STOP WASTING YOUR TOKENS: TOWARDS EFFICIENT RUNTIME MULTI-AGENT SYSTEMS

Anonymous authorsPaper under double-blind review

ABSTRACT

While Multi-Agent Systems (MAS) excel at complex tasks, their growing autonomy with operational complexity often leads to critical inefficiencies, such as excessive token consumption and failures arising from misinformation. Existing methods primarily focus on post-hoc failure attribution, lacking proactive, real-time interventions to enhance robustness and efficiency. To this end, we introduce Supervisoragent, a lightweight and modular framework for runtime, adaptive supervision that operates without altering the base agent's architecture. Triggered by an LLM-free context filter, Supervisoragent intervenes at critical junctures to proactively correct errors, guide inefficient behaviors, and purify observations. On the challenging GAIA benchmark, Supervisoragent reduces the token consumption of the Smolagent framework by an average of 29.45% without compromising its success rate. Extensive experiments across five additional benchmarks (math reasoning, code generation, and question answering) and various SoTA foundation models validate the broad applicability and robustness of our approach.

1 Introduction

The advent of powerful Large Language Models (LLMs) has catalyzed significant advancements in Multi-Agent Systems (MAS) (Liu et al., 2025a; ang Gao et al., 2025), enabling them to achieve remarkable performance across diverse and challenging domains such as mathematical reasoning (Shang et al., 2025), code generation (Lu et al., 2025), and complex question answering (Luo et al., 2025). This progress has spurred research into sophisticated agent architectures, including self-evolving systems that learn from feedback and experience (Shi et al., 2025b; Liu et al., 2025b), and dynamic topologies that adapt to task complexity (Li et al., 2025a;b). However, a critical paradox has emerged: as these autonomous systems become more capable and complex, they often become less robust and economically viable. Systemic inefficiencies lead to prohibitive computational costs, while their intricate internal interactions create new vectors for unpredictable failures (Zhang et al., 2025e).

This lack of robustness stems from the operational complexity of modern MAS, which introduces a significant reliability challenge. (Tian et al., 2025). The long chain of interactions inherent in these systems creates fertile ground for **error propagation** (Dong et al., 2025; Shen et al., 2025). For instance, a single piece of misinformation generated by an agent, a common risk with today's powerful yet occasionally hallucinatory foundation models (Kalai et al., 2025; Farquhar et al., 2024), can be committed to memory and subsequently poison the reasoning of all downstream agents(as explained in Figure 1a). These vulnerabilities mean that even a state-of-the-art MAS can fail on tasks well within its theoretical capabilities, simply due to a lack of operational robustness (Chen et al., 2024).

Furthermore, the issue of **economic efficiency** is a major barrier to the real-world deployment of MAS (Wang et al., 2025a). We identify two primary sources of this inefficiency. First, agents often struggle with excessively long observations, such as verbose web pages or tool outputs, which flood their context windows. This not only inflates token costs but can also obscure critical information, causing the agent to lose focus and derail its task execution (Hosseini et al., 2025). Second, agents may adopt sub-optimal strategies, entering into repetitive action loops or choosing unnecessarily complex paths to a solution (Cemri et al., 2025), further wasting computational resources (see Figure 1a). Our own analysis reveals that cutting-edge architectures can incur staggering token costs, with some GAIA benchmark (Mialon et al., 2023) tasks consuming upwards of two million tokens.

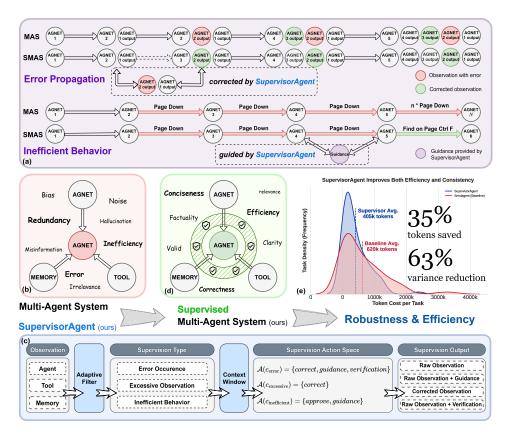


Figure 1: **The SUPERVISORAGENT Framework: Concept and Impact.** (a) Illustrative examples of common failure modes in MAS, including **error propagation** and **inefficient loops**, and the corresponding intervention by our SUPERVISORAGENT. (b) An overview of a conventional MAS, highlighting the high-risk interaction loci (agent-agent, agent-tool, agent-memory) where such failures occur. (c) The core workflow of our SUPERVISORAGENT, which monitors these interactions to provide real-time intervention. (d) The resulting Supervised MAS (SMAS), which integrates the Supervisor to enhance system-wide robustness and efficiency. (e) Performance on GAIA (Level 2), where SMAS (blue) reduces token cost by 35% and variance by 63% versus the baseline (red).

To address these intertwined challenges, we propose SUPERVISORAGENT, a lightweight and modular framework that enhances the **robustness** and **efficiency** of Multi-Agent Systems (MAS) through real-time supervision. Incorporating an adaptive filter, SUPERVISORAGENT enables proactive process control. It adaptively intervenes at critical junctures to mitigate key operational risks: it conducts proactive error diagnosis, provides pragmatic guidance for inefficient behaviors, and performs adaptive observation purification to reduce contextual noise from long observations.

In summary, our main contributions are:

- We propose and implement SUPERVISORAGENT, a novel, lightweight, and non-intrusive metaagent framework for real-time MAS supervision. It improves agent robustness and efficiency through proactive error correction, inefficiency guidance, and adaptive observation purification, without altering the base agents' architecture.
- 2. We conduct extensive experiments on the challenging **GAIA** benchmark and demonstrate a significant **Pareto improvement**. When applied to the Smolagent framework (Roucher et al., 2025), SUPERVISORAGENT reduces token consumption by an average of **29.68%** while maintaining competitive task success rates.
- 3. We validate the **general applicability** of our approach across five additional benchmarks spanning mathematical reasoning (GSM8k-Hard, AIME), code generation (HumanEval, MBPP), and question answering (DROP). Our method consistently delivers substantial efficiency gains, highlighted by a **23.74**% token reduction on HumanEval alongside an accuracy improvement. The framework's effectiveness is further confirmed across various foundation models, including the GPT-4.1, Gemini-2.5-pro, and Qwen3 series.

2 RELATED WORK

The increasing complexity of Multi-Agent Systems (MAS). Recent advancements in Large Language Models have spurred the development of increasingly sophisticated Multi-Agent Systems (MAS) capable of tackling complex, multi-step tasks (Tran et al., 2025; He et al., 2025). Frameworks like Tongyi DeepResearch (Team, 2025), AgentOrchestra (Zhang et al., 2025f), and Aime (Shi et al., 2025a) exemplify this trend, introducing complex features such as hierarchical structures (Zhu et al., 2025; Cheng et al., 2025), dynamic agent management (Wu et al., 2025; Zhang et al., 2025g), and end-to-end training (Li et al., 2025b; Ye et al., 2025). However, this escalating architectural complexity invariably introduces significant challenges in maintaining operational robustness and computational efficiency, which we address in this work.

Failure attribution and robustness. A significant body of work has emerged to address the challenge of MAS robustness, primarily focusing on post-hoc *failure attribution* (Zhang et al., 2025e). Systems like Aegis (Song et al., 2025) and SHIELDA (Zhou et al., 2025) propose taxonomies for failure analysis, while AgenTracer (Zhang et al., 2025b) and A2P (West et al., 2025) introduce methods to better trace the root causes of task failures. While valuable, these methods are fundamentally reactive, analyzing failures after they have occurred. In contrast, our SUPERVISORAGENT is designed for *proactive, real-time intervention*, aiming to detect and mitigate high-risk steps *before* they lead to systemic failure.

Efficient Multi-Agent Systems Another stream of research targets the efficiency of MAS, a critical factor largely driven by token consumption. Most approaches focus on *design-time optimization*. Some prune the system's architecture by eliminating agents with AgentDropout (Wang et al., 2025b) or communication links with SafeSieve (Zhang et al., 2025d). Others generatively construct efficient prompts (Han et al., 2025) or agent topologies from the outset, as seen in MetaAgent (Zhang et al., 2025h), MaAS (Zhang et al., 2025a), and HiVA (Tang et al., 2025). A second direction, *context compression*, aims to reduce token count by summarizing or distilling observations (Chen et al., 2025; Mou et al., 2025). Our work is orthogonal to these methods. Instead of focusing on static design or message content, we introduce **runtime process control**. SUPERVISORAGENT addresses dynamic inefficiencies *during* execution, a complementary approach that can enhance existing systems.

3 Preliminary

In this section, we first establish a formalism for our proposed Supervised Multi-Agent System (SMAS). We then detail the core components of our framework: the SupervisorAgent's action space and the contextual information it leverages for decision-making.

3.1 A FORMALISM FOR SUPERVISED MULTI-AGENT SYSTEMS

Our work is predicated on the idea that the complex, often chaotic, interactions within a Multi-Agent System (MAS) can be actively managed to improve both robustness and efficiency. To formalize this, we introduce the concept of a **Supervised Multi-Agent System**.

