Finding Where the Buck Stops: An Automated Failure Attribution-Based Reflection Framework for Multi-Agent Collaboration

Xiaoqing Wang[1] **Keman Huang**[1*] **Bin Liang**[1] **Hongyu Li**[2]
Xiaoyong Du[1] **Wuqiong Pan**[2]
[1]Renmin University of China
[2]Ant Group
{wangxiaoq, keman, liangb, duyong}@ruc.edu.cn

Abstract

Multi-agent systems (MAS) powered by large language models have shown promise for complex tasks but suffer from high failure rates. Current self-reflection methods for MAS require all agents to reflect upon failure, overlooking a critical reality: failures typically stem from a specific agent leading the task astray, namely the *decisive error agent*, while others merely fulfill their regular duties. Forcing regular-behaving agents to reflect contaminates their memory with wrong insights. Hence, we propose `DoCtOR` (**D**iagn**o**se-then-Correc**t** PP**O**-enhanced **R**eflection), a novel reflection framework that enhances multi-agent collaboration. `DoCtOR` first identifies the *decisive error step* and *decisive error agent* through automated failure attribution, then employs counterfactual reasoning to generate a *corrected decisive error step*, and finally engages only the *decisive error agent* to produce targeted reflections. Experimental results show `DoCtOR` achieves 22%, 26%, and 27% improvements over initial success rates on HotPotQA, ChartQAPro, and Mind2Web datasets, outperforming Reflexion, Retroformer, and COPPER. We further establish the generalizability of our diagnose-then-correct paradigm and demonstrate that in low-resource settings, focusing reflection on reasoning steps after the *decisive error step* achieves comparable quality to reflecting on the complete failure trajectory.

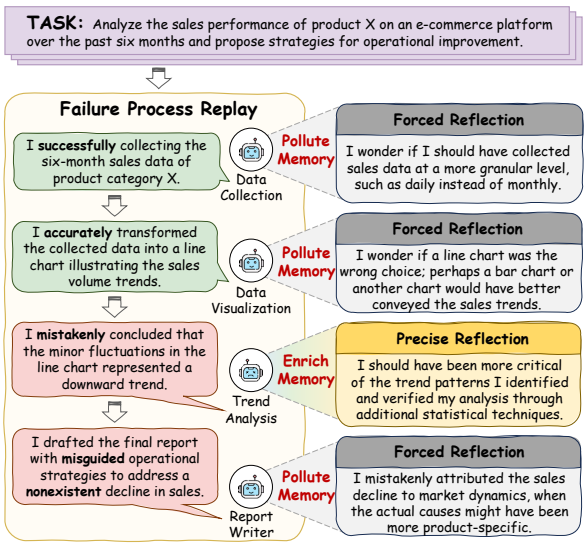

Figure 1: A multi-agent collaboration failure process replay in a data analysis task and the detrimental effects of forced reflection on regular-behaving agents.

1 Introduction

In recent years, multi-agent systems (MAS) powered by large language models have attracted considerable attention and are increasingly regarded as a promising approach for tackling complex tasks (Wu et al., 2024; Qian et al., 2024; Chen et al., 2023; Hong et al., 2024; Fourney et al., 2024). Intuitively, multiple specialized agents working collaboratively should leverage collective intelligence to overcome challenges insurmountable for single agents. However, recent empirical studies (Wang et al., 2024b; Zhang et al., 2025a; Pan et al., 2025) have revealed a perplexing phenomenon: *despite consuming additional computational resources, MAS frequently fail to reliably outperform robust single-agent baselines*. Pan et al. (2025) further documents that prominent MAS, including Chatdev (Qian et al., 2024), MetaGPT (Hong et al., 2024), Magentic-One (Fourney et al., 2024), and AppWorld (Trivedi et al., 2024), demonstrate failure rates between 60% and 86.7%, even on tasks they were specifically designed to excel at. Such alarming failure rates raise fundamental concerns regarding the practical application of MAS.

To enhance collaborative performance in MAS, a natural approach involves collecting extensive collaboration data for agent fine-tuning. Nevertheless, this strategy may compromise the model's general capabilities (Yang et al., 2024), which con-

* Keman Huang (keman@ruc.edu.cn) is the corresponding author.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 7708–7728
October 24-29, 2026 ©2026 Association for Computational Linguistics

tradicts the goal of achieving artificial general intelligence. Therefore, a more prevalent approach is to optimize the collaboration process through ***self-reflection*** mechanisms (Shinn et al., 2023; Yao et al., 2024; Bo et al., 2024), which transform binary or scalar rewards from the environment into verbal reflections, thereby providing additional context to improve task performance.

Existing literature has developed self-reflection methods for MAS, such as COPPER (Bo et al., 2024), which fine-tunes a shared reflector to generate personalized reflections according to agent roles. Specifically, COPPER requires all participating agents to reflect on task trajectories when failures occur. The underlying assumption of COPPER is that every agent shares responsibility for the failure and should reflect on its causes. However, it overlooks an important reality: *When tasks fail, responsibility often falls on a specific agent leading the task astray, namely the decisive error agent, while others merely fulfill their normal duties*.

Figure 1 illustrates this phenomenon through a multi-agent collaboration failure process replay: $Agent_A$ successfully collected the monthly sales data, while $Agent_B$ correctly generated a corresponding line graph. However, $Agent_C$ erroneously interpreted minor fluctuations as a downward trend, leading $Agent_D$ to develop inappropriate strategies based on this misanalysis. In this scenario, $Agent_C$ is the *decisive error agent*, while others are *regular-behaving agents*. ***Forced reflection can be detrimental for regular-behaving agents***: $Agent_A$ might question whether daily data would have been more appropriate, while $Agent_B$ might wonder if a bar chart would have been superior to the line graph. Such misguided reflections contaminate the memory of regular-behaving agents, providing erroneous insights for future task execution and thus probably introducing new errors, such as $Agent_A$ collecting redundant data or $Agent_B$ abandoning line chart visualization methods.

Therefore, we propose **Diagnose-then-Correct PPO-enhanced Reflection** (`DoCtOR`), a novel framework for enhancing multi-agent collaboration. The framework consists of an action module for generating reasoning trajectories and a reflection module that provides feedback and updates agent memory. Within the reflection module, we introduce a lightweight ♣ **diagnosis submodule** for ***automatic failure attribution*** (Zhang et al., 2025b), which identifies the *decisive error step* and the corresponding *decisive error agent* in multi-agent reasoning trajectories. Inspired by Process Reward Models (PRM) (Lightman et al., 2023; Zhang et al., 2025c; Song et al., 2025), we propose a **Process reward-based Automated Failure Attribution** (`ProFA`) method that assigns correctness scores to individual reasoning steps, where the first incorrect step is designated as the decisive error step and its agent as the decisive error agent. Based on the identified failure points, the ♠ **correction submodule** applies counterfactual reasoning to generate a corrected decisive error step, whose correctness is subsequently verified by `ProFA`. Finally, only the decisive error agent performs targeted reflection through the ♥ **reflector model**, enabling *self-correction* and *self-improvement*. The reflector is further fine-tuned through proximal policy optimization (Schulman et al., 2017).

Our contributions can be concluded as follows:

❶ ***Framework Proposal.*** We propose `DoCtOR`, a novel reflection framework that addresses limitations of existing multi-agent reflection methods. Unlike approaches that require all agents to reflect upon failure, `DoCtOR` first identifies the *decisive error step* through automated failure attribution, then employs counterfactual reasoning to generate the *corrected decisive error step*, and finally engages the *decisive error agent* to generate precise reflections while avoiding memory contamination of regular-behaving agents.

❷ ***Experimental Validation.*** Experiments show that ***(I)*** `DoCtOR` exhibits superior reflection capabilities, improving initial success rates by 22%, 26%, and 27% on HotPotQA, ChartQAPro, and Mind2Web datasets respectively, outperforming baselines including Reflexion, Retroformer, and COPPER; ***(II)*** `ProFA` provides effective automated failure attribution, achieving 4%-35% improvement in agent-level accuracy and 9%-28% improvement in step-level accuracy on the Who & When dataset compared to existing methods.

