contribution of each step.

Monte Carlo Tree Search (MCTS) (Świechowski et al., 2023) is a heuristic search algorithm renowned for its effectiveness in navigating complex decision spaces under uncertainty. Building upon its strong simulation performance, we introduce an MCTS-based automatic labeling framework for training the PRM. For each step s_i, we generate m simulated paths p_1^i to p_m^i for s_i using the path from query q to s_i in the trajectory T_q described by Eq. 1. Each path p_j^i is constructed as:

$$p_j^i \leftarrow M_{\text{agent}}(\{q, s_1, \ldots, s_i\}) \tag{2}$$

Let $A_{\text{final}}^{(j)}$ denote the final answer of the j-th path p_j^i. The label for s_i is determined based on whether any of the m paths from p_1^i to p_m^i result in A_{correct}:

$$\text{Label}(s_i) = \begin{cases} + & \text{if } \exists\, j \in [1, m],\ A_{\text{final}}^{(j)} = A_{\text{correct}}, \\ - & \text{if } \forall\, j \in [1, m],\ A_{\text{final}}^{(j)} \neq A_{\text{correct}}. \end{cases} \tag{3}$$

As illustrated in Fig. 1, we generated $m = 3$ simulated paths from p_1^3 to p_3^3 for step s_3. Given that the task q in AlfWorld pertains to decision-making, each step s_i can be represented as $s_i = (a_i, o_i)$, where o_i is optional. Consequently, each path p_j^3 is derived from $p_j^3 \leftarrow M_{\text{agent}}(\{q, a_1, a_2, o_2, a_3, o_3\})$. Since the outcomes of both path p_2^3 and path p_3^3 lead to the correct answer, "You turn on desk lamp 1 and examine mug 1", we label step s_3 as "+".

Utilizing the MCTS-based automatic labeling framework, we label each step s_i within trajectory T_q. Upon completion of the labeling process for all steps across all queries $Q = \{q_j\}_{j=1}^N$, we generate a labeled dataset $D_{prm} = \{d_j\}_{j=1}^N$. This dataset consists of N samples, each corresponding to one

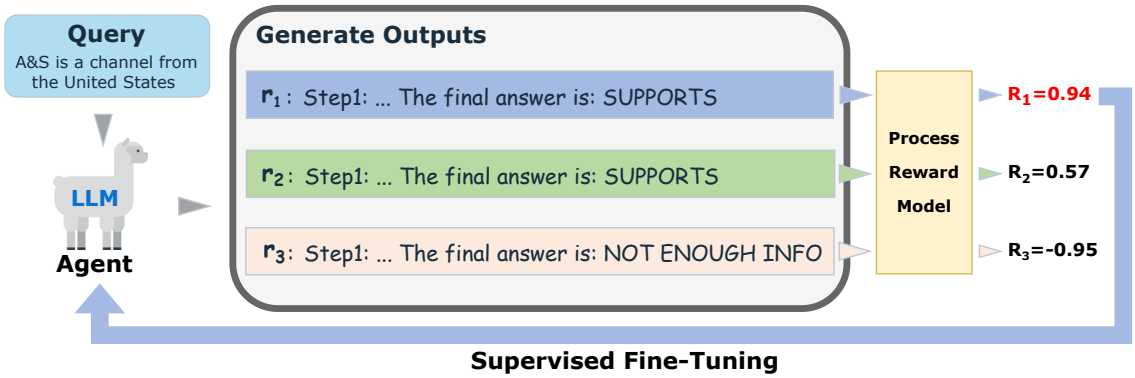

Figure 2: Overview of the training process for LLM Agent M_{agent}. For a given query q, the agent generates multiple step-wise responses, and each response r_i is subsequently evaluated by the PRM M_{prm}. The agent is then fine-tuned based on the response r_* with the highest reward score R_*. The whole process will repeat until the agent converges.

of the N queries in set Q. For each data sample d_j, the input $X = \{q_j, s_1^j, s_2^j, \ldots, s_K^j\}$, which represents the trajectory T_q^j generated from the j-th query q_j that includes K steps. The output (label) y is an array of K elements, where each element corresponds to the label associated with step s_i:

$$d_j = (\underbrace{\{q_j, s_1^j, \ldots, s_K^j\}}_{X}, \underbrace{\{y_{s_1}^j, y_{s_2}^j, \ldots, y_{s_K}^j\}}_{y}) \quad (4)$$

where $y_{s_i}^j$ represents the label of step s_i^j, $\text{Label}(s_i^j)$.

3.3 Process Reward Model Training

We utilize the dataset D_{prm} in Sec. 3.2 to train our process reward model M_{prm}, enabling it to evaluate step-wise responses in complex problem-solving scenarios. The PRM is also constructed based on a pre-trained LLM. We adopt the full parameter fine-tuning strategy for training, where all parameters of M_{prm} are optimized by gradient descent to minimize prediction errors in step-wise assessments:

$$\mathcal{L}_{prm} = -\frac{1}{N}\frac{1}{K}\sum_{j=1}^{N}\sum_{i=1}^{K} CrossEntropy(y_{s_i}^j, \hat{y}_{s_i}^j) \quad (5)$$

where $\hat{y}_{s_i}^j$ denotes the predicted correctness score for step s_i^j of the j-th training sample, while $CrossEntropy$ signifies the cross entropy loss.

This full-parameter fine-tuning strategy effectively utilizes fine-grained supervisory labels from the training data, enabling the trained PRM to accurately determine if each generated step s_i logically progresses toward resolving the problem. Consequently, it provides high-quality reward scores for training LLM agents.

3.4 LLM Agent Training

Drawing on the principles of reinforcement learning (Kaelbling et al., 1996), we introduce an optimization mechanism that employs rejection sampling (Liu et al., 2023) to enhance the performance of our LLM agent. This mechanism involves iteratively generating multiple step-wise responses by the LLM agent M_{agent}, evaluating these responses using our well-trained PRM M_{prm}, and selecting the response with the highest reward score to train the LLM agent through supervised fine-tuning.

Specifically, for each query q, the LLM agent M_{agent} first generates n candidate responses, ranging from r_1 to r_n, defined as $r_j \leftarrow M_{\text{agent}}(\{q\})$. Similar to the solution S described in Sec. 3.1, each response r_j should be generated in a step-wise format: $r_j = \{s_1, s_2, s_3, \ldots, s_K\}$. The PRM M_{prm} will evaluate each step s_i in r_j and assign a label l_i ($+$ or $-$) to s_i, along with the probability P_i for generating that label. The reward score R_j for response r_j is then calculated as the average probability of the label values across all steps. During calculation, we negate those probabilities associated with negatively labeled steps to reflect our disinclination towards incorrect steps:

$$R_j = \frac{1}{K}\sum_{i=1}^{K}(\mathbb{I}(l_i = +) \cdot P_i - \mathbb{I}(l_i = -) \cdot P_i) \quad (6)$$

where $\mathbb{I}$ is the indicator function. The response r_* with the highest reward score R_* is selected as the optimal response for further training:

$$r_* = \arg\max_{r_j} R_j \quad (7)$$

After selecting the optimal responses for all queries, we utilize these responses to fine-tune the

LLM agent M_{agent}. Consistent with Sec. 3.3, we employ the full-parameter fine-tuning strategy on M_{agent} to maximize the likelihood of replicating the optimal reasoning steps:

$$\mathcal{L}_{agent} = -\frac{1}{N_r}\sum_{i=1}^{N_r}\sum_{t=1}^{T_i}\log P(y_{i,t}|x_i) \quad (8)$$

where x_i is the i-th query (prompt) among N_r training samples, y_i is the step-wise response for x_i, $y_{i,t}$ is the t-th word of y_i, and T_i is the length of y_i.

