# SAMULE: Self-Learning Agents Enhanced by Multi-level Reflection

**Yubin Ge**[1], **Salvatore Romeo**[1], **Jason Cai**[1], **Monica Sunkara**[1], **Yi Zhang**[1]
[1]AWS AI Labs
`{yubinge, romeosr, cjinglun, sunkaral, yizhngn}@amazon.com`

## Abstract

Despite the rapid advancements in LLM agents, they still face the challenge of generating meaningful reflections due to inadequate error analysis and a reliance on rare successful trajectories, especially in complex tasks. In this work, we propose SAMULE, a new framework for self-learning agents powered by a retrospective language model that is trained based on Multi-Level Reflection Synthesis. It first synthesizes high-quality reflections across three complementary levels: Single-Trajectory Learning (micro-level) for detailed error correction; Intra-Task Learning (meso-level) to build error taxonomies across multiple trials of the same task, and Inter-Task Learning (macro-level) to extract transferable insights based on same typed errors from diverse task failures. Then we fine-tune a language model serving as the retrospective model to generate reflections during inference. We further extend our framework to interactive settings through a foresight-based reflection mechanism, enabling agents to proactively reflect and adapt during user interactions by comparing predicted and actual responses. Extensive experiments on three challenging benchmarks—TravelPlanner, NATURAL PLAN, and Tau-bench—demonstrate that our approach significantly outperforms reflection-based baselines. Our results highlight the critical role of well-designed reflection synthesis and failure-centric learning in building self-improving LLM agents.

## 1 Introduction

Modern AI agents (Yao et al., 2023; Fourney et al., 2024; Zhao et al., 2024a; Wang et al., 2024a) increasingly rely on Large Language Models (LLMs) as their core reasoning engines, empowering them to understand natural language instructions, reason through multi-step processes, and interact with external environments (Ge et al., 2025; Shen et al., 2025). These models typically undergo pre-training on extensive text corpora, endowing them with unprecedented accuracy in predicting the next token given some context (Ge et al., 2023). However, LLM agents remain fundamentally limited in their ability to autonomously improve from experience, particularly in complex, failure-prone environments (Ji et al., 2024; Wang et al., 2024c). This bottleneck severely restricts their utility in scenarios where learning from failures is critical for long-term success.

Existing experiential learning methods for LLM agents suffer from several core limitations. First, they sometimes fail to generate meaningful and actionable reflections after failures due to inadequate error analysis mechanisms. For example, Reflexion (Shinn et al., 2023) shows limited improvements on complex benchmarks like TravelPlanner (Xie et al., 2024), as it lacks the capacity to deeply diagnose failure causes and therefore produces general and useless strategies for correction.

Second, many sophisticated prompting-based methods, such as Expel (Zhao et al., 2024a), depend heavily on successful trajectories as learning signals and demonstrate the benefits of utilizing knowledge (Ge et al., 2021). This reliance on successful trails makes them impractical in real-world settings where task success is rare and failures are far more prevalent. By failing to harness the rich information embedded in unsuccessful attempts, these methods miss critical opportunities for learning and adaptation, leading to poor generalization in complex environments.

Besides, advanced approaches like Retroformer (Yao et al., 2024b) and CTRL (Xie et al., 2025), which use reinforcement learning (RL) and rewards based on task improvements from synthesized reflections, are sensitive to the quality of the synthesized reflections. In complex tasks where existing reflection algorithms struggle to produce informative and accurate feedback, these RL-based methods suffer from learning meaningless policies.

In this work, we propose SAMULE, a framework

*Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 16591–16610
November 4-9, 2025 ©2025 Association for Computational Linguistics

for **S**elf-learning **A**gents enhanced by **MU**lti-**LE**vel reflections that unlocks the learning potential of past trajectories through training a retrospective model. Specifically, we first design a Multi-Level Reflection Synthesis to synthesize high-quality reflection data across three complementary levels of granularity: **Single-Trajectory Learning (Micro-Level)**: Analyzing individual failed trajectories against reference plans to identify immediate errors and generate targeted corrective strategies; **Intra-Task Learning (Meso-Level)**: Examining multiple trajectories from the same task query to categorize error types and build an error taxonomy, enabling richer, pattern-based feedback; **Inter-Task Learning (Macro-Level)**: Clustering similar errors across diverse task queries to derive high-level, transferable insights that improve future decision-making across tasks. After merge the reflections from different levels as the final target reflection, we then train a language model through SFT, which dynamically generates trajectory-specific reflections for the agent during inference.

We further extend to the interactive setting, where agents decide whether and when to reflect during interactions with users. We introduce foresight-based reflection that compares the agent's predicted user response with the actual response at each turn. When the true response is beyond expectation, the agent triggers a reflection step and adds the generated feedback into its ongoing interaction, enabling real-time correction and adaptation.

We extensively evaluate our approach across three challenging benchmarks: TravelPlanner (Xie et al., 2024), NATURAL PLAN (Zheng et al., 2024), and Tau-bench (Yao et al., 2024a). Experimental results consistently demonstrate that our method substantially outperforms existing reflection-based baselines, particularly in complex, failure-dense environments. Notably, we show that even with simple supervised fine-tuning, our retrospective model—trained on multi-level synthesized reflections—achieves superior performance compared to advanced methods relying on RL. These findings underscore the critical role of well-designed reflection synthesis in efficient learning from past trajectories without the need for costly or unstable RL training. Moreover, the strong results achieved in interactive settings highlight the generalizability of our framework and its ability to support real-time adaptive learning. By leveraging failures as rich learning opportunities and introducing structured reflection at multiple levels, our work represents a significant step toward building more resilient, adaptive, and self-improving agents.

Our contributions are summarized as follows:

- We propose SAMULE, a new self-learning framework that introduces to training a retrospective model based on Multi-Level Reflection Synthesis spanning from micro-level to macro-level analysis.
- We introduce foresight-based reflection to extend our framework to interactive scenarios, enabling agents to proactively reflect and adapt during user interactions by comparing predicted and actual user responses.
- Extensive experiments on TravelPlanner, NATURAL PLAN, and Tau-bench show that our approach significantly outperforms existing reflection-based baselines, particularly in complex benchmarks, highlighting the value of failure-centric learning and multi-level reflection for building self-improving agents.

## 2 Related Work

**Prompt-based self-reflection and self-learning.** One foundational self-reflective capabilities for Gen AI agents was established by Shinn et al. (2023), who introduced Reflexion, a framework enabling language agents to learn from verbal reinforcement through self-reflection. Building on this foundation, research has made significant advances in developing sophisticated self-reflective capabilities through various prompting strategies (Liu et al., 2024; Renze and Guven, 2024; Radha et al., 2024; Zhang et al., 2024). Liu et al. (2024) demonstrated that self-reflection can enhance model safety and reduce bias while maintaining ideological neutrality. Further validation came from Renze and Guven (2024), who documented substantial improvements in problem-solving performance through structured reflective processes. To address the challenge of reflection stability, Zhang et al. (2024) developed Self-Contrast, introducing a novel approach that explores and contrasts diverse solving perspectives. The field has rapidly expanded into experiential learning, with Zhao et al. (2024b) introducing EXPEL for experiential learning in LLM agents with the support of autonomous experience extraction from training tasks, while Gao et al. (2024) advanced the state-of-the-art with a self-evolving framework enabling life-long experiential learning.

Recent work has also explored novel directions, including Qian et al. (2024)'s co-learning framework for software development, Radha et al. (2024)'s Inner Dialogue framework for autonomous reasoning, and Yin et al. (2024)'s self-referential framework for recursive self-improvement.

