

# Self-evolving Agents with reflective and memory-augmented abilities

Anonymous ACL submission

## Abstract

Large language models (LLMs) have made significant advances in the field of natural language processing, but they still face challenges such as continuous decision-making, lack of long-term memory, and limited context windows in dynamic environments. To address these issues, this paper proposes an innovative framework—Self-evolving Agents with Reflective and Memory-augmented Abilities (SAGE). The SAGE framework comprises three agents: the User, the Assistant, and the Checker. By integrating iterative feedback, reflective mechanisms, and a memory optimization mechanism based on the Ebbinghaus forgetting curve, it significantly enhances the agents’ capabilities in handling multi-tasking and long-span information. The agents, through self-evolution, can adaptively adjust strategies, optimize information storage and transmission, and effectively reduce cognitive load. We evaluate the performance of the SAGE framework on AgentBench(Liu et al., 2023b) and long text tasks. Experimental results show that SAGE significantly improves model performance, achieving a 2.26X improvement on closed-source models and an improvement ranging from 57.7% to 100% on open-source models, with particularly notable effects on smaller models. Our datasets are available at <https://anonymous.4open.science/r/SAGE-FD7D/>

## 1 Introduction

In recent years, large language models (LLMs) have made significant progress in the field of natural language processing, demonstrating powerful performance in tasks such as dialogue and text generation(Brown et al., 2020). However, these models still face several challenges: (1) Agents need to continuously make decisions in changing environments and

adapt to new situations and tasks. (2) Agents lack long-term memory mechanisms, which is increasingly evident in situations requiring sustained interaction with the environment(Graves et al., 2016). The limited context window also hinders the model’s ability to handle information over long time spans(Rae et al., 2019).

To address these challenges, researchers have proposed methods such as meta-learning and multi-task learning to enhance the transferability and adaptability of LLM agents. Regarding the issue of limited memory storage, previous research such as MemGPT(Packer et al., 2024) adopts a first-in, first-out (FIFO) queue to forget content, while MemoryBank(Zhong et al., 2023) establishes a forgetting curve based on the insertion time of each item. However, these methods are typically tailored to specific tasks or scenarios, lacking a general framework to systematically improve the performance of LLM agents in complex real-world settings.

Recently, some innovative projects like AutoGPT<sup>1</sup> and BabyAGI<sup>2</sup> have started leveraging LLMs as core controllers, driving the development of agents capable of tackling complex real-world challenges. Nevertheless, existing multi-agent frameworks also face certain drawbacks when handling complex tasks, such as frequent communication and information overload issues. Communication between agents heavily relies on memory to maintain context, and as the interaction history accumulates, the computational resource demands and inference latency significantly increase. These challenges severely impede the efficient deployment and application of agents in real-world scenarios.

<sup>1</sup><https://github.com/Significant-Gravitas/AutoGPT>

<sup>2</sup><https://github.com/yoheinakajima/babyagi>

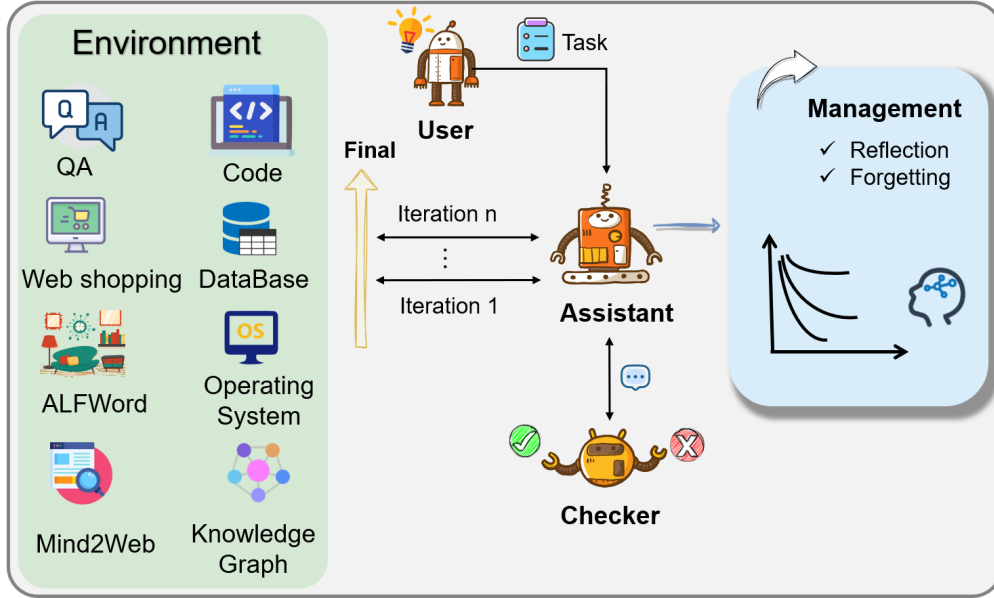


Figure 1: An illustration of the SAGE

In this paper, we propose an innovative framework, Self-evolving Agents with reflective and memory-augmented abilities (SAGE). By enhancing agents’ self-adjustment capabilities through reflection, they can more effectively utilize historical information and make efficient decisions when faced with complex and dynamic tasks. From the perspective of self-evolution, we introduce a memory optimization mechanism based on the Ebbinghaus forgetting curve (Ebbinghaus, 1885). This mechanism helps agents selectively retain key information, optimize information storage and transmission, reduce unnecessary cognitive load, and enhance agents’ capabilities in interaction tasks with the environment. Experimental results show that our approach achieves significant improvements across various benchmarks, particularly excelling in smaller models. Specifically, on AgentBench, the performance of powerful LLMs like GPT-3.5 and GPT-4 is enhanced by up to 2.26X. For open-source models, performance improvements range from 57.7% to 100%. In tasks such as multi-source question answering and code generation, our approach achieves state-of-the-art results (Etezadi and Shamsfard, 2023).

The main contributions of our work are as follows:

- We propose a novel framework, SAGE, which introduces a reflection mechanism to enhance

the self-adjustment capabilities of agents. Without any additional training, this enables agents to more effectively utilize historical information and make efficient decisions when faced with complex and dynamic tasks.

