Multi-Agent Financial Analysis System: From Real Time Data Fetching to Expert Advisory

Yash Srivastava
Netaji Subhas University of technology
Dwarka, Delhi, India
yash231203@hotmail.com

Vikas Garg
Lingayas Vidyapeeth
Faridabad, Haryana, India
vikas.dcrust@gmail.com

Dr. Tapsi Nagpal
Lingayas Vidyapeeth
Faridabad, Haryana, India
tapsiarora@gmail.com

Abstract

This paper suggests an automated financial analysis multi-agent system based on modular rule-based architecture implemented in Python. The design involves independent agents—DataFetcherAgent, NewsAgent, and FinancialExpertAgent—managed by a central CoordinatorAgent [1] [2]. Stock data is fetched using the yfinance library, and sentiment is emulated to replicate actual world news polarity. The framework uses deterministic principles based on financial metrics including price-to-earnings (P/E) ratio, volume, and price change to output explainable investment suggestions. Transparent and extensible in nature, the architecture eschews third-party cloud APIs and allows for complete execution. The output is a structured investment advice: type of recommendation (BUY/SELL/HOLD), risk, confidence score, and sentiment summary. This framework acts as a light and interpretable base for subsequent investigations in agent-based systems, financial intelligence, and rule-based AI and can be further extended to include autonomous agents, natural language processing, machine learning, or LLM-based advisory systems [3][4].

***Keywords*: Multi-agent systems, Financial analysis, Autonomous agents, Sentiment analysis, Investment advisory, Risk Management, Recommendations, LangChain, Agent2Agent, Model Context Protocol**

I. INTRODUCTION

The development of Multi-Agent Systems (MAS) has been instrumental in changing the way intelligent software is constructed through modularity, autonomy, and decentralized decision-making. In the world of financial analytics, MAS allows the view of a complex task that is broken up - data fetching, sentiment estimation, and the actual investment decision, into different agents, while tightly coordinating yet acting independently. This adheres to the notions of Distributed Artificial Intelligence (DAI), and Agent-Oriented Software Engineering (AOSE) and aids in building characteristics to the system such as scalability, fault tolerance, and modularity.

This paper proposes a rule-based financial analysis system that is based on a MAS framework. The MAS consists of lightweight agents related to data fetching, sentiment estimation and decision making, with each agent designed to act in an independent yet coordinated way. The system allows for independence of using premium APIs and cloud services by using data from public market data sources, permitting it to offer transparency, explainability, and full mode capabilities. This developed system is suitable for academic purposes, training purposes, and even semi-professional purposes. [5].

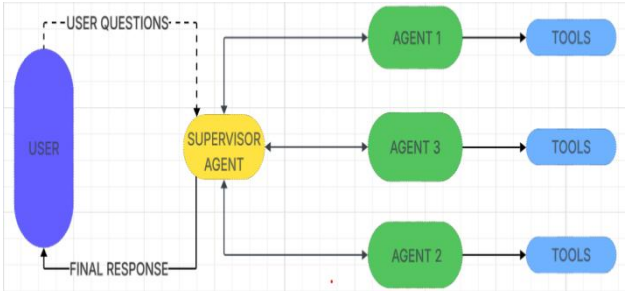

Figure 1: Multi Agent System

As AI systems continue to transform from self-contained, magic boxes specific to certain tasks to collaborative communities and ecosystems of multi-agent systems, the proposed architecture of the Multi-Agent Financial Analysis System can be enriched through the implementation of emerging agent orchestration frameworks. The agent orchestration frameworks mentioned: Agent-to-Agent Protocol (A2A), Model Context Protocol (MCP), and LangChain, are specifically suited to the modular agent design proposed in this paper.

A. Agent-to-Agent (A2A) Protocol

The A2A protocol enables asynchronous communication between autonomous agents, passing contextual

information and task ownership in a decentralized process. The current system has a CoordinatorAgent in charge of both command and control of all sub-agents, but in A2A the FinancialExpertAgent could query the NewsAgent directly if it wanted updated sentiment context, or could request additional data from the DataFetcherAgent if it was missing the P/E ratio [6]. Thus, enabling a more organic and scalable architecture where agents acted like domain specialists, they could negotiate with each other, or work collaboratively. In the high-frequency case, or in general distributed architecture, the system is able to manage fault tolerance, parallelism, and contextually aware agent adaptation, since no agent is solely responsible for everything [2] [7].