❸ ***Practical Solution.*** We demonstrate the generalizability of the *diagnose-then-correct* paradigm in enhancing existing prompt-based reflection methods such as Reflexion. Additionally, our analysis reveals that providing only the reasoning steps following the *decisive error step* achieves comparable reflection quality to complete trajectories, enabling efficient reflection generation under low-resource settings.

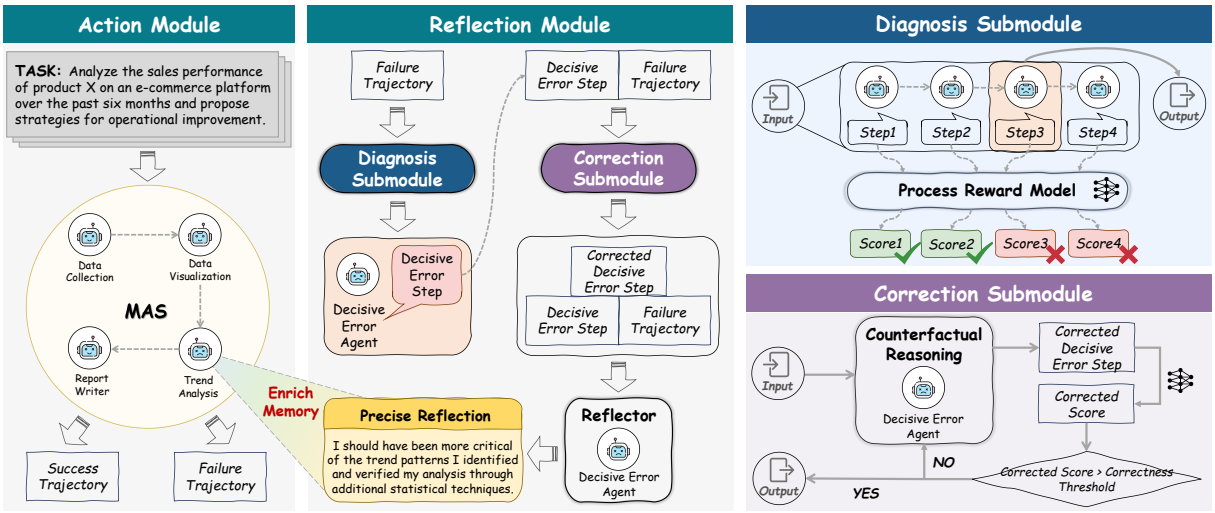

Figure 2: Overview of *DoCtOR* framework. Our framework comprises two components: an action module for generating reasoning trajectories, and a reflection module that provides feedback to action module agents and enriches their memory. The reflection module follows a three-stage workflow: (1) diagnosis submodule identifies the decisive error step and agent; (2) correction submodule generates a corrected decisive error step; and (3) reflector model leverages the previous two stages' outputs to produce targeted reflections for the decisive error agent.

2 Preliminary

2.1 Multi-Agent Collaboration

In this work, we formalize the LLM-based multi-agent system using a tuple (N, S, A, P_{ξ_o}, R), where N represents the number of agents, $S = S_1 \times S_2 \times \cdots \times S_N$ denotes the joint state space, and $A = A_1 \times A_2 \times \cdots \times A_N$ represents the joint action space. Both states S and actions A are described in natural language. At each timestep, the current state s is integrated into a context c, which may include descriptions of previous states to provide historical information, enabling LLMs to make more informed decisions about the next action to take. The state transition function $P_{\xi_o} : S \times A \to S$ governs the system dynamics, where ξ_o captures the inherent randomness in state transitions.

The multi-agent system executes tasks through sequential interactions with the environment, generating trajectories of the form $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_T, a_T\}$, where T denotes the trajectory length. In the cooperative setting, all agents share a common objective and work together to maximize collective performance. Following the standard reinforcement learning paradigm, the ultimate goal is to maximize the cumulative reward, or episode return: $G(\tau) = \sum_{t=0}^{T} R(s_t, a_t)$.

2.2 Automated Failure Attribution

To better address failures in multi-agent collaboration, we formulate the problem of automated failure attribution and introduce the concept of *decisive error step* and *decisive error agent*.

For a failed trajectory τ, we define the *decisive error step* t^* as the first time step where an agent's action a_{t^*} is identified as incorrect. This formulation is based on the observation that in multi-agent collaboration, errors often propagate sequentially. When one agent makes a mistake, subsequent agents may build upon this incorrect foundation, leading to a chain of compounding errors. Therefore, identifying and correcting the first error is more effective than attempting to fix multiple downstream mistakes that stem from the same root cause. Correspondingly, the *decisive error agent* i^* is the agent responsible for the erroneous action a_{t^*} at the *decisive error step* t^*.

3 Method

As illustrated in Figure 2, our *DoCtOR* framework consists of an action module and a reflection module. While the action module comprises multiple agents based on frozen language models (GPT-4o-mini) with inaccessible parameters, the reflection module centers on a *reflector model*, a smaller local language model (Llama-3.1-8B-Instruct) that can be fine-tuned in low-resource environments. Detailed prompts for both the action and reflection modules are provided in Appendix E.

In the remainder of this section, we describe the **reflection module** in detail. Specifally, the *reflector model* (▷ Section 3.3) leverages the decisive

error step identified by the *diagnosis submodule* (▷ Section 3.1) and the corrected decisive error step generated by the *correction submodule* (▷ Section 3.2), and the failed trajectory to generate targeted reflections for the decisive error agent.

3.1 Diagnosis Submodule

Inspired by the process reward model (Lightman et al., 2023; Zhang et al., 2025c; Song et al., 2025), we develop a ***ProFA*** method, which assigns correctness scores to each reasoning step in a multi-agent collaboration trajectory.

Given a failed trajectory τ generated by multi agents, ***ProFA*** assigns a correctness score $c_t \in [0, 1]$ to each step t in the reasoning process:

$$c_t = \textbf{\textit{ProFA}}_\phi(s_t, a_t, \tau_{<t}) \quad (1)$$

where $\textbf{\textit{ProFA}}_\phi$ is a process reward model parameterized by ϕ, and $\tau_{<t}$ represents the trajectory up to step t. Based on these scores, we formulate a binary correctness indicator B_t for each step:

$$B_t = \mathbb{I}(c_t > \gamma) \quad (2)$$

where $\mathbb{I}(\cdot)$ is the indicator function, and γ is a correctness threshold set to 0.5. The rationale for this choice and sensitivity analysis of γ are provided in Appendix D.3. Using this binary classification, we identify the *decisive error step* t^* as the earliest failure point in the trajectory τ:

$$t^* = \min\{t \mid B_t = 0, 0 \leq t \leq T\} \quad (3)$$

The ***ProFA*** model is trained on the *Who & When Pro* dataset, which consists of annotated reasoning trajectories $\mathcal{D} = \{(\tau^{(j)}, \{l_t^{(j)}\}_{t=0}^{T_j})\}_{j=1}^{M}$, where M denotes the total number of trajectories in the dataset. Each trajectory $\tau^{(j)}$ is paired with annotated binary labels $\{l_t^{(j)}\}_{t=0}^{T_j}$, indicating the correctness of every reasoning step from $t = 0$ to T_j, the length of the j-th trajectory. More details about the dataset are provided in Section 4.1. The training objective is to minimize the Binary Cross-Entropy (BCE) loss:

$$\mathcal{L}_{\textbf{\textit{ProFA}}} = -\frac{1}{M}\sum_{j=1}^{M}\sum_{t=0}^{T_j}\text{BCE}(c_t^{(j)}, l_t^{(j)}) \quad (4)$$

3.2 Correction Submodule

Once the failure points have been identified, our framework employs counterfactual reasoning (Bottou et al., 2013; Qin et al., 2019) to generate the *corrected decisive error step*, which is essential for generating informative reflections that can guide future improvements.

Given the *decisive error step* t^* and *decisive error agent* i^*, we generate a counterfactual action $\tilde{a}_{t^*}$ that represents a corrected version of the *decisive error step* action a_{t^*} :

$$\tilde{a}_{t^*} = \mathcal{CR}(s_{t^*}, \tau_{<t^*}, p_{i^*}) \quad (5)$$

where $\mathcal{CR}$ represents the counterfactual reasoning function that takes the current state, the trajectory history up to *decisive error step* t^*, and the *decisive error agent* profile p_{i^*} to produce an alternative action to correct the decisive error.

