CHAIN-OF-COMMAND FRAMEWORK (COCF) FOR LLM AGENTS

Shinyu Yi
shinyu@keio.jp

Hayashi Keiichi
kei1884@keio.jp

Natsumi Machida
machida-natsumi.7611@keio.jp

Rika Morita
rikamorita0201@keio.jp

ABSTRACT

This paper introduces the Chain-of-Command Framework (CoCF), a language and framework for coordinating Large Language Model (LLM) agents in a hierarchical structure. Inspired by military hierarchies, CoCF aims to improve the efficiency and effectiveness of LLM agents in complex tasks by establishing clear roles, responsibilities, and communication protocols within a hierarchical structure. CoCF addresses limitations of current LLM agents, such as hallucination and limited context windows, by enabling specialized agents to collaborate and reason collectively. The paper details the design and implementation of the CoCF language, including its syntax, semantics, and communication protocols, incorporating principles from the AGILE and MOA frameworks. It also presents a prototype system with different LLM agents interacting within the CoCF hierarchy and discusses potential applications, challenges, and future research directions.

1 Introduction

Large Language Models (LLMs) have shown remarkable capabilities in various domains, including natural language understanding, text generation, and problem-solving. Recent research has explored the potential of using LLMs as agents that can interact with each other and their environment to perform complex tasks [1, 2, 3, 4]. However, existing LLM agents face limitations such as hallucination, where they generate incorrect or nonsensical information, and restricted context windows, limiting their ability to handle long documents or conversations [5]. Additionally, coordinating multiple LLM agents effectively remains a challenge. Existing approaches often lack clear hierarchical structures and communication protocols, leading to inefficiencies and potential

conflicts, as highlighted in [6] which mentions the absence of comprehensive benchmarks addressing these challenges in real-world agent applications.

This paper introduces the Chain-of-Command Framework (CoCF) to address these challenges. Inspired by military hierarchies, CoCF establishes a clear chain of command among LLM agents, with different agents assigned specific roles and responsibilities. This hierarchical structure facilitates efficient task delegation, feedback, and coordination, mitigating the limitations of individual agents by enabling collective reasoning and specialized task execution. The CoCF language provides a formal way for agents to communicate and interact within the hierarchy, ensuring clear and unambiguous communication.

The paper is organized as follows: *Analysis* analyzes key concepts and ideas relevant to the context by drawing connections between different approaches to inform the design of CoCF. *Design* details the design of the CoCF language, including its syntax, semantics, and communication protocols, incorporating principles from the AGILE and MOA frameworks. *Implementation* describes the implementation of the CoCF framework and the prototype system. *Experimental Design* presents the experimental setup and discusses evaluation metrics, including the Pareto frontier and cost-aware comparisons. Finally, *Discussion and Future Directions* and *Conclusion* discuss potential applications, challenges, and future research directions.

2 Analysis

Large language models (LLMs) can act as powerful problem-solvers yet remain prone to issues such as hallucinations and limited context handling [1, 2, 3, 4]. Researchers have noted that multi-agent configurations help mitigate these drawbacks by enabling agents to verify one another's outputs and distribute cognitive load, thereby improving robustness and efficiency [7, 8, 9, 10]. Meanwhile, the notion of hierarchical organization–drawn from both AI and human team structures–suggests that a clear chain of command, when supported by well-defined communication protocols like KQML or FIPA-ACL, can streamline information exchange and align agents toward shared objectives [11]. Researchers further propose evolutionary algorithms as a means of optimizing these multi-agent systems, demonstrating how iterative processes can fine-tune factors such as prompt strategies and agent configurations [12]. Beyond these technical mechanisms, studies on frameworks such as AGILE emphasize the value of planning, reflection, and experiential learning for enhancing adaptability in LLM-driven systems [6], while the Mixture of Agents (MOA) approach illustrates how combining diverse LLMs can produce synergistic gains [8]. These findings not only highlight the complexity inherent in orchestrating multiple intelligent agents but also illuminate the promising avenues for designing CoCF, from reinforcing hierarchical structures and communication protocols

to harnessing evolutionary optimization and multi-agent collaboration for more dynamic, scalable outcomes.

3 Design

3.1 Design of CoCF Language

The CoCF language is designed to be human-readable and easily interpretable by LLMs. It uses a simple syntax based on keywords and natural language instructions, enabling seamless communication and coordination within the hierarchical agent structure.

