# IoA: Linking Collaborative Agent Efforts with the Internet of Agents

**Anonymous ACL submission**


## Abstract

The rapid advancement of large language models (LLMs) has paved the way for the development of highly capable autonomous agents. However, existing multi-agent frameworks often struggle with integrating diverse capable third-party agents due to reliance on agents defined within their own ecosystems. They also face challenges in simulating distributed environments, as most frameworks are limited to single-device setups. Furthermore, these frameworks often rely on hard-coded communication pipelines, limiting their adaptability to dynamic task requirements. Inspired by the concept of the Internet, we propose Internet of Agents (IoA), a novel framework that addresses these limitations by providing a flexible and scalable platform for LLM-based multi-agent collaboration. IoA introduces an agent integration protocol, an instant-messaging-like architecture design, and dynamic mechanisms for agent teaming and conversation flow control. Through extensive experiments on general assistant tasks, embodied AI tasks, and retrieval-augmented generation benchmarks, we demonstrate that IoA consistently outperforms state-of-the-art baselines, showcasing its ability to facilitate efficient collaboration among heterogeneous agents. IoA represents a step towards linking diverse agents in an Internet-like environment, where agents can seamlessly collaborate to achieve greater intelligence and capabilities. We believe that this direction holds potential for better multi-agent systems.


## 1 Introduction

The Internet has revolutionized the way people collaborate and share knowledge, connecting individuals with diverse skills and backgrounds from all around the world. This global network has enabled the creation of remarkable collaborative projects, such as Wikipedia[1] and the development of the Linux operating system[2], which would have been impossible for any single person to achieve. The Internet has greatly facilitated collaboration among people, making the impossible possible and pushing the boundaries of human achievement.

The success of the Internet in enabling human collaboration raises an intriguing question: can we create a similar platform to facilitate collaboration among autonomous agents? With the rapid advancements in LLMs (OpenAI, 2023; Reid et al., 2024), we now have autonomous agents capable of achieving near-human performance on a wide range of tasks. These LLM-based agents have demonstrated the ability to break down complex tasks into executable steps, leverage various tools, and learn from feedback and experience (Qin et al., 2023; Wang et al., 2023c; Shinn et al., 2023; Qian et al., 2023b). As the capabilities of these agents continue to grow, and with an increasing number of third-party agents with diverse skills consistently emerging (Chase, 2022; Team, 2023; Significant Gravitas, 2023; Open Interpreter, 2023), it is crucial to explore how we can effectively and efficiently orchestrate their collaboration, just as the Internet has done for humans.

To address this challenge, we propose the concept of the Internet of Agents (IoA), a general framework for agent communication and collaboration inspired by the Internet. IoA aims to address three fundamental limitations of existing multi-agent frameworks (Chen et al., 2023; Wu et al., 2023; Hong et al., 2023; Qian et al., 2023a): (1) **Ecosystem Isolation**: Most frameworks only consider agents defined within their own ecosystems, potentially blocking the integration of various third-party agents and limiting the diversity of agent capabilities and the platform's generality; (2) **Single-Device Simulation**: Nearly all multi-agent frameworks simulate multi-agent systems on a single

[1] https://www.wikipedia.org/

[2] https://www.linux.org/

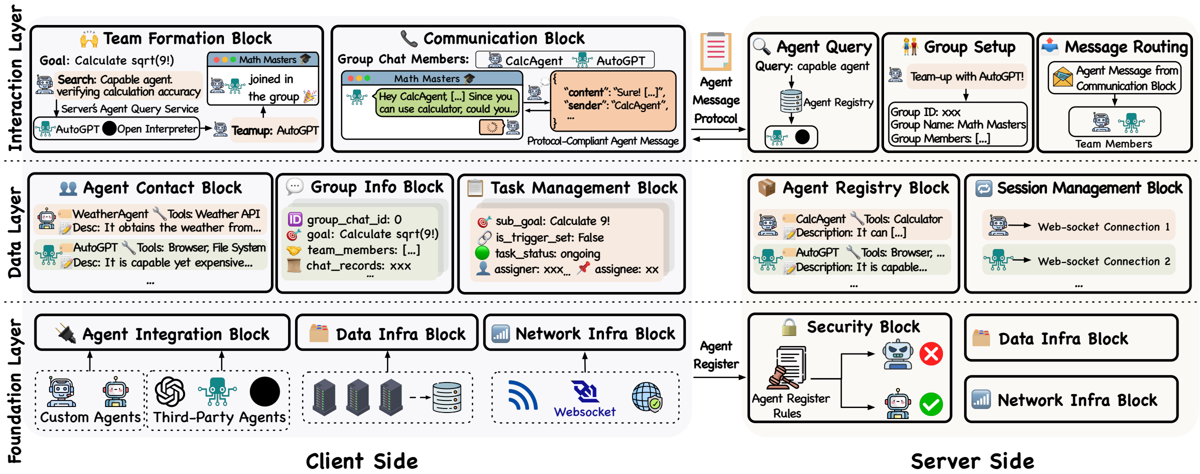



Figure 1: The illustration on the conceptual layered architecture on the design of IoA.

device, which differs significantly from real-world scenarios where agents could be distributed across multiple devices located in different places; (3) **Rigid Communication and Coordination**: The communication process, agent grouping, and state transitions are mostly hard-coded, whereas in real life, humans decide on teammates based on the task at hand and dynamically switch between discussion and task assignment or execution.

To address these limitations, we propose an agent integration protocol that seamlessly incorporates third-party agents on different devices into the framework for effective collaboration. We also introduce an instant messaging app-like framework for agent collaboration. This allows agents to autonomously find potential collaborators, form teams, and communicate within various group chats. Inspired by Speech Act Theory (Searle, 1969) and its application in conventional multi-agent systems (Finin et al., 1994; Labrou et al., 1999), we abstract several conversation states within each group chat. We provide a flexible finite-state machine mechanism enabling agents to autonomously manage conversation states, facilitating discussion and sub-task execution.

We demonstrate the effectiveness of IoA through extensive experiments and comparisons with state-of-the-art autonomous agents. By integrating AutoGPT (Significant Gravitas, 2023) and Open Interpreter (Open Interpreter, 2023), we show that IoA achieves a 66 to 76% win rate in open-domain task evaluations when compared with these agents individually. Furthermore, with only a few basic ReAct agents integrated, IoA outperforms previous works on the GAIA benchmark (Mialon et al., 2023). In the retrieval-augmented generation (RAG) question-answering domain, our framework substantially surpasses existing methods, with a GPT-3.5-based implementation achieving performance close to or even exceeding GPT-4, and effectively surpassing previous multi-agent framework.

The impressive performance of IoA across various domains highlights the potential of this paradigm for autonomous agents. As smaller LLMs continue to advance (Mesnard et al., 2024; Hu et al., 2024; Abdin et al., 2024), running agents on PC or even mobile device is becoming increasingly feasible. This trend opens up new opportunities for deploying multi-agent systems in real-world scenarios, where agents can be distributed across multiple devices and collaborate to solve complex problems. We believe that by further exploring and refining the IoA paradigm, more sophisticated and adaptable multi-agent systems can be developed, ultimately pushing the boundaries of what autonomous agents can achieve in problem-solving and decision-making.

## 2 Framework Design of IoA

In this section, we present the instant-messaging-app-like framework design of IoA, which facilitates effective collaboration among autonomous agents. The framework consists of two main components: server and client. The server is responsible for agent registration, discovery, and message routing, enabling agents running on different devices and with varying capabilities to find each other and communicate. The client acts as a wrapper for different agents, providing the necessary communication functionalities and adapting them to the specified protocol. For agents not designed for communication, an additional LLM within the client handles the communication among agents.

The architecture of both the client and server in

IoA can be structured into three layers: *Interaction Layer*, *Data Layer*, and *Foundation Layer*, as shown in Fig. 1. Each layer has specific responsibilities that contribute to the overall functionality and efficiency of the system. The message protocol between the client and the server plays a crucial role in defining the communication and collaboration mechanisms among agents, enabling information transmission and conversational state switch.

**Interaction Layer.** The Interaction Layer facilitates seamless communication and collaboration between agents, enabling them to interact, respond to ongoing tasks, and form teams as needed.

**Data Layer.** The Data Layer manages information related to agents, group chats, and tasks, organizing and maintaining the data that agents need to collaborate effectively within the framework.

**Foundation Layer.** The Foundation Layer provides the essential infrastructure for agent integration, data management, and network communication, ensuring seamless integration of agents into the system and providing robust data and network services to support their operations.

### 2.1 Client

The client side of IoA integrate and manage diverse agents, ensuring they can collaborate and communicate effectively.