To evaluate the quality of the counterfactual action $\tilde{a}_{t^*}$ we apply ***ProFA*** to compute a new correctness score:

$$\tilde{c}_{t^*} = \textbf{\textit{ProFA}}_\phi(s_{t^*}, \tilde{a}_{t^*}, \tau_{<t^*}) \quad (6)$$

A successful correction should result in $\tilde{c}_{t^*} > \gamma$, indicating that the counterfactual action $\tilde{a}_{t^*}$ successfully addresses the error and can be considered as the *corrected decisive error step*. If $\tilde{c}_{t^*} \leq \gamma$, the correction is deemed unsuccessful, and the counterfactual reasoning process $\mathcal{CR}$ would be invoked again to generate a new alternative action until a satisfactory correction is achieved.

3.3 Reflector Model

The reflector performs targeted reflection for the *decisive error agent*, focusing on identifying the root causes of the *decisive error step* and formulating a completely different, concise, high-level plan aimed at mitigating similar failures in future tasks. To enhance the quality of generated reflections, we further fine-tune the reflector using proximal policy optimization (Schulman et al., 2017).

3.3.1 Instruction and Reflection Generation

For a given failed trajectory τ, we first use the diagnosis submodule to locate the *decisive error step* action a_{t^*}, and identify the corresponding *decisive error agent* profile p_{i^*}. After obtaining the *corrected decisive error step* action $\tilde{a}_{t^*}$ from the correction submodule, we construct the input context $x = (\tau, p_{i^*}, a_{t^*}, \tilde{a}_{t^*})$ for the reflector. The reflector $\mathcal{M}_r$ then generates a targeted reflection y_{i^*} for the *decisive error agent* i^*:

$$y^{i^*} = M_r(\tau, p_{i^*}, a_{t^*}, \tilde{a}_{t^*}) \quad (7)$$

3.3.2 Rating Score of Reflections

For a precise reflection $y_{k,e}^{i^*}$ generated by *decisive error agent* i^* in episode e of task k, let $G_{k,e}$ denote the current episode return and $G_{k,e+1}$ represent the subsequent episode return that incorporates reflection $y_{k,e}^{i^*}$. Since the actor agents are based on frozen language models with low temperature settings (Shinn et al., 2023; Yao et al., 2024; Bo et al., 2024), the injected randomness that leads to return variation ΔG is primarily attributed to the reflection $y_{k,e}^{i^*}$. Hence, we define the rating score of reflection $y_{k,e}^{i^*}$ as:

$$r(y_{k,e}^{i^*}) = G_{k,e+1} - G_{k,e} \tag{8}$$

3.3.3 Optimization of the Reflector

We optimize the reflector $\mathcal{M}_r$ through a three-stage *RLHF* (Ouyang et al., 2022) pipeline. We first collect high-quality reflections with positive scores as demonstration examples and train a supervised reflector π_{SFT} using Supervised Fine-Tuning (SFT) with standard maximum likelihood estimation:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}\left[\log \pi_\theta(y^{i^*} \mid x)\right] \tag{9}$$

Taking construction expenses into account, instead of collecting pairwise responses for each input, we train a regression model to predict the rating scores of input prompts and reflection pairs. We then optimize the reward model R_ϕ by minimizing the Mean Square Error (MSE) loss:

$$\mathcal{L}_{\text{RM}}(\phi) = \mathbb{E}\left[\left(R_\phi(x, y^{i^*}) - r\right)^2\right] \tag{10}$$

Finally, we fine-tune the supervised reflector π_{SFT} using *PPO* with the trained reward model R_ϕ. The reflector learns to generate reflections that maximize the expected reward by minimizing the following loss objective:

$$\mathcal{L}_{\text{PPO}}(\theta) = -\mathbb{E}\left[R_\phi(x, y) - \beta \log \frac{\pi_\theta^{\text{RL}}(y^{i^*} \mid x)}{\pi_{\text{SFT}}(y^{i^*} \mid x)}\right] \tag{11}$$

where β controls the strength of the KL divergence regularization term, ensuring that the fine-tuned model does not deviate too far from the reference model π_{SFT}.

4 Experiments

4.1 Datasets

DoCtOR We evaluate ***DoCtOR*** on HotPotQA (Yang et al., 2018), ChartQAPro (Masry et al., 2025), and Mind2Web (Deng et al., 2023) to evaluate the collaborative abilities of multi-agent systems in multi-hop question answering, chart data analysis, and website interaction: **(I) HotPotQA** is a multi-hop question-answering dataset comprising 113k Wikipedia-based question-answer pairs, specifically designed to assess web retrieval capabilities by requiring multiple reasoning steps to derive answers. **(II) ChartQAPro** is a comprehensive benchmark featuring 1,341 charts encompassing various chart types, including infographics and dashboards, alongside 1,948 questions spanning multiple-choice, hypothetical, and unanswerable formats. **(III) Mind2Web** is a dataset designed for evaluating execution capabilities across diverse website domains through feasible interactions like clicking, selecting, and typing, based on high-level user instructions.

ProFA To develop and evaluate ***ProFA***, we constructed a ***Who & When Pro*** dataset by extracting 1000, 500, and 1000 tasks from HotPotQA, ChartQAPro, and Mind2Web respectively, yielding 667, 379, and 711 failure logs. These logs underwent initial annotation by GPT-4o followed by thorough review by three researchers to identify *decisive error agent* (who) and *decisive error step* (when). We partitioned this dataset in a 7:1.5:1.5 ratio for training, validation, and testing. More details on the dataset construction and annotation process are outlined in Appendix B.2.2.

The ***held-in*** dataset corresponds to the test set of our constructed *Who & When Pro* dataset and contains failure logs from the same domains used during the training of ***ProFA***. To assess ***ProFA***'s generalization capabilities, we employed the *Who & When* dataset from (Zhang et al., 2025b) as our ***held-out*** dataset. This held-out test set comprises 184 failure logs from multi-agent systems with fine-grained annotations about failure-responsible agents and decisive error steps, including two subsets: *Algorithm-Generated* and *Hand-Crafted*.

4.2 Baselines

DoCtOR We compare ***DoCtOR*** with the following self-reflection baselines: **(I) Reflexion** (Shinn et al., 2023): A prompt-based reflection method that generates verbal feedback from environmental signals to enhance task performance. **(II) Retroformer** (Yao et al., 2024): Originally designed for single-agent reflection using policy gradient optimization, we extend this approach to multi-

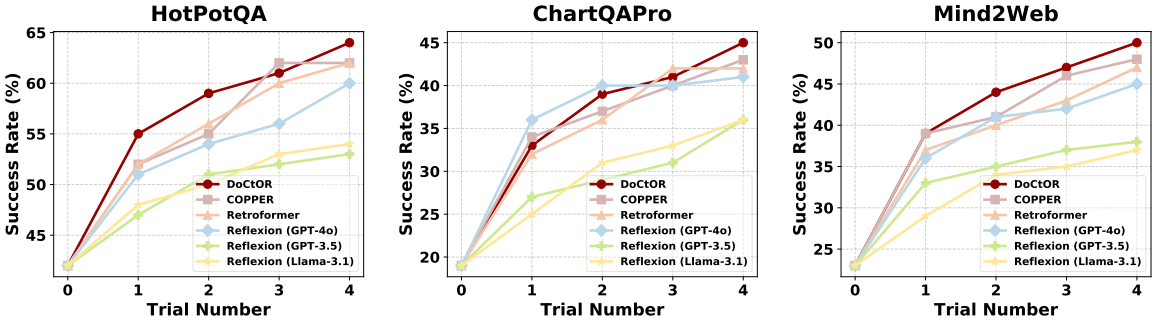

Figure 3: Performance comparison of *DoCtOR* with baselines for *self-reflection*.

agent systems by training a shared reflector with *overall rewards* as supervision signals. **(III) COPPER** (Bo et al., 2024): A multi-agent adaptation of Retroformer that trains a shared reflector using *counterfactual rewards* to assess the contribution of a single agent's reflection within the system, alleviating the credit assignment problem. Detailed model configurations and prompts for all baselines are provided in Appendix B.1 and Appendix F .

