EvoFSM: Controllable Self-Evolution for Deep Research with Finite State Machines

Anonymous ACL submission

Abstract

While LLM-based agents have shown promise for deep research, most existing approaches rely on fixed workflows that struggle to adapt to real-world, open-ended queries. Recent work therefore explores self-evolution by allowing agents to rewrite their own code or prompts to improve problem-solving ability, but unconstrained optimization often triggers instability, hallucinations, and instruction drift. We propose **EvoFSM**, a structured self-evolving framework that achieves both adaptability and control by evolving an explicit Finite State Machine (FSM) instead of relying on free-form rewriting. EvoFSM decouples the optimization space into macroscopic *Flow* (state-transition logic) and microscopic *Skill* (state-specific behaviors), enabling targeted improvements under clear behavioral boundaries. Guided by a critic mechanism, EvoFSM refines the FSM through a small set of constrained operations, and further incorporates a self-evolving memory that distills successful trajectories as reusable priors and failure patterns as constraints for future queries. Extensive evaluations on five multi-hop QA benchmarks demonstrate the effectiveness of EvoFSM. In particular, EvoFSM reaches 58.0% accuracy on the DeepSearch benchmark. Additional results on interactive decision-making tasks further validate its generalization.

1 Introduction

Large Language Model (LLM) based agents have demonstrated remarkable progress in automating Deep Research tasks (Li et al., 2025a; Jin et al., 2025; Li et al., 2025b). However, dominant approaches still rely heavily on pre-defined, static procedural paradigms, typically characterized by fixed, iterative "toolcall-generation-reflection" pipelines (Wei et al., 2025; Zhao et al., 2025). The inherent rigidity of these structures struggles to adapt to the dynamic query paths required by real-world, open-ended research problems, significantly constraining system flexibility and generalization capabilities.

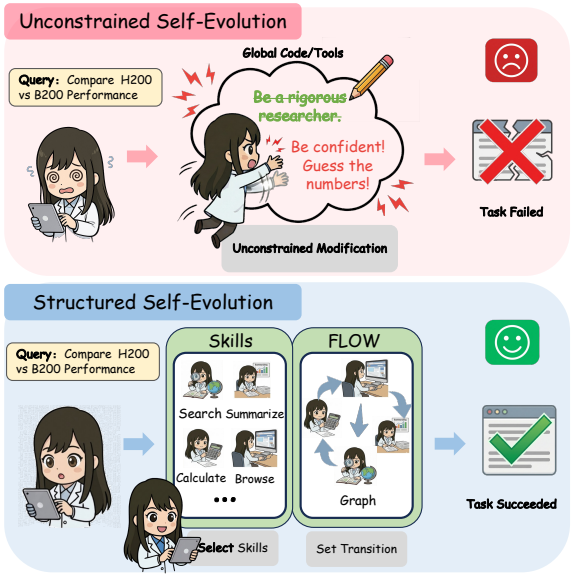

Figure 1: Comparison of unconstrained self-evolution and our structured self-evolution.

To address the limitations of static workflows, the research community has pivoted towards "Self-Evolution" mechanisms (Deppen et al., 2024; Wang et al., 2024; Li et al., 2024), aiming to empower agents to dynamically adjust strategies based on task feedback. However, existing self-evolution methods often employ "unconstrained rewriting," where a Meta-Agent is granted the freedom to rewrite the entire system prompt or tools of worker agents (Schmidhuber et al., 2024). This unconstrained approach poses significant stability challenges, as agents may suffer from core instruction drift, hallucination, or the corruption of robust functional modules during self-modification, potentially causing the system to deteriorate rather than improve (Shao et al., 2024), as shown in Figure 1.

In human organizational collaboration, efficiency gains typically stem from the joint optimization of two dimensions: workflow planning and

specific execution skills. Drawing inspiration from this, we posit that agent evolution should not be a chaotic global rewriting process, but rather a form of structured and controllable optimization. We explicitly decouple the optimization space into two orthogonal dimensions: *Flow* (i.e., macroscopic transition logic) and *Skill* (i.e., microscopic node-specific capabilities). This decomposition transforms the self-evolving process from chaotic modification into a targeted, structured self-evolution mechanism.

In this paper, we propose **EvoFSM**, a structured self-evolving framework designed for deep research and general enough to be applied across diverse tasks and domains. Specifically, EvoFSM first models the complex retrieval-reasoning process as an explicit Finite State Machine (FSM) (Wu et al., 2024). By decomposing uncertain, long-horizon tasks into a state graph with clear transition logic, we establish deterministic behavioral boundaries that guarantee foundational stability. Second, to mitigate the uncontrollability of evolution, EvoFSM employs a "Structured Self-Evolution" mechanism. Rather than allowing free-form rewriting, we restrict the system to modifying the FSM topology only via a set of atomic operations guided by a critic mechanism. This targeted adjustment ensures the system flexibly adapts to new tasks without compromising functional integrity. Finally, to accumulate experience across tasks and enable continual improvement, we introduce a self-evolving memory mechanism that stores successful strategies as priors and failure patterns as constraints for future queries.

By integrating these mechanisms, EvoFSM establishes a structured, self-evolving system. In summary, our main contributions are as follows:

- We introduce EvoFSM, a structured self-evolution framework that models task solving as an FSM, enabling controllable evolution through explicit states and transition logic rather than unconstrained rewriting.
- We introduce a self-evolving memory mechanism that accumulates experience across tasks and uses it to guide future evolution, enabling continual improvement beyond per-task optimization.
- Extensive evaluations on five multi-hop QA benchmarks demonstrate the effectiveness of EvoFSM, and results on two interactive decision-making datasets further support its generality across settings.

