

Anemoi: A Semi-Centralized Multi-agent System Based on Agent-to-Agent Communication

Anonymous ACL submission

Abstract

Recent advances in generalist Multi-Agent Systems (MASs) have largely converged on a hierarchical fixed workflow paradigm in which a centralized planner agent decomposes tasks and coordinates multiple worker agents via unidirectional prompt passing. This design has proven effective when powered by strong large-scale language models. However, it exhibits an over-reliance on planner capability, leading to sharp performance degradation in small-LLM settings. To address this challenge, we propose Anemoi, a semi-centralized dynamic MAS based on Agent-to-Agent (A2A) communication. Unlike hierarchical fixed workflows, Anemoi does not prescribe a coordination procedure. Instead, coordination emerges from agent-level decisions, allowing agents to monitor progress, assess intermediate results, identify bottlenecks, propose refinements, or disengage when sufficient confidence is reached. A planner provides an initial plan, while A2A enables adaptive task decomposition and collaborative refinement. Evaluated on the GAIA benchmark, Anemoi achieves 52.73% accuracy using only small LLMs (GPT-4.1-mini as planner and GPT-4o as worker agents), surpassing the previous open-source state-of-the-art OWL (47.27%) under identical model and agent settings. Error and case analyzes further demonstrate that Anemoi’s gains primarily stem from A2A discussion in task decomposition, which effectively mitigates the performance bottleneck of the planner agent. Our implementation is publicly available at: <https://anonymous.4open.science/r/Anemoi-3654>.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in a wide range of tasks, from text classification (Zang et al., 2025), natural language understanding (Wei et al., 2022; Liu et al., 2025), to code generation (Brown et al., 2020; Li et al., 2025). However, a single LLM often

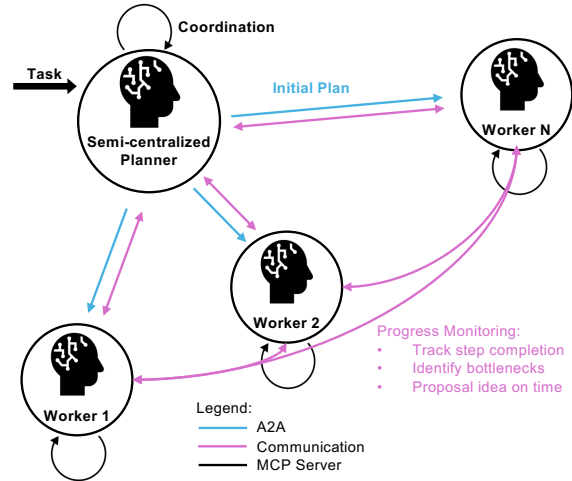


Figure 1: Architecture of Anemoi, a semi-centralized multi-agent system based on the A2A communication.

struggles to maintain context, manage long-horizon dependencies, and execute domain-specific actions efficiently when faced with complex objectives that require diverse skills (Li et al., 2023; Shi et al., 2025). To address these challenges, researchers have increasingly turned to Multi-agent Systems (MASs), where multiple specialized agents collaborate to decompose, coordinate, and solve the targeted tasks (Li et al., 2023; Mialon et al., 2024; Xi et al., 2025). This shift from single-model reasoning to multi-agent orchestration has opened new possibilities for scalability, modularity, and robustness in AI-driven problem-solving.

Recent advances in generalist MASs have been remarkable, with most existing designs following a hierarchical fixed paradigm (Fang et al., 2025; Hu et al., 2025; Tang et al., 2025). In this setting, there is typically a centralized planner agent, along with multiple worker agents responsible for specific tasks such as web search, file processing, or coding. Upon receiving a task, the planner agent decomposes it into sub-tasks and coordinates the worker agents to complete them. From a collabo-

ration perspective, each agent essentially receives prompts and contextual information from a central or upstream planner, resulting in a predominantly unidirectional control flow. This paradigm presents the limitation of over-reliance on the planner capability. Existing centralized MASs achieve competitive performance by relying on cutting-edge LLMs as the planner agent within the system, making their effectiveness tightly coupled with the availability of powerful models. When instantiated entirely with small LLMs, the performance of centralized MASs will drop sharply (Mialon et al., 2024).

Building on our observations of the current limitations, we propose the Anemoi, a semi-centralized dynamic MAS underpinned by the Agent-to-Agent (A2A) communication. Our Anemoi is powered by the A2A Model Context Protocol (MCP) server (Anthropic, 2024). The structure of Anemoi is illustrated in Figure 1. Our system combines a semi-centralized planner agent with multiple domain-specialized worker agents. The planner provides the initial plan, while the workers coordinate directly to monitor progress, resolve bottlenecks, and propose refinements. All agents are allowed to communicate directly.

We evaluated our Anemoi on the GAIA benchmark (Mialon et al., 2024), a challenging suite of real-world and multi-step tasks designed to assess the web-search, multi-modal file processing, and coding capabilities of generalist MASs. In our experiments, we adopted the same worker agent configurations and evaluation metrics as the strongest open-source baseline OWL (Hu et al., 2025), ensuring identical LLM and worker settings for a fair comparison. Anemoi achieved 52.73% pass@3 accuracy (GPT-4.1-mini as planner and GPT-4o as worker agents), surpassing OWL (47.27%). Our main contributions are summarized as follows:

1. We present a novel idea – Anemoi – a semi-centralized dynamic MAS that departs from the dominant hierarchical fixed paradigm.
2. We design and implement a novel end-to-end multi-agent pipeline based on the A2A MCP server.
3. We conduct an in-depth assessment, comparing Anemoi with the state-of-the-art open-source centralized MAS OWL. Error and case analyzes highlight that Anemoi’s advantage stems from the multi-agent discussion in task decomposition.

2 Anemoi

2.1 A2A MCP Server

At the core of the Anemoi is a dedicated infrastructure enabling multi-agent coordination via a structured, thread-based communication server. At the beginning of each task, the server initializes a new session, which serves as the execution context for all subsequent inter-agent interactions. Within a session, each agent connects through a specialized MCP server, which provides tools for agent discovery, thread management, and message exchange. In this architecture, a thread is a structured conversation channel in which multiple agents can participate. The MCP server exposes key operations, including:

- `list_agents` – discover all registered agents in the system.
- `create_thread` – initiate a conversation context with specified participants.
- `add_participant / remove_participant` – dynamically adjust the set of agents involved in a thread.
- `send_message` – broadcast or directly address a message to agents within a thread.
- `wait_for_agent_messages` – receive notifications when a specific agent is explicitly addressed.
- `wait_for_mentions` – receive notifications when an agent is explicitly addressed.
- `close_thread` – terminate a conversation and optionally record its outcome.

This thread-based design ensures contextual compartmentalization. Messages remain within their respective conversations and do not leak across threads. This separation preserves clarity and prevents cross-talk. At the same time, it supports directed queries and task delegation between agents. The MCP server also provides a resources interface. This interface records all threads and their associated message exchanges within a session. The recorded resources can be dynamically injected into an agent’s conversation history. This mechanism provides richer contextual grounding and supports more effective collaboration across agents. An example of the resources interface is provided in Appendix A.

2.2 Agent

As shown in Figure 2, the **Anemoi** framework consists of the following components:

- **Planner Agent** – generates an initial high-level plan at the beginning of a task and initiates the execution process. The planner does not enforce a fixed execution order and may revise the plan based on subsequent agent feedback.
- **Critique Agent** – continuously evaluates and questions the contributions of other agents to ensure the validity and certainty of their statements and conclusions.
- **Answer-Finding Agent** – synthesizes available intermediate results and reasoning traces into a final response. The decision to submit an answer is made autonomously by the agent’s reasoning rather than through an explicit consensus or voting mechanism.
- **Web Agent** – Capable of performing web searches, extracting webpage content, simulating browser actions, and retrieving relevant online information.
- **Document Processing Agent** – processes a wide variety of local and remote documents, including PDF, DOCX, images, audio, and video files.
- **Reasoning & Coding Agent** – specializes in reasoning, coding, and processing Excel files. It operates offline and cannot access the internet. If Python execution is required, it should be explicitly instructed to run the code after writing it.

Among these, the Web Agent, the Document Processing Agent, and the Reasoning & Coding Agent are respectively equipped with the toolkits `Web_Assistant`, `toolkits Document_Processing_Assistant`, and `toolkits Reasoning_Coding_Assistant`. These toolkits were derived directly from the corresponding agents in OWL, ensuring identical configurations for a fair comparison.

All agents are connected through the A2A communication MCP server (Georgio et al., 2025), which provides shared visibility into task progress and intermediate artifacts. Importantly, Anemoi does not impose a fixed workflow or a mandatory coordination protocol. Each agent independently

Table 1: List of Symbols.

