S-Agent: an Agent Collaborative Framework Inspired by the Scientific Methodology

Anonymous ACL submission

Abstract

An increasing number of advancements have been accomplished in agents empowered by Large Language Models (LLM), particularly in resolving simple dialogue tasks. However, existing agents still face intractable robustness issues for solving complex tasks, encountering the cascading hallucinations induced by multi-step invocations of LLM. Certain recent studies utilize multi-step reasoning, planning strategies, and domain workflows to improve the success rate of complex tasks, yet they neglect the scientific methodology that encompasses the accumulated wisdom derived from centuries of scientific inquiry. Drawing inspiration from the scientific methodology, we propose the S-Agent - an agent collaborative framework meticulously designed to actively experiment and refine theories based on the analysis of experimental results, thereby enhancing the deductive capabilities of LLMs and complementing their inductive and communicative strengths. Additionally, we introduce an innovative parallel planning methodology, wherein agents with identical roles collaborate to simultaneously address the same inquiry. Extensive experiments demonstrate the effectiveness and efficiency of our approach. Notably, we achieve a new state-of-the-art 96.3% pass@1 accuracy on the HumanEval coding benchmark with GPT-4.

1 Introduction

Recently, significant advancements have been achieved in the realm of problem-solving through agents founded upon Large Language Models (LLM). Existing studies incorporated step by step reasoning & planning strategies (Yao et al., 2023b; Liu et al., 2023b; Yao et al., 2023a; Zhou et al., 2023), and used tools to extend agents' capabilities(Wu et al., 2023; Yang et al., 2023; Shen et al., 2023). These studies have demonstrated their capability to tackle uncomplicated dialogue tasks. However, current agents continue to confront insurmountable challenges in terms of resilience when it comes to resolving complex tasks, as they encounter the cascading hallucinations brought about by the multi-step invocations of LLM.

In order to improve the proficiency of agents in resolving complex tasks, some recent studies employ domain expertise to guide agents towards adhering to standard operating procedures (SOP) (Hong et al., 2023; Qian et al., 2023). These studies can only serve specialized applications as they rely on domain-specific procedural knowledge. Moreover, the introduction of domain procedures may inadvertently introduce human biases, impeding the agent's ability to discover the factual solutions.

To address the aforementioned issues, we draw inspiration from the scientific method [1] that encompasses the wisdom accumulated through centuries of scientific exploration and has been validated across various disciplines. The magnificent modern science usually adheres to an iterative paradigm: deriving ideas from observations and constructing hypotheses. These hypotheses are then subjected to experimentation, with the outcomes serving as observations that either validate or question the hypotheses. At the heart of this paradigm lies the notion of "falsifiability," introduced by Karl Popper(Popper, 1959), which asserts that there must exist experimental findings capable of disproving the hypotheses. Guided by this paradigm, theory and experiment can mutually inform and enhance scientific understanding.

In this paper, we propose the S-Agent, a multi-agent framework where agents partake in dialogues and collaborations inspired by the scientific methodology. This framework encompasses

[1] scientific method: A method of procedure that has characterized natural science since the 17th century, consisting in systematic observation, measurement, and experiment, and the formulation, testing, and modification of hypotheses: criticism is the backbone of the scientific method.

crucial processes including idea generation, experiment conduction, and the discussion of results. Our experiments demonstrate that the system performs exceptionally well in tackling challenging tasks, such as coding.

To summarize, our key contributions are the following:

- To the best of our knowledge, we are pioneering the integration of formulating, experimenting, and modifying of hypotheses within the LLM agent system. These mechanisms represent the accumulated wisdom of centuries of scientific research and are poised to enhance the credibility and accuracy of LLM agents.
- We present S-Agent, a collaborative framework for agents that integrates an automated workflow planner and a parallel agent task management unit. The framework offers adaptable assistance for developing agents of complex and high-reliability tasks, enabling simultaneous operation at the agent level and significantly enhancing efficiency.
- We conduct extensive experiments to demonstrate the effectiveness and efficiency of our approach. Notably, we achieve a new state-of-the-art 96.3% pass@1 accuracy on the HumanEval coding benchmark with GPT-4.

2 Related work

2.1 Think Like Human

In the nascent stage of the Language Model (LLM) era, expectations were primarily set on direct question-answering capabilities. However, this period also marked the beginning of an intriguing exploration into human-like cognitive processes by AI researchers. GPT-3(Brown et al., 2020), a forerunner in this field, introduced the concept of few-shot learning. This technique mirrors the human capacity to enhance problem-solving skills through exposure to a limited number of examples. Similarly, the Chain of Thought (COT) approach (Wei et al., 2022) marked a significant advancement. By structuring problem-solving in an incremental, step-by-step manner akin to human thought processes, COT demonstrated state-of-the-art performance, at times outperforming even finely-tuned models.

Following this trajectory, the Tree of Thoughts (ToT)(Yao et al., 2023a) framework emerged as a pivotal development. It significantly advanced LLMs' problem-solving abilities by enabling the generation and exploration of multiple intermediate thoughts, guiding them towards a solution. Another notable advancement is ReAct(Yao et al., 2023b), which synergies reasoning with action. This methodology alternates between generating reasoning traces and corresponding actions, refining the process by integrating observation from external environments. This simpler, more intuitive approach has been widely adopted in open-source projects and is now a cornerstone in agent systems. Reflexion(Shinn et al., 2023) takes an even more comprehensive approach by overseeing the entire decision-making process, and providing feedback on the full action sequence.

Most previous methods simply modify the solution according to observation of the past experience and logic analysis, without targeted experiments to collect specific feedback from the environment. Drawing inspiration from the scientific method, S-Agent entails the initial integration of purposefully designed experiments within agent systems to enhance outcomes, yielding promising results. The detailed explanation will be introduced in later section

2.2 Use Tool Like Human

While the Marxist philosophy posits that tool utilization and creation are key distinctions between humans and animals, it's the transformative impact of tools on human evolution and dominance on Earth that is truly noteworthy. This concept finds a parallel in the realm of language models like ChatGPT. Despite their impressive performance and global recognition, these models have inherent limitations, including constrained calculation abilities, restricted access to rare knowledge, and limited proficiency in handling other modalities. Mirroring the human approach to overcoming similar constraints, the strategic use of tools has emerged as an effective solution in augmenting these models.