**Post-training powered by self-reflection.** The integration of self-reflection into post-training optimization has emerged as a powerful approach for enhancing LLM capabilities (Yao et al., 2024b; Feng et al., 2025; Qi et al., 2024; Xiong et al., 2025; Tu et al., 2024; Alabdulkarim et al., 2022). Yao et al. (2024b) pioneered this direction with Retroformer, introducing policy gradient optimization for retrospective learning in language agents. Feng et al. (2025) designed a sophisticated two-stage optimization process combining imitation learning with reinforcement learning, which enhances the data efficiency and training stability. For web-based applications, Qi et al. (2024) developed WebRL, demonstrating how self-evolving online curriculum learning can substantially improve open-source LLM performance. Another promising direction has emerged in self-rewarding mechanisms, exemplified by Xiong et al. (2025)'s work on mathematical reasoning, where models simultaneously generate and evaluate their reasoning without external feedback. The frontier of autonomous self-improvement has been further pushed by Chen et al. (2025), who established a comprehensive framework for scaling autonomous agents through automatic reward modeling. Additional advances include Tu et al. (2024)'s online preference-based reinforcement learning approach and Alabdulkarim et al. (2022)'s work on experiential explanations for reinforcement learning, both contributing to more robust self-reflection mechanisms.

# 3 Methodology

## 3.1 Problem Setup

### 3.1.1 General Setting

We consider a setting where an agent performs complex tasks over a sequence of time steps. Following a standard machine learning setup, we assume access to training data $\mathcal{D}_{\text{train}} = \{(x_{\text{train}}^{(i)}, y_{\text{train}}^{(i)})\}_{i=1}^{N}$, which allows the agent to learn via task exploration. Each input $x_{\text{train}}^{(i)}$ includes an instruction, a task query, and relevant background information, while the corresponding output $y_{\text{train}}^{(i)}$ represents a valid response, such as a human-annotated plan to the input query. We consider $y_{\text{train}}^{(i)}$ as a reference. Particularly, in agentic task completion, $y_{\text{train}}^{(i)}$ does not need to be the unique ground truth but is considered a valid solution to $x_{\text{train}}^{(i)}$. During inference, the agent receives a test input $x_{\text{test}}^{(i)}$ and generates a predicted output $\hat{y}_{\text{test}}^{(i)}$ for evaluation.

### 3.1.2 Reflection Setting

We explore two settings for reflection:

**Non-Interactive**: We follow the self-reflection setup from Reflexion (Shinn et al., 2023), where the agent reflects only after completing a trial. If the trial fails, the corresponding trajectory is reviewed to generate a reflection. This reflection is then appended to the input for the next trial.

**Interactive**: In real-world scenarios, agents often interact with users through multi-turn conversations to complete tasks. In this setting, the agent may not be able to try the task multiple times and therefore cannot defer reflection until after the task concludes. Instead, it must dynamically decide whether to reflect during the interaction.

## 3.2 SAMULE

We propose a framework for self-learning LLM agents by training a retrospective language model that is subsequently used to refine the LLM agent through reflective feedback on its past trajectories.

### 3.2.1 Stage I: Multi-Level Reflection Synthesis

Prior studies show that learners who engage in self-explanation while studying examples perform better and develop more robust knowledge structures than those who do not. A key reason is that as learners explain individual steps (concrete level), they often invoke domain principles or rules (abstract level) to justify why a step is correct or incorrect. This bridges the gap between specific feedback and general understanding (Wang et al., 2022). In fact, having to explain why an error is an error forces learners to integrate the relevant conceptual knowledge with the procedure (Tulis et al., 2016).

Motivated by these findings, we introduce a multi-level reflection framework that synthesizes reflections from the micro-level (detailed, instance-specific) to the macro-level (general, conceptual). Our design aligns with Kolb's experiential learning model, which describes a learning cycle involving concrete experiences, reflective observation, and abstract conceptualization for effective knowledge acquisition (McLeod, 2017). The overview is shown in Figure 1 and Algorithm 1 (the detailed

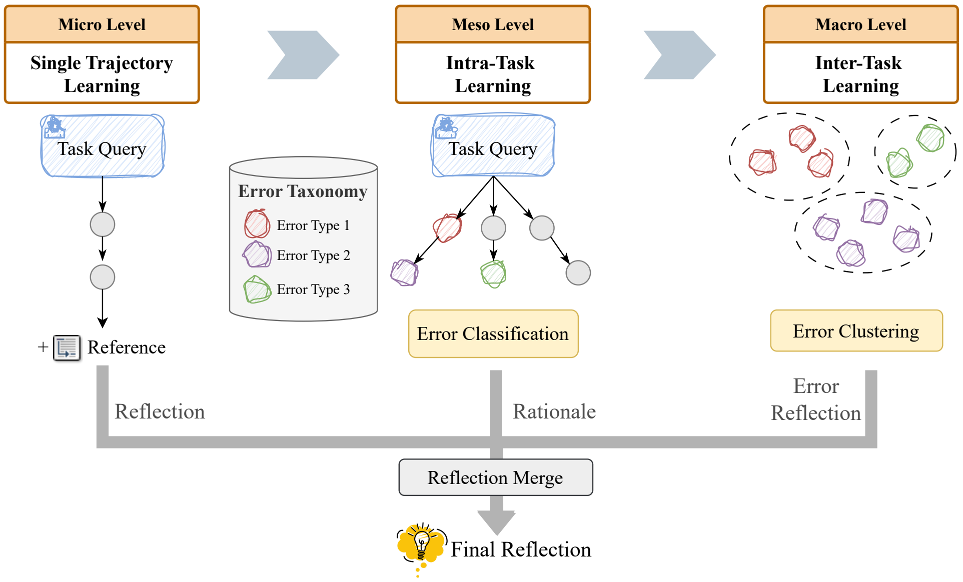


Figure 1: The overview of our proposed Multi-level Reflection Synthesis. The circles represents the steps in a trajectories and the sketched circles in different color, like in red, purple and green, indicating the steps are classified as different types of errors.

prompts are shown in Appendix § C), and the process consists of the following steps:

**Single Trajectory Learning (micro-level)**: At the micro level, we follow Reflexion (Shinn et al., 2023) to iteratively retry each training task query at most $\mathcal{K}$ times. Specifically, at the $k$-th trial, given the current training instance $(x_{\text{train}}^{(i)}, y_{\text{train}}^{(i)})$ and the current reflection $r$ (initialized as an empty string), the agent attempts the task $x_{\text{train}}^{(i)}$ using ReAct (Yao et al., 2023) as the base planning algorithm, resulting in a trajectory $\tau_{i,k} = \text{LLM}_{\text{ReAct}}(x_{\text{train}}^{(i)}, r)$. If the agent fails, we prompt the LLM to perform reflection. Inspired by prior work on contrastive learning from past successes and failures (Sun et al., 2023), we instruct LLM to compare the failed trajectory with the reference output $y_{\text{train}}^{(i)}$, diagnoses the cause of failure, and generates a new, concise plan that addresses the identified issues as the reflection $r_{i,k} = \text{LLM}_{\text{reflect}}(\tau_{i,k}, y_{\text{train}}^{(i)})$. Then in the next retry, the agent augments its context with the updated reflection $r_{i,k}$ to attempt the task again.

**Intra-Task Learning (meso-level)**: This stage focuses on learning from multiple trajectories—both successful and failed—associated with the same task query. For each task $x_{\text{train}}^{(i)}$ we concatenate all attempted trajectories and ask the LLM to identify common failure patterns, incrementally constructing an error taxonomy $\mathcal{E}$. Specifically, the LLM first checks the existing taxonomy, appends any new error types it encounters, $\mathcal{E} = \mathcal{E} \cup \text{LLM}_{\text{error}}(\mathcal{E}, \tau_{i,1}, \ldots, \tau_{i,\mathcal{K}})$. Then the LLM reviews each trajectory $\tau_{i,k}$ to label each action in the trajectories with error types (if applicable) to obtain an error path $\epsilon_{i,k}$, along with a rationale $z_{i,k}$ for the classification, i.e., $\epsilon_{i,k}, z_{i,k} = \text{LLM}_{\text{error}}(\mathcal{E}, \tau_{i,k})$.

**Inter-Task Learning (macro-level)**: At the macro level, we process each error type $e \in \mathcal{E}$ and group trajectories from different task queries that exhibit the error $e$ into a cluster $g_{\text{traj}}$. For each cluster, the LLM is prompted to generate a reflection that generalizes across tasks, identifying recurring failure modes and proposing strategies to mitigate these errors beyond individual trajectories, i.e., $r_e = \text{LLM}_{\text{reflect}}(g_{\text{traj}})$. Then each trajectory is paired with the error type reflection for the corresponding error types it contains.