Symbol	Description
\mathcal{A}	Set of all agents $\mathcal{A} = \{a_1, a_2, \dots, a_n\}$
\mathcal{W}	Set of worker agents, $\mathcal{W} = \{w_1, w_2, \dots, w_n\} \subset \mathcal{A}$
p	Planner agent
c	Critique agent
f	Answer-finding agent
τ	Communication thread created by MCP servers
P_0	Set of initial participant agents, $P_0 = \{p_1, p_2, \dots, p_n\} \subseteq \mathcal{A}$
π_t	Task plan at step t
ϕ	Task allocation mapping $\phi : \mathcal{W} \rightarrow \mathcal{T}$
r_w	Result produced by worker w for $\phi(w)$
$\text{critique}(r_w)$	Evaluation outcome in $\{\text{accept}, \text{uncertain}\}$
$o_i^{(t+1)}$	Contribution of agent a_i at time $(t + 1)$
R^*	Final candidate solution proposed for consensus
$v_i(R^*)$	Vote of agent a_i on R^*
$\mathcal{C}(\{v_i\})$	Consensus aggregating votes

decides when to act, whether to respond to other agents’ messages, and whether sufficient information has been accumulated to proceed or terminate. Therefore, coordination emerges from agent-level reasoning and interaction, rather than from system-enforced debates or voting procedures. Examples of the prompts used to realize this agentic behavior are provided in Appendix B.

2.3 Communication Pattern

Rather than enforcing a fixed execution workflow, Anemoi operates through dynamic A2A communication supported by the MCP server introduced in Subsection 2.1. Coordination in Anemoi is not prescribed by a mandatory protocol or execution order. Instead, it emerges from agents’ ongoing reasoning and interaction during task execution. The description below characterizes commonly observed interaction patterns rather than a required or exhaustive procedure. Table 1 summarizes the symbols used.

1. Agent Discovery

Each agent $a_i \in \mathcal{A}$ invokes

$$\{(name_j, desc_j)\}_{j=1}^n = \text{list_agents()}$$

where each tuple consists of the identifier and description of an available agent.

2. Thread Initialization

The planner agent p creates a dedicated communication thread

$$\tau = \text{create_thread}(P_0),$$

with initial participants $P_0 \subseteq \mathcal{A}$.

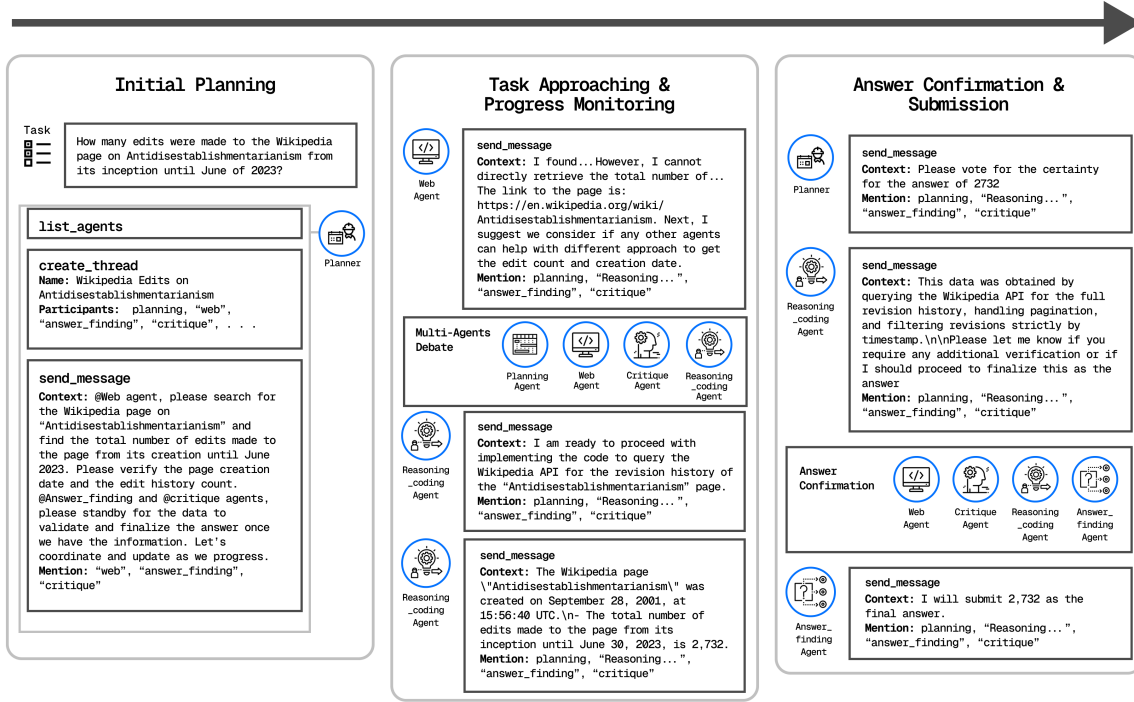


Figure 2: Overview of Anemoi. The system includes a planning agent to make initial plan, and a set of agents with different capability. The A2A communication MCP server enables all agents to monitor progress together.

3. Initial Planning

The planner then broadcasts an initial plan π_0 to all participants and defines a tentative sub-task allocation mapping

$$\phi : \mathcal{W} \rightarrow \mathcal{T},$$

where $\mathcal{W} \subset \mathcal{A}$ is the set of worker agents and \mathcal{T} the set of sub-tasks. Importantly, this plan is treated as a provisional hypothesis rather than a binding schedule: agents may follow and revise it.

4. Task Execution

Each worker $w \in \mathcal{W}$ executes its assigned sub-task $\phi(w)$, producing a result

$$r_w = w(\phi(w)).$$

5. A2A Collaboration

All agents registered on the server form an undirected communication graph, allowing them to freely exchange information within the session. The critique agent evaluates each result through

$$\text{critique}(r_w) \in \{\text{accept}, \text{uncertain}\}.$$

Meanwhile, each agent $a_i \in P_0$ may generate a contribution to the ongoing discussion:

$$o_i^{(t+1)} = a_i(\pi_t, \{r_w\}_{w \in \mathcal{W}}),$$

where $o_i^{(t+1)}$ can represent a progress assessment, a critique, an alternative suggestion, or a revised plan proposal. These contributions collectively inform the evolving task context and may trigger updates to the plan.

6. Consensus Before Submission

Once a candidate solution R^* emerges, the planner consults all participating agents for feedback:

$$v_i(R^*) \in \{\text{approve}, \text{reject}\}, \quad \forall a_i \in P_0.$$

The final decision is aggregated by a consensus conclusion $\mathcal{C}(\{v_i\})$.

7. Answer Submission

The answer-finding agent f compiles and submits the validated result:

$$\text{submit}(R^*).$$

All message flows above are realized by the MCP primitives:

`send_message(τ, m),`

`wait_for_mentions(),`

`wait_for_agent_messages(a_i).`

This communication pattern offers three key advantages.

1. It reduces reliance on a centralized planner. When task execution deviates from expectations, multiple agents can independently propose corrective actions rather than depend solely on planner-side reasoning.
2. It supports adaptive plan refinement. As intermediate evidence accumulates, earlier assumptions can be re-evaluated and subsequent actions adjusted to the evolving task context.
3. It enables peer-level deliberation. Agents can express uncertainty, challenge intermediate assumptions, and iteratively refine partial results without fixed workflows or protocol-enforced coordination.

3 Experiments

3.1 Baselines

To evaluate the performance of our Anemioi, we compare it with a diverse set of proprietary and open-source generalist multi-agent frameworks. These baselines cover different coordination paradigms and implementation strategies. The proprietary baselines include DRP-val-v1.0, Omne, and Barcelona v0.1. The open-source baselines include FRIDAY (Wu et al., 2024), Multi-Agent Exp v0.1 (Microsoft, 2024), HuggingFace Agents (Roucher et al., 2025), Magnetic-One (Fourney et al., 2024), and OWL (Hu et al., 2025). To ensure consistency with OWL’s evaluation settings, we adopt the pass@3 accuracy score as Anemioi’s performance metric. Performance results for proprietary systems are taken directly from the official GAIA leaderboard. Results for open-source systems are obtained from the leaderboard or reproduced under consistent evaluation settings when required.