2 Related Work

2.1 Deep Research Agents

To tackle open-ended and complex information-seeking tasks, recent work has proposed a range of deep research agents. Search-o1 (Li et al., 2025a) integrates agentic RAG with a Reason-in-Documents module to support on-demand knowledge acquisition during long-horizon reasoning. Search-R1 (Jin et al., 2025) further trains LLMs via reinforcement learning to generate search queries and interact with search engines over multiple turns. For richer web interaction, WebThinker (Li et al., 2025b) enables autonomous browsing and report drafting, while WebAgent-R1 (Wei et al., 2025) uses end-to-end reinforcement learning for decision-making in dynamic web environments. RepoMaster (Wang et al., 2025) extends agentic exploration to software engineering through repository structure graphs. Efficiency-oriented methods such as ReSearch (Sun et al., 2024) and ZeroSearch (Xu et al., 2024) improve ranking or reduce supervision. In contrast, EvoFSM uses an explicit FSM to organize deep research, decomposing the process into specialized states with clear transition logic. Crucially, it can adapt the FSM to each query by evolving the workflow based on task feedback and prior experience.

2.2 Self-Evolving Agents

Recent work has explored self-evolving agents that improve during deployment (Shinn et al., 2023; Madaan et al., 2023; Yang et al., 2023). In code and tool domains, LIVE-SWE-AGENT (Deppen et al., 2024) and STELLA (Wang et al., 2024) allow agents to synthesize tools or update their scaffolding on the fly, while ShieldLearner (Ni et al., 2025) learns reusable defenses against jailbreaks. Self-evolution has also been extended to multi-agent settings, e.g., MorphAgent (Li et al., 2024) adapts agent profiles and CoMAS (Wu et al., 2025) drives decentralized co-evolution via interaction rewards. More radical proposals such as the Huxley-Gödel Machine (Schmidhuber et al., 2024) pursue open-ended self-improvement through recursive modifications, and SE-Agent (Lin et al., 2025) optimizes reasoning trajectories through multi-step interaction. Despite these advances, many approaches rely on unconstrained prompt rewriting or global architecture search with limited structural guardrails,

which can lead to instability and failed evolution (Shao et al., 2024). In contrast, our EvoFSM introduces explicit structural guardrails and experience-guided evolution to keep self-improvement stable. It adapts its strategy to each query while preserving core functionality, mitigating the instruction drift and instability commonly observed in unconstrained self-evolution.

3 Methodology

In this section, we present the EvoFSM framework, as illustrated in Figure 2. EvoFSM comprises three core components: FSM Initialization, Structured Self-Evolution, and the Self-Evolving Memory Mechanism. We detail each of these components in the following subsections.

3.1 FSM Initialization

While recent works have explored self-evolution to improve adaptability, these approaches often rely on an unconstrained rewriting mechanism (Shao et al., 2024), which frequently leads to hallucination and instruction drift. To address the uncontrollable evolution, we model the deep research process as a deterministic yet dynamic FSM. Fundamentally, the FSMs serve as a directed behavioral graph, where nodes represent specialized action phases (e.g., Search, Browse) and edges define the logical transitions between them based on runtime context. This graph-based structure grounds the agent's behaviors, serving as a robust backbone that enables the precise, targeted self-evolving optimization described in Section 3.2. We formalize this graph-based system as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{T}, \mathcal{I}, \mathcal{C} \rangle$, structured to align with the dual optimization dimensions of *Flow* and *Skill*:

- $\mathcal{S}$ represents the extensible set of cognitive states (nodes), initially comprising fundamental capabilities such as *Problem Decomposition*, *Search*, and *Browsing*.
- $\mathcal{T} : \mathcal{S} \times \mathcal{H} \rightarrow \mathcal{S}$ constitutes the macroscopic Flow logic. It functions as a dynamic router that determines the next state s_{t+1} based on the current state s_t and runtime context $\mathcal{H}$, thereby shaping the sequential execution path of the collaboration.
- $\mathcal{I}$ represents the set of node-specific system prompts, defining the microscopic Skill dimension. Each instruction $i \in \mathcal{I}$ specifies the operational guidelines and expertise for its corresponding agent.
- $\mathcal{C}$ denotes the Critic Mechanism, a supervisory module that evaluates the final output against the user query. Unlike passive evaluation metrics, $\mathcal{C}$ serves as the active trigger for the self-evolving process, identifying specific failure modes (e.g., lack of quantitative evidence or logical inconsistencies)

Based on the formal definitions above, the FSM initialization proceeds as follows:

When the system receives a user query q, the framework first retrieves the top-k relevant historical strategies from the Experience Pool $\mathcal{E}$, which accumulates the system's historical successful and failed trajectories from previously executed tasks and detailed in Section 3.3, to initialize the FSM configuration, denoted as $\mathcal{M}_{init}$. During the execution phase, the agents first perform the node-specific actions (Skill $\mathcal{I}$) defined by the current state. Since the topology of $\mathcal{M}_{init}$ remains static during this phase, the agent may encounter capability gaps or become trapped in unproductive loops when confronting novel, complex scenarios, especially in deep research scenarios. Consequently, following the execution phase, the critic mechanism $\mathcal{C}$ validates the system's output against the query requirements, where detected failures (e.g., hallucination or incomplete reasoning) serve as trigger signals to activate the Structured Self-Evolution (detailed in Section 3.2) to dynamically reconfigure the FSM structure.

