Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Self-Reflection and Self-Correction in Agentic AI: A Critical Review of Feedback, Verification, Memory, and Failure Modes

Most Anamika Akter Zedni
Rajshahi University of Engineering and Technology
Md Monzurul Islam
Texas State University
Amith Khandakar
amitk@qu.edu.qa
Qatar University
Md Minhazul Islam
Rajshahi University of Engineering and Technology
Mohamed Arselene Ayari
Qatar University

Research Article

Keywords: Agentic AI, Self-correction, Verification, Feedback mechanisms, Autonomous agents, Memory integrity

Posted Date: November 19th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11144434/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Self-Reflection and Self-Correction in Agentic AI: A Critical Review of Feedback, Verification, Memory, and Failure Modes

Most Anamika Akter Zedni[1], Md Monzurul Islam[2], Amith Khandakar[3*], Md Minhazul Islam[1], Mohamed Arselene Ayari[3]

[1]Department of Computer Science & Engineering, Rajshahi University of Engineering and Technology, Rajshahi, Bangladesh.
[2]Department of Civil Engineering, Texas State University, Texas, USA.
[3*]College of Engineering, Qatar University, Doha, Qatar.

*Corresponding author(s). E-mail(s): amitk@qu.edu.qa;
Contributing authors: anamikazedni@gmail.com; monzurul@txstate.edu; minannu2001@gmail.com; ArslanA@qu.edu.qa;

Abstract

Agentic artificial intelligence transforms self-correction from a localized answer-revision problem into a system-level reliability and recovery challenge. In autonomous, stateful, and long-horizon systems, early errors can propagate through reasoning, planning, retrieval, tool use, external actions, multi-agent interaction, and persistent memory. However, existing research often treats self-correction as either response-level refinement or a capability within broader agent architectures. This study develops an agent-level perspective that connects these fragmented views. It distinguishes reasoning, reflection, verification, correction, and adaptation, emphasizing that reasoning or critique alone does not constitute self-correction unless evaluative information produces a subsequent behavioral change. The study introduces a unified corrective lifecycle covering anomaly detection, error diagnosis and localization, corrective evidence acquisition, repair, validation, commitment or rollback, memory updating, and continuation, termination, or escalation. Existing approaches are organized across five dimensions: correction target, feedback source, temporal scope, persistence, and control architecture. A comparative synthesis shows that self-correction is not uniformly beneficial: repeated reflection can improve performance, stagnate, or degrade initially correct behavior. More reliable gains generally arise when feedback

is task-relevant, informative, and at least partially independent of the process that produced the error. The study also identifies key failure modes, including missed errors, unsuccessful repair, over-correction, error amplification, memory contamination, and objective misalignment. These findings motivate research on calibrated intervention, memory integrity, verification, rollback, cost-aware correction, human escalation, and resilience under distribution shift.

Keywords: Agentic AI; Self-correction; Verification; Feedback mechanisms; Autonomous agents; Memory integrity

1 Introduction

1.1 Motivation: Why Agentic Systems Create a Different Correction Problem

Artificial intelligence has undergone a fundamental transition from passive, single-turn language models toward autonomous, stateful agentic systems (Plaat et al. 2025a; Wei et al. 2026). Conventional Large Language Models (LLMs) operate primarily in a one-shot inference paradigm, mapping a given prompt to a static text output without persistent context, interactive capabilities, or environmental decision rights. In this standard setting, an error such as a factual hallucination, logical flaw, or stylistic misalignment remains localized to that specific response. Once generation terminates, the error does not inherently propagate into subsequent external systems or alter downstream decision states. In contrast, agentic AI architectures deploy foundation models as central cognitive engines embedded within closed-loop execution environments (Yao et al. 2022; Du et al. 2026a). Operating over long time horizons, these systems pursue complex, high-level goals through continuous cycles of perception, deliberative planning, tool invocation, environment interaction, and memory maintenance.

In an agentic workflow, the presence of state persistence, delegated autonomy, and sequential decision-making fundamentally transforms the nature of system failures. Rather than remaining isolated, an initial mistake in an agent's reasoning trajectory triggers cascading failures across interconnected system components:

Failure Propagation Across Agentic Workflows

- **Reasoning and Planning:** An ungrounded assumption or miscalculated sub-goal at an early stage corrupts hierarchical plan decomposition, leading the agent into sub-optimal or dead-end reasoning trajectories (Wang et al. 2023a; Yao et al. 2023a).
- **Retrieval and Knowledge Grounding:** A flawed retrieval query or incorrect intermediate inference causes the agent to fetch irrelevant or misleading external evidence, reinforcing initial hallucinations and creating self-perpetuating error loops (Yu et al. 2024; Asl et al. 2025a).
- **Tool Invocation and Code Execution:** Errors in API parameter synthesis or code generation result in failed tool calls, runtime exceptions, or corrupted data structures, directly impairing downstream execution steps (Gou et al. 2024; Olausson et al. 2024a).
- **Environment Interaction:** Malformed external actions modify digital or physical environments in unintended or irreversible ways, causing security vulnerabilities, resource waste, or operational disruptions (Shukla et al. 2025).
- **Persistent Memory:** Storing incorrect reflections, faulty plan traces, or unverified task outcomes in episodic memory buffers contaminates long-term knowledge stores, leading to persistent error propagation across subsequent task sessions (Shinn et al. 2023).
- **Multi-Agent Coordination:** In collaborative ecosystems, an uncorrected error by a single agent spreads through inter-agent communication channels, inducing collusive failure, groupthink, or coordination breakdown across the entire network (Liang et al. 2024; Li et al. 2025).

Because subsequent decisions depend directly on the cumulative execution history, a single uncorrected misstep early in a multi-turn trajectory compounds exponentially, steering the system toward catastrophic task divergence.

These cascading failure modes demonstrate that reliability in agentic AI cannot be treated merely as a prompt-engineering or text-level revision problem. In one-shot generation, post-hoc refinement typically involves re-prompting a model to rephrase or rewrite an isolated answer (Madaan et al. 2023). In an agentic system, however, resolving an error requires a system-level operational recovery: the system must monitor an ongoing trajectory, detect anomalies from noisy environment observations or execution logs, localize the root cause of failure across complex action histories, acquire corrective evidence, roll back or re-plan multi-step actions, and sanitize persistent memory stores. Consequently, agentic self-correction is a system-level reliability and recovery problem, not merely a prompting or answer-revision problem.

As agentic systems are increasingly entrusted with autonomous software development, scientific discovery, and high-stakes operational decision-making, ensuring their dependability becomes paramount (Jiang et al. 2026; Wei et al. 2026). Unchecked intrinsic generation frequently fails to self-correct without structured, feedback-driven

verification mechanisms (Li 2025). There is thus a critical imperative to develop self-corrective control mechanisms that operate dynamically during execution enabling agents to continually monitor their internal reasoning and environmental feedback, evaluate potential state deviations, and autonomously execute targeted repairs or rollbacks before errors propagate beyond recovery.

1.2 Existing Surveys and Research Gap

The rapid evolution of LLMs and autonomous agents has produced several surveys examining self-correction, multi-step reasoning, agent architectures, optimization, and system safety. However, the literature approaches corrective capabilities from two largely distinct perspectives. Dedicated surveys of automated feedback and self-correction, such as Pan et al. (Pan et al. 2024) and Kamoi et al. (Kamoi et al. 2024), provide detailed analyses of feedback sources, correction strategies, and empirical limitations. Their emphasis, however, remains primarily on model-level or response-level correction. In contrast, broader surveys of LLM-based autonomous agents (Wang et al. 2024b; Xi et al. 2025; Chowa et al. 2026) and agent optimization (Du et al. 2026a) examine planning, tool use, memory, multi-agent coordination, and long-horizon behavior, but generally treat reflection and correction as individual capabilities rather than as a complete operational recovery process.

To compare these perspectives systematically, Table 1 evaluates prior surveys across seven dimensions. C1 represents whether self-reflection or self-correction is treated as a central research focus. C2 captures whether correction extends beyond answer revision to agent-level plans, actions, tool calls, retrieval processes, trajectories, or memory. C3 represents systematic treatment of feedback and verification mechanisms, including intrinsic evaluation, external tools, retrieved evidence, execution signals, critics, or human feedback. C4 captures the role of memory and persistence, including the storage, retrieval, updating, or propagation of corrective experience. C5 represents empirical analysis of the conditions under which correction succeeds or fails. C6 captures failure modes and safety concerns, including over-correction, hallucinated critique, memory contamination, error amplification, and security degradation. Finally, C7 represents whether a survey addresses self-correction as an end-to-end lifecycle connecting error detection, diagnosis, evidence acquisition, verification, repair, validation, rollback or commitment, memory updating, stopping, and escalation.

In Table 1, ✓ denotes substantial or core coverage, △ denotes partial or secondary coverage, and × indicates that a dimension is not substantively addressed.

The comparison reveals a consistent division in the existing literature. Surveys centered on self-correction provide strong coverage of feedback and verification mechanisms (C3) and the empirical conditions under which correction succeeds or fails (C5), but generally provide limited treatment of persistent memory (C4) and agent-level trajectory recovery (C2). Conversely, surveys of LLM agents and agent optimization provide broader coverage of agent-level behavior (C2) and memory architectures (C4), while self-correction is typically treated as one capability within a larger architecture rather than as an end-to-end recovery process. The most persistent gap therefore concerns C7: the integration of detection, diagnosis, verification, repair, validation, rollback, memory update, and escalation into a unified corrective lifecycle. This paper

Table 1: Comparative analysis of existing surveys related to LLM self-correction and agentic AI.

Survey Paper	C1	C2	C3	C4	C5	C6	C7
Pan et al. (Pan et al. 2024)	✓	△	✓	×	✓	△	△
Kamoi et al. (Kamoi et al. 2024)	✓	△	✓	×	✓	△	△
Wang et al. (Wang et al. 2024b)	△	✓	△	✓	△	△	×
Xi et al. (Xi et al. 2025)	△	✓	△	✓	△	△	×
Chowa et al. (Chowa et al. 2026)	△	✓	△	✓	△	△	×
Du et al. (Du et al. 2026a)	△	✓	✓	△	△	△	×
Plaat et al. (Plaat et al. 2025b)	△	△	✓	×	✓	△	×
Sun et al. (Sun et al. 2025)	△	△	✓	×	✓	△	×
Raza et al. (Raza et al. 2026)	△	✓	✓	△	△	✓	△
This Review	✓	✓	✓	✓	✓	✓	✓

addresses that gap by treating self-correction as a system-level recovery mechanism operating across stateful and long-horizon agentic workflows.

1.3 Scope and Operational Definition

To establish a rigorous foundation for this review, we delineate the conceptual boundaries and inclusion criteria governing agentic self-correction. Within autonomous systems research, the term *self-correction* is frequently conflated with adjacent capabilities such as multi-step reasoning, iterative sampling, or passive error logging. To resolve this ambiguity, we adopt the following operational criterion as the defining conceptual basis of this review:

Working Definition

A process is treated as self-corrective when information about a previous output, reasoning state, action, trajectory, or memory is evaluated and subsequently used to modify current or future agent behavior.

This criterion establishes that self-correction is fundamentally a closed-loop control process requiring both an evaluative phase and an execution phase. Consequently, reflection or error diagnosis alone does not constitute self-correction unless the resulting evaluation signal directly influences, guides, or alters subsequent agent decisions or behavioral updates.

Self-corrective processes operate across multiple architectural targets within stateful agent workflows (Plaat et al. 2025a; Wei et al. 2026). At the *output level*, agents revise generated text, mathematical solutions, or code blocks in response to diagnostic feedback (Madaan et al. 2023; Olausson et al. 2024b). At the *reasoning level*, agents inspect intermediate thoughts to backtrack, prune invalid branches, or repair logical fallacies (Yao et al. 2023a; Sun et al. 2025). At the *plan and trajectory level*, agents redecompose complex sub-goals or alter execution paths when encountering dead-ends

(Wang et al. 2023a; Yao et al. 2022). At the *tool and action level*, agents synthesize revised API parameters or adjust environment interactions based on execution errors or runtime exceptions (Gou et al. 2024). At the *retrieval level*, agents evaluate information gaps to reformulate queries or verify fetched evidence (Yu et al. 2024; Asl et al. 2025a). Finally, at the *memory level*, agents update persistent episodic buffers or sanitize stored reflections to prevent error compounding across multi-turn horizons (Shinn et al. 2023).

To maintain thematic focus, this survey explicitly distinguishes agentic self-correction from related but conceptually distinct mechanisms:

Scope Boundaries

- **Chain-of-Thought Prompting vs. Self-Correction:** Linear Chain-of-Thought (CoT) generation (Wei et al. 2022) externalizes intermediate reasoning steps but lacks an explicit evaluation signal. CoT is categorized as an enabling precursor (*proto-reflective reasoning*) rather than self-correction, as it generates reasoning traces without verifying or modifying prior steps based on diagnostic feedback.
- **Repeated Sampling and Majority Voting vs. Self-Correction:** Unguided re-sampling, parallel rollout generation, and majority-voting self-consistency (Wang et al. 2023b; Ji et al. 2026a) explore alternative solution paths without evaluating prior errors. Because these methods generate independent samples rather than revising existing state based on feedback, they are excluded from the scope of self-correction.
- **Generic Agent Optimization vs. Self-Correction:** Broad framework optimizations, prompt tuning, or static workflow topologies that enhance baseline task execution without incorporating an explicit, feedback-driven corrective loop (Du et al. 2026a) fall outside this review.
- **Training-Time Optimization vs. Self-Correction:** Parameter-level learning methods (e.g., supervised fine-tuning or reinforcement learning) are included only when an identifiable mechanism of critique, automated verification, or post-hoc feedback is directly embedded within the training pipeline or data-generation loop (Bai et al. 2022; Pan et al. 2024).

Importantly, this scope is not restricted by the origin of the feedback signal or the system control architecture. A process qualifies as agentic self-correction regardless of whether the evaluative signal originates from intrinsic self-critique, external tool execution (e.g., code compilers or formal verifiers), retrieved knowledge, environmental observations, peer agents in multi-agent debate, or human-in-the-loop oversight (Kamoi et al. 2024; Liang et al. 2024). By anchoring our review on this operational definition, we ensure a comprehensive synthesis of mechanisms that enable autonomous agents to monitor, verify, and recover from failures during long-horizon execution.

1.4 Research Questions

While Table 1 establishes the need for this review by identifying structural gaps in existing surveys, the present review addresses these gaps through four research questions:

Research Questions

RQ1. What constitutes self-reflection and self-correction at the agent level?

RQ2. What sources of feedback and verification enable reliable correction?

RQ3. How do correction target, temporal scope, memory, and control architecture affect corrective behavior?

RQ4. Under what conditions does self-correction improve performance, fail, or actively degrade it?

These questions operationalize the literature gaps identified in Table 1. RQ1 addresses the conceptual boundaries of agent-level reflection and correction, corresponding primarily to C1 and C2. RQ2 examines the feedback and verification mechanisms represented by C3. RQ3 addresses correction targets, temporal scope, memory, and control architecture, corresponding to C2, C4, and C7. RQ4 focuses on empirical boundary conditions, failure modes, and safety implications, corresponding to C5 and C6. Together, the four questions structure the subsequent critical synthesis of agentic self-correction.

1.5 Contributions

To address the literature gaps identified in Table 1 and systematically address Research Questions 1 through 4 (RQ1 to RQ4), this review consolidates the study of agentic self-correction into four primary contributions:

First, this paper establishes a conceptual foundation that distinguishes agent-level self-correction from intermediate reasoning generation, single-turn answer refinement, and parameter-free agent optimization (RQ1). By defining self-correction as a closed-loop control process wherein evaluative information directly modifies subsequent agent behavior, this review clarifies the functional boundaries between passive self-reflection and actionable recovery across stateful, long-horizon execution environments.

Second, this review develops a unified framework for the agent-level correction lifecycle (RQ2 and RQ3). Extending beyond simplified generate, critique, and refine paradigms, the framework synthesizes ten interconnected operational stages: anomaly detection, error diagnosis and localization, corrective evidence acquisition, verification, plan or action repair, repair validation, state commitment or rollback, persistent memory updating, trajectory continuation or termination, and human escalation. This framework systematically organizes corrective strategies across diverse targets, feedback origins, temporal horizons, and control topologies.

Third, this review synthesizes empirical findings regarding the conditions under which self-correction enhances task performance, plateaus, or becomes counterproductive (RQ4). Through a critical comparative analysis across mathematical, coding,

retrieval, and multi-agent domains, the review evaluates how feedback signal quality, external verifier grounding, error depth, memory persistence, and inference-time computational overhead govern failure recovery.

Fourth, this review derives a structured research agenda to guide future developments in resilient agentic control. Anchored in identified failure modes and system constraints, the agenda prioritizes key challenges including verifier calibration, verification without ground-truth oracles, long-term memory integrity, adaptive reflection policies, cost-aware correction, robustness under distribution shifts, and formal human escalation protocols.

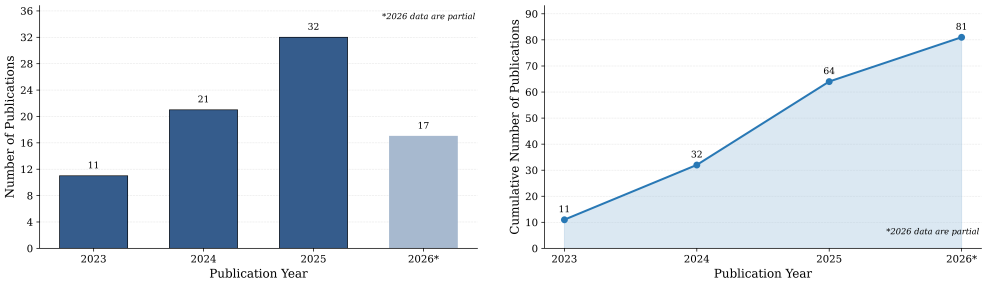
2 Review Methodology and Scope

2.1 Literature Search, Eligibility, and Synthesis

The literature search was conducted using five complementary scholarly discovery and bibliographic resources: *Google Scholar*, *Semantic Scholar*, *Crossref*, *arXiv*, and *DBLP*. Because terminology surrounding self-reflection and self-correction remains heterogeneous, the search strategy combined three families of terms: (i) corrective processes, including *self-reflection*, *self-correction*, *self-refinement*, *self-repair*, *critique*, *verification*, and *self-evaluation*; (ii) agentic-system terms, including *LLM agent*, *language agent*, *autonomous agent*, and *agentic AI*; and (iii) supporting mechanisms, including *memory*, *tool use*, *retrieval*, *verifier*, *feedback*, *multi-agent critique*, and *self-improvement*. Terms from these groups were combined to identify both explicitly named self-correction methods and broader agentic systems in which reflection, evaluation, verification, or recovery formed part of the control architecture.