**Reflection Merge**: We concatenate reflections at all levels for each trajectory $\tau_{i,k}$ and apply a final summarization step to produce a merged reflection $r_{i,k}^{\text{final}}$. This final reflection integrates both instance-specific feedback and generalized error patterns derived from related trajectories.

### 3.2.2 Stage II: Retrospective Model Training

Since our multi-level reflection framework depends on the reference output $y_{\text{train}}^{(i)}$, it cannot be directly applied during inference. To overcome this limitation, we train a smaller language model—referred to as the retrospective model—using the synthesized multi-level reflections.

We construct training examples by concatenat-

**Algorithm 1** Multi-Level Reflection Synthesis

```
Initialize:
Training data $\mathcal{D}_{\text{train}}\{(x_{\text{train}}^{(i)}, y_{\text{train}}^{(i)})\}_{i=1}^{N}$
Trajectory pool $\mathcal{P} \leftarrow \{\}$
Number of training tasks $N$
Maximum retry number $\mathcal{K}$
Error Taxonomy $\mathcal{E} \leftarrow \emptyset$
// Single Trajectory Learning
for task $i = 1$ to $N$ do
    reflection $r_{i,0} \leftarrow$ ""
    for trial $k = 1$ to $\mathcal{K}$ do
        trajectory $\tau_{i,k} \leftarrow \text{LLM}_{\text{ReAct}}(x_{\text{train}}^{(i)}, r_{i,k-1})$
        $\mathcal{P}[i] \leftarrow \mathcal{P}[i] \cup \tau_{i,k}$
        if fail($\tau_{i,k}$) then
            $r_{i,k} \leftarrow \text{LLM}_{\text{reflect}}(\tau_{i,k}, y_{\text{train}}^{(i)}))$
        end if
    end for
end for
// Intra-Task Learning
for task $i = 1$ to $N$ do
    $\mathcal{E} \leftarrow \mathcal{E} \cup \text{LLM}_{\text{error}}(\mathcal{E}, \tau_{i,1}, \ldots, \tau_{i,\mathcal{K}})$
    for trial $k = 1$ to $\mathcal{K}$ do
        $\epsilon_{i,k}, z_{i,k} \leftarrow \text{LLM}_{\text{error}}(\mathcal{E}, \tau_{i,k}))$
    end for
end for
// Inter-Task Learning
for all error $e \in \mathcal{E}$ do
    grouped trajectory $g_{\text{traj}} \leftarrow \emptyset$
    for all trajectory $\tau_{i,k} \in \mathcal{P}$ do
        if error path $\epsilon_{i,k} \supset e$ then
            $g_{\text{traj}} \leftarrow g_{\text{traj}} \cup \tau_{i,k}$
        end if
    end for
    error reflection $r_e \leftarrow \text{LLM}_{\text{reflect}}(g_{\text{traj}})$
end for
// Reflection Merge
for all $\tau_{i,k} \in \mathcal{P}$ do
    grouped error reflection $g_e \leftarrow \emptyset$
    for all error $e \in \mathcal{E}$ do
        if error path $\epsilon_{i,k} \supset e$ then
            $g_e \leftarrow g_e \cup r_e$
        end if
    end for
    final reflection $r_{i,k}^{\text{final}} \leftarrow \text{LLM}_{\text{summ}}(r_{i,k}, z_{i,k}, g_e)$
end for
```

ing the instruction, task query, background information, and trajectory as the input, and using the corresponding synthesized reflection as the target output. The retrospective model is trained via supervised fine-tuning (SFT). During inference, it takes in an agent's trajectory and generates a reflection, thereby enabling reflective feedback without access to reference outputs.

### 3.2.3 Foresight-based Reflection

Motivated by prior work that both foresight and reflection are essential for LLM-based Theory of Mind (Zhou et al., 2023), we introduce foresight-based reflection (shown in Algorithm 2) to extend our framework to the interactive setting, where the agent engages in an interaction trial with the user.

**Algorithm 2** Foresight-based Reflection

```
Initialize:
Actor $\text{LLM}_{\text{ReAct}}$
Self-reflection $\text{LLM}_{\text{reflect}}$
Current task $x^{(i)}$
Maximum step number $\mathcal{H}$
Initialize trajectory $\tau_i \leftarrow$ env.reset()
Initialize reflection $r \leftarrow$ ""
for step $t = 1$ to $\mathcal{H}$ do
    Action $a_t \leftarrow \text{LLM}_{\text{ReAct}}(a_t|\tau_i, r)$
    Predicted response $\mathcal{R}_p \leftarrow \text{LLM}(\tau_i, a_t)$
    True response $\mathcal{R}_t$, done $\leftarrow$ env.step($a_t$)
    if $\text{LLM}_{\text{diff}}(\mathcal{R}_p, \mathcal{R}_t)$ then
        $r \leftarrow \text{LLM}_{\text{reflect}}(\tau_i))$
    end if
    $\tau_i \leftarrow (a_t, \mathcal{R}_t)$
    if done then
        break
    end if
end for
```

In this setting, the agent compares its *predicted* user response with the *actual* response at each turn. Specifically, during each step, the agent first predicts the user's response $\mathcal{R}_p$ by prompting itself based on the current trajectory. Once the true user response $\mathcal{R}_t$ is observed, the agent is asked to compare $\mathcal{R}_p$ and $\mathcal{R}_t$, and then decides whether to reflect due to any significant deviation between expectation and reality. In such cases, the agent is prompted to perform reflection based on the current trajectory. This mechanism enables the agent to detect unexpected user behavior during interaction and adaptively revise its plan, improving alignment and task success in real-time settings.

## 4 Experiments

We introduce experimental details in this section and add implementation details in Appendix § A.

### 4.1 Datasets

**TravelPlanner** (Xie et al., 2024) is a challenging benchmark focused on travel planning scenarios. It offers a comprehensive evaluation environment with nearly four million data records and 1,225 carefully curated planning queries, with each training instance paired with a reference plan. In our experiments, we adopt the *sole-setting*, where relevant background information is provided directly, rather than the *two-staged setting* that requires agents to actively search for information through tool usage. This choice is motivated by computational considerations: trajectories generated under the sole-setting can exceed 10k tokens, while those in the two-staged setting are even longer, rendering training computationally infeasible given our resources. The experiments are conducted on the validation set, as the complete test data for evaluation is not completely released and we are not able to perform reflection based on the task outcomes.

**NATURAL PLAN** (Zheng et al., 2024) assesses the planning capabilities of LLMs using complete information retrieved from real-world tools, including Google Flights, Google Maps, and Google Calendar. It emphasizes structured reasoning over tool outputs to generate feasible and coherent plans. Following prior work (Lee et al., 2025), we exclude Calendar Scheduling tasks, as these can be trivially solved through enumeration. We split the dataset into training and testing sets using an 80:20 ratio.

**Tau-bench** (Yao et al., 2024a) is used to evaluate agent performance under interactive and non-interactive settings. It simulates conversations between a simulated user and an agent equipped with domain-specific APIs and policy guidelines.

### 4.2 Baselines

We compare our approach with the following reflection-based baselines:

**Fewshot**: For the setting that there is no environment that the agent can interact with, such as NATURAL PLAN, we include successful plans into the prompt for fewshot learning.

**ReAct** (Yao et al., 2023): Interleaves reasoning traces and task-specific actions, enabling LLM agents to act and reflect simultaneously.

**Reflexion** (Shinn et al., 2023): Generates verbal reflections based on task feedback and stores them in an episodic memory buffer to guide future decision-making.

**Expel** (Zhao et al., 2024a): Automatically gathers experience from a set of training tasks and extracts insights from successful trajectories or paired successful and failed trajectories.

**Inter-task Error Reflection**: Uses static, generated error-type reflections obtained from Inter-Task Learning to guide agent behavior.