Pioneering efforts such as Visual ChatGPT(Wu et al., 2023) and HuggingGPT(Shen et al., 2023) laid the groundwork by integrating multi-modal models as auxiliary tools, thereby expanding the functionalities of LLMs beyond single-modal capabilities. This trajectory was further propelled by MM-ReAct(Yang et al., 2023), which cleverly incorporated a search engine and Microsoft API services into the mix. This innovative approach has been widely adopted, with OpenAI's ChatGPT introducing plugin functions and Microsoft's New-

Bing exemplifying an LLM integrated with Bing search capabilities.

In today's landscape, the ability to configure and customize tool sets is no longer an optional feature but a fundamental functionality in most agent assistant applications. However, most of the previous work focusing on the correctness of tool selection, in our work, we focused on how to design experiment according to given tools.

2.3 Collaborate like Human

Researchers have ventured beyond exploring human cognition and tool usage, delving into the intricacies of organizational dynamics. This exploration has been facilitated by employing multi-agent systems to create artificial entities that simulate the workings of a company. The innovative concept of role-playing was pioneered by Camel(Li et al., 2023), marking a significant milestone in multi-agent collaboration. In Camel's model, AI users and assistants play distinct roles - one assigns tasks and the other devises solutions. They work in tandem, iterating to resolve user queries effectively.

This concept was expanded in subsequent studies that modeled their operations explicitly after a corporate structure. Notable examples include MetaGPT(Hong et al., 2023) and ChatDev(Qian et al., 2023), which emulate software companies handling programming tasks. These models assign roles like CEO, CTO, product manager, programmer, and designer, systematically reflecting the organizational structure of a real-world company. The adoption of a Standard Operating Procedure (SOP), crafted by professionals and fed into the system, guides the collaborative process, creating a workflow similar to that of a traditional software company.

Building on this foundation, AgentVerse(Chen et al., 2023b) and AutoAgents(Chen et al., 2023a) introduced a job market system, simulating the recruitment of expert agents for specific roles. This approach also generate the SOP. By automating everything, the artificial company of agents achieves a high level of task proficiency and autonomy.

While these models typically employ a sequential waterfall workflow, addressing each component of a task linearly, recent research(Zhang et al., 2023; Chen et al., 2023b; Du et al., 2023) has highlighted the benefits of cooperative approaches, such as debates, in enhancing performance. Our work introduces an innovative parallel planning methodology, where agents with identical roles collaborate to simultaneously address the same question. This parallel approach has been instrumental in boosting performance, demonstrating the efficacy of multi-agent cooperation in complex problem-solving.

3 Methodology

To empower the S-Agent system with the scientific method, we have structured the process into three distinct parts: idea generation, experiment conduction, and panel discussion. Each part is facilitated by purpose-built agents, as illustrated in figure 1.

3.1 Idea Generation

The Idea generation part serves as the initial stage for proposing ideas and formulating hypotheses during the execution process of the agent system. Throughout the experiment iterations, ideas undergo refinement and new ones emerge. In this phase, we specifically design LLM-powered agents, termed **idea generation agents**, to emulate the role of scientists in generating structured ideas easily verifiable by subsequent experiments. Meanwhile, these agents can analyze feedback from discussion parts and revise their ideas accordingly. In programming scenarios, these agents initially receive a coding question as input and directly generate Python solutions. In the next few rounds, the agent shall take the discussion feedback as input and generate new solutions if the solutions generated by the previous ideas do not pass the targeted experiments. As shown in Figure 1, the idea generation agents receive the input and produce ideas. A detailed task adaption will be presented in the experiments section.

3.2 Experiments Conduction

The experiment conduction phase consists of designing appropriate experiments and carrying out the designed experiments. These steps are often considered the most crucial in the scientific method. A well-designed experiment can determine not only the validity of a hypothesis but also highlight its advantages over previous theories when the results support the hypothesis. Conversely, when the results reject the hypothesis, the experiment can pinpoint the specific issues that remain unresolved. This focused feedback can make the iteration of hypotheses much faster, also help avoid deduction analysis from wrong start. In our system, applying this hypotheses experiment alignment phase properly can avoid machine hallucination effectively.

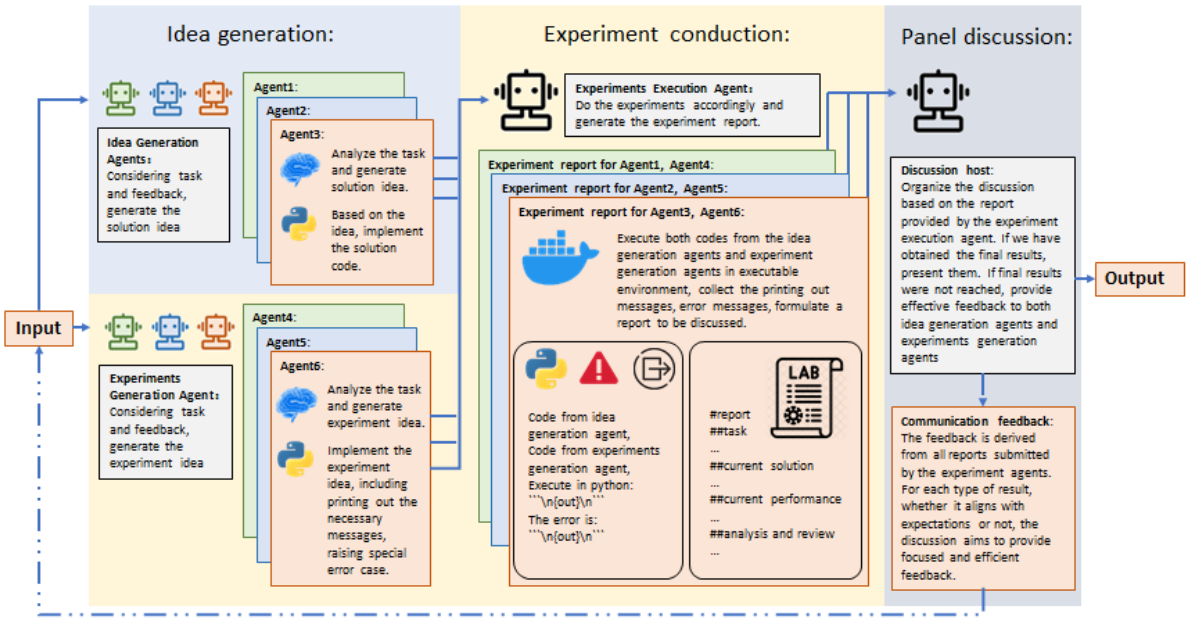

Figure 1: Illustration of S-Agent. The idea generation phase showcases agents responsible for generating ideas, encompassing both solution concepts and corresponding code. The experiment conduction phase begins with the generation of experiment ideas by experiment generation agents, followed by the execution of experiments through experiment execution agents. Experiment reports contained thorough analysis of results and the collection of feedback from the environment are also generated in this phase. In the panel discussion session, a discussion host synthesizes experiment reports from preceding agents, facilitating the generation of valuable feedback. This feedback is systematically organized to serve as input for the subsequent iteration, fostering continuous idea and experiment setting improvement.