A study was considered within scope when it examined an LLM or LLM-based agent and included an identifiable process in which information about a previous output, reasoning state, plan, retrieval result, tool call, action, trajectory, or memory was evaluated and subsequently used to modify behavior. This operational definition distinguishes self-correction from reasoning generation alone. For example, chain-of-thought generation was not treated as self-correction unless the generated reasoning was subsequently evaluated and used to revise an output, reasoning path, plan, or action. Studies were retained when they made empirical or conceptual contributions to reflection, feedback, verification, critique, repair, validation, memory-based adaptation, or related corrective mechanisms. Peer-reviewed journal and conference papers were prioritized, while recent preprints were included when they represented relevant emerging work for which archival versions were not yet available. Generic prompting studies without an explicit corrective process, unrelated reinforcement-learning or self-supervision studies, traditional agents without relevant LLM-based correction, and papers using "reflection" only in a descriptive or metaphorical sense were excluded. Duplicate preprint and published versions were consolidated, with the archival version preferred where available.

Candidate studies were initially screened using titles, abstracts, keywords, and stated methodological contributions, followed by full-text assessment where relevance could not be established reliably from bibliographic information alone. Reference lists



(a) Annual number of publications included in the corpus.

(b) Cumulative number of publications.

Fig. 1: Growth of the reviewed literature on self-correction in LLM-based and agentic AI systems from 2023 to 2026. The 2026 values represent a partial publication year and should therefore not be interpreted as final annual publication counts.

and citation links of highly relevant studies were also examined to identify conceptually important work that did not explicitly use the term "self-correction." The retained literature was synthesized according to common dimensions including correction target, feedback source, verification mechanism, temporal scope, memory and persistence, control architecture, application domain, reported performance effects, computational implications, and documented failure conditions. Because substantial heterogeneity exists across models, benchmarks, feedback mechanisms, evaluation protocols, and computational budgets, the review uses comparative qualitative synthesis rather than statistical meta-analysis. The primary bibliometric characterization covers 2023–2026, while earlier foundational studies are retained where conceptually necessary. Publication counts for 2026 are treated as partial because the observation year was incomplete at the time of the review.

2.2 Characteristics of the Reviewed Literature

The contemporary corpus contains **81 publications** published between 2023 and the partial 2026 observation period. Figure 1 shows both annual and cumulative publication growth. The annual number of identified publications increased from 11 in 2023 to 21 in 2024 and 32 in 2025. Consequently, the cumulative corpus expanded from 11 publications in 2023 to 32 in 2024 and 64 by the end of 2025. An additional 17 publications from 2026 were identified at the time of corpus construction, increasing the cumulative total to 81. Because the 2026 count represents only a partial year, its lower annual value should not be interpreted as evidence of decreasing research activity. The 2023–2025 trend instead indicates substantial growth in research concerned with reflection, verification, and corrective behavior in LLM-based systems.

The corpus also exhibits substantial diversity in authorship and research themes. As shown in Fig. 2(a), the distribution of team size is right-skewed, with a median of five authors per publication and a mean of 6.3. The difference between the mean

and median is driven by a small number of large collaborative studies involving substantially more authors. Most studies, however, are produced by relatively small or medium-sized research teams.

Figure 2(b) further indicates that the reviewed literature is not strongly concentrated around a small number of individual researchers. The most frequently represented authors—Shunyu Yao, Karthik Narasimhan, and Aviral Kumar—each appear in three publications in the corpus, while several other researchers appear twice. These counts should be interpreted as corpus representation rather than author productivity or scholarly impact because the review was constructed according to topical relevance rather than exhaustive author-level bibliometrics.

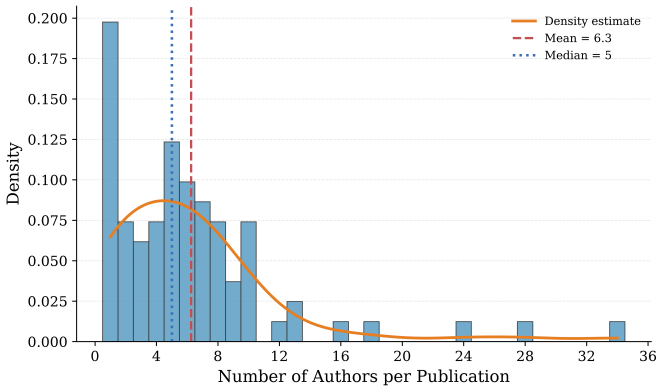
The thematic frequencies in Fig. 2(c) provide an additional descriptive indication of the breadth of the reviewed literature. Among the most recurrent normalized terms are *chain of thought*, *code*, and *self correction*, followed by *test time*, *memory*, *multi agent*, *debate*, and *retrieval augmented generation*. Terms related to reinforcement learning, security, self-reflection, refinement, and optimization also occur across the corpus. These frequencies are not interpreted as measures of scientific importance or methodological superiority; rather, they demonstrate that contemporary self-correction research spans multiple components of agentic systems, including reasoning, executable feedback, retrieval, memory, multi-agent interaction, and test-time adaptation. This diversity motivates the agent-level taxonomy developed in the following sections.

3 Conceptual Foundations of Self-Correction in Agentic AI

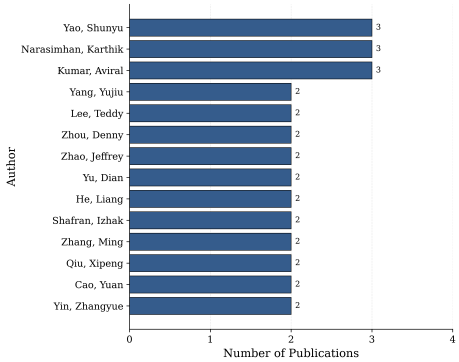
3.1 From Language Models to Agents

The transition from conventional Large Language Models (LLMs) to autonomous agentic architectures represents a fundamental paradigm shift in artificial intelligence, moving from static text generation to stateful, goal-directed operational control (Plaat et al. 2025a; Wei et al. 2026). A conventional or "bare" LLM functions primarily as an ungrounded probability distribution over token sequences. Operating within a single-pass or autoregressive generation paradigm, a bare LLM maps an input prompt directly to an output response without persistent internal state across sessions, without autonomous goal management, and without an interface to execute actions in external physical or digital environments (Chowa et al. 2026; Du et al. 2026a). While techniques such as Chain-of-Thought prompting enhance inference-time deliberation by externalizing intermediate reasoning steps (Wei et al. 2022), the core model remains a passive sequence predictor bound to a static context window.

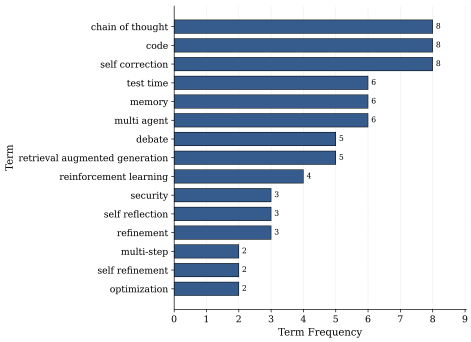
In contrast, an agentic architecture embeds the foundation model as a central cognitive decision core within a broader interactive system (Wang et al. 2024b; Xi et al. 2025). As conceptualized across modern frameworks, an agentic system augments the core language model with explicit functional modules: hierarchical goal perception and planning, multi-tiered memory buffers (comprising short-term working context and long-term vector stores), retrieval mechanisms, and external action interfaces for



(a) Distribution of authors per publication.



(b) Most frequently represented authors.



(c) Most frequent thematic terms.

Fig. 2: Descriptive characteristics of the reviewed literature: (a) distribution of research-team size, (b) authors most frequently represented in the reviewed corpus, and (c) recurrent thematic terms. Author and term frequencies describe the composition of the review corpus and should not be interpreted as measures of scholarly impact or research importance.

tool invocation and environment manipulation (Yao et al. 2022; Shinn et al. 2023; Yu et al. 2024). Operating through iterative perceive-plan-act-observe control loops ($\pi_0 \rightarrow a_1 \rightarrow o_1 \rightarrow r_1 \cdots \rightarrow a_T$), an agentic architecture scales test-time interaction, allowing the system to dynamically adjust its trajectory based on real-world observations and execution feedback (Wei et al. 2026; Gou et al. 2024).

This architectural transformation fundamentally alters the unit and scope of self-correction. In a conventional LLM, correction is localized and output-centric. It targets discrete tokens, factual statements, or single-turn mathematical rationales within an isolated text response (Pan et al. 2024; Kamoi et al. 2024; Madaan et al. 2023). In an agentic system, however, actions, observations, and state representations are

tightly coupled across extended execution histories. Consequently, failures are rarely confined to surface-level language flaws. Instead, errors manifest as trajectory-level structural breakdowns, including sub-goal drift, API parameter synthesis failures, compiler exceptions, ungrounded retrieval integration, or memory contamination (Shinn et al. 2023; Olausson et al. 2024a; Shukla et al. 2025).

Because a single uncorrected misstep in an early step compounds across subsequent state transitions, self-correction in an agentic setting cannot be reduced to prompt re-phrasing or post-hoc text editing. Instead, corrective control must operate across multiple structural targets within the system architecture:

Structural Targets of Self-Correction

- **Reasoning and Planning:** Re-evaluating intermediate thoughts, pruning unviable search branches, and re-decomposing failing sub-goals (Yao et al. 2023a; Wang et al. 2023a).
- **Tool Use and Execution:** Diagnosing runtime exceptions, re-synthesizing API call parameters, and verifying code execution outcomes (Gou et al. 2024; Olausson et al. 2024a).
- **Retrieval and Grounding:** Re-formulating search queries, filtering irrelevant evidence, and verifying information consistency against external knowledge sources (Yu et al. 2024; Asl et al. 2025a).
- **Action and Environment Interaction:** Rolling back unviable state changes, executing corrective maneuvers, and managing physical or digital environmental feedback (Yao et al. 2022).
- **Memory and Persistence:** Sanitizing corrupted episodic buffers, removing false reflections, and updating persistent long-term knowledge to prevent cross-session error propagation (Shinn et al. 2023).

As summarized in Table 2, comparing a bare LLM with an agentic architecture highlights why agentic self-correction must be treated as a system-level recovery problem. Whereas a conventional model corrects a static answer, an agentic system must detect anomalies across noisy execution logs, localize root causes within multi-turn histories, validate candidate repairs, execute state rollbacks, and update persistent memory. Self-correction thus evolves from an answer-refinement technique into an essential system-level control mechanism for operational reliability and fault recovery in stateful AI systems.

3.2 Reasoning, Reflection, Verification, Correction, and Adaptation

A precise distinction among reasoning, reflection, verification, correction, and adaptation is necessary for analyzing self-corrective behavior in agentic AI. These terms are often used interchangeably, although they represent different functions within a

Table 2: Architectural comparison between a conventional LLM and an agentic AI architecture.

Dimension	Conventional / Bare LLM	Agentic AI Architecture
Goal structure	Prompt-bound generation without autonomous goal management	Goal-directed operation with task decomposition and sub-goal management
State persistence	Primarily limited to the current input and context window	Persistent state maintained across steps or episodes through working, episodic, or semantic memory
Planning and search	Primarily sequential generation or prompted reasoning	Multi-step planning, search, replanning, and adaptive control across a trajectory
Memory access	Information available mainly through the immediate context	Dynamic retrieval, storage, and updating of short-term or persistent memory
Tool access	No inherent external action interface; tool use requires an external orchestration layer	Selection and invocation of external tools, APIs, retrieval systems, or executable environments
Observations	Prompt and contextual information supplied to the model	Iterative observations obtained from tools, environments, execution results, or other agents
Actions	Generation of model outputs	External actions such as retrieval, tool invocation, code execution, or environment interaction
Temporal horizon	Typically a single generation or short interaction	Multi-step and potentially long-horizon trajectories across extended workflows
Correction target	Primarily outputs or individual reasoning traces	Outputs, reasoning, plans, retrieval decisions, tool calls, actions, trajectories, and memory

corrective process. This paper treats them as analytically separable stages that may interact as follows:

$$\text{Reasoning} \rightarrow \text{Reflection} \rightarrow \text{Verification} \rightarrow \text{Correction} \rightarrow \text{Adaptation}. \tag{1}$$

This sequence represents a conceptual decomposition rather than a mandatory execution order. Particular systems may combine stages, omit explicit reflection, perform verification before generating a critique, or execute a correction without preserving the resulting information beyond the current interaction.

Reasoning refers to the generation and transformation of intermediate representations or candidate solution paths toward a task objective. In LLM-based systems, this may involve explicit reasoning traces, decomposition of a problem into intermediate steps, or search over alternative solution paths, as illustrated by Chain-of-Thought prompting and Tree of Thoughts (Wei et al. 2022; Yao et al. 2023a). Reasoning itself

does not imply evaluation of previous states. A model may generate an extended reasoning trajectory without determining whether earlier steps are valid.

Reflection refers to the evaluation of a previous output, reasoning state, action, plan, trajectory, or memory item with the purpose of identifying deficiencies or opportunities for revision. Reflection therefore changes the direction of computation from primarily generating a candidate solution to examining an already produced state. Self-Refine, for example, introduces explicit feedback on an initial response before refinement, whereas Reflexion converts task feedback into verbal reflections that can inform subsequent attempts (Madaan et al. 2023; Shinn et al. 2023). Reflection can identify a suspected failure, but the existence of a critique does not establish that the critique is correct.

Verification concerns the assessment of a candidate output, reasoning step, action, or reflective diagnosis against an evaluative signal or criterion. The strength of this signal varies substantially across systems. Verification may rely on intrinsic consistency checks, execution results, unit tests, external tools, retrieved evidence, environmental observations, critic models, other agents, or human feedback (Gou et al. 2024; Olausson et al. 2024a). External signals can provide information unavailable to the original generation process, whereas intrinsic verification may remain constrained by the same model limitations that produced the initial error. Verification should therefore not be equated with access to ground truth; it provides evidence whose reliability depends on the verifier, tool, environment, or evaluation procedure.

Correction occurs when information produced through reflection, verification, or environmental feedback is used to modify the agent's behavior or internal state. The target of correction may be a final answer, reasoning step, plan, retrieval query, tool call, external action, trajectory, or memory entry. Self-Refine performs localized response revision based on generated feedback, while tool-grounded and code-repair systems can revise outputs in response to externally observable failures (Madaan et al. 2023; Gou et al. 2024; Olausson et al. 2024a). Correction is therefore defined by behavioral modification rather than by the generation of critique alone.

Adaptation refers to the persistence of corrective information beyond the immediate repair. A corrected state becomes adaptive when information derived from previous success or failure influences later decisions, episodes, or tasks. This persistence may occur through episodic or reflective memory, updates to stored procedures or policies, or training-time parameter modification. Reflexion illustrates non-parametric adaptation by storing verbal reflections that can influence subsequent trials (Shinn et al. 2023), while broader automated-correction frameworks also include training-time mechanisms through which corrective feedback produces persistent model changes (Pan et al. 2024).

Table 3 summarizes these distinctions. The boundaries are important because the presence of one capability does not imply the presence of the next. In particular, not every reasoning mechanism is reflective. Standard Chain-of-Thought prompting generates intermediate reasoning steps but does not inherently evaluate those steps or diagnose errors (Wei et al. 2022). Reasoning traces may make intermediate states more accessible for subsequent evaluation, but this alone is insufficient to classify the process as reflection.

Similarly, not every reflection produces successful correction. A reflective critique may identify the wrong failure, overlook the causal source of an error, or recommend a modification that degrades an initially correct response. Critical evaluations of LLM self-correction show that models may struggle to improve their outputs when reliable external feedback is unavailable (Kamoi et al. 2024). This distinction separates the ability to produce a plausible critique from the ability to recover from an actual failure.

Verification and correction are also separable. An agent may correctly determine that an output is invalid yet fail to localize or repair the underlying error. Conversely, a system may revise an output without reliable verification and thereby introduce a new error. Evidence from code self-repair similarly indicates that the usefulness of iterative correction depends strongly on the quality of the available feedback and evaluation signal (Olausson et al. 2024a). Consequently, successful agentic self-correction cannot be inferred from the presence of reflection alone. It requires a corrective process in which evaluative information is sufficiently reliable to guide an appropriate behavioral change, with adaptation occurring only when the resulting information is retained for future use.

3.3 CoT and Deliberate Reasoning as Precursors

Reasoning mechanisms provide an important foundation for self-reflective and self-corrective behavior, but reasoning alone should not be treated as correction. The distinction is functional: Chain-of-Thought (CoT) exposes intermediate reasoning, self-consistency diversifies candidate reasoning paths, Tree of Thoughts (ToT) searches among alternatives, and verifier-guided or test-time methods evaluate or select candidate solutions. These mechanisms become part of a corrective process only when an evaluative signal influences subsequent reasoning, action, or state.

Chain-of-Thought prompting is a foundational precursor to reflective control because it elicits explicit intermediate rationales before a final response (Wei et al. 2022). By making intermediate steps available in textual form, CoT creates an observable artifact that can subsequently be inspected, compared, verified, or revised. This property makes it useful for downstream error localization and corrective reasoning. However, producing a rationale does not itself constitute reflection. A CoT trace may contain invalid assumptions, arithmetic errors, unsupported inferences, or reasoning that does not faithfully represent the factors responsible for the model's prediction. Recent work further shows that CoT performance can deteriorate substantially under distribution shifts, reinforcing the distinction between producing a plausible reasoning trace and reliably validating that trace (Zhao et al. 2026). CoT is therefore better understood as an enabling representation for reflection rather than a self-corrective mechanism in its own right.

Self-consistency extends CoT by generating multiple independent reasoning paths and aggregating their final predictions (Wang et al. 2023b). This strategy increases test-time computation and reduces dependence on a single trajectory. Conceptually, however, self-consistency provides an agreement signal rather than an independent correctness signal. A majority answer can still be incorrect when sampled trajectories share systematic biases or common misconceptions. Consequently, self-consistency diversifies reasoning and offers a useful confidence proxy, but it does not necessarily

Table 3: Conceptual distinctions among reasoning, reflection, verification, correction, and adaptation.

Concept	Primary Function	Evaluation or Behavioral Outcome	Representative Mechanisms
Reasoning	Generates or searches candidate solution paths toward a task objective	Processes task context and intermediate states to produce reasoning traces, sub-goals, or candidate solutions	Chain-of-Thought (Wei et al. 2022); Tree of Thoughts (Yao et al. 2023a)
Reflection	Evaluates previously generated outputs, states, actions, or trajectories	Produces critiques, diagnoses, or identified weaknesses that may motivate subsequent revision	Self-Refine (Madaan et al. 2023); Reflexion (Shinn et al. 2023)
Verification	Assesses a candidate or diagnosis against an evaluative signal or criterion	Produces evidence such as execution results, consistency signals, scores, retrieved support, or validation outcomes	CRITIC (Gou et al. 2024); unit-test and execution feedback (Olausson et al. 2024a)
Correction	Modifies agent behavior or state in response to evaluative information	Produces a revised output, repaired reasoning process, modified plan or action, or updated state	Self-Refine (Madaan et al. 2023); tool-grounded correction (Gou et al. 2024)
Adaptation	Preserves corrective information so that it can influence future behavior	Produces persistent changes in memory, procedures, policies, or model parameters	Reflexion memory (Shinn et al. 2023); training-time correction (Pan et al. 2024)

identify why an answer is wrong or specify how the underlying reasoning should be repaired.