3.2 Structured Self-Evolution

Departing from unconstrained modification, EvoFSM constrains the evolution process to a structured mechanism grounded in discrete atomic operations. By leveraging the explicit components of the tuple $\mathcal{M}$, we decouple the optimization space into two dimensions. First, *Flow Operators*, denoted as $\mathcal{O}_{flow}$, reconfigure the collaboration topology governed by the transition function $\mathcal{T}$. Second, *Skill Refinement*, denoted as $\mathcal{O}_{skill}$, enhances the individual expertise of each agent governed by the instruction set $\mathcal{I}$.

The framework evolves the FSM exclusively through a strictly defined set of Atomic Operations. Let $\mathcal{M}_t$ represent the FSM structure at iteration t. The evolution process is formulated as $\mathcal{M}_{t+1} = \mathcal{M}_t \oplus op$, where the operation *op* is cho-

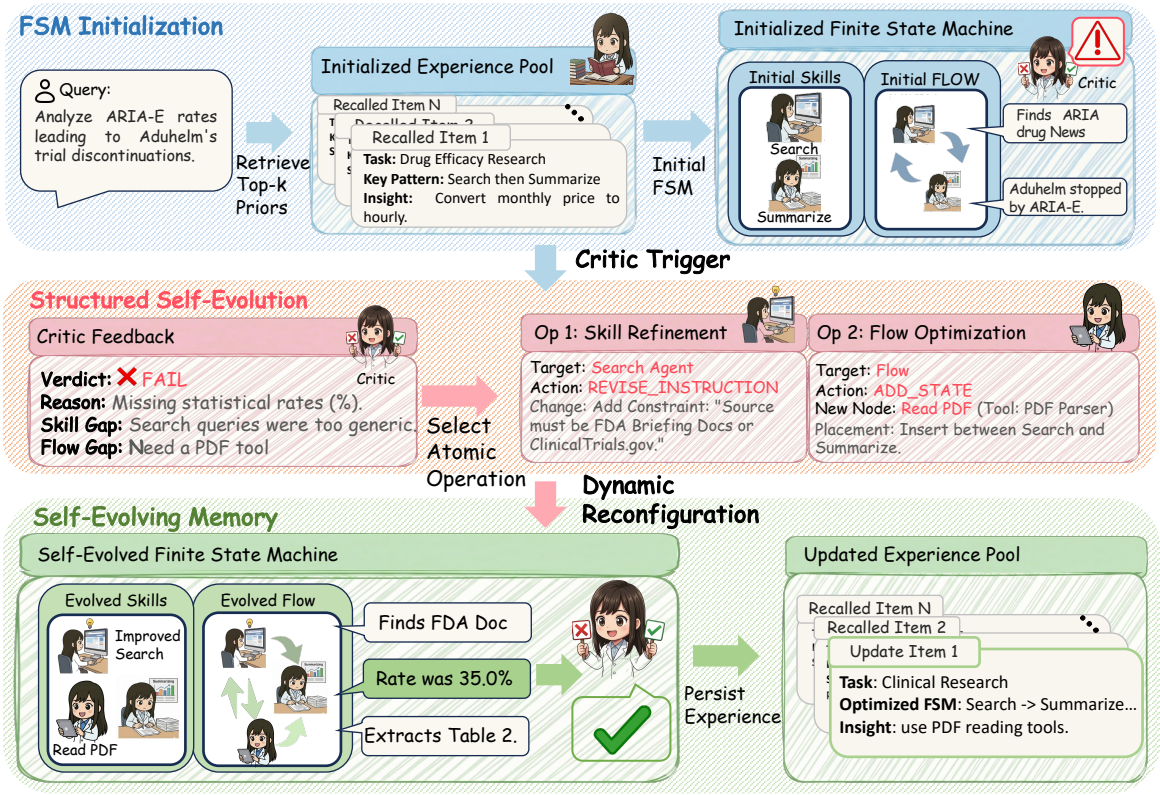

Figure 2: Overview of the EvoFSM framework. Our approach consists of three core components: (1) FSM Initialization, which formalizes the research process as a dynamic finite state machine initialized from prior experiences; (2) Structured Self-Evolution, which employs atomic operations to precisely optimize both the system's skill operators ($\mathcal{O}_{skill}$) and flow operators ($\mathcal{O}_{flow}$) based on critic feedback; and (3) Self-Evolving Memory Mechanism, which distills successful and failure trajectories into an experience pool to facilitate continuous learning and warm-starting for future tasks.

sen from one of two operator sets, corresponding to Flow- and Skill-level evolution.

Flow Operators $\mathcal{O}_{flow}$ These operations reconfigure the collaboration topology by editing the state set and/or the transition logic, without changing any node-specific instructions.

- `ADD_STATE` inserts an intermediate state (e.g., a verification state) to bridge a workflow gap.
- `DELETE_STATE` removes redundant or low-utility states to streamline execution.
- `MODIFY_TRANSITION` adjusts transition conditions (e.g., revisiting Search when the retrieved evidence is insufficient).

Skill Operators $\mathcal{O}_{skill}$ These operations refine node-specific instructions while keeping the global topology unchanged.

- `REVISE_INSTRUCTION` updates the guidelines or constraints for a single state (e.g., instructing the browsing state to prioritize PDFs over news sources to improve evidence quality).