At the Interaction Layer, the client facilitates dynamic communication and team formation. The Team Formation Block identifies suitable collaborators for incoming tasks, streamlining the process of forming effective teams. The Communication Block manages all ongoing group chats related to the current agent, ensuring relevant responses.

The Data Layer of the client handles critical information about agents, group chats, and tasks. The Agent Contact Block functions like a contact list, storing details about previously connected agents and pertinent notes from past interactions. The Group Info Block keeps detailed records of group chats, including histories, member details, and objectives. The Task Management Block tracks the status of all tasks within these group chats, providing agents with the necessary insights to monitor progress and make informed decisions.

At the Foundation Layer, the client ensures robust infrastructure for agent integration and data services. The Agent Integration Block outlines the protocols and interfaces required for seamless integration of third-party agents, such as **`run`**`(task_desc)` $\rightarrow$ `task_id` for task execution and **`read_memory`**`(task_id)` $\rightarrow$ `history` for retrieving task-related memory. The Data Infra Block supports data persistence, retrieval, and access control, while the Network Infra Block manages the communication between the client and the server, ensuring reliable and efficient data transmission.

### 2.2 Server

The server side of IoA manages the overall infrastructure, facilitating agent discovery, group setup, and efficient message routing, while maintaining robust data and network services.

In the Interaction Layer, the server enables agents to discover each other, initiate group chats, and communicate seamlessly. The Agent Query Block allows agents to search for others based on queried keywords, aiding effective team formation. The Group Setup Block streamlines the creation of group chats by allowing agents to specify desired teammates. The Message Routing Block ensures that messages from agents within different group chats are correctly forwarded to the corect members, maintaining the flow of communication.

The Data Layer of the server supports efficient agent queries and group communications. The Agent Registry Block stores detailed information about agents, including their accessible tools, average costs, and capability descriptions, which is crucial for the Agent Query Block to find suitable agents. The Session Management Block maintains active connection sessions between the server and client agents, ensuring that messages can be routed reliably, supporting continuous communication.

At the Foundation Layer, the server provides essential infrastructure for data management, network communication, and security. The Data Infra Block and the Network Infra Block are similar to those in the client. The Security Block plays a critical role in authentication and authorization, ensuring that only authorized agents can connect to the server and participate in communications, maintaining the integrity and security of the framework.

## 3 Key Mechanisms in IoA

In this section, we introduce the key mechanisms implemented in IoA. IoA is built upon the conceptual design presented in Section 2 and incorporates several key features that enable effective collaboration among agents with diverse capabilities. These features include Autonomous Nested Team Formation, Autonomous Conversation Flow Control,

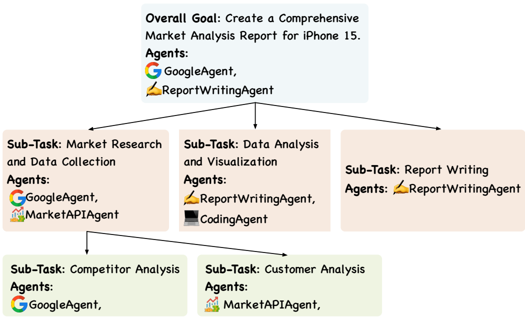



Figure 2: An example of nested team-up mechanism.

and Comprehensive Message Protocol Design. IoA aims to facilitate the dynamic formation of agent teams and streamline communication processes, ultimately enhancing the collective problem-solving capabilities of multi-agent systems.

### 3.1 Autonomous Nested Team Formation

The primary motivation behind the autonomous nested team formation mechanism is to enable scalable and flexible task execution. Traditional multi-agent systems often struggle with static team compositions and limited scalability. By allowing agents to autonomously form nested teams, IoA can dynamically adjust to the complexity and scope of the task at hand, ensuring that agents with the appropriate skills and resources are engaged, leading to more efficient and effective problem-solving.

**Mechanism Overview.** Autonomous nested team formation allows clients to dynamically form and expand teams to tackle complex tasks efficiently. This mechanism leverages the server's capabilities to discover and connect clients based on their skills and characteristics.

Let $\mathcal{C}$ denote the set of all clients in the system. For each client $c_i \in \mathcal{C}$, a description $d_i$ of the integrated agent's skills and capabilities is required upon registration. When a task $t$ is assigned to a client $c_i$, it enters the Team Formation Block with access to two tools: `search_client` and `launch_group_chat`. The `search_client` tool allows the client to discover other clients by querying the server's Agent Registry with a generated list of desired characteristics $\mathcal{L}_d = [l_1, l_2, \ldots, l_k]$. The tool returns a subset of clients $\mathcal{C}_d \subseteq \mathcal{C}$ whose descriptions $d_j$ match the desired characteristics. The `launch_group_chat` tool enables the client to initiate a group chat with the selected clients. The LLM in the client decides which tool to call, considering the retrieved information from the server, local information from the Agent Contact Block, and the task requirements.

**Nested Team Formation.** In some cases, a single group may not be sufficient to complete a task. During discussions, agents may realize they need assistance from other agents with specific expertise. To address this, IoA supports nested team formation, allowing agents to initiate sub-group chats for sub-tasks. Fig. 2 presents a simple example.

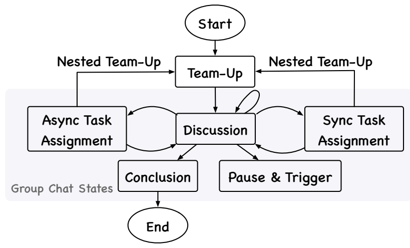



Figure 3: The state transition among different states.

Let $t_l$ be a sub-task assigned to client $c_i$ in group chat $g$. If $c_i$ determines it cannot complete the task alone, it can search for appropriate clients for $t_l$ and initiate a new sub-group chat $g_l$, inviting a subset of other agents to collaborate on the sub-task. This process can continue recursively, forming a tree-like structure with the root representing the initial group chat and branches representing sub-groups.

Furthermore, the nested team formation mechanism helps to manage the complexity of communication within large agent teams. Assuming that the communication graph within each group chat is fully connected, the number of edges in the graph represents the communication complexity. By decomposing a task into sub-tasks and allocating them to sub-group chats, the total number of edges can be reduced from $\frac{|g|(|g|-1)}{2}$ in the original group chat to $\sum_l \frac{|g_l|(|g_l|-1)}{2}$ in the sub-group chats, where $|g|$ and $|g_l|$ are the numbers of clients for task $t$ and $t_l$ respectively. This reduction in communication complexity leads to more efficient and focused collaboration among agents.

### 3.2 Autonomous Conversation Flow Control

Effective communication is crucial for successful collaboration among autonomous agents. Inspired by the Speech Act Theory (Austin, 1975; Searle, 1969) and its application in conventional multi-agent systems (Finin et al., 1994; Labrou et al., 1999), we introduce an autonomous conversation flow control mechanism in IoA that enables agents to effectively coordinate their communication.

**Sequential Speaking Mechanism.** Since the support for LLM interruption is still in its early stages, IoA adopts a sequential speaking mechanism, *i.e.,* only one agent speaks at a time, preventing confusion and maintaining a clear order of communication. The simple and naive mechanism, when combined with the following dynamic features, leads to effective collaboration.

**Finite State Machine for Group Chat States.** We formalize the conversation flow as a finite state machine $M = (S, \Sigma, \delta, s_0, F)$, where:

- $S = \{s_d, s_s, s_a, s_p, s_c\}$ is the set of states representing discussion, synchronous task assignment, asynchronous task assignment, pause & trigger, and conclusion, respectively.
- $\Sigma$ is the input alphabet, which corresponds to the possible actions or events in the conversation flow.
- $\delta : S \times \Sigma \to S$ is the transition function that maps a state and an input to the next state.
- $s_0 = s_d$ is the initial state, representing the start of the conversation in the discussion phase.
- $F = \{s_c\}$ is the set of final states, containing only the conclusion state.

As illustrated in Fig. 3, these states represent different phases of the collaboration process and help agents navigate the conversation more efficiently. The states are closely related to the speech acts defined in the Speech Act Theory, such as assertives (discussion), directives (task assignment), commissives (pause & trigger), and declarations (conclusion) (Searle, 1976).

The *discussion* allows general dialogue among agents, while the *synchronous* and *asynchronous task assignment* states enable task assignment with or without interrupting the ongoing discussion. The *pause & trigger* state introduces a mechanism for pausing the group chat and awaiting completion of specified asynchronous tasks, and the *conclusion* state marks the end of the collaboration.

**Autonomous State Transitions and Next Speaker Selection.** State transitions in the conversation flow are decided by the LLM in the client based on the current context and the progress. The LLM analyzes the messages exchanged and determines the most appropriate state to move the conversation forward, considering existing sub-tasks and their statuses, the need for further discussion, and the overall collaboration goal.