ProFA We compare *ProFA* with four automated failure attribution baselines from (Zhang et al., 2025b) based on GPT-4o model: **(I) All-at-once**: An LLM analyzes the complete failure log to simultaneously identify the *decisive error agent* and *decisive error step*. **(II) Step-by-step**: An LLM examines the failure log step-by-step, terminating upon identifying the first erroneous step as the *decisive error step* and its agent as the *decisive error agent*. **(III) Binary search**: An LLM iteratively narrows down the failure log by half until isolating a single step, identifying as the *decisive error step*. **(IV) Random**: A naive baseline that randomly selects *decisive error agent* and *decisive error step*.

4.3 Implementation Details

4.3.1 Collaboration Setting

We employ the Magnetic-One (Fourney et al., 2024) multi-agent system to execute tasks from HotPotQA, ChartQAPro, and Mind2Web. Magnetic-One is a general multi-agent system designed to solve tasks such as software engineering, data analysis, scientific research, and web navigation. More details regarding Magnetic-One's architecture, multi-agent team configurations across different datasets, and response formats are outlined in Appendix C.

4.3.2 Training

To develop *ProFA*, we conduct full-parameter fine-tuning on Qwen3-1.7B. Furthermore, we fine-tune Llama-3.1-8B-Instruct with *LoRA* (Hu et al., 2022) as the reflector and use GPT-2 as the regression reward model, following the setup in (Bo et al., 2024). More training details are provided in Appendix B.2.

4.3.3 Evaluation

To evaluate *self-reflection* methods including *DoCtOR*, we adopt the **Success Rate** metric following prior work (Shinn et al., 2023; Yao et al., 2024; Bo et al., 2024). A task is considered successful if the generated answer sufficiently matches the ground-truth answer. To quantify this match, we use the F1 score, which enables soft matching rather than exact binary comparison. Details of the F1 score computation are provided in Appendix B.3. Due to computational constraints and consistent with prior work (Shinn et al., 2023; Yao et al., 2024; Bo et al., 2024), we construct the test set by randomly sampling 100 tasks from each dataset.

For assessing *automated failure attribution* methods including *ProFA*, we employ two metrics from (Zhang et al., 2025b): **(I) Agent-Level Accuracy**, the percentage of correctly identified *decisive error agent*; and **(II) Step-Level Accuracy**, the percentage of correctly identified *decisive error step*.

4.4 Main Results

We evaluated the *DoCtOR* framework across HotPotQA, ChartQAPro, and Mind2Web datasets over five trials, as illustrated in Figure 3. Our analysis reveals that *DoCtOR* exhibited superior reflection capabilities compared to all baseline methods including Reflexion, Retroformer, and COPPER. By leveraging *ProFA* to localize the *decisive error agent* and *decisive error step*, then employing counterfactual reasoning to generate the *corrected decisive error step*, *DoCtOR* provides targeted auxiliary information that enables the *decisive error agent* to accurately identify failure causes and formu-

Baseline	Metric	Held-in Dataset			Held-out Dataset	
		HotPotQA	ChartQAPro	Mind2Web	Algorithm-Generated	Hand-Crafted
Random	**Agent-Level Accuracy**	0.37	0.29	0.54	0.28	0.20
	Step-Level Accuracy	0.06	0.09	0.02	0.18	0.04
All-at-Once	**Agent-Level Accuracy**	0.54	0.55	0.19	0.49	0.51
	Step-Level Accuracy	0.28	0.45	0.23	0.10	0.05
Step-by-Step	**Agent-Level Accuracy**	0.45	0.42	0.51	0.28	0.42
	Step-Level Accuracy	0.20	0.42	0.12	0.17	0.11
Binary Search	**Agent-Level Accuracy**	0.37	0.38	0.74	0.32	0.38
	Step-Level Accuracy	0.18	0.24	0.02	0.12	0.09
Our	**Agent-Level Accuracy**	0.85	0.82	0.79	0.53	0.55
	Step-Level Accuracy	0.53	0.78	0.42	0.38	0.20

Table 1: Performance comparison of ***ProFA*** with baselines for *automated fault attribution*.

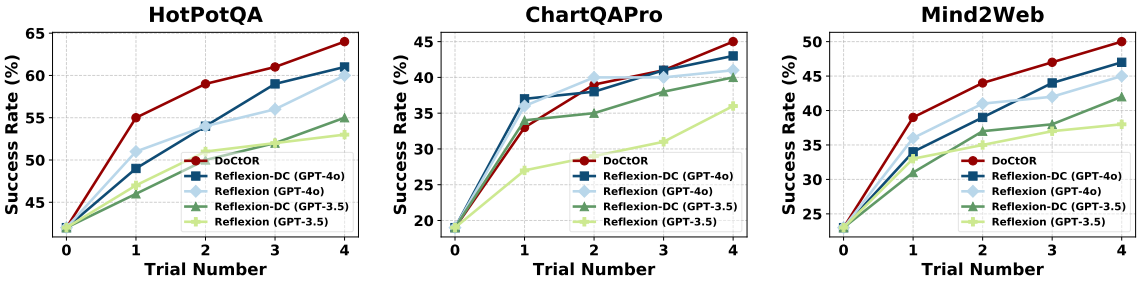

Figure 4: Generalizability of the *diagnose-then-correct* reflection paradigm. We evaluate the paradigm on the prompt-based Reflexion method to verify its generalizability beyond our fine-tuning-based implementation.

late more effective improvement strategies. Consequently, ***DoCtOR*** achieved substantial improvements of 22%, 26%, and 27% over initial success rates on HotPotQA, ChartQAPro, and Mind2Web datasets, respectively, demonstrating the effectiveness of our *diagnose-then-correct* paradigm. Additional experiments on robustness and scalability across different model families and sizes are reported in Appendix D.2.

4.5 Performance of ProFA

As shown in Table 1, ***ProFA*** demonstrated robust performance, achieving approximately 80% *agent-level accuracy* on held-in datasets and 50% on held-out datasets, while *step-level accuracy* reached 53%, 78%, and 42% on held-in datasets and 38% and 20% on held-out datasets respectively.

For *held-in datasets*, the performance improvements over baselines were particularly notable on the HotPotQA dataset, where ***ProFA*** achieved 31% improvement in agent-level accuracy and 25% in step-level accuracy compared to all-at-once, and on the ChartQAPro dataset with 27% improvement in agent-level accuracy and 33% in step-level accuracy compared to all-at-once. For *held-out datasets*, ***ProFA*** maintained its advantage with 4% improvements in agent-level accuracy over all-at-once, while step-level accuracy improved by 20% over random on Algorithm-Generated dataset and 9% over step-by-step on Hand-Crafted dataset.

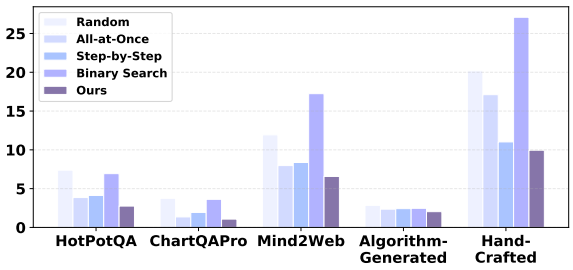

Figure 5: The absolute distance between the predicted and actual *decisive error step*.

Additionally, as illustrated in Figure 5, ***ProFA*** consistently achieved the smallest absolute distance between predicted and actual *decisive error step* across all datasets.

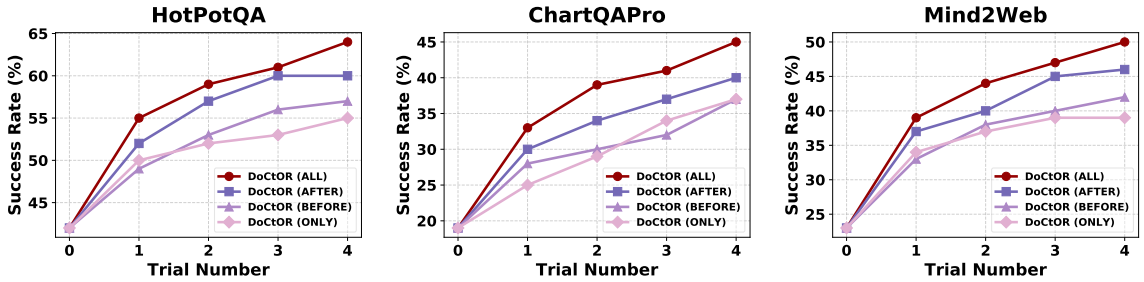

Figure 6: Effect of failure trajectory scope on reflection quality. We investigate the impact of different scopes of failed reasoning trajectories on reflection quality by comparing four strategies: the *full* failure trajectory, steps *after* the decisive error step, steps *before* the decisive error step, and *only* the decisive error step.

