Journal of Space Operations & Communicator (ISSN 2410-0005), Vol. 21 No. 3, Year 2025

Developing AI Agents for Satellite Operations

Zhenping Li

ASRC Federal

Abstract

The large language model (LLM)-powered AI agent for satellite operations is a ReAct (reason and action) agent that senses, reasons, and acts within its environment in a feedback loop. The datasets for satellite states come in two forms: log messages, which are formatted strings, and telemetry, which consists of time series data. Since telemetry in satellite operations cannot be used directly by LLM-powered agents for reasoning, machine learning (ML)-based data processing is necessary to model time series data, predict its behavior, detect changes, and summarize them for the agent's reasoning and action. An agent fits naturally as a decision support component in a satellite digital twin (SDT), where satellite states are modeled and dynamically recalibrated. Changes in satellite states are detected through model-based recalibration and monitoring, serving as inputs for agents to analyze. Monitoring log messages employs in-context learning capabilities to understand transitions in satellite operations as a finite state machine (FSM), facilitating the detection of anomalies and providing insights into how and why they occur. Retrieval-augmented generation (RAG) is integrated with the ReAct agent to provide satellite domain knowledge as the context for the LLM query to reduce hallucinations. An SDT enables humans in the feedback loop to review actions and feedback, fostering reinforcement learning with human feedback (RLHF) to enhance the agent’s reasoning abilities. The model-based simulation in an SDT also offers a testing ground for various agent learning algorithms to improve the reasoning capability. The actions of AI agents in satellite operations leverage existing components in ground systems by sending requests to activate specific command procedures. This approach necessitates AI-centric ground enterprise services with a standard API and message format, such as the model-context-protocol (MCP), to facilitate communication between the agents and ground system components and ensure comprehensive log message management for space and ground assets. AI agents will usher in a new mission operation paradigm that promotes more autonomous operations and a unified human ground system interface with a standard set of natural language for directing components in ground systems.

Key Word: AI Agents, Digital Twin, Satellite Operations, Machine Learning, Finite State Machine.

1 Introduction

An agent[1] can be defined as an application that seeks to achieve a goal by perceiving its environment and acting upon it with the tools at its disposal. The development of AI agents began with the development of early computer programs designed to simulate reasoning and decision-making. The initial agents are the rule-based expert systems. The advent of machine learning in the 1990s accelerated the development of AI agents, enabling them to learn from data and adapt over time. Recent developments in AI agents powered by large language models (LLMs) offer significantly enhanced reasoning, planning, and learning capabilities. These capabilities enable highly autonomous decision-making and task execution based on a pre-defined objective, driving significant advancements across various domains. AI agents enhance operational efficiency and accuracy, lower costs, and facilitate personalized user experiences. Their capacity to learn and adapt over time makes them essential tools for driving innovation and achieving strategic goals. The agent approach in satellite operations is not new; the rule-based agent [2] was developed within the GMSEC framework to facilitate lights-out operations, which has been widely adopted in the industry. It retrieves satellite states from event messages that adhere to the GMSEC [3] standard, and command procedures are initiated if these states meet predefined criteria, such as the fail-over procedure in the event of a system component failure. The rule-based agent is simplistic, as it cannot address complex situations requiring understanding the context of satellite states and possessing analytical and reasoning capabilities.

The LLM-powered AI agents for satellite operations enhance the development of satellite digital twins (SDT)[4]. Since satellite operations are dynamic systems involving thousands of time-dependent datasets, the LLM-powered AI agent cannot use these datasets directly as inputs for reasoning and analysis. ML-based data analysis is essential for modeling time-dependent datasets and detecting changes, providing the agent with a summary of these changes. An SDT, which includes a data analysis component with data models for time-dependent datasets and an LLM-powered agent for decision support, creates a feedback loop that enables automated data monitoring, engineering analysis, and appropriate actions to optimize operations. This has been developed in the Advanced Intelligent Monitoring System (AIMS)[5]. The LLM-powered AI agent is a crucial component in an SDT for intelligent decision support based on outputs from AIMS, the satellite knowledge base, and the reasoning and analytical capabilities of LLMs. The AI agent can propose potential solutions to satellite anomalies or take appropriate actions to optimize satellite operations. The integration of model-based data analysis and recalibration in SDTs with LLM-powered AI agents establishes a feedback loop for dynamic systems that optimizes satellite operations based on mission objectives. Satellite operations are open systems influenced by external events and command procedures. A finite state machine (FSM) [6] can effectively describe an open and dynamic system, providing a framework for formulating the transitions among states triggered by either command procedures or external events. The state equation in satellite operations establishes the foundation for anomaly detection, model-based simulations, and the reasoning and action in a ReAct agent. The context for a command procedure and external events includes the initial and final states before and after a command procedure and/or an external event. Learning and understanding the context in satellite operations is essential for SDT to model telemetry datasets, and reasoning and actions in AI agents.

