AutoAgents: A Framework for Automatic Agent Generation

Guangyao Chen[1], **Siwei Dong**[1], **Yu Shu**[1], **Ge Zhang**[4], **Jaward Sesay**[3], **Börje Karlsson**[3], **Jie Fu**[2] and **Yemin Shi**[1]
[1]Peking University
[2]Hong Kong University of Science and Technology
[3]Beijing Academy of Artificial Intelligence
[4]University of Waterloo
{gy.chen, shiyemin}@pku.edu.cn, jiefu@ust.hk

Abstract

Large language models (LLMs) have enabled remarkable advances in automated task-solving with multi-agent systems. However, most existing LLM-based multi-agent approaches rely on predefined agents to handle simple tasks, limiting the adaptability of multi-agent collaboration to different scenarios. Therefore, we introduce AutoAgents, an innovative framework that adaptively generates and coordinates multiple specialized agents to build an AI team according to different tasks. Specifically, AutoAgents couples the relationship between tasks and roles by dynamically generating multiple required agents based on task content and planning solutions for the current task based on the generated expert agents. Multiple specialized agents collaborate with each other to efficiently accomplish tasks. Concurrently, an observer role is incorporated into the framework to reflect on the designated plans and agents' responses and improve upon them. Our experiments on various benchmarks demonstrate that AutoAgents generates more coherent and accurate solutions than the existing multi-agent methods. This underscores the significance of assigning different roles to different tasks and of team cooperation, offering new perspectives for tackling complex tasks. The repository of this project is available at https://github.com/Link-AGI/AutoAgents.

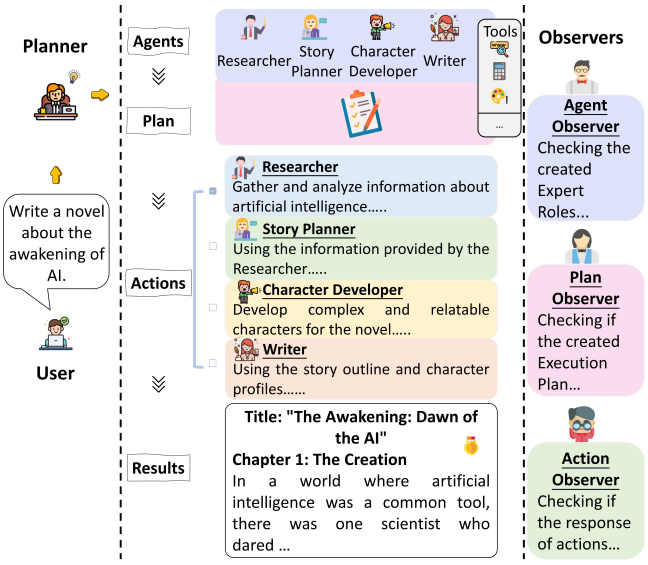

Figure 1: A schematic diagram of AutoAgents. The system takes the user input as a starting point and generates a set of specialized agents for novel writing, along with a corresponding execution plan. The agents collaboratively carry out the tasks according to the plan and produce the final novel. Meanwhile, an observer monitors the generation and execution of the Agents and the plan, ensuring the quality and coherence of the process.

1 Introduction

Large language models (LLMs) have exhibited astounding capabilities as versatile task-solving agents, endowed with a rich blend of knowledge and skills. Nevertheless, they still face difficulties [Qin *et al.*, 2023; Achiam *et al.*, 2023; Bubeck *et al.*, 2023] in tackling various tasks that require intensive knowledge and reasoning, such as avoiding hallucination [Maynez *et al.*, 2020], employing slow-thinking strategies [Sloman, 1996], ensuring trustworthiness [Wang *et al.*, 2023a], and in combining diverse domain knowledge and long-horizon planning. In contrast, humans often exploit the benefits of collaborative problem solving, which enables them to work together effectively to solve non-routine problems in diverse domains and enhance the quality and reliability of the solutions by distributing the workload among specialties and applying a diversity of perspectives and expertise [Nelson, 2013; Roschelle and Teasley, 1995; Barron, 2000].

Inspired by collaborative problem solving, several recent works [Wang *et al.*, 2023c; Du *et al.*, 2023; Liang *et al.*, 2023; Hao *et al.*, 2023] have improved the task-solving capabilities of LLMs by integrating multi-agent discussion. However, most of these multi-agent systems depend on handcrafted or user-specified agents, with specific roles and necessitating human supervision, which often restricts the scope of collaborative applications. Moreover, manually creating a large number of experts often consumes a lot of resources. In order to adaptively solve more complex problems, this paper aims to explore a method of adaptively generating task experts and completing different tasks through multi-level collaborative