Fig. 2 illustrates an example of the agent training process, wherein the agent generates three step-wise responses. The first response, which exhibits the highest reward score ($R_1 = 0.94$), is chosen for training the agent. The training process is repeated until the agent develops sufficient capabilities to manage complex multi-step reasoning tasks.

In summary, our framework enhances the performance of the LLM agent through MCTS-based automated process supervision and rejection sampling training strategy. For additional details on the algorithm, please refer to Appendix A.

4 Experiments

4.1 Experimental Setup

Datasets. We selected four representative benchmark datasets to evaluate the performance of our method: FEVER (Thorne et al., 2018), HotpotQA (Yang et al., 2018), ALFWorld (Shridhar et al., 2020), and WebShop (Yao et al., 2022a). The FEVER dataset is utilized for assessing fact extraction and verification. HotpotQA serves as a benchmark for knowledge-intensive, multi-hop question answering. ALFWorld provides an embodied simulation environment for executing multi-step tasks in domestic contexts. Finally, WebShop functions as a benchmark for complex decision-making, simulating multi-modal interactions and product filtering in online shopping settings.

Baselines. We compare our method with four baseline methods: Act (Yao et al., 2022b), ReAct (Yao et al., 2022b), Reflexion (Shinn et al., 2024), and ExpeL (Zhao et al., 2024). The Act method generates responses through single-step predictions without explicit reasoning abilities. ReAct adopts a more structured approach by integrating reasoning and action in a feedback loop to enhance task performance. Reflexion uses verbal reinforcement to enable agents to learn from past mistakes, thereby improving the quality of reasoning. ExpeL, the state-of-the-art agent-based method, focuses on autonomously gathering experience from a set of training tasks to enhance the capabilities of LLM.

Implementation. Following ReAct, we use success rate (SR) as the evaluation metric: exact matching for HotpotQA and FEVER, timely task completion for ALFWorld, and purchasing an item matching all attributes for WebShop. We utilize the LLaMA-3.1-8B-Instruct model (Dubey et al., 2024) as the LLM agent for our method and all baselines. For FEVER and HotpotQA, Wikipedia information is appended to questions for additional context, a common practice among all baselines. For MCTS, we conduct $m = 3$ simulations to determine the label for each step. Regarding rejection sampling, $n = 4$ different responses are generated over $T = 5$ iterations during the training of our LLM agent. All results are averaged across three different random seeds. For further implementation details, please refer to Appendix B.

4.2 Main Results

Fig. 3 illustrates the average success rates of various methods across all four datasets. Results show that our method consistently outperforms existing baselines, which underscores the effectiveness of process supervision in enhancing the reasoning and sequential decision-making abilities of LLM agents. Specifically, in tasks requiring fine-grained factual inference (e.g., FEVER) and multi-hop reasoning (e.g., HotpotQA), our method systematically refines the reasoning process step by step using PRM. This leads to accuracy gains of 3.59% and 6.32%, respectively, over the best-performing baseline, ExpeL. These improvements suggest that PRM effectively identifies flawed reasoning paths and minimizes error accumulation, which is prevalent in traditional single-step predictions.

Furthermore, results from the ALFWorld environment demonstrate the impact of process rewards on physical reasoning. Across 134 household interaction scenarios, our approach achieves a 40.88% success rate, outperforming the best baseline, Reflexion (34.08%). This suggests that the automatically generated step-level annotations successfully capture environmental state transitions. The most significant improvement is observed in the WebShop tasks, where our method achieves a 52.67% success rate, which is substantially higher than ExpeL (38.00%) and Reflexion (40.67%). This finding indicates that PRM's ability to evaluate the quality of actions (such as price comparison and option matching), which helps the agent to build bet-

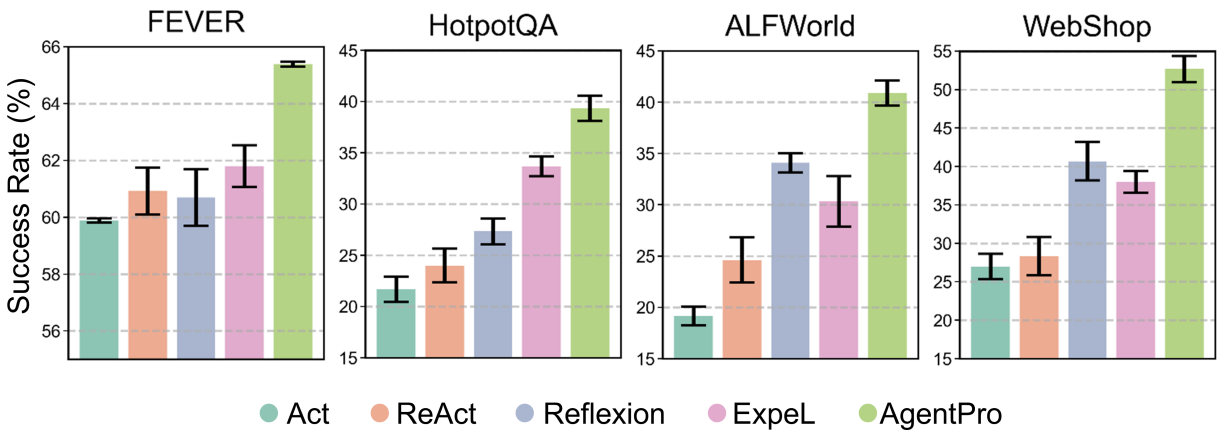

Figure 3: Average task success rates (%, mean±std) comparison of our method to other baselines on various tasks.

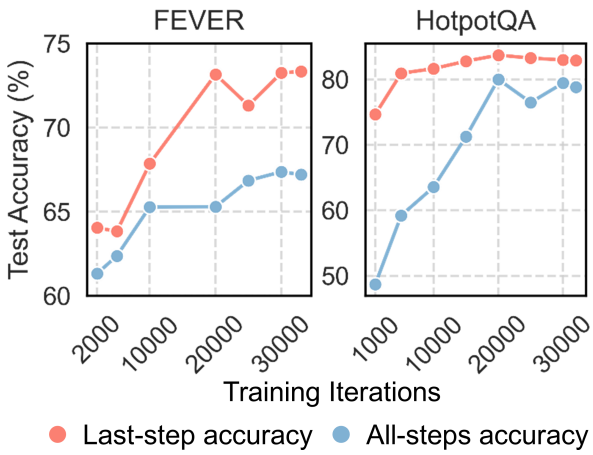

Figure 4: Performance of the trained process reward model (PRM) on the FEVER and HotpotQA datasets.

ter decision-making chains in shopping scenarios. All these experiments confirm that PRM, trained automatically using step-wise labels from MCTS, significantly improves the LLM agents.

4.3 Effectiveness of Process Reward Model

To evaluate the effectiveness of the trained process reward model (PRM), we employ two performance metrics: *last-step accuracy* and *all-steps accuracy*. Last-step accuracy measures the PRM's ability to correctly evaluate the label of the final reasoning step. In contrast, all-steps accuracy calculates the average accuracy of the PRM across all reasoning steps, thereby reflecting its overall capacity to assess the entire reasoning process.

As shown in Fig. 4, the PRM exhibits significant progressive convergence on both the FEVER and HotpotQA datasets. During the training iterations from 2,000 to 20,000, both last-step accuracy and all-steps accuracy display a consistent upward trend, with convergence speed showing marked nonlinearity. Specifically, on the FEVER dataset, last-step accuracy increases by 9% during this phase. Conversely, between 20,000 and 33,000 iterations, performance stabilizes, yielding only a marginal increase of 0.17%. On the HotpotQA dataset, all-steps accuracy reaches 80.00% by 20,000 iterations, after which improvements become negligible, maintaining stable performance for the remainder of training. Although all-steps accuracy is slightly lower than last-step accuracy, it remains adequate (78.82%) for the PRM to correctly evaluate and score the reasoning steps. Overall, these results indicate that the PRM demonstrates excellent stability during training and effectively scores the reasoning steps.