Significant challenges exist in developing LLM-powered AI agents within SDTs for satellite operations, including cybersecurity requirements, the agents' operation in real-time or near-real-time environments, and the integration of agents with ground systems. Datasets involved in satellite operations must remain secure, preventing the AI agent from communicating with LLM models over the open internet. An isolated AI infrastructure within the satellite ground enterprise is essential for LLM-powered AI agents on space missions. AI agents for satellite operations function in real-time or near-real-time contexts, necessitating very short latency in response time. This requirement creates specific software and hardware criteria for the local AI infrastructure. Integrating an AI agent with components in ground systems, such as telemetry and control, mission planning, and flight dynamics subsystems, in an enterprise ground system architecture is crucial. The enterprise ground system architecture, such as the GMSEC reference architecture[3], defines interface and message standards to enable agents to monitor all components in ground systems and take appropriate actions by sending directive request messages to those components. The research and development of the ground system architecture have expanded to include enterprise ground system services that encompass mission operations, data processing, and distribution[7]. AI agents and AI infrastructure will transform the ground enterprise service into an AI-centric ground enterprise service, and the model context protocol (MCP) [8,9] could be the standard for the interaction between AI agents and the ground system component, which has been widely adopted.

This paper aims to address several key issues in the development of AI agents within the framework of SDT, including agent functionalities, architecture, and implementation to meet cybersecurity and operational efficiency requirements and to integrate agents into existing ground systems. Section 2 illustrates how an AI agent functions as a decision-making component in an SDT and how it integrates with components in ground systems. Section 3 discusses the structure of ReAct agents in monitoring telemetry data and log messages, as well as the chatbot agent for user-agent interfaces. Section 4 describes developing AI agents by leveraging AI libraries and other technologies. The summary is presented in Section 5.

2 SDT with AI Agent in Ground Systems

An SDT with LLM-powered AI agents is a crucial component of the ground enterprise in space missions. Figure 1 illustrates the high-level data flow of an SDT within its ground system. It utilizes existing elements in the ground system to synchronize with its physical satellite and receive satellite telemetry, event log messages, and mission planning datasets. The datasets for satellite telemetry are time-series datasets that cannot serve as inputs for LLM-powered reasoning and analysis, necessitating an additional data analysis process to detect and examine potential changes in the telemetry data and summarize these changes in satellite states for the AI agents. AIMS implements the data analysis. The AI agent can leverage its satellite knowledge base and reasoning capabilities to determine actions based on the mission objectives.

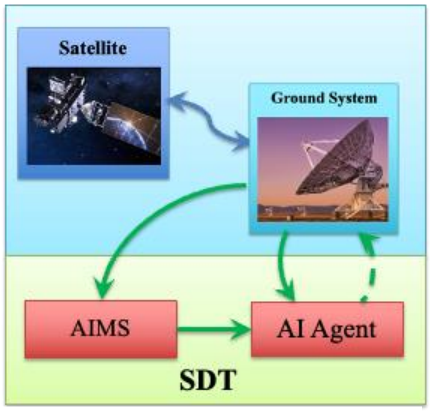

Figure 1. The context diagram of an AI Agent in an SDT shows the ground system sending telemetry to AIMS and event messages to the agent. The agent takes actions with a human in the loop.

AIMS involves creating and recalibrating models for highly diverse telemetry datasets and conducting a post-training analysis to detect changes in these datasets. Changes in a telemetry dataset can include data pattern variations defined as outliers or increases in the standard deviations of data training outputs. Data pattern changes in a dataset occur due to transitions between different states, regarded as events in satellite operations, triggered by command procedures or external events, which lead to data pattern changes in telemetry datasets. Each event may involve a single dataset or multiple datasets across various subsystems, as there are interactions or correlations among subsystems within a satellite. Event representation can link data pattern changes in specific datasets with the state equation in the FSM, thereby enabling anomaly detection: an event with an unknown trigger is viewed as a potential anomaly. The increase in standard deviations may point to a systematic difference between the data model and the actual data or an increase in the noise level of datasets, which requires further evaluation. The input to the LLM-powered AI agent in an SDT is a summary of satellite events and increases in standard deviations.