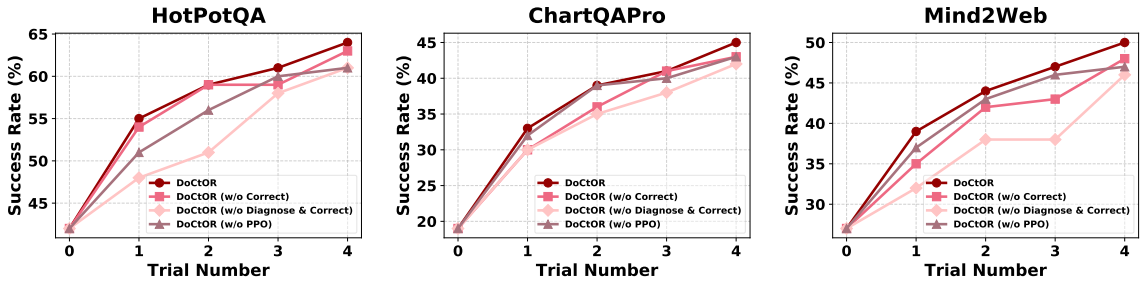

Figure 7: Ablation study. Performance comparison of `DoCtOR` with variants that remove the correction submodule (w/o correct), both diagnosis and correction submodules (w/o diagnose & correct), or proximal policy optimization for the reflector (w/o PPO).

4.6 Generalizability of Diagnose-then-Correct Reflection Paradigm

We evaluate the generalizability of our *diagnose-then-correct* paradigm by applying it to the prompt-based Reflexion method with GPT-3.5 and GPT-4o as base models, as shown in Figure 4. The *diagnose-then-correct* paradigm yields consistent but modest gains for GPT-4o-based Reflexion across all datasets, while achieving substantial improvements for GPT-3.5-based Reflexion on the ChartQAPro and Mind2Web datasets. The pronounced improvement with GPT-3.5 suggests that models with weaker intrinsic reasoning capabilities benefit more from the structured diagnostic and corrective guidance provided by our paradigm.

4.7 Effect of Failure Trajectory Scope on Reflection Quality

As shown in Figure 6, providing steps *after* the decisive error step yields comparable performance to providing a complete multi-agent trajectory. This indicates that steps *after* the decisive error step contain sufficient information for generating high-quality reflection, as the reflector can identify fundamental issues by analyzing error cascades. These insights have significant practical implications for low-resource environments, demonstrating that effective reflection can be achieved by strategically selecting reasoning steps after the decisive error step, thereby optimizing computational efficiency while maintaining reflection quality.

4.8 Ablation Study

We conduct an ablation study to assess the contribution of each component in the `DoCtOR` framework. As illustrated in Figure 7, we remove the correction submodule, both diagnosis and correction submodules, and PPO for the reflector. The results demonstrate that all components are crucial for optimal performance, as removing any of them leads to consistent performance drops across all datasets. In particular, removing both diagnosis and correction submodules causes the largest degradation, underscoring the importance of the *diagnose-then-correct* paradigm.

5 Conclusion

In this work, we introduced `DoCtOR`, a novel reflection framework that significantly enhances multi-agent collaboration. Our key contribution lies in recognizing that failures typically stem from a decisive error agent leading the task astray, while

others merely fulfill their regular duties. Through automated failure attribution and targeted reflection, *DoCtOR* prevents memory contamination for regular-behaving agents while achieving superior reflection quality, paving the way for more effective multi-agent collaboration systems.

Limitations

While the *DoCtOR* framework achieves promising results, several areas remain for future exploration. First, our current implementation focuses on task-oriented multi-agent collaborations with clear success criteria. Extending the framework to open-ended creative tasks or scenarios with subjective evaluation metrics presents an interesting direction for future research. Second, we primarily evaluated *DoCtOR* on English-language datasets and tasks. The framework's performance across multilingual settings and culturally diverse collaborative scenarios would benefit from further investigation to establish broader applicability. Finally, our experiments primarily involve relatively small multi-agent teams. Investigating the framework's scalability with larger multi-agent teams would provide valuable insights for deploying *DoCtOR* in more complex collaborative scenarios.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant Nos. 62441230, 62172425, and 62672497, the Scientific Research Innovation Capability Support Project for Young Faculty under Grant No. SRICSPYF-ZY2025001, and the Fundamental Research Funds for the Central Universities and the Research Funds of Renmin University of China under Grant No. 22XNKJ04.

References

Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. 2024. Reflective multi-agent collaboration based on large language models. *Advances in Neural Information Processing Systems*, 37:138595–138631.

Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual reasoning and learning systems: The example of computational advertising. *The Journal of Machine Learning Research*, 14(1):3207–3260.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. 2023. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, et al. 2024. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*.

Alireza Ghafarollahi and Markus J Buehler. 2025. Sciagents: automating scientific discovery through bioinspired multi-agent intelligent graph reasoning. *Advanced Materials*, 37(22):2413523.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, et al. 2025. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe.

2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Ahmed Masry, Mohammed Saidul Islam, Mahir Ahmed, Aayush Bajaj, Firoz Kabir, Aaryaman Kartha, Md Tahmid Rahman Laskar, Mizanur Rahman, Shadikur Rahman, Mehrad Shahmohammadi, Megh Thakkar, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. 2025. ChartQAPro: A more diverse and challenging benchmark for chart question answering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19123–19151, Vienna, Austria. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Melissa Z Pan, Mert Cemri, Lakshya A Agrawal, Shuyi Yang, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Kannan Ramchandran, Dan Klein, Joseph E. Gonzalez, Matei Zaharia, and Ion Stoica. 2025. Why do multiagent systems fail? In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. 2024. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186.

Lianhui Qin, Antoine Bosselut, Ari Holtzman, Chandra Bhagavatula, Elizabeth Clark, and Yejin Choi. 2019. Counterfactual story reasoning and generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5043–5053, Hong Kong, China. Association for Computational Linguistics.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

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

Kyle Swanson, Wesley Wu, Nash L Bulaong, John E Pak, and James Zou. 2024. The virtual lab: Ai agents design new sars-cov-2 nanobodies with experimental validation. *bioRxiv*, pages 2024–11.

Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. 2023. Medagents: Large language models as collaborators for zero-shot medical reasoning. *arXiv preprint arXiv:2311.10537*.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 2024. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16022–16076.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. 2024b. Rethinking the bounds of llm reasoning: Are multi-agent discussions the key? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6106–6131.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. 2024. Autogen: Enabling next-gen llm applications via multi-agent conversation. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Haoran Yang, Yumeng Zhang, Jiaqi Xu, Hongyuan Lu, Pheng-Ann Heng, and Wai Lam. 2024. Unveiling the generalization power of fine-tuned large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 884–899.

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.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh R N,

Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil L Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. 2024. Retroformer: Retrospective large language agents with policy gradient optimization. In *The Twelfth International Conference on Learning Representations*.

Hangfan Zhang, Zhiyao Cui, Xinrun Wang, Qiaosheng Zhang, Zhen Wang, Dinghao Wu, and Shuyue Hu. 2025a. If multi-agent debate is the answer, what is the question. *arXiv preprint arXiv:2502.08788*.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. 2025b. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *Forty-second International Conference on Machine Learning*.

Wenqi Zhang, Yongliang Shen, Linjuan Wu, Qiuying Peng, Jun Wang, Yueting Zhuang, and Weiming Lu. 2024. Self-contrast: Better reflection through inconsistent solving perspectives. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3602–3622.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025c. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10495–10516, Vienna, Austria. Association for Computational Linguistics.

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.

