Cerebrum (AIOS SDK): A Platform for Agent Development, Deployment, Distribution, and Discovery

Balaji Rama
balaji.rama@rutgers.edu

Kai Mei
kai.mei@rutgers.edu

Yongfeng Zhang
yongfeng.zhang@rutgers.edu

Abstract

Autonomous LLM-based agents have emerged as a powerful paradigm for complex task execution, yet the field lacks standardized tools for development, deployment, and distribution. We present Cerebrum, an open-source platform that addresses this gap through three key components: (1) a comprehensive SDK featuring a modular four-layer architecture for agent development, encompassing LLM, memory, storage, and tool management; (2) a community-driven Agent Hub for sharing and discovering agents, complete with version control and dependency management; (3) an interactive web interface for testing and evaluating agents. The platform's effectiveness is demonstrated through implementations of various agent architectures, including Chain of Thought (CoT), ReAct, and tool-use agents. Cerebrum advances the field by providing a unified framework that standardizes agent development while maintaining flexibility for researchers and developers to innovate and distribute their agents. The live website is at https://app.aios.foundation, the code is at https://github.com/agiresearch/Cerebrum, and video https://app.aios.foundation/video-demo.

1 Introduction

Autonomous LLM-based agents (agents for short) have emerged as a transformative paradigm in applying and advancing the capabilities of Large Language Models (LLMs) beyond text prediction to executing complex tasks through planning, reasoning, tool using, and goal-directed actions (Ge et al., 2023a; Shinn et al., 2024; Li et al., 2023; Deng et al., 2024; Mei et al., 2024). The paradigm scales to real-world issues such as web browsing (Iong et al., 2024; Deng et al., 2024), social simulation (Park et al., 2023; Pang et al., 2024), and decision-making (Hua et al., 2024; Mao et al., 2023).

Although with the fast advancement of LLM-based agent research in the recent year, there still lacks a unified platform for researchers and developers to develop, deploy, and distribute their agents, and for users to discover and use the agents. This demonstration paper introduces *Cerebrum* AgentHub, which not only provides an SDK for agent development and deployment, but also provides a web-based platform for agent developers to share their agents, and for agent users to easily use the agents both through interactive web-based UI interface and through code-based calling of pre-loaded agents through one line of code.

More specifically, *Cerebrum* is a library accompanied with a live demo dedicated to supporting a standardized way to build, run, deploy, and distribute agents and agent components. At the core of the library is a unified framework for constructing diverse agents, containing optimized implementations of popular agent methodologies such as Chain of Thought (CoT) (Wei et al., 2022) and ReAct (Yao et al., 2022), with the goal of supporting implementations of agent variants that are easy to read, extend, and deploy. Furthermore, the library supports the distribution and usage of user-created agents in a centralized agent hub.

2 Related Work

AI Agents have long been considered an important step towards generalist intelligence (Wooldridge and Jennings, 1995; Jennings et al., 1998; Bresciani et al., 2004). Acting as core coordinators, agents are envisioned as intelligent entities that can perceive their surroundings, build memories (Xu et al., 2025a; Wang et al., 2024b), devise plans, and autonomously carry out actions to fulfill tasks (Wang et al., 2024a; Deng et al., 2024; Shi et al., 2025; Xu et al., 2025b; Zhang et al., 2024a).

The emergence of LLMs has drastically expanded the potential for advancing agent technology, as demonstrated by recent breakthroughs (Liu et al., 2024; Zhang et al., 2024b). Traditional prompt-based interactions are typically static, functioning as direct input-output exchanges with limited adaptability. In contrast, LLM-driven agents

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations), pages 135–142

are designed to enable dynamic decision-making processes, granting them the ability to interpret context, generate flexible responses, and act independently (Shinn et al., 2024). This evolution allows agents to transition from handling simple, single-step tasks to becoming versatile, general-purpose problem solvers (Ge et al., 2023a).

3 Cerebrum Library Design

Cerebrum is designed to provide a standardized framework for developing LLM-based agents, addressing the growing need for systematic agent architectures in the artificial intelligence community. The library implements a modular approach that facilitates both research and production deployments while maintaining flexibility for various use cases. The library's architecture consists of two primary components: (1) a layered system for agent composition and (2) a client interface for kernel communication. This dual architecture enables both fine-grained control over agent behavior and rapid development through high-level abstractions.