Definition 1 (Supervised Multi-Agent System (SMAS)). A SMAS is a Multi-Agent System augmented with a meta-level control agent, henceforth referred to as the Supervisor. The Supervisor's objective is to monitor agent interactions in real-time, proactively detecting and mitigating operational risks without altering the core logic of the agents it oversees. In this work, we implement this conceptual Supervisor as a concrete agent named SupervisorAgent.

The fundamental unit of supervision is the **interaction**, which occurs when an agent engages with other system components. We categorize interactions into three primary types(see in Figure 1b):

- 1. **Agent-Agent Interactions:** Communication or delegation between agents. In architectures like ReAct (Yao et al., 2023), where an agent's output becomes another's input, this channel is highly susceptible to the propagation of hallucinated or erroneous information (Shen et al., 2025);
- 2. **Agent-Tool Interactions:** The invocation of external tools or APIs. This interaction is a primary source of external information, but it is also fraught with risks, including factually incorrect, irrelevant, or outdated data that can corrupt the agent's context (Qian et al., 2025);

3. **Agent-Memory Interactions:** The retrieval of information from short- or long-term memory stores. While crucial for self-evolving systems, memory introduces the hazard of acting upon stale or flawed information from past experiences (Xiong et al., 2025).

3.2 THE SUPERVISOR AGENT'S CONTEXT WINDOW

To make informed decisions, the SUPERVISORAGENT is provided with a rich, real-time snapshot of the MAS's state, which we formalize as the *context window*.

Definition 2 (Context Window). The standard context window, W, is a tuple of five key elements:

$$\mathcal{W} = (N, Q_a, Q_l, T_l, S),$$

where N is the name of the agent under review, Q_g and Q_l are the global and local tasks, T_l is the **local trace** of agent A's recent actions and observation summaries, and S is a summary of the agent's latest interaction step. For diagnosing system-wide inefficiencies, we augment this to an extended context window $W_{\text{ext}} = W \cup \{T_g\}$, where T_g is the **global trace** of all agent interactions.

3.3 THE SUPERVISOR'S ACTION SPACE

The role of the Supervisor is to diagnose high-risk interactions and execute a targeted intervention (Figure 2c). We define three primary intervention contexts, $c \in \mathcal{C} = \{c_{\text{error}}, c_{\text{excessive}}, c_{\text{inefficient}}\}$, which activate one of three core supervision strategies:

- **Proactive Error Correction:** Triggered by c_{error} , this strategy aims to diagnose the root cause of an explicit error and provide a direct fix or a verification task to resolve it.
- Adaptive Observation Purification: Triggered by c_{excessive}, this strategy refines excessively long
 or noisy observations to improve the signal-to-noise ratio for the agent.
- Guidance for Inefficiency: Triggered by $c_{\text{inefficient}}$, this strategy provides pragmatic, course-correcting hints for sub-optimal behaviors, while also critically permitting productive, albeit repetitive, processes to continue via an *approve* action.

These strategies are implemented by selecting an action a from the global action space \mathcal{A} . The specific subset of permissible actions, $\mathcal{A}(c)$, is formally defined by the intervention context as follows:

```
\mathcal{A}(c) = \begin{cases} \{correct\_observation, provide\_guidance, run\_verification\} & \text{if } c = c_{\text{error}} \\ \{correct\_observation\} & \text{if } c = c_{\text{excessive}} \\ \{approve, provide\_guidance\} & \text{if } c = c_{\text{inefficient}} \end{cases}
```

The implementation of each action is detailed in Section 4.3.

4 METHODOLOGY

Building upon the formalism of a Supervised Multi-Agent System (SMAS) introduced in Section 3, we now detail the architecture and operational workflow of our SUPERVISORAGENT (illustrated in Figure 2). Our methodology is structured around three fundamental questions: **What** to supervise, **When** to supervise, and **How** to supervise. We defer the specific implementation details, including all hyperparameters and prompts, to Appendix A.3 A.5.

4.1 What to Supervise: High-Risk Interaction Points

The primary targets for our supervision are the three high-risk interaction points defined in our preliminary formalism (Section 3.1, see also Figure 2a): Agent-Agent, Agent-Tool, and Agent-Memory interactions. These points are the primary channels through which errors and inefficiencies are introduced and propagated throughout the system. Our goal is to monitor these specific channels to maintain the operational integrity of the MAS.

4.2 When to Supervise: The Adaptive Filter

While a naive approach might monitor every interaction, the associated computational cost is prohibitive and would undermine our goal of improving efficiency. Therefore, the cornerstone of our

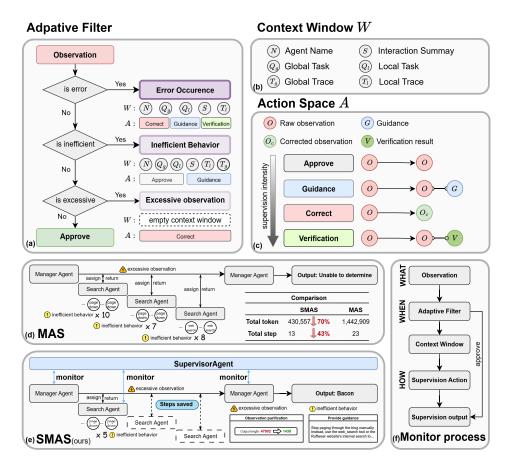


Figure 2: **The architecture and workflow of SUPERVISORAGENT.** (a) The LLM-free adaptive filter for identifying high-risk interactions. (b) The context window, aggregating goals and traces for situational awareness. (c) The spectrum of intervention actions, from simple approval to intensive verification. (d, e) Case study on a GAIA task, comparing the baseline MAS (d) with our SMAS (e), which cuts steps by 43% and token cost by over 70%. (f) The supervise workflow for an interaction, from filtering to a final supervision action.

framework is a lightweight, LLM-free **adaptive filter** designed to trigger supervision only at critical junctures (as shown in the case studies in Figures 2d and 2e). This approach ensures that the Supervisor's resources are deployed judiciously, maximizing impact while minimizing overhead. The filter is designed to be fast and heuristic-based, monitoring the MAS for three pre-defined, high-risk scenarios:

- Error occurrence. The manifestation of an explicit error (e.g., in tool use or code execution) is a critical trigger. Unlike standard MAS that often pass the full error log into a cluttered context for a subsequent agent to debug, our filter immediately flags these events for a focused, real-time intervention.
- Inefficient behavior. An agent may enter a loop of sub-optimal or repetitive actions that, while not explicit errors, lead to high token consumption and latency. Our filter is designed to detect such patterns, such as an agent repeatedly using the page_down action instead of a more direct search strategy.
- Excessive observation length. Interactions with tools can return excessively long and noisy observations (e.g., raw HTML) that inflate costs and distract the agent. Our filter identifies such cases for immediate information purification.

4.3 How to Supervise: Memory-Augmented, Multi-Level Intervention

Once a high-risk interaction is flagged, SUPERVISORAGENT leverages a rich context window and a spectrum of intervention strategies to deliver a nuanced, effective response.

Memory-augmented context window. To make an effective decision, a supervisor must possess a more comprehensive understanding of the system's state than any single agent. This is why SUPERVISORAGENT is conceptualized with its own memory module, not a simple monitor. As illustrated in Figure 2b, this is achieved through a dynamic context window \mathcal{W} , which aggregates the global task Q_g , the agent's local task Q_l , interaction summary S, and its recent action trace T_l . Crucially, for diagnosing complex inefficiencies, SUPERVISORAGENT also accesses the global trace T_g , granting it a holistic perspective that transcends the limited view of any individual agent. This elevated viewpoint is what enables it to provide genuinely strategic guidance.

A spectrum of intervention actions. With this rich context, SUPERVISORAGENT selects an action from a multi-level action space \mathcal{A} , adapting intervention intensity tailored to issue severity (Figure 2c). These actions range from a minimal nudge to a comprehensive correction:

- *approve*: A minimal intervention that permits a productive, albeit repetitive, agent behavior to continue. Primarily used in the *inefficient* context, its purpose is to avoid disrupting a process that is pragmatically the best path forward from its current state.
- *provide_guidance*: A semi-intrusive action that steers an agent away from a sub-optimal strategy or logical flaw. This action appends a concise, directive hint to the existing observation, correcting the agent's reasoning path without altering the core context data.
- *correct_observation*: A direct and forceful intervention that refines the agent's sensory input. It is the sole action for *excessive observations*, where it purifies the content, and is also used in *error* contexts to fix factually incorrect data. This action replaces the original raw observation entirely with a cleaned and corrected version.
- *run_verification*: The deepest intervention, used in complex *error* contexts when internal information is insufficient. It invokes a verification sub-agent for external fact-checking or advanced debugging, returning a definitive, verified result.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We empirically validate the effectiveness of SUPERVISORAGENT through a series of extensive experiments. We begin by outlining our evaluation metrics, datasets, and baselines. For a more detailed description of the experimental settings, please refer to Appendix A.2.

Datasets. We evaluate our method on a diverse suite of six benchmarks spanning three domains. Our primary benchmark is the challenging GAIA validation set (Mialon et al., 2023), which provides a comprehensive test of an MAS's general problem-solving capabilities. To demonstrate broader applicability, we use five additional benchmarks: for mathematical reasoning, we use AIME (HuggingFaceH4, 2024) and a random subset of 600 samples from GSM8k-Hard (Gao et al., 2022); for code generation, we use the full HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) datasets; and for question answering, we use a subset of 800 samples from the DROP (Dua et al., 2019) dataset, following the sampling strategy of prior work (Zhang et al., 2025c).