To equip this mechanism in our system, we designed **experiment generation agents** to do experiments design and **experiment execution agents** to test out hypotheses and produce detailed experiment reports after receiving the results. As illustrated in Figure 1, the experiment generation agents formulated the targeted experiment plan based on proposed idea and execution reports, and the experiment execution agents execute codes with these test cases in execution environment.

3.3 Panel Discussion

Observing the workflow of a scientist reveals that panel discussion, often through paper publications and sharing, is a vital component of the modern academic community. Therefore, when an experiment validates an idea, we facilitate discussion among different agents. This process allows them to review others' work, ensuring internal logic consistency and providing references and inspiration to other agents. The discussion results in either feedback on analyzing experiment results or final answers if the designed experiments successfully verify the proposed ideas. To manage this process effectively, we have designed a specialized agent called a **discussion host**, responsible for aggregating information, assessing previous results, and overseeing the overall status of ongoing discussions.

3.4 Supporting Component

To simplify the deployment of the entire system and reduce execution time, we also include two supporting components named the planner and agent task management.

Automated Workflow Planner The automated workflow planner is a specially designed agent responsible for generating sequences of execution flows for agents and managing their interdependencies. This critical component strategically plans and organizes the order in which agents operate, ensuring a coherent and efficient workflow. To formulate this plan, only the input-output information and the agents' goals are required. As shown in Figure 1, the edges of the directed workflow graph are generated by the planner automatically. This approach draws inspiration from the methodology

introduced in LLMCompiler(Kim et al., 2023). The detailed prompt design and sample illustration are given in the appendix.

Agent Task Management Unit The agent task management unit, drawing inspiration from the instruction fetching mechanism in modern computer architecture, plays a crucial role in determining the optimal execution flow of agents based on the intermediate representation generated by the Planner. Employing a greedy policy, this unit swiftly adds agents to the task list as soon as they become ready for parallel calling. The implementation of this agent task management unit involves a straightforward fetching and queue mechanism, foregoing the need for a dedicated agent system.

4 Experiment

We validate the efficacy of our framework in addressing challenging issues on the coding benchmark HumanEval (Chen et al., 2021) and EvalPlus(Liu et al., 2023a). Additionally, we undertake a targeted investigation into the efficacy of our approach in mitigating hallucinations on the HotpotQA dataset (Yang et al., 2018).

4.1 HumanEval

HumanEval(Chen et al., 2021)(distributed under the MIT license) is a benchmark for code synthesis, which consists of 164 programming problems with several test cases each. The problems in this dataset are designed to test the ability of LLMs to generate functionally correct codes, which means the generated codes can not only execute successfully but also pass the provided test cases, instead of being linguistically similar to the canonical solution. EvalPlus is a benchmark that aims to improve the quality and quantity of test cases for the existing HumanEval benchmark. EvalPlus contains new test cases that can catch more errors and bugs in the LLM-synthesize code.

Our performance on these benchmarks is noteworthy, achieving a Pass@1 (Pass@1 is the probability that a model generates at least one correct solution out of one attempt) of 96.3% on HumanEval and 86.6% on EvalPlus, which set the new SOTA.

4.1.1 Implementation Details

As introduced in previous section, the scientific method consists of idea generation, experiment conduction and panel discussion three phases. Here, we elucidate our methodology for implementing the scientific methodology in the adaptation of the HumanEval dataset. We expound upon our prompt design process and furnish a comprehensive example in the appendix.

During the idea generation step, we assign the idea generation agents the dual responsibility of analysis and coding. The agents initially analyze and delineate a comprehensive strategy for addressing the current task. Subsequently, during the coding phase, they annotate each stage of their proposed solution with explanatory comments. This procedure bears resemblance to the methodology employed by a scientist in comprehending a natural phenomenon: by formulating a hypothesis that elucidates the occurrence of the phenomenon. Typically, such a hypothesis generates provisional forecasts utilizing abstract equations. Success is attained when these equations prove to be accurate and the final forecast corresponds with the observed phenomenon. Analogously, in our methodology, the analysis functions as the overarching hypothesis, while the code and its accompanying comments serve as the equations, with each stage functioning as an individual equation. Through the comments, we acquire insights into the anticipated outcomes of each stage. The annotations are depicted in Figure 2.

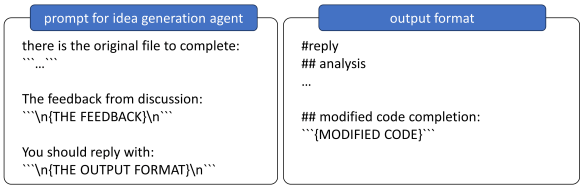

Figure 2: Prompt for idea generation agents and the output template. FEEDBACK is all the previous experiments reports.

Agents responsible for experiment design and experiment execution are specifically tailored for the conduction of experiments. Experiment generation agents are assigned the task of creating specific experiments, determining the experimental inputs, defining the expected outputs, specifying the messages to be displayed upon producing various types of outputs during analysis, and providing implementation code in the format of 'assert f(input)==output, description of the experiment'. The detailed instructions are depicted in Figure 3. This mirrors the methodology employed by experimental scientists when a new theory emerges. Instead of testing every formula within the theory,

experimental scientists conduct experiments with varying expectations using different theories to ascertain the correctness of the theory—whether it be the newly proposed one or the existing one. This process facilitates precise feedback on the validity of the theory. The experiment execution agent is responsible for executing Python code generated from idea generation code within an executable environment and producing the experiment report. The output format of the experiment execution agent is illustrated in Figure 4, and the specifics of the report format are provided in Figure 10 in the appendix. It interprets the standard output and error messages to offer specific feedback, thereby enabling the development of enhanced theories or experiment designs.

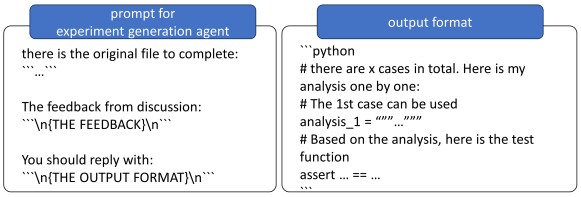

Figure 3: Prompt for experiments generation agent and the output template. FEEDBACK is all the previous experiments reports.