By enforcing these atomic operations, EvoFSM ensures that any modification remains local, interpretable, and reversible.

3.3 Self-Evolving Memory

While the structured evolution framework empowers agents to dynamically adapt to specific queries, treating each task as an isolated optimization problem prevents the system from accumulating valuable experience. To address this limitation and enable continuous improvement, EvoFSM integrates a self-evolving memory mechanism that functions as a repository of successful strategies.

The system maintains an experience pool $\mathcal{E}$ to persist high-quality self-evolving patterns. Upon the completion of a task, a specific reflection agent analyzes the execution trajectory to determine if the task was resolved successfully. In such cases, the final optimized FSM configuration and the sequence

of atomic operations are distilled into a strategy record r. This record encapsulates the query embedding $q_{\text{embedding}}$, the optimized model structure $\mathcal{M}_{optimized}$, and the rationale for the changes.

When the system receives a new query q_{new}, the module retrieves the top-k historical records from the experience pool $\mathcal{E}$, where $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$. In this context, successful strategies ($\mathcal{E}^+$) serve as initialization priors, allowing the agent to inherit effective execution trajectories (e.g., initializing with a search-verify loop for medical diagnoses as shown in Figure 2). Conversely, retrieved failure patterns ($\mathcal{E}^-$) serve as negative constraints, actively warning the transition logic against specific transition paths or tool usages that previously led to dead ends in similar contexts. This mechanism enables the system to exhibit progressively stronger performance by building upon past experiences.

4 Experiments

4.1 Experimental Settings

Datasets and Metrics. We evaluate our EvoFSM on five multi-hop question answering (QA) benchmarks that require aggregating evidence across multiple documents. Specifically, we consider the following benchmarks: HotpotQA (Yang et al., 2018), the first large-scale dataset requiring reasoning over multiple Wikipedia paragraphs; 2WikiMultihopQA (2WIKI) (Ho et al., 2020), which provides multi-hop QA with explicit reasoning paths; MuSiQue (Trivedi et al., 2022), which synthesizes 2–4 hop questions by composing evidence from five single-hop datasets; Bamboogle (Press et al., 2023), a collection of compositional questions that search engines often answer incorrectly; and xbench-DeepSearch (Deepsearch) (Chen et al., 2025), part of the xbench AGI-Aligned series, which provides a Chinese-context question pool for multi-hop QA and deep search.

Baselines. We compare EvoFSM against three baseline methods. Standard RAG performs a single web-search round and appends the top-10 retrieved documents to the prompt as additional context. Agentic RAG[1] introduces an iterative retrieve–reason loop, allowing the model to autonomously issue additional retrieval requests when the current evidence is insufficient. Search-o1 (Li et al., 2025a) further strengthens the agentic RAG pipeline by adding a *Reason-in-Documents* stage, which distills retrieved passages into concise, relevant evidence while preserving the intermediate reasoning process. In addition, to demonstrate the generality of our method, we conduct experiments with a diverse set of LLMs, including proprietary models (GPT-4o(OpenAI, 2024) and Claude-4(Anthropic, 2025)) and open-weight models (Llama-3-70B(Grattafiori et al., 2024), DeepSeek-V3(DeepSeek-AI et al., 2024), and Qwen3-32B(Yang et al., 2025)).

Implementation Details. Our system builds on AutoGen (Wu et al., 2023) for multi-agent orchestration, enabling reliable inter-agent communication and stateful execution. We further provide a tool library, including the Serper[2] API for retrieval and the Jina Reader[3] API for web-page content extraction. To ensure fair comparison of structural efficiency and avoid overfitting, we limit the optimization loop to 3 iterations. We also cap the number of states at 10 to prevent unbounded growth, while still permitting dynamic refinements to state definitions and transitions when necessary.

4.2 Main Results

Table 1 presents the performance comparison of different frameworks across five diverse benchmarks. From these results, we draw three observations.

First, iterative retrieval-and-reasoning frameworks consistently outperform Standard RAG across all benchmarks. For instance, under GPT-4o, Agentic RAG delivers a 15.0% absolute gain on DeepSearch over the single-shot retrieval baseline. This gap reflects a fundamental limitation of one-pass retrieval: the system must surface all necessary evidence in a single query, with limited opportunity to recover from missing or off-target context. In contrast, iterative methods can progressively refine queries and validate intermediate hypotheses against newly retrieved evidence. EvoFSM retains this iterative capability, but makes it more reliable by explicitly structuring the loop into well-defined phases and transitions, reducing wasted iterations and stabilizing evidence accumulation.

Second, the relative comparison among frameworks is largely stable across the evaluated LLMs. In other words, the improvements brought by EvoFSM are not tied to a single backbone or a narrow operating regime; instead, the same design advantages tend to persist when the underlying model

[1] We follow the implementation details in (Li et al., 2025a).