**Retroformer Variant (DPO replacement)** (Yao et al., 2024b): Retroformer learns a retrospective model that tunes language agent prompts based on environment feedback via supervised fine-tuning (SFT) and Proximal Policy Optimization (PPO). However, tasks such as TravelPlanner often involve input lengths exceeding 10k tokens due to long trajectories and rich background context, making PPO-based training memory-intensive. To address this, we replace PPO with Direct Preference Optimization (DPO) for better efficiency. Our comparison on HotpotQA shows that this DPO-based variant achieves performance comparable to the original Retroformer (see details in Appendix § B).

### 4.3 Evaluation Metrics

For TravelPlanner and Tau-bench, we adopt the official evaluation scripts to compute the **Pass Rate**. For NATURAL PLAN, each instance is annotated with a single ground-truth plan. We follow the original work to use **EM-based Accuracy**: accuracy is calculated as the proportion of output plans that exactly match the annotated reference plan.

### 4.4 Main Results

We first present results and the performance change across trial numbers on TravelPlanner and NATURAL PLAN with Claude 3.5 Sonnet-v2 as the actor model in Table 1 and Figure 2. Across both benchmarks, reflection-based methods significantly outperform non-reflective baselines. Even the simplest reflective method, Reflexion, consistently improves over non-reflective methods, underscoring the value of incorporating reflection. Besides, we can also observe the following key findings:

**Cross-Trajectory Reflection Yields Gains.** Reflection across trajectories—such as Inter-task Error Reflection and our approach—substantially outperform single trajectory reflection (Reflexion). On TravelPlanner, Inter-task reaches 9.44%, compared to Reflexion's 5.56%, while our approach achieves 20%. A similar trend is observed on NATURAL PLAN, such as with our method at 60.31%, Inter-Task Error Reflection at 51.56% versus 50% (Reflexion) on the Trip domain. These results suggest that **structured, cross-trial reflection leads to more robust improvement** by abstracting recurring failure patterns across diverse task queries.

**Training Methods Comparison.** Comparing our approach to Inter-task Error Reflection reveals the benefits from training a retrospective language model via SFT. While both methods leverage cross-trial reflection, our model is fine-tuned to dynamically generate trajectory-specific reflections. This leads to improvements from 9.44% to 20% on TravelPlanner, from 51.56% to 60.31% on NATURAL PLAN Trip and from 42% to 48.5% on NATURAL PLAN Meeting. In contrast, Inter-task relies on static, precomputed reflections, which may include irrelevant errors unrelated to the current context.

Moreover, although the Retroformer variant adopts more sophisticated RL techniques, it underperforms relative to our simpler SFT. This indicates that high-quality reflection synthesis is criti-

| TravelPlanner | | NATURAL PLAN | | |
|---|---|---|---|---|
| Methods | Pass Rate | Methods | Accuracy (Trip) | Accuracy (Meeting) |
| ReAct | 4.44 | Fewshot | 44.06 | 38.50 |
| Reflexion | 5.56 | Reflexion | 50.00 | 40.50 |
| Inter-Task Error Reflection | 9.44 | Inter-Task Error Reflection | 51.56 | 42.00 |
| Expel | 0.00 | Expel | 53.79 | 41.50 |
| Retroformer variant | 12.78 | Retroformer variant | 47.50 | 44.00 |
| SaMuLe | **20.00** | SaMuLe | **60.31** | **48.50** |

Table 1: Experimental results on TravelPlanner and NATURAL PLAN with Claude 3.5 Sonnet as the actor model.

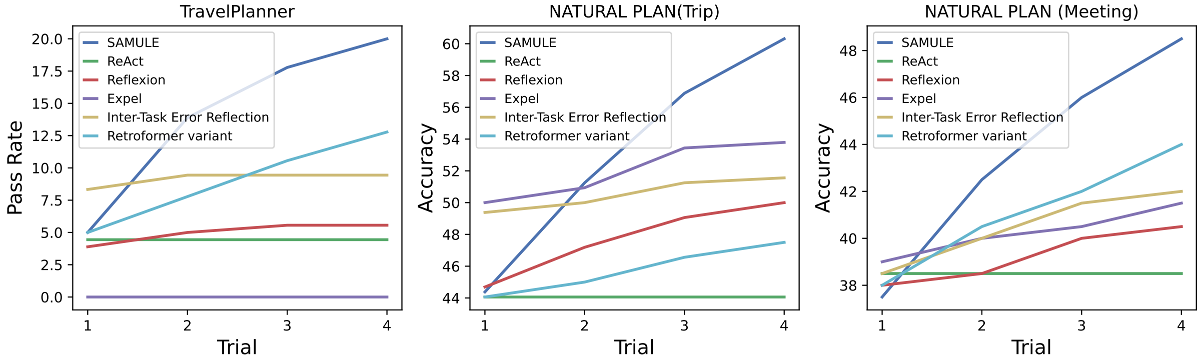


Figure 2: The performance change of different approaches across trials.

cal for training effective retrospective models; even advanced RL methods struggle if the reflection data is uninformative. These findings highlight that **well-designed reflection synthesis, even combined with simple SFT, can yield superior results while being more computationally efficient**.

**Reflexion Struggles in Complex Tasks.** Reflexion and its derivatives—Expel and Retroformer variant—struggle on more challenging benchmarks. In TravelPlanner, Reflexion achieves 5.56%, Retroformer variant improves modestly to 12.78%, and Expel collapses to 0%. These results suggest that **self-reflection alone is insufficient when agents lack a clear understanding of their failure modes**. While Expel and Retroformer variant extend Reflexion with additional prompts or fine-tuning, they remain ineffective under high task complexity. In contrast, our method focuses on diverse error types derived from multiple failures and involve reference to facilitate error analysis, enabling more informative and actionable reflection even in complex, low-success-rate environments.

**Failure Provides Stronger Signals.** Expel performs well on NATURAL PLAN (53.79% for Trip and 41.5% for Meeting) but fails on TravelPlanner (0%), likely due to its reliance on successful and paired (success/failure) trajectories, which are rare in harder tasks. These results reveal the **limitations of success-based reflection in high-error domains** and we show the generated insights by Expel and the detailed analysis in Appendix §F. Our approach instead focuses on failure, using error classification and clustering to extract insight from unsuccessful trials. This strategy proves more effective in environments where success is scarce but failures are abundant and informative by comparing with references. It underscores the value of failure-driven learning as a resilient strategy across tasks of varying complexity.

| **Methods** | **Pass Rate** |
|---|---|
| ReAct | 9.44 |
| Reflexion | 16.67 |
| Inter-Task Error Reflection | 18.89 |
| Expel | 17.22 |
| Retroformer variant | 21.67 |
| SaMuLe | **29.44** |

Table 2: Experimental results on TravelPlanner with Claude 3.7 Sonnet as the actor model.

To further demonstrate the generality and effectiveness of our approach, we evaluated SaMuLe using Claude 3.7 Sonnet as the actor on TravelPlanner in Table 2. The promising results confirm that our approach consistently improves agent performance across different actor models.

### 4.5 Interactive vs. Non-Interactive

We evaluate our framework under both non-interactive and interactive settings using Tau-Bench, and show the results in Table 3.

In the **non-interactive** setting—consistent with

| Methods | Non-Interactive (3 trials) | | Interactive | |
|---|---|---|---|---|
| | Pass Rate (Retail) | Pass Rate (Airline) | Pass Rate (Retail) | Pass Rate (Airline) |
| ReAct | 64.87 | 46.40 | 64.87 | 46.40 |
| Reflexion | 82.61 | 56.00 | 69.75 | 48.50 |
| SAMULE | **87.83** | **66.00** | **75.97** | **55.32** |

Table 3: Experimental results on Tau-Bench.

the previous experiments on TravelPlanner and NATURAL PLAN—our method consistently outperforms baselines in both domains. SEMR achieves 87.83% and 66% pass rates in Retail and Airline, respectively, surpassing Reflexion (82.61%, 56%) and ReAct (64.87%, 46.4%). These results further confirm the effectiveness of our multi-level reflection mechanism when applied to complete trajectories.

