HASHIRU: Hierarchical Agent System for Hybrid Intelligent Resource Utilization

Kunal Pai
UC Davis
kunpai@ucdavis.edu

Parth Shah
Independent Researcher
helloparthshah@gmail.com

Harshil Patel
UC Davis
hpppatel@ucdavis.edu

Saisha Shetty
UC Davis
spshetty@ucdavis.edu

I. Introduction

The landscape of Artificial Intelligence (AI) is being reshaped by the rapid advancements in Large Language Models (LLMs), which exhibit profound capabilities in language understanding, generation, reasoning, and planning [3], [8], [25]. This progress has catalyzed the development of sophisticated AI agents capable of autonomous task execution. Increasingly, the focus is shifting from single-agent systems to Multi-Agent Systems (MAS), where collaborative teams of specialized agents address complex problems beyond the scope of individual agents [9], [32]. Such collaborative approaches hold significant potential in diverse domains like scientific discovery [2], software engineering [24], data analysis, and strategic decision-making [30]. The growing complexity of tasks, highlighted by benchmarks demanding advanced mathematical reasoning (e.g., GSM8K [6], SVAMP [22]), coding (e.g., HumanEval [4], CoDocBench [19]), and graduate-level technical knowledge and reasoning [23], further underscores the need for agentic systems capable of effectively coordinating diverse cognitive resources [31].

Despite this promise, contemporary agentic frameworks often encounter significant limitations. Many suffer from **rigidity**, relying on predefined agent roles and static team structures that hinder adaptation to dynamic task requirements [36]. Furthermore, **resource obliviousness** is prevalent; systems frequently lack mechanisms to monitor and optimize computational resources like API costs, memory usage, and CPU load, leading to inefficiency, particularly when scaling or deploying in resource-constrained environments [21]. This is often exacerbated by a reliance on powerful, proprietary cloud-based LLMs, incurring substantial operational expenses. **Model homogeneity**, the default use of a single powerful LLM for all sub-tasks, neglects the potential efficiency gains from employing a diverse ecosystem of models, including smaller, specialized, or locally-run alternatives [37]. Lastly, while **tool use** is fundamental [20], [35], the ability for agents to autonomously **create and integrate new tools** during operation remains limited, restricting dynamic functional extension and long-term self-improvement without human intervention [28].

To address these challenges, we introduce **HASHIRU (Hierarchical Agent System for Hybrid Intelligent Resource Utilization)**, a novel MAS framework designed for enhanced flexibility, resource efficiency, and adaptability. HASHIRU employs a hierarchical structure led by a central "CEO" agent that dynamically manages a team of specialized "employee" agents, instantiated on demand for specific sub-tasks. A core tenet of HASHIRU is its **hybrid intelligence** approach, strategically prioritizing smaller (e.g., 3B–7B parameter), locally-run LLMs, often accessed via frameworks like Ollama [16], to promote cost-effectiveness and computational efficiency. While prioritizing local resources, the system retains the flexibility to integrate external APIs and potentially more powerful models when justified by task complexity and resource availability, under the CEO's management.

The primary contributions of this work are:

1) A novel MAS architecture combining a **hierarchical control structure** with **dynamic, resource-aware agent lifecycle management** (hiring/firing). This management is explicitly governed by computational budget constraints (cost, memory usage, concurrency) and incorporates an economic model with hiring/firing costs to discourage excessive churn.
2) A **hybrid intelligence model** that prioritizes cost-effective, local LLMs while adaptively incorporating external APIs and potentially larger models, optimizing the efficiency-capability trade-off.
3) An integrated mechanism enabling the **autonomous creation of API tools**, allowing the system to dynamically extend its functional repertoire in response to task demands.
4) The application of an **economic model** (hiring/firing fees) to agent management, promoting efficient resource allocation and team stability.

This paper details the design and rationale behind HASHIRU. Section II discusses related work in agent architectures, dynamic management, resource allocation, model heterogeneity, and tool use. Section 3 elaborates on the HASHIRU architecture and its core mechanisms. Section 4 presents experimental results (or outlines planned experiments), followed by discussion and conclusion in Sections 5 and 6.

II. Background and Related Work

The concept of intelligent agents has evolved significantly from early work in symbolic AI and distributed problem-solving [26], [27] to the current era dominated by LLMs.

