EasyChair Preprint

№ 14692

Enhancing Efficiency and Flexibility of Rapid Prototyping for Scalable Multimodal Intelligent Agents

Muthukumarapandian Chandrasekaran

September 5, 2024

Enhancing Efficiency and Flexibility of Rapid Prototyping for Scalable Multimodal Intelligent Agents

Muthukumarapandian Chandrasekaran[1]
[1] *Ascendion Inc., New Jersey, USA*
muthuchand2013@gmail.com

Abstract. This paper explores the enhancement of efficiency and architectural flexibility in the rapid prototyping and scalable deployment of multimodal intelligent agents through the frameworks SmartTaskAgent (i.e. AutoGen) and CollaborativeAI's (i.e. CrewAI). By integrating large language models (LLMs), these frameworks significantly improve agent functionalities and streamline multi-agent workflows. The discussion addresses key challenges such as role-playing capabilities, prompt robustness, hallucination mitigation, and scalability, along with proposed solutions. The findings suggest substantial advancements in AI applications across various industries, highlighting the potential of these frameworks in enabling rapid prototyping and scalable deployment.

Keywords: *Multi-agent conversation framework, LLM, Large Language Model, agent, Generative AI*

I. INTRODUCTION

In recent years, large language models (LLMs) have demonstrated remarkable potential, with their ability to mimic human intelligence improving rapidly. [1] [2]. This has opened new avenues for using LLMs as fundamental controllers to develop autonomous agents with decision-making abilities akin to humans. These agents, leveraging LLMs for logical reasoning, effective tool utilization, and adaptability, are increasingly being deployed across a variety of real-world applications. [3] [4] [5]. This paper investigates strategies to further the development of multimodal LLM applications that span a wide range of fields and complexities, particularly in enhancing the efficiency and scalability of these systems.

As mentioned in [6], the approach is to use multi-agent conversations to achieve the objectives. Recent advances in LLMs confirm its general feasibility and utility for three key reasons. Firstly, chat-optimized LLMs has demonstrated the ability to incorporate feedback, allowing LLM agents to collaborate through conversations including dialogues where agents support reasoning, observations, critiques, and validation. Secondly, when appropriately configured with prompts and inference settings, a single LLM demonstrates a wide array of capabilities. These skills can be effectively combined in a complementary and modular fashion through conversations among agents with diverse configurations. LLMs have shown the capability to solve difficult tasks by breaking them into simple and smaller subtasks. This division and integration may be done naturally through multi-agent dialogues. So, in the next section multi-agent conversation framework is discussed.

II. MULTI-AGENT CONVERSATION FRAMEWORK: STUDY

To facilitate the creation of complex LLM-based applications across diverse domains, the multi-agent conversation framework adheres to a core design principle of consolidating workflows through multi-agent conversations. This section introduces two foundational concepts: conversable agents and conversation programming. [6] [7].

A. Conversable Agents

In a multi-agent conversation framework, a conversable agent acts as an entity with a designated role, facilitating the exchange of messages for sending and receiving information among other agents. These agents can be equipped with various capabilities, such as those enabled by large language models (LLMs), tools, or human input, and operate according to programmed behavior patterns.

The framework enables agents to utilize capabilities from LLMs, human interactions, and tools, thereby shaping their message processing and responses. This flexibility allows agents to perform di-

verse functions, such as role-playing, state inference, and adaptive learning from conversation history. These agents can handle tasks like coding, providing feedback, and adapting through novel prompting techniques.

In this context, a conversable agent is an entity that facilitates the exchange of messages among other conversable agents, maintaining its internal context based on the messages it processes. These agents can be configured with a variety of capabilities, operating according to behavior patterns outlined in the framework. LLM-backed agents, for example, leverage advanced features such as role-playing, state inference, and adaptive learning to enhance their coding tasks and feedback mechanisms. They also benefit from features like result caching and error handling. Human interaction is supported through human-backed agents, which enable participation based on configurable engagement levels and interaction patterns. Tool-backed agents execute tools via code or function execution, as seen in the default user proxy agent's ability to execute LLM-suggested code or function calls.

The framework supports the creation of customized agents tailored to specific applications by extending built-in templates. For example, the Conversable Agent class serves as the foundational abstraction, supporting integration with LLMs, humans, and tools. Subclasses like AssistantAgent and UserProxyAgent are pre-configured for specific roles: the AssistantAgent functions as an AI assistant using LLMs, while the UserProxyAgent acts as a human interface, managing human inputs and executing code or function calls.