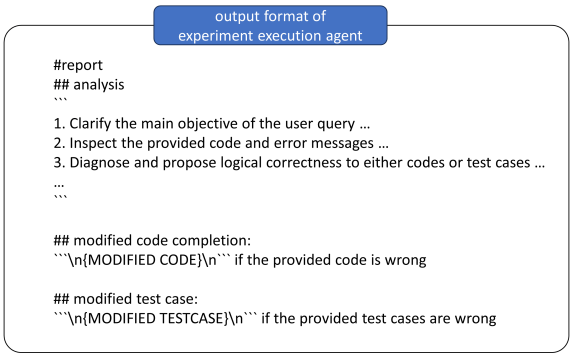

Figure 4: Output format of the experiment execution agent, where detailed analysis and modification of either codes or test cases are produced.

After the experiment reports are generated in experiment conduction section, the discussion host takes on the task of compiling all reports and evaluating whether the issue has been successfully addressed. In cases where the issue remains unresolved, all reports will undergo a comprehensive analysis by each individual responsible for generating ideas in order to enhance their proposed ideas, as indicated in the generated code. Additionally, these reports will be examined by each individual involved in generating experiments to make adjustments to the experiments. In this manner, our framework has established an effective panel discussion mechanism.

During each test case, we initially provide our system with the input. Subsequently, upon the generation of feedback, we designate it as a round. The maximum number of rounds is set at 10; should this threshold be surpassed, the system is deemed unsuccessful. Conversely, if successful, we extract the code from the ultimate outcome and deem it as our solution. Following the completion of all 164 cases, we execute the official script to obtain a Pass@1.

4.1.2 Result and Analysis

We evaluated S-Agent system using both GPT-4 and GPT-3.5. By utilizing GPT-4, our system successfully completed 158 out of 164 tasks (pass@1 = 96.3%), achieving state-of-the-art performance. Furthermore, our system leveraged GPT-3.5 to pass 137 out of 164 tasks (pass@1 = 84.1%), surpassing the performance of all our known works based on the same LLM. The results are summarized in Table 1.

Recent methodologies for coding tasks have endeavored to enhance performance by leveraging the Python execution environment, incorporating reflection mechanisms, and facilitating multi-agent discourse. By involving python execution environment, agent system can actually execute the code and see the validity. Reflection is a mechanism to improve system performance using LLM-generated verbal reinforcement cues from past experience. Different systems handle reflection in various ways: MetaGPT(Hong et al., 2023) uses a special agent to perform reflection; language-agent-tree-search (LATS)(Zhou et al., 2023) employs reflection in a trajectory format; Reflexion(Shinn et al., 2023) focuses on allowing the LLM itself to review and reflect upon the feedback. Specifically, in the coding task, feedback includes the standard output (stdout) and error messages from executing the generated code. Discussion is mainly studied in multi-agent area. By aggregating the idea from different persona, system can normally generate more comprehensive and accurate answer(Du et al., 2023; Chen et al., 2023b).

Previous methodologies simply receive feedback from the Python Interpreter, whereas our approach uniquely empowers the agent to directly influence the feedback, as illustrated in Figure 5 in great de-

	GPT4	ANPL	MetaGPT	AgentVerse	Reflextion	LATS	S-Agent
code execution	×	✓	✓	✓	✓	✓	✓
reflection	×	×	✓	✓	✓	✓	✓
panel discussion	×	×	×	✓	×	×	✓
experiment design	×	×	×	×	×	×	✓
Pass@1 (GPT3.5-base)	-	76.2%	-	75%	-	83.8%	**84%**
(GPT4-base)	67%	86.6%	87.7%	89%	91%	94.4%	**96.3%**

Table 1: Pass@1 result of related works on programming. We refer code execution as the use of python execution environment, reflection as the use of LLM-generated feedback, panel discussion as agents sharing and discuss each others work, experiment design as actively design experiment according to proposed idea and previous feedback. Our S-Agent system stands out from previous works and achieved best pass@1 score, mainly because of involving actively designing experiments.

tail. Within our framework, as delineated in the implementation section, the experiment generation agents configure the experiment to produce intermediate results, leading to additional lines such as printed statements that offer clear insights into any errors. These lines not only exploit feedback information but also enhance it through the efforts of the agent, thereby enriching the feedback with more valuable information, guiding the system towards effectively solving intricate problems. The enriched feedback from experimental results and inter-agent panel discussions has propelled us to achieve state-of-the-art performance.

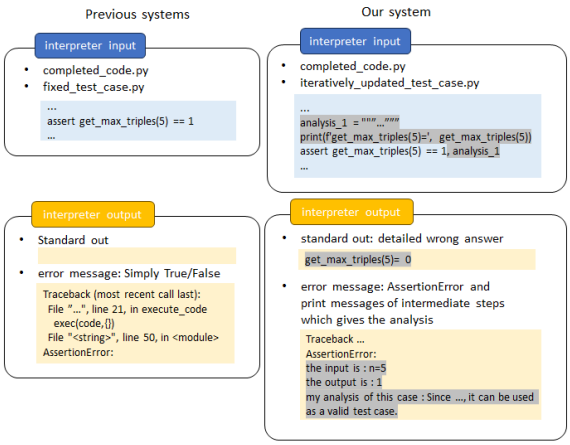

Figure 5: This figure demonstrates the differences in input and output between our system and the previous systems. Because of incorporating specially designed experiments during the python execution step, instead of just providing a binary assessment of code functionality, our system also generate intermediate results and other results based on the experiment requirement, as highlighted in the image.

4.1.3 Ablation Study

We conduct experiments on the HumanEval and EvalPlus datasets to investigate the effectiveness of mechanisms within the S-Agent framework.

Table 2 shows the results of the experimental results of three S-Agent variants including the original framework (Original), S-Agent without Panel discussion mechanism (w/o Panel), S-Agent without Panel discussion and Experiment conduction mechanism (w/o Exp&Panel), S-Agent without the Idea generation, Experiment conduction, and Panel discussion mechanisms (w/o Idea&Exp&Panel). The findings indicate that all three fundamental mechanisms play a beneficial role in enhancing the precision of the S-Agent framework.

S-Agent Variants	HumanEval	EvalPlus
Original	96.3%	89.0%
w/o Panel	92.1%	86.6%
w/o Exp&Panel	84.8%	79.9%
w/o Idea&Exp&Panel	68.3%	65.2%

Table 2: Pass@1 on HumanEval and EvalPlus using GPT-4 under different S-Agent Variants. The experiment employs a single agent for idea generation, another for experiment conduction, and a third for panel discussion.