Modern agentic frameworks leverage LLMs as their cognitive core, enabling sophisticated reasoning, planning, and interaction capabilities [29], [34]. HASHIRU builds upon this foundation while addressing specific limitations observed in the current state-of-the-art.

A. Agent Architectures: Hierarchy and Dynamics

MAS architectures vary widely, including flat, federated, and hierarchical structures [9], [12]. Hierarchical models offer clear control flow and efficient task decomposition but risk bottlenecks and rigidity [10], [11]. HASHIRU utilizes a **CEO-Employee hierarchy** for centralized coordination but distinguishes itself through **dynamic team composition**. Unlike systems with static hierarchies or predefined roles (e.g., CrewAI [7], ChatDev [24]), HASHIRU's CEO actively manages the employee pool based on runtime needs and resource constraints.

B. Dynamic Agent Lifecycle Management

The ability of an MAS to adapt its composition dynamically is crucial for complex environments [15]. Various triggers for agent creation or deletion have been explored, often tied to task structure or environmental changes. HASHIRU introduces a specific mechanism where the CEO agent makes **hiring and firing decisions** based on a cost-benefit analysis considering agent performance, operational costs (API fees, inferred compute), memory footprint (tracked explicitly as a percentage of available resources), and agent concurrency limits. Furthermore, HASHIRU incorporates an **economic model** with explicit "starting bonus" (hiring) and "invocation" (usage) costs. This introduces economic friction, aiming to prevent excessive agent initialization or usage for marginal gains and promote team stability, a nuance often missing in simpler dynamic composition strategies.

C. Resource Management and Agent Economies

Resource awareness is critical for scalable and deployable MAS. Research areas in economics explore mechanisms like market-based auctions or contract nets for resource allocation [5]. HASHIRU implements a more **centralized, budget-constrained resource management model**. The CEO operates within defined limits for financial cost, memory usage (as a percentage of total available memory), and concurrent agent count. This direct management, particularly the focus on memory percentage, suggests an orientation towards practical deployment, potentially on local hardware or edge devices with finite resources, contrasting with cloud-centric systems assuming elastic resources [21]. Frameworks like AutoGen [33] and LangGraph [13] typically rely on implicit cost tracking via API keys without such explicit multi-dimensional resource budgeting and control.

D. Hybrid Intelligence and Heterogeneous Models

Leveraging diverse LLMs with varying capabilities, costs, and latencies is an emerging trend [37]. Techniques like model routing aim to select the optimal model for specific sub-tasks. HASHIRU embraces **model heterogeneity** with a specific strategic focus: **prioritizing smaller (3B–7B), locally-run models via Ollama integration** [16]. This emphasizes cost-efficiency, low latency for simpler tasks, and potential privacy advantages over systems defaulting to large, proprietary cloud APIs (e.g., GPT-4 [18], Claude 3 [1]). While capable of integrating external APIs (potentially invoking larger models), HASHIRU's default operational stance represents a distinct balance point in the capability vs. efficiency trade-off.

E. Tool Use and Autonomous Tool Creation

The ability to use external tools (APIs, functions, databases) is a cornerstone of modern agents, enabled by frameworks like ReAct [35] and built-in function calling [17]. Most systems rely on a predefined tool suite. HASHIRU advances this by incorporating a mechanism for **autonomous API tool creation**. When a required functionality is missing, the CEO can commission the generation (potentially via a specialized agent or code generation process) and deployment of a new API tool within the HASHIRU ecosystem. This capability for self-extension differentiates HASHIRU from systems limited to static toolsets and moves towards greater operational autonomy and adaptability [21], [28].

In summary, HASHIRU integrates concepts from hierarchical control, dynamic MAS, resource management, and tool use, but its novelty lies in the synergistic combination of: (1) dynamic, resource-aware hierarchical management with (2) an economic model for stability, (3) a local-first hybrid intelligence strategy, and (4) integrated autonomous tool creation capabilities. This combination targets key limitations in current agentic systems concerning efficiency, adaptability, cost, and autonomy.

III. HASHIRU System Architecture

The architecture of HASHIRU is designed to directly address the challenges of rigidity, resource obliviousness, and limited adaptability outlined in Section I. It implements a hierarchical, dynamically managed multi-agent system optimized for hybrid resource utilization. This section details the core components and mechanisms underpinning HASHIRU's operation.