As shown in Fig 1, In this framework, conversable agents serve as fundamental building blocks, enabling custom agents to communicate with one another. However, to create applications where agents can effectively advance tasks, developers must be able to define and structure these multi-agent conversations, as discussed in the subsequent sections.
In the multi-agent conversation framework, a conversable agent functions as an entity with a designated role, facilitating the exchange of messages for sending and receiving information among other agents. These agents can be configured with various capabilities, such as those enabled by LLMs, tools, or human input, and operate according to programmed behavior patterns.
The framework enables agents with capabilities from LLMs, humans, and tools, shaping their message processing and responses. This flexibility allows for diverse agent functionalities, such as role-playing, state inference, and adaptive learning from conversation history. These agents can perform coding tasks, provide feedback, and adapt using novel prompting techniques.

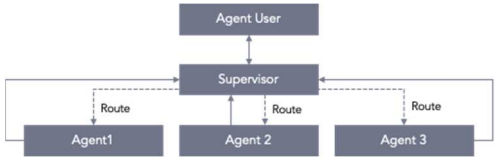

Fig 1: Built-In Multi-Agent Framework – Agents [i]

[i][https://blog.langchain.dev/langgraph-multi-agent-workflows/]

B. Conversation Programming

The framework employs conversation programming, focusing on two main concepts: computation and control flow. Computation involves the actions agents take to generate responses within a conversation, while control flow refers to the sequence or conditions under which these computations occur. The framework supports both static and dynamic conversation flows, allowing for intuitive reasoning about complex workflows through agent interactions.

In multi-agent framework, computations are centered around conversations. An agent performs actions based on the conversations it is involved in, with these actions leading to message passing for subsequent conversations unless a termination condition is met. Similarly, the control flow is guided by conversations—the decisions agents make about which other agents to communicate with, and the sequence of computations depend on the inter-agent conversation. This approach allows for intuitive reasoning about complex workflows through the interaction between agent actions and conversation message passing.

In the conversation programming paradigm, diverse multi-agent conversation patterns can be realized. It supports both static conversations with predefined flows and dynamic conversation flows involving multiple agents.

A customized generated reply function allows an agent to manage the current conversation while initiating interactions with other agents based on the current message and context. Additionally, the LLM determines whether to call a specific function depending on the conversation's status. By involving additional agents through these function calls, the LLM can facilitate dynamic multi-agent conversations.

It also supports more complex dynamic group chats through the built-in GroupChatManager, which can select the next speaker and broadcast its response to other agents. In the next section challenges are discussed.

III. CHALLENGES

While prior research into LLM-based autonomous agents has achieved considerable successes, this field is still emerging and presents several notable challenges that require addressing in its development. Below, we outline several representative challenges.

A. Role-playing capability

Autonomous agents must often assume specific roles, such as coder or researcher, to complete various tasks. Fine-tuning LLMs with data specific to uncommon roles or psychological traits and designing tailored prompts can enhance role-specific performance, though maintaining effectiveness for common roles remains challenging. Furthermore, research suggests that LLMs often fail to accurately model human cognitive psychology, which leads to a lack of self-awareness in conversations [8].

To address these challenges, potential solutions include fine-tuning LLMs with data specific to uncommon roles or psychological traits and designing tailored prompts and architectures [9]. Fine-tuning can enhance role-specific performance but maintaining effectiveness for common roles remains challenging. Alternatively, creating specialized prompts and architectures can improve role-playing abilities, though the extensive design space makes this a complex task.

B. Prompt robustness

LLMs often require complex prompts to ensure logical behavior in agents. However, slight modifications to these prompts can produce vastly different results. Creating a cohesive and robust prompt structure that works across different LLMs is a significant challenge. Previous studies have revealed a notable deficiency in the robustness of prompts for LLMs, whereby slight modifications might provide very disparate results [10], [11]. This issue is further amplified in the context of autonomous agents, which utilize a prompt framework encompassing multiple modules. In such frameworks, a prompt designed for one module can have direct impact on other modules, and the structure of frameworks can change drastically between different LLMs. The creation of a cohesive and robust prompt structure that works with different LLMs is still a significant and unmet task.

Two possible approaches have been proposed to deal with these issues: using GPT to automatically produce prompts or manually creating necessary prompt items through iterative trial and error.

C. Hallucination

Hallucination, where LLMs generate erroneous information with high confidence, is a critical issue for autonomous agents. For example, research has shown that agents can exhibit hallucinatory behavior when given simplistic instructions during code generation tasks, resulting in misleading or incorrect code, ethical issues and security risks [12]. To solve this problem, incorporating human corrective feedback into the iterative process of human-agent communication has been proposed as a viable solution [13].

D. Knowledge boundary

LLMs sometimes possess knowledge far exceeding that of an average person, which can affect the realism of agent simulations. Constraining LLMs' use of knowledge to more closely match human limitations is essential for creating realistic simulation environments. A functional and key application of

LLM-based autonomous agents involves incorporating a variety of real-world human behaviors. The research on human behaviors simulation is well-established, and recent advancements in LLMs have significantly enhanced these capabilities. However, the powerful nature of LLMs can sometimes be a double-edged sword. An effective simulation should faithfully replicate human knowledge, yet LLMs, trained on extensive web-based corpora, often possess knowledge far exceeding that of an average person.