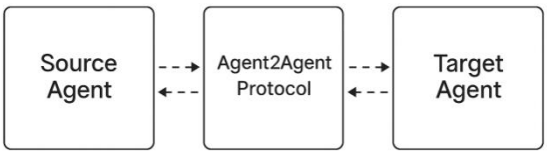

Figure 2: Agent 2 Agent Protocol

B. Model Context Protocol (MCP)

MCP provides a formal abstraction for tools and functional access assigned to each agent. For example, the NewsAgent might be associated with a particular web scraping tool or a pre-trained NLP model, while the DataFetcher might have secure API access to financial databases [8]. The protocol supports:

- Authorization of access (e.g. APIs, scraping tools)
- Model/tool routing (i.e. which model to call based on situational context)
- The encapsulation of execution context (i.e., local vs. cloud)

When an MCP structure is embedded, the system will enable each agent access to only the tools that they should be invoke, modifying behavior based on task requirements and user preferences [21] [22].This means that the system will be able to handle complex workflows such as switching between contextual models (e.g., use FinBERT for financial text, use LLM to summarize) or selection fallback of tools if network/API access is unavailable [9] [25].

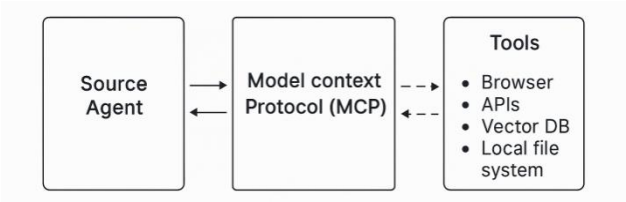

Figure 3: Model Context Protocol

C. LangChain for Agent Orchestration

LangChain offers a solid orchestration layer for multi-agent workflows, toolchains and memory sharing capabilities. In the framework of the proposed system, LangChain could be used as a task manager managing the order and execution of agents, as well as managing inputs/outputs, and possibly providing conversational user interfaces. So for example, the user could ask: "Analyze Apple's stock and tell me if I should buy it today." LangChain could direct that inquiry to the right agents in order: first call the DataFetcherAgent then the NewsAgent, and finally compose output from the FinancialExpertAgent [10] [11].

LangChain supports persistent memory so agents can remember state across queries, and this is a key enabler of an autonomous long-horizon analysis sessions, and long-horizon simulations. By integrating tools/schemas and chaining agents, LangChain positions this system for a future where they interact with the system, as a composite intelligence agent, instead of system performing a series of hardcoded steps.

II. RELATED WORK

A. Multi-Agent Systems in Financial Applications

A Multi Agent System (MAS) refers to a common distributed computational model of autonomous interacting software agents. Within the finance literature, MAS is used for modeling market dynamics, executing trading strategies, managing portfolios, and providing automated financial advice.

Each agent is autonomous; reacts to the environment they interact with, communicate and interact with other agents, and execute their task autonomously based on internal rules or learned policies. Also, MAS allows a complex financial workflow or task to be decomposed into manageable workflows of components and to be able to evolve, grow in complexity and interact asynchronously without interdependence [1] [2].

Let the multi-agent system be represented as:

$$MAS = (A, E, P, C)$$

- **A** = $\{a_1, a_2, \ldots a_n\}$: A finite set of agents

- **E**: The shared environment (e.g., market data, state variables)
- **P: A×E→Actions**: Perception function
- **C: A×A→M**: Communication function producing messages **m Є M**

In our case:

- a_1 = DataFetcherAgent
- a_2 = NewsAgent
- a_3 = FinancialExpertAgent
- a_4 = CoordinatorAgent

The representation, through modular multi agent architecture, satisfies principles of computational efficiency, explainability and flexibility making the model a good fit for real-time financial analysis [12].

B. Sentiment Analysis in Finance

Sentiment analysis is important today for financial systems because it quantifies the underpinning emotional tone of unstructured text data, such as news articles, earnings reports, and analyst commentary. Thus, it attempts to quantify investor sentiment, the mechanics of which drive market volatility and asset momentum [13] [14].