3.1 Layer Architecture

Every agent in the library is fully defined by four foundational building blocks shown in the diagram in Figure 1: (a) an LLM Layer, which manages model interactions and resource allocation, (b) a Memory Layer, which handles context management and state persistence, (c) a Storage Layer, which provides durable storage capabilities, and (d) a Tool Layer, which enables structured interaction with external systems. Most agent development requirements can be addressed through the composition of these four components. These layers represent connections with the corresponding parts of the AIOS kernel (Mei et al., 2024, 2025; Ge et al., 2023b), allowing agents to run on the kernel.

3.1.1 Large Language Model Layer

The Large Language Model (LLM) layer allows agents to utilize LLM cores as their backbones. While each supported model may have unique characteristics, the LLM Layer provides a standardized interface that enables seamless switching between different providers and architectures. Cerebrum, for the most part, is able to determine smart defaults for LLM parameters such as temperature, resource constraints, etc, but also provides additional fine-grained control over these parameters.

3.1.2 Memory Layer

The Memory Layer implements sophisticated working memory management for agents, crucial for maintaining context and enabling informed decision-making. The layer provides configurable memory limits, eviction strategies, and custom policy support. Memory limits are specified in bytes, with default configurations suitable for most applications. It implements an LRU-k eviction approach, where k determines the number of items considered for removal when memory limits are reached. This enables agents to maintain relevant context while efficiently managing computational resources through configurable eviction policies.

3.1.3 Storage Layer

The Storage Layer provides persistent storage capabilities essential for long-term knowledge retention and cross-session continuity. The system supports both traditional hierarchical storage through a root directory structure and modern vector databases for efficient similarity-based retrieval. When vector databases are enabled, the system can be configured with specific embedding models, dimension parameters, and indexing strategies. This flexibility allows for optimization based on specific deployment requirements and enables sophisticated knowledge management capabilities.

3.1.4 Tool Layer

The tool layer implements a comprehensive interface that handles the complexities of tool discovery, loading, and integration with large language models. Through a standardized protocol, it manages tool initialization, parameter validation, and execution flow while maintaining proper error handling.

3.1.5 Overrides Layer

Cerebrum features an optional Overrides Layer that provides fine-grained control over AIOS Kernel parameters. While most standard deployments operate effectively with default configurations, this layer enables advanced customization (e.g., scheduler) for specialized use cases. Modifications can only be performed through carefully designed interfaces. In this way, the modifications made by the developers will not influence the system stability.

3.2 Manager Module

A key component of the library is the *manager* abstraction, which orchestrates agent and tool lifecycle operations. The manager system consists of two