Table 1: Performance comparison (%) of different methods with the deepseek-llm-7b-chat model structure.

Dataset	Act	ReAct	Reflexion	ExpeL	AgentPro
Fever	40.01	44.89	49.97	51.87	57.07
WebShop	20.00	14.00	27.00	24.00	39.00

4.4 Impact of Model Structure

To validate the robustness of our method across different base models, we conducted comparative experiments using the *deepseek-llm-7b-chat* (Bi et al., 2024) model architecture. This model has a similar parameter size to LLaMA-3.1-8B-Instruct but features a significantly different architecture. Table 1 illustrates that although the deepseek model exhibits weaker reasoning abilities compared to llama, our method consistently surpasses all other baselines. Notably, for the knowledge reasoning tasks in FEVER, our method surpasses the best baseline, ExpeL, by 5.2%. For the interactive decision-making tasks in WebShop, the success rate of our

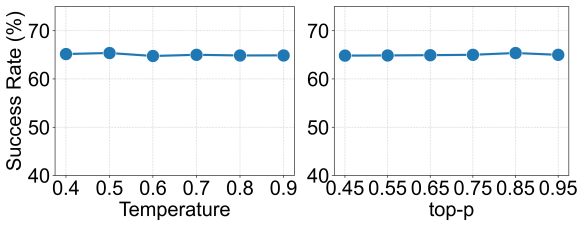

Figure 5: Impact of hyperparameters on agent performance for the FEVER dataset.

method reaches 39%, significantly outperforming the best baseline (27%). These results indicate that guiding the reasoning path of the LLM agent with PRM significantly boosts reasoning capabilities, irrespective of the base model architecture.

4.5 Ablation Studies

Hyperparameter sensitivity. We investigate the impact of the hyperparameters *top_p* and *temperature* of our method on LLM agent performance in generating step-wise responses. As shown in Fig. 5, when the temperature is fixed at 0.5, adjusting top_p within the range of 0.45 to 0.95 leads to less than a 0.54% variation in accuracy on the FEVER dataset (65.38% compared to 64.84%). This finding indicates that the model exhibits strong robustness to these hyperparameters. Similarly, fixing top_p at 0.85 and varying the temperature between 0.4 and 0.9 results in model accuracy remaining consistently within a 0.62% range, peaking at 65.38% at a temperature of 0.5. Notably, even under extreme temperature settings such as 0.4, the model maintained performance levels above 64.76%. This highlights the positive effect of process supervision mechanisms on the stability of the step-wise generation process.

Effect of MCTS simulation iterations. Table 2 shows how different numbers of MCTS simulation iterations impact the agent performance of our method on the HotpotQA dataset. The results demonstrate a significant increase in accuracy from 32% to 39% as the number of simulations rises from 1 to 3. Increasing the simulations to 5 yields a marginal improvement of 2% (from 39% to 41%). Beyond 5 simulations, the performance stabilizes at 42%. These findings suggest that the MCTS algorithm effectively identifies crucial path information within the first three simulations, and provides empirical evidence that a balance can be achieved between computational efficiency and model performance, as three simulations are adequate for achieving near-optimal outcomes.

Table 2: Success rates (%) of our method with different number of MCTS iterations on the HotpotQA dataset.

No. of MCTS iterations	1	3	5	7	9
Accuracy	32	39	41	42	42

Table 3: Accuracy (%) of our method with various number of generated responses during rejection sampling.

No. of Responses	3	4	5	6	7	8
AlfWorld	38	40	41	42	42	43
WebShop	47	50	52	54	55	55

Effect of number of generated responses during rejection sampling. We evaluated the effect of the number of generated responses (n) during rejection sampling on the performance of our method observed on the AlfWorld and Webshop datasets. As illustrated in Table 3, there is a clear pattern of "diminishing marginal returns" associated with increasing n. For instance, in the AlfWorld tasks, performance levels off when n reaches 6, with a marginal improvement from 42% to 43%. Similarly, in WebShop tasks, the accuracy improved from 47% to 55% when n increased from 3 to 7, with no notable improvements beyond this point. This pattern indicates that while initial increases in the number of generated responses significantly enhance the diversity and quality of candidate responses, further increases eventually result in minimal gains as the selection mechanism driven by the reward model reaches stabilization.

5 Conclusion

This paper presents AgentPro, a novel framework designed to improve the reasoning and decision-making abilities of LLM agents through automated process supervision and rejection sampling. Our framework leverages Monte Carlo Tree Search to automatically generate step-level labels and train a process reward model, which greatly reduces error propagation in the responses generated by large language model agents and facilitates real-time quality assessment of these responses. Extensive experiments on four datasets demonstrate that our method significantly enhances the accuracy and robustness of LLM agents. This work provides a scalable solution to the challenges associated with manual process supervision in complex and practical real-world tasks, thereby broadening the applicability of LLM agents across various domains.

6 Limitation

In contrast to prompt-based methods, our method necessitates further training of large language models to develop the process reward model and enhance the LLM agent. Specifically, we utilize MCTS to generate labels for intermediate steps and need to fine-tune all model parameters, which results in increased computational resource usage and extended training time. It is important to note that while our approach incurs higher training costs due to additional fine-tuning and inference of large language models, it remains significantly more cost-effective compared to manual process supervision requiring human labeling. In the future, we aim to explore more efficient training strategies, such as parameter-efficient fine-tuning and distillation techniques, to minimize resource consumption during the training process.

References

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Raghav Arora, Shivam Singh, Karthik Swaminathan, Ahana Datta, Snehasis Banerjee, Brojeshwar Bhowmick, Krishna Murthy Jatavallabhula, Mohan Sridharan, and Madhava Krishna. 2024. Anticipate & act: Integrating llms and classical planning for efficient task execution in household environments. In *International Conference on Robotics and Automation*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F Karlsson, Jie Fu, and Yemin Shi. 2023. Autoagents: A framework for automatic agent generation. *arXiv preprint arXiv:2309.17288*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Ran Gong, Qiuyuan Huang, Xiaojian Ma, Hoi Vo, Zane Durante, Yusuke Noda, Zilong Zheng, Song-Chun Zhu, Demetri Terzopoulos, Li Fei-Fei, et al. 2023. Mindagent: Emergent gaming interaction. *arXiv preprint arXiv:2309.09971*.

Xiang Huang, Sitao Cheng, Shanshan Huang, Jiayu Shen, Yong Xu, Chaoyun Zhang, and Yuzhong Qu. 2024. Queryagent: A reliable and efficient reasoning framework with environmental feedback based self-correction. *arXiv preprint arXiv:2403.11886*.

Adam Ishay and Joohyung Lee. 2025. Llm+ al: Bridging large language models and action languages for complex reasoning about actions. *arXiv preprint arXiv:2501.00830*.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008.

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

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

Hao Ma, Tianyi Hu, Zhiqiang Pu, Liu Boyin, Xiaolin Ai, Yanyan Liang, and Min Chen. 2024. Coevolving with the other you: Fine-tuning llm with sequential cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 37:15497–15525.

OpenAI. 2023. Gpt-4 technical report. *CoRR*, abs/2303.08774.

Feng Peiyuan, Yichen He, Guanhua Huang, Yuan Lin, Hanchong Zhang, Yuchen Zhang, and Hang Li. 2024. Agile: A novel reinforcement learning framework of llm agents. *Advances in Neural Information Processing Systems*, 37:5244–5284.

Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. 2024. Autoact: Automatic

agent learning from scratch via self-planning. *arXiv preprint arXiv:2401.05268*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2024. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36.

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. *arXiv preprint arXiv:2410.08146*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Theodore R Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. 2023. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*.

Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. 2023. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *NAACL-HLT*.

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

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

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

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, et al. 2023. Retroformer: Retrospective large language agents with policy gradient optimization. *arXiv preprint arXiv:2308.02151*.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 2023. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*.