We also investigate how the number of Idea generation and Experiment conduction agents impact the results of the S-Agent framework. As depicted in Table 3, with an increase in the number of agents, there is a clear improvement in the average pass@1 value, accompanied by a reduction in the standard deviation. This indicates that, solely from a performance perspective, increased discussion can substantially enhance both the accuracy and stability of the entire system. This conclusion also aligned with (Du et al., 2023; Yang et al., 2018). This also

aligned with our result in GPT-4 based experiments, the pass@1 increase from 92.1% to 96.3% when increasing the number of agents to 3.

Number of Agents	HumanEval Pass@1 (%)	EvalPlus Pass@1 (%)
1	80.5 ± 1.8	72.0 ± 1.8
3	83.5 ± 1.37	74.4 ± 1.37
5	84.1 ± 0.56	75.6 ± 0.56

Table 3: Pass@1 performance on HumanEval and EvalPlus using GPT-3.5 under different numbers of Idea generation and Experiment conduction agents.

4.2 HotpotQA

The HotpotQA(Yang et al., 2018) dataset, is a crucial benchmark in Natural Language Processing (NLP) and Question Answering (QA). It is specifically designed for models that handle complex, multi-hop question-answering tasks, requiring synthesis of information across multiple text sources from wiki. The HotpotQA dataset is distributed under the CC BY-SA 4.0 license. The code is distributed under the Apache 2.0 license.

Due to the limitation of budget, we sampled a subset(50 qa-pairs) from the original dataset randomly and tested no panel disccusion setting, compared with ReAct(Yao et al., 2023b), our improved the result by 7.4%. The S-Agent surpasses ReAct(Yao et al., 2023b) by offering more accurate and efficient feedback. For instance, when presented with the question "Which show was hosted by Jessica Drake's former spouse?", the GPT-4 model acknowledges the importance of first identifying Jessica Drake's ex-partner and then uncovering the shows he hosted. Jessica has two former spouses, namely "Evan Stone" and "Brad Armstrong," both known in the cinematic industry. The agents tend to consistently generate questions focused on Brad's role as a director. Our framework intervenes by providing feedback that explicitly states "The provided text does not mention any show hosted by Brad Armstrong". This feedback prompts the agent to shift attention away from Brad and focus on Evan Stone, ultimately solving the task. The precise and efficient feedback from our framework is essential in preventing the language model from persisting in incorrect directions, thus reducing the risk of entering a cycle of generating inaccurate information endlessly. The detailed prompt is provided in the appendix.

5 Conclusion

In this paper, we introduce the S-Agent, an innovative multi-agent framework in which agents engage in dialogues and collaborations inspired by the scientific methodology. The framework incorporates the essential processes of hypothesis development, experimentation, and refinement. These processes embody the collective knowledge accumulated over centuries of scientific inquiry and are poised to enhance the credibility and precision of LLM agents. The S-Agent integrates an automated workflow planner and a parallel agent task management unit, providing flexible support for developing agents for complex and high-stakes tasks, facilitating concurrent operation at the agent level. Extensive experiments confirm the efficacy and efficiency of our methodology. Notably, the S-Agent achieves a new state-of-the-art 96.3% pass@1 accuracy on the HumanEval coding benchmark with GPT-4.

6 Limitations and Future Work

The primary objective of this paper is to elucidate the concept of scientific methodology. While our general framework may not exhibit the same level of sophistication as pioneering works like AutoAgents(Chen et al., 2023a) which autonomously generate requisite agents, our framework still requires the manual implementation of specific agents tailored for specialized tasks. In future developments, a key direction of research involves exploring methods to automatically generate agents with finely crafted prompts, presenting an important avenue for further exploration.

Currently, following the generation of the Directed Acyclic Graph (DAG) plan, the plan remains static. However, it is imperative to establish dynamic refinement for this plan. Recent advancements, exemplified by works such as ReAct(Yao et al., 2023b), BabyAGI, and XAgent, have endeavored to enhance plans based on feedback received at each step. While these approaches typically involve linearly designed steps, there is a research gap in developing methods to dynamically refine a DAG-formatted plan with the capability for parallel execution.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind

Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris 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. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Sesay Jaward, Karlsson Börje, Jie Fu, and Yemin Shi. 2023a. Autoagents: The automatic agents generation framework. *arXiv preprint*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, et al. 2023b. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2023. Metagpt: Meta programming for a multi-agent collaborative framework.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael Mahoney, Kurt Keutzer, and Amir Gholami. 2023. An llm compiler for parallel function calling. *arXiv*.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023a. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zeyi Liu, Arpit Bahety, and Shuran Song. 2023b. Reflect: Summarizing robot experiences for failure explanation and correction. *arXiv preprint arXiv:2306.15724*.

Karl Popper. 1959. *The Logic of Scientific Discovery*, 2002 pbk; 2005 ebook edition. Routledge.

Chen Qian, Xin Cong, Wei Liu, Cheng Yang, Weize Chen, Yusheng Su, Yufan Dang, Jiahao Li, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. In *Advances in Neural Information Processing Systems*.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023. Mm-react: Prompting chatgpt for multimodal reasoning and action.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of Thoughts: Deliberate problem solving with large language models.

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

Jintian Zhang, Xin Xu, and Shumin Deng. 2023. Exploring collaboration mechanisms for llm agents: A social psychology view.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2023. Language agent tree search unifies reasoning acting and planning in language models.

A Additional Details

A.1 Automated Workflow Planner

Given a user query, the planner will create a plan to solve the user query with the most parallelizability. The prompt of the planner agent contains the profiles of every agent and the tools it equips. Each task in the plan must have a unique ID, which is strictly increasing. Inputs for each task can either be constants or outputs from preceding actions. If the outputs from preceding actions are needed, we use the format $id to denote the ID of the previous agent task whose output will be replaced with the input. Upon the completion of all agent tasks, we always call join as the last task in the plan to collect all the previous task outputs and formulate a final output. We use the HumanEval coding question $get\_max\_triplets$ to demonstrate the functionality of our LLM planner agents. Figure 6 shows the plan that the planner creates after receiving the coding question as the user query. In this case, user queries are fed into three Coders and the Tester. Afterward, the output from the Coders should be considered as the input to the Expereimnters with the output from the Tester respectively. Then the gathered outputs from these Experiments should be the input of the Discussion Hoster. In the end, we collect all the results and finish this plan.

