Dean&Francis

ISSN 2959-6157

Reliability Challenges of LLM Agents: Systemic Failure Causes and Governance

Xinru Tong[1],

Lyujie Wei[2*],

Ziyi Yan[3]

[1]School of Computer and Information Engineering, China Three Gorges University, Yichang, Hubei, China
[2]The University of Sydney, Sydney, New South Wales, Australia
[3]School of Artificial Intelligence and Computer Science, Wuhan University of Engineering Science, Wuhan, Hubei, China
*Corresponding author: weilyujie1@gmail.com

Abstract:

As Large Language Model (LLM) agents advance toward industrial applications, their systemic reliability challenges become critical obstacles. This review comprehensively analyzes these issues to establish a framework for trustworthy agents, first categorizing key systemic failures into four types: Cognitive-level, Decision-level, Execution-level, and Security & Boundary failures. It then identifies five root causes: training data limitations, model architecture/algorithm defects, imperfect alignment and reinforcement learning, brittle tool/environment interactions, and vulnerabilities in knowledge representation/update. To address these, the paper proposes a dual-track strategy with a multi-level governance system—emphasizing pre-deployment benchmarks, post-deployment monitoring, and a macro-framework covering multi-agent coordination, interpretability, traceability, responsibility attribution, and full lifecycle risk management—alongside intrinsic enhancement of model factuality, reasoning, alignment, and architecture. This paper aims to provide a systematic perspective and structured approach for building more reliable and responsible LLM agents. It promotes the integration of theoretical safety and practical deployment, driving the advancement of LLM agents from the experimental to the industrial level.

Keywords: LLM Agents; Reliability; Systemic Failure.

1. Introduction

Currently, Large Language Model (LLM)-based agents are capable of an operational mode that combines linguistic reasoning with interaction. In other words, they can formulate, maintain, and adjust action plans through dynamic reasoning, while simultaneously interacting with the external environment and integrating additional information into the reasoning process [1]. The industrialization and reliable application of LLM agents have also become a key development direction at present; however, issues of systemic failures—such as factual hallucinations and logical fallacies—pose critical obstacles to their advancement.

Existing relevant research mainly focuses on two

major directions: innovation in agent architecture and innovation in safety detection. Most of these studies concentrate on improving a single performance indicator or repairing local defects, lacking a comprehensive collation and classification of the systemic failure modes of LLM agents, as well as a systematic exploration of the underlying causes behind these failures. This results in fragmented relevant solutions with insufficient targeting.

In view of this, this paper, by classifying, tracing the origins of, and integrating the systemic failure phenomena of LLM agents, provides a clear problem framework for academic research in this field, and further improves research efficiency and the quality of achievement transformation. This paper will focus on three key points: sorting out and classifying the typical systemic failure phenomena of LLM agents in practice, analyzing the underlying causes of these failures, and exploring strategies to mitigate or eliminate failures while constructing a multi-level governance system.

The core innovations of this paper are reflected in the following two aspects: first, it is the first to systematically categorize four major types of systemic failures and identify six underlying causes; second, it proposes a dual-track strategy of "internal capacity enhancement and macro-level governance framework." This strategy integrates quantitative pre-deployment assessment, post-deployment dynamic monitoring, and lifecycle risk management.

2. Phenomena of Systemic Failures

Systemic failures in LLM agents refer to persistent functional abnormalities triggered by the coupling of multiple causative factors in complex industrial environments, which cannot be resolved through adjustments to local components. Such failures cause substantial harm to the accuracy of output results, data security, and the completion level of core tasks.

2.1 Cognitive-level Failures

Cognitive-level failures refer to functional abnormalities where outputs deviate from reality or contain logical contradictions due to deficiencies in knowledge, information integration, or factual judgment capabilities. This category includes factual hallucinations and knowledge conflicts. Factual hallucination refers to the phenomenon where LLM agents generate content that seems plausible but is actually fabricated or contradicts provided information. These can range from minor inaccuracies to completely invented facts. Knowledge conflict occurs when the agent's internal knowledge (acquired from training data) conflicts with externally provided information (e.g., user instructions, tool outputs), leading to inconsistent or erroneous outputs. It places greater emphasis on the cognitive level, highlighting the contradictions inherent in the knowledge itself.