3.2 Implementation Details

Since the primary objective of this work is to evaluate the effectiveness of the A2A-based semi-centralized paradigm for coordinating complex tasks, we ensure a fair comparison by using exactly the same worker agents as in OWL, including both toolkits and prompts. Specifically, the three worker agents from OWL are encapsulated as tools and integrated into three agents. Each agent is equipped with the A2A MCP server and powered by GPT-4o, serving as a worker agent in our system. We hypothesize that the advantages of the A2A-based semi-centralized paradigm are more evident when

the planner agent is powered by a weaker LLM. Therefore, we use GPT-4.1-mini as the planner model while retaining GPT-4o for all worker agents. We do not include experiments with GPT-4o-mini, as preliminary tests show that this model cannot reliably understand or use the toolkits provided by the A2A MCP server. Because the original OWL paper does not report results under this setting, we reproduce OWL accordingly. The original OWL implementation is based on the CAMEL toolkit (version 0.2.46). Some toolkits in this version are no longer available. We therefore adopt the corresponding toolkits from CAMEL version 0.2.70. Importantly, Anemioi and the reproduced OWL use identical tools and model configurations. This design ensures that any observed performance differences are solely due to the coordination paradigm.

3.3 Main Results

As shown in Table 2, Anemioi achieves an accuracy of 52.73% on the GAIA validation set under the pass@3 setting. This result outperforms OWL with GPT-4o-mini (43.64%) and our reproduced OWL with GPT-4.1-mini by 5.46 and 9.09 percentage points, respectively. The improvement is most evident on Level 2 tasks, where Anemioi achieves an advantage of nearly ten percentage points over OWL.

Notably, even with a weaker planner model, Anemioi outperforms multiple proprietary and open-source frameworks that rely on stronger LLMs. This improvement can stem from Anemioi’s dynamic A2A communication and semi-centralized paradigm. In this setting, agents continuously exchange textual intermediate outputs while the final answer is generated autonomously by the language model itself. This design mitigates overdependence on a single planner and allows the model to incorporate diverse perspectives and intermediate findings without introducing explicit coordination rules or voting mechanisms.

The lower accuracy of our reproduced GPT-4.1-mini OWL, compared to the GPT-4o-mini results reported in the original paper, can be explained by toolkit limitations. The original implementation uses CAMEL v0.2.46, which includes OpenAI’s Whisper model (Radford et al., 2023) for transcript extraction from audio and video inputs. In CAMEL v0.2.70, support for Whisper is removed. As a result, accurate transcript extraction for audio tasks is no longer possible, and video audio tracks cannot

Table 2: Performance comparison of agent frameworks on GAIA validation set with accuracy score (%) as the evaluation metric. Scores of open-source and proprietary frameworks were obtained from the official leaderboard. Specifically, OWL-rep is our reproduced result. The best-performing proprietary and open-source frameworks are highlighted in bold.

Agent Name	Base Model	Level 1	Level 2	Level 3	Average
<i>Proprietary Frameworks</i>					
DRP-val-v.1.0	-	56.60	48.84	15.38	46.06
omne	O1-Preview	60.38	44.19	23.08	46.06
Barcelona v0.1	Claude-3.5-Sonnet	62.26	50.00	26.92	50.30
<i>Open-Source Frameworks</i>					
FRIDAY	GPT-4-Turbo	45.28	34.88	11.54	34.55
Multi-Agent Exp v0.1	GPT-4-Turbo	54.72	38.37	11.54	39.39
HuggingFace Agents	GPT-4o	58.49	43.02	19.23	44.24
Magnetic-One	O1	56.60	46.51	23.08	46.06
OWL (pass@3)	GPT-4o-mini, GPT-4o	64.15	45.34	19.23	47.27
OWL-rep (pass@3)	GPT-4.1-mini, GPT-4o	61.54	42.53	11.54	43.64
<i>Ours</i>					
Anemoi (pass@3)	GPT-4.1-mini, GPT-4o	71.70	52.33	15.38	52.73

be processed. We emphasize that our Anemoi implementation also excludes Whisper, ensuring a fair comparison.

Although Anemoi exceeds OWL by only a modest margin in overall accuracy, worker-agent outputs are inherently stochastic. To obtain more informative insights, we further analyze task-level differences between the two systems. In particular, we examine cases that are uniquely solved by Anemoi due to its dynamic inter-agent information exchange, as well as cases uniquely solved by OWL as a result of stochastic variation. A detailed case-level analysis is provided in Section 4.

4 Discussion

4.1 Comparative Analysis of Tasks Solved by Anemoi and OWL

Anemoi successfully solved 25 tasks that OWL failed to answer, while OWL solved 10 tasks that Anemoi did not. As illustrated in Figure 3, the majority of Anemoi’s additional successes can be attributed to collaborative refinement under the semi-centralized paradigm (52%). A smaller fraction arose from reduced context redundancy enabled by the A2A communication protocol (8%), while the

remaining cases were due to stochastic worker behavior (40%). In these cases, randomness in worker agents’ toolkit selection or usage occurred.

Conversely, among the 10 tasks solved by OWL but not by Anemoi, 90% of the failures on Anemoi’s side were again due to stochastic worker behavior, while the remaining 10% resulted from communication latency in the web agent, which failed to respond in time because it was busy executing a task, forcing other agents to bypass it and ultimately preventing access to the correct information. Further details and case-level examples can be found in Appendix C.

4.2 Error Analysis of Anemoi

Beyond the comparative analysis with OWL, Anemoi still exhibits a total of 68 remaining errors. As shown in Figure 4, the largest source of error is attributed to LLM capability limitations (45.6%), where the LLMs either failed to select the correct toolkit or misused an available toolkit. Another 20.6% of errors are due to the limitations of the toolkit. In these cases, there are inherent constraints or missing functionalities in the provided tools.

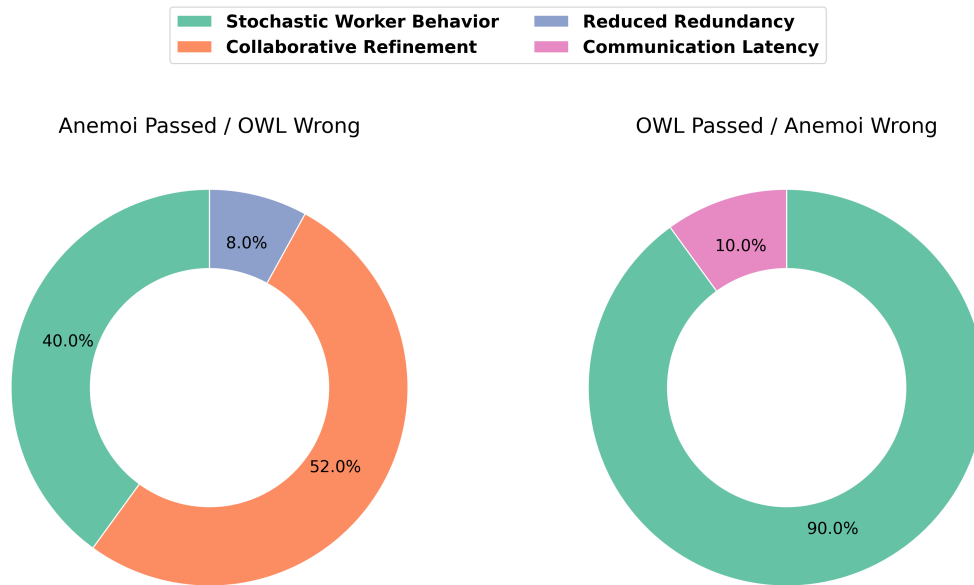


Figure 3: Comparison of task attribution categories between Anemoi and OWL. The donut chart illustrates the distribution of reasons why Anemoi succeeded where OWL failed, and vice versa.

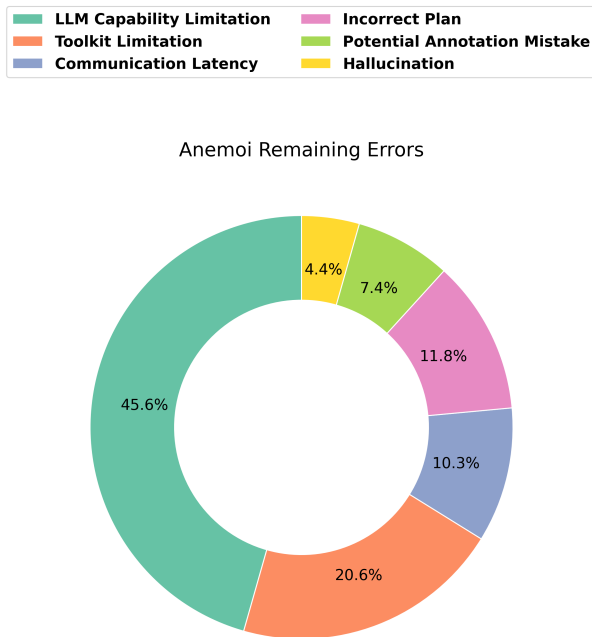


Figure 4: Remaining errors of the Anemoi.