A. Overview

HASHIRU operates on a hierarchical model conceptually similar to a business organization, featuring a central coordinating agent ("CEO") and specialized task-executing agents ("Employees"). Key architectural tenets include:

- **Centralized Coordination within a Dynamic Hierarchy:** A CEO agent manages overall strategy, task allocation, and team composition.
- **Dynamic Lifecycle Management:** Employee agents are instantiated (hired) and terminated (fired) based on runtime requirements and resource constraints, governed by an economic model.
- **Hybrid Intelligence:** Strategic preference for local, computationally cheaper LLMs, while retaining access to external APIs and potentially more powerful models.

- **Explicit Resource Management:** Continuous monitoring and control of costs, memory usage, and agent concurrency against defined budgets.
- **Adaptive Tooling:** Utilization of predefined tools alongside the capability for autonomous creation of new API tools.

Figure 1 illustrates the overall structure and interaction flow.

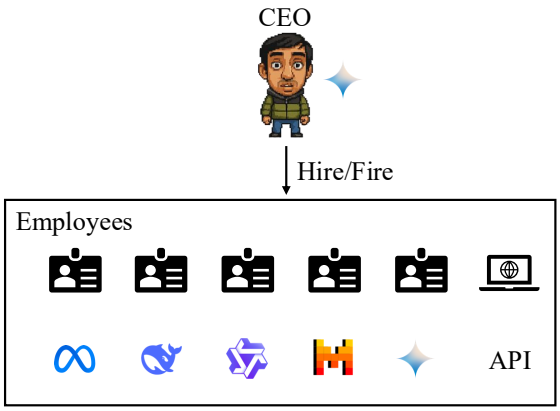

Fig. 1. High-level architecture of the HASHIRU system, illustrating the CEO-Employee hierarchy.

B. Hierarchical Structure: CEO and Employee Agents

The system employs a two-tiered hierarchy:

- **CEO Agent:** This singleton agent serves as the central coordinator and entry point. Its primary responsibilities include:
  - Receiving and interpreting the primary user query or task.
  - Decomposing the main task into smaller, manageable sub-tasks.
  - Identifying the capabilities required for each sub-task.
  - Managing the pool of Employee agents (see Section III-C).
  - Assigning sub-tasks to suitable, active Employee agents.
  - Monitoring the progress and performance of Employee agents.
  - Synthesizing the results from Employee agents into a coherent final output or response.
  - Managing the system's overall resource budget (see Section III-E).
  - Initiating the creation of new tools when required (see Section III-F).
- **Employee Agents:** These are specialized agents instantiated by the CEO to perform specific sub-tasks. Each Employee agent typically wraps an LLM (local via Ollama [16] or external via API) or provides access to a specific tool/API. Key characteristics include:
  - Specialization: Possessing capabilities tailored to certain types of sub-tasks (e.g., code generation, data analysis, information retrieval).
  - Dynamic Existence: Created and destroyed by the CEO based on need and performance.
  - Task Execution: Receives a sub-task description and context from the CEO, executes it, and returns the result.
  - Resource Consumption: Associated with specific resource costs (e.g., API call costs, memory footprint) tracked by the system.

This hierarchical structure facilitates organized task decomposition and result aggregation under centralized control, while the dynamic nature of the Employee pool provides flexibility.

C. Dynamic Agent Lifecycle Management

A core innovation in HASHIRU is the CEO's ability to dynamically manage the Employee agent team through "hiring" (instantiation) and "firing" (termination). This process is driven by a cost-benefit analysis aimed at optimizing task performance within resource constraints.

When a new sub-task requires capabilities not readily available or efficiently provided by the current pool of active Employee agents, the CEO may decide to hire a new agent. Conversely, if an agent is underperforming, consistently idle, excessively costly, or if resource limits are approached, the CEO may decide to fire it. This decision considers multiple factors:

- **Task Requirements:** The specific capabilities needed for pending sub-tasks.
- **Agent Performance Metrics:** Historical success rate, quality of output, or efficiency of existing agents relevant to the task type.
- **Operational Costs:** API costs, estimated computational load, or other costs associated with using the agent's underlying model or tools.
- **Memory Footprint:** The amount of system memory the agent consumes, tracked as a percentage of the total available memory allocated to HASHIRU.
- **Agent Concurrency:** The current number of active Employee agents relative to a predefined limit.