[2] https://serper.dev

[3] https://jina.ai

Model & Framework		Multi-hop QA Benchmarks				
		HotpotQA	**2WIKI**	**MuSiQue**	**Bamboogle**	**Deepsearch**
GPT-4o	Standard RAG	68.2	54.2	42.8	80.0	18.0
	Agentic RAG	76.6	71.4	41.6	84.0	33.0
	Search-o1	77.8	85.6	48.2	**87.2**	35.0
	EvoFSM	**80.2**	**88.8**	**50.4**	82.4	**45.0**
Claude-4	Standard RAG	65.4	30.8	33.4	75.2	17.0
	Agentic RAG	80.6	90.0	51.0	88.8	53.0
	Search-o1	81.2	91.6	51.0	87.2	47.0
	EvoFSM	**82.2**	**91.8**	**57.6**	**91.2**	**58.0**
Llama3-70B	Standard RAG	72.2	57.0	40.4	**84.0**	16.0
	Agentic RAG	61.6	74.4	38.0	67.2	23.0
	Search-o1	76.2	**85.2**	43.0	83.2	24.0
	EvoFSM	**76.6**	75.6	**46.4**	80.4	**28.0**
DeepSeek-v3	Standard RAG	72.6	51.8	41.0	82.4	30.0
	Agentic RAG	65.2	71.2	42.0	66.4	31.0
	Search-o1	74.6	85.0	46.6	84.0	43.0
	EvoFSM	**80.4**	**88.8**	**54.2**	**89.6**	**51.0**
Qwen3-32B	Standard RAG	63.2	41.6	33.8	74.4	20.0
	Agentic RAG	64.6	69.6	34.8	68.8	19.0
	Search-o1	67.0	76.2	37.2	**84.0**	27.0
	EvoFSM	**77.8**	**83.6**	**43.8**	81.6	**32.0**

Table 1: Accuracy (%) of different retrieval frameworks for each backbone model on five challenging multi-hop QA benchmarks. Best results are in **bold** and second-best results are underlined.

changes. This robustness suggests that the performance gains primarily come from the workflow and optimization mechanisms, rather than from model-specific quirks.

Finally, EvoFSM delivers the strongest overall results and a consistent margin over both Agentic RAG and Search-o1. In particular, EvoFSM consistently outperforms Search-o1 across all five LLMs on DeepSearch. For example, it improves by 11.0% with Claude-4 and 10.0% with GPT-4o, and still yields a 4.0% gain with Llama-3-70B. We attribute this additional margin to structured evolution, which makes the iterative retrieve–reason process experience-guided and adaptively evolvable. EvoFSM initializes each new query with relevant historical strategies and then refines its workflow to the specific instance, enabling more robust evidence acquisition and stronger multi-hop reasoning under the same tool budget.

4.3 Ablation Study

To disentangle the contribution of each component in EvoFSM, we perform ablation studies with DeepSeek-v3 as the backbone on five benchmarks. Specifically, we isolate the effects of (i) the structured self-evolution mechanism with memory and (ii) the explicit FSM topology. The results are summarized in Table 2.

Impact of Structured Self-Evolution. We first evaluate the role of the evolution mechanism by removing the optimization loop and memory retrieval, denoted as *w/o Structured Evolution (Static FSM)*. This variant uses a fixed FSM initialized with a manually designed workflow. Specifically, it contains three predefined states—*Search*, *Browsing*, and *Analysis*—together with hand-written instructions and deterministic transition rules. The transition logic routes the state from *Search* to *Browsing* after retrieving candidate sources, proceeds to *Analysis* once sufficient evidence is collected, and otherwise falls back to *Search* to issue refined queries. Since both the state set and the transition conditions are fixed, the workflow cannot be reconfigured during execution. Both the state set and the transition logic remain unchanged throughout execution. The results reveal a substantial performance drop across all datasets, most notably on the DeepSearch benchmark, where accuracy plum-

Method	Multi-hop QA Benchmarks				
	HotpotQA	2WIKI	MuSiQue	Bamboogle	DeepSearch
EvoFSM (Full Model)	**80.4**	**88.8**	**54.2**	**89.6**	**51.0**
w/o Structured Evolution (Static FSM)	77.8 (-2.6)	82.4 (-6.4)	50.8 (-3.4)	85.6 (-4.0)	36.0 (-15.0)
w/o FSM Topology (Unstructured Evolution)	77.0 (-3.4)	83.8 (-5.0)	50.2 (-4.0)	86.4 (-3.2)	42.0 (-9.0)
w/o All (Standard ReAct)	76.6 (-3.8)	84.0 (-4.8)	48.4 (-5.8)	82.4 (-7.2)	34.0 (-17.0)

Table 2: Ablation Study of EvoFSM framework. We systematically remove the structured self-evolution mechanism and the FSM topology to evaluate their contributions.

mets by 15.0% (from 51.0% to 36.0%). This sharp decline underscores the critical limitation of static workflows: without the ability to dynamically apply atomic operations ($\mathcal{O}_{flow}$ and $\mathcal{O}_{skill}$) or recall successful priors from memory, the agent cannot adapt to unforeseen bottlenecks in complex queries, leading to repeated strategic failures such as insufficient verification or data extraction errors.

Necessity of FSM Topology. Next, we investigate the importance of the graph-based control structure by removing the explicit FSM constraints while retaining the self-evolution capability, denoted as *w/o FSM Topology (Unstructured Evolution)*. In this setting, the system allows for free-form prompt rewriting but lacks the deterministic guidance of a state transition graph. We observe a performance degradation of 9.0% on DeepSearch. While prompt evolution provides some adaptability, the absence of a rigid structural backbone leads to unstable behavior. Without the explicit *Flow* logic governed by the FSM, agents are prone to redundant looping or losing track of the research objective, confirming that structural boundaries are essential for keeping self-evolution targeted and stable.

Synergy of Structure and Evolution. Finally, the *w/o All (Standard ReAct)* baseline, which strips away both the FSM structure and the self-evolving mechanism, yields the lowest performance across the board, with a 17.0% drop on DeepSearch. Comparing this baseline with the single-component ablations reveals a compounding effect: combining the FSM structure with self-evolution yields larger gains than either component alone. The FSM provides the necessary stability for long-horizon reasoning, while structured self-evolution ensures that this stability does not degenerate into rigidity. Together, they enable EvoFSM to achieve both high precision and robust adaptability.