Since the number of telemetry datasets ranges from thousands to tens of thousands, managing model recalibration is complex due to efficiency, accuracy, and robustness requirements in near-real-time operational environments. The model recalibration in AIMS must also consider scalability and extensibility requirements in its software architecture to enable rapid deployment into any mission. Refs. 4 and 5 present detailed discussions on the operational concepts for model recalibrations in SDTs and the implementation of a hierarchical component architecture.

In addition to the telemetry data, the information from a ground system to an SDT includes event log messages and mission planning data. Event log messages consist of formatted strings that can be used directly as input for LLMs to monitor the health and safety of space and ground assets. The emergence of LLMs has led to significant research on system health and safety monitoring[10] via event log messages. Monitoring event log messages in AI agents is conducted within the context of the state equation of an FSM, enabling the prediction of a final state from a command procedure and an initial state. Unexpected changes in satellite states are potential anomalies. This process of context learning goes beyond merely monitoring string patterns or searching for keywords in the event log messages. Context learning provides actionable information on why and how an anomaly occurs.

The actions of AI agents are transmitted to components in ground systems as directive request messages, such as invoking a command procedure in a telemetry and control (T&C) component or initiating a mission planning session in a mission planning component. Therefore, integrating AI agents into ground systems for space missions necessitates an enterprise architecture with standard program interfaces and a message standard. This ensures that all components within the ground enterprise are interconnected, facilitating consolidated management of event log messages and monitoring by AI agents. An example of an enterprise ground architecture is the GMSEC[3,11], which defines the API and message standards between components in a ground system and middleware. The AI agents and AI infrastructure in space missions lead to AI-centric ground enterprise services, where the MCP can establish the communications between agents and ground system components, such as the MCP services for T&C, mission planning, and flight dynamics. This enables a close integration of ground system components with AI agents.

3 AI Agent for Satellite Operations

The AI agent for satellite operations is a ReAct (reason and action) agent[12,13] that senses its environment, performs reasoning, and takes actions based on specific objectives in a feedback loop. Figure 2 shows the orchestration of this AI agent that incorporates multiple LLM nodes, each of which plays a particular role as defined by its system message. Additionally, each LLM node is linked to a short-term memory that records previous queries and provides further context for LLM analysis.

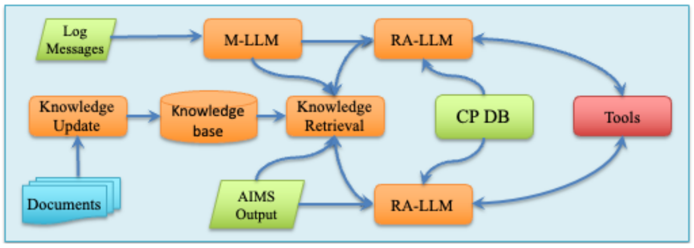

Figure 2. The architecture of an AI agent for satellite operations. M-LLM is the LLM used to monitor log messages, and RA-LLM is the Reasoning and Action LLM with an RAG architecture. The CP DB is the command procedure database for a specific mission.

The agent in Figure 2 has two input data sources: the log messages generated by components in both space and ground assets. Using in-context learning, the M-LLM node monitors mission health and safety [14]. The log messages are formatted strings containing a time tag, message type, and content; some may include a data source specifying the hardware or software. In the context of FSM, the log messages generally consist of two essential parts for satellite operations: the operational states of mission components and the status of command procedures that trigger state transitions in satellite operations. The initial and final operational states provide the context for the command procedures. Thus, monitoring log messages requires implementing a sliding window of log messages to query the LLM for contextual learning[15]. In the event of an error message, the LLM can learn the state information from earlier log messages to understand why and how an error occurred and generate actionable information for the RA-LLM node to reason and take appropriate actions. The sliding-window approach is implemented as the short-term memory attached to the M-LLM node, which can be configured during actual implementation.