Let T = {t_1, t_2... t_n} be a set of financial texts (e.g., headlines, articles). A sentiment function S: T→ [−1, +1] maps each text to a polarity score, where:

- $S(t_i) = +1$: Strong positive sentiment
- $S(t_i) = 0$: Neutral sentiment
- $S(t_i) = -1$: Strong negative sentiment

The aggregate sentiment *S* for a stock s is:

$$S = \frac{1}{n}\sum_{i=1}^{n} S\,(ti),\ S(ti) \in [-1,1]$$

Where $S(t_i)$ is the polarity score derived from each textual input.

Sentiment analysis uses behavioral knowledge to provide more psychologically grounded financial recommendations and be able to do that through a completely interpretable system, which is a complementary analysis for rule based logic [15] [24].

C. Rule-Based Decision Systems

Rule-based decision systems are structured on rules of logic, so these systems will take deterministic IF-THEN rules backed up with an action scheme to generate outputs that you can take action on. When used for financial applications, the advantages of this system are immediately recognizable, such as the transparency, audibility (traceability) and reasoning within the domain [3].

In this example of decision-making, the FinancialExpertAgent processes input in terms of things like sentiment and the P/E ratio and provides a recommendation based on an understanding of the rules and logic but not using machine learning or APIs.

First, there is full explainability as each output is determined from specific conditions in the input - you can always go backwards from an output to conditions that generated a recommendation. Second, there is no training data and no tuning of models, meaning the implementation costs and resources are very low. Third, the rule-based logic is easily portable, which meets the goals of deployable financial analysis software. There are also advantages to embedding domain knowledge into deterministic rules as the agent for determination offered a stable platform for creating intelligent decisions and consistently presented powerful financial insight with clear explanations and potential audits on the fly [4].

III. SYSTEM ARCHITECTURE AND WORKFLOW

The planned financial analysis system is based on a multi-agent architecture; that is, each autonomous agent will have a specific and clearly defined role, based on DAI principles (distributed artificial intelligence), which allows for modularity, scalability and separation of concerns among the different parts of the system. The entire system is contained in a self-sufficient organism; therefore, it is useful for data analysis exercises where the status and prevision of secure and or controlled analysis is required [1][2]. The core system comprises four agents:

- **Coordinator Agent** (Orchestration)
- **DataFetcher Agent** (Market Data Retrieval)
- **News Agent** (Sentiment Simulation)
- **Financial Expert Agent** (Decision Engine)

Each agent is a class-based module in Python, instantiated and called in a sequence controlled by the coordinator.

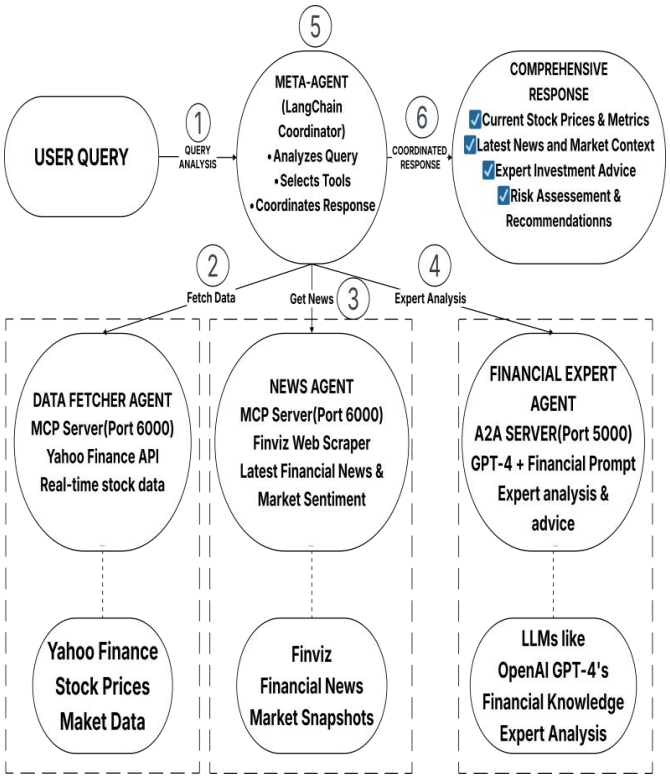

Figure 4: System Workflow

A. DataFetcher Agent

The DataFetcher Agent is the first data access sub-component of the system, which fetches both real and historic stock market data using the yfinance Python library, which is a programmatic access to the financial statistics available on Yahoo Finance. This design choice enables the full execution of the model. [16].