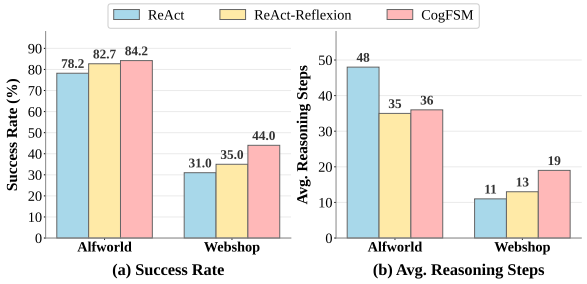

Figure 3: Transferability study on the ALFWorld and WebShop benchmarks. We compare the success rate (a) and average reasoning steps (b) of each method.

4.4 Further Analysis

Cross-Domain Generalization Capabilities. To assess the versatility of EvoFSM beyond deep research, we evaluate it on two interactive decision-making benchmarks, ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022). ALFWorld is a text-based household environment containing 134 tasks spanning six types, such as object manipulation and navigation. The agent interacts through natural-language actions (e.g., moving to locations, picking up objects) and receives textual observations as feedback. WebShop simulates an online shopping platform with 1.18M real-world products and 12k human-written instructions, where the agent must satisfy user requirements by issuing search queries and selecting products through multi-step interactions.

As shown in Figure 3(a), EvoFSM consistently outperforms both ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2023), with particularly strong gains on WebShop. These results suggest that the structured FSM execution helps mitigate degenerate long-horizon behaviors such as redundant looping, and that the benefits of structured self-evolution transfer from deep research to general interactive tasks. In terms of efficiency, Figure 3(b) shows that EvoFSM uses slightly more

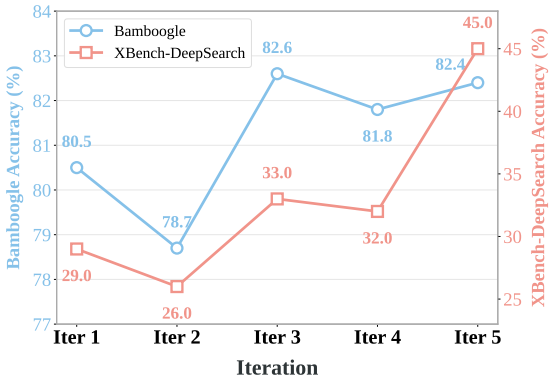

Figure 4: Effect of number of iterations on accuracy in the Bamboogle and DeepSearch benchmarks.

reasoning steps than ReAct. This is expected, since the FSM explicitly allocates steps for validation and disciplined state transitions to improve robustness, trading a modest increase in steps for higher task success rates.

Impact of Optimization Iterations. We investigate the impact of the optimization depth on system performance by varying the number of evolution iterations, as illustrated in Figure 4. Each iteration corresponds to a complete cycle of trajectory analysis, experience extraction, and FSM refinement. The results, based on the GPT-4o backbone, demonstrate a clear positive correlation between iterative optimization and task accuracy. Notably, on the highly complex DeepSearch benchmark, we observe a substantial 16% performance gain as iterations increase, suggesting that intricate research tasks benefit significantly from continuous structural refinement. In contrast, for the relatively straightforward Bamboogle benchmark, performance rapidly plateaus after just three iterations. This saturation indicates that while our self-evolution mechanism is highly effective, the optimal optimization depth is task-dependent. Practically, this suggests that EvoFSM can adaptively balance performance gains against computational overhead, investing more self-evolving steps only where the problem complexity demands it.

Cases Analysis and Discussion. The presented cases qualitatively validate the core hypothesis of EvoFSM: that robust self-evolution requires the distinct treatment of workflow topology and individual expertise. As shown in Figure 5, structural bottlenecks like infinite search loops typically withstand simple prompt adjustments and instead necessitate a topological intervention, such as `ADD_STATE`, to fundamentally alter the reasoning path. Conversely, Figure 6 demonstrates that when the workflow is sound but the output lacks precision, targeted `REVISE_INSTRUCTION` operations effectively sharpen node-specific skills without disrupting the global process. Most importantly, Figure 7 highlights the necessity of synergistic evolution for complex tasks. By simultaneously deploying a `Verifier` node to enforce source quality and refining the `Search Agent` to target legal terminologies, EvoFSM successfully transforms a generic retrieval pipeline into a rigorous legal analysis workflow. Collectively, these examples illustrate that our structured, decoupled framework enables agents to precisely diagnose and repair specific failure modes. This capability stands in sharp contrast to unstructured rewriting approaches, which often struggle to isolate such orthogonal issues, leading to unstable or unpredictable evolution.

5 Conclusion

In this paper, we presented **EvoFSM**, a structured self-evolving framework designed to overcome the limitations of both static workflows and unconstrained self-evolving agents in deep research tasks. By formalizing the research process as a dynamic Finite State Machine, EvoFSM decouples the optimization space into macroscopic *Flow* and microscopic *Skill*. Unlike existing methods that rely on unstructured global rewriting, our approach empowers system to adapt through a set of precise atomic operations, ensuring that the self-evolving process remains targeted and controllable. Furthermore, the integration of a self-evolving memory mechanism allows the system to distill and transfer successful exploration strategies across tasks, facilitating continuous learning. Extensive experiments show that our EvoFSM significantly outperforms strong baselines on five multi-hop QA benchmarks while exhibiting superior generalization in interactive decision-making environments. We believe this paradigm shift—from black-box self-modification to structured, controllable evolution—provides a promising path toward reliable autonomous agents that can tackle complex, open-ended real-world problems.