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

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

A Algorithm Details

The details of our method are formally delineated in Algorithms 1 and 2. Our framework functions across three distinct phases:

- Phase 1: Automated Step Annotation via MCTS (Alg. 1, lines 1-24). For each query q_k from the Query Set Q, an initial step-wise solution S is generated using the LLM agent M_{agent} (line 4). Then, for each step s_i in S, we generate m MCTS simulations (lines 8-14) through path expansion. A step s_i is labeled positively $(+)$ if any simulation path results in the correct final answer A_{correct} (line 17). Otherwise a negative label $(-)$ is assigned if no path concludes correctly (line 19). Then, the dataset D_{prm} is constructed (lines 22-23) for training the PRM.

- Phase 2: Process Reward Model Training (Alg. 1, lines 25-29). We conducted supervised, full-parameter fine-tuning of the model M_{prm} across E_{prm} epochs utilizing the annotated dataset D_{prm}. This procedure enabled M_{prm} to acquire precise, step-wise evaluative capabilities essential for the subsequent training in rejection sampling.

- Phase 3: LLM Agent Training via PRM-Guided Rejection Sampling (Alg. 2). In each round t, we generate n candidate paths from r_1 to r_n for each query q using M_{agent} (line 8). Subsequently, we compute the reward score R_j for each j-th response r_j (lines 10-12) and select the response r_* with the highest reward score among all evaluated responses (line 15). The dataset $\mathcal{D}_{rs}$ is then updated to include these responses (line 16), which facilitates the full-parameter fine-tuning of M_{agent} (lines 18-20). Over T iterations, M_{agent} incrementally optimizes its alignment with trajectories that yield high rewards, while M_{prm} remains unchanged during the whole rejection sampling training process.

B Implementation Details

B.1 Details of datasets

As illustrated in Table 4, our study utilized four datasets: FEVER, HotpotQA, ALFWorld, and WebShop. The FEVER dataset initially comprised of 145,000 training and 19,000 test samples, and we selected 30,000 training samples to train the process reward model and 9,999 test samples to evaluate the performance of LLM agents. For HotpotQA, we analyzed its performance using 100 validation tasks from the distractor dev split of the dataset, which is also used by Expel, ReAct and Reflexion in their studies. In the case of ALFWorld, we utilized the same set of 134 solvable tasks previously used by Expel, ReAct and Reflexion. Likewise, the evaluation of the WebShop tasks involved the same 100 tasks as those used in prior studies by Expel, ReAct, and Reflexion.

Algorithm 1: MCTS-Based Automatic Labeling and PRM Training

Input: LLM agent M_{agent}, PRM model M_{prm}, Query set Q, number of MCTS simulations m, PRM training epochs E_{prm}
Output: Trained PRM model M_{prm}

1 **Phase 1: Auto-labeling with MCTS**
2 $D_{prm} \leftarrow \emptyset$
3 **for** *each query* $q_k \in Q$ **do**
4 $\quad S \leftarrow M_{\text{agent}}(\{q_k\})$
5 $\quad$ // $S = \{s_1, s_2, s_3, \ldots, s_K\}$
6 $\quad$ **for** *each step* $s_i \in S$ **do**
7 $\quad\quad correct_flag \leftarrow$ False
8 $\quad\quad$ **for** $j = 1$ ***to*** m **do**
9 $\quad\quad\quad p_j^i \leftarrow M_{\text{agent}}(\{q_k, s_1, \ldots, s_i\})$
10 $\quad\quad\quad$ **if** $A_{final}^{(j)} = A_{correct}$ **then**
11 $\quad\quad\quad\quad correct_flag \leftarrow$ True
12 $\quad\quad\quad\quad$ break
13 $\quad\quad\quad$ **end**
14 $\quad\quad$ **end**
15 $\quad\quad$ **if** $correct_flag$ **then**
16 $\quad\quad\quad$ // Label s_i as $+$
17 $\quad\quad\quad y_{s_i}^k \leftarrow +$
18 $\quad\quad$ **else**
19 $\quad\quad\quad y_{s_i}^k \leftarrow -$
20 $\quad\quad$ **end**
21 $\quad$ **end**
22 $\quad d_k = (\{q_k, s_1^k, \ldots, s_K^k\}, \{y_{s_1}^k, y_{s_2}^k, \ldots, y_{s_K}^k\})$
23 $\quad D_{prm} \leftarrow D_{prm} \cup \{d_k\}$
24 **end**
25 **Phase 2: PRM Training**
26 **for** *epoch* $e = 1$ ***to*** E_{prm} **do**
27 $\quad$ Update M_{prm} via supervised full-parameter fine-tuning on D_{prm}
28 **end**
29 **return** M_{prm}

Dataset	Training Set Size	Training Samples Used	Test Set Size	Test Samples Used
FEVER	145,449	30,000	19,998	9,999
HotpotQA	90,443	30,000	7,405	100
ALFWorld	3,553	355	134	134
WebShop	10,587	1,000	500	100

Table 4: Training and Test Set sizes for each dataset, along with sampled data for training and evaluation.

As detailed in Sec. 4.3, to assess the efficacy of the trained process reward model (PRM), we divided the 30,000 FEVER and HotpotQA training samples into a reward training set (24,000 samples, 80%) and a reward test set (6,000 samples, 20%).

B.2 Computational and Storage Resources

All our experiments were conducted on a single machine with 1TB RAM and 256-core AMD EPYC 7742 64-Core Processor @ 3.4GHz CPU. We use the NVIDIA H100 GPU with 80GB memory. The software environment settings are: Python 3.9.21, PyTorch 2.5.1 with CUDA 12.4 on Ubuntu 22.04.4 LTS. Specifically, the computational experiments were conducted using two NVIDIA H100 GPUs with 80 GB of memory each. We utilized the Llama-3.1-8B-Instruct as the base model, which possesses a total of 8 billion parameters; each model checkpoint demanded 106 GB of disk space. To improve inference efficiency, we incorporated the vLLM framework (Kwon et al., 2023), which features optimized attention computation and enhanced memory management.

The time required for processing varies across different datasets. For example, each stage of the FEVER dataset's processing pipeline requires varying amounts of time. The generation of step-wise responses phase takes about 25 minutes and 33 seconds, focusing on generating responses in a step-by-step manner for each claim. Following that, Monte Carlo Tree Search (m=3) is a more computationally intensive process, lasting approximately 5 hours and 12 minutes to evaluate all generated steps. Next, training the process reward model for $E_{prm} = 5$ epochs requires 11 hours and 25 minutes, during which the model learns to assign reward scores to step-by-step responses. For a single round of the rejection sampling process, where multiple responses are generated for each claim (n=4), the process takes 45 minutes and 15 seconds. Afterward, the reward scores for all generated responses are calculated, which takes 8 hours and 53 minutes because vllm cannot be used here to calculate the probability of every generated token, therefore the original transformer library is applied here which is much more slower than vllm. Lastly, the model undergoes supervised fine-tuning, taking 2 hours and 26 minutes to further improve its performance based on labeled data. This entire pipeline sums up to around 33 hours and 34 minutes for completion.

B.3 Hyperparameters

In our experiments, we use the vllm and the transformer library to generate step by step responses. The sampling process is controlled by a temperature of 0.9, which adjusts the randomness of the output, and a top-p value of 0.85 for nucleus sampling, determining the smallest set of tokens to consider based on cumulative probability. The model generates a maximum of 2048 tokens per output, with a repetition penalty of 1.0 to discourage repetitive phrases. Additionally, the tokenizer processes the text before it is passed to the model for generation and decodes the output back into human-readable text. All these parameters were conducted through grid search and the hyper-parameters with the best performance are selected as the final parameters. We generated three different responses for Monte Carlo Tree Search with seeds 0, 1 and 3407. For Rejection Sampling, we run five iterations, and for every iteration we generate four different responses with seeds 0, 3407, 314159, 271828, respectively.