A.2 Agent Task Management Unit

Figure 7 shows how the management unit handles the agent tasks. Agents are equipped with the tools that the user provides and tasks are delegated to the associate tool. The management unit synchronously listens to the task queue and schedules tasks as they arrive in the queue based on their dependencies. More specially, in this case, the three coders and the Tester agents execute parallelly at the same time due to empty dependencies. While the Experiment agents cannot execute parallelly until the completion of all Coder agents and Tester Agents. Meanwhile, we shall replace dependency placeholders, i.e. $i, in the args of the agent task with the actual output.

B Additional Results

B.1 Case study for HumanEval

To get a closer look at a specific task, we chose No.147 as a demo because this task is only solved by our system, and it cost 5 rounds of modification to get the final answer. For the task and idea generation agent, refer figure 8, for the experiment generation agent refer figure 9, for the experiment execution agent and the exact augment feedback from python, refer figure 10

Based on such augmented feedback, these agents can get more specific reviews about current code , just like scientists can draw more specific conclusions based on specific experiment results. Here they find out that the problem didn't handle reminder 1 and 2 properly .

Based on this feedback, they will repeat the work procedure. By observing the correct final answer, we would find that the issue at the first draft is that it did not consider three 1s or three 2s can also lead to reminding 0, just like three 0s.

B.2 HotpotQA

B.2.1 dataset

The HotpotQA(Yang et al., 2018) dataset stands as a pivotal benchmark within the realm of Natural Language Processing (NLP) and Question Answering (QA). This dataset serves a distinctive purpose, tailored for assessing models tasked with intricate, multi-hop question-answering assignments that necessitate the synthesis of information from various textual sources. Consequently, the inclusion of HotpotQA in evaluations requires models to showcase sophisticated reasoning and comprehension abilities.

HotpotQA offers two evaluation settings, Fullwiki and Distractor. In the Fullwiki Setting, the dataset provides only questions, and users must retrieve related information from the entire Wikipedia dataset using their Information Retrieval (IR) system. The effectiveness of the search strategy in this setting is crucial, as the content found can significantly influence the results. Users then use their models to answer the question based on this information. In the Distractor Setting, the dataset provides questions along with context from the

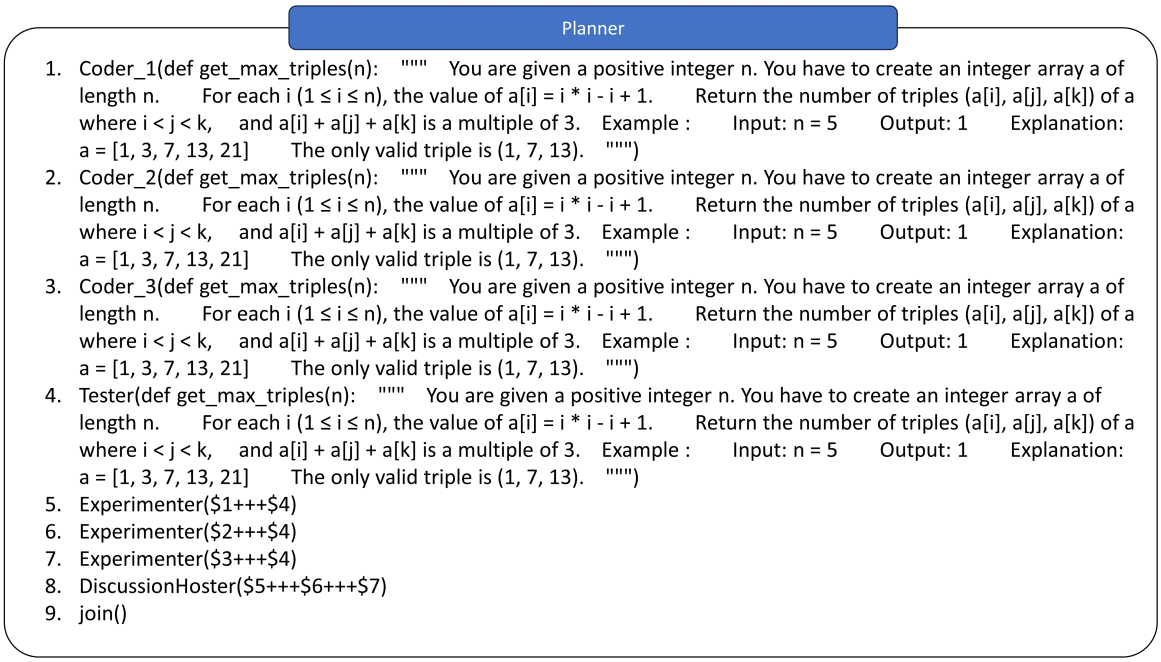

Figure 6: Given a coding task: "def $get\_max\_triplets$: You are given a positive integer n. You have to create an integer array a of length n. For each i $(1 \leq i \leq n)$, the value of $a[i] = i \times i - i + 1$. Return the number of triples $(a[i], a[j], a[k])$ of a where $i < j < k$, and $a[i] + a[j] + a[k]$ is a multiple of 3. Example: Input: $n = 5$, Output: 1, Explanation: $a = [1, 3, 7, 13, 21]$, The only valid triple is $(1, 7, 13)$", the planner agent shall automatically make the detailed plan of the agents' workflow and their dependencies.

Wikipedia dataset, which includes both related and unrelated paragraphs. In this case, the user's model must be able to sift through the shuffled context to find the relevant information and answer the question correctly. In both settings, models are tasked with predicting the answer and identifying the supporting paragraphs in the context. When evaluating performance, we use Exact Match (EM), which measures whether the model's answer precisely matches the ground truth answer.

B.2.2 implementation

In this particular dataset, we empower idea generation agents to create the entire search plan. This includes formulating a set of queries for interaction with Wikipedia and specifying the desired knowledge to be retrieved through these queries. Simultaneously generating pairs of queries and targeted knowledge streamlines the experimental process, especially when assessing the efficiency of the queries. This approach aligns with the scientific method, emphasizing the importance of providing falsifiable ideas.

In this context, the experiment is to evaluate whether the query can retrieve the targeted knowledge, therefore, a rule-based experiment generation agent is employed.

The experiment execution agent in this scenario is more intricate due to the complexity of communicating with the Wikipedia API. During the implementation of this specific experiment execution agent,we let llm automatically choose from several actions. If a Wikipedia page is successfully retrieved by a query, the agent systematically analyzes the information, starting from the first paragraph and summary information box, followed by the table of contents (TOC) of the Wikipedia page.If the answer can be directly generated from this initial information, it is returned promptly. If no answer is found, but certain paragraphs potentially contain details relevant to answering the question, the agent continues to read these paragraphs. The answer is returned if discovered in this process; otherwise, a summary and analysis detailing the reasons for the failure to find an answer are provided.