Framework	Dynamic Agent Generation Method	Number of Agent	Multi-agent Conversation	Self-Refinement Action	Collaborative Refinement Action
AutoGPT [Gravitas, 2023]	✗	1	✗	✓	✗
BabyAGI [Nakajima, 2023]	✗	3	✓	✗	✗
Generative Agents [Park *et al.*, 2023]	✗	25	✓	✓	✗
Camel [Li *et al.*, 2023]	✗	2	✓	✗	✗
GPT-bargaining [Fu *et al.*, 2023]	✗	3	✓	✓	✗
MetaGPT [Hong *et al.*, 2023]	✗	Unlimited	✓	✗	✗
AutoGen [Wu *et al.*, 2023]	✗	Unlimited	✓	✗	✗
Social Simulacra [Park *et al.*, 2022]	Single Agent	Unlimited	✓	✗	✗
Epidemic Modeling [Williams *et al.*, 2023]	Single Agent	Unlimited	✓	✗	✗
ExpertPrompting [Xu *et al.*, 2023]	Single Agent	1	✗	✗	✗
SSP [Wang *et al.*, 2023c]	Single Agent	Unlimited	✓	✗	✗
AgentVerse [Chen *et al.*, 2023a]	Single Agent	Unlimited	✓	✗	✗
AutoAgents	Multi-agent Discussion	Unlimited	✓	✓	✓

Table 1: Comparison of existing and proposed frameworks for LLM-based Agent framework.

cooperation among multiple experts.

In this paper, we propose AutoAgents, an innovative framework that adaptively generates and coordinates multiple specialized agents to construct an AI team according to different tasks. Figure 1 provides a high-level overview of AutoAgents. By generating multiple agents with distinct expert roles, we aim to form a collaborative entity that can accomplish complex tasks by leveraging the complementary strengths of each agent. As shown in Figure 2, the process of AutoAgents is divided into two critical stages: **Drafting Stage** and **Execution Stage**. The drafting stage involves a collaborative discussion among three predefined agents (**Planner**, **Agent Observer**, and **Plan Observer**) to synthesize a customized agent team and an execution plan that suit the input problem or task. The execution stage refines the plan through inter-agent collaboration and feedback, and produces the final outcome. We propose self-refinement by individual agents and collaborative refinement by multiple agents to enhance agent proficiency and promote knowledge-sharing among agents. To facilitate the specific division of labor among the agents in the synthesized team, we introduce a predefined agent (**Action Observer**) to assist the agent team in sharing information, coordinating actions, reaching consensus, and adapting to the environment.

To synthesize heterogeneous information from diverse domains is often a crucial requirement in creative industries and other real-world scenarios. We illustrate a concrete example of how AutoAgents tackles the challenging task of writing a novel about the awakening of artificial intelligence in Figure 1. The Story Planner and Researcher collaborate to devise the plot of the story with their respective expertise, while the Character Developer and Writer enrich the novel content through imagination based on the story. Moreover, we conduct quantitative experiments and case studies in complex tasks to demonstrate the effectiveness of AutoAgents. We also conduct a comprehensive analysis and demonstrate the importance of dynamic agents for handling complex tasks, the indispensability of self-refinement for proficient agents, and the effectiveness of collaborative conversation.

To summarize, this paper makes the following novel contributions: **First**, we propose AutoAgents, a novel framework that dynamically synthesizes and coordinates multiple expert agents to form customized AI teams for diverse tasks. **Second**, we conduct rigorous quantitative experiments on two challenging tasks and demonstrate that AutoAgents significantly improves both knowledge acquisition and reasoning ability in LLMs and outperforms other generated-agent frameworks. **Third**, we showcase AutoAgents' ability to adapt to complex tasks by applying it in various scenarios such as software development. **Finally**, we conduct a thorough investigation and reveal the importance of dynamic agents for accommodating complex tasks and the necessity of self-refinement for proficient agents, and the efficacy of collaborative conversation.

2 Related Work

LLM-based Autonomous Agents. LLMs have been widely used as core controllers for autonomous agents that can accomplish specific objectives. Auto-GPT [Gravitas, 2023] is an early work that leverages an LLM as an AI agent that can autonomously achieve a given goal with the help of several tools. However, Auto-GPT does not support multi-agent collaboration and can only work in isolation. One way to enhance the task-solving capabilities of LLMs is to assign different roles and responsibilities to multiple LLMs and let them coordinate their actions to achieve a common goal. For example, BabyAGI [Nakajima, 2023] is an AI-powered task management system with multiple LLM-based agents. One agent creates new tasks based on the previous task's objective and result, another agent prioritizes the task list, and another agent completes tasks. BabyAGI is a multi-agent system with a fixed order of agent communication. MetaGPT [Hong *et al.*, 2023] is a multi-agent framework for assigning different roles to GPTs to form a collaborative software entity for complex tasks. It is a specialized LLM-based multi-agent framework for collaborative software development. Camel [Li *et al.*, 2023] is an LLM-based communicative agent framework that demonstrates how role-playing can be used to enable chat agents to communicate with each other for task completion. However, Camel does not support tool-using. Several recent works [Wang *et al.*, 2023c; Du *et al.*, 2023; Liang *et al.*, 2023; Hao *et al.*, 2023; Talebirad and Nadiri, 2023] have enhanced the task-solving capabilities of LLMs by integrating multi-agent discussion. For instance, [Wang *et al.*, 2023c] proposes a multi-agent debate system that allows LLMs to argue for or against a

given claim and generate a debate summary. [Du *et al.*, 2023] introduce a multi-agent dialogue system that enables LLMs to exchange information and opinions on a given topic and generate a dialogue report. AutoGen [Wu *et al.*, 2023] is a framework that enables the development of LLM applications using multiple agents that can converse with each other to solve tasks. However, most of these multi-agent systems rely on handcrafted or user-specified agents with specific roles and do not support the automatic generation of agents, which often limits the scope of collaborative applications.