Tree of Thoughts introduces more explicit deliberation by representing intermediate reasoning as a structured search over candidate "thoughts" (Yao et al. 2023a). Rather than committing to a single left-to-right trajectory, ToT generates alternative intermediate states, evaluates their promise, and explores the search space using procedures such as breadth-first or depth-first search. This structure permits lookahead, pruning, and backtracking when a candidate trajectory appears unproductive. On the Game of 24 benchmark, Yao et al. reported that GPT-4 with standard CoT solved 4% of tasks, whereas the reported ToT configuration achieved 74% (Yao et al. 2023a).

This result illustrates the potential value of deliberate search when early reasoning choices strongly constrain later outcomes. Nevertheless, ToT is better interpreted as search with state evaluation than as a general self-correction mechanism. It becomes corrective when information obtained from evaluating a failed or weak branch is used to redirect subsequent reasoning.

Verifier-guided and test-time-compute methods extend this principle by allocating additional inference resources to candidate generation, scoring, search, or intermediate-step evaluation. Process supervision, for example, evaluates intermediate reasoning steps rather than relying only on final-answer correctness (Lightman et al. 2024), while broader test-time-compute frameworks combine repeated sampling, search, verification, and in some cases iterative revision (Ji et al. 2026b). These approaches strengthen the evaluative component of reasoning, but evaluation and correction remain conceptually distinct. Ranking candidates with a verifier selects among alternatives already generated, whereas correction requires evaluative information to modify the subsequent reasoning process, regenerate a failed step, revise a plan, alter a tool interaction, or otherwise change agent behavior.

Latent reasoning introduces a related challenge. Instead of expressing every intermediate step as natural-language tokens, latent-CoT methods perform part or all of the reasoning process in continuous internal representations (Chen et al. 2025). Such approaches may reduce the dependence on explicit token-by-token reasoning and offer efficiency advantages in some architectures, but they also reduce direct access to intermediate reasoning states. This creates an important tension for self-correction: representations that are not explicitly verbalized may be harder to inspect, localize, and validate using conventional textual critics or external verifiers. Latent reasoning is therefore relevant to this review primarily because it changes the observability of the states on which reflective and corrective mechanisms depend.

Taken together, these methods establish the computational substrate on which self-correction can operate. CoT exposes candidate reasoning, self-consistency introduces diversity, ToT enables structured search and backtracking, and verifier-guided inference adds explicit evaluation. None of these properties alone guarantees correction. A reasoning mechanism becomes genuinely corrective only when evaluation of an existing state produces information that is subsequently used to modify the agent's reasoning, action, trajectory, or persistent state.

3.4 Levels and Targets of Correction

Section 3.2 distinguishes the functional stages involved in corrective control, including reflection, verification, correction, and adaptation. A complementary distinction concerns the *target* at which correction is applied. In agentic systems, recovery may operate on a final output, an intermediate reasoning process, an external action, a multi-step plan, persistent memory, or the underlying policy that governs future behavior. These targets form an expanding hierarchy from local and short-lived repair toward broader and more persistent behavioral change.

Output-level correction modifies the final generated artifact, such as a textual response or code output, without necessarily changing the reasoning or execution process that produced it. Iterative refinement methods such as Self-Refine primarily

operate at this level when feedback is used to revise an existing response (Madaan et al. 2023). **Reasoning-level correction** intervenes earlier by modifying intermediate rationales, derivations, or candidate reasoning paths. Search-based approaches such as Tree of Thoughts illustrate how weak reasoning branches can be evaluated, abandoned, or replaced before a final answer is produced (Yao et al. 2023a).

Action- or tool-level correction targets interactions with external systems, including tool selection, retrieval queries, API parameters, code execution, or other environment-facing actions. Tool-grounded approaches such as CRITIC use external feedback to identify and revise such failures (Gou et al. 2024). **Plan- or trajectory-level correction** operates at a broader temporal scale by modifying subgoal sequences, replanning after failed actions, backtracking from unproductive states, or redirecting an ongoing workflow. This level is particularly important in agentic systems because a locally valid action can still contribute to an unsuccessful overall trajectory (Yao et al. 2023b).

Memory-level correction concerns information that persists across steps or episodes. Stored reflections, episodic experiences, facts, or procedures may require revision, validation, replacement, or removal when they are inaccurate or stale. Reflexion demonstrates how reflective information can influence later trials through episodic memory, while also illustrating why the quality of stored corrective information matters (Shinn et al. 2023). Finally, **policy- or learning-level correction** produces persistent changes to the decision process itself through mechanisms such as fine-tuning, preference optimization, or reinforcement learning from corrective feedback (Bai et al. 2022; Pan et al. 2024).

These levels should not be interpreted as a rigid sequence. A single failure may trigger correction at several targets simultaneously. For example, a failed tool call may require parameter revision, trajectory replanning, and an update to persistent memory. The hierarchy therefore characterizes the *scope and persistence of the correction target*, rather than a mandatory order of execution.

4 Unified Taxonomy of Agentic Self-Correction

4.1 Correction Lifecycle

Self-correction in stateful agentic systems extends beyond the conventional generate–critique–refine formulation commonly used for single-turn output revision (Madaan et al. 2023; Pan et al. 2024). Section 3.2 distinguishes the functional concepts involved in corrective control, while Section 3.4 identifies the structural targets at which correction may occur. The present subsection instead characterizes the operational lifecycle through which an agent detects a failure, investigates its cause, acquires corrective evidence, applies a repair, validates the resulting state, and determines how execution should proceed.

A generalized agentic self-correction lifecycle can be represented as shown in Fig. 3.

The lifecycle begins with a **Task / State** that defines the active goal, contextual information, and relevant system state. During **Generate / Act**, the agent produces a response, reasoning step, tool invocation, retrieval action, or environmental action, after which **Observe Outcome** captures the corresponding result, such as an API

Table 4: Levels and targets of correction in agentic AI.

Level	Correction Target	Typical Corrective Operation	Persistence
Output	Final text, code, or generated artifact	Rewrite, edit, or regenerate the final output	Local / turn-level
Reasoning	Intermediate rationales, derivations, or candidate paths	Revise a reasoning step, prune a branch, or regenerate an intermediate state	Working context
Action / Tool	Tool choice, retrieval query, API call, code execution, or environment action	Modify parameters, select another tool, reformulate a query, or retry execution	Step-level
Plan / Trajectory	Sub-goals, plans, or multi-step execution trajectories	Replan, backtrack, switch branches, or recover from a failed sub-goal	Episode-level
Memory	Stored reflections, experiences, facts, or procedures	Validate, revise, replace, delete, or update stored information	Cross-episode
Policy / Learning	Model parameters, preferences, or learned decision policies	Fine-tune, optimize preferences, or update the learned policy	Persistent

response, execution trace, retrieved document, environmental observation, or runtime log (Yao et al. 2023b; Gou et al. 2024; Olausson et al. 2024a). **Detect Anomaly** then determines whether the observed state deviates sufficiently from an expected or acceptable outcome to warrant intervention. Such detection may rely on execution failures, verifier scores, consistency checks, task-specific assertions, or other evaluative signals (Kamoi et al. 2024; Li 2026).

Detection does not, however, establish the cause of failure. **Diagnose / Localize Error** attempts to identify the component or decision responsible for the anomaly, such as an invalid reasoning step, incorrect tool parameter, unsupported assumption, or failed sub-goal (Li 2026). Once the suspected failure has been localized, **Acquire Corrective Evidence** gathers information that can support repair. Depending on the system, this may involve additional retrieval, external tools, execution feedback, documentation, or other grounding sources (Yu et al. 2024; Asl et al. 2025b). **Repair / Revise** then converts diagnostic and evidential information into a candidate modification, such as revising an output, altering a tool call, repairing code, or restructuring part of a plan (Madaan et al. 2023; Zhang et al. 2023).

A candidate repair should remain distinct from a validated repair. **Validate Repaired State** evaluates whether the proposed change resolves the original problem and whether additional failures have been introduced. Validation may rely on unit tests, execution feedback, process-level evaluation, or other verifier signals (Lightman

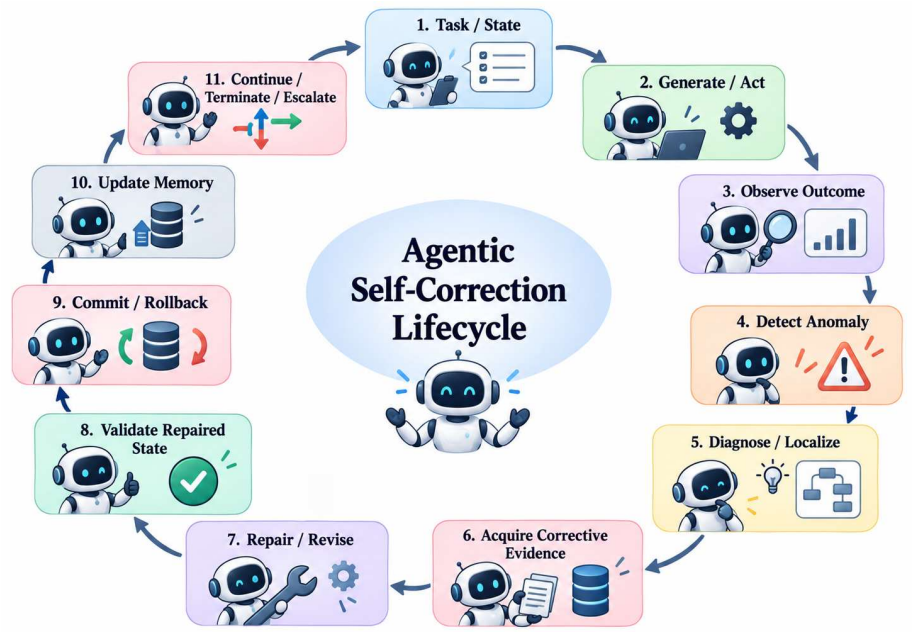


Fig. 3: Unified agentic self-correction lifecycle, showing the transition from task execution and observation to anomaly detection, diagnosis, evidence acquisition, repair, validation, rollback or commitment, memory updating, and subsequent continuation, termination, or escalation.

et al. 2024; Olausson et al. 2024a). The subsequent **Commit / Rollback** stage determines whether the revised state should be accepted or whether execution should return to an earlier state or alternative branch (Yao et al. 2023a). If the corrective episode contains information useful beyond the immediate step, **Update Memory** may store validated reflections, revised experiences, or other persistent information for later reuse (Shinn et al. 2023). Finally, **Continue / Terminate / Escalate** determines whether autonomous execution should resume, stop because the task or resource budget has been exhausted, or transfer control to an external supervisor when recovery remains uncertain or unsafe (Plaat et al. 2025a).

The lifecycle should not be interpreted as a rigid linear pipeline. Existing systems may combine several stages, omit stages that are unnecessary for a particular task, or revisit earlier stages when validation fails (Pan et al. 2024; Du et al. 2026a). For example, diagnosis and repair may occur within a single prompting step, whereas a failed validation may trigger renewed evidence acquisition or alternative repair generation. The framework therefore serves as an analytical model for comparing how different agentic systems organize corrective control rather than as a claim that every self-correction method implements all stages explicitly.

Table 5: Operational stages of the unified agentic self-correction lifecycle.

Lifecycle Stage	Primary Function	Typical Input / Signal	Possible Outcome
Task / State	Establish goal and operational context	User request, system state, memory, environment	Initialized task state
Generate / Act	Produce a candidate response or action	Goal, current state, working context	Reasoning step, response, tool call, or action
Observe Outcome	Capture the consequence of execution	Tool result, API response, runtime log, environment feedback	Observed execution state
Detect Anomaly	Determine whether intervention is required	Verifier score, exception, assertion, inconsistency	Failure or anomaly signal
Diagnose / Localize	Identify the likely source of failure	Execution history, anomaly signal, reasoning trace	Localized error or diagnostic hypothesis
Acquire Evidence	Gather information required for repair	Diagnostic hypothesis, retrieval query, tool output	Corrective or grounding evidence
Repair / Revise	Generate a candidate modification	Diagnosis and corrective evidence	Revised output, plan, action, or code
Validate State	Evaluate whether the repair resolves the failure	Candidate repair, tests, verifier criteria	Validation result
Commit / Rollback	Accept the repair or revert execution	Validation result, confidence, prior state	Updated state or rollback
Update Memory	Preserve validated corrective information	Repair trace, reflection, validated outcome	Updated persistent memory
Continue / Terminate / Escalate	Determine subsequent control flow	Task status, safety criteria, resource limits	Continued execution, termination, or escalation

4.2 Taxonomy Dimensions and Representative Methods

The correction lifecycle in Section 4.1 describes how recovery unfolds operationally, while the hierarchy in Section 3.4 distinguishes the level at which correction is applied. A broader comparison of self-correction mechanisms requires additional dimensions that capture the origin of corrective information, when correction occurs, whether corrective information persists, and how the corrective process is organized. Accordingly,

this paper characterizes agentic self-correction along five complementary dimensions: correction target, feedback source, temporal scope, persistence, and control architecture.

Correction target identifies the component modified during recovery, including final outputs, intermediate reasoning paths, plans, retrieval queries, tool calls, executable actions, memory, or learned policies (Pan et al. 2024). **Feedback source** characterizes the origin of the signal used to evaluate or revise behavior. Such signals may arise from intrinsic self-critique, uncertainty or self-consistency, external tools, retrieved evidence, environmental observations, critic models, multi-agent interaction, or human supervision (Gou et al. 2024; Kamoi et al. 2024; Liang et al. 2024). **Temporal scope** specifies when correction occurs relative to task execution, ranging from generation-time and post-generation revision to within-episode recovery, cross-episode adaptation, and training-time modification (Pan et al. 2024; Shinn et al. 2023).

Persistence describes how long corrective information influences subsequent behavior. Correction may remain confined to the current generation, persist within a working trajectory, be retained in episodic or semantic memory, or become embedded through parameter updates (Shinn et al. 2023; Plaat et al. 2025a). Finally, **control architecture** describes how generation, evaluation, and corrective decisions are organized. Representative configurations include single-model self-critique, generator–critic interaction, verifier-guided control, multi-agent critique or debate, and human-in-the-loop supervision (Madaan et al. 2023; **?**; Raza et al. 2026).

These dimensions are orthogonal rather than mutually exclusive. A single framework may occupy several categories simultaneously. Reflexion, for example, combines trajectory-level correction with environmental or self-evaluative feedback, cross-episode adaptation, and episodic reflective memory (Shinn et al. 2023). CRITIC instead combines output or reasoning correction with tool-grounded verification within the active task context (Gou et al. 2024). The multidimensional representation therefore avoids reducing heterogeneous mechanisms to a single category and permits comparison of methods according to both their corrective capabilities and their operational limitations. Table 6 summarizes representative approaches using this framework.

Table 6: Unified multidimensional taxonomy of representative self-correction methods.

Method	Correction Target	Feedback Source	Verification	Temporal Scope	Persistence	Control Architecture	Main Strength	Main Limitation
Self-Consistency (Wang et al. 2023b)	Reasoning path / output	Agreement among samples	Majority voting across sampled paths	Generation-time	No persistent memory	Single-model ensemble	Reduces dependence on a single sampled reasoning path	Higher inference cost and possible reinforcement of shared model errors
Tree of Thoughts (Yao et al. 2023a)	Reasoning path / plan	Intrinsic state evaluation	Value prompts and branch evaluation	Generation-time	Working / trajectory context	Verifier-guided search	Supports lookahead, branch exploration, and backtracking	Depends on reliable state evaluation and incurs substantial search cost
Self-Refine (Madaan et al. 2023)	Output / rationale	Intrinsic self-feedback	Self-generated critique	Post-generation	Working / turn context	Single-model self-critique	Enables iterative revision without external tools or parameter updates	May reproduce the same epistemic limitations that caused the initial error (Kamoi et al. 2024)
ReAct (Yao et al. 2023b)	Plan / tool call / action	Environmental or tool feedback	Observations returned after actions	Within episode	Working / trajectory context	Reason–act–observe loop	Allows subsequent decisions to incorporate environmental observations	Incorrect observations or actions may propagate through later trajectory steps
Reflexion (Shinn et al. 2023)	Plan / trajectory / memory	Environment feedback / self-evaluation	Task outcomes and success/failure signals	Across episodes	Episodic reflective memory	Agent with reflective memory	Retains lessons across trials without parameter updates	Incorrect reflections may persist and contaminate subsequent decisions
CRITIC (Gou et al. 2024)	Output / reasoning / tool interaction	Tool-grounded feedback	Search, APIs, calculators, or execution results	Post-generation / within episode	Working / task context	Tool-interactive verifier	Grounds critique in information external to the generating model	Depends on tool availability and reliability of external evidence
Self-Edit (Zhang et al. 2023)	Code / executable action	Program execution feedback	Compiler or execution feedback	Post-generation	Working / debugging context	Code-repair loop	Provides actionable feedback for localized program repair	Complex structural defects may require broader reasoning or replanning
Auto-RAG (Yu et al. 2024)	Retrieval query / output	Retrieved evidence	Evidence relevance or information-gap signal	Within episode	Retrieval / trajectory context	Retrieval-guided correction	Acquires additional evidence when information is insufficient	Poor retrieval may introduce irrelevant or misleading evidence
Constitutional AI (Bai et al. 2022)	Output / policy	AI feedback guided by principles	Principle-based critique and preference signals	Training-time	Parameter update	Generator–critic with training	Transforms explicit principles into persistent behavioral adaptation	Depends on the specification and coverage of the underlying principles
Multi-Agent Debate (Liang et al. 2024)	Reasoning path / output	Multi-agent critique	Cross-agent argumentation and aggregation	Post-generation / within episode	Debate history	Multi-agent	Introduces multiple perspectives during evaluation and revision	Adds computational overhead and may retain correlated consensus errors
SWE-Debate (Li et al. 2025)	Plan / code / trajectory	Multi-agent critique and execution feedback	Debate outcomes and test-suite results	Within episode	Trajectory / workspace state	Multi-agent competitive	Combines collaborative critique with executable software feedback	High computational and coordination overhead (Li et al. 2025)

5 Mechanisms for Reflection and Correction

5.1 Intrinsic Self-Evaluation and Refinement

Intrinsic self-correction mechanisms rely primarily on a language model's own generated evaluations and internal uncertainty signals, without requiring external tools, execution environments, retrieved evidence, or direct human intervention at inference time (Madaan et al. 2023; Pan et al. 2024). These approaches are attractive because they can be applied without specialized verification infrastructure and can operate within the same model interaction loop. Their effectiveness, however, depends on whether the model can identify and repair errors that may arise from the same knowledge gaps, reasoning limitations, or calibration failures that produced the initial output (Kamoi et al. 2024).