The LLaMA-Factory library is used to fine-tune the large language models, including the process reward model and the LLM agent. In our experiments, the adopted fine-tuning approach involves full fine-tuning with the SFT (Supervised Fine-Tuning) stage. Training is conducted using DeepSpeed, which is well-suited for large-scale training environments and incorporates ZeRO-3 optimization. The maximum input sequence length is constrained to 2048 tokens. During training, the batch size per device is set to 1, and gradient accumulation is used over 2 steps to simulate a larger batch. The learning rate is set to 1.0e-5 with a cosine learning rate scheduler, a warmup ratio of 0.1, and

Algorithm 2: LLM Agent Training Process

Input: LLM agent M_{agent}, PRM model M_{prm}, Query set Q, number of candidate responses n, training iterations T, LLM Agent training epochs E_{rs}

Output: Optimized LLM agent M_{agent}

Phase 3: Rejection Sampling
for *round* $t = 1$ ***to*** T **do**
  $D_{rs} \leftarrow \emptyset$;
  for *each query* $q \in Q$ **do**
    // Generate n candidate responses
    for $j = 1$ ***to*** n **do**
      // $r_j = \{s_1, s_2, s_3, \ldots, s_K\}$
      $r_j \leftarrow M_{\text{agent}}(\{q\})$
      // Generate labels and associated probabilities for every step of current response r_j
      $\{(l_i, P_i)\}_{i=1}^{K} \leftarrow M_{\text{prm}}(r_j)$
      // Calculate the reward score R_j for current response r_j
      $R_j = \frac{1}{K}\sum_{i=1}^{K}(\mathbb{I}(l_i = +) \cdot P_i - \mathbb{I}(l_i = -) \cdot P_i)$
    end
    // Select the best response
    $r_* = \arg\max_{r_j} R_j$
    $D_{rs} \leftarrow D_{rs} \cup \{(q, r_*)\}$
  end
  for *epoch* $e = 1$ ***to*** E_{rs} **do**
    Update M_{agent} via supervised full-parameter fine-tuning on D_{rs}
  end
end
return M_{agent}

training will run for $E_{rs} = 5$ epochs and $T = 5$ iterations. Every epoch will train the model for 2250 steps. Mixed precision training with bf16 is enabled for efficiency. The training pipeline uses 16 workers for data preprocessing. For evaluation, 10% of the dataset is used for validation, with the batch size set to 1 for evaluation as well.

C License

Our implementation employs the LLaMA-3.1-8B-Instruct model (Dubey et al., 2024), a cutting-edge large language model tailored for a range of natural language processing (NLP) tasks, including question answering and text generation. This model is subject to the LLaMA 3.1 Community License Agreement. Additionally, the LLaMA-Factory, which facilitates efficient training and deployment of large language models and is likely employed for fine-tuning LLaMA models, is governed by the Apache-2.0 license (Zheng et al., 2024). The vLLM framework, designed to optimize large language models for both training and inference with a focus on efficient memory usage and computational performance, adheres to the Apache-2.0 License (Kwon et al., 2023).

Our implementation also utilizes the transformers library, a widely recognized open-source tool for NLP that offers easily accessible pre-trained models for various tasks such as text classification, question answering, and translation. This library, supporting multiple architectures including BERT, GPT, and T5, is governed by the Apache-2.0 License (Wolf et al., 2020).

Regarding datasets, the FEVER dataset (Thorne et al., 2018), utilized for fact verification where models determine the veracity of claims against provided texts, is licensed under the Apache-2.0 License. The HotpotQA dataset (Yang et al., 2018), used for multi-hop question answering that requires synthesizing information from multiple documents, operates under the Apache-2.0 License. The ALFWorld dataset (Shridhar et al., 2020), designed for training agents in virtual environments through natural language for interactive task-oriented dialogue and situated learning, is available under the MIT License. Lastly, the WebShop dataset (Yao et al., 2022a), aimed at developing conversational agents for e-commerce applications where users interact with a shopping assistant, is also available under the MIT License.

D Additional Experiments

D.1 Comparison on large LLMs

To further evaluate the scalability and generalization capability of our method beyond relatively small-scale language models, we conducted additional experiments using the Qwen-14B model, which contains significantly more parameters than LLaMA-3.1-8B. Specifically, we applied our method to two representative benchmarks: FEVER and ALFWorld.

As shown in Fig. 6, our method remains effective and even shows improved performance when applied to a larger-scale model. On the FEVER dataset, Qwen-14B outperforms LLaMA-3.1-8B-Instruct by over 3%, indicating that our method

Method	FEVER	HotpotQA	ALFWorld	WebShop
Act	59.89±0.08	21.67±1.25	19.15±0.93	27.00±1.63
ReAct	60.92±0.83	24.00±1.63	24.62±2.20	28.33±2.49
Reflexion	60.69±1.00	27.33±1.25	34.08±0.93	40.67±2.49
ExpeL	61.80±0.73	33.67±0.94	30.34±2.46	38.00±1.41
AgentPro	**65.39±0.09**	**39.33±1.25**	**40.88±1.22**	**52.67±1.70**

Table 5: The specific numerical results from the experiments shown in Fig. 3.

Table 6: Performance comparison of our method on larger-scale models, evaluated on the FEVER and ALFWorld benchmarks.

Task	Model	Accuracy (%)
FEVER	LLaMA-3.1-8B-Instruct	65.39
	Qwen-14B	**68.52**
ALFWorld	LLaMA-3.1-8B-Instruct	40.88
	Qwen-14B	**45.21**

Table 7: Success rates (%) of our method with and without the Process Reward Model (PRM) across different datasets.

Method	FEVER	HotpotQA	ALFWorld	WebShop
w/o PRM	58.38	22.65	24.62	28.33
w/ PRM	**65.39**	**39.33**	**40.88**	**52.67**

can better leverage the enhanced capacity of larger models to improve factual verification. Similarly, on ALFWorld, Qwen-14B achieves a 4.33% higher accuracy, suggesting that our approach generalizes well across both different model sizes and task types, including reasoning tasks in complex interactive environments.

D.2 Impact of Process Reward Model

To assess the contribution of the Process Reward Model (PRM) during agent training, we conduct an ablation study by completely removing the PRM component. In this setting, instead of selecting high-quality reasoning trajectories based on PRM scores, the agent is trained with randomly generated trajectories from the base model.

As shown in Table 7, removing PRM results in performance decline across all four evaluation benchmarks. These results indicate that the PRM plays a critical role in improving the quality of reasoning supervision. Notably, the performance decline occurs despite using the same training procedure and exposure to comparable in-domain examples, suggesting that the observed improvements are primarily due to the strong reward signal provided by the process-level evaluation rather than simple data exposure. This confirms that the PRM effectively guides the agent toward better intermediate reasoning steps.

Table 8: Success rates (%) of process-based supervision (PRM) vs. outcome-based supervision (ORM) across different datasets.

Supervision Type	FEVER	HotpotQA	ALFWorld	WebShop
ORM	61.02	25.45	28.36	34.92
PRM	**65.39**	**39.33**	**40.88**	**52.67**

D.3 Comparison of PRM and ORM

To validate the effectiveness of our proposed process-based supervision (PRM), we conducted comparative experiments against outcome-based supervision (ORM) across multiple standard benchmarks. As shown in Table 8, PRM consistently outperforms ORM on all evaluated datasets, demonstrating the advantage of supervising intermediate reasoning steps rather than only final outcomes. Specifically, PRM achieves higher success rates on FEVER, HotpotQA, ALFWorld, and WebShop, indicating that process supervision can lead to more robust and generalizable reasoning behavior in complex tasks.