In situations where a Wikipedia page cannot be located with the given query, the agent generates a summary of similar pages returned by the API.

Figure 7: Here is an example of how the agent task management unit works for the coding task: def $get\_max\_triplets$, where tasks with no dependencies execute parallelly.

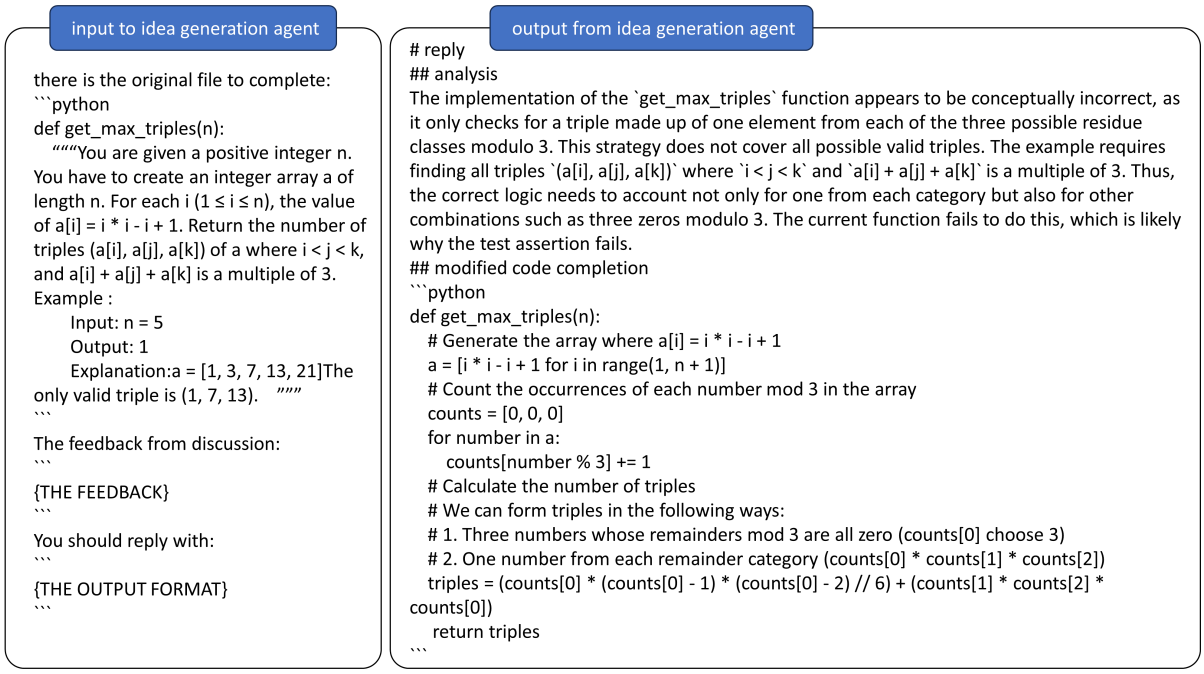

Figure 8: The original task is "You are given a positive integer n. You have to create an integer array a of length n. For each i $(1 \leq i \leq n)$, the value of $a[i] = i \times i - i + 1$. Return the number of triples $(a[i], a[j], a[k])$ of a where $i < j < k$, and $a[i] + a[j] + a[k]$ is a multiple of 3. Example: Input: $n = 5$, Output: 1, Explanation: $a = [1, 3, 7, 13, 21]$, The only valid triple is $(1, 7, 13)$."
At first, one of the idea agents gives a draft saying that for all numbers in a_i , there is only 3 possible reminders when divided by 3 : 0,1,and 2. so it suggest first counts how many numbers reminds 0 ,1,and 2 . than there should be 2 senarieo that the 3 selected 3 numbers from the list satisfy the requirement : all the reminder is 0 or reminder is 0,1,2 .these two can be calculated by the 3 counts .

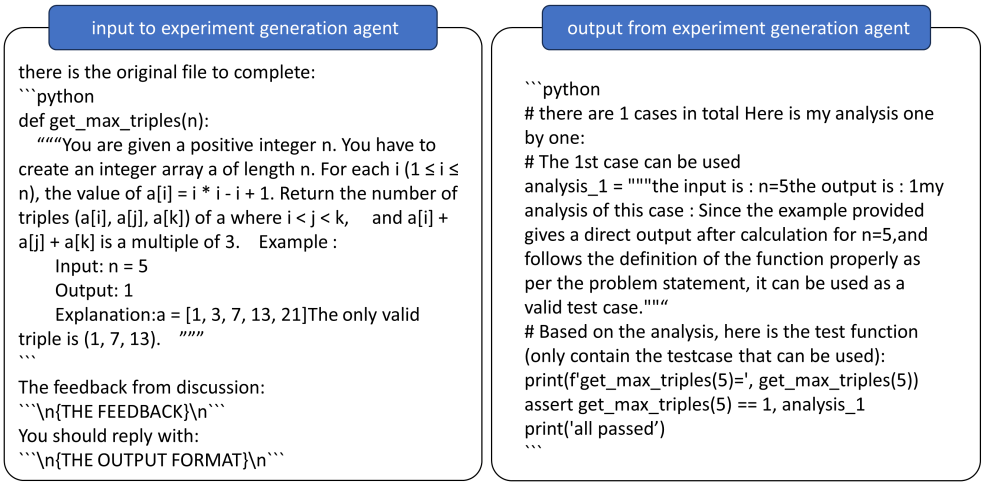

Figure 9: then , one of the experiments agents , design its experiment about the task , first print the result if n=5 , then if the code draft showed that the result is not 5 , raise an error message contains the analysis about why when n =5 it should be the result .