Crucially, HASHIRU incorporates an **economic model** for agent lifecycle events:

- **Hiring Cost ("Starting Bonus"):** A one-time cost incurred when a new agent is instantiated, representing setup overhead.
- **Invocation Cost ("Salary"):** A multi-time cost incurred when an agent is used, representing load on the system, or on payment methods, for using an agent.

These explicit transaction costs discourage excessive agent churn, promoting stability. The CEO must evaluate if the anticipated long-term benefit of replacing an agent outweighs the immediate hiring and firing costs plus any difference in ongoing operational costs. This mechanism directly combats system rigidity and allows adaptation while actively managing

computational budgets and preventing wasteful high-frequency agent turnover.

D. Hybrid Intelligence and Model Management

HASHIRU is designed for **hybrid intelligence**, leveraging a diverse set of cognitive resources. It strategically prioritizes the use of smaller (e.g., 3B–7B parameter), cost-effective LLMs that can be run locally via frameworks like Ollama [16]. This approach enhances efficiency, reduces reliance on expensive external APIs, and potentially improves privacy and latency for certain tasks.

However, the system is not restricted to local models. The CEO agent can integrate and utilize:

- **External LLM APIs:** Access to powerful proprietary models (e.g., GPT-4 [18], Claude 3 [1]) when deemed necessary for complex reasoning or specialized knowledge, subject to cost-benefit analysis.
- **External Tool APIs:** Integration with third-party software or data sources.
- **Self-Created APIs:** Tools generated by HASHIRU itself (see Section III-F).

The CEO manages this heterogeneous pool, selecting the most appropriate resource (local model, external API, tool) for a given sub-task based on perceived difficulty, required capabilities, and the current resource budget status. This allows HASHIRU to balance cost-effectiveness and computational efficiency with the need for high capability when required.

E. Resource Monitoring and Control

Explicit resource management is central to HASHIRU's design, moving beyond simple API key cost tracking. The system, coordinated by the CEO, actively monitors:

- **Financial Costs:** Accumulating costs from external API calls.
- **Memory Usage:** Tracking the memory footprint of active Employee agents, specifically as a percentage of a predefined total available memory resource.
- **Agent Concurrency:** Maintaining a count of concurrently active Employee agents.

These metrics are monitored against predefined **budget limits** or hard caps. If initiating an action (like hiring a new agent) would exceed a budget limit (e.g., push memory usage over 90% of allocated, or exceed the maximum concurrent agent count), the action is prevented. This mechanism ensures the system operates within defined operational constraints, crucial for deployment on devices with limited resources or under strict financial budgets.

F. Tool Utilization and Autonomous Creation

Like many modern agent systems, HASHIRU's agents can utilize predefined tools (functions, external APIs, databases) to interact with the environment and perform actions beyond pure text generation [17], [35].

A distinctive feature of HASHIRU is its capability for **integrated, autonomous tool creation**. If the CEO agent determines, through task analysis or failure analysis of existing agents, that a specific functional capability is required but not available through existing Employee agents or tools, it can initiate a process to create a new tool. This typically involves:

1) Defining the specification for the required tool (inputs, outputs, functionality).
2) Commissioning the generation of the necessary logic (e.g., code implementing the functionality, potentially involving API calls to external services using provided credentials, possibly generated by a specialized code-generating Employee agent).
3) Deploying this logic as a new, callable API endpoint accessible within the HASHIRU ecosystem.
4) Potentially instantiating a new Employee agent dedicated to utilizing this newly created tool.

This mechanism allows HASHIRU to dynamically extend its own functional repertoire over time, tailoring its capabilities to the tasks it encounters without requiring direct manual intervention for every new function, thereby enabling greater autonomy and long-term adaptation.

IV. Experimental Setup

To evaluate the performance, efficiency, and adaptability of HASHIRU, we designed a set of experiments targeting its core architectural features. Our evaluation focuses on assessing the benefits of dynamic resource-aware management, the hybrid intelligence model, and the autonomous tool creation capability compared to relevant baselines. Specifically, we investigate:

- The impact of dynamic agent management with economic constraints on resource utilization (cost, memory) and task performance compared to static configurations.
- The effectiveness of the hybrid (local-first) model strategy versus homogeneous (cloud-only or local-only) approaches across tasks of varying complexity.
- The system's ability to autonomously create and utilize necessary tools when faced with novel functional requirements within a task.