Limitations

Despite the promising performance of EvoFSM in automating deep research, three key limitations remain to be addressed in future work. First, our

framework currently relies entirely on off-the-shelf proprietary LLMs via prompt engineering and in-context learning. We do not perform any fine-tuning or specialized training on the underlying models. While this ensures broad accessibility, it inherently limits the system's efficiency and responsiveness, as general-purpose models may struggle to internalize complex FSM logic without explicit weight updates. Future iterations could benefit significantly from distilling these self-evolving capabilities into smaller, specialized agents. Second, the reliability of the entire self-evolving process hinges on the Critic Mechanism. Since the system operates without external ground truth during deployment, it depends on the Critic (itself an LLM) to accurately diagnose failures. If the Critic hallucinates a successful verification or fails to detect subtle logical errors, the system may learn incorrect patterns or fail to evolve effectively. Developing more robust, verification-guided critics remains an open challenge. Finally, as the agent continuously solves new tasks, the self-evolving memory faces a scalability issue. The experience pool currently grows indefinitely without a mechanism for consolidation or forgetting. Over time, this accumulation may lead to retrieval latency and the retrieval of redundant or outdated strategies. Implementing a long-term memory management system that can abstract, merge, or prune experiences is essential for sustaining performance in life-long learning scenarios.

References

Anthropic. 2025. Claude 4. https://www.anthropic.com/.

Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, Chen Sun, Han Hou, Hui Yang, James Pan, Jianan Lou, Jiayi Mao, Jizheng Liu, Jinpeng Li, Kangyi Liu, and 14 others. 2025. xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations. *Preprint*, arXiv:2506.13651.

DeepSeek-AI, Aixin Liu, Bei Feng, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Lucas Deppen, Rylan Schaeffer, Sashank Santhanam, and Sanmi Koyejo. 2024. Live-swe-agent: Can software engineering agents self-evolve on the fly? *arXiv preprint arXiv:2410.15283*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516.

Jiale Li, Shengyu Zhang, Gang Chen, and Dongxiao Zhang. 2024. Morphagent: Empowering agents through self-evolving profiles and decentralized collaboration. *arXiv preprint arXiv:2407.03158*.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025a. Search-o1: Agentic search-enhanced large reasoning models. *CoRR*, abs/2501.05366.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. 2025b. Webthinker: Empowering large reasoning models with deep research capability. *CoRR*, abs/2505.12345.

Jiaye Lin, Yifu Guo, Yuzhen Han, Sen Hu, Ziyi Ni, Licheng Wang, Mingguang Chen, Hongzhang Liu, Ronghao Chen, Yangfan He, Daxin Jiang, Binxing Jiao, Chen Hu, and Huacan Wang. 2025. Se-agent: Self-evolution trajectory optimization in multi-step reasoning with llm-based agents. *arXiv preprint arXiv:2508.02085*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

Ziyi Ni, Hao Wang, and Huacan Wang. 2025. Shield-learner: A new paradigm for jailbreak attack defense in llms. *arXiv preprint arXiv:2502.13162*.

OpenAI. 2024. Gpt-4o system card. https://openai.com/index/gpt-4o-system-card/.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Jürgen Schmidhuber, Rupesh Kumar Srivastava, and Bas R. Steunebrink. 2024. Huxley-gödel machine: Human-level coding agent development by an approximation of the optimal self-improving machine. *arXiv preprint arXiv:2406.00985*.

Shuai Shao, Qihan Ren, Chen Qian, Boyi Wei, Dadi Guo, Jingyi Yang, Xinhao Song, Linfeng Zhang, Weinan Zhang, Dongrui Liu, and Jing Shao. 2024. Your agent may misevolve: Emergent risks in self-evolving llm agents. *arXiv preprint arXiv:2406.01234*.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

Yue Sun, Tianlong Wang, Qinyuan Cheng, and Xiangyang Liu. 2024. Research: Re-ranking enhanced search for large language models. *CoRR*, abs/2406.00000. (Placeholder for ReSearch if specific ID unavailable, please verify).

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Huacan Wang, Ziyi Ni, Shuo Zhang, Shuo Lu, Sen Hu, Ziyang He, Chen Hu, Jiaye Lin, Yifu Guo, Ronghao Chen, Xin Li, Daxin Jiang, Yuntao Du, and Pin Lyu. 2025. Repomaster: Autonomous exploration and understanding of github repositories for complex task solving. *arXiv preprint arXiv:2505.21577*.

Yijia Wang, Yuxuan Liu, Zuyun Wang, Zihan Liu, Yu Zhang, Fang Liu, and Dongsheng Li. 2024. Stella: Self-evolving llm agent for biomedical research. *arXiv preprint arXiv:2402.06642*.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. 2025. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *CoRR*, abs/2505.16421.

Jiacheng Wu, Chuhui Wu, Zongkai Lin, Zhijie Wang, and Wei Wang. 2025. Comas: Co-evolving multi-agent systems via interaction rewards. In *International Conference on Learning Representations*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *Preprint*, arXiv:2308.08155.