Intrinsic self-correction encompasses several related but functionally distinct strategies. *Self-Refine* provides a canonical example of iterative self-feedback and post-hoc revision (Madaan et al. 2023). A frozen language model first generates an output, produces natural-language feedback on that output, and then revises the response using its own critique (Madaan et al. 2023; Pan et al. 2024). In contrast, *self-consistency* estimates reliability by sampling multiple reasoning paths and selecting the most frequent final answer (Wang et al. 2023b). Although this procedure can reduce dependence on an individual reasoning trajectory, it is better interpreted as an agreement-based selection mechanism than as a complete corrective loop because it does not necessarily diagnose or repair the underlying reasoning error (Kamoi et al. 2024).

Uncertainty-based mechanisms provide another intrinsic signal by estimating whether a model's prediction is unstable or insufficiently supported. Approaches based on token-level uncertainty, semantic entropy, or self-reported confidence can help identify outputs that warrant further evaluation (Zhang et al. 2024; Farquhar et al. 2024). Such signals, however, generally indicate uncertainty rather than directly identifying the location or cause of an error (Farquhar et al. 2024; Li 2026). Similarly, self-critique and intrinsic error-detection prompts ask the model to inspect its own outputs or reasoning traces against task constraints, consistency criteria, or generated evaluation guidelines (Bai et al. 2022; Madaan et al. 2023). These mechanisms can provide useful diagnostic signals, but the resulting critique remains dependent on the evaluator's own capabilities.

The central limitation of intrinsic control is therefore that **the generator and evaluator may share the same epistemic blind spots**. When generation and critique rely on the same or closely related model capabilities, an incorrect belief, missing fact, or flawed reasoning pattern may persist across both stages (Kamoi et al. 2024; Pan et al. 2024). Critical evaluations of LLM self-correction show that prompted intrinsic correction is not consistently reliable under controlled evaluation conditions and can sometimes leave performance unchanged or degrade initially correct responses (Kamoi et al. 2024; Gou et al. 2024). Models may fail to recognize an error, produce an incorrect critique, or revise a correct answer in response to misleading self-feedback (Kamoi et al. 2024).

Error detection and error repair should therefore be treated as distinct capabilities. Recognizing that an answer is uncertain or incorrect does not guarantee that the model can localize the responsible reasoning step or generate an effective repair (Li 2026). Intrinsic mechanisms remain useful because they provide inexpensive and broadly applicable signals for deciding when further reasoning or revision may be warranted. Their reliability, however, depends on the quality of the evaluative signal, and stronger correction generally becomes more dependable when intrinsic judgments are supplemented by independently informative feedback or verification.

5.2 Tool and Verifier-Grounded Correction

To address the epistemic limitations of intrinsic self-evaluation, agentic architectures can incorporate external tool execution, verifiers, and environmental observations into the correction process. The central advantage of this paradigm is that externally grounded signals can provide information that is at least partially independent of the model's original generation process. Such feedback can therefore reveal factual, computational, or behavioral errors that unsupported introspection may fail to detect (Gou et al. 2024; Kamoi et al. 2024).

Tool-interactive correction is exemplified by the CRITIC framework, which combines generation, external tool interaction, critique, and iterative refinement (Gou et al. 2024). Rather than relying solely on self-generated judgments, CRITIC queries resources such as search engines, calculators, APIs, and code interpreters to evaluate aspects of a candidate response. Its evaluations across free-form question answering, mathematical program synthesis, and toxicity reduction illustrate how externally obtained evidence can improve correction when the selected tool provides task-relevant feedback (Gou et al. 2024). The important distinction is not simply the presence of a tool, but whether the tool supplies an informative verification signal that the generator could not reliably produce through introspection alone (Pan et al. 2024).

Executable feedback is particularly useful in software-oriented tasks. Code execution, compiler diagnostics, runtime exceptions, and unit-test outcomes transform generated programs into observable behavioral evidence that can guide subsequent repair (Zhang et al. 2023; Olausson et al. 2024a). These mechanisms should be distinguished from formal verification in the strict sense. Compilers and test suites evaluate syntactic, execution, or behavioral properties, whereas formal methods such as theorem proving, constraint solving, and model checking can establish stronger guarantees relative to an explicit specification. Across these settings, several corrective functions remain separable: detection establishes that a failure has occurred; verification tests a candidate against an external criterion; localization identifies the responsible component or reasoning step; and repair generates a revised solution. Success at one stage does not imply success at the others (Li 2026).

Environmental feedback extends the same principle beyond static outputs. In agentic systems, actions can produce observations that reveal whether a plan, tool call, or trajectory is progressing toward the intended objective. Reason–act–observe frameworks such as ReAct allow subsequent decisions to incorporate these observations, while reflective agents such as Reflexion can use task outcomes as feedback for later attempts (Yao et al. 2023b; Shinn et al. 2023). Simulators can play a related role by

exposing infeasible actions, constraint violations, or downstream consequences before deployment in a real environment.

External verification, however, is only as reliable as the signal it provides. Incomplete test suites, incorrect specifications, noisy simulators, faulty APIs, weak learned verifiers, or misleading environment observations can produce false confidence or misdirect repair. Olausson et al. show that, once repair cost is considered, self-repair gains in code generation can be modest or absent, whereas stronger feedback substantially improves outcomes (Olausson et al. 2024a). This finding indicates that the bottleneck may lie not merely in detecting failure, but in generating sufficiently informative feedback for localization and repair. Related work on iterative AI code generation also raises the possibility that functional improvement can coexist with worsening security properties when automated feedback does not explicitly test for them (Shukla et al. 2025). Thus, externally grounded correction is generally more defensible than unsupported introspection when the verifier is independent, relevant, and trustworthy, but external feedback does not itself guarantee successful or safe recovery.

5.3 Retrieval-Grounded Correction

Canonical Retrieval-Augmented Generation (RAG) is commonly implemented as a retrieve-then-generate pipeline in which documents are retrieved from an external corpus and supplied as context for response generation (Lewis et al. 2020). Although this architecture can reduce reliance on parametric knowledge and improve factual grounding, retrieval alone does not constitute self-correction. Retrieval becomes corrective when an agent evaluates whether its current knowledge or retrieved evidence is inadequate, modifies its retrieval behavior in response to that assessment, and subsequently revises its reasoning, answer, or action using the newly acquired evidence (Yu et al. 2024; Asl et al. 2025a). The defining feature is therefore not access to external documents, but a feedback loop in which retrieval outcomes alter subsequent agent behavior.

A first component of this loop is *information-gap detection*. Rather than retrieving indiscriminately, adaptive systems attempt to determine when external evidence is necessary. FLARE initiates retrieval when the model encounters low-confidence portions of an anticipated generation, allowing retrieval to be triggered by local uncertainty during decoding (Jiang et al. 2023). Adaptive-RAG approaches the problem at the query level by predicting task complexity and routing requests toward no-retrieval, single-step, or iterative retrieval strategies (Jeong et al. 2024). Self-RAG similarly learns reflection signals that govern whether retrieval is required and whether retrieved passages adequately support generation (Asai et al. 2024). These approaches illustrate a broader transition from fixed retrieval policies toward conditional retrieval based on estimated information need.

A second function is *query correction and reformulation*. When initial retrieval fails to provide sufficient evidence, a corrective agent can decompose the information need, reformulate the query, or generate additional searches conditioned on its evolving reasoning state. Auto-RAG follows this general pattern by iteratively identifying missing information and adapting subsequent retrieval operations (Yu et al. 2024). IRCoT provides an earlier example of this interaction by interleaving reasoning steps

with retrieval, allowing intermediate reasoning to determine which evidence should be requested next (Trivedi et al. 2023). In this setting, retrieval is not merely repeated; each subsequent query is informed by unresolved reasoning requirements.

The third function concerns *evidence verification*. Retrieved information can itself be irrelevant, incomplete, contradictory, or incorrect, making retrieval quality a distinct bottleneck. Corrective RAG (CRAG) introduces a retrieval evaluator that estimates document relevance and can invoke alternative web search when the initially retrieved evidence is judged inadequate (Yan et al. 2024). Self-RAG evaluates retrieved passages through learned reflection signals, while FAIR-RAG introduces structured evidence assessment to distinguish supported findings from unresolved information gaps (Asai et al. 2024; Asl et al. 2025a). RAG-Gym extends this idea toward sequential optimization by treating retrieval and evidence-selection decisions as actions within an agentic search process (Xiong et al. 2025). Collectively, these methods show that access to more evidence is not sufficient; the system must also estimate whether that evidence is relevant and trustworthy enough to support revision.

The final stage is *grounded answer revision*, in which verified evidence changes an earlier hypothesis, reasoning step, or generated response. This stage distinguishes corrective retrieval from pipelines that simply append additional context without reconsidering prior conclusions. Reported results from recent adaptive systems suggest that targeted retrieval can improve performance on knowledge-intensive and multi-hop tasks. For example, FAIR-RAG reports an F1 score of 0.453 on HotpotQA, corresponding to an 8.3-point absolute improvement over the iterative baselines reported in that study (Asl et al. 2025a). Auto-RAG similarly reports gains across open-domain question-answering settings while adapting retrieval iterations to task requirements (Yu et al. 2024). Such results are most informative when interpreted as evidence for adaptive evidence acquisition rather than as proof that additional retrieval is inherently beneficial.

Retrieval-grounded correction remains vulnerable to several failure modes. Irrelevant but plausible passages can redirect reasoning toward incorrect conclusions, while repeated query reformulation can produce query drift that progressively departs from the original task (Yu et al. 2024; Yan et al. 2024). Conflicting, stale, or low-quality evidence may also mislead learned evaluators, causing incorrect information to be treated as corroboration. Moreover, iterative retrieval introduces additional latency and inference cost, creating a need for stopping policies that balance the expected value of further evidence against its computational expense (Jeong et al. 2024; Xiong et al. 2025). Retrieval-grounded correction is therefore most effective when information-gap detection, query adaptation, evidence evaluation, and revision operate as a coordinated control loop rather than as repeated retrieval alone.

5.4 Memory-Enhanced Correction

Short-term working context allows an agent to revise outputs within a single interaction, whereas persistent memory extends corrective behavior across episodes by retaining prior failures, critiques, successful trajectories, and reusable strategies (Shinn et al. 2023; Plaat et al. 2025a). In this sense, memory transforms correction from a transient response-level process into a mechanism for cross-episodic adaptation (Du

et al. 2026a; Wei et al. 2026). This persistence, however, introduces a corresponding reliability risk: memory converts one-shot correction into persistent adaptation, but persistence also allows incorrect reflections and failed experiences to influence future behavior. Consequently, the value of memory-enhanced correction depends not only on what an agent remembers, but also on how experiences are abstracted, retrieved, revised, and discarded.

Memory-enhanced correction can be decomposed into five closely related functions. The first is *experience capture*, in which an agent records execution histories, environmental outcomes, tool failures, critiques, or other signals associated with successful and unsuccessful attempts. Reflexion provides a foundational example by converting task outcomes into natural-language reflections that can be reused in subsequent trials (Shinn et al. 2023). The second function is *abstraction and consolidation*. Rather than preserving complete interaction traces indefinitely, more recent systems attempt to transform raw experiences into compact and reusable representations. Approaches such as A-MEM and Evo-Memory illustrate a broader shift toward dynamically structured and self-evolving memory stores that organize accumulated experience for later reuse (Xu et al. 2025; Wei et al. 2025).

The third function is *memory retrieval*. Corrective experience is useful only when the agent can identify which prior failures, strategies, or reflections are relevant to the current task. Early reflective-memory approaches largely relied on retrieving textual experiences from accumulated context, whereas recent systems increasingly treat retrieval itself as a learnable control problem (Shinn et al. 2023). Memory-R1 and MEM1, for example, represent efforts to coordinate memory selection with reasoning so that only task-relevant experiences influence subsequent decisions (Yan et al. 2026; Zhou et al. 2026). The fourth function is *memory update and forgetting*. Persistent stores must accommodate new evidence, reconcile redundant or conflicting experiences, and prevent obsolete information from dominating future behavior. Recent work therefore explores reflective memory management, selective updating, consolidation, pruning, and coordination between short-term and long-term memory (Tan et al. 2025; Yu et al. 2026). The final function is *cross-episodic correction*, in which retrieved experience conditions later planning, reasoning, sub-goal decomposition, or tool use, allowing prior failures to influence future trajectories rather than being rediscovered independently (Shinn et al. 2023; Wei et al. 2026).

The principal limitation is that storing a reflection is not equivalent to validating that the reflection is correct. Memory modules may preserve self-generated diagnoses that were never independently verified, creating risks of memory contamination, stale reflections, and misaligned experience reuse (Du et al. 2026a; Raza et al. 2026). If an agent incorrectly attributes the cause of a failure, the resulting lesson can be stored as if it were valid and later retrieved as corrective guidance. Repeated reuse can then transform a local diagnostic error into a persistent behavioral bias (Kamoi et al. 2024; Li 2026). Similar problems arise when retrieval mechanisms select experiences that are superficially similar but causally irrelevant to the current task, particularly under distribution shift (Plaat et al. 2025a).

Memory-enhanced correction should therefore be treated as a controlled persistence problem rather than simply a storage problem. Reliable designs require selective

write policies, relevance-aware retrieval, mechanisms for revising or forgetting obsolete lessons, and independent checks on high-impact memories before they are repeatedly reused. Persistent memory can substantially extend the temporal scope of agentic self-correction, but the same persistence can amplify incorrect lessons when memory quality is not actively maintained.

5.5 Multi-Agent and External Critique

A prominent strategy for reducing the limitations of single-model introspection is to distribute generation, critique, and verification across multiple agents or specialized evaluator roles (Liang et al. 2024; Pan et al. 2024; Plaat et al. 2025a). The underlying rationale is that separating generation from evaluation can increase corrective diversity and reduce dependence on a single reasoning trajectory, but multiple agents do not necessarily provide independent evidence. Agents derived from similar model families may share factual blind spots, reasoning biases, or erroneous assumptions, limiting the extent to which additional participants improve verification.

The simplest configuration is *generator–critic separation*, in which one model or role generates a candidate solution while another evaluates its quality and proposes revisions. The separation may be architectural, involving distinct models, or functional, using different prompts or roles within the same model (Madaan et al. 2023; Pan et al. 2024). Related systems extend this pattern through *peer critique*, where agents inspect one another's intermediate reasoning, outputs, tool interactions, or proposed actions before subsequent revision. Such externalized critique can expose inconsistencies that remain undetected during isolated generation, particularly when feedback is supplemented by tools or other grounded signals (Gou et al. 2024; Raza et al. 2026).

Multi-agent debate introduces a more structured interaction in which agents defend competing hypotheses, challenge intermediate arguments, and iteratively revise their positions before an answer is selected or synthesized (Liang et al. 2024; Li et al. 2025; Zhou and Chen 2025). This design seeks to induce reasoning diversity rather than simply generate multiple independent samples. SWE-Debate, for example, applies competitive debate to software issue resolution, allowing agents to challenge alternative fault hypotheses and proposed repairs (Li et al. 2025). Heterogeneous debate architectures further attempt to increase diversity by assigning different capabilities, roles, or reasoning strategies to participating agents (Zhou and Chen 2025). The potential benefit is not merely agreement, but the explicit exposure and examination of competing assumptions.

A related direction is *recursive critique*, in which an initial critique is itself evaluated, challenged, or refined through additional evaluative passes (Wen et al. 2026). Such mechanisms extend correction beyond a single generator–critic exchange by introducing meta-evaluation of the feedback process. Recursive self-critiquing has been investigated as a possible approach to scalable oversight when direct evaluation is difficult, although additional critique layers also create further opportunities for erroneous assessments to propagate (Wen et al. 2026). Dedicated *judge or verifier agents* provide another architectural pattern by ranking, scoring, or arbitrating between candidate

outputs. Agent-as-a-Judge and M-MAD illustrate efforts to use agent-based evaluators and multi-agent deliberation for structured assessment against explicit criteria (Zhuge et al. 2025; Feng et al. 2025).

These architectures can improve error detection by introducing alternative reasoning paths, adversarial challenge, role specialization, and explicit comparison between candidate solutions. Structured debate has been shown to encourage divergent reasoning and can reduce the tendency of a single model to persist with an initially incorrect line of thought (Liang et al. 2024). Specialization may further separate solving, checking, and fault-localization functions, potentially improving the quality of corrective feedback (Li et al. 2025; Zhou and Chen 2025).

The principal limitation is that multiple agents should not be conflated with independent verification. Agents sharing similar pretrained representations may reproduce correlated errors and converge on an incorrect consensus (Kamoi et al. 2024; Raza et al. 2026). Debate can also produce conformity, persuasive but unsupported arguments, or repeated reinforcement of an erroneous premise (Liang et al. 2024; Kamoi et al. 2024). Comparative analyses further suggest that some apparent benefits of debate may overlap with those obtainable through aggregation or voting, raising the question of when interaction itself adds value beyond sampling additional candidate solutions (Choi et al. 2025). Judge agents introduce their own calibration and preference biases, while recursive critique can amplify an inaccurate initial diagnosis (Zhuge et al. 2025; Feng et al. 2025; Wen et al. 2026). Multi-round interaction also increases communication, context, and inference costs. Consequently, multi-agent critique is most defensible when diversity among agents is paired with explicit verification criteria, heterogeneous evidence sources, or externally grounded signals rather than relying on consensus alone.

5.6 Training-Time Correction and Self-Improvement

A fundamental distinction in agentic self-correction concerns whether corrective behavior is applied only during inference or is converted into persistent model adaptation. Inference-time correction modifies the current output, reasoning state, action, or trajectory without changing model parameters, whereas training-time correction uses corrective experience to produce persistent behavioral change through parameter or policy updates. Test-time reflection can therefore recover from local failures within an active interaction, but repeated critique and revision increase token use, latency, and sensitivity to prompting and feedback quality (Madaan et al. 2023; Kamoi et al. 2024). Moreover, because the underlying parameters remain unchanged, similar failures may recur across future episodes unless corrective experience is stored externally or incorporated into subsequent training (Pan et al. 2024; Du et al. 2026a). Training-time approaches seek to internalize useful corrective patterns so that later behavior can improve without reproducing the full inference-time correction loop (Bai et al. 2022; Hu et al. 2024).

This process begins with *critique generation*. Corrective supervision may consist of natural-language diagnoses, revised responses, normative judgments, preference labels, or verified solution trajectories. Constitutional AI provides an influential example in which a model critiques and revises its responses according to an explicit set of

principles, after which AI-generated comparisons contribute to subsequent alignment training (Bai et al. 2022). Related self-refinement approaches use teacher-generated or self-generated critiques to identify deficiencies in candidate responses and construct improved demonstrations (Ranaldi and Freitas 2024; Hu et al. 2024). The key transition from inference-time correction to training-time adaptation occurs when these transient critiques and revisions are transformed into reusable optimization signals.

Such signals can be represented as supervised correction traces, preference pairs, or rewards. Supervised fine-tuning can train on sequences containing an initial response, critique, and corrected response, thereby distilling parts of the refinement process into model behavior (Hu et al. 2024; Pan et al. 2024). Preference-based approaches instead construct pairs such as $(y_{\text{preferred}}, y_{\text{dispreferred}})$ and optimize the model toward higher-quality corrected responses. Self-Refine Instruction-Tuning combines refinement-oriented demonstrations with preference optimization to encourage improved reasoning behavior (Ranaldi and Freitas 2024), while Self-Refinement Tuning uses self-generated feedback and preference signals to strengthen persistent refinement capabilities (Hu et al. 2024). More generally, reinforcement-learning approaches such as SCoRe explicitly train models on multi-turn self-correction trajectories so that the ability to revise an initial answer becomes part of the learned policy rather than an exclusively prompted behavior (Kumar et al. 2025).