A Related Work

A.1 LLM-based Multi-Agent Systems

Recent years have witnessed substantial attention toward Large Language Model (LLM)-based agent systems within the artificial intelligence domain (Wang et al., 2024a; Zhao et al., 2024). Building upon single-agent architectures, LLM-based multi-agent systems have experienced rapid advancement, demonstrating significant progress in complex task decomposition and world simulation capabilities. The growing interest in these systems stems from their capacity to handle intricate multi-step tasks while dynamically interacting with diverse environments, making them particularly suitable for real-world applications (Li et al., 2023). Multi-agent systems have been increasingly explored across various domains, including software engineering (Qian et al., 2024; Hong et al., 2024), drug discovery (Gottweis et al., 2025; Swanson et al., 2024), scientific simulation (Tang et al., 2023; Ghafarollahi and Buehler, 2025), and general-purpose intelligence (Fourney et al., 2024; Wu et al., 2024; Chen et al., 2023; Li et al., 2023).

However, recent studies (Wang et al., 2024b; Zhang et al., 2025a; Pan et al., 2025) reveal that multi-agent deliberation methods do not consistently outperform simpler single-agent baselines, even with additional computational resources for reasoning. These findings suggest the need for optimized collaborative mechanisms to enhance multi-agent problem-solving capabilities significantly.

A.2 Self-Reflection of Large Language Models

Reflection mechanisms have gained significant attention as they employ self-reflection feedback to provide specific improvement directions for agents. These approaches enable agents to learn from previous errors, avoid recurring mistakes, and perform better in subsequent attempts. Early investigations focused on optimizing responses based on singular feedback (Madaan et al., 2023; Chen et al., 2024) or comparative analysis between multiple models (Du et al., 2024; Zhang et al., 2024), but failed to develop comprehensive task understanding from past experiences. Later work (Shinn et al., 2023) examined previous trajectories and environmental rewards to generate reflections, incorporating these insights into subsequent contexts. While this approach enables iterative enhancement, its effectiveness depends largely on the model's inherent reflective capabilities. To address this limitation, (Yao et al., 2024) proposed Retroformer, which approximates reflection rewards through differentials between consecutive outcomes and trains a reflector via policy optimization. More recently, (Bo et al., 2024) extended this methodology to multi-agent systems with COPPER, employing counterfactual rewards to evaluate reflection quality for each agent and determine whether reflections merit inclusion in agent memory.

However, existing reflection frameworks overlook a critical nuance: when tasks fail, not every agent bears responsibility for the failure. More commonly, a specific agent leads the task astray while others simply fulfill their normal duties. To address this limitation, we propose ***DoCtOR***, a novel reflection framework that enhances multi-agent collaboration through targeted diagnostic and corrective mechanisms.

B Implementations Details

B.1 Baseline Models

To ensure fair comparison, all baselines employ GPT-4o-mini for their action modules, consistent with our ***DoCtOR*** framework's action module. For the reflection modules, we adopt different model configurations based on each method's training requirements. Fine-tuning-based approaches, including Retroformer and COPPER, utilize Llama-3.1-8B-Instruct as the base model, matching the foundation model of our ***DoCtOR*** reflector. For the prompt-based Reflexion method, we evaluate performance across three base models: Llama-3.1-8B-Instruct, GPT-4o, and GPT-3.5, providing comprehensive comparison results.

B.2 Training

B.2.1 DoCtOR

Data Collection To fine-tune our reflector, we extracted 1000, 500, and 1000 tasks from each dataset respectively, with a maximum of 3 trials per task, yielding 3565 reflection instances.

Training Details We employed Low-Rank Adaptation (LoRA) for parameter-efficient fine-tuning and implemented Reinforcement Learning from Human Feedback (RLHF) through HuggingFace's trl package. Our training pipeline incorporated a three-stage approach: supervised fine-tuning, reward modeling, and policy optimization. For supervised fine-tuning, we conducted a grid search including epochs 1, 2, 3, 4, batch sizes 8, 16, 32, 64,

128, and learning rates 1e-4, 2e-4, 3e-4, 4e-4, 5e-4 using a validation set of 100 instances. The reward model was trained with a learning rate of 1e-5, 3 epochs, and a batch size of 16. For PPO training, we utilized lower learning rates 1e-5, 2e-5, 3e-5, 4e-5, 5e-5 to ensure stable convergence. All models were fine-tuned using 4-bit quantized LoRA adapters, resulting in efficient parameter utilization (0.015%-0.06% of base model parameters).

B.2.2 ProFA

Data Collection To develop and evaluate ***ProFA***, we constructed a ***Who & When Pro*** dataset through a systematic data collection and annotation process. We extracted 1000, 500, and 1000 tasks from HotPotQA, ChartQAPro, and Mind2Web, respectively, yielding 667, 379, and 711 failure logs from unsuccessful multi-agent collaboration attempts. The annotation process involved a two-stage approach to ensure data quality.

First, we conducted initial automated annotation using GPT-4o to provide preliminary identification of *decisive error agents* (who) and *decisive error steps* (when) within each failure log. Subsequently, three experienced researchers independently reviewed and refined these annotations. Each researcher examined the multi-agent interaction traces, analyzing causal relationships between agent actions and task failures to determine the precise moment when the decisive error occurred and which agent was responsible. The three researchers achieved a Fleiss' kappa score of 0.83 during the annotation process, indicating substantial agreement, which falls into the "almost perfect agreement" range (0.81-1.00) on the standard Fleiss' kappa interpretation scale.

Training Details We fine-tuned a Qwen3-1.7B model using a token classification approach. We employed the PRMTrainer from *trl* with the following hyperparameters: learning rate of 5e-5, weight decay of 0.01, batch size of 8, maximum sequence length of 2048, and 3 training epochs. We implemented a warmup ratio of 0.1 with gradient accumulation steps set to 1.

B.3 Evaluation

Our framework employs the F1 score to evaluate generated answer quality by balancing precision and recall. The calculation process involves normalizing both prediction and ground truth responses. For categorical responses (yes, no, or no answer), the function assigns zero metrics when predictions do not match the ground truth. For standard text responses, the algorithm tokenizes the normalized texts and identifies overlapping tokens. Precision is calculated as the ratio of common tokens to prediction tokens, while recall represents the ratio of common tokens to ground truth tokens. The F1 score is computed as:

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{12}$$

The Mind2Web benchmark, while ultimately evaluated on task-level execution correctness, presents challenges analogous to structured question-answering tasks, requiring agents to generate web navigation instructions in a specific format (e.g., "Answer: A/B/C/D Action:CLICK/SELECT/TYPE Value:XXX"). We selected the F1 score as our reward function because it effectively balances precision and recall in structured outputs, assessing both the accuracy of included elements and the comprehensiveness of responses. This approach provides more fine-grained feedback than binary task-level execution correctness metrics alone. Our preliminary experiments confirm this benefit, showing approximately 10.2% performance improvement when using F1-based rewards compared to binary task-level execution correctness signals.

B.4 Reproducibility

All experiments were conducted on two NVIDIA A100-80G GPUs. To ensure experimental rigor, we configured different temperature settings for the base models in each module. For the action module's base model, we set the temperature to 0 and top-p to 1 following prior work (Shinn et al., 2023; Yao et al., 2024; Bo et al., 2024), thereby isolating the randomness inherent in language model generation from the effects of reflections. In contrast, for the reflection module's base model, we employed a temperature of 0.8 to encourage more creative and diverse reflection generation.

C Collaboration Settings

C.1 Magentic-One Multi-Agent Framework

Our implementation utilizes the Magentic-One framework, which employs a hierarchical multi-agent structure for complex task solving. The system comprises several specialized agents working in coordinated collaboration:

- **Orchestrator**: Functions as the central coordinator responsible for task decomposition, strategic planning, and adaptive management of the agent workflow. The Orchestrator monitors execution progress and delegates subtasks to specialized agents.

- **WebSurfer**: Specializes in web-based information retrieval and browser interaction. This agent navigates URLs, performs searches, interacts with webpage elements (clicking, typing), and extracts relevant content through the browser's accessibility tree and set-of-marks prompting techniques.

- **Coder**: Focuses on code generation, data analysis, and artifact creation. This agent processes information collected by other agents, implements algorithmic solutions, and generates computational outputs required for task completion.

- **ComputerTerminal**: Provides system-level access for executing commands, running programs, and installing necessary libraries. This agent enables the framework to interact with the operating environment and execute computational processes.