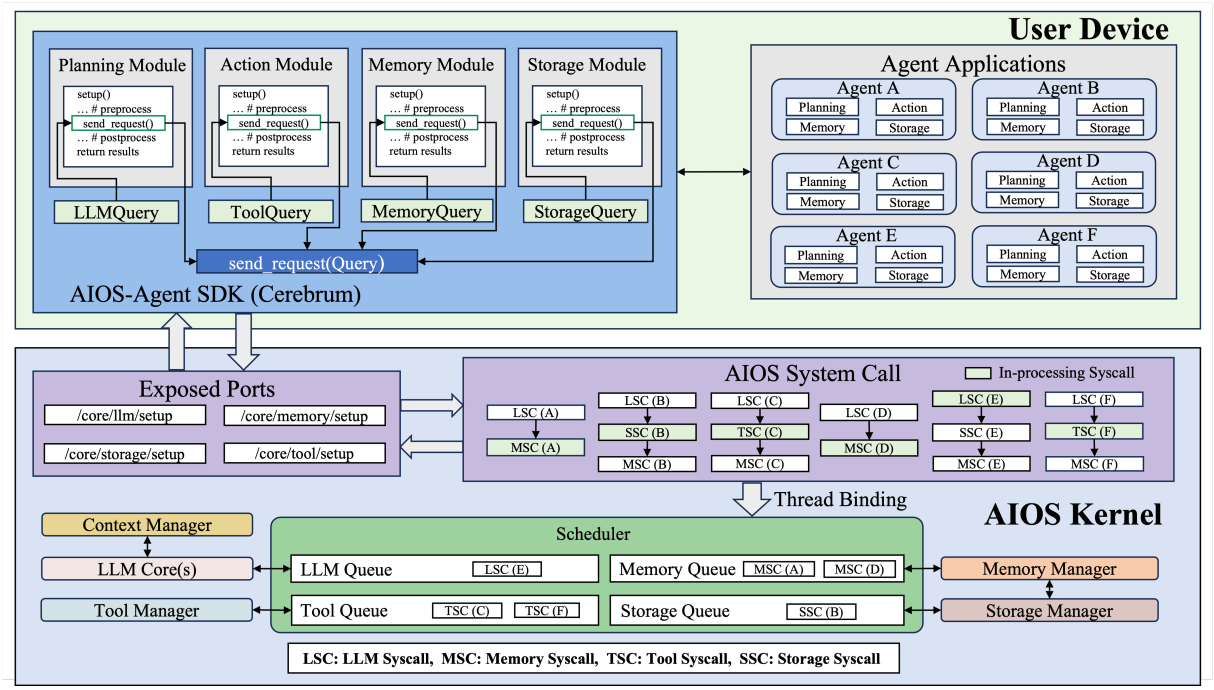

Figure 1: The architecture of Cerebrum on the basis of AIOS.

specialized components: the agent manager and the tool manager. These managers share largely identical functionality, with minor differences in their handling of their respective artifacts. Their primary responsibilities encompass distribution, versioning, caching, packaging, downloading and uploading.

3.2.1 Distribution, Version Control, Dependency Resolution

The framework maintains a centralized repository for agents, supporting versioning and dependency management. Each agent is uniquely identified by a triple of author, name, version, enabling precise version control and reproducibility of agents.

3.2.2 Caching, Packaging

To optimize system performance and minimize network overhead, the Cerebrum manager designs and implements the caching mechanism for both tools and agents. The packaging provides bundling of agents and their dependencies, allowing for reproducible deployments across different environments. And the cerebrum employs version-aware storage and retrieval strategies, enabling efficient management of different component versions while ensuring consistency across deployments.

3.2.3 Upload, Load, Download

Cerebrum supports direct uploading of packaged .agent and .tool files to their respective hubs through a streamlined interface. The download functionality is provided with built-in verification and integrity checks. The loading of agents is dynamic This loading is dynamic, allowing agents to be instantiated and used at run-time while maintaining proper isolation, which ensures operational stability and prevents unintended interactions between different agents and tools.

3.3 Client Interface and Auto-configuration

The library also features a client interface to both allow users and agents themselves to interact with the AIOS kernel, as well as the AIOS kernel to interact with agents. Users can use the client to run agents on the AIOS kernel, while AIOS uses the kernel to download, load, and run agents and tools. Additionally, we feature Auto- classes that abstract around multiple components of the Cerebrum library to allow for 1-2 line loading and deployment.

The client interface serves as the bridge between the application logic (layers + manager) and the AIOS kernel. The interface adopts a declarative configuration approach, where The client system implements a builder pattern that maintains strict initialization order dependencies while providing a fluent interface for component composition.

The client interface is augmented by a set of Auto classes that provide factory methods for reusable agent components. Similar to the Auto classes in the Transformers library, these components handle the complexities of initialization with sensible defaults and full configurability.