In the **interactive** setting, our approach also outperforms baselines, achieving 75.97% and 55.32% pass rates in Retail and Airline, respectively. Notably, Reflexion also benefits from interaction, improving over its non-interactive performance in Retail. This suggests that real-time feedback allows reflective agents to detect and correct partial trajectory errors, leading to better outcomes.

Overall, these results demonstrate that our framework generalizes well across both interactive and non-interactive contexts. The ability to reflect both retrospectively (on complete trajectories) and incrementally (during interactions) makes our method a practical solution for real-world applications requiring adaptive and context-aware reasoning.

### 4.6 Ablation Study: When to Add Reference?

A key constraint of our proposed framework is its reliance on a reference output during training, such as $y_{\text{train}}^{(i)}$, which is sometimes unavailable in many existing benchmarks. We hypothesize that such references provide critical guidance for error analysis—especially in complex tasks where agents frequently fail and self-reflection alone may lead to hallucination or superficial feedback.

We therefore investigate the impact of providing references at different stages of our multi-level reflection synthesis. We focus on the TravelPlanner dataset and compare the following three variants:

- **No Reference**: No reference $y_{\text{train}}^{(i)}$ is provided.
- **Single + Intra (Reference)**: The reference $y_{\text{train}}^{(i)}$ is provided during both Single Trajectory and Intra-Task Learning stages.
- **Single (Reference) + Intra (No Reference)**: The reference $y_{\text{train}}^{(i)}$ is provided only during the Single Trajectory Learning, but omitted during Intra-Task Learning.

| Method Variant | Pass Rate (%) |
|---|---|
| No Reference | 18.33 |
| Single + Intra (Reference) | 15.56 |
| Single (Reference) + Intra (No Reference) | **20.00** |

Table 4: Pass rates on TravelPlanner for different strategies of providing reference during reflection.

Table 4 shows that providing reference only during the Single Trajectory Learning stage yields the best performance (20.00%), outperforming both the no-reference variant (18.33%) and the variant that includes references in both stages (15.56%). This supports our hypothesis that **reference can be particularly useful at the micro level**, where the model benefits from detailed, item-by-item comparison between its trajectory and the reference. For example, if the agent selects a restaurant not present in the reference plan, it can reflect on the discrepancy and infer a potential mistake, such as choosing a restaurant scheduled before arrival time.

Surprisingly, providing the reference at both Single Trajectory and Intra-Task Learning results in degraded performance. We attribute this to the benchmarks like TravelPlanner, where $y_{\text{train}}^{(i)}$ is not a unique solution. **Overexposing the reference may narrow the model's focus**, encouraging it to align too closely with one specific plan and ignore other legitimate errors not captured by the reference. This suggests that while micro-level references enrich error reasoning, excessive reliance on them during the meso-level can reduce diversity in error detection and hinder generalization.

### 4.7 Error Reduction Analysis

One key feature of SAMULE is that we focus on addressing errors, especially we perform error taxonomy construction and error classification during the meso level. To further validate the effectiveness of our reflections in addressing these errors, we conducted an error reduction evaluation by adding final synthesized reflections to agents, retrying queries, and reclassifying the resulting errors. We compare

| Methods | TravelPlanner | NATURAL PLAN (Trip) | NATURAL PLAN (Meeting) |
|---|---|---|---|
| Reflexion | 0.13 | 0.42 | 0.22 |
| SAMULE | **0.67** | **0.73** | **0.53** |

Table 5: Error reduction rates achieved by our synthesized reflections and Reflexion.

it with the reflections generated by Reflexion and show the results in Table 5. This quantitative result shows that our proposed multi-level and error-prone reflection synthesis significantly improves the agent's ability to address identified error types.

### 4.8 Qualitative Analysis

Due to limited space, we provide some qualitative examples, including an example of constructed error taxonomy during the meso-level of our proposed Multi-level Reflection Synthesis in Appendix § E and a comparison between our approach and the Retroformer Variant on TravelPlanner in Appendix §D. The qualitative comparison illustrates that our trained retrospective model identifies the true issues in the agent's output plan, enabling the agent to correct its mistakes in the subsequent trial. In contrast, the Retroformer Variant incorrectly diagnoses irrelevant errors, which biases the agent toward focusing on unrelated factors—such as geographic constraints and meal timing considerations—that are not actually implicated in the original plan. This highlights the superior diagnostic capability of our approach in guiding more targeted and meaningful corrections.

## 5 Conclusion

In this work, we introduced SAMULE, a new framework for self-learning LLM agents via training a retrospective language model enhanced by multi-level reflection synthesis. By focusing on the Single-Trajectory, Intra-Task, and Inter-Task learning, our method synthesizes reflections via systematically analyzing past trajectories from micro-levl to macro-level, which identifies recurring error patterns and formulates more effective plans. To operationalize this reflection-driven learning, we train a retrospective language model using the synthesized reflections. Experimental results on the TravelPlanner, NATURAL PLAN, and Tau-bench demonstrate that our approach improves agent performance in complex planning tasks, underscoring the value of structured, cross-trial reflection for self-improvement. Our findings highlight the potential of multi-level reflective learning via SFT as a general paradigm for enhancing the performance of LLM agents.

## 6 Limitations

While our method demonstrates strong performance and significantly enhances agents' ability to learn from failures through multi-level reflection, several limitations remain:

**Static Error Taxonomy Limits Continual Learning.** Our framework constructs an error taxonomy during the offline reflection synthesis process to guide failure analysis and reflection generation. However, this taxonomy remains static throughout inference. As agents encounter new tasks or previously unseen failure patterns, the existing taxonomy may become incomplete or outdated. Future work could explore incremental taxonomy construction and online adaptation methods to support lifelong learning in dynamic environments.

**Computational Overhead in Multi-Level Reflection Synthesis.** Generating and organizing reflection data at multiple levels introduces additional computational costs during the data preparation phase. Although the final retrospective model is lightweight and efficient at inference time, the offline processes of trajectory analysis, error taxonomy construction, and cross-task clustering are resource-intensive, especially for large-scale datasets with long trajectories. Future research could investigate more scalable reflection synthesis techniques or efficient memory management strategies to mitigate this overhead.

## References

Amal Alabdulkarim, Madhuri Singh, Gennie Mansi, Kaely Hall, and Mark O. Riedl. 2022. Experiential explanations for reinforcement learning. *arXiv preprint*.

Zhenfang Chen, Delin Chen, Rui Sun, Wenjun Liu, and Chuang Gan. 2025. Scaling autonomous agents via automatic reward modeling and planning. *arXiv preprint*.

Xueyang Feng, Bo Lan, Quanyu Dai, Lei Wang, Jiakai Tang, Xu Chen, Zhenhua Dong, and Ji-Rong Wen. 2025. Improving retrospective language agents via joint policy gradient optimization. *arXiv preprint*.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, and 1 others. 2024. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*.

Jinglong Gao, Xiao Ding, Yiming Cui, Jianbai Zhao, Hepeng Wang, Ting Liu, and Bing Qin. 2024. Self-evolving GPT: A lifelong autonomous experiential learner. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6385–6432, Bangkok, Thailand. Association for Computational Linguistics.

Yubin Ge, Ly Dinh, Xiaofeng Liu, Jinsong Su, Ziyao Lu, Ante Wang, and Jana Diesner. 2021. Baco: A background knowledge-and content-based framework for citing sentence generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1466–1478.

Yubin Ge, Devamanyu Hazarika, Yang Liu, and Mahdi Namazifar. 2023. Supervised fine-tuning of large language models on human demonstrations through the lens of memorization. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Yubin Ge, Salvatore Romeo, Jason Cai, Raphael Shu, Monica Sunkara, Yassine Benajiba, and Yi Zhang. 2025. Tremu: Towards neuro-symbolic temporal reasoning for llm-agents with memory in multi-session dialogues. *arXiv preprint arXiv:2502.01630*.

Hitesh Golchha, Sahil Yerawar, Dhruvesh Patel, Soham Dan, and Keerthiram Murugesan. 2024. Language guided exploration for RL agents in text environments. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 93–102, Mexico City, Mexico. Association for Computational Linguistics.