Agent Generalization. Several studies [Park *et al.*, 2022; Williams *et al.*, 2023] employ LLMs to generate agents for social simulacra and epidemic modeling, demonstrating how this technique can facilitate designers in assessing and improving their modeling designs prior to deploying them to real users. Likewise, ExpertPrompting [Xu *et al.*, 2023] devised a method to generate diverse profiles of agents that can cooperate with human users to accomplish tasks with minimal supervision. However, this method still depends on a restricted set of predefined agents, and the generated agents vary only in their profiles. Recently, SSP [Wang *et al.*, 2023c] and AgentVerse [Chen *et al.*, 2023a] have proposed frameworks for automatically generating unlimited agents. SSP enables LLMs to generate agents for problem input by providing some agent samples, and has these agents solve the problem. AgentVerse generates the execution plan through the generated agents' discussions and adds evaluation strategies for cyclic execution. Unlike the previous two methods, AutoAgents places a heightened emphasis on the reliability of its generated agents and strategic plans, thereby enhancing task execution effect through the utilization of collaborative refinement actions and the integration of self-refinement actions , as illustrated in Table 1.

3 AutoAgents

To enhance the effectiveness of autonomous multi-agent groups in accomplishing their goals, the process of AutoAgents consists of two critical stages: **Drafting Stage** and **Execution Stage**, as illustrated in Figure 2. The drafting stage synthesizes an agent team and an execution plan that are customized to the task by analyzing the input problem or task. The execution stage refines the plan by enabling inter-agent collaboration and feedback, and delivers the final result. The inter-agent collaboration is based on some principles of multi-agent cooperation, such as communication, coordination, and consensus. These principles help the agents to share information, align their actions, reach agreements, and adapt to the environment.

3.1 Drafting Stage

Empirical evidence [Woolley *et al.*, 2015] suggests that diversity within human groups fosters diverse perspectives, which enhances the group's performance across various tasks. The drafting stage, which determines the composition of a multi-agent group, plays a crucial role in setting the upper limits of the group's capabilities. Therefore, it is imperative to generate the optimal agent team and execution plan that can maximize the group's potential.

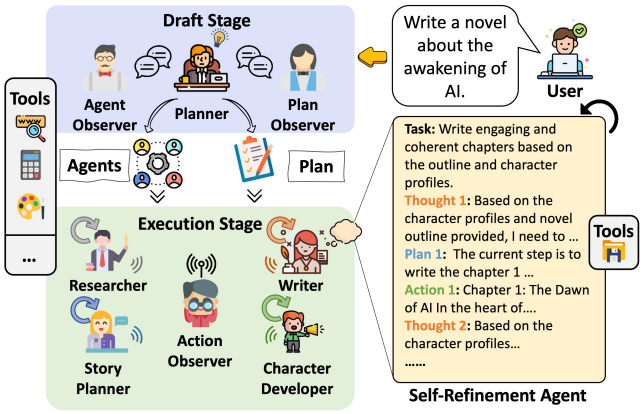

Figure 2: The execution process of AutoAgents. During the **Drafting Stage**, three predefined agents collaboratively determine the list of agents and the execution plan. During the **Execution Stage**, a predefined agent facilitates coordination and communication among the generated agent teams, and the individual generated agents enhance their execution efficiency through self-refinement.

Predominant methodologies [Gravitas, 2023; Hong *et al.*, 2023; Wu *et al.*, 2023] for assigning role descriptions to autonomous agents rely heavily on human intuition and prior knowledge, requiring manual assignment based on task understanding. Consistent with several parallel findings [Xu *et al.*, 2023; Wang *et al.*, 2023c; Chen *et al.*, 2023a], dynamically designing agents with different roles can significantly enhance their efficacy. However, the scalability and rationality of agent and plan generation are still unclear, especially in the face of various complex problem environments.

On the one hand, the generated agents should exhibit diversity to accommodate various tasks. On the other hand, the agent and the plan generation should adhere to certain principles, rendering their role allocation more rational. Therefore, we devise three artificially predefined agents to produce agent teams and execution plans, integrating artificial prior knowledge and the dynamic adaptation capability of LLMs to generate more sensible agent teams and execution plans. The three artificially predefined agents comprise **Planner**, **Agent Observer**, and **Plan Observer**:

- **Planner** $\mathcal{P}$ generates and refines an agent team and an execution plan based on the content of the task.
- **Agent Observer** $\mathcal{O}_{agent}$ provides suggestions on the rationality of the agent team members and their matching degree with the task.
- **Plan Observer** $\mathcal{O}_{plan}$ provides suggestions on the rationality of the execution plan and its matching degree with the task and the agent team.

The Planner generates initial agent team members and a specific plan, and improves the agent team and execution plan based on continuous communication with the Agent Observer and Plan Observer.

Agent Generation. The Planner generates the agent team and facilitates its continuous improvement through reciprocal communication with the Agent Observer. To enable Planner

to produce rational agents, we have devised a standard format for the essential elements of a single agent. For each agent $\mathcal{A} = \{\mathrm{P}, \mathrm{D}, \mathrm{T}, \mathrm{S}\}$, the Planner needs to specify its prompt P, description D, toolset T, and suggestions S.