Baselines. On several benchmarks, we compare SUPERVISORAGENT against a comprehensive set of agentic systems equipped with web-browsing and code execution capabilities. These baselines fall into two categories: (1) single agent execution methods: including vanilla LLM, Self Consistency CoT(3 answers) (Wang et al., 2023), and CodeAgent (Roucher et al., 2025); and (2) multi-agent systems, including Smolagent (Roucher et al., 2025), MetaAgent (Zhang et al., 2025h), OWL(role playing) (Hu et al., 2025), and AWorld (Yu et al., 2025).

Implementation details. When considering the integration of SUPERVISORAGENT with existing MAS, we select Smolagent (Roucher et al., 2025) as our primary baseline due to its open-source availability, robust performance, and modular architecture that facilitates seamless integration. Besides, some current MASs' strong capabilities stem from comprehensive MCP tools, which essentially provide high-quality information that greatly simplifies the agent's task. As for Smolagent, its capabilities mostly arise from the agentic interactions inside of it, which is more suitable for evaluating the effectiveness of SUPERVISORAGENT.

Table 1: Overall performance on the GAIA validation set. Our SMAS consistently reduces the average token cost comparing to Smolagent baseline (Smol.) while achieving competitive pass@k success rates.

Method	Avg. Acc.	Avg. Tokens (K)	L1 Acc.	L1 Tokens (K)	L2 Acc.	L2 Tokens (K)	L3 Acc.	L3 Tokens (K)
CodeAgent	40.00	120.40	56.60	92.84	34.88	131.90	23.08	138.54
OWL	45.40	111.07	56.56	67.72	43.02	110.36	29.16	209.34
OAgents	49.09	340.50	66.04	260.27	47.67	358.63	19.23	444.11
Smolagent	50.91	527.76	62.26	298.51	53.49	619.59	19.23	691.33
AWorld	60.00	128.27	67.92	69.61	62.79	164.08	34.62	133.65
				pass@1				
Smol.	50.91	527.76	62.26	298.51	53.49	619.59	19.23	691.33
+ SMAS (ours)	50.91	371.12 \29.68%	62.26	258.28 \13.48%	51.16	404.96 _34.64%	26.92 \\ 7.69\%	489.22 \29.23%
				pass@2				
Smol.	58.18	467.19	69.81	275.85	59.30	548.02	30.77	589.92
+ SMAS(ours)	57.58	387.70 \17.02%	73.58 †3.77%	270.07 \(\psi_2.10\)%	54.65	417.41 \(\pi23.83\)%	34.62 †3.85%	529.20 \10.29%
				pass@3				
Smol.	61.82	502.40	71.70	282.14	63.95	605.05	34.62	611.87
+ SMAS (ours)	63.03 †1.21%	369.52 \26.45%	75.47 †3.77%	276.84 11.88%	62.79	409.05 \132.39%	38.46 13.84%	427.72 _30.10%

Framework Selection We selected Smolagent as our primary experimental testbed due to its modular architecture, which is conducive to seamless integration. Critically, Smolagent's capabilities stem primarily from its internal agentic interactions rather than powerful external tools(e.g. web APIs or solvers). This provides an ideal, controlled environment to isolate and evaluate the direct impact of our SUPERVISORAGENT on an agent's core reasoning and communication processes.

Metrics. For GAIA and the code generation benchmarks, we report the standard pass@k metric. For our main baseline, Smolagent, we report pass@1, 2, and 3. For math reasoning, we report the final solve rate (%). For question answering, we report the F1 score for DROP. In all experiments, we meticulously track and report the total token consumption as a primary measure of efficiency.

5.2 RESULTS AND ANALYSIS

Significant efficiency gains with competitive accuracy. The main experimental results, presented in Table 1, confirm the substantial benefits of SUPERVISORAGENT. On the GAIA validation set, when integrated with the Smolagent framework, SUPERVISORAGENT achieves an average token reduction of 29.68% at pass@1, while maintaining a statistically equivalent success rate. Notably, the efficiency gains are even more pronounced on more difficult tasks, with token savings reaching 32.39% on Level 2 and 30.10% on Level 3 tasks at pass@3.

Across the other five benchmarks, SUPERVISORAGENT mostly delivers a Pareto improvement. In mathematical reasoning, it improves the solve rate on AIME by 6.67% while simultaneously reducing token costs by 18.92%. Most strikingly, in code generation, it not only holds competitive accuracy on HumanEval, but also cuts token consumption by 23.74%. These results underscore SUPERVISORAGENT's ability to act as a universal efficiency enhancer across diverse problem domains.

Model-Agnostic generalization. To demonstrate that the benefits of SUPERVISORAGENT are architectural rather than model-specific, we evaluated it with three different powerful LLMs as its inference engine on GAIA. As shown in Figure 4b, SUPERVISORAGENT consistently yields significant token savings and maintains robust performance across all models, including GPT-4.1, Gemini-2.5-pro, and Qwen3-235B. This validates that our supervision framework is a model-agnostic component that can enhance a wide variety of LLM-powered agent systems.

Improving robustness and performance consistency. Beyond average performance, we define robustness as the consistency of an agent's performance. As illustrated by the violin plots in Figure 3, SUPERVISORAGENT *significantly reduces the variance in token consumption per task.* The distributions for the supervised MAS are visibly shorter and wider, indicating a more concentrated and predictable performance profile. The bar chart on the right further quantifies this, showing a marked decrease in token cost variance, especially for the more complex Level 2 and 3 tasks. This demonstrates that *our method not only makes the MAS more efficient on average but also more reliable and less prone to extreme resource consumption outliers.*

Table 2: **Generalization across diverse benchmarks.** SUPERVISORAGENT consistently reduces token costs while maintaining or improving accuracy on tasks spanning mathematical reasoning, code generation, and question answering. All reported gains are relative to the Smolagent baseline.

Method	Metrics	GSM-hard	AIME	HumanEval	MBPP	DROP
Vanilla	Acc / F1 (%)	67.17	26.67	76.82	80.09	76.36
	Avg. Tokens (K)	0.37	2.01	0.28	0.27	0.46
CoT SC(3-shot)	Acc / F1 (%)	69.01	30.00	77.78	81.26	77.72
	Avg. Tokens (K)	2.62	14.26	1.42	1.29	2.73
OWL	Acc / F1 (%)	72.48	33.33	90.74	79.08	79.85
	Avg. Tokens (K)	15.67	56.11	31.87	54.80	11.47
MetaAgent	Acc / F1 (%)	72.14	26.67	74.08	79.86	78.16
	Avg. Tokens (K)	4.35	6.24	2.59	6.39	1.43
Smolagent	Acc / F1 (%)	74.33	30.00	92.07	85.68	81.08
	Avg. Tokens (K)	11.59	59.14	40.91	111.07	12.01
+ SMAS (ours)	Acc / F1 (%)	75.50	36.67	92.68	84.43	79.80
	Avg. Tokens (K)	10.55 \ \displayseq 8.92%	47.95 ↓18.92%	31.19 \(\pm23.74\)%	103.71 \6.62%	11.34 \square.60%

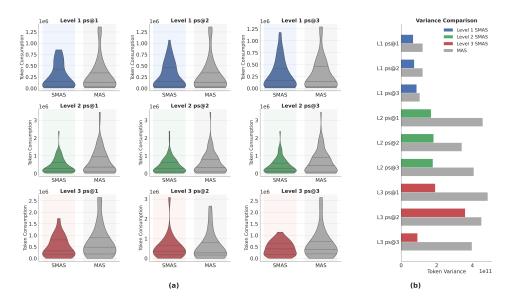
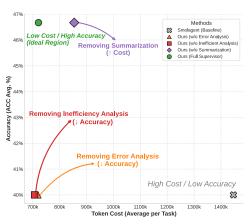


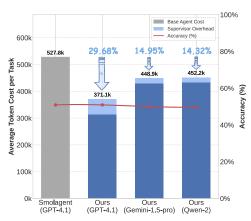
Figure 3: SUPERVISORAGENT enhances performance consistency on the GAIA benchmark. (a) Violin plots of token cost distributions, revealing the more compact and predictable performance of our Supervised MAS (SMAS). (b) A direct comparison quantifying the substantial reduction in token cost variance achieved by our SMAS across all difficulty levels.

Ablation Study Our ablation study, conducted on the 30 most token-intensive GAIA tasks, dissects the contribution of SUPERVISORAGENT's three core strategies (Table 3, Figure 4a). We compare the full SUPERVISORAGENT against three ablated versions: **w/o Correction** (disabling Proactive Error Correction), **w/o Guidance** (disabling Guidance for Inefficiency), and **w/o Purification** (disabling Adaptive Observation Purification). The results reveal that Purification is the primary driver of token reduction. Concurrently, removing the Correction and Guidance modules leads to a significant drop in accuracy, confirming their critical role in ensuring task success and robustness. This highlights a clear trade-off: while Purification is key to efficiency, the other two strategies are crucial for maximizing performance, justifying their marginal token cost by preventing costly failures.