The RA-LLM nodes in Figure 2 perform reasoning and analysis based on the M-LLM node and AIMS outputs, and take appropriate actions based on the command procedure database (CPDB). Since knowledge related to satellite operations is generally proprietary and mission-specific, it is not included in the input data for LLM training. This leads to a deficiency in domain-specific knowledge and updated information about satellites. An RAG approach addresses this gap by retrieving specific domain knowledge from the satellite knowledge base as context for the original query to the LLM, thus reducing hallucinations. The basic RAG architecture[15] consists of two processes: the indexing model, which ingests documents related to satellite operations into an embedding store that is shown as the knowledge update routine, and the knowledge retrieval, which matches the input query with the information in the embedding store by calculating similarity in the vector representations. RAG is a rapidly developing research field[16], featuring more advanced techniques to improve retrieval accuracy and efficiency. Numerous options are available for embedding models used in document indexing, retrieval, and embedding stores that could be implemented as a simple database. One improvement in RAG regarding retrieval efficiency and accuracy involves creating two separate embedding stores for space and ground segments, as the issues concerning these assets do not correlate in most situations. The retrievals from the two embedding stores can be merged for the LLM query.

The input documents for the satellite knowledge base within the AI Agent include hardware and software manuals, user manuals for both space and ground assets, metadata for command procedures, a satellite command and telemetry database, and any documents related to satellite operations. These documents enable AI agents to troubleshoot anomalies and plan corrective actions by invoking the appropriate command procedures or recommending solutions to satellite engineers. The knowledge base must adapt to emerging states or issues in satellite operation, with manual or automatic updates, ensuring that the AI Agent remains current.

The RA-LLM nodes perform reasoning tasks for potential anomalies detected in log messages or AIMS outputs, using inputs from the satellite knowledge base as context to troubleshoot these anomalies, identify root causes, and propose actions to address the problems. Depending on the nature of the anomalies, the reasoning process may be complex or straightforward; some could be rule-based. One possible solution is to implement the chain of thought (COT)[17], which is defined by prompt instructions[18], to enhance reasoning outcomes. The CPDB defines the command procedures available in a mission. A command procedure λ in satellite operations leads to a state transition from an initial state S_i to the final state S_f, which can be expressed as

$$\lambda: S_i \to S_f \tag{1}$$

where the state $S = \{S_i^d,\ S_j^c\}$ is a collection of discrete S_i^d and continuous S_j^c variables. The continuous variable S_j^c represents a collection of time- and state-dependent datasets that can be modeled using machine learning algorithms. Thus, the entries in the CPDB follow a schema with the elements $\{S_i, name, A_i, S_f, c\_name\}$ corresponding to initial state, the procedure name, arguments for the procedure, final state, and the component name in a ground system that runs the command procedure. The CPDB also represents a set of rules, in which the initial state S_i determines the condition for invoking the corresponding procedure. The RA-LLM nodes select command procedures from the CPDB as the action outputs and invoke the tools to send directive request messages to the corresponding components in the ground systems.

The tools provide a review option for engineers to give feedback to RA-LLM nodes before executing a command procedure, as such procedures cannot be carried out in operational environments without verifying their effectiveness and safety. This is reinforcement learning from human feedback (RLHM), since satellite engineers are domain experts in satellite operations with well-defined procedures to manage most situations, particularly in legacy missions. The AI agent can also utilize the model-based simulation platform in an SDT to test new command procedures and implement the agent learning algorithms[19,20] to evaluate their effectiveness and safety.

Figure 3 illustrates a chatbot agent that provides a standard human-machine interface for satellite operations. The chatbot offers two functionalities: answering users' questions about the hardware and software of both the space and ground segments and invoking tools integrated with the LLM node, such as executing command procedures for components in ground systems or directly sending commands to the satellite. The advantage of the chatbot approach is that it standardizes the human ground system interface using a common set of natural languages, regardless of the mission-specific components in the ground systems. This enables a seamless transition from one mission to another. The tools may also include generating daily, weekly, or monthly reports and displaying the current operational status.

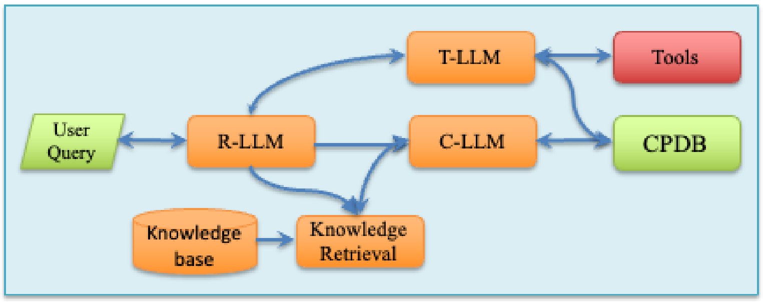

Figure 3. The chatbot agent for the human-machine interface. The R-LLM node is the routing node to forward the user query to either the T-LLM node for invoking the tools or the C-LLM for answering users' questions on software/hardware in space and ground segments. The CPDB and knowledge base are the same entities as those in the Figure. 2.