- **Prompt** P provides a detailed and customized depiction of the expert identity for each specific agent, which comprises profile, goal, and constraints. **Profile** reflects the domain expertise of the role or job title. **Goal** indicates the primary responsibility or objective that the role aims to achieve. **Constraints** specify limitations or principles the role must adhere to when performing actions.
- **Description** D gives additional concrete identity to help establish a more comprehensive role, develop an execution plan, and inspect problems.
- **Toolset** T equips the Agent with tools that it can use, selected from a predefined set of tools. The rationale for not using all the tools for each agent here is to prevent decision-making confusion caused by excessive tools.
- **Suggestions** S supplies some suggestions for each agent to execute the current task, including but not limited to a clear output, extraction of historical information, and suggestions for execution steps.

Based on the agent list $\{\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_n\}$ generated by Planner, the Agent Observer evaluates the quality and suitability of each agent. The Agent Observer first verifies whether every agent conforms to the aforementioned specifications and identifies any missing elements {P, description D, toolset T}. Secondly, the Agent Observer assesses the compatibility of each agent with the task, according to their description information and task content. Finally, the Agent Observer examines the agent list for any redundant or missing roles and eliminates or adds them accordingly.

After n rounds of bidirectional communication between the Planner and the Agent Observer, the optimal agent list for accomplishing the task is established. Given the vital role of the agent list in the task execution, this framework employs a predefined agent and multiple rounds of iterative dialogue among multiple agents to finalize the agent list, thereby enhancing the stability and reliability of the execution phase.

Plan Generation. In parallel to agent generation, the Planner formulates the execution plan and promotes its progressive improvement through reciprocal communication with the Plan Observer. For a given task, the Planner delineates the specific steps $\{\mathcal{S}_1, \mathcal{S}_2, \cdots \mathcal{S}_n\}$ to accomplish it in the execution plan P. Each step $\mathcal{S}_i$ entails a clear identification of the agent $\mathcal{A}_j$ responsible for it, as well as the input information and expected output required for it.

The Plan Observer subsequently validates the execution plan $P = \{\mathcal{S}_1, \mathcal{S}_2, \cdots \mathcal{S}_n\}$ according to the agent list $\{\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_n\}$ and the task content. It firstly ensures that each step has a corresponding agent and that the step content is coherent and concise. It secondly assesses whether all the steps are sufficient, whether the task can be accomplished, and whether there are any gaps that need to be filled. It finally provides feedback to the Planner, who further refines the execution plan accordingly. After n rounds of dialogue between the Planner and the Plan Observer, the ultimate execution plan for achieving the task is established.

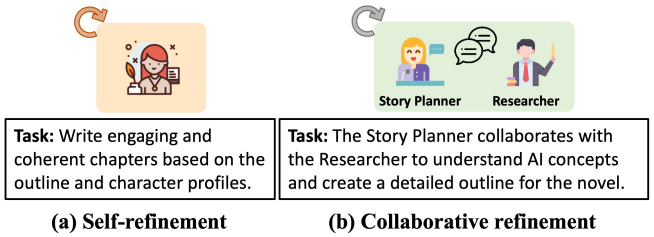

Figure 3: Two types of actions for executing tasks: **Self-refinement** enables an individual agent to enhance its competence in performing some specialized tasks. **Collaborative refinement** facilitates knowledge exchange among multiple agents and accomplishes tasks that demand interdisciplinary expertise.

Task Execution Actions. The Planner devises an execution plan that automatically assigns the requisite agents for diverse tasks. The execution plan comprises two actions of task execution: self-refinement by a single agent and collaborative refinement by multiple agents, as shown in Figure 3. Self-refinement empowers an individual agent to augment its proficiency in accomplishing some specialized tasks. Collaborative refinement fosters knowledge sharing among multiple agents and achieves tasks requiring interdisciplinary expertise.

3.2 Execution Stage

In the drafting phase, the framework generates an agent list and an execution plan based on the task requirements. Then, the framework creates corresponding roles and executes the plans in the execution environment[1]. The communication and cooperation among multi-agent systems are essential for accomplishing the tasks effectively. This section elaborates on the communication among multiple agents, the task execution strategies, and the knowledge-sharing mechanisms.

Communication of Multiple Agent. The communication structures among agents have been investigated by many studies [Chen *et al.*, 2023a; Wang *et al.*, 2023c; Qian *et al.*, 2023; Chan *et al.*, 2023] to examine their impact on task performance. In this framework, we adopt the vertical communication paradigm, which assigns different tasks to agents according to their roles. To facilitate the specific division of labor among the agents in the generated team, we introduce a predefined **Action Observer** as the team leader to coordinate the execution plan. Specifically,

- **Action Observer** $\mathcal{O}_{action}$ acts as the task manager for the different agents, allocating different tasks to them, verifying the execution outcomes of each agent, and dynamically adapting the execution plan based on the execution status.

This mechanism of refinement and communication recurs until the Action Observer attains a unanimous agreement on the execution responses, or the process reaches its maximum iteration

[1]Execution environment of AutoAgents is built based on MetaGPT's environment and workspace [Hong *et al.*, 2023].