3.2 Syntax and Semantics

The basic structure of CoCF is as follows:

```
<Rank> <Agent ID>: <Command> <Task Description> <Parameters>
```

The *Rank* indicates the hierarchical level of the agent issuing the command (e.g., General, Officer, Soldier), establishing the authority and responsibility for task delegation and feedback. The *Agent ID* is a unique identifier for the agent, ensuring clear identification and addressing within the hierarchy. The *Command* specifies the action to be performed (e.g., *EXECUTE*, *REPORT*, *REQUEST*), defining the core communication protocols for task management and coordination. The *Task Description* provides a natural language description of the task, allowing for flexible and expressive task assignments. Finally, *Parameters* are optional parameters for the task, providing further specificity and control over task execution. For example, a General agent might issue the following command to an Officer agent:

```
General G1: EXECUTE "Analyze market trends for electric vehicles"
{ timeframe: "next 5 years", region: "North America" }
```

This command instructs the Officer agent with ID *O2* to perform the specified market analysis with the given parameters.

CoCF defines a hierarchical structure with different LLM agents assigned specific roles and responsibilities, inspired by military ranks. The General is the highest-ranking agent in the CoCF hierarchy, responsible for overall strategy and task delegation. Generals receive high-level objectives and decompose them into smaller sub-tasks, assigning these sub-tasks to Officer agents. They also oversee the progress of the entire operation, receive reports from Officers, and make adjustments to the strategy as needed. Officers are middle-ranking agents responsible for overseeing the execution of sub-tasks assigned by Generals. Officers further decompose these sub-tasks into even smaller

tasks and delegate them to Soldier agents. They monitor the progress of Soldier agents, provide guidance and support, and aggregate the results to report back to the General. Soldiers are the lowest-ranking agents responsible for executing individual tasks assigned by Officers. Soldiers may use tools, access external resources, or interact with the environment to complete their assigned tasks. They report their progress, results, and any encountered issues back to their supervising Officer. This hierarchical structure, inspired by military organizations, ensures clear lines of authority and responsibility, facilitating efficient task management and coordination. Each agent has a well-defined role and understands its responsibilities within the hierarchy, minimizing confusion and conflicts.

CoCF establishes clear communication protocols for task delegation, feedback, and coordination, drawing inspiration from established agent communication languages like KQML and FIPA-ACL. Higher-ranking agents delegate tasks to lower-ranking agents using *EXECUTE* commands, which include a clear task description and any necessary parameters. Lower-ranking agents provide feedback to higher-ranking agents using *REPORT* commands, including progress updates, results of completed tasks, and any potential issues encountered. Agents can use *REQUEST* commands to seek information or assistance from other agents within the hierarchy, facilitating collaboration and information sharing between agents. To further standardize communication, CoCF utilizes the concept of "protocol documents (PDs)" introduced in [13]. PDs provide a machine-readable description of the communication protocols, ensuring that all agents have a clear and consistent understanding of the communication rules. This reduces ambiguity and facilitates interoperability between different agents.

CoCF incorporates elements from the AGILE and MOA frameworks to enhance its capabilities. It integrates the AGILE framework's principles of planning, reflection, and learning [6]. Agents are encouraged to plan their tasks before execution, breaking down complex tasks into smaller, more manageable steps. After completing a task, agents reflect on their performance, identifying areas for improvement. This reflection process facilitates learning from experience, enabling agents to adapt and improve their strategies over time. CoCF also adopts the MOA framework's approach of combining multiple LLMs to enhance performance [8]. This can be achieved by having multiple Soldier agents work on the same task independently. Their individual outputs are then aggregated by an Officer agent, which synthesizes the different perspectives and generates a final result. This approach leverages the strengths of different LLMs and reduces the risk of individual biases or errors.

4 Implementation

The CoCF language is implemented using Python and the OpenAI API. A prototype system is developed with different LLM agents interacting within the CoCF hierarchy. The system includes separate classes for General, Officer, and Soldier agents, each with specific methods for communication, task execution, and feedback. These classes encapsulate the roles and responsibilities of each agent type within the CoCF hierarchy. A communication module handles message passing between agents according to the CoCF communication protocols, ensuring that messages are correctly formatted, routed to the appropriate recipients, and processed according to the defined protocols. A task execution module manages the execution of tasks by Soldier agents, providing mechanisms for agents to access and use tools, interact with external resources, and perform computations. A feedback and reporting module processes feedback from lower-ranking agents and generates reports for higher-ranking agents. It aggregates results, summarizes progress, and identifies potential issues to provide a comprehensive overview of the task execution. The prototype system also utilizes TextGrad, to "differentiate" text and prioritize novel-seeking behaviors.