- We introduce a memory optimization mechanism based on the Ebbinghaus forgetting curve. This helps agents selectively retain key information, reducing the issue of information overload in multi-agent systems.
- SAGE achieves improvements over strong baselines in multiple challenging real-world tasks and achieves state-of-the-art results on benchmarks. This framework can be extended to other LLMs, with particularly notable improvements in smaller models.

## 2 Related work

### 2.1 Self-Improvement of Reasoning and Decision-Making

A lot of research is focused on making large language models (LLMs) better at improving themselves. Some researchers are working on using carefully crafted prompts to help models learn how to get better, although this usually only works for one-off tasks. Others are tweaking how models get

**User:** Answer the question based on the given passages. Only give me the answer and do not output any other words. The following are given passages:  
 The "trust system" (sometimes incorrectly called "trustee system") was a penitentiary system of discipline and security enforced in parts of the United States until the 1980s .....

**Assistant:** Gates v. Collier

**Checker:**  
**Incorrect answer:** Neglecting specific years: Assistant did not accurately confirm the specific years of each case, leading to an incorrect judgment of the chronological order.  
**Insufficient understanding of the question:** When answering the question, Assistant may not have fully understood or carefully analyzed the provided information, resulting in a misjudgment.

**Reflection:**  
 When answering the question about which case, "Miller v. California" or "Gates v. Collier," was brought up first, I mistakenly thought "Gates v. Collier" preceded "Miller v. California." In reality, "Miller v. California" came first. This indicates that I made an error in judging the chronological order of events.  
**Improvement measures:** Carefully verify information: When answering questions involving time sequences, it is essential to carefully check the specific timing of each event to ensure accuracy.  
**Timeline tools:** When dealing with multiple events or cases, use timeline tools to help clarify the chronological order and avoid oversights.  
**Review mechanism:** Before submitting the final answer, establish a self-review mechanism to recheck all relevant information and details to ensure the accuracy of the response.  
**Conclusion:** This mistake serves as a reminder for me to be more careful and cautious when handling questions related to time and sequence. Through reflection and improvement, I hope to provide more accurate and reliable answers in the future and prevent similar errors from occurring.

**Assistant with SAGE response:**  
 Miller v. California

Figure 2: The illustration of an example HotpotQA with SAGE. Please refer to the appendix B for more examples.

126 feedback during tasks, which helps them get bet- 155  
 127 ter at thinking things through(Huang et al., 2022). 156  
 128 There’s also work on using strategies like random 157  
 129 beam searches(Meister et al., 2022) to help models 158  
 130 make smarter decisions and assess their own work. 159  
 131 Most current methods rely on quick, one-off tweaks 160  
 132 and learning strategies that need lots of resources 161  
 133 and hands-on tech help(Tian et al., 2024). This paper 162  
 134 introduces a self-reflection mechanism, showing that 163  
 135 LLMs can keep getting better and produce higher 164  
 136 quality work across different tasks, all without need- 165  
 137 ing extra training(Zhang et al., 2024a; Cheng et al., 166  
 138 2024; Jeong et al., 2024).

139 **2.2 Memory Mechanism for LLM-based Agents**

140 In LLM-based agents, the memory module is a crit- 168  
 141 ical component responsible for storing, processing, 169  
 142 and retrieving task-related information. This mem- 170  
 143 ory plays a pivotal role in how the agent accumulates 171  
 144 knowledge, handles historical experiences, and sup- 172  
 145 ports its decision-making processes. To enhance 173  
 146 the self-evolution capabilities of these agents, re- 174  
 147 searchers are focused on designing and optimizing 175  
 148 these memory modules(Raffel et al., 2020). Past re- 176  
 149 search has covered various designs and implementa- 177  
 150 tions of memory modules. This includes integrating 178  
 151 information from different trials to boost reasoning 179  
 152 abilities or storing information in natural language 180  
 153 to enhance the module’s interpretability and user- 181  
 154 friendliness(Wada et al., 2019). While there has been 182

155 progress, further improvements in self-adjustment 156  
 157 capabilities and memory management efficiency are 158  
 159 still needed to better address complex problems in 160  
 161 real-world applications. Our proposed memory opti- 162  
 163 mization mechanism is designed to help agents better 164  
 165 manage and adapt to dynamic and complex task en- 166  
 167 vironments(Kynoch et al., 2023; Singh et al., 2024; 168  
 169 Zhang et al., 2024b).

163 **3 Method**

164 In this section, we detail the proposed Self-Adjusting 165  
 166 Generative Environment (SAGE) framework. The 167  
 168 SAGE framework aims to enhance the iterative im- 169  
 170 provement and memory management capabilities 171  
 172 of agents through three main components: iterative 172  
 173 feedback, reflection, and MemorySyntax. First, in 173  
 174 the iterative feedback process, the assistant (*A*) con- 174  
 175 tinuously optimizes its output based on feedback 175  
 176 from a checker (*C*). Second, the reflection mech- 176  
 177 anism enables the assistant to analyze experiences 177  
 178 of task success and failure and store these experi- 178  
 179 ences in memory to make better decisions in future 179  
 180 tasks. Finally, the MemorySyntax method combines 180  
 181 Ebbinghaus’s forgetting curve and linguistic knowl- 181  
 182 edge to optimize the assistant’s memory and external 182  
 storage management, allowing it to effectively pro-  
 cess and retain important information. Next, we will  
 detail the specific implementation and workflow of  
 these components.

### 3.1 Iterative Feedback

During the iterative feedback and improvement phase, the assistant ( $A$ ) in the SAGE framework iteratively receives feedback from the checker ( $C$ ) to refine its output. This process continues until the checker deems the assistant’s output correct or the iteration limit is reached.