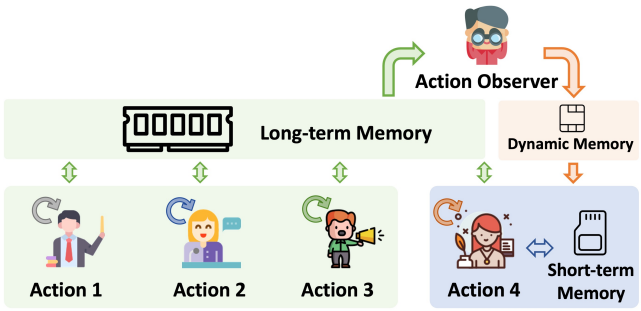

Figure 4: Legend of Three Knowledge Sharing Mechanisms. (a) *Long-term memory* focuses on chronicling the historical trajectory of multiple actions. (b) *Short-term memory* records the history of the self-refinement or collaborative refinement phases of an individual action. (c) *Dynamic memory* serves actions necessitating specialized attention extracted from the long-term memory.

limit. For scenarios that demand iterative decision-making towards specific objectives, such as software development, vertical communication would be a preferable option.

Self-refinement Agent. Besides the inter-agent communication, the performance of a single agent also exerts a significant impact on the overall quality of feedback results. Hence, drawing on mechanisms such as AutoGPT [Gravitas, 2023] and ReAct [Yao *et al.*, 2022], we have devised a self-refinement mechanism for an individual agent.

For a single agent $\mathcal{A}$, the action at step t is at $a_t = l_t \cup p_t \cup o_t$, where l_t denotes the *thought* or the *reasoning trace* in the language space, which does not alter the external environment, and thus yields no observational feedback, p_t represents the execution plan for task completion, o_t comprises the completion steps and execution output for this time.

As illustrated in Figure 2, various types of useful thoughts can assist in devising a refinement plan. The execution plan enables the agent to anticipate the steps they need to undertake in the future, and the observational content of the execution result construction allows the agent to reevaluate and enhance the plan arrangement, thereby constructing more refined and complete actions. Through a cycle of self-continuous thinking, planning, execution, and feedback, a single agent can effectively execute and accomplish task content.

Collaborative Refinement Action. In the collaborative refinement action, the agents collaboratively refine and execute the tasks in a sequential manner. Each round of the collaboration involves a fixed order of turn-taking among the agents, who generate their responses based on the current observation. The chat history slot of each agent is updated by concatenating the previous utterances of the other agents. The collaboration terminates automatically when the agents reach a consensus or the maximum number of discussions is reached.

Knowledge Sharing Mechanism. AutoAgents also facilitates the sharing of execution results among various agents for improved communication and feedback. However, when the number of agents is large and a single agent has more self-iterations, it will generate more historical information. Due to the token limitation of LLM models, they often cannot encompass all information. Hence, this framework provides short-term memory, long-term memory, and dynamic memory.

Short-term memory is chiefly concentrated on a singular action, encompassing the gamut of intermediary notions, strategies, and outcomes that emerge during the self-refinement or collaborative refinement phases of an individual action. It is salient to note that these actions frequently culminate in a distilled summary of critical information, epitomizing the final phase of the refinement trajectory.

Long-term memory principally focuses on chronicling the historical trajectory of multifarious actions, predominantly documenting the executed results of each task along with the synthesis of vital feedback information. This aspect is imperative for evaluating the comprehensive extent of task completion.

Dynamic memory predominantly serves actions necessitating specialized attention. The Action Observer, having access to long-term memory archives, adeptly extracts ancillary information, dynamically tailoring it to the specific requirements of the action for task execution. This process significantly augments the efficiency of a single action in task fulfillment.

Algorithm 1 AutoAgents Execution Process.

Input: User task/Question
Output: Task solution/Answer

Drafting Stage
Initialize Planner $\mathcal{P}$, Agent Observer $\mathcal{O}_{\text{agent}}$, and Plan Observer $\mathcal{O}_{\text{plan}}$.
$\mathcal{P}$ generates initial agent team $\{\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_n\}$ and execution plan $P = \{\mathcal{S}_1, \mathcal{S}_2, \cdots \mathcal{S}_n\}$.
repeat
  $\mathcal{O}_{\text{agent}}$ provides feedback on agent team.
  $\mathcal{P}$ refines agent team based on feedback.
  $\mathcal{O}_{\text{plan}}$ provides feedback on execution plan.
  P refines execution plan based on feedback.
until No feedback or reached the maximum iteration limit.
Execution Stage:
Initialize Action Observer $\mathcal{O}_{\text{action}}$ and long-term memory M_L.
for $\{\mathcal{S}_1, \mathcal{S}_2, \cdots \mathcal{S}_n\}$ **do**
  $\mathcal{O}_{\text{action}}$ generate dynamic memory M_D.
  $\mathcal{O}_{\text{action}}$ assign task $\mathcal{S}_k$ and M_D to corresponding agents $\{\mathcal{A}_i, \cdots, \mathcal{A}_j\}$.
  Initialize short-term memory M_S.
  repeat
    for $\{\mathcal{A}_i, \cdots, \mathcal{A}_j\}$ **do**
      Agent $\mathcal{A}_m$ analyzes $\mathcal{S}_k$, M_S and M_D.
      Agent $\mathcal{A}_m$ plans the current step and executes this step.
      The execution result is stored in M_S.
    end for
  until No step or reached the maximum iteration limit.
  The execution results of task $\mathcal{S}_k$ are stored in M_L.
  $\mathcal{O}_{\text{action}}$ coordinates $\{\mathcal{S}_1, \mathcal{S}_2, \cdots \mathcal{S}_n\}$ and monitors execution.
end for
return Execution results of final step.