The structure of the chatbot agent resembles the adaptive RAG[21], where the R-LLM functions as a routing classifier for the input query. The T-LLM node invokes the appropriate tools based on users' input and is linked to the command procedure database, which is a specialized RAG structure. Since the CPDB contains structured datasets, the indexing process with an embedding model is no longer necessary. The C-LLM node is used to answer users' queries on hardware/software questions in both space and ground segments. It has the same RAG structure as the RA-LLM nodes in Figure 2.

4 The AI Agent Development

There are some system-level requirements for AI agents in satellite operations:

1. Due to cybersecurity requirements, local AI infrastructure is required for running LLMs, which requires both hardware and software.
2. The real-time or near-real-time operational environments lead to the performance requirement for running LLM models.
3. AI-centric ground enterprise services featuring the API and messaging standard for integrating AI agents with ground system components.
4. Cloud-based application environments that require AI agents to implement a REST microservice, since the future ground enterprise service will be cloud-based.

The local AI infrastructure requires a mini AI data center to operate the LLM inference engine or enable the fine-tuning of LLMs with domain-specific knowledge, which also addresses the performance requirement of the LLM query. Software platforms like Ollama[22] and Xinference[23] enable running LLMs on the local machine. These platforms offer many open-source LLMs and embedding models for the RAG.

The technologies for building AI agents have progressed rapidly since the emergence of LLMs, with many open-source libraries and development platforms available, including options like Langgraph and Langchain[24] in the Python environment, as well as Spring AI[25] and Langchain4J [26] in the Java ecosystem. These libraries provide APIs for creating agents with LLMs, enabling RAG by binding embedding models and embedding stores with LLMs while allowing customization for specifying model inputs and outputs and adding advanced techniques to improve RAG performance. There are numerous options for LLMs, embedding models, and embedding stores, making a trade study necessary to select the appropriate LLM, embedding model, and embedding store based on cost, efficiency, output accuracy, and security requirements. An important criterion for choosing an embedding store is its ability to provide a consolidated database server for both the embedding database and the CPDB. Furthermore, an AI agent must integrate the technology that provides the REST microservice for web applications to address the cloud computing requirements for the ground enterprise services. One of the technologies for REST microservices is Spring Boot[27], which offers a Java package for implementing the REST microservice, facilitating integrations between microservices and an agent. Thus, leveraging these libraries and platforms for agent development is crucial for reducing development risks and costs.

Finally, developing the AI agent using AI code assistance is crucial to speeding up development, testing, and documentation. AI code assistance has become more powerful with advanced LLMs; using it in software development is no longer optional; it has become a necessity. Agent development requires expertise in AI agent development platforms and libraries, REST microservices libraries, and prompt engineering. Considerable time would be needed to grasp the technologies involved in AI agent development. The extensive knowledge provided by AI code assistance allows for creating software routines that integrate these libraries based on users' requirements, thereby significantly enhancing development productivity. Software developers can learn these technologies while working with them, supported by AI code assistance.

5 Conclusion and Summary

The introduction of AI agents in satellite operations has the potential to usher in a new operational paradigm. The AI agent could act as the satellite operator, monitoring the performance of both space and ground assets and initiating command procedures when necessary to ensure that these assets operate at optimum levels. The AI agent could also function as a satellite engineer, overseeing satellite health and safety, troubleshooting anomalies by leveraging its reasoning capabilities and satellite knowledge base, and proposing possible workarounds and solutions. A chatbot agent provides a unified human-agent interface. It answers users' queries and offers tools to generate reports and initiate command procedures. This allows for a standard set of mission-control languages, regardless of mission specifics, significantly simplifying operational procedures using natural language without the need to interact with a mission's individual components or tools in ground systems.

The LLM-powered AI agent for satellite operations naturally fits as a decision support component in an SDT, where the ML model-based recalibration and monitoring detect changes in time series datasets and provide a summary of these changes to the agents for reasoning, analysis, and proposing or taking appropriate actions. The agent implements reinforced learning with human feedback (RLHF), allowing engineers in the feedback loop to review the action outputs from the agents, ensuring reliability and safety. An SDT offers model-based high-fidelity simulations, which create a rich testing ground to verify agent outputs and explore more advanced algorithms that enhance agent reasoning capabilities through reinforcement learning.