#### 3.1.1 Initialization Phase

**Role Assignment:** The SAGE framework consists of three agents: the user ( $U$ ), the assistant ( $A$ ), and the checker ( $C$ ). The user ( $U$ ) represents the agent that initiates the task and, upon receiving prompt  $PU$ , assumes the role of task proposer. The assistant ( $A$ ) represents the agent that generates text and actions based on observations from the environment and, upon receiving prompt  $PA$ , generates text and actions based on these observations. The checker ( $C$ ) is responsible for evaluating the assistant’s output and providing feedback, assuming this role upon receiving prompt  $PC$ .

**Task Assignment:** The user ( $U$ ) assigns a task description and a successful instance to the assistant ( $A$ ) as initial input to start the interaction.

#### 3.1.2 Actual Interaction Phase

In the actual interaction phase, the assistant generates the appropriate output at each time point  $t$  based on the task description and instance provided by the user. Specifically, the assistant generates text and actions  $ot$  based on the current instruction  $st$  and necessary information:

$$ot \sim \pi\theta(ot|st, rt, ft^i) \quad (1)$$

where  $\pi\theta$  denotes the assistant’s policy,  $rt$  represents the reward score for task performance, and  $ft^i$  represents the feedback provided by the checker at the  $i$ -th iteration.

The environment provides feedback  $rt$ , including possible changes or new information. Subsequently, the checker evaluates the assistant’s output  $ot$  and provides feedback  $ft^i$ . If the assistant’s output format is incorrect, the BLEU metric is used to compare all possible action choices, selecting the closest match as the assistant’s action for that step. The assistant iteratively adjusts its output based on the checker’s feedback  $ft^i$  until the maximum trial num-

ber  $N$  is reached or the checker deems the output correct.

#### 3.1.3 Evolutionary Goals and Directions

Based on the current iteration feedback, the assistant generates new evolutionary goals:

$$\mathcal{E}^{t+1} = (\mathcal{A}^{t+1}, \mathcal{D}^{t+1}) \quad (2)$$

where the evolutionary capability  $\mathcal{A}^{t+1}$  refers to memory optimization mechanisms and the evolutionary direction  $\mathcal{D}^{t+1}$  refers to self-adjustment capabilities. The assistant adjusts its policy according to the new evolutionary goals and directions:

$$\pi\theta^{t+1} = \psi(\pi\theta^t, \mathcal{E}^{t+1}) \quad (3)$$

### 3.2 Memory Management

In the SAGE framework, memory is divided into two types: Short-Term Memory (STM) and Long-Term Memory (LTM).

**Short-Term Memory** is used to store information that is immediately relevant to the current task. It is highly volatile and has limited capacity. As the agent processes new information and makes decisions, short-term memory is frequently updated. Trajectory history is used as short-term memory to help the assistant recall and process information in the short term (Mnih et al., 2015).

**Long-Term Memory** is used to store information deemed important and useful for future tasks. Compared to short-term memory, long-term memory has a larger capacity and can retain information for longer periods. The assistant’s generated self-reflections  $ft$  are stored in long-term memory (Graves et al., 2016).

#### 3.2.1 Reflection

The reflection module provides the assistant with a sparse reward signal, such as a binary success state (success/failure), the current trajectory, and its persistent memory. The assistant analyzes these inputs and stores the learned lessons in memory to make better decisions in future attempts. The assistant generates self-reflection  $ft$  to provide feedback for future attempts, which is more informative than scalar rewards and is stored in the agent’s memory  $M$ . This process can be represented by the following equations:



$$ft = \text{ref}(o1 : t, r1 : t) \quad (4)$$

$$M \leftarrow M \cup \{ft\} \quad (5)$$

where  $\text{ref}$  denotes the reflection function.

### 3.2.2 MemorySyntax

The MemorySyntax method combines the Ebbinghaus forgetting curve and linguistic knowledge to simulate the memory and forgetting mechanisms in the human brain, applying them to the agent’s memory and external storage management. Let  $It$  represent the information received at time  $t$ , and  $R(It, \tau)$  represent the retention rate of information  $It$  after time  $\tau$ . According to the Ebbinghaus forgetting curve, we have:

$$R(It, \tau) = e^{-\frac{\tau}{S}} \quad (6)$$

where  $S$  represents the strength of the information, which is related to the importance and complexity of the information.

MemorySyntax optimizes the forgetting mechanism in the agent’s memory by adjusting sentence structure in the order of part-of-speech priority. Let  $It^*$  represent the optimized information, then we have:

$$R(It^*, \tau) = \begin{cases} e^{-\frac{\tau}{S^*}}, & \text{if } It^* \text{ is stored in } Ms \\ e^{-\frac{\tau}{S}}, & \text{if } It^* \text{ is stored in } Ml \end{cases} \quad (7)$$

where  $S^* > S$  indicates that the optimized information has a longer retention time in working memory ( $Ms$ ).

When the importance of information decreases to a certain threshold, it transfers from working memory ( $Ms$ ) to long-term memory ( $Ml$ ) or is completely forgotten. We need to update the agent’s memory during this process. Let  $Mt$  represent the agent’s memory state at time  $t$  and  $It^*$  represent the optimized information, then the memory update process can be expressed as:

$$Mt + 1 = \begin{cases} Mt \cup \{It^*\}, & \text{if } R(It^*, \tau) \geq \theta_1 \\ Mt \setminus \{It^*\}, & \text{if } R(It^*, \tau) < \theta_2 \\ Mt, & \text{otherwise} \end{cases} \quad (8)$$

where  $\theta_1$  and  $\theta_2$  represent the thresholds for retaining information in working memory ( $Ms$ ) and completely forgetting it, respectively, with  $\theta_1 > \theta_2$ .

When  $R(It^*, \tau) \geq \theta_1$ , it indicates that the information’s importance is high enough to be retained in working memory ( $Ms$ ). When  $R(It^*, \tau) < \theta_2$ , it indicates that the information’s importance is very low and can be completely forgotten. When  $\theta_2 \leq R(It^*, \tau) < \theta_1$ , it indicates that the information’s importance is between the two thresholds and should be transferred to long-term memory ( $Ml$ ).