The agent is responsible for extracting key quantitative features required for financial evaluation, including:

- **Current Market Price** (P_{curr})
- **Previous Closing Price** (P_{prev})
- **Trading Volume** (V)
- **Market Capitalization** (M)
- **Price-to-Earnings (P/E) Ratio** (PE)

These metrics form the **financial state vector** F used in downstream decision-making:

$$F = [Pcurr, Pprev, V, M, PE]$$

The agent also computes derived indicators such as percent price change:

$$\Delta P\% = (\frac{Pcurr - Pprev}{Pprev}) \times 100$$

This quantitative data is sent as a structured dictionary object to the Coordinator Agent, where it is then combined with sentiment data for hybrid reasoning [5].

One of the many advantages of this agent is its integrated simplicity and extensibility. Even though it uses yfinance solely for prototype and educational purposes, it could easily be modified for use with authenticated data streams like Polygon.io or Alpha Vantage. Further, as it returns simple raw numerical features that are interpretable, it functions well in both rule-based systems, as well as statistical learner models.

B. News Agent

The News Agent serves as the behavioral analysis portion of the multi-agent architecture, forecasting market sentiment while operating in environments that do not offer NLP services via an API or where cost is a concern. In traditional financial systems, financial sentiment is gathered in real time using machine learning models from data acquired from news headlines, finance blog postings, and social media. The News Agent in this system generates sentiment using a more simplistic, rule-based approach [14] [15].

The agent generates three key sentiment-based features:

- **Polarity** (S) — a categorical value in {−1, 0, +1} representing negative, neutral, or positive sentiment
- **Confidence Score** — an integer from 0 to 100 quantifying certainty of the assigned sentiment
- **Synthetic Summary** — a basic descriptive string emulating news-driven market commentary

Formally, the sentiment vector S output by the agent is defined as:

S = [Sentiment Polarity, Confidence Score]

The structure is designed to the output provided by NLP-based financial sentiment classifiers. In other words, segmentation of the problem into downstream agents (i.e. the Financial Expert Agent) can take place.
The potential exists in a future iteration or extended configuration of the agent to switch it out for broader sentiment pipelines that utilize natural language processing (BERT, FinBERT, or other LLM based Zero shot classifiers etc) with live API inputs from something

like NewsAPI, or the r/stocks subgroup on Reddit [17][18]. This modularity allows you to move from offline simulation to online deployment.

C. Financial Expert Agent

The Financial Expert Agent serves as the final decision-making element in the multi-agent architecture. It synthesizes both fundamental indicators, which are obtained by the DataFetcher Agent, and behavioral sentiment indicators, which are produced by the News Agent in order to provide investment recommendations that are interpretable by human investors [3]. This agent makes recommendations using a rule-based deterministic model that focuses on explainability.

The heart of this agent consists of a rule engine which evaluates the incoming market data and its corresponding sentiment polarity against a pre-defined logic file and returns a BUY recommendation, a SELL recommendation or a HOLD recommendation. The logic is defined using conditional statements based on expert heuristics [4]. Formally, let the recommendation rule be defined as:

- **BUY**: If sentiment = +1 and PE < 30 → Recommend BUY.
- **SELL**: If sentiment = –1 → Recommend SELL.
- **HOLD**: In all other cases →Recommend HOLD.

Each decision is provided with a confidence score (scaled from the sentiment component), a risk level (LOW, MEDIUM or HIGH), and a short analysis summary that explains the basis for the recommendation. This structure keeps the agent's output compliant with interpretability criteria, to comply with financial advice, meaning that it can be utilized for educational, research, and proof-of-concept dimensions of usability.

In addition, the agent is modular and would permit any increase to its capability. The developer could replace the current rule-set with fuzzy logic, a decision tree, or a probabilistic inference model - all depending on complexity and the availability of data. While simple, this design establishes a baseline for validating rule-based strategies and demonstrating explainable AI principles in finance.