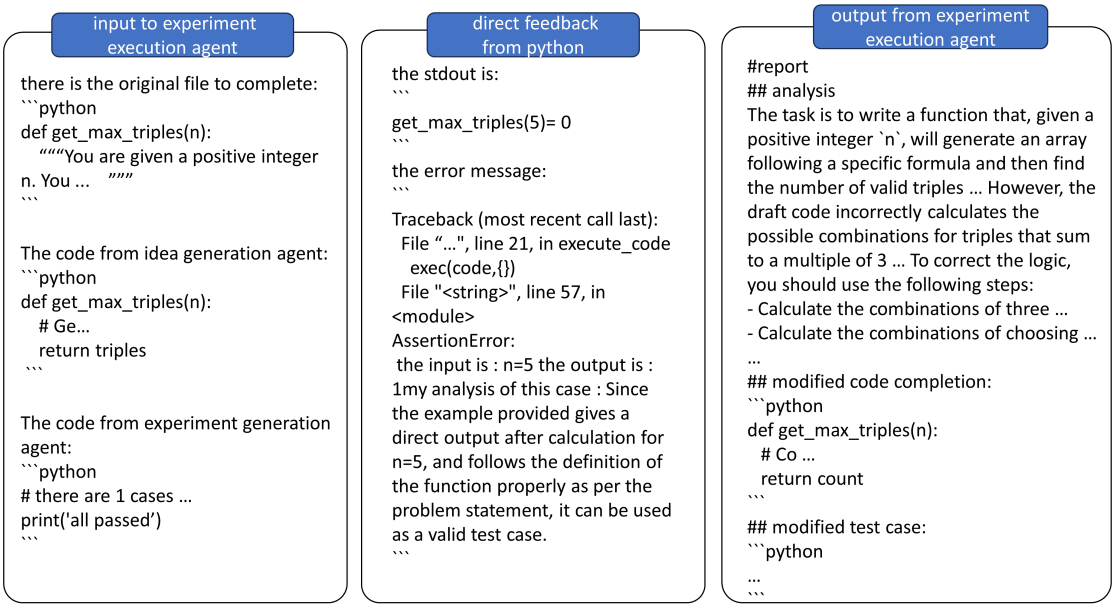

Figure 10: then the experiment excuted in a python interpreter , and give stdout , error message , the high light part of the error message is added by the experiment agent . this part is augmented feedback of current code draft .

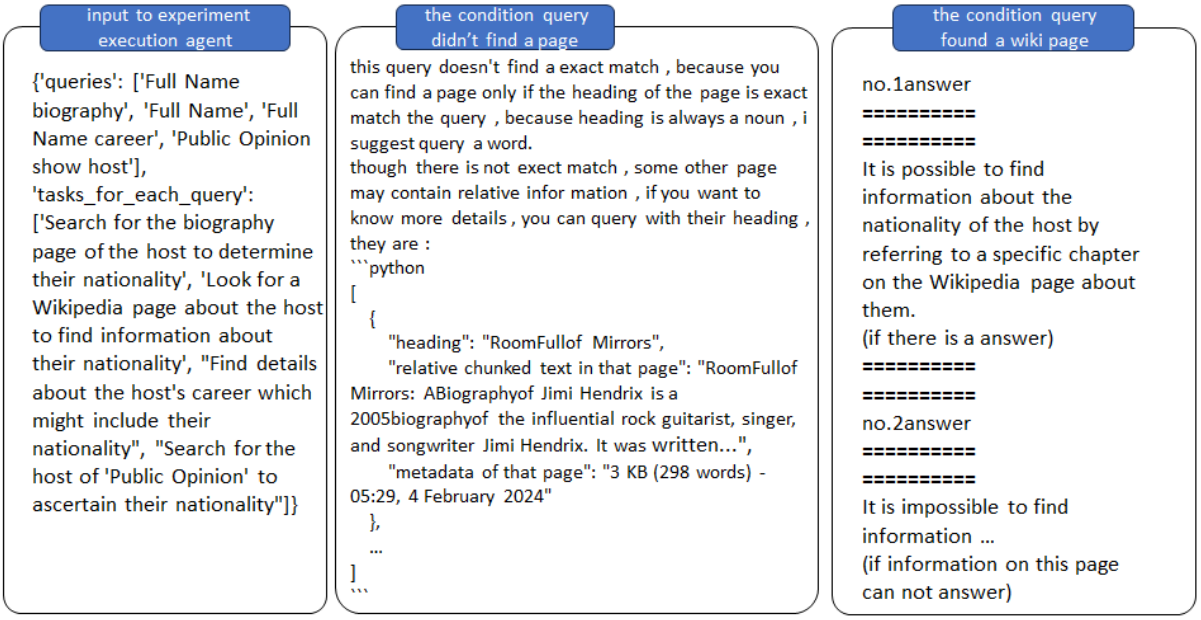

Figure 11: Input output example for experiment execution agents in HotpotQA

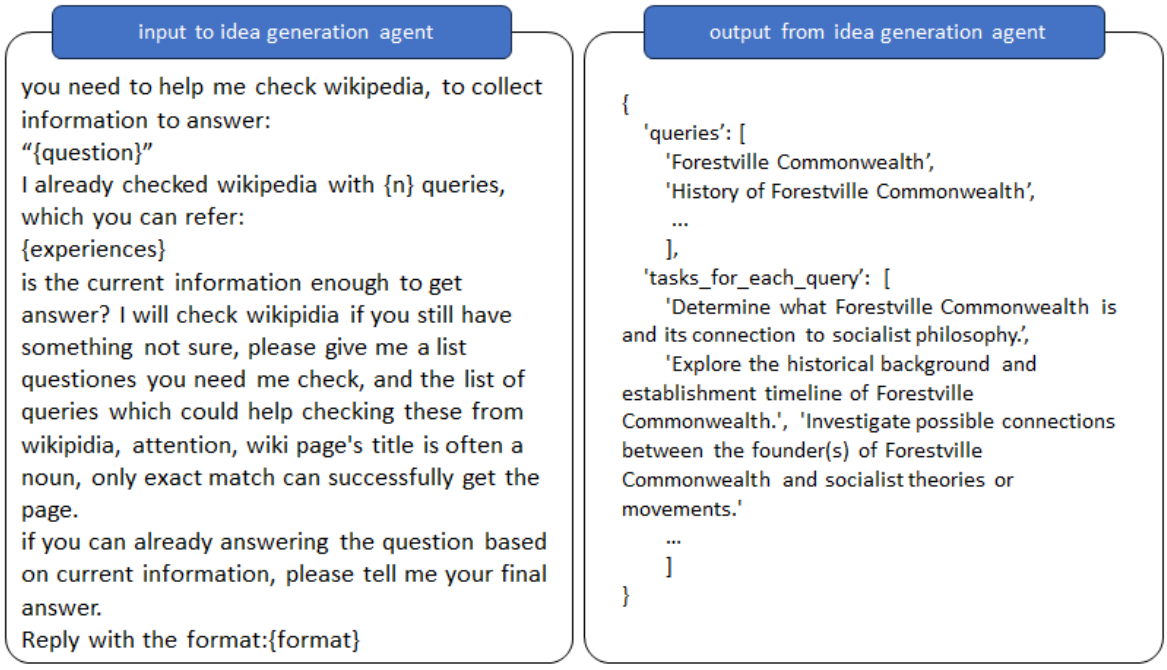

Figure 12: Input output example for idea generation agents in HotpotQA