C.2 Dataset-Specific Collaboration Settings

We tailored our multi-agent configurations to address the unique requirements of three datasets:

- **HotPotQA**: For multi-hop question answering tasks, we utilized Magentic's predefined agent ensemble consisting of the Orchestrator, WebSurfer, FileSurfer, Coder, and Terminal agents. This configuration effectively supports the information retrieval and reasoning chains necessary for complex question answering.

- **ChartQAPro**: To address chart-based analytical queries, we extended the base configuration by implementing a custom ImageAgent with specialized capabilities for visual content interpretation. We developed two complementary tools for image question answering and visual description generation. The resulting team composition (Orchestrator, ImageAgent, Coder, and Terminal) enables comprehensive chart data analysis.

- **Mind2Web**: For website interaction tasks, we developed a specialized WebAgent defined as a helpful assistant with expertise in website design, navigation, and task execution. This agent works in conjunction with the Orchestrator, Coder, and Terminal agents to navigate complex web interfaces and execute multi-step interactions.

C.3 Response Format

To facilitate systematic evaluation and enhance analytical consistency across datasets, we implemented standardized response formats. For HotPotQA and ChartQAPro datasets, we developed a structured answer template, as illustrated in Figure 9. For the Mind2Web dataset, we established a more structured format that normalizes responses into a combination of Answer, Action, and Value components, also depicted in Figure 8.

Final answer prompt for Mind2Web

We are working on the following task: {task}
We have completed the task. The above messages contain the conversation that took place to complete the task. Based on the information gathered, provide the final answer to the original request.

The answer should be prefixed with <FINAL> and suffixed with </FINAL>. Do not include any other text after the </FINAL> tag.

Follow these rules strictly:
(1) If the correct action is not in the choice list , please select A. 'None of the above',
and the final answer should be \"<FINAL> Answer: A. </FINAL>\".

(2) If you want to choose CLICK operation, the final answer should be \"<FINAL> Answer: \{letter\}.\nAction: CLICK </FINAL>\".
Example: <FINAL> Answer: D.\nAction: CLICK </FINAL>

(3) If you want to choose SELECT operation, the final answer should be \"<FINAL> Answer: \{letter\}.\nAction: SELECT\nValue: {selected_value}} </FINAL>\".
Example: <FINAL> Answer: B.\nAction: SELECT\nValue: New Passport Only </FINAL>

(4) If you want to choose TYPE operation, the final answer should be \"<FINAL> Answer: \{letter\}.\nAction: TYPE\nValue: {input_text}} </FINAL>\".
Example: <FINAL> Answer: D.\nAction: TYPE\nValue: 60505 </FINAL>

Figure 8: The final answer prompt for Mind2Web.

Final answer prompt for HotPotQA & ChartQAPro

We are working on the following task:
{task}We have completed the task. The above messages contain the conversation that took place to complete the task. Based on the information gathered, provide the final answer to the original request.

The answer should:
1.directly address the original request in the shortest possible form (preferably a word or a phrase, not a full sentence).
2.be prefixed with <FINAL> and suffixed with </FINAL>, e.g., \"<FINAL> PowerPoint </FINAL>\".

Do not include any other text after the </FINAL> tag.

Figure 9: The final answer prompt for HotPotQA & ChartQAPro .

D Additional Experiments

D.1 Effect of Test Set Size

To comprehensively evaluate the scalability and consistency of our method, we conducted extensive experiments across different test set sizes on the HotPotQA benchmark.

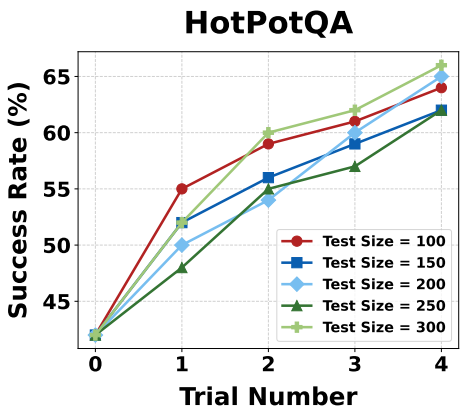

Figure 10: Performance on HotPotQA under different Test Set Sizes.

As shown in Figure 10, our method consistently improves performance across all test set sizes throughout the trials. The stability of this improvement across varying scales indicates that the method's effectiveness is robust and generalizes well to larger evaluation sets.

D.2 Robustness Across Model Families and Sizes

To evaluate scalability and robustness, we tested our framework with different model families and sizes.

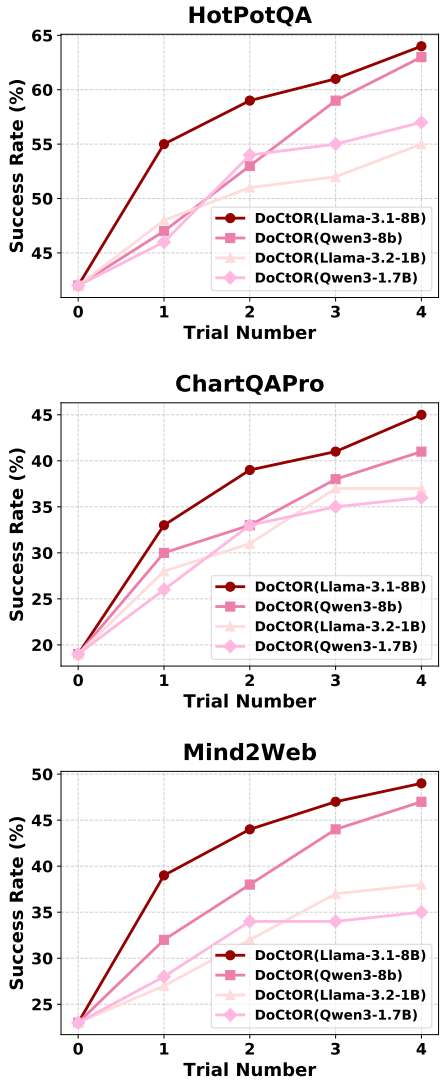

Figure 11: Robustness Across Model Families and Sizes.

As illustrated in Figure 11, the results first validate our choice of Llama-3.1-8B while also demonstrating that our framework consistently improves performance across diverse model families and sizes, with larger models (8B) generally achieving better results than smaller ones (1-2B).

D.3 Effect of Correctness Threshold γ

The parameter γ is set to 0.5 based on standard practices in PRM implementations within widely-used frameworks like TRL and OpenRLHF, which

use 0.5 as the threshold for evaluating step correctness.

Furthermore, we conducted extensive experiments to examine how the correctness threshold γ affects final reflection performance.

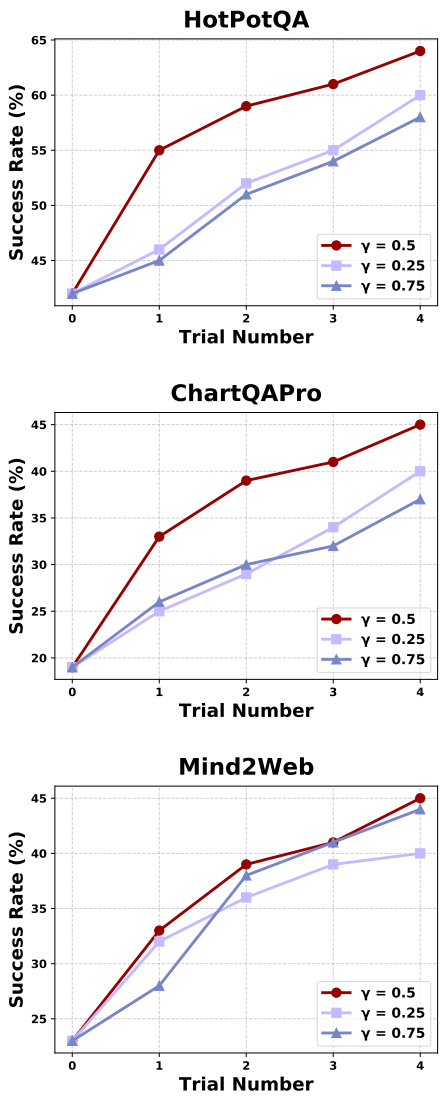

Figure 12: Effect of Correctness Threshold γ.

As shown in Figure 12, the results demonstrate that while $\gamma = 0.5$ generally yields the best performance across datasets, our method shows robust improvement patterns regardless of the specific threshold chosen.