2.2 Decision-level Failures

Decision-level failures are functional abnormalities occurring during the formulation of action plans and execution of decisions for complex tasks, caused by deficiencies in planning capability, strategy adjustment mechanisms, or goal perception, leading to deviation from the task objective or falling into ineffective loops. Task deviation refers to the agent's output contradicting the goal of the task or not fully accomplishing the task as intended. It focuses more on the relevance between the output and the objective. It may be indirectly caused by knowledge conflicts or by other factors, representing a higher-level error. Entering a loop or deadlock describes the phenomenon where, when attempting to solve a multi-step problem, the agent may repeatedly execute the same sequence of actions, failing to recognize the strategy's failure or to consider alternative paths [1, 2].

2.3 Execution-level Failures

Execution-level failures are functional abnormalities occurring when LLM agents translate planned decisions into concrete actions such as tool calls or command execution. Caused by deviations in action generation, poor environmental interaction, or system coordination faults, they lead to task interruption, distorted results, or triggering cascading disruptions. This type of failure primarily manifests as API call errors and parameter format errors. When an LLM calls external tools (e.g., search engines, databases), it needs to generate structured requests that comply with interface specifications. However, in practice, issues like missing fields, parameter mismatches, or non-standard formats often cause the call to fail or return empty results. In database retrieval and interactive tasks, this "request format mismatch → call failure" is particularly common. It represents a typical manifestation of conversion errors between language generation and program execution, reflecting the current lack of stability and standardization in the tool interface invocation phase of agents [3].

2.4 Failures at the Security and Boundary Levels

In deployment, failures also arise at the boundary of safety and stability. Under adversarial or ambiguous prompts, models that otherwise appear well aligned may still produce hate speech, discriminatory remarks, or unsafe recommendations, indicating that guardrails do not generalize beyond the tuning distribution [2]. Extended dialogue

ISSN 2959-6157

introduces privacy risk: through stepwise elicitation, an attacker can reconstruct prior conversation content or surface strings memorized during pretraining (for example, names and addresses), revealing weak memory isolation and data handling safeguards [4]. A further concern is sporadic instability. With inputs held constant, the system sometimes emits irrelevant or internally incoherent responses that resist deterministic reproduction. Although infrequent, such episodes undermine trust where decisions must be auditable and consistent.

3. Causes of Systemic Failures

3.1 Limitations of Training Data

What the model can assert with confidence is bounded by corpus coverage, curation quality, and update regularity. Despite multi stage preprocessing, heterogeneous sources import semantic conflict that reemerges as knowledge inconsistency under user input or external grounding. Duplication and formatting noise persist through scale tolerant filters, reinforcing shallow heuristics and inflating overconfidence. Standard filtering pipelines remain vulnerable to adversarial phrasing and multilingual evasion, allowing toxic and unsafe material to survive and reappear through sophisticated prompting [2]. Batch based data aggregation lacks timestamp indexing and semantic versioning, which leads to temporal collisions and factual drift as old and new claims coexist without resolution. Without explicit causal grounding, the model cannot reliably resolve contradictions or infer temporal precedence. Additional training on downstream data provides only coarse grained correction and often induces parameter interference, amplifying the risk of catastrophic forgetting. Privacy risks persist due to insufficient deidentification; pattern based anonymization fails to neutralize indirect identifiers or context linked leakage, allowing multi turn reconstruction of memorized personal data [4]. These failures reflect fundamental limitations in current data governance paradigms, which prioritize scale and coverage over semantic structure and provenance. Without a shift toward modular, timestamped, and causally consistent knowledge integration, model reliability will remain constrained by the instability of its training substrate.

This paper believe this is one of the most basic and difficult problems in LLMs. If the data is wrong, outdated, or unclear, the model will make mistakes no matter how good the training is. These problems cannot be fixed easily by fine-tuning or adding external tools. To solve this, people need better data collection, with clear time labels, fewer conflicts, and stronger quality checks.

3.2 Model Architecture and Algorithmic Constraints

Several limits are structural rather than incidental. The learning objective rewards plausibility rather than truth; when evidence is thin, hallucination persists. Autoregressive decoding affords little global search or principled backtracking, so multistep tasks can stall in loops or dead ends even when remedies are nearby [3]. As contexts lengthen, attention diffuses and position encodings saturate, leading to instruction forgetting and goal drift. Sampling introduces low probability but high impact deviations that produce incoherence and resist exact reproduction. Prompting strategies such as Chain of Thought help on some tasks, but gains are uneven across domains, which suggests limited intrinsic algorithmic organization [5].

This problem comes from how the model itself works. Autoregressive models generate one token at a time and cannot easily check their own logic. This makes it hard for them to plan or correct mistakes. Even new prompting methods like Chain-of-Thought help only in some cases. This paper think these problems are part of the model's structure and cannot be fully solved without changing the core architecture.