Training-time self-improvement can further become iterative: a model generates new trajectories, receives internal or external evaluations, filters candidate corrections, and updates its policy using the resulting data (Kumar et al. 2025; Du et al. 2026a). This creates the possibility of converting successful correction episodes into increasingly persistent capabilities. However, persistence also changes the consequences of erroneous feedback. An incorrect inference-time critique may damage a single trajectory, whereas an incorrect training signal can affect behavior across many future tasks.

For this reason, repeated self-generated feedback should not be equated with reliable self-improvement. If hallucinated critiques or misdiagnosed errors are admitted into the training set, the optimization process may reinforce rather than eliminate the underlying failure (Kamoi et al. 2024; Pan et al. 2024). Preference or reward misspecification can similarly encourage models to optimize proxy signals rather than the intended corrective objective, while repeated learning from narrow synthetic trajectories can reduce diversity or weaken behavior outside the training distribution (Du et al. 2026a; Wen et al. 2026). Persistent optimization may also introduce safety trade-offs that are not captured by the primary task metric. Evidence from iterative code-generation settings, for example, indicates that functional improvement under automated feedback can coexist with degradation in security properties when those properties are absent from the evaluation signal (Shukla et al. 2025).

Training-time correction should therefore be understood as the *distillation of corrective feedback into persistent behavior*, not simply as repeated self-training. Its reliability depends on the quality of the critique, the construction and filtering of training signals, the breadth of the training distribution, and the ability of evaluators to detect both task errors and unintended regressions. Inference-time correction and

training-time adaptation are consequently complementary: the former provides flexible local recovery, while the latter can preserve validated corrective experience across future interactions.

6 When Does Self-Correction Work?

6.1 Cross-Study Empirical Evidence

Table 7 synthesizes evidence from 21 representative empirical studies to examine when self-correction improves performance, yields limited benefit, or becomes counterproductive. Rather than cataloguing methods, the comparison emphasizes differences in correction mechanism, feedback source, verifier independence, iteration strategy, computational requirements, and observed failure conditions. The selected studies span intrinsic refinement, search-based reasoning, tool-grounded correction, executable verification, retrieval-grounded recovery, memory-based adaptation, multi-agent critique, and training-time self-improvement. Particular attention is given to negative results, equal-compute comparisons, and cases in which apparently successful correction depends on the quality of an external verifier or feedback signal.

Table 7: Cross-study empirical evidence on self-correction mechanisms and their boundary conditions.

Study	Setting	Correction / Feedback	Independent Verifier?	Reported Outcome	Cost / Iterations	Failure / Boundary Condition
Madaan et al. (2023) (Madaan et al. 2023)	Code readability and mathematics; Codex / ChatGPT	Iterative natural-language refinement using intrinsic self-feedback	No	Readability: 37.4 → 51.3 (+13.9 pp); PAL Math: 71.3 → 26.7 (-44.6 pp), relative to direct generation	1–4 iterations; 2–4× inference tokens	Self-generated critique can improve open-ended outputs while misdiagnosing errors in more constrained reasoning tasks.
Wang et al. (2023) (Wang et al. 2023b)	Mathematical reasoning; PaLM-540B	Parallel reasoning-path sampling with majority-vote consensus	No	GSM8K: 56.5 → 74.4 (+17.9 pp) relative to single-path CoT	$k = 40$–64 sampled generations	Consensus cannot resolve systematic errors shared across sampled reasoning paths.
Yao et al. (2023) (Yao et al. 2023a)	Deliberate reasoning / planning; GPT-4	Tree search with model-generated branch evaluation and backtracking	Partial	Game of 24: 4.0 → 74.0 (+70.0 pp) relative to single-path CoT	Depth 1–4; substantially more generation and evaluation calls	Incorrect value estimates can eliminate valid solution branches and propagate search errors.
Bai et al. (2022) (Bai et al. 2022)	Safety and alignment; 52B RLHF model	Principle-guided critique and revision followed by RLAIF	No	Reported substantial reduction in harmful responses relative to the helpful RLHF baseline	16k critique–revision examples plus subsequent training	Behavior depends on the specification, interpretation, and coverage of the constitutional principles.
Shinn et al. (2023) (Shinn et al. 2023)	Code and embodied tasks; GPT-4 / GPT-3.5	Episodic reflection using environmental rewards and execution feedback	Yes	HumanEval: 68.1 → 91.0 (+22.9 pp); ALFWorld: 73.0 → 97.0 (+24.0 pp) relative to standard ReAct	1–12 trials plus reflection and memory	Incorrect reflections may persist across episodes when stored without adequate validation.
Gou et al. (2024) (Gou et al. 2024)	Mathematics and QA; ChatGPT	Tool-interactive critique using Python, search, and external tools	Yes	GSM8K: 72.5 → 78.2 (+5.7 pp); without tools: 77.0, relative to PoT baseline	1–3 iterations plus tool calls	Benefits depend on whether the external tool returns relevant and reliable evidence.
Kamoi et al. (2024) (Kamoi et al. 2024)	Reasoning; GPT-4 / LLaMA-2-70B	Intrinsic post-hoc self-correction using model-generated evaluation	No	No consistent improvement over strong initial prompting	1–3 rounds; 2–3× inference calls	Correction is unreliable when the evaluator has no information beyond the model's original knowledge and reasoning.
Huang et al. (2024) (Huang et al. 2024)	Mathematical reasoning; GPT-4-Turbo	Iterative intrinsic self-evaluation and revision	No	Accuracy can deteriorate relative to the initial CoT response	1–3 additional inference turns	The model may revise an initially correct answer when no reliable external signal indicates whether correction is necessary.
Olausson et al. (2024) (Olausson et al. 2024a)	Program repair; GPT-3.5 / GPT-4	Execution- and unit-test-guided self-repair	Yes	Modest gain over Pass@1; limited or no advantage over Pass@k under matched compute	1–5 repair attempts; equal-compute evaluation	Failure detection does not guarantee successful bug localization or repair synthesis.

Continued on next page

Table 7 continued from previous page

Study	Setting	Correction / Feedback	Independent Verifier?	Reported Outcome	Cost / Iterations	Failure / Boundary Condition
Ranaldi and Freitas (2024) (Ranaldi and Freitas 2024)	Mathematics and commonsense QA; Llama2-7B / Mistral-7B	Instruction tuning and DPO using teacher-generated refinement traces	Partial	GSM8K: 53.2 → 64.6 (+11.4 pp); CSQA: 54.9 → 64.3 (+9.4 pp) relative to standard instruction tuning	Approx. 3k demonstrations; standard test-time inference	Persistent improvement depends on the correctness of teacher-generated critiques and revisions.
Hu et al. (2024) (Hu et al. 2024)	Mathematical reasoning; Llama-2-7B	Self-feedback preference optimization using intrinsic preference pairs	No	GSM8K: 48.2 → 57.5 (+9.3 pp) relative to the base SFT model	Offline DPO; no iterative test-time correction	Incorrect self-generated preferences can be internalized during training.
Liang et al. (2024) (Liang et al. 2024)	Mathematics and factual reasoning; GPT-3.5-Turbo	Multi-agent debate using peer critique and judge arbitration	Partial	GSM8K: 77.0 → 85.0 (+8.0 pp) relative to single-agent CoT	2–3 debate rounds; multiple agents per round	Correlated reasoning errors, conformity, or judge bias can reduce the benefit of debate.
Renze and Guven (2024) (Renze and Guven 2024)	Medical QA; GPT-4 / Llama-2-70B	Prompt-based reflection using model-generated evaluative feedback	No	GPT-4: 78.6 → 93.2 (+14.6 pp); Llama-2-70B: 37.6 → 67.2 (+29.6 pp)	1–2 additional reflection turns	Improvement varies substantially with base-model capability and reflection strategy.
Yu et al. (2024) (Yu et al. 2024)	Open-domain QA; Llama-3-8B / GPT-3.5	Iterative retrieval and query reformulation using information-gap signals and retrieved evidence	Partial	2Wiki: 36.8 → 42.1 (+5.3 pp) relative to FLARE; reported lower latency than Self-RAG	1–4 adaptive retrieval iterations	Incorrect information-gap detection or repeated reformulation can introduce query drift and irrelevant evidence.
Aghajani Asl et al. (2025) (Asl et al. 2025a)	Multi-hop QA; Llama-3-8B-Instruct	Structured evidence assessment and iterative query refinement	Partial	HotpotQA F1: 37.0 → 45.3 (+8.3 pp) relative to Auto-RAG	1–3 rounds plus evidence-assessment and retrieval passes	Incorrect relevance judgments or misleading retrieved passages can redirect subsequent reasoning.
Li et al. (2025) (Li et al. 2025)	Repository-level software engineering; Claude 3.5 Sonnet	Multi-agent debate over fault hypotheses and repairs with test-suite feedback	Yes	SWE-bench Lite: 32.1 → 39.4 (+7.3 pp) relative to single-agent SWE-agent	2–4 debate rounds; multiple agent calls	Agents may converge on the same incorrect fault localization while incurring substantial coordination cost.
Shukla et al. (2025) (Shukla et al. 2025)	Software security; GPT-4o / Claude 3.5	Iterative code repair under compiler and unit-test feedback	Yes	Functional correctness improves while reported vulnerability incidence increases	1–5 repair cycles	Optimizing tested functionality can leave untested security properties unchanged or degraded.
Choi et al. (2025) (Choi et al. 2025)	Multi-agent decision making; GPT-3.5-Turbo	Multi-agent debate compared with majority voting	No	Debate does not consistently outperform majority voting under matched computational conditions	3 debate rounds / $k = 5$ voting samples	Interaction may provide limited additional information when participating agents exhibit strongly correlated errors.

Continued on next page

Table 7 continued from previous page

Study	Setting	Correction / Feedback	Independent Verifier?	Reported Outcome	Cost / Iterations	Failure / Boundary Condition
Ma et al. (2025) (Ma et al. 2025)	Competition mathematics; Llama-3.1-70B-Instruct	Verifier-guided self-verification and correction using explicit correctness feedback	Yes	MATH500: 54.2 → 77.4 (+23.2 pp); AIME24: 16.7 → 23.3 (+6.6 pp) relative to self-refinement without oracle guidance	2 verifier-guided correction stages	Removing reliable verifier guidance substantially weakens correction and can reverse apparent gains.
Wen et al. (2026) (Wen et al. 2026)	Claim verification; Qwen2.5-72B-Instruct	Recursive higher-order critique in which critiques are themselves evaluated	Partial	TEM4 accuracy: 34.38 → 67.19 (C1) → 71.88 (+37.50 pp)	1–3 critique levels; multiple higher-order passes	An incorrect initial diagnosis can be propagated or amplified through subsequent meta-critique.
Li (2026) (Li 2025)	Complex reasoning; DeepSeek / GPT-3.5	Intrinsic and hint-assisted iterative correction	No	GPT-3.5 intrinsic repair: 26.8%; iterative: 67.9%; DeepSeek intrinsic repair: 16.7%	1–3 iterative correction turns	High error-detection capability does not necessarily translate into successful error repair.

Across these studies, three broad empirical patterns emerge. First, *intrinsic correction is highly task- and model-dependent rather than uniformly beneficial*. Self-refinement and reflective prompting can produce substantial gains in some settings (Madaan et al. 2023; Renze and Guven 2024), yet controlled evaluations also show stagnation or degradation when the model must judge its own reasoning without sufficiently informative feedback (Kamoi et al. 2024; Huang et al. 2024; Li 2025). The relevant distinction is therefore not simply whether reflection occurs, but whether the resulting feedback contains information capable of discriminating between correct and incorrect trajectories.

Second, the largest and most consistent improvements tend to occur when correction is coupled to *task-relevant external or independently informative signals*. Execution results, environmental rewards, retrieved evidence, test suites, and explicit correctness verifiers can constrain the corrective search space more effectively than unsupported self-evaluation (Gou et al. 2024; Shinn et al. 2023; Asl et al. 2025a; Ma et al. 2025). However, verifier availability alone is insufficient. In software repair, for example, execution feedback may identify that a candidate fails without revealing how the defect should be localized or repaired (Olausson et al. 2024a). Retrieval systems similarly depend on the relevance and reliability of the evidence admitted into the correction loop (Asl et al. 2025a).

Third, additional correction iterations should not be interpreted as free performance gains. Equal-compute comparisons indicate that iterative repair may be matched or surpassed by independent sampling or aggregation when successive critiques contribute little new information (Olausson et al. 2024a; Choi et al. 2025). The effectiveness of correction therefore depends on the informational value of each additional evaluation step, not merely on iteration count. Moreover, optimizing against incomplete feedback can improve the measured objective while degrading unobserved properties, as illustrated by security regressions during iterative code improvement (Shukla et al. 2025). Persistent mechanisms introduce an analogous risk because incorrect reflections may survive beyond the episode in which they were produced (Shinn et al. 2023). Taken together, the evidence suggests that self-correction is most reliable when feedback is informative, sufficiently independent of the original error process, relevant to the target failure, and inexpensive enough to justify further intervention.

6.2 Evidence of Successful Correction

Across the empirical evidence, successful self-correction depends less on the number of reflection rounds than on the informational quality of feedback, the localizability of the error, and the control policy determining whether further correction is warranted. Reliable verification is the strongest recurring condition. Unsupported intrinsic evaluation can stagnate or degrade correct responses when the generator and evaluator share the same blind spots (Kamoi et al. 2024; Huang et al. 2024), whereas explicit correctness signals and process-level verification can substantially improve recovery (**?**Ma et al. 2025). Executable feedback is particularly useful because program execution, unit tests, compilers, calculators, and tool outputs constrain the space of plausible repairs (Gou et al. 2024; Zhang et al. 2023). Even then, detecting a failed execution does not guarantee successful localization or repair (Olausson et al. 2024a).

Grounding also matters in knowledge-intensive settings. Adaptive retrieval can support correction when additional evidence addresses a genuine information gap and is subsequently evaluated for relevance and consistency (Yu et al. 2024; Asl et al. 2025a; **?**). Similarly, critic quality matters more than simply introducing another critique stage. Stronger or functionally separated evaluators can provide more informative diagnostics (Madaan et al. 2023; Ranaldi and Freitas 2024), while multi-agent critique is useful only when additional agents contribute genuinely diverse evidence rather than correlated judgments (Liang et al. 2024).

Correction difficulty also increases with error depth. Localized factual, computational, or execution errors are generally more tractable than failures rooted in missing knowledge, incorrect global plans, or long-horizon trajectories (Li 2025; Olausson et al. 2024a). Persistent memory can extend successful correction across episodes when stored experiences are accurate and selectively retrieved (Shinn et al. 2023; Du et al. 2026a; Tan et al. 2025), but unverified memories can instead preserve incorrect lessons. Finally, effective correction requires stopping policies that limit unnecessary revision. Confidence, verification outcomes, or diminishing expected gains can indicate when further iteration is unlikely to justify its cost (Zhang et al. 2024; Kamoi et al. 2024; Huang et al. 2024). Successful correction therefore emerges from the interaction of *feedback reliability, error localizability, persistence quality, and intervention control*, rather than reflection frequency alone.

6.3 Intrinsic versus Externally Grounded Correction

A central distinction in agentic self-correction concerns the source and independence of corrective feedback. Intrinsic mechanisms rely primarily on signals generated by the model itself, including natural-language self-critique, confidence estimates, or agreement across sampled reasoning paths (Madaan et al. 2023; Wang et al. 2023b). Their principal advantages are low deployment overhead, broad task applicability, and independence from external tools or environments. However, because generation and evaluation often draw on the same underlying representations, intrinsic correction remains vulnerable to shared epistemic blind spots, false confidence, hallucinated critique, and unnecessary revision (Kamoi et al. 2024; Pan et al. 2024). Recent empirical analysis further shows that intrinsic self-correction can induce answer instability, prompt sensitivity, and cognitive biases when reliable external evidence is unavailable (Zhang et al. 2025).

Intrinsic correction is nevertheless not uniformly ineffective. It can succeed when the model already possesses the knowledge needed for repair, when errors are sufficiently localized, or when self-verification has been explicitly trained rather than merely prompted (Madaan et al. 2023; Li 2025). ReVISE, for example, learns verification and refinement through preference-based training, illustrating that intrinsic verification can become more reliable when the capability itself is optimized rather than assumed to emerge from prompting (Lee et al. 2025). Similarly, ProgCo structures verification through self-generated and self-executed programs, and reports further improvement when actual program tools are introduced (Song et al. 2025). These results qualify broader evidence that unsupported introspection can stagnate or degrade initially correct answers (Kamoi et al. 2024; Huang et al. 2024).

Externally grounded correction instead introduces evidence from execution environments, retrieved documents, tools, environmental observations, or explicit correctness criteria (Gou et al. 2024; Shinn et al. 2023; Yu et al. 2024). Such signals can constrain the corrective search space and provide information unavailable to the original generator. Code execution, unit tests, and rule-based checks can expose concrete behavioral failures (Zhang et al. 2023; Olausson et al. 2024a), while verifier-guided reasoning can substantially improve repair when the verification signal is trustworthy (Ma et al. 2025). External grounding is not inherently reliable, however. Incomplete tests may miss security regressions (Shukla et al. 2025), and retrieval-based systems may admit irrelevant or misleading evidence (Yan et al. 2024).

The resulting trade-off is therefore not simply *intrinsic versus extrinsic*, but *cost versus informational value*. Intrinsic feedback is inexpensive and scalable, whereas externally grounded feedback generally requires additional infrastructure, retrieval, execution, or supervision (Du et al. 2026a; Raza et al. 2026). Reliable correction ultimately depends on whether the feedback introduces sufficiently independent, relevant, and trustworthy information to discriminate between competing error hypotheses.

6.4 Error Depth and Correction Difficulty

Understanding why self-correction succeeds in some settings but fails in others requires considering not only whether an error is detected, but also how deeply it is embedded in the agent's reasoning and execution process. Across recent studies, a consistent distinction emerges between *error detection*, recognizing that an output or state is incorrect; *error localization*, identifying the responsible step or component; and *error repair*, generating an effective corrective action (Li 2026; Kamoi et al. 2024). Success at the first stage does not imply success at the latter two. This distinction is particularly important in long-horizon settings, where the observable failure may occur several steps after its underlying cause (Olausson et al. 2024a; Deshpande et al. 2025).

At the shallowest level, *surface errors* include formatting violations, syntax mistakes, arithmetic slips, and other locally observable defects. Such failures are comparatively tractable because prompts, compiler diagnostics, or execution feedback can often identify a narrow repair target (Madaan et al. 2023; Olausson et al. 2024a). *Local reasoning errors* occur when an intermediate deduction is incorrect while the broader solution strategy remains viable. Step-level evaluation or verifier-guided reasoning can therefore support targeted revision without discarding the entire reasoning path (Lightman et al. 2024; Ma et al. 2025). Greater difficulty arises with *semantic or model-knowledge errors*, where the required fact, concept, or relation is missing or incorrectly represented. In such cases, repeated introspection may reproduce the same misconception unless corrective evidence is introduced through retrieval or another external source (Kamoi et al. 2024; Yu et al. 2024).