5 Experiment Design

To evaluate the performance and effectiveness of CoCF, experiments are designed to compare its performance with single LLM agents and other multi-agent frameworks. The experiments focus on tasks that require coordination, planning, and problem-solving, such as collaborative writing, complex question answering, and strategic decision making. The experiments can utilize established benchmarks and datasets relevant to each task, such as the HotpotQA dataset for complex question answering and the MultiWOZ dataset for strategic decision making.

The results of the experiments are analyzed based on various metrics, including task completion rate, accuracy, efficiency, and communication overhead. In addition to these metrics, the evaluation incorporates the concept of "Pareto frontier" from [14] to analyze the trade-off between different performance metrics. For example, we can evaluate the trade-off between accuracy and efficiency, identifying CoCF configurations that achieve the best balance between these two objectives. The evaluation also considers cost-aware comparisons, as highlighted in [14], taking into account the computational cost of using different LLMs and the overall cost of running the CoCF framework.

6 Discussion and Future Directions

The CoCF language and framework offer a promising approach for coordinating LLM agents in complex tasks. The hierarchical structure and clear communication protocols facilitate efficient task

delegation, feedback, and coordination. By enabling specialized agents to collaborate and reason collectively, CoCF can mitigate limitations of current LLM agents, such as hallucination and limited context windows.

CoCF has the potential to be applied in various domains, including software development, content creation, and research and analysis. Challenges like scalability, adaptability, robustness, and explainability will be addressed through further research and development. The potential for agent protocol interoperability will be investigated to enable CoCF agents to interact with agents from other frameworks. Potential drawbacks, such as increased communication overhead and potential bottlenecks, will be analyzed and mitigated through strategies like distributed decision-making or redundant agents.

CoCF, as a framework, utilizes the principle of "inversion of control". This means that the framework itself dictates the flow of control, calling into the agent code as needed. This approach provides a standardized structure and ensures that agents adhere to the defined communication protocols and task execution procedures.

By addressing these challenges and exploring future research directions, CoCF can become a valuable tool for developing and deploying LLM agents in various real-world applications.

7 Conclusion

This paper introduced the Chain-of-Command Framework (CoCF), a novel language and framework for coordinating LLM agents. CoCF leverages a hierarchical structure inspired by military organizations and clear communication protocols to improve the efficiency and effectiveness of LLM agents in complex tasks. By incorporating principles from the AGILE and MOA frameworks, CoCF enables agents to plan, reflect, learn, and collaborate effectively. The paper detailed the design and implementation of the CoCF language, presented a prototype system, and discussed potential applications, challenges, and future research directions.

In conclusion, CoCF offers several key advantages: improved coordination, reduced hallucination, enhanced context handling, increased adaptability, and leveraging multiple LLMs. CoCF has the potential to significantly impact various domains, including software development, content creation, and research and analysis.

References

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

[2] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

[3] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), March 2024.

[4] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023.

[5] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model, 2023.

[6] Peiyuan Feng, Yichen He, Guanhua Huang, Yuan Lin, Hanchong Zhang, Yuchen Zhang, and Hang Li. Agile: A novel reinforcement learning framework of llm agents, 2024.

[7] Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need, 2024.

[8] Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities, 2024.

[9] Minghao Wu, Yulin Yuan, Gholamreza Haffari, and Longyue Wang. (perhaps) beyond human translation: Harnessing multi-agent collaboration for translating ultra-long literary texts, 2024.

[10] Pengyu Cheng, Tianhao Hu, Han Xu, Zhisong Zhang, Zheng Yuan, Yong Dai, Lei Han, Nan Du, and Xiaolong Li. Self-playing adversarial language game enhances llm reasoning, 2025.

[11] Wei Tao, Yucheng Zhou, Yanlin Wang, Wenqiang Zhang, Hongyu Zhang, and Yu Cheng. Magis: Llm-based multi-agent framework for github issue resolution, 2024.

[12] Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*, January 2025.

[13] Samuele Marro, Emanuele La Malfa, Jesse Wright, Guohao Li, Nigel Shadbolt, Michael Wooldridge, and Philip Torr. A scalable communication protocol for networks of large language models, 2024.

[14] Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. Ai agents that matter, 2024.