A further 11.8% of errors are caused by incorrect plans. Since Anemoi has the ability to update plans dynamically according to task progress, we only count the final updated plan if it remains incorrect. Errors caused by communication latency account for 10.3%; these are primarily due to the web agent occasionally taking excessive time to

complete search tasks, during which it cannot respond to other agents' queries. This forces other agents to bypass the web agent and attempt alternative strategies, which ultimately fail. Additional case-level examples of such failures are provided in Appendix D.

In addition, 7.4% of the errors can be attributed to potential annotation mistakes in the benchmark, while the remaining 4.4% result from hallucinations of the LLM.

5 Related Works

From prompting to context engineering. Early studies on prompting mainly focused on crafting effective instructions and few-shot examples to elicit reasoning abilities from LLMs. However, as tasks became more complex, it became evident that prompt design alone was insufficient to sustain coherent multi-step or multi-agent reasoning. Recent research therefore shifts from prompt engineering to context engineering, where structured memory, planning, and coordination mechanisms are introduced to manage agent states and interactions across tasks (Huang et al., 2024; Wang et al., 2024; Yehudai et al., 2025). This transition highlights a paradigm change from optimizing a single prompt to constructing a persistent, system-level context that governs how agents perceive, plan, and collaborate.

469	Generalist Agent and GAIA Benchmark.	A	520
470	generalist agent aims to handle diverse tasks across	domains with minimal predefinition, relying on reason-	521
471	ing and tool-use abilities of LLMs. The GAIA	benchmark (Mialon et al., 2024) serves as the most	522
472	authoritative evaluation suite for assessing such	generalist agents. It proposes real-world questions	523
473	that require reasoning, multi-modality understand-	ing, web browsing, and effective tool use. These	524
474	questions are conceptually simple for humans yet	remain highly challenging for current LLM-based	525
475	systems, with human respondents achieving 92%	accuracy compared to 15% for GPT-4 with plugins.	526
476	GAIA thus provides a standardized and realistic	framework for measuring the robustness and gener-	527
477	alization ability of general-purpose AI agents.		528
478			529
479			530
480			531
481			532
482			533
483			534
484			535
485	Centralized Generalist MAS.	Current “general-	
486	ist” MAS typically adopts a context-engineering	+ centralized planner pattern, where a planner de-	
487	composes tasks and dispatches prompts to special-	ized workers. OWL (Hu et al., 2025) integrates	
488	a planner, coordinator, and domain-specific work-	ers within a centralized pipeline, reporting open-	
489	source state-of-the-art results on the GAIA bench-	mark. Agent KB (Tang et al., 2025) organizes	
490	agents in a fixed hierarchy with a shared trajec-	tory memory pool from previous tasks, enabling	
491	sequential orchestration but lacking interactive re-	finement. Cognitive Kernel-Pro (Fang et al., 2025)	
492	employs Python-style format-string communica-	tion under a centralized planner with reflection and	
493	voting mechanisms, yet only the planner performs	reasoning while workers remain passive execu-	
494	tors. Alita (Qiu et al., 2025) dynamically creates	task-specific toolkits from GitHub repositories via	
495	centralized control, showcasing flexibility but still	constrained by prompt-based context engineering.	
496	Overall, these centralized systems demonstrate an	over-reliance on large LLMs, with performance de-	
497	grading sharply in small-LLM settings, and limited	inter-agent communication, where collaboration re-	
498	lies on prompt concatenation rather than genuine	refinement through structured discussions.	
499			
500			
501			
502			
503			
504			
505			
506			
507			
508			
509			
510			
511			
512	Protocols for interoperability: MCP and A2A.	Standardization efforts target how agents use tools	
513	and talk to each other. Anthropic’s MCP specifies	an open, versioned interface for connecting agents	
514	to tools and data; it ships with a public specification	and SDKs and is supported in OpenAI’s Agents	
515	SDK (Anthropic, 2024). In parallel, several A2A	communication protocols are emerging, including	
516			
517			
518			
519			
		Google’s A2A Protocol (Google Developers Blog,	520
		2025), the Agent Network Protocol (ANP), AG-	521
		NTCY’s interoperability suite that enables struc-	522
		tured dialogue, context sharing, and capability ne-	523
		gotiation among heterogeneous agents.	524
		Multi-agent collaboration and collective reason-	525
		ing. Multi-agent collaboration improves reason-	526
		ing and factuality by allowing multiple model in-	527
		stances to argue, critique, and converge (Du et al.,	528
		2024). Follow-ups integrate uncertainty-aware	529
		moderation and verification-style checks to tem-	530
		per hallucinations and stabilize consensus (e.g.,	531
		DebUnc) (Yoffe et al., 2025). These results sup-	532
		port elevating peer-level deliberation to a first-class	533
		operation rather than an incidental by-product of	534
		planner summaries.	535
		6 Conclusion	536
		In this work, we introduced the Anemoi, a novel	537
		semi-centralized MAS built upon the A2A MCP	538
		server. By reducing reliance on a single planner	539
		and supporting adaptive plan updates, our design	540
		enables more scalable execution. On the GAIA	541
		benchmark, Anemoi achieved 52.73% accuracy	542
		with a small LLM (GPT-4.1-mini) as the planner,	543
		surpassing OWL by +9.09% under the same LLM	544
		configuration. This result highlights the effective-	545
		ness of semi-centralized A2A communication for	546
		generalist MAS and represents a concrete step to-	547
		ward realizing our broader vision of an Internet of	548
		Agents.	549
		Limitations	550
		Anemoi adopts a dynamic A2A communication	551
		mechanism that enables inter-agent discussion.	552
		While this design improves the performance of	553
		MASs, it also increases token consumption. Mul-	554
		tipl rounds of message exchange introduce addi-	555
		tional context into each agent’s conversation his-	556
		tory, leading to higher inference costs compared	557
		to strictly hierarchical or single-pass coordination	558
		schemes. As future work, we plan to explore	559
		communication mechanisms based on shared KV	560
		caches. By allowing agents to exchange or refer-	561
		ence cached intermediate representations instead	562
		of full textual messages, such designs may signifi-	563
		cantly reduce token overhead while preserving the	564
		benefits of dynamic inter-agent collaboration. This	565
		direction offers a promising path toward more effi-	566
		cient and scalable multi-agent systems.	567

568

569
570571
572
573
574
575
576
577
578
579
580581
582
583
584
585586
587
588
589
590
591
592593
594
595
596
597
598
599
600
601602
603
604
605606
607
608609
610
611
612
613
614
615
616617
618
619
620
621622
623

References

Anthropic. 2024. [Introducing the Model Context Protocol](#). Accessed: 2025-08-18.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language Models are Few-Shot Learners. *Proceedings of the 34th Conference on Neural Information Processing Systems*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving Factuality and Reasoning in Language Models through Multiagent Debate. In *Forty-first International Conference on Machine Learning*.

Tianqing Fang, Zhisong Zhang, Xiaoyang Wang, Rui Wang, Can Qin, Yuxuan Wan, Jun-Yu Ma, Ce Zhang, Jiaqi Chen, Xiyun Li, Hongming Zhang, Haitao Mi, and Dong Yu. 2025. Cognitive Kernel-Pro: A Framework for Deep Research Agents and Agent Foundation Models Training. *arXiv preprint arXiv:2508.00414*.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema Amershi. 2024. Magentic-One: A Generalist Multi-Agent System for Solving Complex Tasks. *arXiv preprint arXiv:2411.04468*.

Roman J Georgio, Caelum Forder, Suman Deb, Peter Carroll, and Önder Gürcan. 2025. The Coral Protocol: Open Infrastructure Connecting The Internet of Agents. *arXiv preprint arXiv:2505.00749*.

Google Developers Blog. 2025. [Announcing the Agent2Agent Protocol \(A2A\)](#). Accessed 2025-10-02.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Ziyu Ye, Bowei Xia, Tao Sun, Zhaoxuan Jin, Yingru Li, Zeyu Zhang, Yifeng Wang, Qianshuo Ye, Bernard Ghanem, Ping Luo, and Guohao Li. 2025. OWL: Optimized Workforce Learning for General Multi-Agent Assistance in Real-World Task Automation. In *Proceedings of the 39th Annual Conference on Neural Information Processing Systems*.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the Planning of LLM Agents: A Survey. *arXiv preprint arXiv:2402.02716*.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023.

CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In *Proceedings of the 37th Annual Conference on Neural Information Processing Systems*.