D. Coordinator Agent

The Coordinator Agent is the main processing agent for the multi-agent financial analysis system. It manages the execution pipeline and the data flow between specialized agents, including the DataFetcher Agent, News Agent, and Financial Expert Agent [19]. This agent implements the Mediator design pattern in that it decouples agent interactions while maintaining a clear and ordered sequence of operations.

The Coordinator receives a single stock symbol SSS as input and performs the analysis in three sub-tasks:

- **Data Retrieval** - Calls the fetch() method of the DataFetcher Agent to retrieve structured numerical stock data, which is represented as F.
- **Sentiment Analysis** - Calls the analyze() method of the News Agent to simulate (or retrieve) behavioral sentiment, represented as S.
- **Recommendation** - Calls the generate() method of the Financial Expert Agent, passing both F and S, which returns a final recommendation R Є {BUY, SELL, HOLD}.

The coordination function C could be formally represented as a composition function:

$$C(S) = D(F, S)$$
$$= FinancialExpert(DataFetcher(S), NewsAgent(S))$$

Where:

- **S**: The stock symbol (e.g., AAPL, TSLA)
- **F**: Stock feature vector from DataFetcher Agent
- ***S***: Sentiment vector from News Agent
- **D**: Decision logic from the Financial Expert Agent

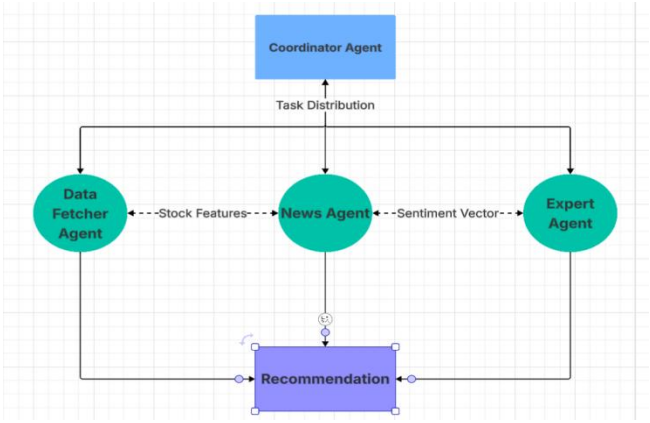

Figure 5: Interaction flow of the Financial Analysis System

In system design terms, the Coordinator Agent acts as:

- A **controller**: It controls the sequential invocation of agents.
- A **data integrator**: It aggregates and passes structured data between agents.

- A **result formatter**: It consolidates outputs into a final structured dictionary or JSON object for downstream use (CLI, API, or report generation).

IV. ALGORITHMIC DESIGN

Input: stock_symbol (e.g., "AAPL")
Output: Final Recommendation ∈ {BUY, SELL, HOLD}, along with confidence score and risk level.

The structured algorithm that outlines the core logic and flow of Multi-Agent Financial Analysis System is as follows:

1. Initialize the DataFetcherAgent
2. Initialize the NewsAgent
3. Initialize the FinancialExpertAgent
4. Fetch stock data using DataFetcherAgent.fetch(S)
5. If stock data is invalid or missing, go to Step 15
6. Perform sentiment analysis using NewsAgent.analyze(S)
7. Extract sentiment score and confidence level
8. If sentiment score is not valid, set it to neutral (0)
9. Generate recommendation using FinancialExpertAgent.generate(S, stock_data, news_data)
10. If sentiment score is +1 and PE ratio < 30, set recommendation to BUY
11. If sentiment score is –1, set recommendation to SELL
12. If none of the above conditions are satisfied, set recommendation to HOLD
13. Compile final result with recommendation, confidence score, risk level, and summary
14. Return the final output
15. Return default result with recommendation and risk as HOLD and risk level as HIGH due to data insufficiency

V. IMPLEMENTATION AND RESULTS

It is possible to launch the system from the command line or invoked programmatically via its CoordinatorAgent class. When the user enters a valid stock ticker symbol the following will happen:

Data Fetching:
The DataFetcherAgent utilizes the yfinance Python package to retrieve stock fundamentals including:

- Current market price
- Previous close
- Trading volume
- Market capitalization
- Price-to-Earnings (P/E) ratio

News Simulation:
The NewsAgent currently simulates the sentiment data. The NewsAgent will randomly assign a sentiment score (+1, -1, or 0) along with a confidence level as done in external news sentiment analysis pipelines.