Let $\mathcal{M}_t$ be the set of messages exchanged up to time step $t$, and let $f_{\text{LLM}} : \mathcal{M}_t \times S \to S \times \mathcal{A}$ be the decision function of the LLM that maps the conversation history and current state to the next state and the next speaker. The next state $s_{t+1}$ and the next speaker $a_{t+1}$ are determined as follows:

$$(s_{t+1}, a_{t+1}) = f_{\text{LLM}}(\mathcal{M}_t, s_t).$$

The selection of the next speaker $a_{t+1}$ ensures that the most relevant agents are involved in the conversation at the appropriate times, promoting efficient information exchange and problem-solving.

### 3.3 Comprehensive Message Protocol Design

The effectiveness of the autonomous nested team formation and conversation flow control mechanisms in IoA relies on a robust message protocol. This protocol enables seamless communication and collaboration among agents by encapsulating all necessary information required for various mechanisms to function properly.

**Protocol Overview and Key Fields.** The agent message protocol in IoA is designed for extensibility and flexibility, facilitating effective multi-agent collaboration. The protocol consists of two main components: a header and a payload.

The header contains essential metadata about the message, ensuring correct addressing and processing. Key fields in the header include:

- `sender`: The unique identifier of the agent sending the message.
- `group_id`: The identifier of the group chat to which the message belongs.

The payload carries the main content of the message, varying by message type. It can include:

- `message_type`: Indicates the purpose of the message (e.g., discussion, task assignment, pause & trigger).
- `next_speaker`: The identifier(s) of the agent(s) expected to respond.

This structure contains other fields to support the diverse functionalities of IoA effectively. A detailed explanation and example of the message protocol can be found in Appendix C.1.

To ensure seamless communication and coordination, both the client and server components of IoA implement the message protocol. When a client sends a message, it encodes it according to the protocol and transmits it to the server. The server parses the message, extracts relevant information from the header, and routes it to the appropriate group chat based on the `group_id`. Upon receiving a message, the client decodes it and processes it accordingly. This consistent implementation ensures that all agents can understand and respond to messages correctly, regardless of their roles or tasks, maintaining a coherent and efficient collaboration process.

# 4 Experiments

To demonstrate IoA's effectiveness and versatility, we conducted extensive experiments on diverse tasks, from general AI assistance tasks to embodied agent and retrieval-augmented generation challenges. Our aim is to showcase its ability to facilitate collaboration among agents with different capabilities and highlight its adaptability across various problem domains. We compare IoA's performance against state-of-the-art approaches in each task category.[3] Due to page limits, we placed the RAG question-answering experiment in Appendix B and the analysis of team formation mechanisms and IoA's cost in Appendix A.

## 4.1 General AI Assistant Tasks

We present the results of IoA on two benchmarks that challenge the framework's ability to handle diverse, real-world tasks: GAIA (Mialon et al., 2023), which consists of a set of multi-step QA tasks that require the use of tools, and a manually crafted benchmark on non-QA tasks.

### 4.1.1 GAIA Benchmark

The GAIA benchmark (Mialon et al., 2023) is a collection of real-world questions that assess an AI system's ability to solve complex tasks by combining multiple skills, such as natural language understanding, reasoning, and external knowledge integration. The benchmark consists of three difficulty levels, each requiring a higher degree of capability and collaboration among agents.

**Setups.** We integrate four basic ReAct agents (Yao et al., 2023) into IoA, each has access to a tool, including a web browser, a code interpreter, a Wikidata searcher and a YouTube video transcript downloader. We compare IoA performance against several state-of-the-art agent systems. Each framework's performance is assessed across the three difficulty levels of the GAIA benchmark, as well as an overall performance metric. Refer to Appendix C.4.1 for more implementation details.

**Analysis.** The results of our experiments on the GAIA benchmark are presented in Table 1. IoA, with only basic ReAct agents integrated, achieves the highest overall performance, outperforming all other approaches. Notably, IoA excels in the higher difficulty levels (Level 2 and 3), where the tasks require more advanced reasoning and collaboration skills. This showcases the effectiveness of the framework's communication mechanisms and its ability to facilitate collaboration among agents.

Compared to AutoGen, which also employs a multi-agent approach, IoA demonstrates superior performance across two out of three difficulty levels without specific tuning on the framework's prompt on this task. This can be attributed to the framework's more advanced collaboration mechanisms, such as the autonomous team-up and conversation flow control. These features enable agents to dynamically form teams and carry out sub-tasks more effectively, leading to better overall performance on complex tasks.

The GAIA benchmark results highlight the potential of IoA as a powerful tool for orchestrating diverse agents for solving real-world, multi-step problems. By providing a flexible and efficient platform for agent collaboration, IoA enables even basic agents to achieve SOTA performance, surpassing more sophisticated standalone agents. This underscores the importance of effective communication and coordination in multi-agent systems and validates the design choices of IoA.

### 4.1.2 Open-Ended Instruction Benchmark

To further demonstrate the versatility and effectiveness of IoA in handling a wide range of real-world tasks, we curated a benchmark consisting of 153 open-ended instructions. These instructions are categorized into four main categories: search & report, coding, math, and life assistance. Unlike the GAIA benchmark, which primarily focuses on question-answering tasks with deterministic answers, our manually crafted benchmark includes a higher proportion of non-QA tasks that require generative responses. We believe this benchmark better reflects the diverse nature of real-world tasks that AI assistants are expected to handle.

**Setups.** For this experiment, we integrate two state-of-the-art third-party agents, AutoGPT (Significant Gravitas, 2023) and Open Interpreter (Open Interpreter, 2023), into IoA, see Appendix C.4.2 for the integration details. By integrating these capable agents into IoA, we aim to showcase the framework's ability to facilitate collaboration among diverse, independently developed agents. Given the high agreement between GPT and humans in judging response quality demonstrated in previous work (Chiang et al., 2023; Zheng et al., 2023a; Chan et al., 2023), we employ GPT-4-1106-preview as the judge. For each task

[3] Unless specified, we used the GPT-4-1106-preview with temperature 0.1 in our experiments.

| Models | Agent Type | Level 1 | Level 2 | Level 3 | Overall |
|---|---|---|---|---|---|
| GPT-4 | 👤 | 15.09 | 2.33 | 0.00 | 6.06 |
| GPT-4-Turbo | 👤 | 20.75 | 5.81 | 0.00 | 9.70 |
| AutoGPT-4 (Significant Gravitas, 2023) | 👤 | 13.21 | 0.00 | 3.85 | 4.85 |
| GPT-4 + Plugins (Mialon et al., 2023) | 👤 | 30.30 | 9.70 | 0.00 | 14.60 |
| FRIDAY (Wu et al., 2024) | 👤 | 45.28 | 34.88 | 11.54 | 34.55 |
| AutoGen (Wu et al., 2023) | 👥 | **54.72** | 38.37 | 11.54 | 39.39 |
| IoA | 👥 | 50.94 | **40.70** | **15.38** | **40.00** |

Table 1: The performance on the validation set of GAIA benchmark.

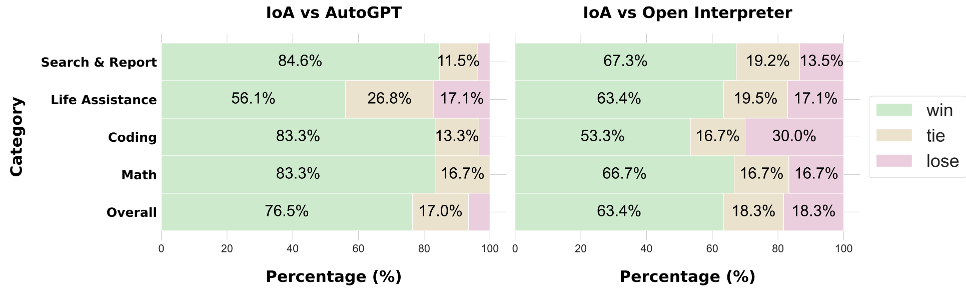



Figure 4: Win rates on the open-ended instruction benchmark between IoA, AutoGPT, and Open Interpreter.

in the benchmark, we compare the quality of the final answer generated by IoA with the answers provided by AutoGPT and Open Interpreter independently. Following Zheng et al. (2023a), we alter the order of responses in the prompt when using GPT-4 to determine the preference between two provided answers. Only when one competitor is consistently determined as better than the other across both orderings, the response is counted as a "win". This helps to mitigate potential biases introduced by the order of presentation.