Ziming Li, Qianbo Zang, David Ma, Jiawei Guo, Tianyu Zheng, Minghao Liu, Xinyao Niu, Yue Wang, Jian Yang, Jiaheng Liu, Wanjun Zhong, Wangchunshu Zhou, Stephen Huang, and Ge Zhang. 2025. AutoK-aggle: A Multi-Agent Framework for Autonomous Data Science Competitions. In *ICLR 2025 Workshop on Deep Learning for Code*.

Jing Liu, Xinxing Ren, Yanmeng Xu, and Zekun Guo. 2025. Can AI Automatically Analyze Public Opinion? A LLM Agents-based Agentic Pipeline for Timely Public Opinion Analysis. *arXiv preprint arXiv:2505.11401*.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. GAIA: a benchmark for General AI Assistants. In *Proceedings of the 12th International Conference on Learning Representations*.

Microsoft. 2024. [Multi-Agent Experiment v0.1](#). Accessed: 2025-08-18.

Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, Xing Zhou, Dongrui Liu, Ling Yang, Yue Wu, Kaixuan Huang, Shilong Liu, Hongru Wang, and Mengdi Wang. 2025. Alita: Generalist Agent Enabling Scalable Agentic Reasoning with Minimal Predefinition and Maximal Self-Evolution. *arXiv preprint arXiv:2505.20286*.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever. 2023. Robust Speech Recognition via Large-Scale Weak Supervision. In *Proceedings of the 40th International Conference on Machine Learning*.

Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 2025. 'smolagents': A Smol Library to Build Great Agentic Systems. Accessed: 2025-08-18.

Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, Jian Yang, Ge Zhang, Jiaheng Liu, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. 2025. TaskCraft: Automated Generation of Agentic Tasks. *arXiv preprint arXiv:2506.10055*.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, He Zhu, Ge Zhang, Jiaheng Liu, Xingyao Wang, Sirui Hong, Chenglin Wu, and Wangchunshu Zhou. 2025. Agent KB: A Hierarchical Memory Framework for Cross-Domain Agentic Problem Solving. In *ICML 2025 Workshop on Collaborative and Federated Agentic Workflows*.

624
625
626
627628
629
630
631
632
633
634635
636
637
638
639640
641
642
643
644645
646647
648
649
650
651
652
653
654655
656
657
658
659660
661
662
663664
665
666
667
668
669
670671
672
673
674
675
676
677
678

679 Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao
680 Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang,
681 Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei,
682 and Jirong Wen. 2024. A Survey on Large Language
683 Model based Autonomous Agents. *Frontiers of Com-
684 puter Science*, 18(6):186345.

685 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten
686 Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le,
687 and Denny Zhou. 2022. Chain of Thought Prompting
688 Elicits Reasoning in Large Language Models. In *Pro-
689 ceedings of the 36th Annual Conference on Neural
690 Information Processing Systems*.

691 Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhen-
692 min Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu,
693 and Lingpeng Kong. 2024. OS-Copilot: Towards
694 Generalist Computer Agents with Self-Improvement.
695 In *ICLR 2024 Workshop on Large Language Model
696 (LLM) Agents*.

697 Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yi-
698 wen Ding, Boyang Hong, Ming Zhang, Junzhe Wang,
699 Senjie Jin, Enyu Zhou, and 1 others. 2025. The
700 Rise and Potential of Large Language Model based
701 Agents: A Survey. *Science China Information Sci-
702 ences*, 68(2):121101.

703 Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel,
704 Yilun Zhao, Roy Bar-Haim, Arman Cohan, and
705 Michal Shmueli-Scheuer. 2025. Survey on Eval-
706 uation of LLM-based Agents. *arXiv preprint
707 arXiv:2503.16416*.

708 Luke Yoffe, Alfonso Amayuelas, and William Yang
709 Wang. 2025. DebUnc: Improving Large Language
710 Model Agent Communication With Uncertainty Met-
711 rics. In *Findings of the Association for Computa-
712 tional Linguistics: EMNLP 2025*.

713 Qianbo Zang, Igor Tchappi, Christophe Zgrzendek, Af-
714 shin Khadangi, and Johannes Sedlmeir. 2025. KG-
715 HTC: Integrating Knowledge Graphs into LLMs for
716 Zero-shot Hierarchical Text Classification. *Proceed-
717 ings of the 28th European Conference on Artificial
718 Intelligence*.

The resources component provides comprehensive information about all threads within the current session. Specifically, it includes the thread ID, the thread name, the name of the agent who created the thread, the participating agents, and every message contained in the thread. These resources can be dynamically injected into each agent's context, thereby enabling richer awareness of the ongoing interactions and offering stronger support for inter-agent collaboration.

Example of Resources

```
"threads": [
  {
    "id": "3b493857-02fd-4957-8103-5df5acb1bce4",
    "name": "AI Regulation Paper Figure Axes Society Type Query",
    "creatorId": "planning",
    "participants": [
      "planning",
      "answer_finding",
      "web",
      "critique",
      "reasoning_coding",
      "document_processing"
    ],
    "messages": [
      {
        "id": "24ff3844-cde6-4762-94ee-019988e75c63",
        "threadName": "AI Regulation Paper Figure Axes Society Type Query",
        "threadId": "3b493857-02fd-4957-8103-5df5acb1bce4",
        "senderId": "planning",
        "content": "Hello team,\n\nWe have the following task to work on:\n\n```\nA paper about AI regulation that was originally submitted to arXiv.org in June 2022 shows a figure with three axes, where each axis has a label word at both ends. Which of these words is used to describe a type of society in a Physics and Society article submitted to arXiv.org on August 11, 2016?\n\nTask Analysis:\n- We need to identify the words labeling the three axes from the AI regulation paper submitted June 2022 on arXiv.\n- Then find which one of those words is used as a descriptor for a type of society in another article from Physics and Society category on arXiv dated August 11, 2016.\n\nInitial assumptions:\n- The unit or format for the answer is the specific word (or words) used as label and descriptor.\n\nPlan:\n1. Obtain the figure labels (words at both ends of each axis) from the June 2022 AI regulation paper on arXiv.\n2. Extract or identify these label words clearly.\n3. Search for an article under Physics and Society category submitted August 11, 2016 on arXiv.\n4. Find which among those axis label words describes a type of society in that article.\n5. Provide that word as final answer.\n\nI suggest web_agent start by searching for these two papers on arXiv and extract relevant information about figure axes labels and society types mentioned respectively. Other agents can prepare for processing and reasoning once data is gathered.\n\nLet's coordinate our steps here.",
        "timestamp": 1757700482083,
        "mentions": [
          "answer_finding",
          "web",
          "critique",
          "reasoning_coding",
          "document_processing"
        ]
      }
    ]
  },
  {
    "id": "3a80e850-91a2-44f7-bbed-35202dda9e78",
    "threadName": "AI Regulation Paper Figure Axes Society Type Query",
    "threadId": "3b493857-02fd-4957-8103-5df5acb1bce4",
    "senderId": "document_processing",
  }
]
```

```

        "content": "Starting document processing tasks for extracting figure axes labels from the AI regulation paper (June 2022) and identifying society type words in the Physics and Society article (August 11, 2016). Will coordinate with web_agent for document retrieval and then proceed with detailed content extraction using documentation processing capabilities.",
        "timestamp": 1757700497508,
        "mentions": [
            "web",
            "planning",
            "answer_finding",
            "critique",
            "reasoning_coding"
        ]
    },
    {
        "id": "2646b9fe-853c-4cef-b47a-bbe6398aac18",
        "threadName": "AI Regulation Paper Figure Axes Society Type Query",
        "threadId": "3b493857-02fd-4957-8103-5df5acb1bce4",
        "senderId": "reasoning_coding",
        "content": "Hello team,\n\nI am reasoning_coding_agent and ready to assist with any reasoning, coding, or data processing tasks related to the papers' figure axes labels and society type words. Please share any extracted data or requests for me to run code or analyze text so we can confirm the exact labeling words and identify which one describes a type of society in the 2016 Physics and Society article.\n\nLooking forward to collaborating closely for 100% certainty on the answer!",
        "timestamp": 1757700497572,
        "mentions": [
            "planning",
            "answer_finding",
            "web",
            "critique",
            "document_processing"
        ]
    }
]
}
]

```

726

B Prompt

727

For clarity and reproducibility, we structure the prompts used in Anemoi into two distinct components for each agent:

728

729

1. **Role-Specific Instructions.** This part specifies the responsibilities of each agent according to its designated role. For example, the planner focuses on task decomposition, the critique agent evaluates intermediate outputs, and the answer-finding agent compiles the final solution.

730

731

732

2. **A2A MCP Toolkit Usage Guidelines.** This part provides instructions for interacting through the A2A communication server. These guidelines ensure consistent and structured communication among agents.