Expert Rule Evaluation:
The FinancialExpertAgent applies a fixed rule set (as shown in Section III) to decide whether to BUY, SELL, or HOLD the asset. This decision is based on:

- The simulated sentiment value
- The P/E ratio threshold (typically < 30)

The result returned from the CoordinatorAgent.analyze(symbol) method is a structured JSON object containing:

- **stock_data**: Numerical financial indicators retrieved from yfinance.
- **news_data**: Simulated sentiment and confidence.
- **expert_analysis**: Final recommendation with confidence and explanation.

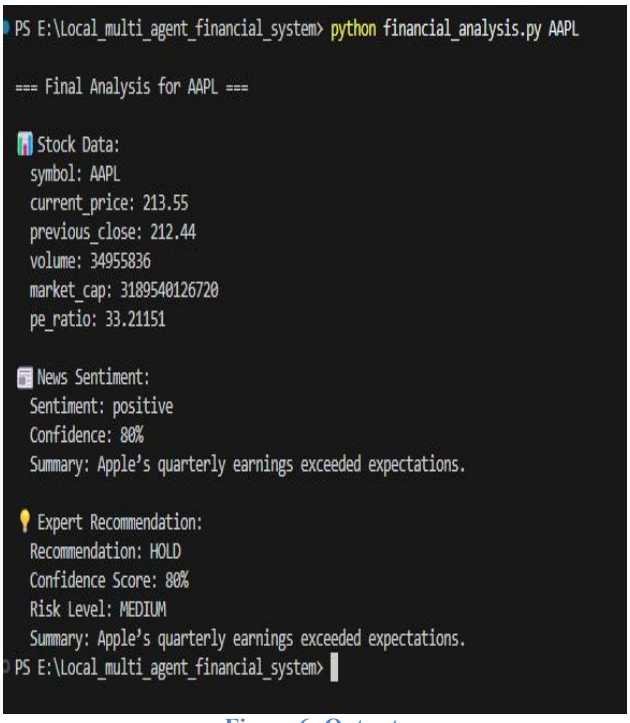

Figure 6: Output

VI. CONCLUSION AND FUTURE-SCOPE

This research presented the Multi-Agent Financial Analysis System, which consists of four interdependent agents: Data Fetcher, News Agent, Financial Expert, and a central Coordinator. After fetching stock data and sentiment indicators from several sources, the system considers multiple indicators and generates a human-readable output in the form of BUY, SELL, and HOLD recommendations. The agent-based system creates a modular solution and reduces future maintenance and upgrades while providing flexibility, and possibly

creativity, to practitioners working both in academic and professional environments[17][18].

The system output shows the benefits of employing rule-based logic and locally fetching information (using yfinance) to provide data-driven outputs for low-resource or air-gapped solutions that have privacy, cost, and interpretability requirements. The sentiment engine simulated sentiment to a certain extent, which ultimately allows it to output the sentiment it achieves while adhering to identifiable thresholds to make recommendation decisions; this can be done without heavy weight machine learning infrastructure, or external key management.

In the future, a primary pathway is increasing agent interoperability and tasking by providing autonomous coordination metrics. The modular nature of the system lends itself to the A2A (Agent-to-Agent) Protocol in which agents (e.g., DataFetcher, NewsAgent, FinancialExpert, etc.) would be free to communicate dynamically with each other without relying on a single centralized controller. Agents would be able to request, forward, or refine data with each other, simulating collaborative processes which we have observed in the professional financial world [24]. As an example, a FinancialExpert agent would be able to refer to a NewsAgent autonomously in dealt with conflicting signals, creating contextualization and flexibility in decision making [17][23].

The system has already adopted both Model Context Protocol (MCP) and Agent-to-Agent (A2A) communication, facilitating agents to function independently with contextual access to existing tools such as datasets, scraping utilities, analysis modules, etc. With LangChain for orchestration, the system is capable of executing multiple sequential financial tasks in a coordinated, multi-step manner and explainably. Future enhancements could involve frameworks such as LangGraph, facilitating memory-aware, event-driven agent-based workflows which could help adaptability and autonomy further. The evolution sets up the system as not only a financial co-pilot, but also a framework for architecting intelligent, domain-agnostic agents. The agent could be deployed, for example, as an intelligent healthcare diagnostic instrument, a learner-agnostic personalized learning assistant, or while performing real-time audit trail generation—as just a few use cases. Its growth speaks to doing more than just financial fulfillment as an AI decision making system. [3][20].