A. Evaluation Tasks

We selected tasks demanding complex reasoning, multi-perspective analysis, and interaction, suitable for exercising HASHIRU's hierarchical coordination and dynamic capabilities. The tasks fall into two main categories:

1) Academic Paper Review: This task evaluates HASHIRU's capacity to critically assess academic work by simulating the peer-review process. Given one or more scientific papers (e.g., in PDF format), the system must generate a review summary and ultimately recommend acceptance or rejection. This probes HASHIRU's ability to decompose evaluation criteria, delegate tasks to specialized agents (e.g., novelty assessment, methodological rigor, clarity), and manage resources effectively across long and complex documents.

2) Reasoning and Problem-Solving Tasks: To evaluate broader reasoning, knowledge retrieval, and problem-solving capabilities under different constraints, we employ a set of challenging benchmarks and puzzle-like tasks:

- **Humanity's Last Exam [23]:** A benchmark designed to test graduate-level technical knowledge and complex reasoning across multiple domains. Success requires deep understanding and sophisticated problem-solving, likely necessitating access to powerful external LLMs managed effectively within HASHIRU's hybrid framework.
- **NYT Connections [14]:** This popular puzzle requires identifying hidden semantic relationships or themes to categorize 16 words into four distinct groups. Solving this involves associative reasoning, broad world knowledge, and potentially hypothesis testing across different potential groupings, testing knowledge access and combinatorial reasoning coordination.
- **Wordle:** The daily word puzzle requires deductive reasoning to identify a five-letter word within six guesses, using feedback on correct letters and positions. This tests logical deduction, constraint satisfaction, and vocabulary knowledge. It serves as a good test case for comparing the efficiency (speed, cost, number of guesses) of local versus external models for iterative reasoning. We assume interaction via a simulated game environment.
- **Globle:** This geographic deduction game requires identifying a target country based on proximity feedback from guesses. It tests geographic knowledge retrieval, spatial reasoning, and iterative strategy refinement based on feedback (distance, direction). We assume interaction via a simulated game environment.

These diverse reasoning tasks challenge the system's ability to leverage appropriate cognitive resources (local vs. external models), potentially create simple tools, and coordinate problem-solving strategies effectively.

B. Baselines for Comparison

To quantify the benefits of HASHIRU's features, we will compare its performance against several baseline configurations:

- **Static-HASHIRU:** A version with a fixed, predefined set of Employee agents (e.g., one generalist agent per potential role identified in paper analysis), disabling dynamic hiring/firing.
- **Cloud-Only HASHIRU:** HASHIRU operating exclusively with a powerful external LLM API and online function-calling for all agents, disabling the use of local models.
- **Local-Only HASHIRU:** HASHIRU operating exclusively with smaller, local LLMs (e.g., selected models via Ollama) for all agents.
- **HASHIRU (No-Economy):** HASHIRU with dynamic hiring/firing enabled but without the explicit costs, to isolate the impact of the economic model on agent churn and stability.

C. Evaluation Metrics

We will evaluate performance using a combination of quantitative and qualitative metrics:

- **Task Success Rate / Quality:**
  - Percentage of tasks successfully completed (binary for games, potentially graded for analysis based on rubrics).
  - Quality of output for analysis tasks (human evaluation based on relevance, coherence, accuracy, completeness).
  - Accuracy for information extraction tasks.
  - Number of guesses/turns required for game tasks.
- **Resource Consumption:**
  - Total cost incurred from external API calls.
  - Peak and average memory usage (% of allocated budget).
  - Wall-clock time per task.
  - Number and type (local/external) of LLM calls.
- **System Dynamics and Adaptability:**
  - Number of Employee agents hired and fired during tasks.
  - Frequency of agent churn (hires+fires / task duration or steps).
  - Number and utility of autonomously created tools (if applicable).

REFERENCES

[1] Anthropic. The Claude 3 model family: Opus, Sonnet, Haiku. Model Card, March 2024. Accessed: 2025-05-01.