By this means, we can simulate the memory and forgetting mechanisms in the human brain, enabling the agent to manage its memory and external storage resources more effectively. Working memory ( $Ms$ ) retains the most important and recent information, long-term memory ( $Ml$ ) stores some important but infrequently used information, and unimportant information is completely forgotten. This mechanism helps alleviate the problem of memory capacity limitations and improves the agent’s performance in tasks that require long-term memory.

## 4 Experiment

To demonstrate the capabilities and performance of SAGE in coordinating autonomous agent groups to collaboratively complete tasks, we conduct extensive quantitative experiments on benchmark tasks. We use a public benchmark: AgentBench, a multi-dimensional evolutionary benchmark, from which we select six tasks. These tasks evaluate the reasoning and decision-making abilities of LLMs acting as agents in multi-turn open-ended generation settings. To comprehensively assess the agents’ long-context understanding capabilities, we select four widely adopted tasks in the domain of long text. These tasks reflect the agents’ programming abilities(LCC(Guo et al., 2023), RepoBench-P(Liu et al., 2023a)) and reasoning abilities(HotpotQA<sup>3</sup>, TriviaQA<sup>4</sup>).

### 4.1 Evaluation on AgentBench

**Task Description** AgentBench includes scenarios based on CODE (Knowledge Graph, OS, DB), GAME (ALFWorld)(Shridhar et al., 2021), and WEB (WebShop(Yao et al., 2023), Mind2Web(Deng

<sup>3</sup><https://hotpotqa.github.io/>

<sup>4</sup><https://github.com/mandarjoshi90/triviaqa>

Table 1: aseline Performance on AgentBench without SAGE Framework

LLM Type	Models	VER	OS	DB	KG	ALF	WS	M2W
API	GPT-4(OpenAI, 2023)	0613	42.4	32.0	57.4	78.0	67.1	27.0
	GPT-3.5(Ouyang et al., 2022)	0613	31.6	15.7	25.9	17.0	64.1	16.0
OSS	Llama2-7b(Touvron et al., 2023)	chat	0.0	0.0	0.0	0.0	4.4	0.0
	Codellama-7b(Rozière et al., 2024)	instruct	5.7	2.6	0.0	0.0	16.3	0.0
	Qwen1.8b(Bai et al., 2023)	chat	2.7	1.4	6.8	0.0	6.6	0.6
	Qwen-7b(Bai et al., 2023)	chat	5.6	4.8	0.0	34.0	0.0	0.0
	Chatglm2-6b(Du et al., 2022)	v1.1	0.0	0.0	0.0	0.0	0.3	4.9

Table 2: Performance on AgentBench with SAGE Framework

LLM Type	Models	VER	OS	DB	KG	ALF	WS	M2W
API	Gpt-4	0613	49.7	39.8	63.1	82.0	67.8	32.0
	Gpt-3.5	0613	38.3	35.6	37.6	23.0	72.1	28.0
OSS	Llama2-7b	chat	8.4	10.2	25.0	5.0	10.4	15.0
	Codellama-7b	instruct	18.4	19.2	27.0	12.5	40.2	15.0
	Qwen1.8b	chat	18.7	15.1	45.3	10.5	11.4	13.6
	Qwen-7b	chat	22.2	18.0	48.0	38.5	13.6	15.0
	Chatglm2-6b	v1.1	15.2	16.3	17.0	5.0	10.3	14.9

et al., 2023)). Due to page limitations, please refer to the appendix A for detailed information.

**Baselines** We evaluate API-based commercial models GPT-3.5 and GPT-4. For open-source models, we evaluate Llama2, Codellama, Qwen, and ChatGLM2. We truncate dialogue history that exceeds the model length limit and typically use greedy decoding.

**Results** As shown in Table 2, in the AgentBench test, our method significantly improves the performance of various models, especially smaller ones. Although GPT-3.5 and GPT-4 have already achieved high scores in the benchmark tests, their performance has also improved notably with the adoption of SAGE, with improvements reaching up to 2.26 times in the Database task. Llama2-7b has been enhanced to a state with certain capabilities, demonstrating the significant effect of this method on relatively weaker models.

Additionally, CodeLlama-7b and Qwen-1.8B also show substantial improvements. Notably, Qwen-1.8B, after using our method, performs close to GPT-3.5, highlighting its potential as a general agent. The originally error-prone Llama2, through feedback mechanisms and memory optimization, exhibits a

significant reduction in basic errors, proving that our method not only activates the agent capabilities of the model but also effectively reduces fundamental errors and logical mistakes in complex tasks.

## 4.2 Evaluation of Long-Context Tasks

**Task Description** We evaluate the agent’s code generation and reasoning abilities on the following four long-text tasks:

- i. **LCC Dataset (Guo et al., 2023)** The LCC dataset is derived from the original long code completion dataset. This dataset includes the first few lines of long code as context and the next line of code as the answer. We use Precision, Recall, and F1 as evaluation metrics.
- ii. **RepoBench-P (Liu et al., 2023a)** measures the system’s ability to retrieve the most relevant code snippets from other files as cross-file context, use both cross-file and within-file context to predict the next line of code, and handle complex tasks that combine retrieval and next-line prediction. We also use Precision, Recall, and F1 as evaluation metrics.

Table 3: Comparison of Performance Across Different Methods

	LCC			RepoBench-P			HotpotQA	TriviaQA
	Precision	Recall	F1	Precision	Recall	F1	F1	F1
Reflexion	77.72	81.00	79.28	78.73	81.86	80.25	11.26	11.23
Beam search	78.98	79.32	79.12	78.75	81.02	79.87	10.26	12.13
SAGE	78.76	79.88	79.29	79.27	83.28	81.22	22.06	22.76

Table 4: Ablation study for memory optimization

	OS	DB	KG	ALF	WS	M2W
<b>Qwen-1.8B (w/o memo)</b>	10.4	22.6	6.8	0.0	26.6	5.0
<b>Qwen-1.8B (w memo)</b>	18.7	28.3	45.3	10.5	31.4	25.1
<b>Codellama-7B (w/o memo)</b>	9.7	2.7	0.0	0.0	14.3	5.0
<b>Codellama-7B (w memo)</b>	23.4	41.3	48.0	12.5	58.7	15.0

iii. **HotPotQA** (Yang et al., 2018) is a dataset based on Wikipedia, containing 113k question-answer pairs. It challenges the agent to parse content and reason over several supporting documents. We use answer F1 as the evaluation metric.

iv. **TriviaQA** (Joshi et al., 2017) is a reading comprehension dataset containing question-answer pairs with evidence paragraphs. We filter out paragraphs with fewer than 1,000 words as potential examples from TriviaQA. We use answer F1 as the evaluation metric.