**Analysis.** The results of this experiment, presented in Fig. 4, demonstrate that when orchestrating AutoGPT and Open Interpreter with IoA, it consistently outperforms both agents alone across all four categories, achieving win rates ranging from 56.1% to 84.6%. IoA's superior performance can be attributed to its ability to enable effective collaboration among the integrated agents, leveraging their complementary strengths to generate high-quality responses. Overall, IoA achieves a win rate of 76.5% against AutoGPT and 63.4% against Open Interpreter, highlighting its capability to efficiently gather and synthesize information and facilitate collaborative coding tasks.

The ability of IoA to seamlessly integrate diverse, independently developed agents holds great potential for creating more capable and versatile agent systems. By leveraging the strengths of existing agents and enabling them to collaborate effectively, IoA can tackle a wider range of tasks and generate higher-quality responses compared to individual agents. This highlights the importance of developing flexible and efficient platforms for agent collaboration, as they can significantly enhance the performance of AI assistants across various domains. As more advanced and specialized agents emerge, the potential of IoA to integrate them and facilitate their collaboration grows, paving the way for the development of increasingly sophisticated and user-centric AI solutions.

| Model | Metric | Cabinet | Sweep | Sandwich | Sort | Rope |
|---|---|---|---|---|---|---|
| Central Plan (oracle) | Success | 0.90 | **1.00** | 0.96 | 0.70 | 0.50 |
| | #Step | <u>4.0</u> | 8.4 | <u>8.8</u> | 8.6 | <u>2.3</u> |
| Roco Dialog | Success | 0.75 | 0.70 | 0.70 | 0.70 | **0.70** |
| | #Step | 4.7 | <u>7.9</u> | 9.1 | <u>5.4</u> | 2.4 |
| IoA | Success | **1.00** | 0.80 | **1.00** | **1.00** | **0.70** |
| | #Step | 4.6 | 8.5 | 8.9 | 5.8 | 2.6 |

Table 2: Average success rate and the number of steps on different tasks from RoCoBench.

## 4.2 Embodied Agent Tasks

Embodied AI aims to develop agents that can perceive, understand, and interact with their physical environment. To evaluate the performance of IoA in embodied agent tasks, we conduct experiments on RoCoBench (Mandi et al., 2023), a recently proposed benchmark for assessing the collaboration and communication capabilities of embodied agents. RoCoBench consists of six collaborative tasks, each requiring two agents with partial observation of the environment to work together to achieve a common goal.

**Setups.** We compare IoA against two baselines from Mandi et al. (2023): Central Plan and Roco Dialog. Central Plan assumes a central agent has access to complete information of the environment and can control the two embodied agents. Roco Dialog is a multi-agent framework designed specifically for this task, where two agents communicate, and make decisions independently.

As RoCoBench does not require agents to interact with tools but instead expects them to output action plans in a specific format, we do not integrate external agents into IoA. Instead, we provide the environment observations to the two clients and extract their action plans from their discussion. The implementation details can be found in Appendix C.4.3. To ensure a fair comparison, we run both IoA and Roco Dialog with the same GPT-4-1106-preview model for 10 runs on each task and report the average success rate and the number of steps taken. The results for Central Plan are directly taken from Mandi et al. (2023). Note that the Pack Grocery task in RoCoBench is discarded due to the errors in the released benchmark.

**Analysis.** Table 2 presents the average success rate and the number of steps required to complete each task. Despite not being specifically designed for embodied tasks, IoA outperforms Roco Dialog, a multi-agent framework tailored for this benchmark, on four out of five tasks in terms of success rate. IoA achieves perfect scores on the Cabinet, Sandwich, and Sort tasks, demonstrating the effectiveness of its communication and collaboration mechanisms in enabling embodied agents to work together towards a common goal. Remarkably, IoA's performance is superior or comparable to the Central Plan baseline, which assumes access to oracle information and full observability of the environment. However, IoA generally consumes slightly more decision steps to complete the tasks. Still, the number of steps is fairly close to Roco Dialog and Central Plan on all the tasks. Considering that IoA is a general multi-agent framework and not specifically designed for this task, we think the increase in the number of steps is acceptable.

## 5 Related Work

**LLM-based Agents** Recent advancements in LLMs, such as GPT (OpenAI, 2023), Claude (Anthropic, 2024), and Gemini (Reid et al., 2024), have led to the development of highly capable AI agents. These agents engage in natural language interactions and perform diverse tasks. Researchers have enhanced LLM-based agents by integrating external tools and knowledge sources (Nakano et al., 2021; Yao et al., 2023; Schick et al., 2023; Shen et al., 2023). Advances agents like OS-Copilot facilitate generalist interactions across web browsers and code terminals (Wu et al., 2024), while OpenDevin focuses on autonomous software development tasks (OpenDevin Team, 2024). Other notable developments include XAgent, AutoGPT and Open Interpreter for complex task solving (Team, 2023; Significant Gravitas, 2023; Open Interpreter, 2023), and Voyager for open-ended embodied tasks in Minecraft (Wang et al., 2023a). These advancements have laid the foundation for more sophisticated and versatile LLM-based agents.

**LLM-based Multi-Agent Systems** Building on the success of individual LLM-based agents, researchers have explored multi-agent systems composed of these agents. Early works demonstrated using LLMs to simulate multi-agent interactions and emergent behaviors (Park et al., 2023). Frameworks like AgentVerse (Chen et al., 2023) and AutoGen (Wu et al., 2023) provide infrastructure for agent collaboration. In software development, systems like ChatDev (Qian et al., 2023a) and MetaGPT (Hong et al., 2023) automate coding, testing, and debugging processes. Despite these advancements, significant limitations remain, such as the lack of support for integrating diverse third-party agents, the inability to support distributed multi-agent systems, and the reliance on hard-coded communication protocols and state transitions. IoA aims to address these limitations, offering a flexible and scalable platform for LLM-based multi-agent collaboration to tackle complex real-world problems effectively.

## 6 Conclusion

In this paper, we introduce IoA, a novel framework for LLM-based multi-agent collaboration inspired by the Internet. IoA addresses limitations of existing frameworks by providing a scalable platform for integrating diverse agents, enabling distributed collaboration, and introducing dynamic mechanisms for teaming and conversation control. Our experiments show IoA consistently outperforms state-of-the-art baselines. We believe IoA will serve as a foundation for future research, enabling integration of diverse, specialized agents and opening new possibilities for multi-agent systems.

## Limitations

While IoA demonstrates significant potential for enabling effective collaboration among heterogeneous agents, there are several limitations to be addressed in future work. Firstly, since there is no existing benchmark requiring a large-scale agent team, the effectiveness of the nested team formation mechanism has not yet been comprehensively evaluated.

Secondly, further research is needed to investigate the performance and adaptability of IoA in more diverse and realistic settings, such as multimodal communication, adversarial environments, and more realistic partially observable scenarios (RocoBench is partially observable, but it operates within a simulated environment). Additionally, the long-term stability and robustness of the framework in extended collaboration sessions remain to be evaluated.

Despite these limitations, IoA represents a significant step towards realizing the vision of an Internet of Agents, and we believe that addressing these challenges will pave the way for more advanced and practical multi-agent systems.

## References


Marah I Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat S. Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Dan Iter, Amit Garg, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Chen Liang, Weishung Liu, Eric Lin, Zeqi Lin, Piyush Madan, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Xia Song, Masahiro Tanaka, Xin Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Michael Wyatt, Can Xu, Jiahang Xu, Sonali Yadav, Fan Yang, Ziyi Yang, Donghan Yu, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *CoRR*, abs/2404.14219.

Anthropic. 2024. Introducing the next generation of claude. Accessed: 2024-06-14.

John Langshaw Austin. 1975. *How to do things with words*, volume 88. Oxford university press.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. Chateval: Towards better llm-based evaluators through multi-agent debate. *CoRR*, abs/2308.07201.

Harrison Chase. 2022. LangChain.

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

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Timothy W. Finin, Richard Fritzson, Donald P. McKay, and Robin McEntire. 1994. KQML as an agent communication language. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94), Gaithersburg, Maryland, USA, November 29 - December 2, 1994*, pages 456–463. ACM.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *Preprint*, arXiv:2011.01060.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, and Chenglin Wu. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *CoRR*, abs/2308.00352.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zhen Leng Thai, Kai Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *CoRR*, abs/2404.06395.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *Preprint*, arXiv:1705.03551.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew

Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc V. Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Yannis Labrou, Tim Finin, and Yun Peng. 1999. Agent communication languages: the current landscape. *IEEE Intell. Syst.*, 14(2):45–52.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Zhao Mandi, Shreeya Jain, and Shuran Song. 2023. Roco: Dialectic multi-robot collaboration with large language models. *CoRR*, abs/2307.04738.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Cristian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, and et al. 2024. Gemma: Open models based on gemini research and technology. *CoRR*, abs/2403.08295.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. GAIA: a benchmark for general AI assistants. *CoRR*, abs/2311.12983.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. Webgpt: Browser-assisted question-answering with human feedback. *CoRR*, abs/2112.09332.