[2] Daniil A Boiko, Robert MacKnight, and Gabe Gomes. Emergent autonomous scientific research capabilities of large language models. *arXiv preprint arXiv:2304.05332*, 2023.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. OpenAI Codex paper; introduced HumanEval benchmark.

[5] Scott H. Clearwater, editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1996.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Dataset introduced: GSM8K (Grade School Math 8K).

[7] CrewAI Inc. Crewai. https://www.crewai.com/, 2025. Accessed: 2025-05-01.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[9] Ali Dorri, Salil S Kanhere, and Raja Jurdak. Multi-agent systems: A survey. *Ieee Access*, 6:28573–28593, 2018.

[10] Matthew E Gaston and Marie DesJardins. Agent-organized networks for dynamic team formation. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 230–237, 2005.

[11] Matthew E Gaston and Marie DesJardins. Agent-organized networks for multi-agent production and exchange. In *Proceedings of the 20th national conference on Artificial intelligence-Volume 1*, pages 77–82, 2005.

[12] Bryan Horling and Victor Lesser. A survey of multi-agent organizational paradigms. *The Knowledge engineering review*, 19(4):281–316, 2004.

[13] LangChain. Langgraph: A framework for agentic workflows. https://www.langchain.com/langgraph, 2024. Accessed: May 1, 2025.

[14] Angel Yahir Loredo Lopez, Tyler McDonald, and Ali Emami. Nyt-connections: A deceptively simple text classification task that stumps system-1 thinkers. *arXiv preprint arXiv:2412.01621*, 2024.

[15] Duncan McFarlane, Vladimír Marík, and Paul Valckenaers. Guest editors' introduction: Intelligent control in the manufacturing supply chain. *IEEE Intelligent Systems*, 20(1):24–26, 2005.

[16] Ollama Team. Ollama. https://ollama.com/, 2023. Accessed: 2025-05-01.

[17] OpenAI. Function calling. OpenAI API Documentation, 2023. Accessed: 2025-05-01.

[18] OpenAI. Gpt-4 technical report, 2023.

[19] Kunal Pai, Premkumar Devanbu, and Toufique Ahmed. CoDocBench: A dataset for code-documentation alignment in software maintenance. *arXiv preprint arXiv:2502.00519*, 2024.

[20] Aaron Parisi, Yao Zhao, and Noah Fiedel. Talm: Tool augmented language models, 2022.

[21] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*, UIST '23, page 1–22, New York, NY, USA, 2023. Association for Computing Machinery.

[22] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021. Introduces the SVAMP challenge dataset.

[23] Long Phan, Alice Gatti, Ziwen Han, et al. Humanity's last exam, 2025.

[24] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.

[25] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[26] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2010.

[27] Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[28] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023.

[29] 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 Ji-Rong Wen. A survey on large language model based autonomous agents, 2023.

[30] Yuning Wang, Junkai Jiang, Shangyi Li, Ruochen Li, Shaobing Xu, Jianqiang Wang, and Keqiang Li. Decision-making driven by driver intelligence and environment reasoning for high-level autonomous vehicles: a survey. *IEEE Transactions on Intelligent Transportation Systems*, 24(10):10362–10381, 2023.

[31] Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. Benchmarking complex instruction-following with multiple constraints composition, 2024.

[32] Michael Wooldridge. *An introduction to multiagent systems*. John wiley & sons, 2009.

[33] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed H. Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

[34] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Linyi Yang, Ting Ruan, Yongquan Yang, Peng Li, Yitao Chang, and Yanlin Wang. The rise and potential of large language model based agents: A survey, 2023.

[35] 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. arXiv:2210.03629.

[36] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models, 2023.

[37] Wangchunshu Zhou, Jianshu Chen, Jialong Wu, Yiheng Xu, Kexin Wang, Jintian Zhang, Yuan Gao, Zhiyong Wu, Kevin Tian, Yubo Feng, Linyi Yang, Bokai Quan, Cong Yu, Yuhang Wang, Shishen Lan, Yan Wang, Hong-Cheng Guo, Chaoyu Chen, Tianxiang Sun, Jin Xiong, Yi Lu, Peng Li, Lichao Sun, Lifan Yuan, Hang Li, and Xiangang Li. Agents: An open-source framework for large language model based autonomous agents, 2023.