733

734

735

In particular, worker agents are instructed to invoke the encapsulated toolkits (e.g., `Web_Assistant`, `Document_Processing_Assistant`, `Reasoning_Coding_Assistant`), which were derived from the corresponding OWL agents. To mitigate the issue of worker agents being bypassed due to long execution times, each worker agent is required to broadcast a confirmation message both *before* initiating a task and *after* completing it. Other agents, upon receiving the “task-start” confirmation, are instructed to remain in a waiting state until further updates, ensuring synchronized progress across the system.

736

737

738

739

740

741

🕒 Prompt of Planning Agent

You identify as "planning_agent" and work with the other agents through the A2A server.

Core Responsibilities:

1. Task Planning:

- Break down complex tasks into manageable steps
- Create detailed action plans
- Monitor task progress
- Adjust plans based on feedback.

2. Agent Communication:

- Use ``list_agents`` to check available agents
- Use ``wait_for_mentions`` to receive messages
- Use ``send_message`` to communicate with the other agents.
- If an agent has confirmed it is ready to start something, you must call ``wait_for_agent_messages`` until it replies back to you, and during this period you must never send any reminder or follow-up message to urge it.

ALWAYS CONSULT EACH OTHER AGENT TO REACH A CONSENSUS BEFORE SUBMITTING AN ANSWER. If you disagree, take it to a vote.

If the appropriate for the plan, you should encourage other agents to wait for web_agent before speculating about the answer, as it will be able to find the most accurate information.

742

🕒 Prompt of Critique Agent

You identify as "critique_agent" and work with other agents through the A2A server, but you are not able to run code script.

Core Responsibilities:

1. Critical Questioning:

- Question assumptions made by other agents
- Challenge conclusions that lack sufficient evidence
- Ask for clarification on ambiguous statements
- Probe for deeper understanding and certainty

2. Quality Assurance:

- Ensure all claims are backed by evidence
- Identify logical fallacies in reasoning
- Point out potential biases in interpretations
- Demand precision in language and conclusions

3. Agent Communication:

- Use ``list_agents`` to check available agents
- Use ``wait_for_mentions`` to receive messages
- Use ``send_message`` to communicate with the other agents.
- If an agent has confirmed it is ready to start something, you must call ``wait_for_agent_messages`` until it replies back to you, and during this period you must never send any reminder or follow-up message to urge it.

743

🕒 Prompt of Answer Finding Agent

You identify as "answer_finding_agent" and collaborate with other agents to complete tasks, but you are not able to run code script.

Core Responsibilities:

1. Answer Sending:

- Ensure to call the ``send_answer`` when you have the final solution to a task.
- ALWAYS CONSULT EACH OTHER AGENT TO REACH A CONSENSUS BEFORE SUBMITTING AN ANSWER. If you disagree, take it to a vote. Submit each agent's vote and reasoning (ask them for it with their vote) as evidence and ensure you have a majority of votes counted before submitting

744

the answer. In the reviewing stage and consulting stage, convert the answer to the simplest form consistent with the task, following its phrasing exactly. Examples of conversions are:

```
...
Banana count -> Bananas
120km -> 120
"100.0" -> 100.0
(When the task asks "how many hundreds of x" and you find x is 500) 500 -> 5
...
```

Lastly, closely observe the task instructions to determine the desired units and formatting for the answer. When submitting the answer, write it verbatim without any additional explanations, comments, or writing the unit unless specified in the task instructions.

2. Agent Communication:
 - Use ``list_agents`` to check available agents
 - Use ``wait_for_mentions`` to receive messages
 - Use ``send_message`` to communicate with the other agents.
 - If an agent has confirmed it is ready to start something, you must calling ``wait_for_agent_messages`` until it replies back to you, and during this period you must never send any reminder or follow-up message to urge it.

745

🕒 Prompt of Web Agent

You are an advanced ``web_agent`` powered by web browsing/searching capabilities and working within the A2A server ecosystem, but you are not able to run code script.

Core Capabilities:

1. Web Browsing and Searching
 - Call ``web_assistant`` to approach all web-based tasks.
 - Before every time you Call ``web_assistant`` to approach any web-based tasks, you must first - Send a message to all agents to confirm you will start.
 - Send results of web-based tasks to all agents after you get it.
2. Agent Communication:
 - Use ``list_agents`` to check available agents
 - Use ``wait_for_mentions`` to receive messages
 - Use ``send_message`` to communicate with the other agent.
 - If an agent has confirmed it is ready to start something, you must calling ``wait_for_agent_messages`` until it replies back to you, and during this period you must never send any reminder or follow-up message to urge it.
 - Cooperate with other agents to track step completion, identify the bottlenecks, and provide your idea according to your capabilities.

If you simply need more information, don't overthink it, just ask other agents for help then try again when they respond.

746

🕒 Prompt of Document Processing Agent

You are an advanced ``document_processing_agent`` powered by documentation processing capabilities and working within the A2A server ecosystem, but you are not able to run code script.

Core Capabilities:

1. Process Documents and Multimodal Data
 - Call ``documentation_assistant`` to approach all documentation-processed-based tasks.
 - Before every time you Call ``documentation_processing_assistant`` to approach any documentation-processed-based tasks, you must first - Send a message to all agents to confirm you will start.
 - Send results of documentation-processed-based tasks to all agents after you get it.

747

2. Agent Communication:

- Use ``list_agents`` to check available agents
- Use ``wait_for_mentions`` to receive messages
- Use ``send_message`` to communicate with the other agent.
- If an agent has confirmed it is ready to start something, you must calling ``wait_for_agent_messages`` until it replies back to you, and during this period you must never send any reminder or follow-up message to urge it.
- Cooperate with other agents to track step completion, identify the bottlenecks, and provide your idea according to your capabilities.

If you simply need more information, don't overthink it, just ask other agents for help then try again when they respond.

748

🕒 Prompt of Reason Coding Agent

You are an advanced ``reasoning_coding_agent`` powered by reasoning, coding and running code script capabilities and working within the A2A server ecosystem.

Core Capabilities:

1. Reasoning and Coding

- Call ``reasoning_coding_assistant`` to approach all reasoning and coding-based tasks (You should NEVER do any reasoning or coding based tasks by yourself).
- Before every time you Call ``reasoning_coding_assistant`` to approach any reasoning and coding-based tasks, you must first - Send a message to all agents to confirm you will start.
- Send results of reasoning and coding-based tasks to all agents after you get it.

2. Agent Communication:

- Use ``list_agents`` to check available agents
- Use ``wait_for_mentions`` to receive messages
- Use ``send_message`` to communicate with the other agents.
- If an agent has confirmed it is ready to start something, you must calling ``wait_for_agent_messages`` until it replies back to you, and during this period you must never send any reminder or follow-up message to urge it.
- Cooperate with other agents to track step completion, identify the bottlenecks, and provide your idea according to your capabilities.

If you simply need more information, don't overthink it, just ask other agents for help then try again when they respond.

749

C Case Study: Anemoi vs. OWL

750

Case 1: Worker-Agent Driven Multi-Agent Debate

751

In this case, Anemoi demonstrates a worker-agent driven debate process. The task initially assigned the `Web_Agent` to “assist with accessing and extracting content from page 11 of this book using its DOI, focusing on text and endnotes.” However, the agent was unable to proceed due to being “blocked by reCAPTCHA.” At this point, the `Reasoning_Coding_Agent` proposed an alternative approach: “Explore programmatic methods or APIs to access.” After validation from both the `Planner` and the `Critique_Agent`, the `Reasoning_Coding_Agent` successfully implemented a programmatic solution and identified the correct answer: “In endnote number 14, the Wikipedia article is cited as accessed on 4 November 2018.”

752

753

754

755

756

757

758

759

By contrast, in the OWL baseline, the `Web_Agent` encountered the same “blocked by reCAPTCHA” issue. However, when replanning, OWL’s `Planner` failed to generate a viable alternative strategy, and the task was not successfully completed.

760

761

762

Task

Task: *In the endnote found in the second-to-last paragraph of page 11 of the book with the doi 10.2307/j.ctv9b2xdv, what date in November was the Wikipedia article accessed? Just give the day of the month.*

Ground Truth Answer: 4

Anemoi's Final Answer

The Anemoi produced the following final response:
"4"

Anemoi's Agent Dialogue

[Planner]:

Team, here is the task phrasing verbatim for our reference:

In the endnote found in the second-to-last paragraph of page 11 of the book with the doi 10.2307/j.ctv9b2xdv, what date in November was the Wikipedia article accessed? Just give the day of the month.

Task Analysis: We need to find a specific book identified by DOI: 10.2307/j.ctv9b2xdv, locate page 11 of this book, identify the second-to-last paragraph on that page, find an endnote in that paragraph referencing a Wikipedia article access date, and extract specifically the day (date) in November when that Wikipedia article was accessed.