Table 3: **Ablation study of SUPERVISORAGENT's components** on the 30 most token-intensive tasks from the GAIA validation set.

Method	Avg. Acc.	Avg. Token	Level 1 Avg. Token	Level 2 Avg. Token	Level 3 Avg. Token
Smolagent	40.00	1,446,526	933,013	2,037,437	1,369,131
+ SMAS (w/o Correction)	40.00	719,075 \$\psi_50.28\%	426,786 _54.26%	755,543 \62.91%	974,895 \(\pmu\)28.79%
+ SMAS (w/o Guidance)	40.00	706,831 \$_51.14\%	453,623 _51.38%	913,109 _55.18%	753,761 444.95%
+ SMAS (w/o Purification)	46.67 + 6.67%	851,747 \41.11%	585,411 \ \ 37.26\%	990,769 \$\frac{1.37\%}{}	979,061 \(\pi \text{28.49\%}\)
+ SMAS	46.67 16.67%	721,332 _50.13%	522,364 \44.01%	960,694 _52.85\%	680,939 \$\psi_50.26\%





(a) Ablation study on GAIA.

(b) Model Generalization of SUPERVISORAGENT.

Figure 4: **Ablation study and model generalization of SUPERVISORAGENT.** (a) Ablation study on challenging GAIA tasks, dissecting the distinct contributions of each module to the framework's overall efficiency and robustness. (b) Validation of model-agnosticism, showing that SUPERVISORAGENT consistently delivers token savings across diverse foundation models.

6 DISCUSSION AND CONCLUSION

In this work, we introduced **SUPERVISORAGENT**, a lightweight and non-intrusive meta-agent framework that enhances the robustness and efficiency of Multi-Agent Systems. Through real-time, adaptive supervision, SUPERVISORAGENT mitigates common failure modes and reduces computational overhead using three core strategies: proactive error correction, pragmatic inefficiency guidance, and adaptive observation purification. Our extensive experiments demonstrate a significant Pareto improvement. On the challenging GAIA benchmark, SUPERVISORAGENT reduces token consumption by an average of 29.68% while maintaining competitive task success rates, a crucial step towards building more practical and scalable agentic systems.

Our work also yields critical insights for the broader field. First, we discovered that seemingly "noisy" information, such as HTML structure and truncation cues, serves as a vital signal for ReAct-style agents; overly aggressive purification can paradoxically harm performance. This highlights a fundamental trade-off between information density and the preservation of environmental texture. Second, our focus on token cost underscores the need for a more holistic efficiency evaluation for MAS. A comprehensive analysis must also account for the frequency and complexity of external tool API calls, which offload significant burdens from the LLM. This very trade-off informed our choice of Smolagent as a primary baseline; its reliance on internal agentic reasoning, rather than powerful external tools, provided a controlled environment to isolate and evaluate our supervisor's impact on the interaction process itself.

These insights inform several promising avenues for future work. While our SUPERVISORAGENT proved robust, developing a self-evolving, memory-augmented version remains an exciting opportunity. Other key directions include creating more sophisticated purification techniques that address the "noise-as-signal" problem and developing a universal resource consumption metric for MAS. Ultimately, we posit that incorporating such real-time, meta-level supervision is a foundational component for building the next generation of truly scalable and reliable autonomous systems.

ETHICS STATEMENT

Our work aims to improve the reliability and efficiency of Multi-Agent Systems, a crucial step for developing practical and beneficial autonomous technologies. We believe that by introducing a mechanism for real-time supervision, our framework provides a paradigm not only for performance optimization but also for enhancing the safety and predictability of future agentic systems. Our research was conducted on publicly available benchmarks, did not involve private user data, and adheres to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

We are committed to ensuring our work is reproducible. The core architecture and logic of SUPERVI-SORAGENT are detailed in Section 4, with theoretical formalisms in Section 3. For direct replication, we provide all implementation details and final prompts in Appendix A.3, A.5, and our full source code, including experiment scripts, will be made publicly available. The datasets and metrics used in our extensive experiments (Section 5) are all based on publicly available benchmarks, allowing for direct comparison and validation of our results.

REFERENCES

Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian, Zhenghailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: On path to artificial super intelligence, 2025. URL https://arxiv.org/abs/2507.21046.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent llm systems fail?, 2025. URL https://arxiv.org/abs/2503.13657.

Junzhi Chen, Juhao Liang, and Benyou Wang. Smurfs: Multi-agent system using context-efficient dfsdt for tool planning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3281–3298, 2025.

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

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases, 2024. URL https://arxiv.org/abs/2407.12784.

Yuyang Cheng, Yumiao Xu, Chaojia Yu, and Yong Zhao. Hawk: A hierarchical workflow framework for multi-agent collaboration, 2025. URL https://arxiv.org/abs/2507.04067.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

- Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. A practical memory injection attack against llm agents, 2025. URL https://arxiv.org/abs/2503.03704.
- Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246/.
- Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, jun 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07421-0. URL https://www.nature.com/articles/s41586-024-07421-0.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*, 2022.
- Yichen Han, Bojun Liu, Zhengpeng zhou, Guanyu Liu, Zeng Zhang, Yang Yang, Wenli Wang, Isaac N Shi, Yunyan, Lewei He, and Tianyu Shi. Mapgd: Multi-agent prompt gradient descent for collaborative prompt optimization, 2025. URL https://arxiv.org/abs/2509.11361.
- Junda He, Christoph Treude, and David Lo. Llm-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 34(5):1–30, 2025.
- Peyman Hosseini, Ignacio Castro, Iacopo Ghinassi, and Matthew Purver. Efficient solutions for an intriguing failure of LLMs: Long context window does not mean LLMs can analyze long sequences flawlessly. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 1880–1891, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.128/.
- Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Qiguang Chen, Zeyu Zhang, Yifeng Wang, Qianshuo Ye, Bernard Ghanem, Ping Luo, and Guohao Li. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation, 2025. URL https://arxiv.org/abs/2505.23885.
- HuggingFaceH4. Aime 2024 dataset. https://huggingface.co/datasets/ HuggingFaceH4/aime_2024, 2024.
- Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. Why language models hallucinate, 2025. URL https://arxiv.org/abs/2509.04664.
- Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation, 2025a. URL https://arxiv.org/abs/2507.18224.
- Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, Hongxuan Lu, Tianrui Qin, Chenghao Zhu, Yi Yao, Shuying Fan, Xiaowan Li, Tiannan Wang, Pai Liu, King Zhu, He Zhu, Dingfeng Shi, Piaohong Wang, Yeyi Guan, Xiangru Tang, Minghao Liu, Yuchen Eleanor Jiang, Jian Yang, Jiaheng Liu, Ge Zhang, and Wangchunshu Zhou. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic rl, 2025b. URL https://arxiv.org/abs/2508.13167.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, Yuheng Cheng, Suyuchen Wang, Xiaoqiang Wang, Yuyu Luo, Haibo Jin, Peiyan Zhang, Ollie Liu, Jiaqi Chen, Huan Zhang, Zhaoyang Yu, Haochen Shi, Boyan Li, Dekun Wu, Fengwei Teng, Xiaojun Jia, Jiawei Xu, Jinyu Xiang, Yizhang Lin, Tianming Liu, Tongliang Liu, Yu Su, Huan Sun, Glen Berseth, Jianyun Nie, Ian Foster, Logan Ward, Qingyun Wu, Yu Gu, Mingchen Zhuge, Xinbing Liang, Xiangru Tang, Haohan Wang, Jiaxuan You, Chi Wang, Jian Pei, Qiang Yang, Xiaoliang Qi, and Chenglin Wu. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems, 2025a. URL https://arxiv.org/abs/2504.01990.

- Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners, 2025b. URL https://arxiv.org/abs/2504.14239.
- Xu Lu, Weisong Sun, Yiran Zhang, Ming Hu, Cong Tian, Zhi Jin, and Yang Liu. Requirements development and formalization for reliable code generation: A multi-agent vision, 2025. URL https://arxiv.org/abs/2508.18675.
- Yajie Luo, Yihong Wu, Muzhi Li, Fengran Mo, Jia Ao Sun, Xinyu Wang, Liheng Ma, Yingxue Zhang, and Jian-Yun Nie. An entity linking agent for question answering, 2025. URL https://arxiv.org/abs/2508.03865.
- Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants, 2023. URL https://arxiv.org/abs/2311.12983.
- Xinyi Mou, Chen Qian, Wei Liu, Xuanjing Huang, and Zhongyu Wei. Ecolang: Efficient and effective agent communication language induction for social simulation, 2025. URL https://arxiv.org/abs/2505.06904.
- Cheng Qian, Emre Can Acikgoz, Hongru Wang, Xiusi Chen, Avirup Sil, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Smart: Self-aware agent for tool overuse mitigation, 2025. URL https://arxiv.org/abs/2502.11435.
- Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 'smolagents': a smol library to build great agentic systems. https://github.com/huggingface/smolagents, 2025.
- Ning Shang, Yifei Liu, Yi Zhu, Li Lyna Zhang, Weijiang Xu, Xinyu Guan, Buze Zhang, Bingcheng Dong, Xudong Zhou, Bowen Zhang, Ying Xin, Ziming Miao, Scarlett Li, Fan Yang, and Mao Yang. rstar2-agent: Agentic reasoning technical report, 2025. URL https://arxiv.org/abs/2508.20722.
- Xu Shen, Yixin Liu, Yiwei Dai, Yili Wang, Rui Miao, Yue Tan, Shirui Pan, and Xin Wang. Understanding the information propagation effects of communication topologies in llm-based multi-agent systems, 2025. URL https://arxiv.org/abs/2505.23352.
- Yexuan Shi, Mingyu Wang, Yunxiang Cao, Hongjie Lai, Junjian Lan, Xin Han, Yu Wang, Jie Geng, Zhenan Li, Zihao Xia, Xiang Chen, Chen Li, Jian Xu, Wenbo Duan, and Yuanshuo Zhu. Aime: Towards fully-autonomous multi-agent framework, 2025a. URL https://arxiv.org/abs/2507.11988.
- Yucheng Shi, Wenhao Yu, Zaitang Li, Yonglin Wang, Hongming Zhang, Ninghao Liu, Haitao Mi, and Dong Yu. Mobilegui-rl: Advancing mobile gui agent through reinforcement learning in online environment, 2025b. URL https://arxiv.org/abs/2507.05720.
- Kevin Song, Anand Jayarajan, Yaoyao Ding, Qidong Su, Zhanda Zhu, Sihang Liu, and Gennady Pekhimenko. Aegis: Taxonomy and optimizations for overcoming agent-environment failures in llm agents, 2025. URL https://arxiv.org/abs/2508.19504.
- Jinzhou Tang, Jusheng Zhang, Qinhan Lv, Sidi Liu, Jing Yang, Chengpei Tang, and Keze Wang. Hiva: Self-organized hierarchical variable agent via goal-driven semantic-topological evolution, 2025. URL https://arxiv.org/abs/2509.00189.

- Tongyi DeepResearch Team. Tongyi-deepresearch. https://github.com/Alibaba-NLP/DeepResearch, 2025.
- Fangqiao Tian, An Luo, Jin Du, Xun Xian, Robert Specht, Ganghua Wang, Xuan Bi, Jiawei Zhou, Ashish Kundu, Jayanth Srinivasa, Charles Fleming, Rui Zhang, Zirui Liu, Mingyi Hong, and Jie Ding. An outlook on the opportunities and challenges of multi-agent ai systems, 2025. URL https://arxiv.org/abs/2505.18397.
- Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. Multi-agent collaboration mechanisms: A survey of llms, 2025. URL https://arxiv.org/abs/2501.06322.
- Ningning Wang, Xavier Hu, Pai Liu, He Zhu, Yue Hou, Heyuan Huang, Shengyu Zhang, Jian Yang, Jiaheng Liu, Ge Zhang, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. Efficient agents: Building effective agents while reducing cost, 2025a. URL https://arxiv.org/abs/2508.02694.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.
- Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. Agentdropout: Dynamic agent elimination for token-efficient and high-performance llm-based multi-agent collaboration, 2025b. URL https://arxiv.org/abs/2503.18891.
- Alva West, Yixuan Weng, Minjun Zhu, Zhen Lin, and Yue Zhang. Abduct, act, predict: Scaffolding causal inference for automated failure attribution in multi-agent systems, 2025. URL https://arxiv.org/abs/2509.10401.
- Feijie Wu, Zitao Li, Fei Wei, Yaliang Li, Bolin Ding, and Jing Gao. Talk to right specialists: Routing and planning in multi-agent system for question answering, 2025. URL https://arxiv.org/abs/2501.07813.
- Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts llm agents: An empirical study of experience-following behavior, 2025. URL https://arxiv.org/abs/2505.16067.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.
- Rui Ye, Shuo Tang, Rui Ge, Yaxin Du, Zhenfei Yin, Siheng Chen, and Jing Shao. Mas-gpt: Training llms to build llm-based multi-agent systems, 2025. URL https://arxiv.org/abs/2503.03686.
- Chengyue Yu, Siyuan Lu, Chenyi Zhuang, Dong Wang, Qintong Wu, Zongyue Li, Runsheng Gan, Chunfeng Wang, Siqi Hou, Gaochi Huang, Wenlong Yan, Lifeng Hong, Aohui Xue, Yanfeng Wang, Jinjie Gu, David Tsai, and Tao Lin. Aworld: Orchestrating the training recipe for agentic ai, 2025. URL https://arxiv.org/abs/2508.20404.
- Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. *arXiv preprint arXiv:2502.04180*, 2025a.
- Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. Agentracer: Who is inducing failure in the llm agentic systems?, 2025b. URL https://arxiv.org/abs/2509.03312.
- Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. Aflow: Automating agentic workflow generation, 2025c. URL https://arxiv.org/abs/2410.10762.

- Ruijia Zhang, Xinyan Zhao, Ruixiang Wang, Sigen Chen, Guibin Zhang, An Zhang, Kun Wang, and Qingsong Wen. Safesieve: From heuristics to experience in progressive pruning for llm-based multi-agent communication, 2025d. URL https://arxiv.org/abs/2508.11733.
- Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems, 2025e. URL https://arxiv.org/abs/2505.00212.
- Wentao Zhang, Liang Zeng, Yuzhen Xiao, Yongcong Li, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. Agentorchestra: A hierarchical multi-agent framework for general-purpose task solving, 2025f. URL https://arxiv.org/abs/2506.12508.
- Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23378–23386, 2025g.
- Yaolun Zhang, Xiaogeng Liu, and Chaowei Xiao. Metaagent: Automatically constructing multiagent systems based on finite state machines, 2025h. URL https://arxiv.org/abs/2507.22606.
- Jingwen Zhou, Jieshan Chen, Qinghua Lu, Dehai Zhao, and Liming Zhu. Shielda: Structured handling of exceptions in llm-driven agentic workflows, 2025. URL https://arxiv.org/abs/2508.07935.
- He Zhu, Tianrui Qin, King Zhu, Heyuan Huang, Yeyi Guan, Jinxiang Xia, Yi Yao, Hanhao Li, Ningning Wang, Pai Liu, Tianhao Peng, Xin Gui, Xiaowan Li, Yuhui Liu, Yuchen Eleanor Jiang, Jun Wang, Changwang Zhang, Xiangru Tang, Ge Zhang, Jian Yang, Minghao Liu, Xitong Gao, Jiaheng Liu, and Wangchunshu Zhou. Oagents: An empirical study of building effective agents, 2025. URL https://arxiv.org/abs/2506.15741.

CONTENTS Introduction **Related Work Preliminary** 3.3 4 Methodology How to Supervise: Memory-Augmented, Multi-Level Intervention **Experiments** 5.1 **Discussion and Conclusion** A Appendix

A APPENDIX

A.1 LLM USAGE

The large language model (LLM) was utilized as a writing assistant during the preparation of this manuscript. Its application was strictly limited to improving the clarity and grammatical accuracy of the text. Specific uses included rephrasing sentences for better flow and translating initial concepts and drafts from Chinese to English. All core scientific contributions, including the conceptualization of our SUPERVISORAGENT framework, the design of the methodology and experiments, and the analysis and interpretation of the results, are solely the work of the authors. The authors take full responsibility for all claims and the final content of this paper.

A.2 EXPERIMENTAL SETUP

A.2.1 DATASETS

Here, we provide a detailed introduction to the datasets used in this paper:

- GAIA (Mialon et al., 2023) serves as a benchmark designed to evaluate next-generation LLMs that possess enhanced capabilities through the incorporation of tools, efficient prompting strategies, and access to external search resources. This benchmark comprises over 450 challenging questions, each with a clear and unequivocal answer, necessitating varying degrees of tooling and autonomy for resolution. Accordingly, the questions are categorized into three distinct levels: Level 1 is expected to be solvable by proficient LLMs, while Level 3 signifies a substantial increase in the model's capabilities. Each level includes a fully public development set for validation purposes, as well as a test set containing private answers and associated metadata. In our experiments, we utilize the test set, which encompasses 164 tasks.
- **GSM-hard** (Gao et al., 2022) is an advanced version of the GSM8K mathematics reasoning dataset (Cobbe et al., 2021). This enhanced dataset presents models with increased challenges, featuring larger numerical values and more complex relationships within the problems.
- AIME-2024 (HuggingFaceH4, 2024) is a dataset comprising problems derived from the American Invitational Mathematics Examination (AIME) 2024. AIME is a prestigious mathematics competition for high school students, recognized for its challenging problems that span various mathematical domains. This benchmark serves multiple purposes: it evaluates the mathematical reasoning capabilities of LLMs, assesses their problem-solving abilities on complex mathematical challenges, and investigates AI performance on structured mathematical tasks.
- HumanEval (Chen et al., 2021) is a dataset released by OpenAI that includes 164 programming problems, each containing a function signature, a docstring, a body, and several associated unit tests. These problems were handwritten to ensure that they were not included in the training dataset for code-generation models. This benchmark is crucial for evaluating code-generation models, providing a structured set of challenges in Python that facilitates the assessment of both the quality and correctness of code produced by language models.
- MBPP(Mostly Basic Python Problems Dataset) (Austin et al., 2021) comprises approximately 1,000 crowd-sourced Python programming problems that are specifically designed to be solvable by entry-level programmers. The dataset covers essential programming fundamentals and standard library functionalities. Each problem includes a task description, a corresponding code solution, and three automated test cases.
- **DROP**(Data Retrieval Open Answering) Dua et al. (2019) is a reading comprehension benchmark that requires discrete reasoning over paragraphs. This dataset consists of 96,000 questions developed through crowd sourcing and adversarial methods. It challenges systems to resolve references within the questions, which may point to multiple input positions. The tasks entail performing discrete operations, such as addition, counting, and sorting, necessitating a substantially more comprehensive understanding of paragraph content than that demanded by prior datasets. In our experiment, we sampled 800 tasks for evaluation.