Deeper failures affect the agent's control structure rather than an isolated response. *Plan-level errors* involve inappropriate decomposition, subgoal ordering, or strategy selection and may require replanning, search, or backtracking rather than local editing (Yao et al. 2023a; Wang et al. 2023a). *Trajectory-level errors* emerge across sequences of actions, tool calls, and observations, including instruction drift, looping, and cumulative state corruption (Tayebati et al. 2026). Because the causal step may

be temporally distant from the eventual failure, diagnosis increasingly depends on reasoning over complete execution histories and identifying dependencies across intermediate states (Li et al. 2025; Deshpande et al. 2025). Finally, *memory-induced errors* arise when incorrect reflections, stale experiences, or irrelevant prior trajectories are retrieved and reused (Shinn et al. 2023; Tan et al. 2025). Persistent storage can therefore transform a local diagnostic mistake into a recurring source of future error (Du et al. 2026a; Yu et al. 2026).

These observations align with the proposed *Error Depth Hypothesis*: as model capability improves, the residual errors may increasingly consist of deeper reasoning, knowledge, planning, or trajectory failures that are harder to localize and repair (Li 2026). This helps explain the associated Accuracy-Correction Paradox, in which stronger initial accuracy does not necessarily imply greater intrinsic repair capability (Li 2026). Agentic self-correction should therefore be evaluated not only by whether failures are detected, but by whether the system can correctly attribute their cause at the appropriate depth and select a repair mechanism matched to that failure.

6.5 Cost-Effectiveness of Self-Correction

Evaluating the practical value of agentic self-correction requires comparing performance under matched or approximately equal computational budgets rather than treating additional reflection as intrinsically beneficial. The relevant objective is therefore not maximum accuracy after unrestricted deliberation, but the marginal quality gain obtained per additional unit of computation, interaction, or verification. From this perspective, the trade-off between iterative self-repair and alternative uses of test-time compute, such as independent sampling, search, or verifier allocation, becomes a central design question (Snell et al. 2024; Olausson et al. 2024a).

Empirical comparisons show that repeated repair does not always dominate simpler sampling strategies once token usage or model calls are controlled. In program synthesis, Olausson et al. show that self-repair can provide limited advantage over generating additional independent candidates under matched compute, particularly when the feedback signal does not substantially improve error localization (Olausson et al. 2024a). Self-consistency provides a complementary baseline by allocating computation to diverse reasoning paths rather than repeatedly revising one trajectory (Wang et al. 2023b). Similar concerns arise in multi-agent settings, where debate can consume multiple rounds of communication without consistently outperforming majority aggregation under comparable budgets (Choi et al. 2025). These findings suggest that additional computation is useful only when successive correction steps contribute information beyond what could be obtained from resampling or aggregation.

Cost-effectiveness should therefore be assessed in terms of tokens, model invocations, tool calls, retrieval operations, and verifier interactions rather than accuracy alone (Wang et al. 2024a; Ji et al. 2026a). Verifier calls are particularly valuable when they reduce uncertainty, identify a specific failure, or prevent unnecessary revision. Tool-grounded critique can justify its additional cost when external evidence materially constrains the repair space (Gou et al. 2024), while recent test-time scaling work similarly emphasizes the importance of allocating computation toward informative

verification rather than undirected additional reasoning (Setlur et al. 2025). Conversely, repeated reflection with weak or redundant feedback can exhibit diminishing returns and may eventually induce over-correction or reinforce an earlier diagnostic error (Madaan et al. 2023; Gou et al. 2024; Kamoi et al. 2024).

A promising response is adaptive compute allocation. Rather than applying a fixed number of correction rounds, an agent can condition further reasoning on estimated task difficulty, uncertainty, verifier disagreement, or the expected value of another intervention (Lin et al. 2026; Snell et al. 2024). Adaptive retrieval offers a related principle by reserving additional evidence acquisition for queries that require it (Jeong et al. 2024). Cost-effective self-correction therefore depends on deciding not only how to repair an error, but also whether another correction step is worth its computational cost.

6.6 Domain Dependence of Self-Correction

Empirical evidence indicates that self-correction is strongly domain-dependent because domains differ in error observability, verifier reliability, localization difficulty, feedback latency, and consequence severity. The same reflective mechanism may therefore succeed in one setting while providing little benefit, or even causing degradation, in another.

In mathematical and logical reasoning, final answers are often objectively checkable, yet deeper setup and derivation errors can remain difficult to localize (Huang et al. 2024; Li 2026). Unsupported intrinsic evaluation is particularly vulnerable when the model shares the misconception responsible for the initial error (Kamoi et al. 2024). Process-level supervision, explicit correctness verifiers, and formal checks can improve recovery by evaluating intermediate reasoning rather than relying solely on final-answer confidence (Lightman et al. 2024; Ma et al. 2025). Software engineering is comparatively amenable to corrective feedback because compilers, execution traces, and unit tests expose observable behavioral failures (Gou et al. 2024; Olausson et al. 2024a). However, passing functional tests does not establish broader correctness. Iterative repair may leave security vulnerabilities or other unmeasured software properties unresolved (Shukla et al. 2025; Feng et al. 2026).

For retrieval and multi-hop question answering, correction depends less on executable verification and more on whether an agent can detect missing information, reformulate queries, and distinguish supporting evidence from irrelevant or contradictory passages (Yu et al. 2024; Asl et al. 2025a). Retrieval can repair parametric knowledge gaps, but poor evidence selection can instead introduce query drift or retrieval-induced hallucination (Yan et al. 2024). Multimodal, physics, and embodied environments add another layer of difficulty because failures may originate in perception, spatial state estimation, physical constraints, or delayed environmental consequences (Shinn et al. 2023). In such settings, perceptual or simulation-based feedback can reveal failures that textual self-critique alone cannot observe (Shende and Camburn 2026).

Scientific agents similarly operate under partially observable and expensive verification conditions. Hypothesis generation, literature synthesis, experimentation, and interpretation form extended feedback loops in which evidence may be noisy, delayed,

or incomplete (Wei et al. 2026; Yuan et al. 2025). Finally, alignment and safety introduce a specification problem: correction can only optimize properties represented by the governing principles, evaluators, or tests (Bai et al. 2022). Improving visible task performance may therefore coexist with degradation in unmeasured safety, privacy, or security dimensions (Raza et al. 2026; Saha et al. 2026).

Overall, domains with precise, externally observable feedback generally support more reliable correction, whereas scientific, multimodal, long-horizon, and safety-critical settings remain harder because verification is incomplete, delayed, or itself uncertain.

7 Failure Modes, Safety, and Boundary Conditions

7.1 Failure to Detect Errors

Self-correction cannot begin when an agent fails to recognize that correction is needed. Error-detection reliability therefore places an upper bound on every downstream recovery mechanism. In agentic systems, undetected errors are especially consequential because they can propagate into planning, retrieval, tool use, API calls, and persistent memory before becoming observable (Pan et al. 2024; Tayebati et al. 2026). If an erroneous state is initially judged valid, subsequent diagnosis, repair, validation, and rollback mechanisms may never be triggered (Kamoi et al. 2024; Li 2026).

A major source of detection failure is the mismatch between model confidence and correctness. Because generation and self-evaluation often rely on the same underlying representations, models can remain highly confident in hallucinated or logically invalid outputs (Farquhar et al. 2024; Zhang et al. 2024). This creates three important distinctions. First, *confidence is not correctness*: fluent or high-probability generations may still be false (Farquhar et al. 2024). Second, *uncertainty detection is not equivalent to error detection*: entropy-based measures can identify unstable generations but may miss confident, systematic errors (Farquhar et al. 2024; Tayebati et al. 2026). Third, *detecting an anomaly is not the same as localizing its cause*: recognizing that an outcome is wrong does not reveal which reasoning step, tool call, or stored assumption produced the failure (Li 2026).

These limitations become more severe in long-horizon workflows. A malformed tool parameter, misleading retrieval result, or incorrect intermediate assumption may appear locally plausible and remain latent across several steps (Yu et al. 2024; Tayebati et al. 2026). The eventual failure may surface only after subsequent actions have compounded the original error or after contaminated experience has entered persistent memory (Shinn et al. 2023; Tayebati et al. 2026). Consequently, improving downstream repair algorithms is insufficient when the trigger for correction is itself poorly calibrated. Reliable agentic recovery requires detection mechanisms that combine confidence estimates with task-relevant verification and trajectory-level observability.

7.2 Detection without Successful Repair

Successful error detection does not guarantee successful recovery. A central finding across recent self-correction studies is that error detection, localization, and repair are

separable capabilities: an agent may recognize that an output or trajectory is wrong without knowing where the failure originated or how to correct it (Li 2026; Kamoi et al. 2024). This distinction is especially important in agentic systems because the observable failure may be temporally or causally distant from the responsible reasoning step, tool call, or memory state.

The first breakdown occurs when detection succeeds but localization fails. Binary reward signals, compiler failures, and unit-test results can establish that a candidate is incorrect while providing limited information about the underlying cause (Olausson et al. 2024a; Lightman et al. 2024). In long-horizon workflows, attribution becomes harder because an early planning or tool-use error may only become visible several actions later (Deshpande et al. 2025; Li et al. 2025). Without structured fault attribution, the agent may revise the wrong component of the trajectory.

A second breakdown occurs when localization succeeds but repair synthesis fails. Even when a verifier identifies a defective code region, unsupported claim, or flawed reasoning step, the generator may lack the knowledge or reasoning capability needed to produce a valid replacement (Madaan et al. 2023; Olausson et al. 2024a). Similar limitations arise in retrieval-grounded settings, where recognizing that evidence is insufficient does not ensure that subsequent retrieval or revision will resolve the missing information (Asl et al. 2025a). A repair attempt may therefore leave the original error unchanged or introduce a new one (Kamoi et al. 2024).

This functional separation helps explain the Accuracy-Correction Paradox, in which relatively strong error recognition does not translate into comparable repair success (Li 2026). Effective agentic recovery consequently requires more than a failure signal: it requires diagnostic feedback that identifies the likely source of failure and provides enough information to support a targeted repair.

7.3 Failed and Harmful Correction

Triggering a correction loop does not guarantee successful recovery. Detection, localization, and repair are separable capabilities, and unreliable feedback can make correction actively harmful (Kamoi et al. 2024; Li 2026). Three failure patterns are particularly important.

First, an agent may detect a failure without locating its cause. A trajectory can be recognized as unsuccessful while the responsible reasoning step, tool call, factual assumption, or memory item remains unclear (Deshpande et al. 2025; Li 2026). Even when localization succeeds, repair may still fail if the model lacks the knowledge or reasoning capability required to construct a valid alternative (Kamoi et al. 2024; Olausson et al. 2024a). This separation contributes to the Accuracy-Correction Paradox, where error awareness does not reliably translate into successful recovery (Li 2026).

Second, correction can degrade an initially correct state. Mandatory reflection, inaccurate self-critique, or unreliable verifier feedback may cause unnecessary revision and replace a valid answer with an inferior one (Madaan et al. 2023; Huang et al. 2024). Third, socially framed feedback introduces a distinct vulnerability. User disagreement or insistence may be treated as evidence of factual error even when no objective evidence is supplied, producing sycophantic answer switching (Chen et al. 2024; Gou et al. 2024).

Repeated interaction can further amplify these effects through critique reinforcement, answer switching, and trajectory drift (Choi et al. 2025; Shukla et al. 2025). Reliable correction therefore requires not only a repair mechanism, but also a preservation criterion that retains the current state when proposed revisions lack sufficient evidence of improvement.

7.4 Error Amplification, Memory Contamination, and Long-Horizon Propagation

In agentic systems, failed correction can propagate beyond the immediate response. An incorrect diagnosis may produce a faulty repair, weak validation may accept it, and the resulting state may subsequently influence planning, tool use, or memory (Du et al. 2026b; Tayebati et al. 2026). The result is a transition from a local error to a persistent trajectory-level failure (Shinn et al. 2023; Raza et al. 2026).

One mechanism is error amplification. Once an incorrect critique or repair is accepted, later reasoning may treat it as a valid premise, allowing errors to compound rather than remain independent (Li 2026). This risk increases when generators, critics, and validators share similar blind spots, because validation may reinforce rather than interrupt the faulty state (Kamoi et al. 2024; Wen et al. 2026).

A second mechanism is memory contamination. The problem is not memory itself, but storing and reusing unverified reflections, stale conclusions, or incorrect summaries (Shinn et al. 2023; Tan et al. 2025). Once retrieved, contaminated experience can bias future planning and generalize an earlier mistake beyond its original context (Yu et al. 2026).

Finally, long-horizon workflows separate causes from consequences. Early errors may remain locally plausible and become visible only several actions later, making the original failure difficult to attribute (Deshpande et al. 2025; Tayebati et al. 2026). Recovery is further complicated when external actions or memory updates have already committed the corrupted state (Raza et al. 2026; Deshpande et al. 2025). Robust correction therefore requires validation before memory commitment, provenance-aware memory, trajectory-level diagnosis, and rollback mechanisms rather than purely local revision.

7.5 Security, Safety, and Objective Misalignment

A more fundamental failure occurs when correction succeeds according to the available evaluator but degrades properties that the evaluator does not measure. Correction is only as complete as the objective and verifier used to judge improvement. Repeated optimization against incomplete rewards, narrow tests, or evaluator blind spots can therefore improve visible task performance while reducing broader system quality (Bai et al. 2022; Raza et al. 2026).

This problem appears across several domains. In software engineering, functionally successful repair may still retain or introduce security vulnerabilities when evaluation depends primarily on compilation or unit tests (Shukla et al. 2025; Saha et al. 2026). Robustness can similarly degrade when revisions overfit to the current verifier or task

instance, leaving the resulting policy vulnerable to distribution shift or adversarial perturbation (Raza et al. 2026). In multimodal and physical environments, syntactically valid outputs may violate spatial or physical constraints if evaluation lacks perceptual grounding (Shende and Camburn 2026).

Alignment presents the same structural problem. Rule-based critique can only enforce the dimensions represented by its principles or evaluators, leaving unmeasured safety properties outside the optimization target (Bai et al. 2022). Factuality is likewise not guaranteed by improved fluency, consistency, or confidence; refinement may produce a more persuasive response without making it more accurate (Farquhar et al. 2024; Zhang et al. 2024).

These failures resemble proxy optimization: the correction loop improves what is measured while overlooking what matters but is absent from the metric. High-stakes agents therefore require multi-dimensional validation that combines task success with security, robustness, factuality, physical constraints, and safety rather than relying on a single corrective objective.

8 Open Challenges and Research Agenda

8.1 Calibrated Self-Evaluation and Adaptive Reflection

A central open challenge in agentic AI is determining when an agent should distrust its current output, reasoning state, or trajectory enough to trigger reflection, external verification, or abstention. Unlike static prediction, agentic calibration must account for uncertainty that evolves across reasoning steps, tool interactions, retrieved evidence, and environmental observations (Ji et al. 2026a; Tayebati et al. 2026). Reflecting after every step is neither computationally efficient nor necessarily beneficial, because additional inference increases token cost and latency while creating opportunities for unnecessary revision (Kamoi et al. 2024; Olausson et al. 2024a). Conversely, insufficient inspection can allow locally plausible errors to propagate through planning, tool use, and persistent memory (Du et al. 2026a; Tayebati et al. 2026). Calibration must therefore serve as a control signal for deciding whether further corrective computation is warranted.

Addressing hallucination illustrates the difficulty of this problem. Uncertainty estimates based on token probabilities or semantic variation can reveal unstable generations, but uncertainty is not equivalent to incorrectness. Models may remain confident in false or unsupported outputs, limiting correction policies that rely on a single internal confidence signal (Farquhar et al. 2024; Zhang et al. 2024). More reliable triggering may therefore require combining heterogeneous evidence, including semantic uncertainty, process-level verifier scores, critic disagreement, execution failures, retrieval evidence, and perceptual feedback in multimodal settings (Gou et al. 2024; Ma et al. 2025; Shende and Camburn 2026; Tayebati et al. 2026). When uncertainty cannot be sufficiently reduced, selective prediction should permit abstention or escalation rather than forcing autonomous continuation (Ji et al. 2026a). Adaptive policies may consequently choose among continuing execution, reflecting locally, retrieving evidence, invoking a verifier, generating alternative candidates, or escalating unresolved cases (Yu et al. 2024; Ji et al. 2026a).

Several research directions follow from this control perspective. First, learned reflection-trigger policies could replace fixed iteration schedules by estimating the expected benefit of further reasoning or verification (Lin et al. 2026; Setlur et al. 2025). Second, calibrated multi-signal estimators should integrate internal uncertainty with external execution and perceptual evidence rather than treating any single signal as authoritative (Farquhar et al. 2024; Shende and Camburn 2026; Tayebati et al. 2026). Third, trigger thresholds should be risk-dependent: false-positive reflection wastes resources and can induce over-correction, whereas false-negative triggering permits undetected failures to propagate (Lin et al. 2026; Huang et al. 2024). Fourth, calibration should be studied over complete trajectories, since confidence may drift as context length, tool interactions, and memory retrieval accumulate (Du et al. 2026a; Tayebati et al. 2026). Finally, intervention decisions should remain auditable, with explicit records of the signals that caused reflection, abstention, or escalation. Such traceability is important for transparency and accountability in consequential agentic systems (Raza et al. 2026).

The broader challenge is therefore not simply improving confidence estimation, but learning when uncertainty, evidence, risk, and expected correction value jointly justify intervention.

8.2 Verification without Perfect Oracles

In domains such as formal reasoning or code generation, symbolic solvers, compilers, and execution tests can provide relatively direct correctness signals. A harder challenge arises when ground truth is unavailable, ambiguous, delayed, or prohibitively expensive to obtain (Pan et al. 2024; Raza et al. 2026). The central research question is therefore how an agent can determine whether a proposed correction is genuinely better when no reliable oracle exists. This problem is particularly acute in scientific hypothesis generation, strategic planning, creative tasks, real-world actions, and safety or alignment judgments, where correctness may depend on incomplete evidence or normative criteria (Bai et al. 2022; Wei et al. 2026). Multimodal settings further widen this oracle gap because validity may depend jointly on language, images, video, spatial states, simulation dynamics, or physical constraints (Raza et al. 2026; Shende and Camburn 2026). A textual verifier may therefore approve a plan that remains visually, spatially, or physically invalid when executed (Shende and Camburn 2026).

Existing automated evaluators provide only partial solutions. Intrinsic self-evaluation, model-as-judge systems, majority agreement, and peer critique can inherit correlated errors or systematic preference biases, particularly when generators and evaluators have similar capabilities and training distributions (Kamoi et al. 2024; Liang et al. 2024). Likewise, scalar reward models and narrow evaluation rubrics may capture only a subset of the intended objective, allowing apparent improvement while unmeasured properties such as factuality, safety, or physical validity deteriorate (Du et al. 2026a; Setlur et al. 2025).