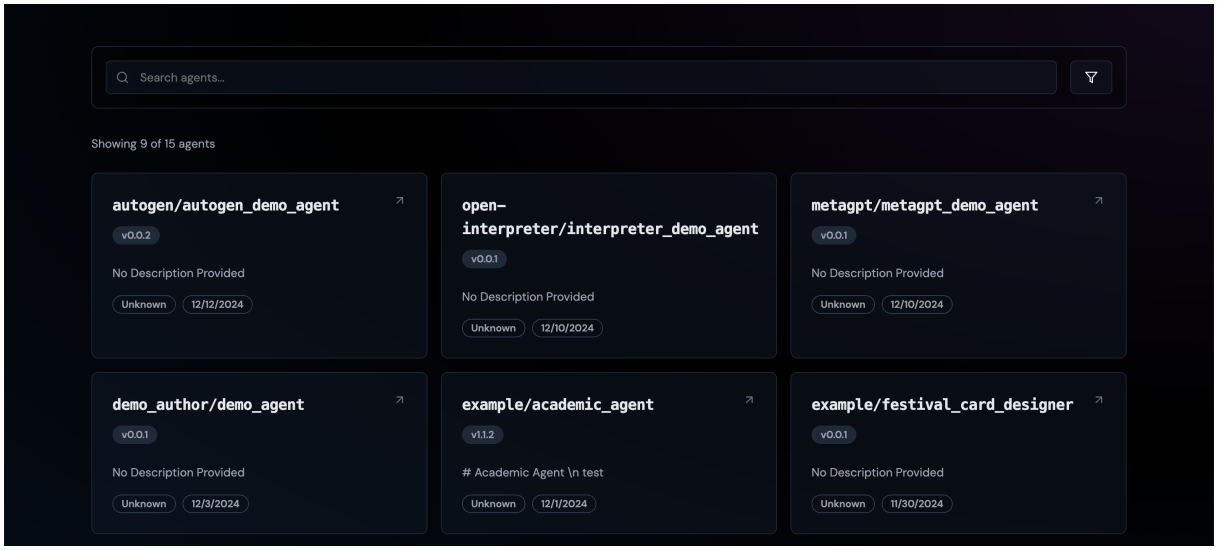

Figure 2: Cerebrum AgentHub Demo: https://app.aios.foundation/agenthub

```
# Load the agent
agent = AutoAgent.from_preloaded("
    example/academic_agent")

# Use the agent
response = agent.run({
    'task': "Your input here"
})
```

4 Agent Standard

The library provides a comprehensive agent building framework that emphasizes composition over inheritance. Unlike traditional agent frameworks that often require deep understanding of implementation details and rigid structual foundations, Cerebrum's building system enables rapid development through high-level abstractions while maintaining access to low-level controls when needed.

The framework introduces the concept of agent specifications - declarative definitions that describe an agent's capabilities, resource requirements, and behavioral patterns. These specifications can be composed, extended, and modified dynamically, enabling flexible agent architectures that can adapt to different deployment scenarios. Its implemented resource management strategies include: ① Automatic tool resolution and dependency management. ② Dynamic resource allocation based on agent specifications.

A key innovation in the building framework is its handler system, which provides extension points for customizing agent behavior without modifying core components. These handlers can intercept and modify agent operations at various stages, enabling behavior patterns while maintaining the benefits of the standard agent lifecycle.

5 Community Agent Hub

Cerebrum implements an open-source distribution platform for AI agents, following a model similar to Hugging Face's hub architecture. The Community Agent Hub serves as a central repository where researchers can freely share, discover, and utilize agents that conform to the Cerebrum agent specification. The hub itself is a hosted, publicly accessible server featuring both an overall listing of all agents and agent-specific pages.

Agents and tools are stored in an encrypted, hashed, and compressed format, containing references to their individual component files. Individual agent landing pages provide comprehensive information including: ① Version control and release history. ② Direct access to the agent's inference API endpoints via Agent Chat. ③ Licensing information and README documentation. ④ Source code accessibility for transparency and reproducibility. ⑤ Usage instructions.

A current limitation of the hub is the absence of a formal vetting process for uploaded agents. Future work may explore implementing security scanning, performance validation, and compliance checking mechanisms.

6 Community Agent Chat

To facilitate direct interaction with and evaluation of agents, we provide a public chat interface that enables real-time communication with agents hosted in the Community Hub. This interface serves as both a research tool for analyzing agent behavior and a demonstration platform for agent capabilities.