Open Interpreter. 2023. Open Interpreter.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

OpenDevin Team. 2024. OpenDevin: An Open Platform for AI Software Developers as Generalist Agents. `https://github.com/OpenDevin/OpenDevin`. Accessed: ENTER THE DATE YOU ACCESSED THE PROJECT.

Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA, 29 October 2023- 1 November 2023*, pages 2:1–2:22. ACM.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023a. Communicative agents for software development. *CoRR*, abs/2307.07924.

Chen Qian, Yufan Dang, Jiahao Li, Wei Liu, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2023b. Experiential co-learning of software-developing agents. *CoRR*, abs/2312.17025.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *CoRR*, abs/2307.16789.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy P. Lillicrap, Jean-Baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, Ioannis Antonoglou, Rohan Anil, Sebastian Borgeaud, Andrew M. Dai, Katie Millican, Ethan Dyer, Mia Glaese, Thibault Sottiaux, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, James Molloy, Jilin Chen, Michael Isard, Paul Barham, Tom Hennigan, Ross McIlroy, Melvin Johnson, Johan Schalkwyk, Eli Collins, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Clemens Meyer, Gregory Thornton, Zhen Yang, Henryk Michalewski, Zaheer Abbas, Nathan Schucher, Ankesh Anand, Richard Ives, James Keeling, Karel Lenc, Salem Haykal, Siamak Shakeri, Pranav Shyam, Aakanksha Chowdhery, Roman Ring, Stephen Spencer, Eren Sezener, and et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *CoRR*, abs/2403.05530.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

John R. Searle. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press.

John R Searle. 1976. A classification of illocutionary acts1. *Language in society*, 5(1):1–23.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving AI tasks with chatgpt and its friends in hugging face. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Significant Gravitas. 2023. AutoGPT.

XAgent Team. 2023. Xagent: An autonomous agent for complex task solving.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *CoRR*, abs/2305.16291.

Haotian Wang, Xiyuan Du, Weijiang Yu, Qianglong Chen, Kun Zhu, Zheng Chu, Lian Yan, and Yi Guan. 2023b. Apollo's oracle: Retrieval-augmented reasoning in multi-agent debates. *ArXiv*, abs/2312.04854.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2023c. MINT: evaluating llms in multi-turn interaction with tools and language feedback. *CoRR*, abs/2309.10691.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023d. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen LLM applications via multi-agent conversation framework. *CoRR*, abs/2308.08155.

Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. 2024. Os-copilot: Towards generalist computer agents with self-improvement. *CoRR*, abs/2402.07456.

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. *Preprint*, arXiv:1809.09600.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023a. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

# A Analysis

## A.1 Team Formation Precision

Evaluating the precision of the autonomous team formation mechanism is crucial for IoA, which is designed to tackle complex tasks by forming effective teams of agents. However, existing benchmarks do not provide a suitable environment for large-scale agent experiments. To address this limitation, we develop a team formation benchmark using GPT-4, consisting of 625 diverse tasks, each paired with at least two different agents, resulting in a total of 1500 agents. These dummy agents (with only a description, and no actual implementation) are registered to IoA's server to assess the accuracy of the autonomous team formation mechanism. Detailed data construction processes can be found in Appendix E.

**Setups.** We report the recall of team members, measuring the proportion of labeled agents included in the group chat. Given the large agent pool, recruited agents may not always be in the labeled set but can still be highly relevant to the task. To account for this, we calculate the cosine similarity between the description embeddings of non-retrieved labeled agents and recruited agents. Specifically, we examine whether any recruited agent is within the top 10 most similar agents to a

Figure 5: Recall of the agents on our team formation benchmark.

| | Recall |
|---|---|
| w/o similarity | 0.414 |
| w/ similarity | 0.751 |

Figure 6: Cost analysis of standalone agents and IoA-integrated agents on the open-ended instruction benchmark.

| Setting | Cost per Task |
|---|---|
| AutoGPT (Standalone) | \$0.39 |
| Open Interpreter (Standalone) | \$0.16 |
| AutoGPT (in IoA) | \$0.33 |
| Open Interpreter (in IoA) | \$0.13 |
| IoA Communication | \$0.53 |
| IoA Communication (Dedup.) | \$0.28 |
| IoA Overall | \$0.99 |
| IoA Overall (Dedup.) | \$0.74 |

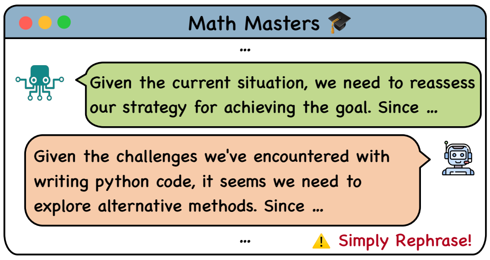



Figure 7: An example of the repeated communication.

non-retrieved labeled agent. If true, it is counted as successfully recalled.

**Analysis.** The results shown in Fig. 5 indicate that the team formation mechanism achieves a recall rate of 41.4% without similarity matching. This recall rate serves as a lower bound on performance, as many recruited agents, while not identical to the labeled agents, are still highly relevant to the tasks. Incorporating similarity matching enhances the recall rate to 75.1%, indicating that the mechanism can effectively identify and recruit agents with semantically similar capabilities. It is important to recognize that top 10 matching still does not perfectly reflect the accuracy of team formation, as agents not in the top 10 most similar can still be highly relevant to the task.

The high recall rate with similarity matching underscores the capability of IoA to form precise and effective teams. This precision is critical for addressing complex tasks requiring a combination of skills and knowledge from various agents. By leveraging semantic similarity, IoA ensures that the formed teams closely align with task requirements, thereby maintaining the integrity and effectiveness of the team formation process.

### A.2 Cost and Sub-Optimal Communication Pattern Analysis

To evaluate the economic feasibility and potential for optimization of the IoA, we conduct a cost analysis on the open-ended instruction benchmark (Section 4.1.2), where AutoGPT and Open Interpreter are integrated. We compare the average cost per task for these agents when operating individually and when integrated into the IoA.

As shown in Fig. 6, when integrated into IoA, the costs of both agents are decreased due to the task decomposition for each task. However, the IoA introduces an additional communication cost of \$0.53 per task, resulting in an overall cost of \$0.99.

During our analysis, we observed unexpected and suboptimal communication patterns that contributed to the high communication cost. One notable pattern was the repetition of information, where the LLMs in the clients would repeat or rephrase previous chats from themselves or others, leading to a stagnation in progress. This phenomenon was particularly prevalent after several asynchronous task assignments. Although each task assignment did not require immediate waiting, as the conversation progressed, new decisions had to be made based on the conclusions from previously assigned and not yet completed asynchronous tasks. Despite providing the client LLMs with the option to switch the group chat state to pause & trigger, they sometimes fail to switch, as illustrated in Fig. 7. This drawback in LLM is also observed in other multi-agent work (Li et al., 2023; Mandi et al., 2023).

To quantify the impact of this suboptimal communication pattern, we manually removed the repetitions and recalculated the token numbers and corresponding costs. Surprisingly, this resulted in a nearly 50% reduction in communication costs, as shown in the "Dedup." rows of Fig. 6. This finding aligns with observations from other multi-agent communication frameworks, suggesting that while modern LLMs are well-aligned to be effective chatbot assistants, they may not be optimally aligned to be efficient communicating agents. Agents should not only complete the given tasks accurately but also communicate effectively with others, understanding conversation states and making proper decisions. This insight raises new research questions regarding the agent alignment of LLMs and highlights the need for further investigation in this area.

Despite the current cost overhead and suboptimal communication patterns, the IoA demonstrates significant potential for enabling effective collaboration among heterogeneous agents. By addressing these challenges through prompt optimization, protocol refinement, and the development of more sophisticated frameworks under the concept of IoA, we believe that the cost of communication can be significantly reduced. As research progresses, IoA and similar approaches will become increasingly attractive and economically viable solutions for complex multi-agent systems.

# B Retrieval-Augmented Generation Experiment

We further evaluate the communication effectiveness of IoA on retrieval-augmented generation (RAG) tasks (Lewis et al., 2021). In RAG tasks, agents need to retrieve relevant information and communicate with each other to arrive at the correct answer, making it another ideal testbed for assessing communication effectiveness.

**Setups.** Following the setup in Apollo's Oracle (Wang et al., 2023b), we use GPT-3.5-turbo-0125 as the LLM in the clients and provide clients with two evidence pools: Wikipedia and Google. We design three scenarios: one with two clients having access to different evidence pools (marked as *Partial*), requiring them to communicate and exchange information gathered from different sources; and two scenarios where all agents have access to both evidence pools, with one scenario involving two clients and the other involving three clients. We evaluate IoA on four datasets: TriviaQA (Joshi et al., 2017), Natural Questions (NQ) (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), and 2WikiMultiHopQA (Ho et al., 2020). From each dataset, we randomly sample 250 question-answer pairs. Implementation details can be found in Appendix C.4.4.