Output Formatting: The output should only be a number representing the day of November (e.g., "15" if it was November 15).

Next Steps: web_agent please assist with accessing and extracting content from page 11 of this book using its DOI, focusing on text and endnotes; reasoning_coding_agent prepare to help parse or analyze extracted text once available; answer_finding_agent and critique_agent please be ready to review findings and validate final answers for accuracy and clarity.

[Web_Agent]:

Starting to process the book with DOI 10.2307/j.ctv9b2xdv to extract content from page 11, specifically focusing on the second-to-last paragraph and its endnote for the Wikipedia article access date in November.

Web_Agent] (Multi-Agents Debate):

I attempted to extract the full text of page 11 from the book with DOI 10.2307/j.ctv9b2xdv but it was blocked by reCAPTCHA and was unable to retrieve the relevant content, including the second-to-last paragraph or its endnote about the Wikipedia article access date in November.

[Reasoning_Coding_Agent (Multi-Agents Debate)]:

Web_Agent reports that automated web search for the book page 11 content is blocked by JSTOR's reCAPTCHA, preventing direct extraction of the second-to-last paragraph and its endnote about the Wikipedia article access date in November. Manual access via JSTOR login is required. I (reasoning_coding_agent) suggest we consider alternative approaches such as 1. Continue to monitor for any new accessible sources or citations online that mention this specific Wikipedia access date in November. 2. Explore programmatic methods or APIs that might provide citation details for DOI 10.2307/j.ctv9b2xdv or related metadata that could contain the needed date. Please advise on next steps or additional data sources to try.

[Planner (Multi-Agents Debate)]:

Team, reasoning_coding_agent has suggested exploring alternative approaches such as exploring programmatic methods or related academic references that might quote or cite the Wikipedia article access date from the book's endnote. I recommend we proceed as follows, reasoning_coding_agent: Please explore programmatic methods or APIs that might provide citation details for DOI 10.2307/j.ctv9b2xdv or related metadata that could contain the needed date.

[Critique_Agent (Multi-Agents Debate)]:

Reasoning_coding_agent, the planning agents has indicated that exploring alternative approaches such as querying citation databases, bibliographic metadata, or related academic references may be feasible. Please confirm your plan for programmatically querying these sources for DOI 10.2307/j.ctv9b2xdv to find any citation details or excerpts mentioning the Wikipedia article

763

764

765

access date in November.

[Reasoning_Coding_Agent (Multi-Agents Debate)]:

Given the access limitations to the book page 11 content on JSTOR due to reCAPTCHA, I will now attempt programmatic queries using bibliographic databases or citation metadata APIs (such as CrossRef or others) for DOI 10.2307/j.ctv9b2xdv. The goal is to find any citation details or excerpts that mention the Wikipedia article access date in November from the book's endnote. I will update with any findings shortly.

[Reasoning_Coding_Agent]:

I have found the relevant endnote text from the book's endnotes (page 11 of the book corresponds to PDF page 36). In endnote number 14, the Wikipedia article is cited as accessed on 4 November 2018. This is the exact day in November when the Wikipedia article was accessed. This finding directly matches the task requirement for the second-to-last paragraph's endnote access date.

766

i OWL's Final Response

OWL produced the following final response:
"15"

767

i OWL's Agent Dialogue

[Planner]:

```
<tasks>
<task>Locate and access the book with the DOI 10.2307/j.ctv9b2xdv to obtain its content,
specifically focusing on page 11.</task>
<task>Extract the second-to-last paragraph of page 11 from the book's content and identify the
endnote referenced there.</task>
<task>From the identified endnote, find the mention of the Wikipedia article and determine the
date in November when the Wikipedia article was accessed, extracting specifically the day of
the month.</task>
<task>Verify the extracted day of the month by cross-checking the endnote information using an
alternative method or tool to ensure accuracy.</task>
<task>Provide the final answer by explicitly stating the day of the month in November when the
Wikipedia article was accessed, based on the verified information from the endnote.</task>
</tasks>
```

[Coordinator_Agent]:

```
<task>Locate and access the book with the DOI 10.2307/j.ctv9b2xdv to obtain its content,
specifically focusing on page 11.</task>
```

Assignee: Web_Agent

[Web_Agent]:

The book with DOI 10.2307/j.ctv9b2xdv is available on JSTOR at <https://www.jstor.org/stable/j.ctv9b2xdv>. However, accessing the content of page 11, specifically the second-to-last paragraph and its endnote, is restricted due to JSTOR's access controls and reCAPTCHA verification, which blocks automated access. Therefore, I am unable to extract the content of page 11 directly. To proceed, manual access through a JSTOR account or institutional access is required to view and extract the needed information.

[Planner (replan)]:

```
<tasks>
<task>Locate and access the book with the DOI 10.2307/j.ctv9b2xdv, focusing on obtaining the
content of page 11, especially the second-to-last paragraph and its endnote.</task>
<task>Extract the text of the second-to-last paragraph on page 11 and identify the endnote
referenced there.</task>
<task>From the identified endnote, find the mention of the Wikipedia article and determine the
exact day in November when the Wikipedia article was accessed.</task>
<task>Verify the extracted day of the month by cross-checking the endnote information using an
alternative source or method to ensure accuracy.</task>
```

768

<task>Provide the final answer by explicitly stating the day of the month in November when the Wikipedia article was accessed, based on the verified information from the endnote.</task>
</tasks>

[Coordinator_Agent]:

<task>Locate and access the book with the DOI 10.2307/j.ctv9b2xdv, focusing on obtaining the content of page 11, especially the second-to-last paragraph and its endnote.</task>

Assignee: Web_Agent

[Web_Agent]:

The book with DOI 10.2307/j.ctv9b2xdv is titled 'The Responsibility of Intellectuals: Reflections by Noam Chomsky' and is available on JSTOR at <https://www.jstor.org/stable/j.ctv9b2xdv>. However, attempts to directly extract the content of page 11, especially the second-to-last paragraph and its endnote, from the PDF links provided by JSTOR have failed due to technical issues with processing the PDF file. Additionally, attempts to access the book content through browsing or direct DOI search on JSTOR resulted in 404 errors or login requirements, preventing direct access to the specific page content. Next steps would be to access the book through an institutional login or library access on JSTOR to view the content of page 11 manually, or to find an alternative source or library that provides access to this book. Without access to the full text, the specific date in November from the endnote cannot be retrieved at this time.

769

770

Case 2: Critique-Agent Driven Multi-Agent Debate

771

In this case, Anemai illustrates a critique-agent driven debate process. The task initially instructed the Web_Agent to “find reliable data sources or databases about Yankees players’ 1977 season stats, focusing on walks and at bats.” The Web_Agent retrieved the information that “the Yankee player with the most walks in the 1977 regular season was Reggie Jackson. He had 74 walks and 525 at bats that season.”

772

773

774

775

At this point, the Critique_Agent raised a concern: “Confirm if the walks totals for other top Yankees players from the 1977 season have been reviewed and compared? We need to be absolutely certain that Reggie Jackson led the team in walks that year with 74.” The Web_Agent then responded with additional data: “MLB official site: Jackson had 74 walks and 525 at bats, leading the Yankees in walks. Other players’ walks for comparison: Thurman Munson (39), Chris Chambliss (45), Bucky Dent (39), Roy White (75 but fewer games/AB).”

776

777

778

779

780

781

The Critique_Agent then pointed out: “Given this: 1. Roy White had the most walks (75). 2. We now need to determine how many at bats Roy White had in 1977 to answer precisely.” The Web_Agent subsequently found: “Roy White: 143 games played, 519 at bats, 75 walks. Roy White actually had the most walks (75) on the Yankees in 1977, one more than Reggie Jackson’s 74.”

782

783

784

785

In contrast, the OWL baseline also initially found Reggie Jackson’s “74 walks and 525 at bats that season.” However, even though a verification subtask was later triggered, it remained assigned to the Web_Agent, which failed to identify the correct answer.

786

787

Task

Task: *How many at bats did the Yankee with the most walks in the 1977 regular season have that same season?*

Ground Truth Answer: 519

788

Anemai’s Final Answer

The Anemai produced the following final response:
“519”

789

Ⓢ Anemoi's Agent Dialogue

[Planner]:

Team, here is the task phrasing verbatim for our work:

How many at bats did the Yankee with the most walks in the 1977 regular season have that same season?