A.2.2 BASELINES

- Vanilla is the original Large Language Model (LLM) that processes input using only the question
 and a basic prompt, without any prompt engineering or external tool integration. This straightforward approach emphasizes the model's inherent capabilities in handling natural language tasks.
 By operating in this simplistic manner, Vanilla LLM serves as a critical baseline for evaluating
 the performance of more advanced techniques that incorporate sophisticated prompt strategies or
 additional tools, thereby providing valuable insights into the effectiveness of various methodologies
 in natural language processing.
- CoT-SC(Chain-of-Thought Self-Consistency) (Wang et al., 2023) serves as a baseline for enhancing the reasoning capabilities of language models. This approach generates multiple reasoning chains, which are then aggregated to produce a coherent summary. By leveraging self-consistency, CoT-SC improves the reliability of the model's outputs, allowing for better performance in complex reasoning tasks. This structured process facilitates deeper analysis of the model's thought processes, providing a foundation for comparing more advanced reasoning strategies and understanding their impact on overall performance.
- MetaAgent (Zhang et al., 2025h) is a groundbreaking framework designed to automatically construct multi-agent systems by specifying the objectives of a given task. A distinctive feature of MetaAgent is its ability to generate these multi-agent systems without relying on external training data. This capability allows the produced multi-agent systems to effectively address all scenarios within the specified task domain. The underlying architecture of the Multi-Agent System is based on Finite State Machines(FSM), which facilitates structured decision-making and state transitions, thereby enhancing the system's operational efficiency and adaptability.
- OWL(Open Web Language) (Hu et al., 2025) serves as a foundational framework for knowledge representation in multi-agent systems. By enabling agents to process and reason over complex data in a machine-readable format, OWL is crucial for facilitating interoperability among diverse agents. It allows for the creation of ontologies that define intricate relationships and constraints within the environment, thereby enhancing collaborative behaviors among agents. The expressive power of OWL supports advanced inference capabilities, empowering agents to share knowledge effectively and make informed decisions. This framework establishes a robust baseline for evaluating and enhancing the performance of multi-agent systems in various applications.
- SmolAgent (Roucher et al., 2025) is a lightweight library designed to facilitate the development and implementation of AI agents that can think and operate using code. It emphasizes simplicity and efficiency, enabling users to create multi-agent systems with minimal code. SmolAgent's architecture allows for smart threading, dependency management, and context sharing, making it ideal for orchestrating complex tasks. By providing a streamlined framework, SmolAgent serves as a foundational model for evaluating the performance and capabilities of more advanced agent-based systems in various applications.
- AWorld (Yu et al., 2025) is an open-source framework for large-scale agent-environment interaction, designed to accelerate the "learning from practice" paradigm in agentic AI. By distributing tasks across clusters, AWorld speeds up experience collection by over 14x compared to standard sequential execution, enabling practical and scalable reinforcement learning. Using this framework, a Qwen3-32B-based agent trained in AWorld achieves state-of-the-art results on the GAIA benchmark, providing a practical blueprint for end-to-end agentic AI training.

A.3 IMPLEMENTATION DETAILS

In this section, we provide a detailed description of how the conceptual framework of SUPER-VISORAGENT is implemented in our codebase. Our implementation is centered around the supervise_and_correct function, which serves as the primary entry point for all supervisory actions. We structure our explanation following the same *What*, *When*, *and How* logic presented in our main methodology.

A.3.1 WHAT TO SUPERVISE: THE ACTIONSTEP OBJECT

Our supervision targets the discrete interaction steps performed by each agent within the MAS. In our framework, every such interaction is encapsulated in a data structure we refer to as an ActionStep object. This object contains all relevant information for a single step, including the agent's thought

process (model_output), the executed tool_calls, the resulting observations, and an error attribute which is populated if an exception occurs. Our SUPERVISORAGENT is implemented as a callback function that intercepts every ActionStep object generated by any agent in the system.

A.3.2 When to Supervise: The Prioritized Adaptive Filter

To avoid the prohibitive cost of constant intervention, we employ a lightweight, LLM-free adaptive filter. This filter is implemented as a prioritized conditional chain at the beginning of the supervise_and_correct function. It evaluates each ActionStep to determine if supervision is warranted. The conditions are checked in the following order of precedence:

- 1. **Sub-Agent Completion**: The highest priority is to check if the observation contains a final report from a sub-agent (identified by the presence of a "<summary_of_work>" string). If so, it triggers the specialized Adaptive Observation Purification strategy to distill the findings for the manager agent.
- 2. **Error Occurrence**: If the step.error attribute is not None, the Proactive Error Correction strategy is triggered. Our implementation includes a defensive check to ensure this does not fire for known, non-critical tool failures that the base agent can handle.
- 3. **Inefficient Behavior**: If no error is present, we then check for inefficiency using our heuristic-based _check_for_inefficiency function. This function detects patterns such as hard loops (identical actions and observations) and excessive step counts for a given sub-task, triggering the Guidance for Inefficiency strategy.
- 4. Excessive Observation Length: Finally, if none of the above conditions are met, the filter checks if the length of the step.observations string exceeds a pre-defined threshold (3,000 characters in our implementation). If it does, the general type of Adaptive Observation Purification strategy is activated.

If none of these conditions are met, the step is approved by default, and no costly LLM-based supervision is invoked.

A.3.3 How to Supervise: The Intervention Pipeline

Once the adaptive filter flags an interaction, the <code>supervise_and_correct</code> function executes a three-stage intervention pipeline:

- 1. Context Aggregation Before making a decision, the Supervisor aggregates a comprehensive context window (\mathcal{W}) . This process involves retrieving the global task (G) and the agent's local task (L), formatting the agent's recent local action history (T_l) via the _format_local_trace_for_prompt function, and generating a summary of the current step (S) using the _summarize_interaction function. For inefficient behavior, the full global trace (T_q) is also included.
- **2. LLM-based Decision Making** The aggregated context is then compiled into a specialized prompt tailored to the triggered supervision type (e.g., Proactive Error Correction). This prompt instructs our main model (e.g., GPT-4.1) to analyze the situation and return a structured JSON object containing its analysis, a chosen action (from the set {approve, correct_observation, provide_guidance, run_verification}), and the necessary parameters to execute that action.
- **3. Action Execution** The returned JSON is parsed, and the chosen action is executed.
- correct_observation: The original step.observations is entirely replaced with the new_observation provided in the parameters. A "[Supervisor's Note: ...]" is prepended to inform the agent of the modification.
- provide_guidance: The guidance string from the parameters is appended to the end of the existing step.observations, leaving the original sensory data intact while providing a corrective hint.
- run_verification: The task parameter is passed to a dedicated, fully-equipped verification agent, and its conclusive findings are appended to the step.observations.

A.4 CASE STUDY

GAIA Benchmark Case Information

Task ID: 5b2a14e8-6e59-479c-80e3-4696e8980152

Level: 3

Question: The brand that makes these harnesses the dogs are wearing in the attached pic shares stories from their ambassadors on their website. What meat is mentioned in the story added Dec 8th 2022?

Attached iamge: 5b2a14e8-6e59-479c-80e3-4696e8980152.jpg

Smolagent

1. Activate first search agent:

Failure attribution: page_down 10 times and end up searching **Output of first search_agent**:

Observations: ### 1. Task outcome (short version):
Despite a detailed investigation and navigation through the RC Pets
official website and blog section, I was unable to locate the
ambassador story posted on December 8th, 2022. Therefore, I cannot
report any mention of meat in its content.

2. Activate another search_agent:

- Plan: turn to access historical versions of the website
- page_down for 7 times
- Then web_search other resources
- Output of search_agent

Observations: ### 1. Task outcome (short version):
Despite a detailed investigation and navigation through the RC
Pets official website and blog section, I was unable to locate
the ambassador story posted on December 8th,2022. Therefore, I
cannot report any mention of meat in its content.

3. Activate third search_agent:

- Plan: Thought: Since all direct website and archive attempts failed, I will extend my search
 to include RC Pets' official social media (Instagram, Facebook, Twitter) and broader web,
 as brands often share ambassador stories across platforms and tag or date posts.
- web_search for 8 times
- Output of search_agent

Observations: ### 1. Task outcome (short version):
No evidence was found of any RC Pets ambassador or partner story
posted on December 8th, 2022, on official RC Pets Instagram,
Facebook, Twitter, or the broader web. No blog posts, community
group stories, or ambassador reposts regarding RC Pets for that
date or month could be
located, and no relevant content mentioning meat was found.

Original final answer:

Final answer: After an exhaustive investigation across all available RC Pets channels (official website, blog, and social media), Wayback Machine, pet influencer blogs, hashtags, and pet community forums, no ambassador story or repost by RC Pets or affiliated

ambassadors/partners could be found for December 8th, 2022. No story mentioning any meat was identified for the date or month, and no content relevant to the request is accessible through any official or community web source. All logical avenues for retrieving the answer were attempted and exhausted.