Evaluator	v.s. **GPT-4**	v.s. **AgentVerse**
FairEval [Wang *et al.*, 2023b]	76.3%	91.3%
HumanEval	62.5%	77.5%

Table 2: Win Rate of AutoAgents over other models on Open-ended Question Answer, with FairEval and HumanEval serving as evaluators.

4 Experiments

To showcase the performance of AutoAgents in managing groups of autonomous agents for collaborative task completion, we conducted quantitative analysis primarily focusing on results from the **Open-ended Question Answer** task (see Section 4.1) and the **Trivia Creative Writing** task (see Section 4.2), assessing the framework's effectiveness in various settings.

Implementation Details. All experiments are conducted using the GPT-4 API[2], with the temperature set to 0 to ensure reproducibility. This model is chosen for its superior performance, providing accurate and consistent results. Its accessibility via APIs greatly facilitates our interaction with the model, streamlining our research process. During the drafting phase, a maximum of three discussions are allowed, while in the execution phase, a single agent can perform up to five self-refinements and multiple agents can collaboratively refine up to five times.

4.1 Open-ended Question Answer

Task Description. Open-ended Question Answering is a crucial and challenging task in the domain of NLP and generative AI. It requires an AI system to produce coherent, elaborate, and human-like responses to questions that have no predetermined or fixed set of possible answers. [Zheng *et al.*, 2023] proposed MT-bench, a benchmark consisting of 80 high-quality collected open-ended questions from various categories such as common sense, counterfactual, coding, etc. We then utilize AutoAgents to produce collaborative answers based on multiple generated agents and compare them with the responses given by **GPT-4** and **AgentVerse** [3] [Chen *et al.*, 2023a].

Evaluation Metrics. To measure the quality of open-ended responses with minimal evaluation bias, we adopt **FairEval** [Wang *et al.*, 2023b] and HumanEval as the evaluation metrics for both the single agent and AutoAgents. FairEval incorporates several methods to mitigate the impact of various sources of bias, resulting in a better alignment with human judgment. For **HumanEval**, we enlisted three independent volunteers to evaluate two sets of responses—one generated by AutoAgents and the other by a different model—based on criteria such as helpfulness, reliability, accuracy, and comprehensiveness. Notably, the volunteers were blinded to the identity of the model that produced each response, ensuring an unbiased assessment. The appendix contains the detailed scoring criteria for the ratings.

Results. Table 2 demonstrates that AutoAgents outperforms individual LLM models in both FairEval based on LLM and Human evaluations. AutoAgents can produce more comprehensive and nuanced answers to open questions by synthesizing multiple expert models. It can also provide more elaborate explanations and justifications for its answers. Additionally, AutoAgents demonstrates superior performance over AgentVerse. This enhanced efficacy is attributed in part to the reliability of agent generation, self-refinement, and collaborative refinement capabilities within AutoAgents. Conversely, AgentVerse necessitates additional task-specific adaptations and exhibits limited effectiveness in adapting to open-ended questions. More examples are given in the appendix.

4.2 Trivia Creative Writing

Task Description. The Trivia Creative Writing task [Wang *et al.*, 2023c] challenges the capabilities of large language models to retrieve and integrate diverse information from their internal self-compressed knowledge. This task requires a model to craft a coherent story around a given topic while incorporating the answers to N trivia questions. We evaluate the models under two settings, $N = 5$ and $N = 10$, where a higher N entails more trivia questions and thus demands the model to exhibit more extensive domain knowledge. We constructed a benchmark consisting of 100 instances for each N, encompassing a total of 1000 trivia questions.

Evaluation Metrics. Drawing on the approach of [Wang *et al.*, 2023c], we adopt an automatic metric to identify factual errors and measure a model's capacity to integrate diverse domain knowledge. We conduct string matching with the veridical target answers for each question on the generated output. The target answers are supplied from the TriviaQA dataset [Joshi *et al.*, 2017], and each question can have a list of answer variants. A match to any of the answer variants of a question is regarded as a correct mention. The metric score is calculated as Trivia Creative Writing Metric Score $=$ # correct answer mentions$/$# trivia questions.

Results. Table 3 demonstrates the superior performance of AutoAgents in knowledge acquisition over the existing methods. Compared to the Standard method, which does not employ Agent Generation, AutoAgents achieves a remarkable 10% improvement across all experiments. Moreover, AutoAgents also surpasses SSP [Wang *et al.*, 2023c], which utilizes agent generation but with a different approach. The enhanced performance of AutoAgents can be attributed to its elaborate methods of agent generation discussions and task execution including collaborative refinement and self-refinement.