Yenzhe Wu, Kaiyan Chang, and Debin Lin. 2024. Stateflow: Enhancing llm dialogues with finite state machines. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Can Xu, Qingxiu Dong, Ziwei Ji, Daya Guo, and Yeyun Gong. 2024. Zerosearch: Zero-shot search query generation for llms. *CoRR*, abs/2401.00000. (Placeholder for ZeroSearch if specific ID unavailable, please verify).

An Yang, Anfeng Li, Baosong Yang, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *Preprint*, arXiv:1809.09600.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Shu Zhao, Tan Yu, Anbang Xu, Japinder Singh, Aaditya Shukla, and Rama Akkiraju. 2025. Parallelsearch: Train your llms to decompose query and search subqueries in parallel with reinforcement learning. *arXiv preprint arXiv:2508.09303*.

A Illustrative Examples

Case 1: Flow Evolution via `ADD_STATE`

User Query: "What are the specific environmental impacts of the continuous construction of the Three Gorges Dam recorded in 2023 reports?"

Initial FSM Execution (Failed):

- `[Search Agent]` → Queries "Three Gorges Dam environmental impact 2023".
- `[Browse Agent]` → Reads generic Wikipedia pages.
- `[Search Agent]` → Queries again (Loop detected).
- *Outcome:* The system enters a "Search-Browse" infinite loop, unable to find specific 2023 reports, leading to a timeout.

Evolution (Flow Dimension):

- *Diagnosis:* The standard loop lacks a mechanism to verify the *specificity* of the date in retrieved documents.
- *Atomic Operation:* `ADD_STATE(Verifier)`.
- *New Topology:* `Search` → `Browse` → **`Verifier`** → `Synthesize`.

Evolved FSM Execution (Success):

- ... (Search & Browse steps) ...
- **`[Verifier Agent]`** → "The retrieved documents are from 2020. We need specific 2023 data. Refining query to 'Three Gorges Dam annual report 2023 pdf'."
- `[Search Agent]` → Executes refined query.
- *Outcome:* Successfully locates the 2023 annual environmental report and extracts specific data points.

Figure 5: An example of *Flow Evolution*. EvoFSM identifies a reasoning deadlock and structurally intervenes by injecting a new `Verifier` state, enabling the system to break out of a low-quality search loop.

Case 2: Instruction Evolution via `REVISE_INSTRUCTION`

User Query: "Compare the battery energy density of the latest EV models from Tesla, BYD, and Nio launched in Q4 2023."

Initial FSM Execution (Suboptimal):

- `[Search Agent]` → Finds comparison articles.
- `[Browse Agent]` → Summarizes content: "Tesla has high density, BYD uses Blade battery..."
- *Outcome:* The answer is vague and qualitative, missing the specific "Wh/kg" numerical values required for a rigorous comparison.

Evolution (Instruction Dimension):

- *Diagnosis:* The `Browse Agent` is summarizing too aggressively, losing critical quantitative details.
- *Atomic Operation:* `REVISE_INSTRUCTION(Browse Agent)`.
- *Modification:* Append constraint: "Do not summarize numerical data. Extract exact values with units (e.g., Wh/kg) verbatim from the text."

Evolved FSM Execution (Success):

- `[Browse Agent] (Evolved)` → Extracts: "Tesla Model 3 Highland: 260 Wh/kg; BYD Seal: 150 Wh/kg..."
- *Outcome:* A precise, data-driven comparison table is generated.

Figure 6: An example of *Instruction Evolution*. EvoFSM detects information loss in the `Browse` node and fine-tunes its system prompt without altering the overall workflow, achieving higher precision.

Case 3: Synergistic Evolution (Flow + Skill)

User Query: "Analyze how the EU AI Act (late 2023 draft) regulates open-source foundation models compared to proprietary ones, citing specific Articles."

Phase 1: Initial Failure (Static FSM)

- **Execution:** `Search Agent` queries "EU AI Act open source". `Browse Agent` summarizes TechCrunch articles.
- **Failure Mode:** The system produces a vague summary ("It has exemptions") without citing specific articles, as it relies on secondary news sources.
- **Diagnosis:** 1. *Flow Deficit*: The workflow lacks a verification step to distinguish official legal texts from news. 2. *Skill Deficit*: The `Search Agent` uses generic keywords instead of legal terminology.

Phase 2: Dual-Dimension Evolution

- **Flow Action ($\mathcal{O}_{flow}$):** `ADD_STATE(Legal_Verifier)`. *Purpose:* To explicitly filter retrieved documents for official legislative formatting (e.g., "Recital", "Article").
- **Skill Action ($\mathcal{O}_{skill}$):** `REVISE_INSTRUCTION(Search Agent)`. *Refinement:* "Constraint: Do not search for news. Construct queries targeting specific legal sections (e.g., 'EU AI Act Article 53 exemption')."

Phase 3: Evolved Success

- **New Workflow:** `Search (Legal-Focused)` $\rightarrow$ `Browse` $\rightarrow$ **`Legal_Verifier`** $\rightarrow$ `Synthesize`.
- **Execution:** 1. Search Agent executes a precise query. 2. Browse Agent reads the official PDF. 3. `Legal_Verifier` confirms: "Found Recital 60i and Article 53(2)."
- **Outcome:** Synthesizes a legally accurate comparison citing the exact clauses.

Figure 7: An example of *Synergistic Evolution*. EvoFSM simultaneously reconfigures the collaboration topology (adding a Verifier) and sharpens the individual search expertise. This dual optimization enables the agent to pivot from superficial news summarization to rigorous legal analysis.