Priyanshu Gupta, Shashank Kirtania, Ananya Singha, Sumit Gulwani, Arjun Radhakrishna, Gustavo Soares, and Sherry Shi. 2024. MetaReflection: Learning instructions for language agents using past reflections. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8369–8385, Miami, Florida, USA. Association for Computational Linguistics.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhenlan Ji, Daoyuan Wu, Pingchuan Ma, Zongjie Li, and Shuai Wang. 2024. Testing and understanding erroneous planning in llm agents through synthesized user inputs. *arXiv preprint arXiv:2404.17833*.

Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. 2025. Evolving deeper llm thinking. *arXiv preprint arXiv:2501.09891*.

Yanhong Li, Chenghao Yang, and Allyson Ettinger. 2024. When hindsight is not 20/20: Testing limits on reflective thinking in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3741–3753.

Fengyuan Liu, Nouar AlDahoul, Gregory Eady, Yasir Zaki, and Talal Rahwan. 2024. Self-reflection makes large language models safer, less biased, and ideologically neutral. *arXiv preprint*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*.

Saul McLeod. 2017. Kolb's learning styles and experiential learning cycle. *Simply psychology*, 5.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. 2024. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint*.

Chen Qian, Yufan Dang, Jiahao Li, Wei Liu, Zihao Xie, YiFei Wang, Weize Chen, Cheng Yang, Xin Cong, Xiaoyin Che, Zhiyuan Liu, and Maosong Sun. 2024. Experiential co-learning of software-developing agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5628–5640, Bangkok, Thailand. Association for Computational Linguistics.

Santosh Kumar Radha, Yasamin Nouri Jelyani, Ara Ghukasyan, and Oktay Goktas. 2024. Iteration of thought: Leveraging inner dialogue for autonomous large language model reasoning. *arXiv preprint*.

Matthew Renze and Erhan Guven. 2024. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint*.

Ming Shen, Raphael Shu, Anurag Pratik, James Gung, Yubin Ge, Monica Sunkara, and Yi Zhang. 2025. Optimizing llm-based multi-agent system with textual feedback: A case study on software development. *arXiv preprint arXiv:2505.16086*.

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.

Chen Sun, Wannan Yang, Thomas Jiralerspong, Dane Malenfant, Benjamin Alsbury-Nealy, Yoshua Bengio, and Blake Richards. 2023. Contrastive retrospection: honing in on critical steps for rapid learning and generalization in rl. *Advances in Neural Information Processing Systems*, 36:31117–31139.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. 2024. A survey on self-evolution of large language models. *arXiv preprint*.

Songjun Tu, Jingbo Sun, Qichao Zhang, Xiangyuan Lan, and Dongbin Zhao. 2024. Online preference-based reinforcement learning with self-augmented feedback from large language model. *arXiv preprint*.

Maria Tulis, Gabriele Steuer, and Markus Dresel. 2016. Learning from errors: a model of individual processes. *Frontline Learning Research*, 4(2):12–26.

Chengwei Wang, Junyi Li, Haiyan Li, Yijing Xia, Xiaoyu Wang, Yufei Xie, and Jinyang Wu. 2022. Learning from errors? the impact of erroneous example elaboration on learning outcomes of medical statistics in chinese medical students. *BMC medical education*, 22(1):469.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. 2024a. Rethinking the bounds of llm reasoning: Are multi-agent discussions the key? *Preprint*, arXiv:2402.18272.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. 2024b. Rethinking the bounds of llm reasoning: Are multi-agent discussions the key? *Preprint*, arXiv:2402.18272.

Renxi Wang, Haonan Li, Xudong Han, Yixuan Zhang, and Timothy Baldwin. 2024c. Learning from failure: Integrating negative examples when fine-tuning large language models as agents. *arXiv preprint arXiv:2402.11651*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. Travelplanner: a benchmark for real-world planning with language agents. In *Proceedings of the 41st International Conference on Machine Learning*, pages 54590–54613.

Zhihui Xie, Liyu Chen, Weichao Mao, Jingjing Xu, Lingpeng Kong, and 1 others. 2025. Teaching language models to critique via reinforcement learning. *arXiv preprint arXiv:2502.03492*.

Wei Xiong, Hanning Zhang, Chenlu Ye, Lichang Chen, Nan Jiang, and Tong Zhang. 2025. Self-rewarding correction for mathematical reasoning. *arXiv preprint*.

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.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. 2024a. $\tau$-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

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. 2024b. Retroformer: Retrospective large language agents with policy gradient optimization. In *The Twelfth International Conference on Learning Representations*.

Xunjian Yin, Xinyi Wang, Liangming Pan, Xiaojun Wan, and William Yang Wang. 2024. Gödel agent: A self-referential agent framework for recursive self-improvement. *arXiv preprint*.

Wenqi Zhang, Yongliang Shen, Linjuan Wu, Qiuying Peng, Jun Wang, Yueting Zhuang, and Weiming Lu. 2024. Self-contrast: Better reflection through inconsistent solving perspectives. *arXiv preprint*.

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

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

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V Le, Ed H Chi, and 1 others. 2024. Natural plan: Benchmarking llms on natural language planning. *arXiv preprint arXiv:2406.04520*.

Pei Zhou, Aman Madaan, Srividya Pranavi Potharaju, Aditya Gupta, Kevin R McKee, Ari Holtzman, Jay Pujara, Xiang Ren, Swaroop Mishra, Aida Nematzadeh, and 1 others. 2023. How far are large language models from agents with theory-of-mind? *arXiv preprint arXiv:2310.03051*.

## A Implementation Details

We tested Claude 3.5 sonnet-v2 and Claude 3.7 sonnet as the backbone actor model for LLM agents,

| Method | Success Rate (%) |
|---|---|
| Retroformer (SFT + reward model training + PPO) | 43 |
| Retroformer Variant (SFT + DPO) | **44** |

Table 6: Success rates of Retroformer and its variant on HotPotQA.

and also use Claude 3.5 sonnet-v2 to synthesize high-quality reflections through our multi-level reflection synthesis. For the non-interactive setting, we follow previous work (Yao et al., 2024b; Shinn et al., 2023) we run the agent for at most 4 trials.

As for training the retrospective model, we select QWEN 2.5 3B. Models are trained with LoRA (Hu et al., 2022) and DeepSpeed stage 3 for saving memory, and are optimized using AdamW (Loshchilov and Hutter, 2018). We set the learning rate to $5e-5$, and the learning rate was updated using a linear decay schedule with an end value of 0. We set the total training epochs to 15, and the batch size to 1. The training was performed on 8 NVIDIA A100 Tensor Core GPUs and it took about 1 hour for one training run.

## B Comparison of Retroformer and Its Variant on HotPotQA

While the original Retroformer (Yao et al., 2024b) has demonstrated effectiveness on relatively simple benchmarks such as HotPotQA (Yang et al., 2018), our research focuses on more complex and challenging agentic benchmarks where trajectories are significantly longer. For instance, on **TravelPlanner**, agent trajectories can exceed 10,000 tokens, introducing substantial challenges for model training due to long-context limitations and increased memory consumption.

To facilitate a fair comparison and mitigate out-of-memory issues caused by lengthy inputs, we evaluate the original Retroformer (SFT + reward model training + PPO) against its more memory-efficient variant (SFT + DPO) on the HotPotQA benchmark. Specifically, we follow the same reward shaping strategy used in Retroformer, where the reward is computed based on task improvement after incorporating a generated reflection. This allows us to pair high-reward samples with low-reward samples for PPO training. Both models are trained using the Qwen-2.5 3B language model.

As shown in Table 6, the Retroformer variant achieves a comparable success rate to the original Retroformer (44% vs. 43%). This result confirms our hypothesis that DPO can serve as a more memory-efficient alternative to PPO without sacrificing performance. Therefore, we adopt the Retroformer variant in our subsequent experiments to better accommodate the computational demands of long-context agentic tasks.