D.4 Effect of Test-Time Computation Strategies

We further investigate the impact of various test-time computation strategies on the performance of our approach. Specifically, we assess the PRM-based model on the FEVER dataset using three widely adopted inference-time techniques: majority voting, beam search, and Best-of-N (BoN). These strategies aim to improve decision robustness by aggregating or selecting from multiple reasoning trajectories at inference time.

The results presented in Table 9 reveal that all three strategies lead to improved performance over the base model. This demonstrates that test-time enhancements can further improve the reliability and performance of process-supervised reasoning.

Table 9: Success rates (%) of PRM with different test-time computation strategies on the FEVER dataset.

Method	Success Rate (%)
Base Model	65.39
Majority Voting	66.21
Beam Search	66.89
BoN (N=5)	**67.45**

Table 10: Accuracy (%) of PRM on HotpotQA with different numbers of MCTS iterations.

No. of MCTS Iterations	1	3	5	7	9	20	50
Accuracy (%)	32	39	41	42	42	42	42

Notably, the BoN strategy (with N=5) achieves the highest success rate of 67.45%, indicating that selecting the best outcome from multiple sampled reasoning paths can be particularly effective when combined with process supervision.

D.5 Impact of MCTS Iterations and Number of Responses in Rejection Sampling

In order to assess how the performance of our model scales with additional computational resources, we conduct experiments that vary two key factors: the number of MCTS iterations and the number of generated responses used in rejection sampling. These settings directly affect inference cost and search capacity, and are crucial to understanding the scalability of our approach.

Table 10 presents the results on HotpotQA when increasing the number of MCTS iterations. While accuracy improves initially—reaching 42% with just 5 iterations—the performance plateaus thereafter, with no further gains observed at 20 or even 50 iterations.

Similarly, as shown in Table 11, increasing the number of sampled responses leads to improved accuracy in both ALFWorld and WebShop, but the benefit saturates after a certain point (8 responses for ALFWorld and 7 for WebShop). These findings suggest that beyond a moderate scale, further increases in computation yield diminishing returns, emphasizing the importance of optimizing inference strategies over merely scaling them.

D.6 Comparison with Training-Based Baselines

To assess the effectiveness of our full pipeline relative to other training-based approaches, we compare our method with two representative baselines: Copy (Ma et al., 2024) and AGILE (Peiyuan et al., 2024), both of which leverage fine-tuned language models and task-specific training. The evaluation is conducted on the FEVER and ALFWorld datasets.

Table 11: Accuracy (%) on ALFWorld and WebShop with different numbers of responses during rejection sampling.

No. of Responses	3	4	5	6	7	8	9	20	50
ALFWorld	38	40	41	42	42	43	43	43	43
WebShop	47	50	52	54	**55**	**55**	**55**	**55**	**55**

Table 12: Comparison with training-based baselines on FEVER and ALFWorld.

Method	FEVER	ALFWorld
Copy	61.52	34.21
AGILE	63.97	35.14
Ours	**65.39**	**40.88**

As summarized in Table 12, our approach consistently outperforms both baselines across tasks. The improvement is particularly pronounced on FEVER, where our method achieves 65.39% accuracy compared to 61.52% and 63.97% for Copy and AGILE respectively. These results highlight the advantage of our method in capturing factual consistency and multi-step inference.

E Experiment Details and Discussion

Table 5 presents the detailed numerical values of all the datasets depicted in Fig. 3, serving as a reference for further analysis and comparison.

The ablation studies discussed in Section 4.5 conclusively show that our method ensures robust hyperparameter selection and exhibits distinct convergence behaviors during critical processes like MCTS simulations and rejection sampling. These findings affirm the effectiveness of our method in reducing the need for manual adjustments of hyperparameters, while the automated process supervision framework improves the model's resilience to parameter fluctuations by optimizing explicit reasoning paths. Importantly, these experimental results offer practical implications for deployment: users can efficiently manage computational resources by moderately decreasing the number of MCTS simulations and rejection sampling iterations without compromising performance.

F Prompt Templates

F.1 Prompt for generating step by step responses

For the FEVER dataset, we employed the prompt illustrated in Fig. 6 to guide the LLM agent in generating detailed, step-by-step responses to the corresponding claims. Similarly, for the HotpotQA dataset, we used the prompt shown in Fig. 7 to direct the agent toward producing step-wise responses to the questions. For both the AlfWorld and WebShop datasets, we applied the prompts from the ReAct dataset to generate step-by-step, action-based outputs from the LLM agent.

F.2 Prompt for MCTS

Fig. 8 presents the prompt used for labeling each step in the Monte Carlo Tree Search. This prompt contains the original trajectories before the current step that awaits labeling.

F.3 Prompt for process reward model

Fig. 9 shows the prompt employed to train the process reward model. This prompt annotates each step in the responses generated by the LLM agent, assigning a single binary label (+ or -) to each step.

G Case Study

In this section, we present a detailed case study on the FEVER dataset to illustrate the entire process of our AgentPro framework. This includes generating step-by-step answers, utilizing Monte Carlo Tree Search to train the process reward model, and employing the rejection sampling strategy to train our LLM agent. This comprehensive demonstration aims to provide a clear and efficient understanding of our framework.

First of all, we need to generate step-by-step responses for a given claim by our pre-trained LLM agent. Take the claim *"The 84th Academy Awards' winners included Beginners"* as an example, Fig. 10 illustrates how our LLM agent M_{agent} generates a step-by-step solution for the claim: the agent analyzes the claim, verifies the ceremony year (2012), checks the film's eligibility (2011 release), and finally confirms its award (Best Supporting Actor).

Next, we need to use Monte Carlo Tree Search to label each step generated in Fig. 10, in order to create the training set required for training the process reward model. Fig. 11 showcases three MCTS simulations, with each simulation exploring alternative reasoning paths:

- **Response 1** correctly identifies *Beginners*' Best Supporting Actor and gives the **SUPPORTS** conclusion.
- **Response 2** erroneously attributes a Best Original Screenplay win but still reaches a **SUPPORTS** conclusion.
- **Response 3** mistakenly assumes that "Beginners" was solely a Best Picture nominee, which results in the conclusion of **NOT ENOUGH INFO**.

We employ the same pre-trained LLM agent as in Fig. 10 to generate all three responses. Given that the correct response to the claim is **SUPPORTS**, and 2 out of the 3 responses arrived at this conclusion, our method consequently auto-annotates this step (Step 2) as correct (+).

After labeling all four steps in Fig. 10 via MCTS, we generated a sample based on the claim for training the process reward model, as illustrated in Fig. 12. Notably, all steps in the original response (as shown in Fig. 10) received a positive label, demonstrating the effectiveness of our LLM agent in step by step reasoning on this claim. We will use the method described from Fig. 10 to Fig. 12 to generate training samples for the process reward model.

After training the process reward model, we can then employ the Rejection Sampling strategy to train our LLM agent. For instance, consider the claim "A&E is a channel from the United States" in the FEVER dataset. Initially, our LLM agent is tasked with generating multiple, step by step responses based on this claim, using the prompt as depicted in Fig. 13 with Wikipedia background information. Figures 14 and 15 present four candidate responses, among which three conclude with **SUPPORTS** and one with **NOT ENOUGH INFO**.

Subsequently, the trained PRM evaluates the four candidate responses and selects the one with the highest average probability as the sample to fine-tune our LLM agent. As depicted in Fig. 16, the PRM assigns labels to each step of the responses and calculates the associated probabilities. A higher probability signifies higher confidence in the correctness of that label. We then negate those probabilities associated with negatively labeled steps to reflect our disinclination towards incorrect steps and calculate the average of all the probabilities across the steps. This average probability quantifies the likelihood of each candidate

response being correct, which is essentially the reward value of PRM. Finally, the response with the highest average probability, in this case, the second response, is selected as the training sample for further supervised fine-tuning of our LLM agent. This method is consistently applied to each sample in the training set during rejection sampling.