**Analysis.** As shown in Table 3, IoA significantly improves upon various baselines and achieves performance surpassing or comparable to Apollo's Oracle, the previous multi-agent framework specific to this task. IoA with 3 clents consistently outperforms GPT-3.5 with different prompting strategies and surpasses Apollo's Oracle on three out of four datasets. This highlights the effectiveness of the multi-agent collaboration facilitated by IoA, enabling agents to leverage their collective knowledge and reasoning abilities to generate accurate responses.

Notably, IoA with 2 agents and complete access to evidence pools achieves the best performance on the HotpotQA and 2WikiMultiHopQA datasets, surpassing both Apollo's Oracle and IoA with 3 agents. This suggests that the optimal number of agents may vary depending on the complexity and nature of the task, and that more agents do not always guarantee better performance. The comparison between the partial and complete scenarios in IoA with 2 agents demonstrates the importance of effective communication. Even when agents have access to different knowledge sources, simulating an information-asymmetric scenario, IoA achieves remarkable performance, outperforming Apollo's Oracle on two out of four datasets. This suggests that the framework's communication mechanisms enable agents to effectively exchange and synthesize information from diverse sources, compensating for the lack of complete information in individual agents.

These results underscore the generalizability and adaptability of IoA, as it can facilitate effective collaboration among client agents in various settings, even without the integration of specialized third-party agents. The framework's ability to enable agents to leverage their collective knowledge and communicate effectively, even in information-asymmetric scenarios, positions it as a powerful tool for enhancing the performance of multi-agent systems in retrieval-augmented generation tasks and beyond.

# C Implementation Details of IoA

In this appendix, we provide a comprehensive overview of the implementation details for each module in the client and server layers of IoA.

## C.1 Message Protocol

To support the functionalities of IoA introduced in Section 3, we have designed a comprehensive agent message protocol that facilitates efficient communi-

| Model | TriviaQA | NQ | HotpotQA | 2WikiMultiHopQA |
|---|---|---|---|---|
| GPT 4 | 0.902 | 0.692 | 0.566 | 0.284 |
| GPT 3.5 Turbo | 0.778 | 0.532 | 0.384 | 0.210 |
| + Zero-Shot CoT | 0.772 | 0.588 | 0.410 | 0.190 |
| + Self Consistency | 0.818 | 0.622 | 0.408 | 0.206 |
| + Reflxion | 0.762 | 0.586 | 0.378 | 0.254 |
| + Multi-Agent Debate1 | 0.798 | 0.648 | 0.394 | 0.186 |
| + Multi-Agent Debate2 | 0.756 | 0.576 | 0.450 | 0.334 |
| Apollo's Oracle | 0.834 | 0.662 | 0.542 | 0.350 |
| IoA + 2 Agents (Partial) | 0.803 | **0.708** | 0.478 | 0.449 |
| IoA + 2 Agents (Complete) | 0.820 | 0.671 | **0.586** | **0.530** |
| IoA + 3 Agents (Complete) | **0.908** | <u>0.682</u> | <u>0.575</u> | <u>0.519</u> |

Table 3: All the comparative test results. IoA based on GPT-3.5 exceeds GPT4 on some datasets. Excluding the GPT4 results, we highlight the best results in bold, and the second best results in underlined. Partial indicates that different agents have access to different evidence pool, while Complete means all evidence pool are accessible to all agents.

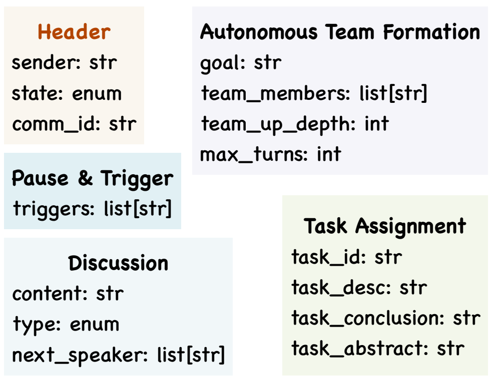



Figure 8: Fields in the IoA message protocol.

cation and coordination among agents. The protocol, as illustrated in Fig. 8, consists of several fields that cater to the specific requirements of various mechanisms within the framework.

Firstly, the protocol includes the following header for all message types:

- `sender` (str): The name or unique identifier of the agent sending the message.
- `state` (enum): The current state of the group chat associated with the message, which can be either team formation or communication.
- `comm_id` (str): The unique identifier of the group chat to which the message belongs.

To support the autonomous team formation mechanism, the protocol incorporates the following fields:

- `goal` (str): The objective or task that the current group chat aims to accomplish.
- `team_members` (list[str]): The names or unique identifiers of the agents required for the current group chat.
- `team_up_depth` (int): The depth of the current nested team formation, used to determine if the maximum allowed depth has been reached.
- `max_turns` (int): The maximum number of discussion turns allowed for the current group chat. If exceeded, the group chat will be forced into the conclusion phase.

For facilitating the discussion phase, the protocol includes the following fields:

- `content` (str): The actual content of the current message.
- `type` (enum): Specifies the next dialogue state, which can be discussion, task assignment, or conclusion.
- `next_speaker` (list[str]): The name(s) or unique identifier(s) of the agent(s) expected to speak next. In the discussion state, `next_speaker` is limited to a single agent, while in the task assignment state, it can include multiple agents, indicating that the current message contains multiple task assignments.

To support the task assignment mechanism, the protocol incorporates the following fields:

- `task_id` (str): The automatically generated unique identifier for the current task.
- `task_desc` (str): The description of the task assigned to the client, extracted from the chat.
- `task_conclusion` (str): The conclusion or result provided by the client after completing the assigned task.
- `task_abstract` (str): A concise summary of

the completed task.

Lastly, to support the pause & trigger mechanism, the protocol includes the following field:

- `triggers` (list[str]): A list of task IDs that require a trigger to be set.

By adhering to this comprehensive agent message protocol for sending and receiving messages, clients within IoA can effectively achieve autonomous team formation and conversation flow control. The protocol ensures that all necessary information is communicated among agents, enabling seamless collaboration and coordination in various task scenarios.

## C.2 Client

The client component of IoA plays a crucial role in enabling the integration and collaboration of heterogeneous agents. It consists of three layers: the Foundation Layer, the Data Layer, and the Interaction Layer. Each layer comprises several modules that work together to facilitate efficient communication, data management, and agent coordination. In this subsection, we provide a detailed overview of the implementation of each module within the client's layers.

### C.2.1 Foundation Layer

**Network Infrastructure Module** In IoA, all clients maintain a persistent connection to the server using the WebSocket protocol, similar to an instant messaging application. When a client sends a message, it is transmitted to the server, which parses the `comm_id` field in the message and forwards it to the other clients in the corresponding group chat via their respective WebSocket connections. The real-time nature of WebSocket ensures that messages are delivered promptly, enabling clients to receive and respond to messages without delay.

**Data Infrastructure Module** To support the data storage and retrieval requirements of the upper-level Data Layer modules, we employ SQLite as the primary database solution. SQLite provides a lightweight and efficient means of persisting and accessing data related to agent contacts, group information, and task management. By leveraging SQLite, the client can store and retrieve information about encountered agents, group chat details, and task assignments, ensuring data consistency and availability throughout the collaboration process.

**Agent Integration Module** The Agent Integration Module defines the protocol that third-party agents must adhere to in order to seamlessly integrate with IoA. Currently, the agent integration protocol in IoA requires agents to implement a function `def run(task_desc: str) -> str`, which accepts a task description as input and returns a summary of the task completion. This simple yet effective protocol allows diverse agents to be incorporated into the framework, enabling them to contribute their unique capabilities to the collaboration process. As IoA evolves, the integration protocol can be extended to support more advanced functionalities and interaction patterns.

### C.2.2 Data Layer

**Agent Contact Module** The Agent Contact Module is responsible for maintaining a record of the clients that the current client has previously collaborated with. It stores information such as the names and descriptions of these clients, providing a valuable reference for future collaborations. The module aims to support the client in evaluating and storing collaboration outcomes after each task, allowing it to make informed decisions when forming teams for subsequent tasks. During the team formation process, the information stored in this module is included in the prompt to assist the client in selecting the most suitable partners based on prior experiences.