## C Designed Prompts for Multi-Level Reflection Synthesis

We list the designed prompts for our proposed multi-level reflection synthesis on TravelPlanner, and for other benchmarks we slightly modify the instruction to accommodate for the corresponding benchmark. Specifically, we show the prompt for Single-Level Learning in Figure 3, the prompt for constructing the error taxonomy during Intra-Task Learning in Figure 4, the prompt for error classification during Intra-Task Learning in Figure 5 and the prompt for Inter-Task Learning in Figure 6.

## D Qualitative Example on TravelPlanner

We include a qualitative example of our approach on TravelPlanner in Figure 7 and the output from Retroformer Variant in Figure 8. Due to the limited space, we only show part of the output. In the output plan by the LLM agent, it violates two constraints: it selected a wrong accommodation which requires minimum 2 nights stay but only scheduled to stay for 1 night; the total cost of the plan is $2064 and exceeds the budget $ 1900. Then in the generated reflection by our approach, it pointed out these two errors and leads the agent to fix it in the next trial. However, Retroformer Variant identifies wrong errors, including geographic coordination issues and meal scheduling errors, as the geographic and meal timing information are not specified in the agent's output plan.

## E Example of Generated Error Taxonomy

We show an example of the constructed error taxonomy during the meso-level of our Multi-level Reflection Synthesis on TravelPlanner in Figure 9.

## F Generated Insights by Expel

We present the insights generated by Expel on TravelPlanner in Figure 10, and those on NATURAL PLAN in Figures 11 and 12. Overall, we observe that the insights produced for TravelPlanner are overly general, primarily focusing on cost-related issues while overlooking other critical error types,

```
You are an advanced reasoning agent that can improve based on self refection.
You will be given a previous reasoning trial in which you were given access to
an automatic cost calculation environment, a travel query to give a plan and
relevant information.  Only the selection whose name and city match the given
information will be calculated correctly.  Meanwhile, you are also given one
valid plan to the given query as a reference to facilitate your analysis, though
not the only valid one.  Now by comparing your previous trial with the valid
plan and verifying factuality of your previous trial with the given information,
give a comprehensive diagnosis of your previous trial in a few sentences without
mentioning the valid plan. Then devise a new, concise, high level plan that aims
to mitigate potential failures. Use complete sentences.

Given information: ${text}

Previous trial:
Query: ${query} ${scratchpad}

Valid plan: ${annotated_plan}

Reflection:
```

Figure 3: The designed prompt for Single-Level Learning.

such as accommodation policy violations and travel time planning errors. This can lead the agent to have bias towards cost-related analysis in its future trials and ignore other errors, which further hurts the agent performance and even leads the pass rate on TravelPlanner to 0%. Such a limitation is likely a result of Expel's strict reliance on learning from successful trajectories, which are scarce in complex environments where agents struggle to produce successful plans.

You are an advanced reasoning agent that aims at improving reasoning through self refection. You were tasked to give a valid plan to a travel query given its relevant information and an automatic cost calculation environment. Only the selection whose name and city match the given information will be calculated correctly. You have tried to solve this task multiple times, and you will also be given your previous reasoning paths of these trials with their final results (either success or fail). Now, you need to analyze all reasoning paths of these trials by checking whether each action step leads to a successful final plan and what types of error you have made at each action step.

Step-by-step Instruction:
1. Read all information and analyze all given trials based on the given information. Carefully check how failed trials violate information or constraints based on the meta data from the given information, which caused the failure.
2. Based on your analysis, come up with a complete error taxonomy for categorizing the all common errors that you have made for actions in different failed trials. You should merge similar error types as one. You will be given the error taxonomy from previous reflections, and you can add new error types.
3. Generate your output in JSON format as one dictionary which contains the following keys:
a. "error_taxonomy": a list of strings containing all error type names without explanation.
b. "rationale": your explanation of giving this error taxonomy including how the failed trials fail and how the valid plan, and successful trials success.

Given Information: ${text}

Query: $ {query}

Previous Trials: $ {trials}

Error Taxonomy: $ {error_taxonomy}

Figure 4: The designed prompt for constructing the error taxonomy during the Intra-Level Learning.

```
You are an advanced reasoning agent that aims at improving reasoning through self refection. You were tasked to give a valid plan to a travel query given its relevant information and an automatic cost calculation environment. Only the selection whose name and city match the given information will be calculated correctly. You have tried to solve this task multiple times, and you will be given a reasoning path of one previous trials with their final results (either success or fail). Each reasoning path consists of a sequence of thought, action and observation at each timestep. Now, you need to reflect on the reasoning path of the current trial by classifying the error type for each action step and analyze whether each action step leads to a successful final plan.

Step-by-step Instruction:
1. Read all information and analyze the current trial.
2. For the current step in the current trial, examine if it’s valid and correct by checking whether it satisfies the given information and commonsense. Additionally, you need to compare it with the provided successful plan that satisfies the query to analyze the error. Then based on the given error taxonomy and the rationale on how the error taxonomy is proposed from all trials to the query, classify any potential errors by giving corresponding error types if it contains errors. You can only select the error types from the error taxonomy and you can leave it empty if there is no error.
3. Give a critique as your explanation of each error classification for the current step in the current trial by considering whether the current step contributes to a valid final plan. Do not refer to other trials in your critique.
4. Generate your output in a JSON format as a list of dictionaries. If there is no error, such as for successful trials, you can leave the list empty. Each dictionary item contains the following keys:
a. ”error_type”: one error type for the current step. It must be from the given error taxonomy.
b. ”critique”: the critique of the current step for the classified error type.

Given Information: $ {text}

Query: $ {query}

Error Taxonomy:
$ {error_taxonomy} Error Taxonomy Rationale:
$ {rationale}

$ {trial_id}:
$ {current_trial}

Valid Plan:
$ {annotated_plan}

Current Step:
$ {current_step}
```

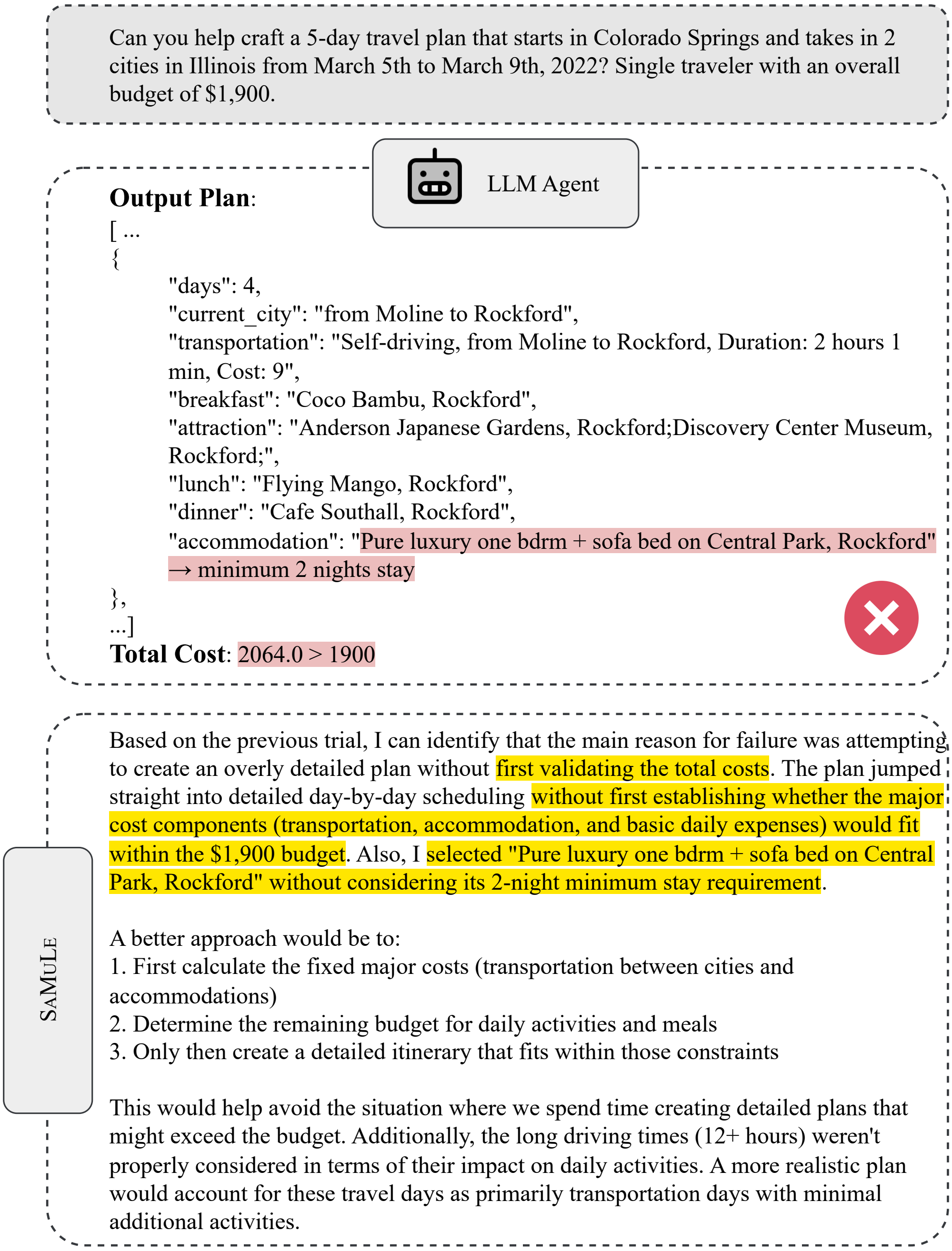
Figure 5: The designed prompt for constructing the error classification during the Intra-Level Learning.