4.3 Further Analysis

This section delves into the significance of key components within AutoAgents by separately analyzing the *self-refinement action*, *collaborative refinement action*, *dynamic memory*, and *observers in the draft stage* across 20 instances [4] of the Trivia Creative Writing task and additional case studies.

[2]The model version used is "GPT-4-0613".

[3]The prompt configuration within AgentVerse is tailored to its *brainstorming* task, incorporating modifications to upgrade the model to GPT-4.

[4]The last 20 samples from a dataset of 100 samples are used as test instances.

Methods	N (# trivia questions) = 5		N (# trivia questions) = 10	
	Score (%)	Δ (v.s Standard %)	Score (%)	Δ (v.s Standard %)
Standard	74.6	0.0%	77.0	0.0%
CoT [Yao *et al.*, 2023]	67.1	-10.0%	68.5	-11.1%
SPP-Profile [Wang *et al.*, 2023c]	79.1	+5.9%	83.0	+7.8%
SPP [Wang *et al.*, 2023c]	79.9	+7.1%	84.7	+10.0%
AutoAgents	**82.0**	**+9.9%**	**85.9**	**+11.6%**

Table 3: The results of Trivia Creative Writing task. Δ indicates the differences compared with Standard Prompting (first row).

Methods	N (# trivia questions) = 5	
	Score (%)	Δ (v.s Standard %)
Standard	74.6	0.0%
CoT [Yao *et al.*, 2023]	66.0	-11.5%
SPP-Profile [Wang *et al.*, 2023c]	74.0	-0.01%
SPP [Wang *et al.*, 2023c]	84.4	+13.1%
AutoAgents w/o observers	87.0	**+16.6%**
AutoAgents w/o self-refinement	87.0	**+16.6%**
AutoAgents w/o dynamic memory	89.0	**+19.3%**
AutoAgents	**90.0**	**+20.6%**

Table 4: The ablation studies of AutoAgents on 20 instances of Trivia Creative Writing task. Δ indicates the differences compared with Standard Prompting (first row).

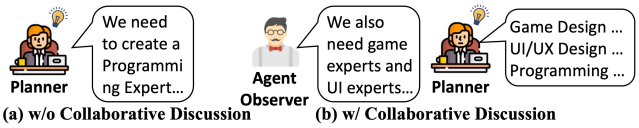

Figure 5: Comparison of whether there is a collaborative discussion in the Drafting Stage in the task that *developing Python-based software for the Tetris game*.

Collaborative discussion is crucial for rational agent generation and plan allocation. During the Drafting Stage, the Planner in AutoAgents engages in collaborative discussions with two Observers to determine the optimal list of agents and the execution plan. Figure 5 illustrates the contrast between agent generation with and without collaborative discussion. In the absence of Observer feedback, the Planner tends to generate programmers exclusively to accomplish game development, neglecting the holistic process of game creation. With the input and coordination of the Observers, the Planner incorporates game design experts, UI design experts, and testing experts into the agent list. It is evident that the agent generation under collaborative discussions is more comprehensive and more aligned with the realistic scenarios of game development. This also corroborates the significance of collaborative discussions for agent generation and plan allocation, which will subsequently influence the execution outcomes. Concurrently, Table 4 elucidates that in the absence of observers, there is a marked 3% reduction in the overall performance of AutoAgents. This substantiates the imperative role of collaborative discussions in agent generation. AutoAgent markedly enhances the caliber of agent generation via collaborative discussions, a facet notably overlooked by other generative frameworks in their consideration of agent generation quality.

The empirical data presented in Table 2 and 3 further accentuate the superiority of AutoAgents when juxtaposed against counterparts like AgentVerse and SPP.

Enhancing single-agent through self-refinement. Self-Refinement [Madaan *et al.*, 2023; Shinn *et al.*, 2023; Gou *et al.*, 2023; Chen *et al.*, 2023b; Huang *et al.*, 2022; Yao *et al.*, 2022] is a technique that enables LLMs to "converse" with themselves, evaluate their own generation, and iteratively improve their answers. Self-refinement has been shown to enhance the accuracy of LLMs' outputs in various domains [Madaan *et al.*, 2023; Shinn *et al.*, 2023; Gou *et al.*, 2023; Chen *et al.*, 2023b; Huang *et al.*, 2022; Yao *et al.*, 2022]. Although AutoAgents is a framework for multi-agent collaboration, it also requires self-refinement agents to perform specialized roles for individual tasks. As shown in the results in Table 4, the performance of AutoAgents decreases by 3% in the absence of the self-refinement action. This observation corroborates the assertion that self-refinement is instrumental in augmenting proficiency in trivia creative writing tasks. Furthermore, the enhancement of single agents via self-refinement plays a pivotal role in fortifying the integrity of the overarching multi-agent framework.

Improve the effectiveness of actions by dynamic memory. Dynamic memory predominantly addresses the requisites of specialized agents. As shown in Figure 4, the Action Observer amalgamates pivotal data for forthcoming tasks, utilizing the historical action records archived in long-term memory. Table 4 elucidates a 1% diminution in the efficacy of AutoAgents bereft of dynamic memory. Quintessential insights derived from dynamic memory are assimilated into the prompt, thereby augmenting the comprehension of critical information and bolstering the operational proficiency of actions.