REFERENCES

[1] N. R. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development," Autonomous Agents and Multi-Agent Systems, vol. 1, no. 1, pp. 7–38, 1998.

[2] M. Wooldridge, An Introduction to MultiAgent Systems, 2nd ed., Wiley, 2009.

[3] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed., Pearson, 2021.

[4] O. Biran and C. Cotton, "Explanation and justification in machine learning: A survey," IJCAI Workshop on Explainable AI, 2017.

[5] R. Zhang, Y. Zhang, and H. Qi, "Data quality in financial data analytics," Journal of Financial Data Science, vol. 2, no. 4, pp. 50–63, 2020.

[6] a2aproject, "A2A: Agent to Agent Protocol specification and SDKs," [Online]. Available: https://github.com/a2aproject/A2A, 2025.

[7] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," Autonomous Robots, vol. 8, pp. 345–383, 2000.

[8] S. Yao, N. Shinn, et al., "ReAct: Synergizing reasoning and acting in language models," arXiv preprint, arXiv:2210.03629, 2022.

[9] G. Mialon, et al., "Augmented language models: A survey," arXiv preprint, arXiv:2302.07842, 2023.

[10] H. Chase, "LangChain Documentation," [Online]. Available: https://docs.langchain.com, 2023.

[11] LangChain, "LangChain GitHub Repository," [Online]. Available: https://github.com/langchain-ai/langchain, 2023.

[12] S. Das and A. Mukherjee, "Modeling stock market dynamics using multi-agent systems," Proc. 38th Winter Simulation Conference, pp. 1541–1547, 2006.

[13] R. J. Shiller, Narrative Economics: How Stories Go Viral and Drive Major Economic Events, Princeton University Press, 2017.

[14] H. Chen, P. De, Y. J. Hu, and B. H. Hwang, "Wisdom of crowds: The value of stock opinions transmitted through social media," The Review of Financial Studies, vol. 27, no. 5, pp. 1367–1403, 2014.

[15] A. Mishra and L. Dey, "Financial news analytics using sentiment analysis," Procedia Computer Science, vol. 132, pp. 385–395, 2018.

[16] yfinance, "Yahoo Finance Python Library," [Online]. Available: https://pypi.org/project/yfinance/, 2024.

[17] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint, arXiv:1810.04805, 2019.

[18] D. Araci, "FinBERT: Financial sentiment analysis with pre-trained language models," arXiv preprint, arXiv:2006.08097, 2019.

[19] OpenAI, "OpenAI API Documentation," [Online]. Available: https://platform.openai.com/docs, 2024.

[20] N. Bostrom, Superintelligence: Paths, Dangers, Strategies, Oxford University Press, 2014.

[21] Model Context Protocol, "Model Context Protocol specification," [Online]. Available:

https://github.com/modelcontextprotocol/modelcontextprotocol, 2025.

[22] C. Jeong, "A study on the MCP × A2A framework for enhancing interoperability of LLM-based autonomous agents," arXiv preprint, arXiv:2506.01804, 2025.

[23] Y. Yu, H. Li, Z. Chen, Y. Jiang, Y. Li, D. Zhang, R. Liu, J. W. Suchow, and K. Khashanah, "FinMem: A performance enhanced LLM trading agent with layered memory and character design," arXiv preprint, arXiv:2311.13743, 2023.

[24] A. Sahu, V. Garg, and T. Nagpal, "Pages and Popcorn: An adaptive recommender system for enhancing book and movie discovery," International Journal on Science and Technology (IJSAT), vol. 16, no. 3, 2025.

[25] M. Wawer and J. A. Chudziak, "Integrating traditional technical analysis with AI: A multi-agent LLM-based approach to stock market forecasting," arXiv preprint, arXiv:2506.16813, 2025.

[26] A. E. Jeong, A. Singh, and G. K. Gupta, "A survey of agent interoperability protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP)," arXiv preprint, arXiv:2505.02279, 2025.