You are an advanced reasoning agent that aims at improving reasoning through self refection. You were tasked to give a valid plan to each travel query given its relevant information and an automatic cost calculation environment. You have tried to solve multiple queries and made different errors in your trajectories that led to the failures of tasks. Now, you will be given multiple actions that you made with the same classified error type from different trials and each of these action has a corresponding critique regarding the error type. Your task is to give a comprehensive and high level diagnosis for the failure across different trials, and devise a new, concise, high level plan that aims to mitigate the same error. Use complete sentences.

Error Type: ${error_type}

$ {error_trajectories}

Figure 6: The designed prompt for the Inter-Level Learning.



Can you help craft a 5-day travel plan that starts in Colorado Springs and takes in 2 cities in Illinois from March 5th to March 9th, 2022? Single traveler with an overall budget of $1,900.

**Output Plan**:
[ ...
{
"days": 4,
"current_city": "from Moline to Rockford",
"transportation": "Self-driving, from Moline to Rockford, Duration: 2 hours 1 min, Cost: 9",
"breakfast": "Coco Bambu, Rockford",
"attraction": "Anderson Japanese Gardens, Rockford;Discovery Center Museum, Rockford;",
"lunch": "Flying Mango, Rockford",
"dinner": "Cafe Southall, Rockford",
"accommodation": "Pure luxury one bdrm + sofa bed on Central Park, Rockford"
→ minimum 2 nights stay
},
...]
**Total Cost**: 2064.0 > 1900

Based on the previous trial, I can identify that the main reason for failure was attempting to create an overly detailed plan without first validating the total costs. The plan jumped straight into detailed day-by-day scheduling without first establishing whether the major cost components (transportation, accommodation, and basic daily expenses) would fit within the $1,900 budget. Also, I selected "Pure luxury one bdrm + sofa bed on Central Park, Rockford" without considering its 2-night minimum stay requirement.

A better approach would be to:
1. First calculate the fixed major costs (transportation between cities and accommodations)
2. Determine the remaining budget for daily activities and meals
3. Only then create a detailed itinerary that fits within those constraints

This would help avoid the situation where we spend time creating detailed plans that might exceed the budget. Additionally, the long driving times (12+ hours) weren't properly considered in terms of their impact on daily activities. A more realistic plan would account for these travel days as primarily transportation days with minimal additional activities.

Figure 7: An example of the generated reflection by our approach on TravelPlanner.

Retroformer Variant

The comprehensive diagnosis reveals multiple critical issues in the failed travel planning attempt. Here's a detailed summary of the key problems and how to address them:

Primary Planning Errors:
1. Geographic Coordination Issues:
    - Failed to properly verify restaurant locations across different cities
    - Selected accommodations without checking their availability in correct cities
    - Mixed venue types (e.g., restaurants with attractions) without careful verification

Solution: Implement strict geographic verification protocols including:
    - Cross-reference all locations with correct city matches
    - Confirm accommodation availability in destination cities only
    - Limit attraction selections to actual locations in each city
    - Verify all venues exist at requested times

2. Timing and Resource Management:
    1. Meal Scheduling Errors:
        - Planned unrealistic breakfast time after late arrival
        - Served meals immediately upon arriving at each location\n- Did not account for travel time in meal planning

    Solution: Establish clear meal scheduling protocol:
        - Start meals 2-3 hours after arriving
        - Maintain consistent meal intervals
        - Allow buffer time for transitions between activities
        - Consider group size when planning meals
3. Budget Allocation:
    1. Poor Resource Distribution:
        - Front-loaded expenses on early days
        - Insufficient consideration of daily spending caps
        - Uneven distribution of budget across expenses

    Solution: Implement systematic budget tracking:
        - Calculate daily budget cap ($380-$400)
        - Group activities by cost-effective timing
        - Reserve 20-25% for future days
        - Monitor total expenses throughout planning process

...

Figure 8: Part of the generated reflection by Retroformer Variant to the same input in Figure 7.

```
Location Verification Error
Accommodation Child Policy Violation
Accommodation Selection Error
Travel Time Scheduling Error
Accommodation Minimum Stay Violation
Travel Time Planning Error
Accommodation Pet Policy Violation
Inefficient Resource Allocation
Geographic Data Misinterpretation
Restaurant Timing Error
Budget Allocation Error
Accommodation Smoking Policy Violation
Transportation Planning Error
Insufficient Data Recognition
Invalid Location Selection
Budget Constraint Violation
Accommodation Capacity Violation
Query Requirement Mismatch
Attraction Planning Error
Transportation Cost Error
Incomplete Day Planning
Restaurant Selection Error
House Rules Violation
Flight Connection Error
Attraction Distribution Error
Accommodation Party Policy Violation
```

Figure 9: The constructed error taxonomy on TravelPlanner.

```
- Calculate the total cost of the entire trip after planning each day to ensure it stays within the given budget, making adjustments as necessary. {4}
- Allocate the budget strategically across all days of the trip, considering factors such as transportation costs, accommodation prices, and planned activities for each day. {4}
- Start by outlining the major components of the trip that fit the given constraints before diving into detailed planning. {3}
- When the total cost exceeds the budget, prioritize essential elements of the trip and look for cost-saving alternatives in transportation, accommodation, or activities to bring the plan within budget constraints. {3}
- When encountering an error in planning (e.g., invalid dinner option), quickly adjust and retry with a valid alternative. {2}
- Use the CostEnquiry action frequently to get real-time cost updates for each day’s plan, allowing for immediate adjustments if necessary. {2}
```

Figure 10: The generated insights by Expel on TravelPlanner.

- Always start by clearly defining the travel query parameters, including origin, destination, dates, and any specific requirements. {2}
- Utilize the automatic cost calculation environment efficiently by inputting all relevant information accurately and completely. {2}
- Consider multiple travel options and compare their costs, duration, and convenience before recommending a plan. 2
- When presenting a travel plan, provide a clear breakdown of costs, itinerary details, and any important notes or restrictions. {2}

Figure 11: The generated insights by Expel on NATURAL PLAN (Trip).

- Begin by gathering all essential meeting parameters, including participants, preferred dates and times, duration, and meeting objectives. {2}
- Ensure all participants’ availability is accurately collected to avoid scheduling conflicts. Leverage shared calendars or scheduling tools when possible. {2}
- Consider alternative time slots and meeting formats (virtual, in-person, hybrid) to accommodate diverse participant preferences and logistical constraints. {2}
- When proposing a meeting schedule, clearly outline the agenda, expected duration, and any required preparation to ensure effective participation. {1}

Figure 12: The generated insights by Expel on NATURAL PLAN (Trip).