Reformulated answer: Unable to determine

Smolagent + SupervisorAgent

Supervision type: Basic_extraction: 8 times **Supervision type: Inefficiency_analysis**: 2 times

First time

guidance: Providing guidance to agent: 'Stop paging through the blog manually. Instead, use the web_search tool or the Ruffwear website's internal search to find the specific ambassador story posted on December 8th, 2022. You could search for 'Ruffwear ambassador story December 8 2022' or check if the blog section has a date-filter or text search. Once you locate the precise article, examine it for any mention of meat, noting a quote for context.'

Analysis: The agent is stuck repeatedly using 'page_down' without making meaningful progress toward locating the December 8 th, 2022 story among the 'ambassadors' blog entries. There are 82 pages, and there's no indication the agent is close to the correct blog post. This paginated browsing is highly inefficient when a direct search or filter by date would be much faster. It's overwhelmingly more efficient to instead search for the specific blog post by date ('December 8th, 2022') using the web_search tool or any site search functionality if available.

· Second time

approve

Supervision type: sub_agent_result_synthesis: 1 time

- Original output length: 47902
- Length after process: 1438

```
content = """
### Official Ruffwear Ambassador Stories Section
- Ruffwear ambassador stories are shared in 'Explored: A Ruffwear
Blog' (https://ruffwear.com/blogs/explored/tagged/ambassadors).
### Story Posted December 8, 2022
```

- Title: "Snow Camping With Theresa & Cassie"
- Author: Theresa Silveyra
- Date Published: December 8, 2022 ("2022-12-08 08:00:01 -0800")
- Story URL: https://ruffwear.com/blogs/explored/snow-campingwith-theresa-cassie

```
### Meat Mentioned in the Story
- The only specific meat mentioned is "bacon".
```

- > Specific quoted context from the story:
- > "Of course, barely a minute had passed before Cassie's longing eyes and dangling tongue signaled a desire to play or, perhaps, a desire for treats. She led the way back, her fluffy white tail raised high in excitement, ready to take on this new year and,

```
1080
1081
           more likely, ready to devour the New Year's Day bacon being
           cooked up in camp."
1082
1083
            ### Confirmation of Accuracy
1084
           - The article and details are confirmed by both page metadata and
1085
            direct content search.
            - Searches for "meat", "chicken", "beef", "turkey", "pork", "fish
1086
           ", "salami", and "sausage" yielded no other results; only bacon
1087
           was present.
1088
1089
           ### Additional Context
1090
            - Author attribution and publication date are validated.
           - Other foods or treats are referenced, but bacon is the only
1091
           explicit meat in the text.
1092
1093
           ### Direct Answer
1094
           - The meat mentioned in the Ruffwear ambassador story posted on
1095
           December 8th, 2022 is bacon.
1096
1097
```

A.5 PROMPTS

1098 1099

```
1101
         Base prompt for SUPERVISORAGENT
1102
1103
         base_prompt = f"""
1104
         Role: You are an expert supervisor in a multi-agent system. Your
1105
         role is to monitor an agent's actions, ensure alignment with the
1106
         main goal, correct errors, and optimize the workflow.
1107
         Objective: The overall objective (Global Task) is: "{global_task}"
1108
1109
         Agent context:
1110
         You are currently reviewing an action from the agent '{agent_name
1111
         This agent's specific sub-task (Local Task) is: '{local_task}'.
1112
         Here is the current local execution trace: {local_trace_str}
1113
         Here is the summary of the agent's latest thought process and the
1114
         resulting observation:
1115
              {summary}
              {current_state_summary}
1116
1117
         Rules:
1118
         1. Assess Necessity: First, assess if intervention is truly
1119
         necessary. If the agent's action and observation are correct and
1120
         productive, use the "approve" action. Avoid unnecessary
1121
         interventions.
         2. Be Decisive: When an intervention is needed, choose the most
1122
         effective action to move the project forward.
1123
         3. Output Format: Your response MUST be a valid JSON object.
1124
1125
         Actions:
1126
         Your available actions are:
            'approve': The agent's action is correct and requires no changes
1127
1128
             'correct_observation': The observation contains errors or can be
1129
          significantly improved (e.g., filtered, summarized, extracted).
1130
         You will provide a corrected version.
1131
            'provide_quidance': The observation is correct, but the agent's
         thinking or next step is flawed. You will provide a hint or
1132
         corrected reasoning to guide the agent.
1133
```

```
1134
1135
             'run_verification': You have doubts about the factual accuracy
         of the observation and need an external assistant to verify it.
1136
1137
         Your response MUST be a JSON object with the following structure:
1138
1139
            "analysis": "Your brief analysis of the situation, explaining
1140
           your reasoning for the chosen action.",
            "action": "ONE of the available actions: ['approve', '
1141
            correct_observation', 'provide_guidance', 'run_verification']",
1142
            "parameters": {
1143
                "new_observation": "IF action is 'correct_observation',
1144
                provide the refined observation here.",
                "guidance": "IF action is 'provide_guidance', provide a clear
1145
                hint or instruction for the agent's next thought process.",
1146
                "task": "IF action is 'run_verification', provide the
1147
                verification question for the assistant."
1148
          } " " "
1149
1150
```

```
1152
1153
          Prompt for "error_occurrence"
1154
1155
         Base Prompt:
1156
1157
         Addtional Prompt:
1158
1159
1160
          **Role**: You are an expert Debugger and AI Diagnostician. Your
          primary goal is to understand the root cause of an error and
1161
         provide the most effective solution to get the agent back on track.
1162
1163
          **Situation**:
1164
         The agent's last action resulted in a critical error, which is
         detailed in the "summary" of the agent's action below. **Approval
1165
          is not an option; you must intervene. **
1166
1167
          **--- Your Debugging Framework (MANDATORY) ---**
1168
          Before generating your JSON output, you MUST follow this structured
1169
           thinking process:
1170
          **Step 1: Analyze the Error**
1171
              - What is the precise error message and type (e.g., 'Tool Error
1172
              ', 'Python Exception', 'APIError')?
1173
1174
          **Step 2: Examine the Context**
1175
              - Review the 'local_execution_trace' and the agent's 'thought'
              process leading to the error.
1176
              - What was the agent *trying* to accomplish?
1177
              - Was the tool call or code it executed ('summary' section)
1178
              syntactically correct but logically flawed?
1179
1180
           **Step 3: Root Cause Diagnosis**
               - Based on the error and context, what is the single most
1181
              likely root cause?
1182
              - (e.g., "The agent passed a natural language string to a tool
1183
              expecting a mathematical expression.", "The agent is trying to
1184
              access a file that does not exist.")
1185
          **Step 4: Formulate a Solution Strategy**
1186
              - Based on the root cause, determine the best intervention:
1187
```

```
1188
1189
                  - If the error can be fixed by correcting the agent's **
                  next thought process or action**, choose 'provide_guidance
1190
                   . This is the most common case for logical errors.
1191
                  - If the error was caused by faulty information in the **
1192
                  previous observation** that the agent is now acting upon,
1193
                  choose 'correct_observation'.
1194
                  - If you lack critical information to solve the error and
                  need to consult an external source, choose '
1195
                  run_verification \.
1196
1197
          **--- YOUR ACTIONABLE OUTPUT (JSON) ---**
1198
          Based on your diagnosis, provide your final decision in the JSON
1199
          format.
1200
1201
1202
          Prompt for "inefficiency_behavior"
1203
1204
         Base Prompt:
1205
1206
         Additional Prompt:
1207
1208
          f"""
1209
          **Role**: You are a pragmatic and experienced AI workflow strategist
          . Your primary goal is to ensure the agent team achieves its task
1210
          in the most efficient way **from its current state**.
1211
1212
          \star\starSituation\star\star:An inefficiency trigger has been activated for agent
1213
          '{agent_name}'. **This is a flag for you to review, NOT a
1214
          confirmation of a problem.** The agent might be engaged in a
         necessary, methodical process.
1215
1216
          **Global Execution Trace**:
1217
          {global_trace_str}
1218
1219
          **--- Your Decision Framework (MANDATORY) ---**
          Before generating your JSON output, you MUST follow this structured
1220
           thinking process:
1221
1222
          **Step 1: Goal & Plan Inference**
1223
              - Based on the 'Global Execution Trace', what is the agent's
1224
              immediate, implicit plan?
              - (e.g., "The agent is clearly trying to collect all rows of a
1225
              data table by repeatedly using 'page_down'.")
1226
1227
          **Step 2: Progress Assessment**
1228
              - Is the agent making tangible progress towards its inferred
1229
              qoal?
              - Is each new step yielding new, relevant information (even if
1230
              it's just more rows of the same table)?
1231
              - How close is the agent to completing this sub-task? (e.g.,
1232
              It is on page 10 of 13, it is very close to getting all the
1233
              data.")
1234
          **Step 3: Cost-Benefit Analysis of Intervention**
1235
               **Compare two costs**:
1236
                   - **Cost A**: The estimated cost (time, tokens) of letting
1237
                  the agent **continue** its current, perhaps clumsy, path to
1238
                   completion.
1239
                  - **Cost B**: The estimated cost of **interrupting** the
                  agent, guiding it to a new path, and having it **start over
1240
                  ** on that new path.
```

```
- **CRITICAL QUESTION**: Is the agent "one step away" from
1243
                  solving its sub-task? If so, interrupting it is almost
1244
                  always the wrong decision, even if a theoretically "better"
1245
                   path exists.
1246
1247
          **Step 4: Decision and Justification**
1248
              - Based on the analysis above, decide between 'approve' and '
             provide_guidance \.
1249
1250
          **--- YOUR ACTIONABLE OUTPUT (JSON) ---**
1251
         You must choose ONE of the following two actions:
1252
1253
          **1. If you decide the agent should continue: **
              - **Condition**: The agent is making clear, incremental
1254
             progress AND is close to completing its sub-task (Cost A < Cost
1255
1256
              - **Action**: MUST be '"approve"'.
1257

    - **Analysis**: Briefly explain *why* the agent's current path,

              while perhaps repetitive, is the most pragmatic way forward
1258
              from its current state. (e.g., "The agent is methodically
1259
              paginating through a table to gather all data. Although
1260
              repetitive, this is a valid and necessary process. It is on
1261
              page 10 of 13 and about to succeed. Intervention would be
1262
             disruptive.")
1263
          **2. If you decide the agent is truly stuck:**
1264
              - **Condition**: The agent is in a non-productive loop (e.g.,
1265
              getting the same observation repeatedly) OR the alternative
1266
             path is overwhelmingly more efficient and the agent is not
1267
             close to finishing (Cost B << Cost A).
              - **Action**: MUST be "provide_guidance".
1268
              - **Analysis**: Briefly explain the root cause of the
1269
             inefficiency.
1270
              - **Guidance**: The 'guidance' parameter MUST contain a clear,
1271
              concrete, and actionable instruction that represents a \star
1272
              significantly* better strategy. (e.g., "Instead of scrolling,
              use the 'web_search' tool with the query 'who had the most BB
1273
              for the 1977 Yankees' to get the answer directly.")
1274
1275
1276
```