**Comparison Methods:** We compared two methods that use the Self-refine mechanism: Beam Search and Reflexion. Beam Search is a decoding algorithm that integrates self-assessment guidance through stochastic beam search. Reflexion gains experience from past trials in a verbal form.

#### Evaluation Results:

**Code Completion Task:** In the LCC dataset, the SAGE agent performs excellently on Precision, Recall, and F1 metrics, showing significant improvement compared to Beam Search. Specifically, the SAGE agent effectively reviews previous predictions, uses memory mechanisms to identify and correct errors, thereby improving the accuracy and consistency of code completion, and reducing repetitive and erroneous information. In contrast, while Beam Search also employs some self-optimization strategies, it often fails to maintain the same level of precision and consistency when dealing with complex code

structures and cross-file context, leading to slightly inferior performance.

**Reasoning Tasks:** In the HotPotQA and TriviaQA datasets, the SAGE agent significantly outperforms Reflexion in F1 scores. The SAGE agent can more effectively integrate and utilize multi-document information when handling complex reasoning tasks and can review and optimize its answers after each reasoning session through the reflection mechanism, ensuring progressive improvement during the answering process. Memory optimization enables the agent to maintain mastery of important information over a long period, thus maintaining efficiency and accuracy in complex question-answering tasks. In contrast, although Reflexion (Shinn et al., 2023) relies on past trial experience, its self-optimization ability is insufficient when faced with the complexity of multi-document reasoning and long contexts, making it difficult to achieve the same performance improvement.

Overall, the SAGE agent outperforms Beam Search and Reflexion in various tasks, demonstrating its strong capabilities in complex code generation and reasoning tasks.

### 4.3 Error analysis

As shown in Figure 3, the SAGE framework significantly improves agent performance across multiple tasks, particularly excelling in the WS task. This is mainly attributed to the iterative feedback mechanism, which gradually optimizes the assistant’s out-

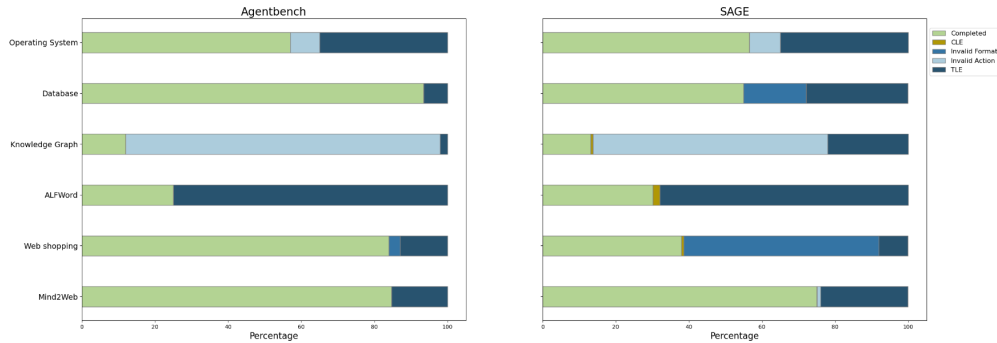


Figure 3: Distribution of various execution results across six tasks. (CLE: Exceeded Context Limit, TLE: Surpassed Task Limit). Task limits exceeded are the main reason for incomplete tasks, pointing to limitations in LLM agents’ reasoning and decision-making within constrained timeframes.

put through continuous interaction between the assistant and the checker. Moreover, in the OS and DB tasks, CLE and invalid format errors are almost completely eliminated, while invalid action errors are significantly reduced. This is largely due to the reflection mechanism, which helps the assistant learn from each task, reducing logical and invalid format errors. For detailed data, please refer in the appendix.

#### 4.4 Ablation Study

We conduct ablation experiments on the Qwen-1.8B and CodeLlama-7B models in AgentBench, with results shown in Table 4, testing the effectiveness of memory optimization methods. The results show that without memory optimization, the Qwen-1.8B model performs relatively weakly across various datasets. However, once memory optimization methods are introduced, the agent’s performance significantly improves, especially in the KG task, increasing from 6.8 to 48.0, and in the ALF task, rising from 0.0 to 10.5. This indicates that the memory optimization mechanism plays an important role in enhancing the capability of smaller parameter models in handling complex tasks. Similarly, for the CodeLlama-7B model, performance on some datasets is also relatively average without the memory optimization mechanism. After optimization, the performance significantly improves, particularly in the DB task, increasing from 2.7 to 41.3, and in the WS task, rising from 14.3 to 58.7. Overall, the CodeLlama-7B model performs better than the Qwen-1.8B model on most tasks. Specifically, in the DB and WS tasks, CodeLlama-7B shows a greater improvement

after memory optimization, increasing from 2.7 to 41.3 and from 14.3 to 58.7, respectively, which is far higher than the corresponding improvements of Qwen-1.8B. This indicates that models with a larger number of parameters exhibit stronger adaptability and processing capabilities when dealing with certain types of data (Liu et al., 2023c; Shulman, 2023; Gambella et al., 2021).

## 5 Conclusion

In this paper, we propose the SAGE framework, which significantly enhances agents’ self-adjustment and memory management abilities in complex and dynamic tasks by introducing reflective mechanisms and memory optimization. Experimental results show that the SAGE framework achieves significant performance improvements across various benchmarks, especially in smaller models. In the AgentBench test, the SAGE framework not only enhances the performance of strong baseline models like GPT-3.5 and GPT-4 but also significantly improves the performance of open-source models. Through feedback mechanisms and memory optimization, the SAGE framework effectively reduces basic errors and logical mistakes in complex tasks. Particularly in smaller models, it enables them to handle complex tasks that they previously could not manage.