3.3 Imperfect Alignment and Reinforcement Learning from Human Feedback

Alignment improves typical behavior but does not reliably control rare or adversarial cases. The underlying training objective optimizes for human preference rather than factual accuracy, which causes the model to reinforce plausible but incorrect patterns when feedback signals are ambiguous or inconsistent [6, 2]. Reinforcement learning from human feedback selects outputs that appear helpful or agreeable, even if they contain subtle errors, leading to smoother and more confident responses that amplify hallucination under uncertainty. This preference optimization introduces a semantic drift from truth supervision, especially when training data lacks clear verification signals or annotator consensus. Refusal policies are often miscalibrated, blocking harmless requests while failing to reject adversarial inputs that exploit prompt ambiguity or role play scenarios [6]. This reflects a lack of robust generalization in the learned safety boundaries, which rely on narrow distributions seen during alignment fine tuning. Overemphasis on helpfulness expands the response space and increases vulnerability to task shift, while strict filtering suppresses clarification seeking behavior that is essential for complex multi step reasoning [6]. Residual susceptibility to indirect prompt injection and scenario manipulation persists even after safety tuning, exposing

blind spots in adversarial coverage and response calibration [2]. These outcomes point to a structural limitation in current alignment frameworks, which substitute preference conformity for epistemic grounding. Without a supervision signal anchored in objective correctness and adversarial completeness, alignment may distort model behavior in high risk or ambiguous settings.

Alignment methods like Reinforcement Learning from Human Feedback (RLHF) focus on making answers sound helpful or polite, not always correct. This can cause the model to give wrong but confident answers. Also, the model may refuse safe questions or fail to reject dangerous ones. This paper argues that this is a serious issue in current training objectives. Fixing it needs better reward design that cares more about truth, not just human preference.

3.4 Interaction with Complex Tools and Environments

Many failures emerge when natural language plans must drive executable actions. Mapping intent to an API call is brittle, with omitted fields, type mismatches, and malformed payloads, especially in retrieval, database access, and web automation [1, 2]. Long or highly structured returns (logs, tables, nested JSON) are misread, with wrong fields extracted and errors propagated downstream [1, 2]. In the absence of validation, retries, and sanity checks, local faults cascade; public agent benchmarks repeatedly expose this brittleness [1, 2]. When tool outputs diverge from expectation, the system often fails to diagnose causes or explore alternatives, and it either loops or halts prematurely [3, 2]. Tool mediation also enlarges the privacy surface: unminimized sensitive content may be forwarded to untrusted services or echoed later, underscoring the need for isolation by default and strict data minimization [4].

Tool use failures are common, but they are not as deep as data or model problems. Most issues come from format errors, bad outputs, or missing checks. These can be improved with better design, like clear API rules, error handling, and retry steps. So people see this as a fixable problem with engineering work.

3.5 Vulnerability of Prompt Engineering and In-Context Learning

Prompt Engineering (PE) and In-Context Learning (ICL) establish the foundational behavioral control for LLM agents [7]. However, their core nature is that of a textual specification mechanism, rather than a hard-coded logic with formal verification. This design inherent to language models means the control layer lacks formal robustness guarantees, making the system's reliability critically dependent on the quality and structure of the textual input. Consequently, the agent exhibits high sensitivity and inherent brittleness to prompt ambiguity, contradictory exemplars, or structural deficiencies in the instructions. This specification dependency directly gives rise to several systemic failure modes: 1. Vague instructions frequently mislead the model's interpretation of the core task's objective function. During multi-step execution, this often results in the agent prioritizing a statistically safe output sequence over the precise user intent, leading to Task Deviation or Goal Drift. 2. When Few-Shot Examples (FSEs) contain conflicting information, ICL forces the activation of competing knowledge subsets, inducing severe Knowledge Conflict and prompt-induced Contextual Hallucination [8]. 3. In multi-step reasoning frameworks such as Chain-of-Thought (CoT) or ReAct, suboptimal prompt structure—particularly the lack of clear structural guidance or stop conditions—causes the model to erroneously repeat actions, resulting in the agent entering an action-reasoning deadlock or Looping. 4. For tool-using agents, the LLM's mechanism for generating structured API calls is highly brittle. Any imprecision in the tool schema definition can lead to a Malformed API Call (e.g., parameter mismatch), consequently triggering a Runtime Error and forcing task failure.

3.6 Defects in Knowledge Representation and Update Mechanisms

The core constraint on LLM reliability originates from its knowledge storage paradigm: knowledge is implicitly encoded as billions of parameter weights, existing primarily as statistical associations rather than structured, symbolic logic [9]. This inherent representation scheme and the subsequent parameter-level updates are the deep-seated root causes of systemic failures.