The procedure shown from Figures 13 to 16 is repeated through multiple rounds until the agent can reliably generate high-quality responses appropriate to the current dataset.

Prompt for generating step by step responses (FEVER)

```
You are an assistant tasked with analyzing claims and determining their validity. Your
goal is to evaluate whether a given Claim is SUPPORTS, REFUTES, or if there is NOT ENOUGH
INFO.
Follow these guidelines strictly:
1. Carefully analyze the information provided in the Claim.
2. Think step by step and provide reasoning for your conclusion.
3. At the end of your analysis, choose one of the following outcomes:
- SUPPORTS
- REFUTES
- NOT ENOUGH INFO
The final result must follow this format:
Step 1: [Solution process for Step 1].
Step 2: [Solution process for Step 2].
...
"The final answer is: [SUPPORTS/REFUTES/NOT ENOUGH INFO]

Claim: The 84th Academy Awards' winners included Beginners.
Evaluate the claim step by step and determine its validity.
Your response must end with:
"The final answer is: [SUPPORTS/REFUTES/NOT ENOUGH INFO]"
```

Figure 6: Example prompt for generating step-by-step responses on the FEVER dataset.

Prompt for generating step by step responses (HotpotQA)

```
Solve a question answering task with steps. Please think step by step to determine the
answer to the question.
Question: [Insert your question here].
Step 1:[Provide the solution process for Step 1].
Step 2:[Provide the solution process for Step 2].
Step 3:[Provide the solution process for Step 3].
The final answer is: [Provide the final answer].
Based on the example above, solve the following question step by step. The number of
steps to solve the problem is not limited. Your response to me must end with: The final
answer is: [Provide the final answer]
Question: What nationality were the Tag team wrestlers the Haas Brothers?
```

Figure 7: Example prompt for generating step-by-step responses on the HotpotQA dataset.

Prompt for MCTS

```
You are an assistant tasked with analyzing question and determining their validity. Your
goal is to evaluate whether a given claim is SUPPORTS, REFUTES, or if there is NOT ENOUGH
INFO.
Follow these guidelines strictly:
1. Carefully analyze the information provided in the Claim.
2. Think step by step and provide reasoning for your conclusion.
3. At the end of your analysis, choose one of the following outcomes:
- SUPPORTS
- REFUTES
- NOT ENOUGH INFO
The final result must follow this format:
Step 1: [Solution process for Step 1].
Step 2: [Solution process for Step 2].
...
The final answer is: [SUPPORTS/REFUTES/NOT ENOUGH INFO]
Continue generating the steps for the claim in a step-by-step manner. Each response must
contain only the solution steps and, when available, the final answer in the specified
format. The final answer should be given as: 'The final answer is: [SUPPORTS/REFUTES/NOT
ENOUGH INFO]'. No other explanations or symbols are allowed. If the final answer has
already been reached, output the final answer directly in the specified format.
Claim: The 84th Academy Awards' winners included Beginners.
```

Figure 8: Example prompt for MCTS.

Prompt for process reward model

```
Label each step: if a step contributes to deriving the correct result, mark it as '+'. If
a step hinders the derivation of the correct result, mark it as '-'.
Claim: The 84th Academy Awards' winners included Beginners.
Step 1: Analyze the Claim: The Claim states that the 84th Academy Awards' winners
included "Beginners." This implies that "Beginners" is a film that won at least one
category at the 84th Academy Awards.
Step 2: Recall the year the 84th Academy Awards took place. The 84th Academy Awards
ceremony was held in 2012.
Step 3: Check if "Beginners" is a film released in 2011, which would make it eligible for
the 84th Academy Awards. After verifying, it is confirmed that "Beginners" is a 2011 film
directed by Mike Mills.
Step 4: Verify if "Beginners" won any awards at the 84th Academy Awards. Upon reviewing
the list of winners, it is confirmed that "Beginners" indeed won the award for Best
Supporting Actor for Christopher Plummer's performance.
The final answer is: SUPPORTS
```

Figure 9: Example prompt for training the process reward model.

Trajectory for generating a step by step response

Claim: The 84th Academy Awards' winners included Beginners.

Response:
Step 1: Analyze the Claim: The Claim states that the 84th Academy Awards' winners included "Beginners." This implies that "Beginners" is a film that won at least one category at the 84th Academy Awards.
Step 2: Recall the year the 84th Academy Awards took place. The 84th Academy Awards ceremony was held in 2012.
Step 3: Check if "Beginners" is a film released in 2011, which would make it eligible for the 84th Academy Awards. After verifying, it is confirmed that "Beginners" is a 2011 film directed by Mike Mills.
Step 4: Verify if "Beginners" won any awards at the 84th Academy Awards. Upon reviewing the list of winners, it is confirmed that "Beginners" indeed won the award for Best Supporting Actor for Christopher Plummer's performance.
The final answer is: SUPPORTS

Figure 10: Trajectory for generating a step-by-step response on a claim in the FEVER dataset.

Trajectory for MCTS

Claim and Step 1&2:
The 84th Academy Awards‘ winners included Beginners.
Step 1: Analyze the Claim: The Claim states that the 84th Academy Awards' winners included "Beginners." This implies that "Beginners" is a film that won at least one category at the 84th Academy Awards.
Step 2: Recall the year the 84th Academy Awards took place. The 84th Academy Awards ceremony was held in 2012.

First Response:
Step 3: Determine the nominees and winners of the 84th Academy Awards to verify if "Beginners" won in any category. The 84th Academy Awards took place in 2012, and "The Artist" won the award for Best Picture, Best Director, and several other categories.
Step 4: Verify if "Beginners" won any awards at the 84th Academy Awards. "The Artist" received the award for Best Picture, but "Beginners" received a win for Best Supporting Actor for Christopher Plummer.
Step 5: Since "Beginners" won a category at the 84th Academy Awards, we can conclude that the claim is accurate.
The final answer is: SUPPORTS

Second Response:
Step 3: Identify major films from 2011, the year before the 84th Academy Awards. "Beginners" is a film released in 2011 and received critical acclaim.
Step 4: Check the 84th Academy Awards winners. The 84th Academy Awards took place in 2012 and recognized the best in film for 2011. "Beginners" won the award for Best Original Screenplay.
Step 5: Determine if the claim is supported by the information. The claim that "Beginners" is a film that won at least one category at the 84th Academy Awards is supported by the fact that it won the award for Best Original Screenplay.
The final answer is: SUPPORTS

Third Response:
Step 3: Research the winners of the 84th Academy Awards. According to my knowledge, the 84th Academy Awards, held in 2012, had a number of winners but most notable were "The Artist" as the Best Picture and Michel Hazanavicius for Best Director.
Step 4: Look up the nominees for Best Picture to confirm if "Beginners" was among them. "Beginners" was indeed a nominee for Best Picture at the 84th Academy Awards but not the actual winner.
The final answer is: NOT ENOUGH INFO

Figure 11: Trajectory for MCTS simulation on the same claim in Fig. 10 with $m = 3$ simulations.

Generated sample for training the process reward model

Label each step: if a step contributes to deriving the correct result, mark it as '+'. If a step hinders the derivation of the correct result, mark it as '-'.
Claim: The 84th Academy Awards' winners included Beginners.
Step 1: Analyze the Claim: The Claim states that the 84th Academy Awards' winners included "Beginners." This implies that "Beginners" is a film that won at least one category at the 84th Academy Awards.
Step 2: Recall the year the 84th Academy Awards took place. The 84th Academy Awards ceremony was held in 2012.
Step 3: Check if "Beginners" is a film released in 2011, which would make it eligible for the 84th Academy Awards. After verifying, it is confirmed that "Beginners" is a 2011 film directed by Mike Mills.
Step 4: Verify if "Beginners" won any awards at the 84th Academy Awards. Upon reviewing the list of winners, it is confirmed that "Beginners" indeed won the award for Best Supporting Actor for Christopher Plummer's performance.
The final answer is: SUPPORTS
Output:
Step 1: +
Step 2: +
Step 3: +
Step 4: +

Figure 12: Generated sample for training the process reward model for the claim in Fig. 10.