## Limitations

Despite the significant improvements achieved by the SAGE framework, several limitations remain. The iterative feedback process can become computationally intensive, leading to increased latency and higher



resource consumption, which may not be feasible for real-time applications or systems with limited computational power. Additionally, managing short-term and long-term memory introduces complexity and overhead, requiring further optimization. The framework’s performance heavily relies on the quality and accuracy of the checker; incorrect feedback can lead to suboptimal solutions. While the SAGE framework has shown effectiveness on specific benchmark tasks, its generalization across a broader range of tasks needs validation. It may struggle with tasks involving high levels of ambiguity or requiring deep contextual understanding. The MemorySyntax method’s thresholds for retaining or forgetting information are critical and can affect performance, necessitating careful tuning. The use of sparse reward signals may not provide sufficient granularity for learning, requiring more nuanced feedback mechanisms. Finally, the evaluation of the SAGE framework has primarily been conducted on specific datasets, introducing a risk of evaluation bias. Broader, more diverse evaluations are needed to ensure robustness and general applicability. Addressing these limitations will require further research and development to refine the framework, optimize its components, and validate its performance across diverse and complex tasks.

## Ethics Statement

The development and application of the Self-evolving Agents with Reflective and Memory-augmented Abilities (SAGE) framework presented in this paper adhere to the principles of ethical research and innovation. We acknowledge the broader impact of deploying autonomous agents in complex and dynamic environments and have taken the following measures to ensure ethical compliance:

1. **Data Privacy and Security:** Our framework does not involve the collection or processing of personal data. However, we emphasize the importance of data privacy and security in the application of similar technologies and recommend the implementation of robust data protection measures.

2. **Transparency:** We provide a comprehensive description of the SAGE framework, its mechanisms, and its decision-making processes to ensure transparency in its functioning.

3. **Bias and Fairness:** We are aware of the potential

for algorithmic bias in AI systems and have taken steps to minimize such biases in the design of our framework. We encourage further research into the ethical implications of AI decision-making.

4. **Accountability:** We maintain that the developers and deployers of AI systems should be accountable for their systems’ actions and outcomes. SAGE is designed to provide clear audit trails for its decisions.

5. **Informed Consent:** In cases where human interaction is involved, we advocate for the principle of informed consent, ensuring that all participants are aware of the AI’s role and its implications.

6. **Limitations:** We acknowledge the limitations of our work, including the potential for the framework to be misused or to fail under certain conditions. We call for ongoing research to address these concerns.

7. **Future Research:** We recommend that future work in this area should continue to consider ethical implications, including the long-term societal effects of self-evolving AI agents.

We believe that by addressing these ethical considerations, we can contribute to the responsible development and deployment of AI technologies.

## References

- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. [Qwen technical report](#). *Preprint*, arXiv:2309.16609.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Nee-lakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#). *Preprint*, arXiv:2005.14165.