The chat system implements a mention-based in-

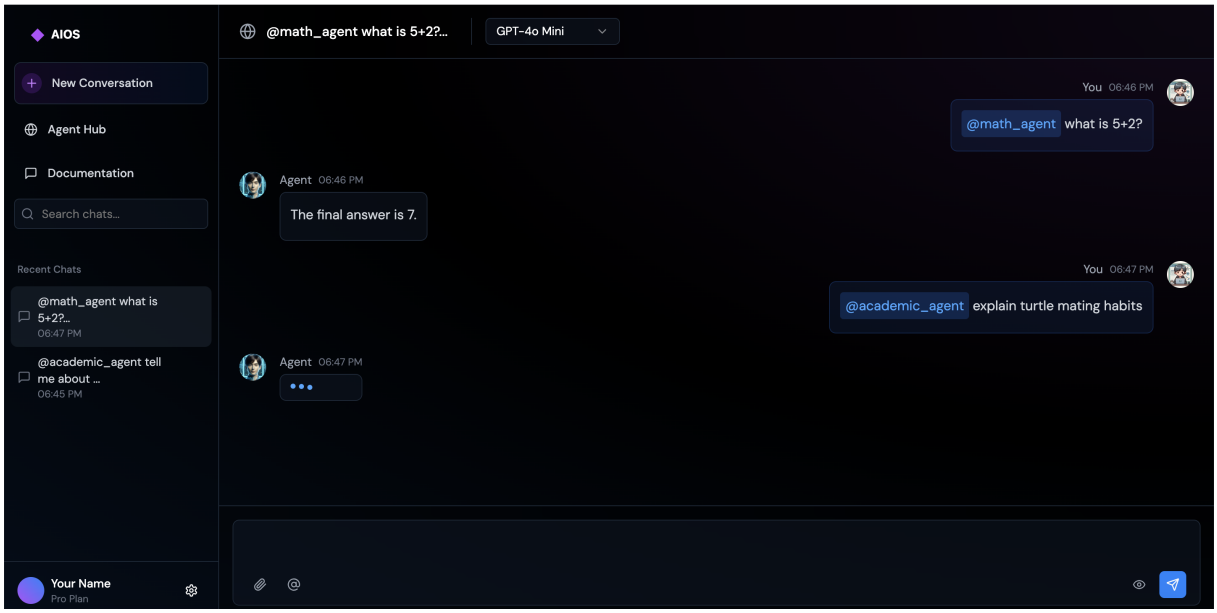

Figure 3: AgentChat Page: https://app.aios.foundation/chat

teraction model using the @ syntax (e.g. name query), where users can invoke specific agents. While the current implementation supports single agent interactions, multiple agents can be interacted with in a single conversation. Rate-limiting mechanisms are in place to ensure system stability and fair resource allocation.

Users can interact with any agents stored in the agent hub, which are served on a remote AIOS instance. The agent chat functions as a wrapper around a Cerebrum client that communicates with this remote instance. The system supports persistent, on-device chat and conversation memories, allowing users to maintain multiple different chats, which can also be deleted as needed.

7 Applications

To demonstrate the real-world applications of Cerebrum, we implemented four distinct agents that showcase different prompting techniques and capabilities: Chain of Thought (CoT), ReAct, a baseline chatbot, and a tool-augmented agent. These implementations serve to validate the flexibility and expressiveness of our agent specification framework.

7.1 Baseline Chatbot

To establish a performance baseline, we implemented a standard chatbot agent that maps input directly to output without intermediate reasoning.

$$P(y|x) = \text{LLM}(\text{prompt}(x)) \quad (1)$$

This serves as a control for evaluating the benefits of more sophisticated prompting techniques.

7.2 Chain of Thought Agent

Chain of Thought prompting (Wei et al., 2022; Wang et al., 2022; Jin et al., 2024) enables step-by-step reasoning in language models. The process can be formalized as follows:

Given input query x, the agent generates intermediate reasoning steps $s_1, \ldots, s_n$ before producing final output y:

$$P(y|x) = \sum_{s_1,\ldots,s_n} P(y|s_n)P(s_n|s_{n-1})\ldots P(s_1|x) \quad (2)$$

The prompt template is implemented as:

$$\text{prompt}(x) = \text{“Let's approach this step by step:”} + x \quad (3)$$

Each reasoning step s_i is explicitly generated and tracked, allowing for: ① Verification of logical consistency. ② Identification of failure points. ③ Analysis of reasoning patterns.