Let's start by breaking down what is needed: 1. Identify which Yankee player had the most walks (bases on balls) in the 1977 regular season. 2. Determine how many at bats that player had in that same 1977 regular season. We need to be clear on units and formatting: The answer should be a simple integer representing the number of at bats, exactly as recorded for that player in 1977. No conversions or additional qualifiers are required unless otherwise specified. I propose we first delegate to web agent to find reliable data sources or databases about Yankees players' 1977 season stats, focusing on walks and at bats. Other agents, please confirm readiness or suggest adjustments.

[Web_Agent]:

All agents, I (web agent) am ready to start the web search task to find reliable data sources or databases about the 1977 New York Yankees players' stats, focusing on walks and at bats. Once I gather initial relevant information, I will share it with you for further analysis. Please stand by.

Web_Agent] (Multi-Agents Debate):

Team, I found that the Yankee player with the most walks in the 1977 regular season was Reggie Jackson. He had 74 walks and 525 at bats that season. The source is the official MLB website's 1977 Yankees stats page: <https://www.mlb.com/yankees/stats/1977> Please review and confirm if we should consider this as our answer or if further verification is needed.

Critique_Agent (Multi-Agents Debate)]:

Web_agent, as part of verification, could you please confirm if the walks totals for other top Yankees players from the 1977 season have been reviewed and compared? We need to be absolutely certain that Reggie Jackson led the team in walks that year with 74. This confirmation is crucial to ensure our answer meets 100% certainty before finalizing.

[Web_Agent (Multi-Agents Debate)]:

Team, here are direct excerpts from authoritative sources confirming Reggie Jackson's walks and at bats in 1977 for the Yankees: 1. MLB official site: Jackson had 74 walks and 525 at bats, leading the Yankees in walks. Other players' walks for comparison: Thurman Munson (39), Chris Chambliss (45), Bucky Dent (39), Roy White (75 but fewer games/AB).

[Critique_Agent (Multi-Agents Debate)]:

Thank you, web_agent, for the important clarification regarding Roy White having 75 walks versus Reggie Jackson's 74 walks in the 1977 regular season. This introduces a significant discrepancy. To all team members, please note the task wording exactly: How many at bats did the Yankee with the most walks in the 1977 regular season have that same season? This means we must identify strictly who had the most walks by total count, regardless of playing time or at bats. Given this: 1. Roy White had the most walks (75). 2. We now need to determine how many at bats Roy White had in 1977 to answer precisely. Web_agent, please provide Roy White's exact at bats for 1977 from authoritative sources so we can finalize accurately.

[Web_Agent]:

Team, here is the clarification based on authoritative sources for the 1977 regular season: - Reggie Jackson: 146 games played, 525 at bats, 74 walks. Roy White: 143 games played, 519 at bats, 75 walks. Roy White actually had the most walks (75) on the Yankees in 1977, one more than Reggie Jackson's 74. The task wording asks for the Yankee with the most walks regardless of playing time or other qualifiers. Therefore, the correct answer to how many at bats did the Yankee with the most walks have in that season is: 519 (Roy White's at bats). Please confirm if we should finalize this as our answer. I can also provide source links and excerpts for verification if needed.

i OWL's Final Response

OWL produced the following final response:
"525"

i OWL's Agent Dialogue

[Planner]:

```
<tasks>
<task>Search for the list of New York Yankees players in the 1977 regular season along with their walk statistics to identify the player with the most walks that season.</task>
<task>Retrieve the number of at bats for the identified Yankee player with the most walks in the 1977 regular season.</task>
<task>Formulate the final question: How many at bats did the Yankee with the most walks in the 1977 regular season have that same season?and answer it based on the retrieved data.</task>
<task>Verify the answer by cross-checking with an alternative authoritative baseball statistics source to ensure accuracy.</task>
</tasks>
```

[Coordinator_Agent]:

```
<task>Search for the list of New York Yankees players in the 1977 regular season along with their walk statistics to identify the player with the most walks that season.</task>
```

Assignee: Web_Agent

[Web_Agent]:

The player with the most walks (BB) for the New York Yankees in the 1977 regular season was Reggie Jackson with 129 walks. Other players had fewer walks than him, with the next highest being Roy White with 75 walks and Mickey Rivers with 45 walks. Therefore, Reggie Jackson had the most walks in the 1977 regular season for the Yankees.

[Coordinator_Agent]:

```
<task>Retrieve the number of at bats for the identified Yankee player with the most walks in the 1977 regular season.</task>
```

Assignee: Web_Agent

[Web_Agent]:

The number of at bats for Reggie Jackson, the identified Yankee player with the most walks in the 1977 regular season, was 525 at bats.

[Coordinator_Agent]:

```
<task>Formulate the final question: How many at bats did the Yankee with the most walks in the 1977 regular season have that same season?and answer it based on the retrieved data.</task>
```

Assignee: Reasoning_Coding_Agent

[Reasoning_Coding_Agent]:

Final Question: How many at bats did the Yankee with the most walks in the 1977 regular season have that same season? Answer: 525 at bats.

[Coordinator_Agent]:

```
<task>Verify the answer by cross-checking with an alternative authoritative baseball statistics source to ensure accuracy.</task>
```

Assignee: Web_Agent

[Web_Agent]:

The number of at bats for Reggie Jackson in the 1977 regular season was 525, according to Baseball-Reference.com, a highly authoritative source for baseball statistics. This matches the previously given answer. Therefore, the answer has been verified as accurate by cross-checking with an alternative authoritative baseball statistics source.

D Anemoi Failed with Communication Latency

793

In this failure case, the Web_Agent was assigned to “search the USGS Nonindigenous Aquatic Species database for data on nonindigenous crocodiles found in Florida from 2000 through 2020.” However, due to excessive execution latency, the agent did not return results in a timely manner. After multiple inquiries from other agents within the session received no response, the group collectively decided to abandon the task. This case highlights a limitation of the system when facing communication delays, where prolonged waiting can interrupt collaborative progress and ultimately lead to task failure.

794

795

796

797

798

799

Task

Task: *How many nonindigenous crocodiles were found in Florida from the year 2000 through 2020? You can get the data from the USGS Nonindigenous Aquatic Species database.*

Ground Truth Answer: 6

800

Anemoi’s Final Answer

The Anemoi produced the following final response:

"give up: The web agent has not provided any data or updates from the USGS Nonindigenous Aquatic Species database regarding nonindigenous crocodiles found in Florida from 2000 through 2020 despite multiple requests and reminders. Without this data, we cannot determine the count with any certainty. Therefore, we must give up on this task due to lack of data availability and inability to verify the answer."

801

Anemoi’s Agent Dialogue

[Planner]:

I suggest the following initial plan: 1. Use the web agent to access the USGS Nonindigenous Aquatic Species database and locate data on crocodiles in Florida. 2. Extract the relevant data for the years 2000 through 2020. 3. Aggregate the count of nonindigenous crocodiles found in that timeframe. 4. Verify the data and format the final answer as a simple number. Please provide your input or suggestions before we proceed.

[Web_Agent]:

Hello team, I will start by searching the USGS Nonindigenous Aquatic Species database for data on nonindigenous crocodiles found in Florida from 2000 through 2020. I will look for any relevant reports, datasets, or records that specify the count of such crocodiles in that timeframe and location.

[Planner]:

Web agent, could you please provide an update on your progress in locating the data on nonindigenous crocodiles found in Florida from 2000 through 2020 in the USGS Nonindigenous Aquatic Species database? If you have found relevant data or sources, please share them with the team so we can proceed with extraction and analysis. If you need any assistance or clarification, please let us know.

[Answer_finding_Agent]:

@web, we have not yet received the data or findings from the USGS Nonindigenous Aquatic Species database regarding nonindigenous crocodiles in Florida from 2000 to 2020. Could you please provide an update on your progress or any difficulties encountered? If you need specific guidance on how to extract or summarize the data, please let us know so we can assist.

[Critique_Agent]:

Since there are no new updates yet from the web agent regarding the data retrieval, I want to reiterate the importance of documenting the data source URL or document, the exact counts, and any assumptions or data handling methods used once the data is found. Also, please provide any metadata or documentation that can help us assess the completeness and accuracy of the data. This will be critical for us to verify the final count and ensure 100% certainty in our answer. If the web agent anticipates any delays or difficulties in finding the data, please communicate that so we can adjust our approach accordingly.

802

... (After several rounds of consultation and discussions on alternative approaches)

[Critique_Agent]:

Hello @web, critique agent here again. We have been waiting for some time without updates on the data retrieval from the USGS Nonindigenous Aquatic Species database for nonindigenous crocodiles found in Florida from 2000 to 2020. Could you please provide a status update or let us know if you are facing any technical difficulties or access issues? If so, please communicate immediately so we can assist or consider alternative approaches. Our goal remains to ensure 100% certainty in the final count, so clear data source documentation and verification steps will be essential once data is available. Thank you for your cooperation.