Future verification architectures should therefore combine heterogeneous and partially independent evidence sources. Specialized verifier ensembles may provide collective intelligence synergy when critics contribute complementary expertise rather

than merely repeating the same judgment (**?**Zhou and Chen 2025). Cross-modal verification should compare textual reasoning against simulations, perceptual observations, spatial constraints, and environment feedback in domains where language alone is insufficient (Shende and Camburn 2026). Complex claims may also be decomposed into smaller components that can be checked against retrieved evidence, causal constraints, or counterfactual tests (Farquhar et al. 2024; Asl et al. 2025a). When automated evaluators remain uncertain or disagree, calibrated aggregation and human escalation should be treated as legitimate verification outcomes rather than forcing a binary accept-or-reject decision (Raza et al. 2026; Setlur et al. 2025).

Verification without perfect oracles also raises transparency, trust, and fairness concerns. Automated judges may systematically favor particular styles, reasoning formats, or candidate characteristics, making apparently objective acceptance decisions difficult to interpret (Liang et al. 2024; Raza et al. 2026). Future systems should therefore expose the evidence, constraints, evaluator disagreement, and uncertainty supporting each verification decision. The research objective is not a universally perfect oracle, but verification mechanisms that explicitly represent what has been checked, what remains uncertain, and when autonomous validation is insufficient (Raza et al. 2026).

8.3 Memory Integrity, Updating, and Forgetting

As agentic architectures move from transient reflection toward persistent, stateful operation, memory becomes an active component of self-correction rather than a passive repository of past interactions (Shinn et al. 2023; Du et al. 2026b; Yu et al. 2026) . This raises a fundamental research question: which corrective experiences should persist, how should their reliability be represented, and when should they be revised, forgotten, or invalidated? Retaining trajectories and reflections can help agents avoid repeating previous failures, but storing unverified, hallucinated, or poorly attributed lessons creates the risk of memory contamination (Kamoi et al. 2024; Tan et al. 2025). Once reused, incorrect memories may influence later planning, reasoning, or tool selection and allow a local diagnostic error to persist across otherwise unrelated interactions (Du et al. 2026b; Tayebati et al. 2026). Persistent self-correction therefore requires explicit policies governing what is written to memory as well as what is retrieved from it.

Several unresolved problems arise across the memory lifecycle. First, both successful and failed trajectories can contain useful corrective information, yet failure alone does not establish which component of the trajectory should be retained as a lesson (Shinn et al. 2023; Du et al. 2026b). Second, new observations may contradict previously stored reflections, requiring mechanisms that distinguish genuinely obsolete knowledge from context-dependent differences (Tan et al. 2025; Yu et al. 2026). Third, when an earlier reflection or observation is later invalidated, dependent memories and decisions may also require re-evaluation rather than treating each memory item independently (Deshpande et al. 2025; Yu et al. 2026). A related challenge is preventing narrow experiences from being overgeneralized to tasks or environments for which they provide little evidence.

Future memory architectures should consequently move beyond retrieval based primarily on semantic similarity. Provenance-aware and versioned memory could record

the source, supporting evidence, validation status, temporal context, and dependencies associated with each stored experience (Raza et al. 2026; Yu et al. 2026). Validation-before-commit policies could place uncertain reflections in temporary or quarantined storage until corroborating evidence becomes available (Setlur et al. 2025; Du et al. 2026a). Contradiction detection and reversible updating could support targeted revision when newer evidence conflicts with existing experience (Tan et al. 2025; Raza et al. 2026). Selective forgetting and temporal decay are equally important because memories can become stale as environments, tools, and task distributions change (Tan et al. 2025; Du et al. 2026a).

Memory integrity also has implications for transparency, accountability, trust, and fairness. Persistent memories should retain enough provenance to explain why a stored lesson influenced a later action and to identify which decisions depend on information that is subsequently invalidated (Raza et al. 2026). Similar mechanisms are needed to prevent biased observations or unsupported assumptions from becoming repeatedly reinforced through retrieval (Raza et al. 2026; Tan et al. 2025). As increasingly autonomous and self-regulating agentic systems emerge, reliable memory will therefore require not only greater capacity, but controlled writing, verification, revision, provenance tracking, and principled forgetting.

8.4 Long-Horizon Credit and Blame Assignment

A fundamental open challenge for autonomous agents is determining which earlier decision caused a failure observed much later in an execution trajectory. As agentic workflows increasingly combine reasoning, planning, retrieval, tool use, memory, and environmental interaction, failures are rarely reducible to isolated output-level errors (Du et al. 2026b; Tayebati et al. 2026). Terminal success-or-failure signals reveal that a trajectory was unsuccessful but provide limited information about the causal source, the earliest defective state, or the point from which recovery should begin (Deshpande et al. 2025; Li 2026). Long-horizon self-correction therefore requires credit and blame assignment across the full sequence of intermediate decisions.

The central difficulty is the temporal and structural separation between root causes and visible symptoms. An off-target retrieval query, incorrect tool argument, or flawed planning decision may appear locally reasonable and allow execution to continue for several steps (Yu et al. 2024; Tayebati et al. 2026). Its consequences may emerge only after later subgoals depend on the corrupted state, making the final failure temporally distant from the initiating error (Shinn et al. 2023; Deshpande et al. 2025). Attribution becomes harder when several individually plausible decisions interact to produce failure across planning, memory, tools, and environmental state (Du et al. 2026b; Raza et al. 2026). Multi-agent systems add another layer because errors may emerge from coordination failures, shared assumptions, or interactions among specialized agents rather than from one identifiable component (Liang et al. 2024; Raza et al. 2026). In multimodal environments, relevant causal evidence may additionally span textual reasoning, visual observations, spatial states, and physical constraints (Shende and Camburn 2026).

Future research should consequently move from outcome-level evaluation toward process-level attribution. Causal trajectory representations could record dependencies among observations, plans, retrieved evidence, tool arguments, memory accesses, and inter-agent exchanges, providing explicit execution provenance (Deshpande et al. 2025; Raza et al. 2026). Counterfactual replay could test whether replacing a particular intermediate decision changes the eventual outcome, helping distinguish causal failures from downstream symptoms (Raza et al. 2026). Hierarchical trace segmentation may further separate high-level planning errors from local implementation or tool-use failures (Li et al. 2025; Deshpande et al. 2025). Process-level evaluators could then assess intermediate states rather than relying exclusively on terminal outcomes (Lightman et al. 2024; Tayebati et al. 2026).

Credit and blame assignment is also inseparable from rollback and accountability. An agent should be able to reconstruct which observations, memories, critiques, and actions contributed to a consequential decision (Raza et al. 2026). Reliable attribution should ultimately identify not only what failed, but also the earliest recoverable checkpoint, which subsequent states were contaminated, and whether external actions remain reversible (Shinn et al. 2023; Raza et al. 2026). This would shift self-correction from local patching toward causal, trajectory-aware recovery.

8.5 Cost-Aware Correction and Escalation Policies

A critical research challenge in agentic AI is determining when corrective intervention is worth its computational, monetary, temporal, and supervisory cost. Rather than optimizing final accuracy alone, corrective control should account for the marginal improvement obtained from additional intervention:

$$\text{Efficiency Ratio} = \frac{\Delta \text{ Expected Solution Quality}}{\text{Total Intervention Cost}} \tag{2}$$

This formulation raises a broader control question: given uncertainty, expected benefit, risk, and resource constraints, should an agent continue autonomously, self-correct, acquire additional evidence, consult another evaluator, escalate to a human, or terminate? Intervention costs can accumulate through reasoning tokens, repeated model calls, verifier evaluations, retrieval operations, external tools, multi-agent communication, environmental interaction, latency, financial expenditure, and human review (Olausson et al. 2024a; Ji et al. 2026a). Applying reflection or verification indiscriminately can therefore create substantial computational scaling demands, particularly when several critics, retrievers, or repair stages are invoked at each decision point (Madaan et al. 2023; Gou et al. 2024).

Future corrective architectures should treat these alternatives as a resource-allocation problem. In multi-agent systems, collective intelligence synergy is beneficial only when additional agents provide complementary information that justifies their communication and inference costs (Liang et al. 2024; Choi et al. 2025). Similarly, adaptive control should select among accepting the current state, reflecting, resampling, retrieving evidence, invoking a verifier, consulting another agent, requesting human review, rolling back, abstaining, or terminating according to expected utility (Yu et al. 2024; Ji et al. 2026a). This requires estimating not only the probability

of error, but also its potential consequence and the expected value of each available intervention.

Human oversight is particularly important in the broader human-centric vision associated with Industry 5.0. Rather than serving exclusively as a final fallback, human expertise could be allocated selectively to cases where it provides the greatest marginal reduction in uncertainty or risk (Piccialli et al. 2025; Raza et al. 2026). More autonomous, self-regulating cognitive systems, sometimes discussed in relation to emerging Industry 6.0 concepts, would place greater responsibility on agents to allocate corrective resources independently (Alva and Pandey 2026). Such autonomy requires explicit escalation boundaries, auditable stopping criteria, and governance mechanisms that prevent cost optimization from overriding safety requirements (Shavit et al. 2023; Raza et al. 2026).

Several directions merit further investigation. Expected-value-of-computation policies could estimate whether another reasoning or verification step is likely to justify its cost (Snell et al. 2024; Lin et al. 2026). Learned stopping and risk-sensitive escalation policies could terminate low-value reflection while routing consequential, low-confidence decisions toward stronger oversight (Wang et al. 2024a; Setlur et al. 2025). Budget-aware routing could further choose among lightweight verifiers, retrieval, specialized agents, and human experts according to task complexity and anticipated error severity (Asl et al. 2025a; Piccialli et al. 2025). Finally, evaluation should compare correction strategies under matched budgets and report Pareto tradeoffs among quality, token consumption, latency, monetary cost, and safety rather than accuracy in isolation (Olausson et al. 2024a; Ji et al. 2026a).

8.6 Robustness under Distribution Shift

A fundamental open challenge for autonomous agents is ensuring that self-correction remains reliable outside the conditions under which it was developed and evaluated. The central question is whether a corrective mechanism that performs well on familiar benchmarks continues to provide positive gains when tasks, environments, tools, knowledge, users, or error distributions change. Real-world deployments may encounter shifts in task difficulty, environment dynamics, API behavior, tool availability, retrieval corpora, parametric knowledge, error types, multimodal inputs, adversarial conditions, and model calibration (Zhao et al. 2025; Tayebati et al. 2026; Raza et al. 2026). Corrective mechanisms can themselves overfit these operating conditions: a verifier calibrated on familiar failures may miss novel errors, stored experience may become obsolete, reflection triggers may become miscalibrated, and tool-dependent repair strategies may fail when external interfaces change (Kamoi et al. 2024; Olausson et al. 2024a; Zhao et al. 2025).

Robustness should therefore be evaluated across multiple components of the correction lifecycle rather than through aggregate task accuracy alone. Relevant layers include the base agent, error-detection mechanism, verifier, repair policy, persistent memory, and escalation controller (Du et al. 2026a; Tayebati et al. 2026; Raza et al. 2026). Failure in any one layer may undermine the remaining corrective pipeline. Multimodal agents introduce additional sensitivity because visual occlusion, sensor noise, changing spatial configurations, or altered physical conditions can generate failure

modes that are absent from text-only evaluation (Shende and Camburn 2026). Distribution shift may also affect trust and fairness when calibration or verification reliability varies across languages, populations, modalities, or contexts, potentially causing corrective mechanisms to reinforce rather than mitigate systematic biases (Raza et al. 2026).

Future evaluation should therefore emphasize cross-domain and temporal robustness. Benchmarks should measure whether correction remains beneficial as knowledge becomes stale and task distributions change (Zhao et al. 2025; Piccialli et al. 2025). Tool-outage and corruption tests should examine whether agents degrade gracefully when APIs fail, return altered outputs, or become adversarially manipulated (Raza et al. 2026; Tayebati et al. 2026). Verifiers should be evaluated on previously unseen error classes while calibration drift is monitored across extended trajectories (Farquhar et al. 2024; Setlur et al. 2025; Tayebati et al. 2026). Memory systems should likewise support invalidation or reduced reliance on experiences whose assumptions no longer hold, while agents should abstain or escalate when distribution shift exceeds reliable operating bounds (Tan et al. 2025; Lin et al. 2026; Raza et al. 2026).

These questions become increasingly important as agentic systems move from human-centric collaborative settings associated with Industry 5.0 toward more autonomous and self-regulating industrial architectures discussed under emerging Industry 6.0 concepts (Piccialli et al. 2025; Alva and Pandey 2026). Greater autonomy requires not merely stronger correction under benchmark conditions, but correction mechanisms whose detection, verification, memory, repair, and escalation policies remain dependable when their original operating assumptions no longer hold (Alva and Pandey 2026; Raza et al. 2026).

9 Conclusion

This review examined self-correction in agentic AI as a system-level control and recovery problem rather than as a simple extension of reasoning generation or answer refinement. The first central conclusion is that reasoning, reflection, verification, and correction are related but functionally distinct processes. Generating additional reasoning steps, producing a critique, or expressing uncertainty does not by itself constitute self-correction. A process becomes genuinely corrective only when information about a previous output, reasoning state, action, trajectory, or memory is evaluated and subsequently used to modify agent behavior. This distinction is particularly important for agentic systems, where failures may propagate across multiple decisions rather than remaining confined to a single generated response.

Second, the reviewed evidence indicates that reliable correction depends more strongly on the quality, relevance, and independence of feedback and verification than on the number of reflection iterations. Repeated intrinsic self-evaluation can be useful when an agent already possesses the knowledge and capability required for repair, but it can also reproduce the same epistemic blind spots responsible for the initial failure, misdiagnose correct outputs, or induce unnecessary revision. Tool execution, retrieved evidence, environmental observations, test suites, and explicit verifiers often

provide stronger corrective signals because they introduce information beyond unsupported introspection. Nevertheless, externally grounded feedback is not inherently reliable: incomplete tests, noisy retrieval, weak evaluators, or misspecified objectives can validate incorrect repairs or optimize only a narrow subset of the intended behavior.

Third, agentic self-correction must extend beyond final-answer revision. Reliable recovery may require intervention at the level of intermediate reasoning, plans, retrieval queries, tool calls, external actions, multi-step trajectories, and persistent memory. Correction therefore involves not only detecting and repairing an error, but also determining whether a proposed repair should be validated, committed, rolled back, stored in memory, discarded, or escalated for external oversight. Long-horizon operation makes these decisions especially important because locally plausible errors or incorrect reflections can persist, compound, and influence future behavior.

Several limitations of the current evidence base should also be recognized. Existing studies employ heterogeneous tasks, models, feedback mechanisms, evaluation metrics, computational budgets, and definitions of self-correction, which makes direct cross-study comparison difficult. Reported improvements may also depend strongly on benchmark-specific verifier availability, while many open-ended, scientific, multimodal, and real-world tasks lack reliable ground-truth oracles. Moreover, the rapid evolution of agentic architectures means that recent findings frequently appear first as preprints, and evidence on long-term memory integrity, trajectory-level recovery, distribution shift, and sustained autonomous operation remains comparatively limited. These constraints caution against interpreting improvements reported in individual benchmarks as evidence of generally reliable self-correction.

Future research should therefore move beyond fixed reflection loops toward adaptive corrective control. Priority directions include calibrated mechanisms for deciding when correction is necessary, verification under imperfect or absent oracles, causal localization of failures across long trajectories, provenance-aware memory updating and forgetting, cost-aware allocation of reasoning and verifier calls, robust rollback mechanisms, and principled human escalation. Evaluation should likewise move toward matched-compute comparisons and broader measures that jointly consider task performance, factuality, robustness, security, safety, latency, and computational cost.

Ultimately, reliable self-correction is both a performance problem and a safety and control problem. The objective is not simply to make autonomous agents reflect more frequently, but to ensure that they can determine *when* correction is warranted, *what* evidence should guide it, *which* state should be modified, *whether* the resulting repair is trustworthy, and *when* autonomous recovery should stop or escalate. Progress toward dependable agentic AI will therefore require corrective mechanisms that are calibrated, verifier-grounded, memory-safe, cost-aware, and robust enough to prevent local failures from becoming persistent system-level errors.

Statements and Declarations

Funding. The authors did not receive support from any organization for the submitted work.

Competing interests. The authors have no relevant financial or non-financial interests to disclose.

Author contributions. Most Anamika Akter Zedni: conceptualization, literature identification, writing, original draft. Md Minhazul Islam: literature identification, writing, original draft. Md Monzurul Islam: supervision, writing, review and editing. Amith Khandakar: supervision, writing, review and editing. Mohamed Arselene Ayari: writing, review and editing. All authors read and approved the final manuscript.

Data availability. No new experimental datasets were generated. This is a critical review of previously published literature. All sources analysed in the study are cited in the reference list.

Ethics approval. Not applicable. The work did not involve human participants or animals. **Use of generative AI.** Large language model and agentic AI tools were used to assist with literature organisation, language editing, and manuscript formatting. The authors reviewed, verified, and approved all scientific claims, interpretations, citations, and the final text, and take full responsibility for the content of this manuscript.

Consent to participate. Not applicable.

Consent to publish. Not applicable.