For instance, when simulating user behavior in selecting movies, it is essential that LLMs operate without prior knowledge of the movies. Nevertheless, LLMs might have already obtained information about these movies, leading to decisions based on extensive knowledge that real-world users would not possess. This challenge underscores the importance of constraining LLMs' use of knowledge that users would not typically possess to create realistic agent simulation environments [14]. In the next section, based on the attribute author has compared two frameworks SmartTaskAgent and CollaborativeAI.

IV. Comparative Analysis of SmartTaskAgent and CollaborativeAI

SmartTaskAgent and CollaborativeAI serve distinct purposes within the AI landscape. SmartTaskAgent is tailored for tasks requiring high-quality content and code generation with ease of use, making it ideal for smaller projects and individual users. CollaborativeAI, with its extensive range of AI capabilities and scalability, is better suited for large enterprises seeking comprehensive AI solutions. The choice between the two platforms will largely depend on the specific needs, scale, and technical expertise of the user or organization.

TABLE 1:COMPARATIVE ANALYSIS OF SMARTTASKAGENT VERSES COLLABORATIVEAI

Features	***SmartTaskAgent***	***CollaborativeAI***
Overview	AI-driven tool for generating content and code	Comprehensive AI platform offering NLP, computer vision, and data analytics
Key Features	Content generation, Code assistance, Customization	NLP capabilities, Computer vision, Data analytics
Strengths	Efficiency: Automates tasks Accuracy: High-quality, coherent content User-friendly: Intuitive interface	Versatility: Broad range of AI applications Scalability: Handles large-scale projects Integration: Seamlessly integrates with business systems
Weaknesses	Dependency on training data Limited scope: Focused on content and code	Complexity: Steeper learning curve Cost: Potentially expensive for small businesses
Efficiency and Flexibility	SmartTaskAgent's modular architecture allows for rapid prototyping by enabling developers to easily integrate and test different components, thereby reducing development time and effort.	CollaborativeAI, with its robust orchestration capabilities, ensures seamless communication between multiple agents, enhancing their collective efficiency in performing complex tasks.
Scalability	Modular scalability, focusing on the independent scaling of specific components for rapid iteration and efficient resource utilization.	Holistic scalability, emphasizing seamless integration and management of multiple agents for large-scale, high-concurrency environments.
Scope of Functionality	Specialized in content and code generation	Broad range of AI services (NLP, computer vision, etc.)
Ease of Use	User-friendly for technical and non-technical users	May require more technical expertise
Customization and Integration	Fine-tuning models for specific needs	Comprehensive integration options for enterprises
Cost Efficiency	Cost-effective for smaller projects or individuals	Higher investment but justified for larger enterprises
https://docs.crewai.com/ https://microsoft.github.io/autogen/docs/Getting-Started/		

Hence, SmartTaskAgent is a cutting-edge AI framework that streamlines the creation, deployment, and interaction of AI agents. It enables agents to exchange messages and generate replies using models, tools, human inputs, or a combination of these, effectively representing both real-world and abstract entities such as individuals and algorithms. The framework simplifies the implementation of

complex workflows through agent collaboration, facilitating the design and execution of practical workflows. The latest version of SmartTaskAgent features Dynamic Group Chat and Finite State Machine Graphs, which offer flexibility by allowing contextual selection of the next speaker and specifying legal or prohibited transitions, respectively. This extensibility and composability allow for the enhancement of simple agents with customizable components, resulting in modular and easy-to-maintain sophisticated agents. Despite these strengths, there are areas where improvements are needed. The current method for specifying an agent's role and persona is limited, as system messages can be challenging for designers to create effectively. Additionally, the process of configuring tools and enabling function calling within SmartTaskAgent is not intuitive, lacking necessary abstractions and restricting tool use to models that support the OpenAI-compatible tool call API. The max_round attribute in GroupChat, which sets a limit on the number of conversations turns, is suboptimal because it doesn't allow for a dynamic assessment of the conversation's value. Furthermore, the focus on conversable agents is restrictive, as many business workflows do not involve conversation and are ill-suited to being forced into such a framework. Enhancing these aspects would significantly improve the framework's usability and effectiveness across various applications.

In evaluating CollaborativeAI's framework for autonomous agents, several strengths and areas for improvement have been identified. The framework excels in its structured approach to defining agents through role, goal, and backstory attributes, which enhances clarity and methodical design. Explicit task definition and hierarchical scoping of tools further bolster its functionality, making it adaptable for various operational needs.