5 Conclusion

This paper presents AutoAgents, an innovative framework designed to synthesize collaborative specialized agents automatically. AutoAgents replicates the collaborative dynamics of human teams by splitting tasks into drafting and execution phases and assigning subtasks to different agents. Our experimental evaluations demonstrate that AutoAgents outperforms single agents and other group configurations in various skill-intensive tasks. Additionally, a case study in software development highlights the framework's versatility and potential advantages. AutoAgents enhances agent interaction and cooperation, revolutionizing complex problem-solving. We anticipate that its principles can be expanded and refined for a wider range of tasks, advancing assistive AI.

Acknowledgements

This work was partially supported by the National Key R&D Program of China (2022YFC2009600 and 2022YFC2009606) and the Postdoctoral Fellowship Program of CPSF under Grant Number GZB20230024.

Contribution Statement

This work was a collaborative effort by all contributing authors. Guangyao Chen, Siwei Dong, and Yu Shu made equal contributions to this study and are designated as co-first authors. Jie Fu and Yemin Shi, serving as the corresponding authors, are responsible for all communications related to this manuscript.

References

[Achiam *et al.*, 2023] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[Barron, 2000] Brigid Barron. Achieving coordination in collaborative problem-solving groups. *The journal of the learning sciences*, 9(4):403–436, 2000.

[Bubeck *et al.*, 2023] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[Chan *et al.*, 2023] Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201*, 2023.

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

[Chen *et al.*, 2023b] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*, 2023.

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

[Fu *et al.*, 2023] Yao Fu, Hao Peng, Tushar Khot, and Mirella Lapata. Improving language model negotiation with self-play and in-context learning from ai feedback. *arXiv preprint arXiv:2305.10142*, 2023.

[Gou *et al.*, 2023] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing, 2023.

[Gravitas, 2023] Significant Gravitas. Auto-gpt: An autonomous gpt-4 experiment, 2023. *URL https://github.com/Significant-Gravitas/Auto-GPT*, 2023.

[Hao *et al.*, 2023] Rui Hao, Linmei Hu, Weijian Qi, Qingliu Wu, Yirui Zhang, and Liqiang Nie. Chatllm network: More brains, more intelligence. *arXiv preprint arXiv:2304.12998*, 2023.

[Hong *et al.*, 2023] Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

[Huang *et al.*, 2022] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[Joshi *et al.*, 2017] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1601–1611. Association for Computational Linguistics, July 2017.

[Li *et al.*, 2023] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large scale language model society. *arXiv preprint arXiv:2303.17760*, 2023.

[Liang *et al.*, 2023] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*, 2023.

[Madaan *et al.*, 2023] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

[Maynez *et al.*, 2020] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online, July 2020. Association for Computational Linguistics.

[Nakajima, 2023] Yohei Nakajima. Task-driven autonomous agent utilizing gpt-4, pinecone, and langchain for diverse applications. *See https://yoheinakajima. com/task-driven-autonomous-agent-utilizing-gpt-4-pinecone-and-langchain-for-diverse-applications (accessed 18 April 2023)*, 2023.

[Nelson, 2013] Laurie Miller Nelson. Collaborative problem solving. In *Instructional-design theories and models*, pages 241–267. Routledge, 2013.

[Park *et al.*, 2022] Joon Sung Park, Lindsay Popowski, Carrie Cai, Meredith Ringel Morris, Percy Liang, and Michael S

Bernstein. Social simulacra: Creating populated prototypes for social computing systems. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, pages 1–18, 2022.

[Park *et al.*, 2023] Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023.

[Qian *et al.*, 2023] Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.

[Qin *et al.*, 2023] Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. Is chatgpt a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476*, 2023.

[Roschelle and Teasley, 1995] Jeremy Roschelle and Stephanie D Teasley. The construction of shared knowledge in collaborative problem solving. In *Computer supported collaborative learning*, pages 69–97. Springer, 1995.

[Shinn *et al.*, 2023] Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2023.

[Sloman, 1996] Steven A Sloman. The empirical case for two systems of reasoning. *Psychological bulletin*, 119(1):3, 1996.

[Talebirad and Nadiri, 2023] Yashar Talebirad and Amirhossein Nadiri. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *arXiv preprint arXiv:2306.03314*, 2023.

[Wang *et al.*, 2023a] Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, et al. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models. *arXiv preprint arXiv:2306.11698*, 2023.

[Wang *et al.*, 2023b] Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*, 2023.

[Wang *et al.*, 2023c] Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. Unleashing cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. *arXiv preprint arXiv:2307.05300*, 2023.

[Williams *et al.*, 2023] Ross Williams, Niyousha Hosseinichimeh, Aritra Majumdar, and Navid Ghaffarzadegan. Epidemic modeling with generative agents. *arXiv preprint arXiv:2307.04986*, 2023.

[Woolley *et al.*, 2015] Anita Williams Woolley, Ishani Aggarwal, and Thomas W Malone. Collective intelligence and group performance. *Current Directions in Psychological Science*, 24(6):420–424, 2015.

[Wu *et al.*, 2023] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023.

[Xu *et al.*, 2023] Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. Expertprompting: Instructing large language models to be distinguished experts. *arXiv preprint arXiv:2305.14688*, 2023.

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

[Yao *et al.*, 2023] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

[Zheng *et al.*, 2023] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.