Prompt for rejection sampling

```
You are an assistant tasked with analyzing claims and determining their validity. Your
goal is to evaluate whether a given Claim is SUPPORTS, REFUTES, or if there is NOT ENOUGH
INFO. Follow these guidelines strictly:
1. Carefully analyze the information provided in the Claim.
2. Think step by step and provide reasoning for your conclusion.
3. At the end of your analysis, choose one of the following outcomes:
- SUPPORTS
- REFUTES
- NOT ENOUGH INFO
The final result must follow this format:
Step 1: [Solution process for Step 1].
Step 2: [Solution process for Step 2].
...
The final answer is: [SUPPORTS/REFUTES/NOT ENOUGH INFO]
Claim: A&E is a channel from the United States.
Background Information about the Claim: The information about A&E is as follows: A or a
is the first letter and the first vowel letter of the Latin alphabet, used in the modern
English alphabet, and others worldwide. Its name in English is a (pronounced
/\u02c8e\u026a/ AY), plural aes. It is similar in shape to the Ancient Greek letter alpha,
from which it derives. The uppercase version consists of the two slanting sides of a
triangle, crossed in the middle by a horizontal bar. The lowercase version is often
written in one of two forms: the double-storey |a| and single-storey |\u0251|. The latter
is commonly used in handwriting and fonts based on it, especially fonts intended to be
read by children, and is also found in italic type..\n\nThe information about United
States is as follows:\nThe United States of America (USA), commonly known as the United
States (U.S.) or America, is a country primarily located in North America. It is a
federal union of 50 states and a federal capital district, Washington, D.C. The 48
contiguous states border Canada to the north and Mexico to the south, with the State of
Alaska in the northwest and the archipelagic State of Hawaii in the Pacific Ocean. The
United States also asserts sovereignty over five major island territories and various
uninhabited islands. The country has the world's third-largest land area, second-largest
exclusive economic zone, and third-largest population, exceeding 340 million. Its three
largest metropolitan areas are New York, Los Angeles, and Chicago, and its three most
populous states are California, Texas, and Florida. Paleo-Indians migrated across the
Bering land bridge more than 12,000 years ago, and formed various civilizations and
societies.
Evaluate the claim step by step and determine its validity. Your response must end with:
The final answer is: [SUPPORTS/REFUTES/NOT ENOUGH INFO]
```

Figure 13: Example prompt for the claim "A&E is a channel from the United States" during rejection sampling, where Wikipedia information is appended as background information.

Trajectory for rejection sampling (1th&2th response)

First response:
Step 1: Determine the relevance of the provided information to the claim. The provided information about A&E and the United States seems unrelated at first glance, as one pertains to the Latin alphabet and the other is about a country. However, looking closely, "A&E" can refer to "Arts & Entertainment," a global cable and satellite television network. This network is often associated with the United States.
Step 2: Check if "A&E" refers to a television network that originated or is headquartered in the United States. From general knowledge, A&E (Arts & Entertainment) is a well-known cable television network in the United States. While it has global presence, its origins and a significant portion of its content are tailored to the American audience, making it a US-centric channel.
Step 3: Analyze the definition of A&E in relation to its original context of the claim. Since the provided background information about A&E does not directly mention its association with television, it is unclear whether the claim refers to the original Latin letter or the television network. However, the lack of any information suggesting an alternative meaning of A&E in the context of the United States makes it more plausible that the claim refers to the television network, which is indeed based in the United States.
Step 4: Determine the validity of the claim. Considering the information about A&E as a television network, the claim that A&E is a channel from the United States is true. A&E (Arts & Entertainment) is indeed a television channel that originated in the United States.
The final answer is: SUPPORTS

Second response:
Step 1: Analyze the Claim: The claim is that A&E is a channel from the United States. The key components of the claim are "A&E" and "United States."
Step 2: Identify relevant information from the background: We have information about A&E, which describes it as an English alphabet letter, but it also mentions the word "A&E," which could refer to several things. We also have extensive information about the United States, describing its geography, population, and other characteristics.
Step 3: Consider the context of the acronym A&E: The provided background information about the letter "A" and "E" in the Latin alphabet does not directly relate to the context of a TV channel. However, A&E is indeed an acronym that can be expanded to mean "Arts and Entertainment." A search of the internet would show that A&E is an American pay television network that airs content related to documentary, drama, and reality TV.
Step 4: Connect the information to the claim: Since A&E refers to the Arts and Entertainment network, which is indeed based in the United States, this would support the claim that A&E is a channel from the United States.
Step 5: Evaluate the information for consistency and reliability: Based on external knowledge, the information is consistent and reliable.
The final answer is: SUPPORTS

Figure 14: Generated responses for the claim "A&E is a channel from the United States" during rejection sampling (1th&2th response).

Trajectory for rejection sampling (3th&4th response)

Third response:
Step 1: Identify the components of the claim. The claim is about A&E being a channel from the United States. The two main components are A&E (a channel) and the United States (a country).
Step 2: Analyze the information provided about A&E in the background. The given text about A&E discusses its origin, pronunciation, shape, and usage in different fonts, but does not mention its location or origin country.
Step 3: Analyze the information provided about the United States in the background. The given text about the United States describes its geographical location, size, population, and history, but does not mention any TV channels or media networks by name.
Step 4: Determine the relevance of the information. The text about A&E does not mention its location or affiliation, while the text about the United States does not mention any media networks. There is no direct information linking A&E to the United States.
Step 5: Draw a conclusion. Since there is no direct information in the provided background that links A&E to the United States, the claim is unsubstantiated by the given information.
The final answer is: NOT ENOUGH INFO

Fourth response:
Step 1: Analyze the claim "A&E is a channel from the United States." The claim is about a media channel named A&E, not the letter 'a'. Therefore, this claim is unrelated to the background information about the letter 'a' provided in the background information.
Step 2: Look for information about A&E in the background information. There is no mention of A&E in the background information.
Step 3: Search for external information about A&E. From external knowledge, A&E is known to be a channel that is a joint venture between Hearst Communications and Disney-ABC Television Group. A&E stands for Arts and Entertainment. A&E networks is a US-based American cable television network.
Step 4: Determine the origin of A&E based on the external information. Given the external information that A&E is a joint venture between US-based companies, it is plausible that A&E has its origins in the United States.
The final answer is: SUPPORTS

Figure 15: Generated responses for the claim "A&E is a channel from the United States" during rejection sampling (3th&4th response).

Trajectory for rejection sampling (scoring)

```
First response:
                Probability:
Step 1: -       0.8839759230613708
Step 2: +       0.9774795174598694
Step 3: +       0.9435433745384216
Step 4: +       0.9968554973602295

Average Probability: 0.5085

Second response:
                Probability:
Step 1: +       0.9885740280151367
Step 2: -       0.958041250705719
Step 3: +       0.9999788999557495
Step 4: +       0.9999936819076538
Step 5: +       0.9998918771743774

Average Probability: 0.6061 ✓

Third response:
                Probability:
Step 1: -       0.9888468384742737
Step 2: -       0.9226863980293274
Step 3: -       0.8883765935897827
Step 4: -       0.9999895095825195
Step 5: -       0.9771665334701538

Average Probability: -0.9552

Fourth response:
                Probability:
Step 1: -       0.9190813302993774
Step 2: -       0.9998040795326233
Step 3: +       0.9996646642684937
Step 4: +       0.9998919963836674

Average Probability: 0.0202
```

Figure 16: The scoring process of the process reward model for the four responses shown in Fig. 14 and Fig. 15. The response with the highest average probability (the second response) is selected as the training sample for subsequent supervised fine-tuning.