However, the framework could benefit from enhancements in several areas. It lacks clear support for contextual agent selection beyond Sequential and Hierarchical processes, with the Consensual process needing more definition. Additionally, the approach of assigning agents to tasks rather than the reverse may require rethinking for improved usability. The max iter attribute, used to limit agent turns, lacks guidance on optimal setting methodology, suggesting a need for more flexible termination conditions. Furthermore, while hierarchical processes are supported, the implicit creation of 'manager' agents should be made explicit, with provisions for multi-level hierarchy to better support complex workflows.

Overall, addressing these aspects would enhance CollaborativeAI's framework, making it more robust and adaptable for designing autonomous agents across diverse applications and workflows.

V. Conclusion

In conclusion, optimizing efficiency and architectural flexibility in rapid prototyping and scalable frameworks for multimodal intelligent agents represents a critical advancement in artificial intelligence. The integration of LLMs within frameworks such as SmartTaskAgent (i.e.AutoGen) and CollaborativeAI's (i.e. CrewAI) underscores their potential to significantly enhance the functionality of intelligent agents through efficient multi-agent conversations and dynamic interaction models. These frameworks facilitate the modular development of agents capable of performing a wide range of tasks with high accuracy and adaptability. However, to fully leverage these capabilities, there is a need to address current limitations such as the refinement of role-playing functionalities, robustness of prompts, and mitigation of hallucinations. By overcoming these challenges, LLM-driven frameworks can achieve greater architectural flexibility and efficiency, enabling the deployment of sophisticated, scalable multimodal intelligent agents that can adeptly handle complex, real-world scenarios across diverse applications.

REFERENCES

[1] B. B. Tom, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, "Language models are few-shot learners," in *34th Conference on Neural Information Processing Systems*, Vancouver, Canada, 2020.

[2] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozire, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave and G. Lample, "LLaMA: Open and Efficient Foundation Language Models," *ArXiv,* vol. abs/2302.13971, 2023.

[3] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," *ArXiv,* vol. abs/2210.03629, 2023.

[4] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, R. Zheng, X. Fan, X. Wang, L. Xiong, Y. Zhou, W. Wang, C. Jiang, Y. Zou, X. Liu, Z. Yin, S. Dou, R. Weng, W. Cheng, Q. Zhang, W. Qin, Y. Zheng, X. Qiu, X. Huang and T. Gui, "The rise and potential of large language model based agents: A survey," *arXiv,* vol. arXiv:2309.07864, 2023.

[5] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. Xin Zhao, Z. Wei and J. Wen, "A survey on large language model based autonomous agents," *Front. Comput. Sci.,* vol. 18, no. 186345, pp. 1-26, 2024.

[6] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. Awadallah, R. W. White, D. Burger and C. Wang, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," *arXiv,* vol. 2308.08155, pp. 1-43, 2023.

[7] J. Zhang, "EcoAssistant - Using LLM Assistants More Accurately and Affordably," Microsoft, 9 November 2023. [Online]. Available: https://microsoft.github.io/autogen/blog/tags/rag/. [Accessed 7 July 2024].

[8] K. A. Fischer, "Reflective Linguistic Programming (RLP): A Stepping Stone in Socially-Aware AGI (SocialAGI)," *arXiv,* vol. abs/2305.12647, pp. 1-12, 2023.

[9] C. Li, J. Wang, Y. Zhang, K. Zhu, W. Hou, J. Lian, F. Luo, Q. Yang and X. Xie, "Large Language Models Understand and Can be Enhanced by Emotional Stimuli," *arXiv,* vol. arXiv:2307.11760v7, pp. 1-32, 2023.

[10] T. Yue Zhuo, Z. Li , Y. Huang , F. Shiri , W. Wang , G. Haffari and Y.-F. Li, "On Robustness of Prompt-based Semantic Parsing with Large Pre-trained Language Model: An Empirical Study on Codex," in *17th Conference of the European Chapter of the Association for Computational Linguistics*, Stroudsburg, PA, USA, 2023.

[11] Z. Gekhman, N. Oved, O. Keller, I. Szpektor and R. Reichart, "On the robustness of dialogue history representation in conversational question answering: a comprehensive study and a new prompt-based method," *Transactions of the Association for Computational Linguistics,* vol. 11, no. 11, pp. 351-366, 2022.

[12] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto and P. Fung, "Survey of Hallucination in Natural Language Generation," *ACM Comput. Surv.,* vol. 55, no. 12, 2023.

[13] Y. Dong, X. Jiang, Z. Jin and G. Li, "Self-collaboration Code Generation via ChatGPT," *ACM Trans. Softw. Eng. Methodol.,* 2024.

[14] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang and M. S. Bernstein, "Generative Agents: Interactive Simulacra of Human Behavior," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, San Francisco, CA, USA, 2023.