**Group Info Module** The Group Info Module manages all group chat-related information, including the following fields:

- `comm_id` (str): The unique identifier of the group chat.
- `goal` (str): The objective or task that the group chat aims to accomplish.
- `team_members` (str): The list of agents participating in the group chat.
- `state` (str): The current state of the group chat (e.g., team formation, discussion, task assignment, conclusion).
- `conclusion` (str | None): The final outcome or conclusion reached by the group chat.
- `team_up_depth` (int): The depth of the nested team formation within the group chat.

- max_turns (int): The maximum number of communication turns allowed in the group chat.

By organizing and persisting this information, the Group Info Module enables clients to maintain a coherent view of the ongoing collaborations and their progress.

**Task Management Module** The Task Management Module is responsible for storing and tracking the tasks assigned within each group chat. It maintains the following fields for each task:

- task_id (str): The unique identifier of the task.
- task_desc (str): The detailed description of the task.
- task_abstract (str): A concise summary of the task.
- assignee (str): The agent assigned to complete the task.
- status (enum): The current status of the task (e.g., pending, in progress, completed).
- conclusion (str | None): The final result or outcome of the task.

By keeping track of task-related information, the Task Management Module enables clients to monitor the progress of assigned tasks and ensures that all task-related data is readily available for reference and decision-making purposes.

#### C.2.3 Interaction Layer

**Team Formation Module** As briefly introduced in Section 3.1, when a client receives a task, it is equipped with two essential tools: `search_agent(desc: list[str]) -> list[agent]` and `team_up(team_members: list[str] | None) -> comm_id`. The client must decide whether to utilize the search_agent tool to find agents on the server that match the specified description, or to directly call the team_up tool based on the discovered agents and historical collaboration information. If the client invokes team_up without specifying any agents, it implies that the task will be completed by a single agent. To prevent infinite loops, IoA imposes a limit on the maximum number of tool calls, set to 10 by default. If the client reaches this limit without successfully launching a group chat, it is forced to invoke the team_up tool to initiate the collaboration process.

**Communication Module** The Communication Module handles the core functionalities of message generation and message reception. When a client generates a message, IoA processes it according to the agent message protocol. If the message type is conclusion, the client enters the conclusion phase, where it provides a final answer to the group chat goal based on the accumulated chat records and task completion information. In the case of a pause & trigger message, the framework prompts the client to generate the task IDs that require triggers and broadcasts them to all group members. For discussion or task assignment messages, they are directly broadcast to all participants in the group chat.

Upon receiving a message, the client parses it according to the agent message protocol. If the next_speaker field does not include the current client, the message is simply added to the group chat history. However, if the client is designated as the next speaker, it must take appropriate actions based on the message type. For discussion messages, the client generates a response to continue the conversation. In the case of sync or async task assignment messages, the client extracts its assigned task from the chat record, summarizes it, and specifies the relevant information to be passed to the integrated agent. The agent then executes the task based on the summarized description and relevant chat messages, returning the result upon completion. If the message type is pause & trigger, the client updates the corresponding task triggers in the Task Management Module.

The Communication Module, in conjunction with the other modules in the Interaction Layer and Data Layer, enables seamless and structured collaboration among agents. By adhering to the well-defined agent message protocol and leveraging the functionalities provided by the various modules, clients can effectively participate in discussions, assign tasks, and coordinate their actions to achieve the desired goals.

### C.3 Server

The server component of IoA serves as the central hub for agent coordination, communication, and management. It comprises three layers: the Foundation Layer, the Data Layer, and the Interaction Layer. Each layer contains modules that work together to facilitate agent registration, discovery, and message routing. In this subsection, we provide a detailed description of the implementation of each module within the server's layers.

### C.3.1 Foundation Layer

**Network Infrastructure Module and Data Infrastructure Module** The Network Infrastructure Module and Data Infrastructure Module in the server are largely similar to their counterparts in the client. However, the server's Data Infrastructure Module incorporates the use of the Milvus vector database to support the construction and maintenance of the Agent Registry. Milvus enables efficient similarity search and retrieval of agent information based on their characteristics, allowing the server to provide clients with the functionality to discover and match agents effectively.

**Security Module** While the Security Module is not extensively utilized in the current implementation of IoA, we acknowledge its crucial role in ensuring the integrity and reliability of the framework in real-world deployments. This module is responsible for verifying and controlling the integration of third-party agents into the clients, preventing malicious agents from compromising the entire framework. As IoA evolves, the Security Module will be enhanced to provide robust authentication, authorization, and monitoring mechanisms, safeguarding the collaborative environment from potential security threats.

### C.3.2 Data Layer

**Agent Registry Module** The Agent Registry Module maintains a comprehensive record of all clients integrated into the server. When a client connects to the server, it is required to provide a detailed description of the integrated agent, including its name and capability description. This information is stored in the Agent Registry, enabling similarity matching based on agent characteristics. The Agent Registry serves as a central repository for agent information, facilitating agent discovery and team formation processes.

**Session Management Module** The Session Management Module is responsible for managing the WebSocket connections of all online agents and keeping track of the group chats they participate in. It maintains a mapping between agents and their respective WebSocket connections, as well as the associations between agents and group chats. When a client sends a message, the Session Management Module ensures that the message is properly routed to all clients involved in the corresponding group chat, guaranteeing reliable and efficient communication within the collaborative environment.

### C.3.3 Interaction Layer

**Agent Query Module** The Agent Query Module handles incoming requests from clients seeking to discover and match agents based on specific characteristics. Upon receiving a query request, the module converts the provided characteristics into vector representations and performs similarity matching against the agents stored in the Agent Registry. The implementation of this module can vary depending on the specific requirements and scalability needs of the framework. For instance, techniques such as BM25 or other information retrieval methods can be employed to enhance the matching process and improve the relevance of the returned agent results.

**Group Setup Module** The Group Setup Module is responsible for handling client requests to create new group chats. When a client submits a request to set up a group chat, specifying the desired team members, the Group Setup Module processes the request and initializes a new group chat instance. It assigns a unique `comm_id` to the newly created group chat and notifies all participating clients about their inclusion in the chat. The Group Setup Module works in conjunction with the Session Management Module to ensure that the necessary WebSocket connections and mappings are established for efficient communication within the group chat.

**Message Routing Module** The Message Routing Module plays a critical role in facilitating communication between clients within group chats. When a client sends a message, the Message Routing Module receives the message and parses it according to the agent message protocol. Based on the `comm_id` specified in the message, the module identifies the corresponding group chat and forwards the message to all clients associated with that chat. The Message Routing Module leverages the information maintained by the Session Management Module to ensure accurate and timely delivery of messages to the intended recipients.

The server component of IoA, with its carefully designed modules and interactions, provides a robust and efficient infrastructure for agent coordination, communication, and management. By leveraging the capabilities of the Foundation Layer, Data Layer, and Interaction Layer, the server enables seamless agent discovery, team formation, and message exchange, fostering a collaborative environment where diverse agents can work together to achieve common goals.

As IoA continues to evolve, the server component will be further enhanced to incorporate advanced features such as load balancing, fault tolerance, and scalability, ensuring that the framework can handle the growing demands of real-world multi-agent systems. Additionally, the Security Module will be strengthened to provide comprehensive security measures, safeguarding the integrity and confidentiality of agent interactions within the framework.

### C.4 Implementation Details of Different Experiments

In this section, we provide an overview of the implementation details for each experiment conducted to evaluate the performance of IoA.

#### C.4.1 GAIA

For the GAIA benchmark, IoA integrated four ReAct agents: Web Browser, Code Executor, YouTube Transcript Downloader, and Wikidata Searcher. The tools provided to Web Browser and Code Executor agents are adapted from the AutoGen framework with minor modifications to ensure compatibility with IoA. To address the YouTube-related tasks in GAIA, we develop a YouTube video transcript downloader based on PyTube[4]. For videos without readily available transcripts, the tool employs the Whisper model to transcribe spoken language into text. Similarly, we adapt the Wikidata tool from Langchain[5] to fit the IoA ecosystem. These adaptations showcases a key feature of IoA: when a task requires a specific tool, it can be easily integrated into the system through its implementation and agent adaptation, enabling it to participate in task completion.

Due to budget constraints, we conduct performance testing on the GAIA validation set. Despite this limitation, the results provide valuable insights into the effectiveness of IoA in handling complex, multi-step tasks.

#### C.4.2 Open-Ended Instruction Benchmark

To create a diverse and challenging benchmark for evaluating the performance of IoA on open-ended tasks, we construct a set of 153 instructions spanning four categories: search & report, coding, math, and life assistance. The benchmark construction process involved three main steps:

First, we select the instructions based on the real-world complex tasks used by XAgent (Team, 2023). These instructions were categorized into the four aforementioned groups. Second, to increase the diversity of the benchmark, we manually create an additional 10 complex tasks. Finally, we use the Self-Instruct method (Wang et al., 2023d) to generate approximately 200 instructions, using the previously selected instructions as seeds. After manual screening and modification, we obtained the additional 94 instructions, resulting in a total of 153 tasks. The benchmark eventually consists of 52 search & report tasks, 30 coding tasks, 30 math tasks, and 41 life assistance tasks. By incorporating a diverse set of open-ended instructions, this benchmark allows for a comprehensive evaluation of the performance and versatility of IoA in handling a wide range of real-world scenarios. We show one example instruction for each category in Fig. 10.