AI agents in ground systems lead to an AI-centric ground enterprise service, where the interfaces between AI agents and components in ground systems can be defined as MCP services that have been widely adopted. The AI-centric ground enterprise service presents new opportunities for developing LLM-powered AI agents. In addition to the AI agent for satellite operations, AI agents for AI-centric ground enterprise services also include agents for processing and distributing payload data and an AI workflow for mission planning, which are outside the scope of the current research. Further research on enhancing RAG accuracy and efficiency and improving the agents' reasoning capabilities is in progress.

References

[1] Xi Z H, Chen W X, Guo X, et al. The rise and potential of large language model based agents: a survey. Sci China Inf Sci, 2025, 68(2): 121101, https://doi.org/10.1007/s11432-024-4222-0.

[2] Zhenping Li and C. Savkli, "Autonomic computing for spacecraft ground systems," 2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06), Pasadena, CA, 2006, pp. 8 pp.-520, doi: 10.1109/SMC-IT.2006.21.

[3] See https://github.com/nasa/GMSEC_API.

[4] Zhenping Li, "Satellite Digital Twins", Journal of Satellite Operations & Communicator, July, 2024, https://opsjournal.org/DocumentLibrary/Uploads/AIAA-paper-DT-ZPLI.pdf

[5] Zhenping Li "Revolutionize Satellite Health and Safety Monitoring with Advanced Intelligent Monitoring System" ASRC Federal White Paper, August 2023, doi:10.13140/RG.2.2.20068.55686.

[6] E. A. Lee and S. A. Seshia, "Introduction to Embedded Systems - A Cyber-Physical Systems Approach," Second Edition, MIT Press, 2017.

[7] Steven R. Peterson, "Ground Enterprise Evolution at NESDIS", Space OPS 2018, May, 2018. DOI: 10.2514/6.2018-2403.

[8] Anthropic, "Introducing the Model Context Protocol", https://www.anthropic.com/news/model-context-protocol, Nov. 2024.

[9] Yingxian Yang et al. "A Survey of AI Agent Protocols", arXiv:2504.16736, DOI: 10.48550/arXiv.2504.16736, April 2025.

[10] Siraaj Akhtar, Saad Khan, and Simon Parkinson, "LLM-based Event Log Analysis Techniques: A Survey", arXiv:2502.00677v1, Feb. 2025. DOI: 10.48550/arXiv.2502.00677

[11] Zhenping Li, Cetin Savkli, and Dan Smith, "Increasing Operational Values of System Event Messages", The Fifth International Symposium on Reducing Cost of Spacecraft Ground Systems and Operations (RCSGSO) in 2003. https://descanso.jpl.nasa.gov/RCSGSO/Paper/A0031Paper.pdf

[12] Lilian Weng, "LLM-powered Autonomous Agents". Lil'Log. June 2023. https://lilianweng.github.io/posts/2023-06-23-agent/

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

[14] Hongwei Jin, George Papadimitriou, Krishnan Raghavan, Pawel Zuk, Prasanna Balaprakash, Cong Wang, Anirban Mandal, and Ewa Deelman. "Large Language Models for Anomaly Detection in Computational Workflows: from Supervised Fine-Tuning to In-Context Learning". arXiv preprint arXiv:2407.17545 (2024).

[15] Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, Thomas Back, “Reasoning with Large Language Models, a Survey”, arXiv: 2407.11511, (2024).

[16] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang, “Retrieval-Augmented Generation for Large Language Models: A Survey”, arXiv:2312.10997V5 (2024).

[17] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, Denny Zhou, “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”, arXiv preprint arXiv:2201.11903v6 (2023)

[18] Weng, Lilian. Prompt Engineering. Lil’Log. https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/, May 2023.

[19] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, Shunyu Yao, “Reflexion: Language Agents with Verbal Reinforcement Learning”, arXiv:2303.11366, Oct. 2023.

[20] Vishnu Sarukkai, Zhiqiang Xie, Kayvon Fatahalian, “Self-Generated In-Context Examples Improve LLM Agents for Sequential Decision-Making Tasks”, arXiv:2505.00234v2, May 2025.

[21] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, Jong C. Park, “Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity”, arXiv:2403:14403v2, March. 2024.

[22] See https://ollama.com/ for Ollama.

[23] See https://inference.readthedocs.io/en/stable/index.html for Xinference.

[24] See https://www.langchain.com/ for Langchain and Langgraph.

[25] See https://spring.io/projects/spring-ai for the details of the Spring AI.

[26] See https://docs.langchain4j.dev/ for Langchain4J.

[27] See https://spring.io/projects/spring-boot for Spring Boot