7.3 ReAct Agent

ReAct (Yao et al., 2022) combines reasoning and action in an interleaved manner. We implement this as a Markov Decision Process where:

- State space S: Current context + reasoning history
- Action space A: {Thought, Action, Observation}
- Transition function $T(s'|s, a)$: Updates state based on chosen action

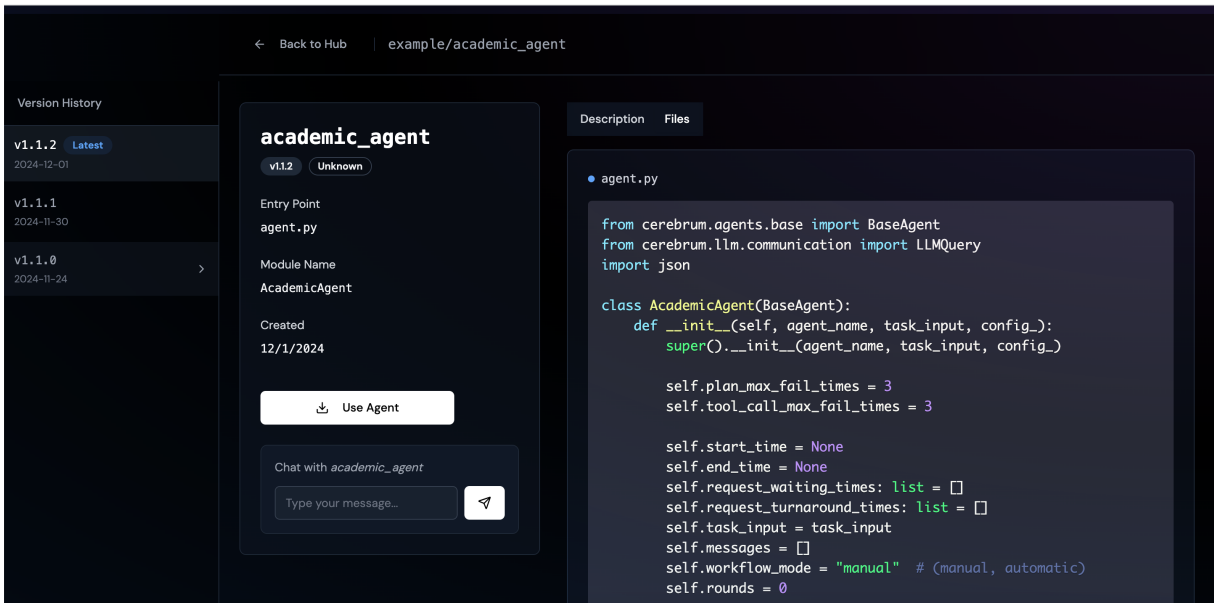

Figure 4: Agent Details: https://app.aios.foundation/agents/example/academic_agent

- Policy $\pi(a|s)$: Determines next action given current state

The agent follows the cycle:

$$\text{Thought} \xrightarrow{\text{leads to}} \text{Action} \xrightarrow{\text{generates}} \text{Observation} \xrightarrow{\text{informs}} \text{Thought} \quad (4)$$

Formally, at each step t:

$$a_t \sim \pi(\cdot|s_t) \quad (5)$$

$$s_{t+1} = T(s_t, a_t) \quad (6)$$

7.4 Tool-Augmented Agent

The tool-augmented agent demonstrates Cerebrum's external tool integration capabilities. The agent employs a hierarchical decision process:

1. Tool Selection:

$$P(\text{tool}|x) = \text{softmax}(f_{\text{select}}(x))$$

2. Tool Parameter Generation:

$$\text{params} = f_{\text{params}}(x, \text{tool})$$

3. Tool Execution:

$$\text{result} = \text{execute}(\text{tool}, \text{params})$$

4. Response Generation:

$$y = f_{\text{respond}}(x, \text{result})$$

Where f_{select}, f_{params}, and f_{respond} are learned functions implemented via prompt engineering.

8 Conclusion

We presented Cerebrum, a platform for developing, deploying, and distributing LLM-based agents. The platform addresses fundamental challenges in the agent development ecosystem through three key innovations: (1) a modular four-layer architecture that standardizes agent development while maintaining flexibility, (2) a community-driven Agent Hub that facilitates agent sharing and discovery, and (3) an interactive chat interface for direct agent evaluation and testing. Our implementations of various agent architectures, including CoT, ReAct, and tool-augmented agents, demonstrate the platform's versatility and effectiveness.

Looking forward, we envision several directions for future work. First, enhancing the Agent Hub with formal security and performance validation mechanisms would increase trust and reliability in shared agents. Second, expanding the tool layer to support more complex multi-agent interactions and collaborative scenarios could enable more sophisticated agent behaviors. Finally, developing standardized benchmarks and evaluation frameworks specifically for testing agents built with Cerebrum would help quantify and improve agent performance across different architectures and use cases.