**Evaluation Methodology.** For IoA, we consider the final conclusion generated by the agents as the final answer. However, since AutoGPT (Significant Gravitas, 2023) and Open Interpreter (Open Interpreter, 2023) complete tasks in multiple steps and do not inherently generate a conclusion, we prompted them to provide a detailed conclusion as the final answer after task completion.

Inspired by the pairwise comparison evaluation method used in MT-Bench (Zheng et al., 2023b), we employ GPT-4 to evaluate the responses of IoA against AutoGPT and Open Interpreter. To mitigate potential biases introduced by the order of the responses, we alternate the order of the two responses when presenting them to GPT-4 for evaluation. A result is counted as a *win* for a system only when it is consistently determined to be superior to its competitor in both orderings. In cases where the performance is inconsistent across the two orderings, the result is considered a *draw*.

#### C.4.3 Embodied Agent Tasks

For the RocoBench experiments, we adhere to the original paper's methodology, which relies on discussions and parsing specific formatted strings from the discussion results to determine the embodied agent's actions, rather than using agents to call tools directly. We implement two clients that communicate without integrated agents, requiring

[4] https://github.com/pytube/pytube

[5] https://python.langchain.com/v0.1/docs/integrations/tools/wikidata/

them to output strings in the RocoBench format at the conclusion stage. These strings are then parsed and used to interact with the environment using RocoBench's predefined parsing functions. This approach serves as a validation of IoA's client implementation and communication mechanism design.

To accommodate the varying requirements of different tasks in RocoBench, we adopt task-specific settings. For the Sort, Sandwich, and Sweep tasks, which exhibit strong interdependencies between steps, we retained the chat history and continued each new action discussion based on the previous group chat. In contrast, for the Cabinet and Rope tasks, where the steps were less interdependent, we initiated a new group chat for each action to optimize costs. Other settings remained consistent with the Roco Dialog baseline.

#### C.4.4 Retrieval-Augmented Generation

For the retrieval-augmented generation (RAG) question-answering task, we follow the settings outlined in Apollo's Oracle. We provide agents with two evidence pools: one derived from Wikipedia and the other from Google. For Wikipedia, we utilize Pyserini's pre-built index of Wikipedia content up to January 20, 2021, retrieving the top 10 most relevant results for each query. For Google, we directly access the Google Search API, returning the top 5 most relevant results for each query. These tools were made available to the client-side LLMs, enabling them to query relevant information during discussions and ultimately provide well-informed answers.

To evaluate the performance of IoA on the RAG task, we randomly sample 500 entries from the validation or test sets of the four datasets. After the model generates answers, we employ GPT-4 for answer evaluation. Specifically, we provide GPT-4 with the dataset answers and the model's answers, requiring it to output its reasoning in a Chain of Thought (CoT) manner before providing a final correctness judgment.

## D Visualization of RocoBench

We provide the visualization of RocoBench at Fig. 9. The **cabinet task** requires three agents to collaborate: two agents open and hold the cabinet door while the third agent retrieves two cups from inside the cabinet and places them onto coasters that match the color of the cups. The **sweep task** involves two agents coordinating their actions: one agent controls a broom to sweep cubes, while the other agent holds a bucket to collect the cubes, and finally, they dump all the cubes into a dustbin. In the **sandwich task**, two agents work together to pick up ingredients and stack them according to a given recipe. The **sort task** requires three agents to place three cubes onto coasters with matching colors. Since each agent can only reach a limited area, they must coordinate their movements. Lastly, the **rope task** involves agents moving a rope into a bracket. They must communicate effectively to decide the correct path for maneuvering the rope.

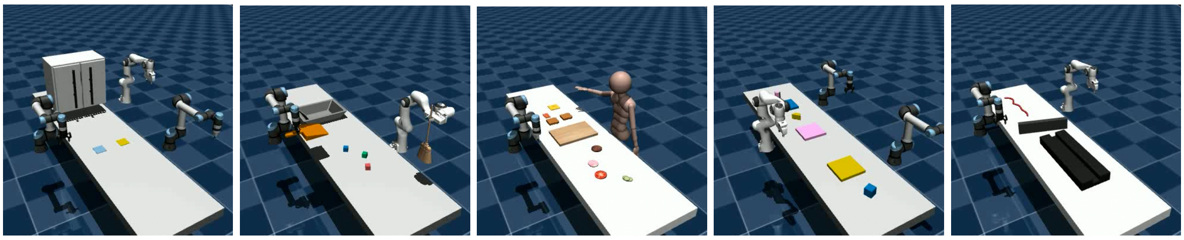
## E Simulated Environment for Team Formation Evaluation

To construct a simulated environment for evaluating the team formation mechanism, we employ GPT-4-1106-preview to generate a diverse set of tasks and agents. The dataset construction process involved the following steps:

1. Task Generation:
    - Using ChatGPT-4, we generate 399 distinct categories of theme keywords, covering various domains such as sports, lifestyle, and entertainment.
    - From these categories, we randomly select 25 themes and task GPT-4 with generating task descriptions related to at least four themes from the selected set, thus obtaining a task that require diverse agents with different capabilities.
    - Task descriptions are generated in JSON format using the GPT-4 API, ensuring a structured and consistent representation.
2. Agent Generation:
    - After generating the tasks, for each task, we again prompt GPT-4 to construct at least two agents with varying capabilities for the given task, including the name of the agent and the description of the agent.
    - The agent profile format is designed to align with the server-side agent registry, facilitating seamless integration and interaction within IoA.

An example of a generated task description in JSON format is as follows:

```
{
  "task_id": "xxx",
```



(a) Cabinet (b) Sweep (c) Sandwich (d) Sort (e) Rope

Figure 9: The different environments in RocoBench.

```
  "task_description": "Develop a mobile app that helps users plan and manage their personal finance, including budgeting, expense tracking, and investment suggestions."
}
```

Similarly, an example of an agent profile in JSON format is:

```
{
  "agent_name": "FinanceGuru",
  "agent_description": "FinanceGuru is a highly skilled agent specializing in personal finance management. It has extensive knowledge of budgeting techniques, expense tracking tools, and investment strategies. FinanceGuru can provide personalized recommendations based on a user's financial goals and risk tolerance."
}
```

By generating a diverse set of tasks and agents with varying capabilities, we create a comprehensive simulated environment for evaluating the team formation mechanism. This environment enables us to assess the effectiveness of IoA in assembling appropriate teams based on task requirements, addressing the limitations of existing benchmarks in providing suitable large-scale agent evaluation scenarios.

Please complete the function according to its comment.
def minimumTime(grid: List[List[int]]) -> int:
"""
You are given a m x n matrix grid consisting of non-negative integers where grid[row][col] represents the minimum time required to be able to visit the cell (row, col), which means you can visit the cell (row, col) only when the time you visit it is greater than or equal to grid[row][col].

You are standing in the top-left cell of the matrix in the 0th second, and you must move to any adjacent cell in the four directions: up, down, left, and right. Each move you make takes 1 second.

Return the minimum time required in which you can visit the bottom-right cell of the matrix. If you cannot visit the bottom-right cell, then return -1.

Example 1:

Input: grid = [[0,1,3,2],[5,1,2,5],[4,3,8,6]]
Output: 7
Explanation: One of the paths that we can take is the following:
- at t = 0, we are on the cell (0,0).
[...]

Constraints:

[...]
"""

After you complete the function, display the content of the script as res.py directly.

**Coding**

In a country, there are cities connected by one-way roads. It's known that from any city, there is a route (possibly passing through other cities) leading to the capital. Prove that it's possible to choose one road from each city in such a way that all chosen roads lead directly or indirectly to the capital.

**Math**

Review three smartphone models (Apple iPhone 13, Samsung Galaxy S22, and Google Pixel 6) based on camera quality, battery life, user interface, and price to decide the best buy.

**Search & Report:**

I am a 35-year-old software engineer who is vegan and looking to optimize for a balanced diet containing 2500 calories per day. Create a personalized weekly meal plan for me. Include three meals and two snacks per day, paying close attention to incorporating a variety of protein sources to meet daily protein needs. Provide a detailed grocery list that organizes ingredients by aisle for a standard grocery store layout.

**Life Assistant**

Figure 10: Example instructions from different categories in our open-ended instruction benchmark