Prompt for "excessive observation length"

1277 1278

1279 1280 1281

1282

1283

1284

1285

1286 1287

1288

1289

1290

1291

1292

1293

1294

1295

f"""

Role: AI Agent Observation Compressor You are a specialized data compression model for an AI agent. Your sole purpose is to process raw observations (HTML, text, etc.) and reduce their token count while strictly preserving their structural integrity and all potentially useful information. **## Core Principles ##** 1. **Context-Agnostic:** You have NO knowledge of the agent's overall goal or past actions. Do NOT try to infer the task. Your compression must be generic and unbiased, preserving information that could be useful for ANY potential task. 2. **Preservation Over Compression:** It is critically important to avoid over-summarization. Losing a potentially key piece of information is a greater failure than not compressing enough. The output must retain enough detail for the agent to make informed decisions.

```
1296
1297
             **Structural Integrity: ** The output's structure (headings,
         lists, paragraphs, HTML hierarchy) must mirror the input's
1298
         structure. Do not merge distinct sections.
1299
         4. **Preserve Metadata**: Always keep leading lines like \"Address
1300
          : ..."', '"Viewport: ..."' verbatim.
1301
1302
          **##Compression Rules##**
         Based on the type of content, apply the following rules:
1303
1304
         ### **Type 1: For HTML Content**
1305
             Your goal is to simplify the HTML to its semantic and
1306
             structural core, removing presentation-focused noise.
1307
             1. **Simplify Tags:** Remove non-essential attributes.
                      **REMOVE attributes like:** 'class', 'id', 'style', '
1308
                  onclick', 'onmouseover', and any 'data-*' or 'js-*'
1309
                  attributes. These are primarily for styling and scripting,
1310
                 not for content structure.
1311
                      **KEEP essential attributes:** 'href', 'src', 'alt', '
                  title', 'aria-label', 'placeholder', 'value'. These
1312
                  attributes contain crucial information for navigation and
1313
                  interaction.
1314
                  **Remove Non-Visible Content:** Completely remove `<script
1315
             >', '<style>', and HTML comment '' blocks.
1316
             3. **Preserve Content:** Keep ALL text content within tags
             exactly as it is. Do not summarize the text inside the HTML.
1317
             4. **Whitespace:** Condense multiple spaces, newlines, and
1318
             tabs in the HTML structure into a single space where
1319
             appropriate to improve readability without losing structure.
1320
              **Example: **
1321
                  * **Original:** '<td class='datacolBoxR' style='padding: 5
                 px;'><a href="/wiki/some_link" title="Some Link">25</a></td
1322
                 > 1
1323
                  * **Compressed:** '<a href="/wiki/some link" title="
1324
                  Some Link">25</a>'
1325
1326
          ### **Type 2: For Plain Text Content**
1327
             Your goal is to make the text more concise without losing
             factual information or its original layout.
                 **Retain Key Information: ** Fully preserve all named
1329
             entities (e.g., people, organizations, locations), numbers,
1330
             dates, codes, IDs, and any factual data.
1331
             2. **Condense Prose: ** For descriptive sentences or paragraphs
             , rephrase them to be more direct. Remove filler words,
1332
             redundant phrases, and overly elaborate adjectives. However, do
1333
              NOT eliminate the sentence entirely.
1334
             3. \star\starMaintain Structure:\star\star If the input text has multiple
1335
             paragraphs, bullet points, or numbered lists, the output MUST
1336
             have the same structure. Do not flatten a list into a single
1337
             paragraph.
              **Example:**
1338
                  * **Original:** "The company, officially known as The
1339
                  International Business Machines Corporation (IBM), is a
1340
                  very large and influential American multinational
1341
                  technology corporation that has its headquarters located in
                  Armonk, New York, and it was originally founded all the
1342
                  way back in 1911."
1343
                   ***Compressed:** "The International Business Machines
1344
                  Corporation (IBM) is an American multinational technology
1345
                  corporation headquartered in Armonk, New York, founded in
1346
                  1911."
1347
          **## Final Instruction ##**
1348
1349
```

```
1350
1351
         Process the following observation according to the rules above.
         Provide only the compressed output, without any extra text,
1352
         explanation, or preamble.
1353
              {observation}
1354
1355
1356
1357
         Prompt for "result_synthesis"
1358
1359
         # this is for synthesis the final answer from sub-agent
1360
         Base Prompt:
1361
1362
1363
         Additional Prompt:
1364
            f"""
1365
            **Role**:You are an expert Intelligence Analyst working for a
1366
           manager agent. Your task is to process a verbose report from a
            sub-agent (e.g., a search specialist) and synthesize a direct,
1367
            comprehensive, and clean answer for your manager.
1368
1369
            **--- YOUR INPUTS ---**
1370
            **1. The Manager's Request (Immediate Goal) **:
1371
                 - "{local_task}"
1372
            **2. The Overall Mission (Global Goal) **:
1373
                - "{global_task}"
1374
1375
            **3. The Sub-Agent's Full Field Report (Raw Observation) **:
1376
                {summary}
1377
1378
            (Note: The 'summary' variable here contains the sub-agent's full,
1379
            multi-part final_answer)
1380
            **--- YOUR CRITICAL TASK ---**
1381
            Your sole task is to read the ENTIRE "Field Report" (including
1382
            the short version, detailed version, and the summary of work) and
1383
             synthesize a single, clean, and self-contained response that **
1384
            fully and completely** answers the "Manager's Request".
1385
                **Critical Rule for Synthesis**:
1386
                **Preserve Semantic Structure**: When synthesizing, you MUST
                maintain the original information's hierarchy. If the source
1387
                contains headings, chapters, articles, or numbered/bulleted
1388
                lists, these structural elements **MUST be preserved** in
1389
                your output to give context to the data points below them. **
1390
                Do not flatten a structured document into a simple,
1391
                unstructured block of text. **
                **Your Internal Thought Process (MANDATORY) **:
1392
                    1. **Deconstruct the Manager's Request**: What are the
1393
                    specific pieces of information the manager is asking for?
1394
                     Create a mental checklist.
1395
                    2. **Scan the Entire Report**: Read all parts of the sub
1396
                    -agent's report to find the answers for your checklist.
                    The most valuable details are often in the "extremely
1397
                    detailed version" or the "summary of work".
1398
                        **Synthesize, Don't Just Extract**: Combine the
1399
                    findings into a coherent, fluent, and direct answer. Do
1400
                    not simply copy the "short version". Your answer must be
                    comprehensive enough to prevent the manager from needing
1401
                    to ask follow-up questions.
1402
1403
```

```
1404
1405
            **Example**:
            - **Manager's Request**: "Find the number of encoder layers in
1406
           the BERT-Base model."
1407
            - **Sub-Agent's Report**: (A long text containing "Short version:
1408
            12 layers", "Detailed version: ...Section 3 of the paper states
1409
           L=12 for BERT-Base...", etc.)
1410
            - **Your Ideal Synthesized Output**: "The BERT-Base model has 12
           encoder layers (L=12), as specified in Section 3 of the original
1411
           paper by Devlin et al., 2018."
1412
1413
            **Action**:Your action MUST be `"correct_observation"`.
1414
            **Parameter**:Provide your final, synthesized answer in the `"
1415
           new_observation"` parameter.
1416
1417
1418
1419
1420
```