E Prompts when Deploying DoCtOR

The action module prompt for HotPotQA

Please answer the question below to the best of your ability. This task may require retrieving information from external web pages.

Carefully analyze any provided links or references, extract relevant information, and formulate a complete and accurate response.

If you need to synthesize information from multiple sources, ensure your answer remains coherent and directly addresses the question posed.

Question: <question>

Below is the decisive error step during your previous attempt to answer this question, along with reflections from the relevant agents on why this step led to an incorrect final answer, as well as reflections on why the final answer was incorrect.
Please carefully review these reflections and attempt to answer the question again.

Decisive error step:
{decisive_error_step}

Reflections:
{reflections}

Figure 13: The action module prompt for HotPotQA when deploying *DoCtOR*.

The action module prompt for Mind2Web

You are a helpful assistant that is great at website design, navigation, and executing tasks for the user.

{final_html}
Based on the HTML webpage above, try to complete the following task:
Task: {confirmed_task}

Previous actions:
{previous_actions}

What should be the next action? Please select from the following choices
(If the correct action is not in the page above, please select A. 'None of the above'):
{choices_text}

Below is the decisive error step during your previous attempt to answer this question, along with reflections from the relevant agents on why this step led to an incorrect final answer, as well as reflections on why the final answer was incorrect.
Please carefully review these reflections and attempt to answer the question again.

Decisive error step:
{decisive_error_step}

Reflections:
{reflections}

Figure 14: The action module prompt for Mind2Web when deploying *DoCtOR*.

The action module prompt for ChartQAPro

You are given a factoid question that you need to answer based on the provided image. Your answer should be a single word, number, or phrase.

If the question is unanswerable based on the information in the provided image, your answer should be unanswerable.

Do not generate units. But if numerical units such as million, m, billion, B, or K are required, use the exact notation shown in the chart. If there are multiple answers, put them in brackets using this format ['Answer1', 'Answer2']. Remember to generate the final answer only without any additional text!

Question: <question> Image Path: <image>

Below is the decisive error step during your previous attempt to answer this question, along with reflections from the relevant agents on why this step led to an incorrect final answer, as well as reflections on why the final answer was incorrect.
Please carefully review these reflections and attempt to answer the question again.

Decisive error step:
{decisive_error_step}

Reflections:
{reflections}

Figure 15: The action module prompt for ChartQAPro when deploying *DoCtOR*.

The counterfactual reasoning prompt

You are a member of a multi-agent team tasked with answering questions.
In prior trials, your team was unsuccessful in answering the question: {prompt}.
{decisive_error_step} is the decisive error step that directly leads to the final wrong answer.
Here is the dialogue history before the decisive error step:
{previous_steps}

Please employ counterfactual reasoning to analyze: if you had taken an alternative action at that decisive error step, would the task have succeeded?

If you can identify such a counterfactual action that would lead to task success, please provide this alternative action.

Requirements:
1) Maintain the exact original format including step number and agent name (e.g., "[Step X] AgentName:")
2) Only modify the problematic parts while keeping the rest of the structure identical
3) Ensure the new step would actually solve the current issue
4) Output ONLY the complete corrected step with no additional explanation

Figure 16: The counterfactual reasoning prompt when deploying *DoCtOR*.

The reflector prompt

{agent_profile}
You are also an advanced reasoning agent that can improve based on self refection.

In prior trials, the team was unsuccessful in answering the question: {question}.
The final answers provided by the team are enclosed with the markers <FINAL> and </FINAL>.
If <FINAL> and </FINAL> markers are found, potential failure reasons include but are not limited to:
1. Misunderstanding the task objective
2. Lack of necessary background knowledge
3. Providing a final answer without thorough validation

If no <FINAL> and </FINAL> markers are found, it indicates that the task was prematurely terminated. Potential reasons for this include but are not limited to:
1. Reaching the predefined maximum number of dialogue turns
2. Inefficiency due to repetitive task handling
3. Restricted access to files/webpages
4. Network interruptions

Here is the multi-agent dialogue history:
{conversation_history}

{decisive_error_step} was the decisive error step that directly led to the final incorrect answer, and you are the agent directly associated with this step.
This is the correct version of decisive error step :
{correct_decisive_error_step}

Now, you need to carefully analyze the multi-agent dialogue history, reflecting on why you produced this decisive error step, first explaining in one sentence why this decisive error step caused the final answer to be incorrect, then explaining in one sentence why the final answer provided by our team was not correct for this question, and finally proposing a completely different, concise, high-level plan in one sentence.

Note: If you are pretty sure that the final answer is correct, it may be that the answer can be expressed in multiple formats.
For example, a birthdate can be rephrased from year-month-day to month-day; similarly, a person's name can be rephrased from first and last name to first, middle, and last name.
Make a plan to rephrase your answer.
The rephrased answer should still directly address the original request in the shortest possible form (preferably a word or a phrase, not a full sentence).

Figure 17: The reflector prompt when deploying *DoCtOR*.

F Prompts when Deploying Reflexion, Retroformer, and COPPER

The action module prompt for HotPotQA

Please answer the question below to the best of your ability. This task may require retrieving information from external web pages.

Carefully analyze any provided links or references, extract relevant information, and formulate a complete and accurate response.

If you need to synthesize information from multiple sources, ensure your answer remains coherent and directly addresses the question posed.

Question: <question>

Here are the reflections on why your previous attempt to answer this question was incorrect; carefully review these reflections and try to answer the question again:
{reflections}

Figure 18: The action module prompt for HotPotQA when deploying Reflexion, Retroformer, and COPPER.

The action module prompt for ChartQAPro

You are given a factoid question that you need to answer based on the provided image. Your answer should be a single word, number, or phrase.

If the question is unanswerable based on the information in the provided image, your answer should be unanswerable.

Do not generate units. But if numerical units such as million, m, billion, B, or K are required, use the exact notation shown in the chart. If there are multiple answers, put them in brackets using this format ['Answer1', 'Answer2']. Remember to generate the final answer only without any additional text!

Question: <question> Image Path: <image>

Here are the reflections on why your previous attempt to answer this question was incorrect; carefully review these reflections and try to answer the question again:
{reflections}

Figure 19: The action module prompt for ChartQAPro when deploying Reflexion, Retroformer, and COPPER.

The action module prompt for Mind2Web

You are a helpful assistant that is great at website design, navigation, and executing tasks for the user.

{final_html}
Based on the HTML webpage above, try to complete the following task:
Task: {confirmed_task}

Previous actions:
{previous_actions}

What should be the next action? Please select from the following choices
(If the correct action is not in the page above, please select A. 'None of the above'):
{choices_text}

Here are the reflections on why your previous attempt to answer this question was incorrect; carefully review these reflections and try to answer the question again:
{reflections}

Figure 20: The action module prompt for Mind2Web when deploying Reflexion, Retroformer, and COPPER.

The reflector prompt

You are a member of a multi-agent team tasked with answering questions. You specialize in identifying the causes of the team's failures and enhancing the team's problem-solving capabilities through reflection.
In prior trials, the team was unsuccessful in answering the question: {question}.

Here is the multi-agent dialogue history:
{conversation_history}

The final answers provided by the team are enclosed with the markers <FINAL> and </FINAL>.
If <FINAL> and </FINAL> markers are found, potential failure reasons include but are not limited to:
1. Misunderstanding the task objective
2. Lack of necessary background knowledge
3. Providing a final answer without thorough validation

If no <FINAL> and </FINAL> markers are found, it indicates that the task was prematurely terminated. Potential reasons for this include but are not limited to:
1. Reaching the predefined maximum number of dialogue turns
2. Inefficiency due to repetitive task handling
3. Restricted access to files/webpages
4. Network interruptions

Now, you need to start by explaining in one sentence why the final answer provided by our team was not correct for this question, and then offer a completely different, concise, high-level plan in one sentence.

Note: If you are pretty sure that the final answer is correct, it may be that the answer can be expressed in multiple formats.
For example, a birthdate can be rephrased from year-month-day to month-day; similarly, a person's name can be rephrased from first and last name to first, middle, and last name.
Make a plan to rephrase your answer.
The rephrased answer should still directly address the original request in the shortest possible form (preferably a word or a phrase, not a full sentence).

Figure 21: The reflector prompt when deploying Reflexion, Retroformer, and COPPER.