This statistical grounding, rather than Causal Logic, is the mechanism behind Factual Hallucination. The model selects the most "fluent" sequence continuation based on probability, which, in complex or low-frequency knowledge domains, may be objectively incorrect. Lacking a verifiable world model, the model confidently asserts erroneous information. This knowledge base is further destabilized by updates. Since knowledge in the parameter space is highly shared, introducing new information via fine-tuning is an intrinsically destructive process where new and old knowledge structures compete, leading to Catastrophic Forgetting. This results in Knowledge Conflict where the model outputs contradictory answers on issues involving temporal evolution, demonstrating extreme cognitive instability. Lastly, the knowledge boundaries are

ISSN 2959-6157

intrinsically fuzzy. For low-frequency Marginal Knowledge or rare queries, the associated parameter weights are weak. In this low-confidence region, the model fails to compensate for the knowledge gap through logical deduction, and its output degenerates into a more random sequence prediction, leading to semantically illogical and Unpredictable Anomalous Behavior that severely elevates reliability risks in specialized applications.

4. Reliability Enhancement and Systemic Governance Strategies

Enhancing the reliability of LLM agents requires a dual-track strategy involving the intrinsic reinforcement of technical capabilities and the establishment of systemic governance through macro-level frameworks. An effective governance structure must cover the entire lifecycle of the agent, from design to operation and maintenance.

4.1 Quantitative Assessment and Continuous Monitoring

4.1.1 Pre-Deployment static assessment

This article aims to conduct a static assessment before deployment, establishing a reliability baseline through two key methods and systematically identifying the weaknesses of the system. The first step is system benchmarking, which should not merely focus on simple success rates like AgentBench and WebArena tests, but rather define granular reliability metrics. When evaluating, it is necessary to track the execution efficiency, which includes steps, latency, and severe misclassification, etc. Faults should also be divided into known patterns, such as execution loops or API format errors. This can obtain an objective and repeatable snapshot that is consistent with the specific fault axis of the agent. The second method is structured adversarial testing, which uses the "red team" approach to highlight safety boundaries. The key techniques involved include fuzzing instructions to assess their rigidity and bias sensitivity, evaluating the resilience of knowledge conflicts based on contradictory injection, and detecting tool patterns. This is done to identify vulnerabilities in the API call generation process and ensure that the system has strong robustness against runtime errors.

4.1.2 Post-Deployment dynamic monitoring

This article mentions that dynamic monitoring after deployment focuses more on continuously tracking the behavior of agents based on the baselines established in actual scenarios. This is different from static testing. At this stage, granularity metrics are used to monitor the efficiency and stability of agent execution at runtime. The monitoring system needs to detect deviations in real time and classify them. Map faults such as different delays or unexpected loops to specific error patterns. The active detection mechanism needs to remain effective all the time so as to identify newly emerged vulnerabilities. By leveraging structured input, it can be verified whether the agent can still maintain the boundaries of its security and reliability in the continuous interaction with external tools and users.

4.2 Intrinsic Reliability Enhancement Techniques

Intrinsic enhancement techniques are designed to reinforce the core capabilities of the model and agent architecture at their root, directly lowering the probability of failure.

4.2.1 Strengthening model core capabilities

Strengthening core capabilities requires navigating the trade-off between performance gains and computational overhead. Knowledge Augmentation connects agents to external data via Retrieval-Augmented Generation (RAG) to mitigate hallucinations, or employs Knowledge Editing for parametric corrections. However, RAG introduces latency and retrieval risks (e.g., garbage-in-garbage-out), while Knowledge Editing faces scalability issues and catastrophic forgetting. To improve Reasoning and Planning, strategies like Chain-of-Thought (CoT) or Tree-of-Thought (ToT) structure reasoning paths. Yet, these substantially increase inference costs without formally guaranteeing reliability, leaving them prone to error propagation. Finally, Alignment seeks to harmonize model behavior with human values, a task complicated by the subjectivity of "values" and the risk of over-alignment leading to excessive caution or refusal bias.

4.2.2 Optimization of agent architecture

Architectural optimization seeks robustness through modularity but introduces complexity in component interaction. Incorporating Memory Mechanisms manages context limitations but adds latency and consistency challenges, creating risks of contextual hallucination where irrelevant memories are retrieved. Self-Reflection and Correction Loops enable self-critique to improve decision quality but incur high computational costs. Moreover, their effectiveness is bounded by the model's reasoning capacity; a fundamental failure often renders self-correction fruitless. Finally, Robust Tool Learning aims to stabilize external interactions. While validation mechanisms enhance reliability, they increase complexity, and autonomous tool learning remains prone to errors due to the inherent brittleness of structured output generation.