Through its open-source nature and emphasis on standardization, Cerebrum aims to accelerate research and development in the rapidly evolving field of LLM-based agents, while fostering a collaborative ecosystem for sharing and building upon existing agent implementations.

References

Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. 2004. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8:203–236.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Yingqiang Ge, Wenyue Hua, Kai Mei, Jianchao Ji, Juntao Tan, Shuyuan Xu, Zelong Li, and Yongfeng Zhang. 2023a. OpenAGI: When LLM Meets Domain Experts. *Advances in Neural Information Processing Systems*, 36.

Yingqiang Ge, Yujie Ren, Wenyue Hua, Shuyuan Xu, Juntao Tan, and Yongfeng Zhang. 2023b. LLM as OS, Agents as APPs: Envisioning AIOS, Agents and the AIOS-Agent Ecosystem. *arXiv preprint arXiv:2312.03815*.

Wenyue Hua, Ollie Liu, Lingyao Li, Alfonso Amayuelas, Julie Chen, Lucas Jiang, Mingyu Jin, Lizhou Fan, Fei Sun, William Wang, et al. 2024. Game-theoretic llm: Agent workflow for negotiation games. *arXiv preprint arXiv:2411.05990*.

Iat Long Iong, Xiao Liu, Yuxuan Chen, Hanyu Lai, Shuntian Yao, Pengbo Shen, Hao Yu, Yuxiao Dong, and Jie Tang. 2024. Openwebagent: An open toolkit to enable web agents on large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 72–81.

Nicholas R Jennings, Katia Sycara, and Michael Wooldridge. 1998. A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1:7–38.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. *arXiv preprint arXiv:2401.04925*.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for mind exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Liangwei Yang, Zuxin Liu, Juntao Tan, Prafulla K Choubey, Tian Lan, Jason Wu, Huan Wang, et al. 2024. Agentlite: A lightweight library for building and advancing task-oriented llm agent system. *arXiv preprint arXiv:2402.15538*.

Shaoguang Mao, Yuzhe Cai, Yan Xia, Wenshan Wu, Xun Wang, Fengyi Wang, Tao Ge, and Furu Wei. 2023. Alympics: Language agents meet game theory. *arXiv preprint arXiv:2311.03220*.

Kai Mei, Wujiang Xu, Shuhang Lin, and Yongfeng Zhang. 2025. Eccos: Efficient capability and cost coordinated scheduling for multi-llm serving. *arXiv:2502.20576*.

Kai Mei, Xi Zhu, Wujiang Xu, Wenyue Hua, Mingyu Jin, Zelong Li, Shuyuan Xu, Ruosong Ye, Yingqiang Ge, and Yongfeng Zhang. 2024. Aios: Llm agent operating system. *arXiv e-prints*.

Xianghe Pang, Shuo Tang, Rui Ye, Yuxin Xiong, Bolun Zhang, Yanfeng Wang, and Siheng Chen. 2024. Self-alignment of large language models via multi-agent social simulation. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Zeru Shi, Kai Mei, Mingyu Jin, Yongye Su, Chaoji Zuo, Wenyue Hua, Wujiang Xu, Yujie Ren, Zirui Liu, Mengnan Du, Dong Deng, and Yongfeng Zhang. 2025. From commands to prompts: LLM-based semantic file system for aios. In *The Thirteenth International Conference on Learning Representations*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024b. Agent workflow memory. *arXiv preprint arXiv:2409.07429*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Michael Wooldridge and Nicholas R Jennings. 1995. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025a. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

Wujiang Xu, Yunxiao Shi, Zujie Liang, Xuying Ning, Kai Mei, Kun Wang, Xi Zhu, Min Xu, and Yongfeng Zhang. 2025b. Instructagent: Building user controllable recommender via llm agent. *arXiv preprint arXiv:2502.14662*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. 2024a. Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents. *arXiv preprint arXiv:2410.02644*.

Jianguo Zhang, Tian Lan, Rithesh Murthy, Zhiwei Liu, Weiran Yao, Juntao Tan, Thai Hoang, Liangwei Yang, Yihao Feng, Zuxin Liu, et al. 2024b. Agentohana: Design unified data and training pipeline for effective agent learning. *arXiv preprint arXiv:2402.15506*.