714	Nicolas Usunier, Thomas Scialom, and Gabriel Syn-	Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Ben-	767
715	naeve. 2024. <a href="#">Code llama: Open foundation models for</a>	gio, William W. Cohen, Ruslan Salakhutdinov, and	768
716	<a href="#">code</a> . <i>Preprint</i> , arXiv:2308.12950.	Christopher D. Manning. 2018. <a href="#">Hotpotqa: A dataset</a>	769
717	Noah Shinn, Federico Cassano, Edward Berman, Ashwin	<a href="#">for diverse, explainable multi-hop question answering</a> .	770
718	Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023.	<i>Preprint</i> , arXiv:1809.09600.	771
719	<a href="#">Reflexion: Language agents with verbal reinforcement</a>	Shunyu Yao, Howard Chen, John Yang, and Karthik	772
720	<a href="#">learning</a> . <i>Preprint</i> , arXiv:2303.11366.	Narasimhan. 2023. <a href="#">Webshop: Towards scalable real-</a>	773
721	Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté,	<a href="#">world web interaction with grounded language agents</a> .	774
722	Yonatan Bisk, Adam Trischler, and Matthew	<i>Preprint</i> , arXiv:2207.01206.	775
723	Hausknecht. 2021. <a href="#">ALFWorld: Aligning Text and</a>	Dan Zhang, Ziniu Hu, Sining Zhoubian, Zhengxiao Du,	776
724	<a href="#">Embodied Environments for Interactive Learning</a> . In	Kaiyu Yang, Zihan Wang, Yisong Yue, Yuxiao Dong,	777
725	<i>Proceedings of the International Conference on Learn-</i>	and Jie Tang. 2024a. <a href="#">Sciglm: Training scientific lan-</a>	778
726	<i>ing Representations (ICLR)</i> .	<a href="#">guage models with self-reflective instruction annota-</a>	779
727	David Shulman. 2023. <a href="#">Optimization methods in deep</a>	<a href="#">tion and tuning</a> . <i>Preprint</i> , arXiv:2401.07950.	780
728	<a href="#">learning: A comprehensive overview</a> . <i>Preprint</i> ,	Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen,	781
729	arXiv:2302.09566.	Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-	782
730	Arjun Singh, Nikhil Pandey, Anup Shirgaonkar, Pavan	Rong Wen. 2024b. <a href="#">A survey on the memory mech-</a>	783
731	Manoj, and Vijay Aski. 2024. <a href="#">A study of optimiza-</a>	<a href="#">anism of large language model based agents</a> . <i>Preprint</i> ,	784
732	<a href="#">tions for fine-tuning large language models</a> . <i>Preprint</i> ,	arXiv:2404.13501.	785
733	arXiv:2406.02290.	Wanjun Zhong, Lianhong Guo, Qiqi Gao, and Yan-	786
734	Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian	lin Wang. 2023. <a href="#">Memorybank: Enhancing large lan-</a>	787
735	Yu, Haitao Mi, and Dong Yu. 2024. <a href="#">Toward self-</a>	<a href="#">guage models with long-term memory</a> . <i>arXiv preprint</i>	788
736	<a href="#">improvement of llms via imagination, searching, and</a>	<i>arXiv:2305.10250</i> .	789
737	<a href="#">criticizing</a> . <i>Preprint</i> , arXiv:2404.12253.		
738	Hugo Touvron, Louis Martin, Kevin Stone, Peter Al-	<b>A Detailed Dataset Information</b>	790
739	bert, Amjad Almahairi, Yasmine Babaei, Nikolay Bash-	(1) <b>Operating systems</b> Integrating LLMs into op-	791
740	lykov, Soumya Batra, Prajjwal Bhargava, Shruti Bho-	erating systems has immense potential for au-	792
741	sale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer,	tomating and optimizing tasks. This integra-	793
742	Moya Chen, Guillem Cucurull, David Esiobu, Jude	tion requires a secure, user-friendly interface to	794
743	Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cyn-	ensure effective LLM-OS interaction. LLMs	795
744	thia Gao, Vedanuj Goswami, Naman Goyal, Anthony	must accurately understand the OS context for	796
745	Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan,	informed operations, prioritizing safety to pre-	797
746	Marcin Kardas, Viktor Kerkez, Madian Khabsa, Is-	vent misuse. Additionally, the system should	798
747	abel Kloumann, Artem Korenev, Punit Singh Koura,	effectively handle errors and provide clear feed-	799
748	Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Di-	back to users, enhancing overall interaction and	800
749	ana Liskovich, Yinghai Lu, Yuning Mao, Xavier Mar-	control. Addressing these aspects can transform	801
750	tinnet, Todor Mihaylov, Pushkar Mishra, Igor Moly-	computer interaction and efficiency across vari-	802
751	bog, Yixin Nie, Andrew Poulton, Jeremy Reizen-	ous industries.	803
752	stein, Rashi Rungta, Kalyan Saladi, Alan Schelten,	(2) <b>Database</b> The ability of LLMs to operate on	804
753	Ruan Silva, Eric Michael Smith, Ranjan Subrama-	real databases via SQL is critical due to the im-	805
754	nian, Xiaoqing Ellen Tan, Binh Tang, Ross Tay-	portance and complexity of database analysis	806
755	lor, Adina Williams, Jian Xiang Kuan, Puxin Xu,	in everyday activities. Previous research has	807
756	Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela	highlighted the effectiveness of LLMs in au-	808
757	Fan, Melanie Kambadur, Sharan Narang, Aurelien Ro-	tomating database access, such as with T5QL, a	809
758	driguez, Robert Stojnic, Sergey Edunov, and Thomas	new SQL generation method. Furthermore, fine-	810
759	Scialom. 2023. <a href="#">Llama 2: Open foundation and fine-</a>	tuned LLMs (like GPT-3.5) have demonstrated	811
760	<a href="#">tuned chat models</a> . <i>Preprint</i> , arXiv:2307.09288.	the ability to extract and link complex scientific	812
761	Takashi Wada, Tomoharu Iwata, and Yuji Matsumoto.	information from texts, obtaining structured	813
762	2019. Unsupervised multilingual word embedding		
763	with limited resources using neural language models.		
764	In <i>Proceedings of the 57th Annual Meeting of the As-</i>		
765	<i>sociation for Computational Linguistics</i> , pages 3113–		
766	3124.		

814 knowledge from unstructured text and building  
815 extensive databases.

816 **(3) WebShop** WebShop is an innovative simulation  
817 of an e-commerce website environment, featur-  
818 ing 1.18 million real-world products and 12,087  
819 crowd-sourced text instructions. It challenges  
820 agents to navigate various types of webpages  
821 and perform diverse actions to find, customize,  
822 and purchase products based on given instruc-  
823 tions. WebShop’s challenges include under-  
824 standing compositional instructions, query (re-  
825 )formulation, dealing with noisy text on web-  
826 pages, and strategic exploration.

827 **(4) Knowledge Graphs** LLMs’ utilization in con-  
828 structing and interacting with knowledge graphs  
829 (KG) presents a promising opportunity to en-  
830 hance semantic understanding and information  
831 retrieval. This involves assessing the models’  
832 ability to generate and interpret complex inter-  
833 relations within data, facilitating more intuitive  
834 and context-aware responses. Effective LLM  
835 performance in this domain could significantly  
836 improve AI’s reasoning and decision-making  
837 capabilities based on structured knowledge.

838 **(5) Mind2Web** Mind2Web (M2W) is a dataset de-  
839 signed for developing web agents capable of  
840 performing complex tasks on real websites via  
841 language instructions. It features over 2,000  
842 tasks across 137 sites from 31 domains. M2W’s  
843 real web environments and diverse user interac-  
844 tions make it a crucial platform for advancing  
845 AI navigation capabilities.

846 **(6) ALFWorld** ALFWorld bridges interactive  
847 TextWorld environments with embodied tasks  
848 from the ALFRED dataset, enabling agents  
849 to learn abstract strategies and apply them to  
850 real-world tasks. It supports abstract reason-  
851 ing and concrete execution, allowing agents to  
852 plan actions in a text-based simulator and exe-  
853 cute these tasks in a visual environment. This  
854 approach enhances agent generalization and  
855 problem-solving skills across various domains,  
856 such as language understanding and visual nav-  
857 igation, by leveraging a modular design that  
858 simplifies research improvements.

## B Examples

### Cases study on TriviaQA in using GPT with SAGE.

#### User:

Answer the question based on the given passages. Only give me the answer and do not output any other words. The following are given passages:

History of canals in Great Britain - London Canal Museum History of canals in Great Britain Home > History > Canals in Britain In the history of canals, Britain was not a pioneer. The Chinese can claim that the Grand Canal of China was one of the first, in the tenth century, a. . . . .

Answer the question based on the given passages. Only give me the answer and do not output any other words.

Question: Which metal is produced by the Bessemer Process?