5. Conclusion

This paper provided a comprehensive systematic analysis of the reliability challenges encountered by LLM agents as they progress toward industrial application. The study initially categorized the typical systemic failure phenomena observed in practice into four core types: Cognitive-level Failures (Factual Hallucination and Knowledge Conflict), Decision-level Failures (Task Deviation and Deadlocks/Loops), Execution-level Failures (API Call Errors and Result Misinterpretation), and Security and Boundary Failures (Harmful Content Generation and Privacy Leakage).

By tracing the origins of these failures, this work revealed a multi-layered set of deep-seated causes. These underlying factors stem from the inherent limitations of Training Data and Model Architecture, the imperfections of Alignment and RLHF, the brittleness of Interaction with Complex Tools and Environments, and the fundamental Vulnerability of Prompt Engineering and the structural Defects in Knowledge Representation and Update Mechanisms (including the statistical nature of knowledge and catastrophic forgetting).

To counter these systemic challenges, a multi-tiered, integrated Governance System was proposed. This system advocates for enhancing reliability through two parallel strategies:

Intrinsic Reliability Enhancement: Strengthening the core capabilities of the model and architecture by improving Knowledge and Factuality via Retrieval-Augmented Generation (RAG) and Knowledge Editing, enhancing Reasoning and Planning through advanced prompting (Chain-of-Thought/Tree-of-Thought), deepening Alignment and Values, and optimizing the agent architecture with Memory Mechanisms, Self-Reflection Loops, and Robust Tool Calling.

Systemic Governance Frameworks: Establishing a macro-level framework that includes Quantitative Assessment and Continuous Monitoring (Pre-Deployment Benchmarking and Post-Deployment Drift Monitoring); promoting Coordination and Alignment of Multi-Agent Systems; implementing Interpretability, Traceability, and Responsibility Attribution to ensure transparency and accountability; and embedding Lifecycle Risk Management by making Impact Assessment and the design of mitigation strategies mandatory from the initial design phase.

Although this study relies primarily on theoretical analysis and literature integration, and the proposed governance framework requires further empirical validation in diverse industrial scenarios, it establishes a critical foundation. However, this research offers a systematic analytical perspective and a comprehensive governance roadmap for constructing more reliable, trustworthy, and socially responsible LLM agents, addressing a critical gap in the current fragmented literature.

6. Authors Contribution

All the authors contributed equally and their names were listed in alphabetical order.

References

[1] Yao S, Zhao J, Yu D, Du N, Shafran I, Narasimhan K R, Cao Y. React: Synergizing reasoning and acting in language models. In: The eleventh international conference on learning representations, 2022, October.

[2] Zhou S, Xu F F, Zhu H, Zhou X, Lo R, Sridhar A, Neubig G. Webarena: A realistic web environment for building autonomous agents. arXiv preprint, 2023.

[3] Liu X, Yu H, Zhang H, Xu Y, Lei X, Lai H, Tang J. Agentbench: Evaluating llms as agents. arXiv preprint, 2023.

[4] Chu J, Sha Z, Backes M, Zhang Y. Reconstruct your previous conversations! Comprehensively investigating privacy leakage risks in conversations with GPT models. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2024.

[5] Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, Chi E, Le Q V, Zhou D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. Advances in Neural Information Processing Systems, 2022, 31: 24124-24137.

[6] Ouyang L, Wu J, Jiang X, Almeida D, Wainwright C L, Mishkin P, Zhang C, Agarwal S, Slama K, Ray A, Schulman J, Hilton J, Kelton F, Miller L, Simens M, Askell A, Welinder P, Christiano P, Leike J, Lowe R. Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems, 2022, 31: 27730-27744.

[7] Lester B, Al-Rfou R, Constant N. The power of scale for parameter-efficient prompt tuning. arXiv preprint, 2021.

[8] Min S, Lyu X, Holtzman A, Artetxe M, Lewis M, Hajishirzi H, Zettlemoyer L. Rethinking the role of demonstrations: What makes in-context learning work?. arXiv preprint, 2022.

[9] Petroni F, Rocktäschel T, Riedel S, Lewis P, Bakhtin A, Wu Y, Miller A. Language models as knowledge bases?. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), 2019: 2463-2473.

[10] Micheli V, Alonso E, Fleuret F. Transformers are sample-efficient world models. arXiv preprint, 2022.

[11] Titzer B L. Whose baseline compiler is it anyway?. In: 2024 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), 2024: 207-220. IEEE.