References

Alva LR, Pandey B (2026) Agentic ai systems in the age of generative models: architectures, cloud scalability, and real-world applications. Artificial Intelligence Review 59(3):88. https://doi.org/10.1007/s10462-025-11458-6, URL https://doi.org/10.1007/s10462-025-11458-6

Asai A, Wu Z, Wang Y, et al (2024) Self-rag: Learning to retrieve, generate, and critique through self-reflection. In: Kim B, Yue Y, Chaudhuri S, et al (eds) International Conference on Learning Representations, pp 9112–9141, URL https://proceedings.iclr.cc/paper_files/paper/2024/file/25f7be9694d7b32d5cc670927b8091e1-Paper-Conference.pdf

Asl MA, Asgari-Bidhendi M, Minaei-Bidgoli B (2025a) FAIR-RAG: Faithful adaptive iterative refinement for retrieval-augmented generation. arXiv preprint arXiv:251022344 URL https://arxiv.org/pdf/2510.22344v1, arXiv:2510.22344 [cs.CL]

Asl MA, Asgari-Bidhendi M, Minaei-Bidgoli B (2025b) Fair-rag: Faithful adaptive iterative refinement for retrieval-augmented generation. arXiv preprint arXiv:251022344

Bai Y, Kadavath S, Kundu S, et al (2022) Constitutional ai: Harmlessness from ai feedback. arXiv preprint arXiv:221208073 URL https://arxiv.org/abs/2212.08073

Chen W, Huang Z, Xie L, et al (2024) From yes-men to truth-tellers: Addressing sycophancy in large language models with pinpoint tuning. In: Proceedings of the 41st International Conference on Machine Learning

Chen X, Zhao A, Xia H, et al (2025) Reasoning beyond language: A comprehensive survey on latent chain-of-thought reasoning. arXiv preprint arXiv:250516782 arXiv:2505.16782 [cs.CL]

Choi HK, Zhu J, Li S (2025) Debate or vote: Which yields better decisions in multi-agent large language models? In: Belgrave D, Zhang C, Lin H, et al (eds) Advances in Neural Information Processing Systems, vol 38, Main Conference. Curran Associates, Inc., pp 101732–101764, https://doi.org/10.52202/085713-3405, URL https://proceedings.neurips.cc/paper_files/paper/2025/file/934252acd87f254d5d4672fbde283bd2-Paper-Conference.pdf

Chowa SS, Alvi R, Rahman SS, et al (2026) From language to action: A review of large language models as autonomous agents and tool users. Artificial Intelligence Review 59:71. https://doi.org/10.1007/s10462-025-11471-9

Deshpande D, Gangal V, Mehta H, et al (2025) Trail: Trace reasoning and agentic issue localization. arXiv preprint arXiv:250508638

Du S, Zhao J, Shi J, et al (2026a) A survey on the optimization of large language model-based agents. ACM Computing Surveys 58(9):1–37. https://doi.org/10.1145/3789261

Du S, Zhao J, Shi J, et al (2026b) A survey on the optimization of large language model-based agents. ACM Computing Surveys 58(9):1–37

Farquhar S, Kossen J, Kuhn L, et al (2024) Detecting hallucinations in large language models using semantic entropy. Nature 630(8017):625–630. https://doi.org/10.1038/s41586-024-07421-0, URL https://doi.org/10.1038/s41586-024-07421-0

Feng D, Yun B, Wang AY (2026) From junior to senior: Allocating agency and navigating professional growth in agentic ai-mediated software engineering. In: Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems. ACM, https://doi.org/10.1145/3772318.3791642

Feng Z, Su J, Zheng J, et al (2025) M-MAD: Multidimensional multi-agent debate for advanced machine translation evaluation. In: Che W, Nabende J, Shutova E, et al (eds) Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Vienna, Austria, pp 7084–7107, https://doi.org/10.18653/v1/2025.acl-long.351, URL https://aclanthology.org/2025.acl-long.351/

Gou Z, Shao Z, Gong Y, et al (2024) CRITIC: Large language models can self-correct with tool-interactive critiquing. arXiv preprint arXiv:230511738 URL https://arxiv.org/abs/2305.11738

Hu C, Hu Y, Cao H, et al (2024) Teaching language models to self-improve by learning from language feedback. arXiv:2406.07168

Huang J, Chen X, Mishra S, et al (2024) Large language models cannot self-correct reasoning yet. In: Kim B, Yue Y, Chaudhuri S, et al (eds) International Conference on Learning Representations, pp 32808–32824, URL https://proceedings.iclr.cc/paper_files/paper/2024/file/8b4add8b0aa8749d80a34ca5d941c355-Paper-Conference.pdf

Jeong S, Baek J, Cho S, et al (2024) Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In: Duh K, Gomez H, Bethard S (eds) Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). Association for Computational Linguistics, Mexico City, Mexico, pp 7036–7050, https://doi.org/10.18653/v1/2024.naacl-long.389, URL https://aclanthology.org/2024.naacl-long.389/

Ji Y, Li J, Xiang Y, et al (2026a) A survey of test-time compute: From intuitive inference to deliberate reasoning. Computational Linguistics pp 1–55. https://doi.org/10.1162/COLI.a.629, URL https://doi.org/10.1162/COLI.a.629, https://direct.mit.edu/coli/article-pdf/doi/10.1162/COLI.a.629/2602905/coli.a.629.pdf

Ji Y, Li J, Xiang Y, et al (2026b) A survey of test-time compute: From intuitive inference to deliberate reasoning. Computational Linguistics pp 1–51

Jiang J, Wang F, Shen J, et al (2026) A survey on large language models for code generation. ACM Transactions on Software Engineering and Methodology 35(2):1–72. https://doi.org/10.1145/3747588

Jiang Z, Xu F, Gao L, et al (2023) Active retrieval augmented generation. In: Bouamor H, Pino J, Bali K (eds) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Singapore, pp 7969–7992, https://doi.org/10.18653/v1/2023.emnlp-main.495, URL https://aclanthology.org/2023.emnlp-main.495/

Kamoi R, Zhang Y, Zhang N, et al (2024) When can LLMs actually correct their own mistakes? A critical survey of self-correction of LLMs. Transactions of the Association for Computational Linguistics 12:1417–1440. https://doi.org/10.1162/tacl_a_00713

Kumar A, Zhuang V, Agarwal R, et al (2025) Training language models to self-correct via reinforcement learning. In: Yue Y, Garg A, Peng N, et al (eds) International Conference on Learning Representations, pp 54523–54549, URL https://proceedings.iclr.cc/paper_files/paper/2025/file/871ac99fdc5282d0301934d23945ebaa-Paper-Conference.pdf

Lee H, Oh S, Kim J, et al (2025) ReVISE: Learning to refine at test-time via intrinsic self-verification. In: Proceedings of the 42nd International Conference on Machine Learning, Proceedings of Machine Learning Research, vol 267. PMLR, pp 33616–33634

Lewis P, Perez E, Piktus A, et al (2020) Retrieval-augmented generation for knowledge-intensive nlp tasks. In: Larochelle H, Ranzato M, Hadsell R, et al (eds) Advances in Neural Information Processing Systems, vol 33. Curran Associates, Inc., pp 9459–9474, URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

Li H, Shi Y, Lin S, et al (2025) SWE-Debate: Competitive multi-agent debate for software issue resolution. arXiv preprint arXiv:250723348 URL https://arxiv.org/abs/2507.23348

Li Y (2025) Decomposing LLM self-correction: The accuracy-correction paradox and error depth hypothesis. arXiv preprint arXiv:260100828 URL https://arxiv.org/pdf/2601.00828v1, arXiv:2601.00828 [cs.AI]

Li Y (2026) Decomposing llm self-correction: The accuracy-correction paradox and error depth hypothesis. arXiv preprint arXiv:260100828 URL https://arxiv.org/abs/2601.00828

Liang T, He Z, Jiao W, et al (2024) Encouraging divergent thinking in large language models through multi-agent debate. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, pp 17889–17904, https://doi.org/10.18653/v1/2024.emnlp-main.992

Lightman H, Kosaraju V, Burda Y, et al (2024) Let's verify step by step. International Conference on Learning Representations (ICLR) 2024:39578–39601

Lin J, Zeng X, Zhu J, et al (2026) Plan and budget: Effective and efficient test-time scaling on reasoning large language models. In: The Fourteenth International Conference on Learning Representations, URL https://openreview.net/forum?id=ctspw4CqbS

Ma R, Wang P, Liu C, et al (2025) S^2R: Teaching LLMs to self-verify and self-correct via reinforcement learning. In: Che W, Nabende J, Shutova E, et al (eds) Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Vienna, Austria, pp 22632–22654, https://doi.org/10.18653/v1/2025.acl-long.1104, URL https://aclanthology.org/2025.acl-long.1104/

Madaan A, Tandon N, Gupta P, et al (2023) Self-refine: Iterative refinement with self-feedback. arXiv preprint arXiv:230317651 URL https://arxiv.org/abs/2303.17651

Olausson TX, Inala JP, Wang C, et al (2024a) Is self-repair a silver bullet for code generation? In: Kim B, Yue Y, Chaudhuri S, et al (eds) International Conference on Learning Representations, pp 36545–36593, URL https://proceedings.iclr.cc/paper_files/paper/2024/file/9ddc141bdbf9d1db510cefff56c586ad-Paper-Conference.pdf

Olausson TX, Inala JP, Wang C, et al (2024b) Is self-repair a silver bullet for code generation? In: International Conference on Learning Representations, pp 36545–36593

Pan L, Saxon M, Xu W, et al (2024) Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. Transactions of the Association for Computational Linguistics 12:484–506. https://doi.org/10.1162/tacl_a_00660

Piccialli F, Chiaro D, Sarwar S, et al (2025) Agentai: A comprehensive survey on autonomous agents in distributed ai for industry 4.0. Expert Systems with Applications 291:128404. https://doi.org/https://doi.org/10.1016/j.eswa.2025.128404, URL https://www.sciencedirect.com/science/article/pii/S0957417425020238

Plaat A, van Duijn M, van Stein N, et al (2025a) Agentic large language models: A survey. Journal of Artificial Intelligence Research https://doi.org/10.1613/jair.1.18675, URL https://arxiv.org/pdf/2503.23037v3, arXiv:2503.23037 [cs.AI]

Plaat A, Wong A, Verberne S, et al (2025b) Multi-step reasoning with large language models, a survey. ACM Computing Surveys 58(6):1–35. https://doi.org/10.1145/3774896

Ranaldi L, Freitas A (2024) Self-refine instruction-tuning for aligning reasoning in language models. In: Al-Onaizan Y, Bansal M, Chen YN (eds) Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Miami, Florida, USA, pp 2325–2347, https://doi.org/10.18653/v1/2024.emnlp-main.139, URL https://aclanthology.org/2024.emnlp-main.139/

Raza S, Sapkota R, Karkee M, et al (2026) TRiSM for agentic AI: A review of trust, risk, and security management in LLM-based agentic multi-agent systems. AI Open 7:71–95. https://doi.org/10.1016/j.aiopen.2026.02.006

Renze M, Guven E (2024) Self-reflection in large language model agents: Effects on problem-solving performance. 2024 2nd International Conference on Foundation and Large Language Models (FLLM) pp 516–525. https://doi.org/10.1109/FLLM63129.2024.10852426

Saha S, Chen J, Mayers S, et al (2026) Breaking the code: Security assessment of ai code agents through systematic jailbreaking attacks. Transactions of the Association for Computational Linguistics 14:2081–2102. https://doi.org/10.1162/TACL.a.792, URL https://doi.org/10.1162/TACL.a.792, https://direct.mit.edu/tacl/article-pdf/doi/10.1162/TACL.a.792/2624547/tacl.a.792.pdf

Setlur A, Rajaraman N, Levine S, et al (2025) Scaling test-time compute without verification or rl is suboptimal. arXiv preprint arXiv:250212118

Shavit Y, Agarwal S, Brundage M, et al (2023) Practices for governing agentic ai systems. Research Paper, OpenAI pp 1–23

Shende P, Camburn B (2026) Perceptual self-reflection in agentic physics simulation code generation. arXiv preprint arXiv:260212311 URL https://arxiv.org/pdf/2602.12311v1, arXiv:2602.12311 [cs.SE]

Shinn N, Cassano F, Gopinath A, et al (2023) Reflexion: language agents with verbal reinforcement learning. In: Oh A, Naumann T, Globerson A, et al (eds) Advances in Neural Information Processing Systems, vol 36. Curran Associates, Inc., pp 8634–8652, https://doi.org/10.52202/075280-0377, URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf

Shukla S, Joshi H, Syed R (2025) Security degradation in iterative ai code generation: A systematic analysis of the paradox. In: 2025 IEEE International Symposium on Technology and Society (ISTAS), pp 1–8, https://doi.org/10.1109/ISTAS65609.2025.11269659

Snell C, Lee J, Xu K, et al (2024) Scaling llm test-time compute optimally can be more effective than scaling model parameters. arXiv preprint arXiv:240803314

Song X, Wu Y, Wang W, et al (2025) ProgCo: Program helps self-correction of large language models. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Association for Computational Linguistics, pp 944–959, https://doi.org/10.18653/v1/2025.acl-short.73

Sun J, Zheng C, Xie E, et al (2025) A survey of reasoning with foundation models: Concepts, methodologies, and outlook. ACM Computing Surveys 57(11):1–43. https://doi.org/10.1145/3729218

Tan Z, Yan J, Hsu IH, et al (2025) In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In: Che W, Nabende J, Shutova E, et al (eds) Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Vienna, Austria, pp 8416–8439, https://doi.org/10.18653/v1/2025.acl-long.413, URL https://aclanthology.org/2025.acl-long.413/

Tayebati S, Kumar D, Darabi N, et al (2026) Tracer: Trajectory risk aggregation for critical episodes in agentic reasoning. arXiv preprint arXiv:260211409

Trivedi H, Balasubramanian N, Khot T, et al (2023) Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In: Rogers A, Boyd-Graber J, Okazaki N (eds) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Toronto, Canada, pp 10014–10037, https://doi.org/10.18653/v1/2023.acl-long.557, URL https://aclanthology.org/2023.acl-long.557/

Wang J, Jain S, Zhang D, et al (2024a) Reasoning in token economies: Budget-aware evaluation of LLM reasoning strategies. In: Al-Onaizan Y, Bansal M, Chen YN (eds) Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Miami, Florida, USA, pp 19916–19939, https://doi.org/10.18653/v1/2024.emnlp-main.1112, URL https://aclanthology.org/2024.emnlp-main.1112/

Wang L, Xu W, Lan Y, et al (2023a) Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In: Rogers A, Boyd-Graber J, Okazaki N (eds) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Toronto, Canada, pp 2609–2634, https://doi.org/10.18653/v1/2023.acl-long.147, URL https://aclanthology.org/2023.acl-long.147/

Wang L, Ma C, Feng X, et al (2024b) A survey on large language model based autonomous agents. Frontiers of Computer Science 18(6):186345. https://doi.org/10.1007/s11704-024-40231-1

Wang X, Wei J, Schuurmans D, et al (2023b) Self-consistency improves chain of thought reasoning in language models. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, URL https://openreview.net/forum?id=1PL1NIMMrw

Wei J, Wang X, Schuurmans D, et al (2022) Chain-of-thought prompting elicits reasoning in large language models. In: Advances in Neural Information Processing Systems

Wei T, Sachdeva N, Coleman B, et al (2025) Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. arXiv preprint arXiv:251120857

Wei T, Li TW, Liu Z, et al (2026) Agentic reasoning for large language models. arXiv preprint arXiv:260112538

Wen X, Lou J, Lu X, et al (2026) Scalable oversight for superhuman ai via recursive self-critiquing. arXiv:2502.04675

Xi Z, Chen W, Guo X, et al (2025) The rise and potential of large language model based agents: A survey. Science China Information Sciences 68(2):121101. https://doi.org/10.1007/s11432-024-4222-0

Xiong G, Jin Q, Wang X, et al (2025) RAG-Gym: Systematic optimization of language agents for retrieval-augmented generation. arXiv preprint arXiv:250213957 URL https://arxiv.org/pdf/2502.13957v2, arXiv:2502.13957 [cs.CL]

Xu W, Liang Z, Mei K, et al (2025) A-mem: Agentic memory for llm agents. In: Belgrave D, Zhang C, Lin H, et al (eds) Advances in Neural Information Processing Systems, vol 38, Main Conference. Curran Associates, Inc., pp 17577–17604, https:

//doi.org/10.52202/085713-0593, URL https://proceedings.neurips.cc/paper_files/paper/2025/file/19909c36f51abc4856b4560aff3d36d6-Paper-Conference.pdf

Yan S, Yang X, Huang Z, et al (2026) Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In: Liakata M, Moreira VP, Zhang J, et al (eds) Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, San Diego, California, United States, pp 12805–12825, https://doi.org/10.18653/v1/2026.acl-long.583, URL https://aclanthology.org/2026.acl-long.583/

Yan SQ, Gu JC, Zhu Y, et al (2024) Corrective retrieval augmented generation. arXiv preprint arXiv:240115884

Yao S, Zhao J, Yu D, et al (2022) ReAct: Synergizing reasoning and acting in language models. arXiv preprint arXiv:221003629 URL https://arxiv.org/pdf/2210.03629v3, arXiv:2210.03629 [cs.CL]

Yao S, Yu D, Zhao J, et al (2023a) Tree of thoughts: Deliberate problem solving with large language models. In: Oh A, Naumann T, Globerson A, et al (eds) Advances in Neural Information Processing Systems, vol 36. Curran Associates, Inc., pp 11809–11822, https://doi.org/10.52202/075280-0517, URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf

Yao S, Zhao J, Yu D, et al (2023b) ReAct: Synergizing reasoning and acting in language models. URL https://arxiv.org/abs/2210.03629

Yu T, Zhang S, Feng Y (2024) Auto-RAG: Autonomous retrieval-augmented generation for large language models. arXiv preprint arXiv:241119443 URL https://arxiv.org/abs/2411.19443

Yu Y, Yao L, Xie Y, et al (2026) Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In: Liakata M, Moreira VP, Zhang J, et al (eds) Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, San Diego, California, United States, pp 21457–21483, https://doi.org/10.18653/v1/2026.acl-long.981, URL https://aclanthology.org/2026.acl-long.981/

Yuan J, Yan X, Zhang B, et al (2025) Dolphin: Moving towards closed-loop auto-research through thinking, practice, and feedback. In: Che W, Nabende J, Shutova E, et al (eds) Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Vienna, Austria, pp 21768–21789, https://doi.org/10.18653/v1/2025.acl-long.1056, URL https://aclanthology.org/2025.acl-long.1056/

Zhang K, Li Z, Li J, et al (2023) Self-edit: Fault-aware code editor for code generation. In: Rogers A, Boyd-Graber J, Okazaki N (eds) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Toronto, Canada, pp 769–787, https://doi.org/10.18653/v1/2023.acl-long.45, URL https://aclanthology.org/2023.acl-long.45/

Zhang Q, Wang D, Qian H, et al (2025) Understanding the dark side of LLMs' intrinsic self-correction. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, pp 27066–27101, https://doi.org/10.18653/v1/2025.acl-long.1314

Zhang X, Peng B, Tian Y, et al (2024) Self-alignment for factuality: Mitigating hallucinations in llms via self-evaluation. arXiv:2402.09267

Zhao C, Tan Z, Ma P, et al (2025) Is chain-of-thought reasoning of LLMs a mirage? a data distribution lens. arXiv preprint arXiv:250801191 URL https://arxiv.org/pdf/2508.01191v4, arXiv:2508.01191 [cs.AI]

Zhao C, Tan Z, Ma P, et al (2026) Is chain-of-thought reasoning of LLMs a mirage? a data distribution lens. In: Liakata M, Moreira VP, Zhang J, et al (eds) Findings of the Association for Computational Linguistics: ACL 2026. Association for Computational Linguistics, San Diego, California, United States, pp 15231–15261, https://doi.org/10.18653/v1/2026.findings-acl.749, URL https://aclanthology.org/2026.findings-acl.749/

Zhou Y, Chen Y (2025) Adaptive heterogeneous multi-agent debate for enhanced educational and factual reasoning in large language models. Journal of King Saud University: Computer and Information Sciences 37:330. https://doi.org/10.1007/s44443-025-00353-3

Zhou Z, Qu A, Wu Z, et al (2026) Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. In: Vondrick C, Hariharan B, Raffel C, et al (eds) International Conference on Learning Representations, pp 58413–58438, URL https://proceedings.iclr.cc/paper_files/paper/2026/file/5fc8b3bdfbb9167b5144df5d3fae4616-Paper-Conference.pdf

Zhuge M, Zhao C, Ashley DR, et al (2025) Agent-as-a-judge: Evaluate agents with agents. In: Singh A, Fazel M, Hsu D, et al (eds) Proceedings of the 42nd International Conference on Machine Learning, Proceedings of Machine Learning Research, vol 267. PMLR, pp 80569–80611, URL https://proceedings.mlr.press/v267/zhuge25a.html