Answer:

#### Assistant:

The content of the file seems to be an HTML document related to the administrative division of Turkey, including population statistics, charts, and maps. This file contains scripts, metadata, and structured data typical for a web page.

#### Checker:

Clear Objectives: When answering user questions, the focus should be on directly addressing the question without introducing irrelevant information. Document Handling: Before confirming whether the document content is relevant to the question, detailed aspects of the document should not be directly addressed.

#### Reflection:

Issue Description: The user wants to understand why the system outputs irrelevant HTML document content based on user input and seeks detailed feedback and reflection. Reasons for the Error: Misinterpretation of Input: The system mistakenly assumed that the user uploaded a file and attempted to read its content, whereas the user only provided text input. Processing Logic Error: Without clearly determining the user's intent, the system directly parsed and outputted file content. User Experience Problems: Irrelevant Output: The user expects concise, relevant answers, but the system provided lengthy and irrelevant HTML content, affecting the user experience. Poor Communication: Failure to accurately understand and respond to the user's real needs resulted in off-topic responses. In-depth Reflection and Improvement Suggestions: Clarify User Intent: Accurate Parsing of User Input: Enhance the system's natural language processing capabilities to accurately distinguish whether the user uploaded a file or simply provided text input. User Feedback Mechanism: Implement a user feedback mechanism to promptly adjust the understanding of user intent. For example, confirm the issue to verify if the user uploaded a file. Optimize Processing Logic: Layered Processing Flow: In system design, separate file processing logic from text response logic to avoid misjudgments. For instance, when a user uploads a file, process the file content separately; when a user inputs text, provide relevant answers directly. Pre-processing Check: Prior to handling files, perform pre-processing checks to confirm if the file content meets expectations. . . . .

**LLMs with SAGE response:** Steel.

#### Ground truth

Steel.



### Cases study on LCC in using GPT with SAGE.

#### User:

```
import networkx as nx import pickle from Queue import PriorityQueue import copy import random import string import sys class MultiDiGraphEdgeKey(nx.MultiDiGraph): .....
```

#### Assistant:

You will be given the history of a past experience where you were given a task and failed to complete it. Reflect on the strategy and actions taken. Devise a concise, new plan of action that accounts for your previous mistakes. Here are two examples: Example 1: If you failed to remember to lock the door, plan to double-check the door each time. Example2: If you missed a deadline, plan to set reminders earlier. ....

#### Checker:

The given code is a part of the Camerini algorithm, which is used for ranking edges in a directed graph. This algorithm is particularly useful for problems involving the identification of optimal or near-optimal substructures, like arborescences or spanning trees, in complex networks. The algorithm has two main phases:

Collapsing Phase: This phase identifies and collapses cycles in the graph, creating supernodes. This simplification helps in managing cycles more efficiently and prepares the graph for the expansion phase.

Expanding Phase: After collapsing cycles into supernodes, this phase expands the supernodes back into their original structure, resulting in a ranked arborescence, where the edges are sorted or prioritized based on certain criteria. ....

#### LLMs with SAGE response:

```
Z.addedge(eu, ev, **data.copy()) self. init(graph, Y, Z) A = best(root) self.init(graph) e , d = self.next(A, Y, Z) w = self.getgraphscore(A) - d if d != float('inf') else float('inf') P.put((-w, e, A, Y, Z)) solutions.append(A) return solutions
def bestbranching(self, root): return self.best(root)
def bestarborescencenorooted(self, root): return self.best(root)
def bestarborescencerooted(self, root): return self.best(root)
def getgraphscore(self, graph): return sum(data[self.attr] for u, v, data in graph.edges(data=True))
def findroots(self, graph): roots = [node for node, degree in graph.indegree() if degree == 0] return roots
.....
```

## Cases study on Repobench-P in using GPT with SAGE.

### User :

Please complete the code given below.

```
{src//kademlia//operation//KadRefreshOperation.java
public class KadRefreshOperation implements Operation
private final KadServer server;
private final KademliaNode localNode;
private final KademliaDHT dht;
private final KadConfiguration config;
public KadRefreshOperation.....
```

### Assistant:

To devise a new plan of action, let's analyze the previous mistakes and come up with a solution: Mistake 1: In the KadRefreshOperation class, the execute() method calls the BucketRefreshOperation and ContentRefreshOperation without any error handling or exception handling. This can lead to unexpected errors and failures. Solution: Add proper error handling and exception handling in the execute() method of the KadRefreshOperation class. This will ensure that any errors or exceptions are properly handled and the execution can continue smoothly.....

### Checker:

In the 'KadRefreshOperation' class, there is no error handling when the 'execute()' method calls 'BucketRefreshOperation' and 'ContentRefreshOperation'. Analysis: If 'BucketRefreshOperation' or 'ContentRefreshOperation' throws an exception, the entire 'execute()' method will fail, potentially causing the interruption of the entire system's refresh operation. In the 'StoreOperation' class, the 'execute()' method does not handle the case of existing content in the DHT. Analysis: The 'execute()' method always returns 1 as the number of storage nodes, which is clearly incorrect. The correct number of nodes should be returned based on whether the content already exists in the DHT. Solution: Before storing the content, check if it already exists. If it does, return 0; otherwise, store the content and return the actual number of storage nodes.

.....

### LLMs with SAGE response:

```
private final transient KademliaRoutingTable routingTable; private final transient KadConfiguration config;
private final transient MessageFactory messageFactory;
/* Scheduled refresh operation */ private final Timer refreshTimer;
/** * Initialize the Kademlia system * * @param ownerId The ID for this owner * @param nodeId
The ID for this node * @param inetAddress The address for this node * @param port The port for this
node * @param config Configuration for the Kad system * * @throws java.io.IOException */ public
JKademliaNode(String ownerId, KademliaId nodeId, InetAddress inetAddress, int port, KadConfiguration
config) throws IOException { this.ownerId = ownerId; this.config = config;
/* Initialize the Node */ this.localNode = new Node(nodeId, inetAddress, port);
/* Initialize the Server */ this.server = new KadServer(localNode.getNodeId(), config);
.....
```