SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

An Agentic Ai Framework For Autonomous Reasoning, Tool Selection, And Task Execution Using Large Language Models

Azhar Ushmani

VP, Security Engineering, PIMCO Austin, Texas – USA. E-mail: Azhar.ushmani2026@gmail.com

Abstract
The increasing complexity of AI-assisted workflows requires autonomous systems capable of reasoning, selecting tools, and executing multi-step tasks beyond conventional text generation. Existing large language model (LLM) assistants often depend on static prompting and exhibit limitations in planning, tool selection, execution monitoring, and recovery from intermediate failures. This study proposes an Agentic AI Framework (AAIF) that integrates task interpretation, goal decomposition, autonomous reasoning, dynamic tool selection, task execution, execution-state monitoring, error recovery, and final-response generation. The LLM acts as the central reasoning and orchestration engine, while external software capabilities are organized through a structured tool registry. AAIF is computationally evaluated on representative tasks with simple, moderate, and complex workflows and compared with Direct LLM, Fixed-Tool LLM, and Planning-Only Agent approaches. Performance is assessed using task success rate, tool-selection accuracy, planning success rate, execution efficiency, average execution time, and recovery success rate. The proposed framework achieves 94.2% task success, 96.1% tool-selection accuracy, 93.4% planning success, 91.8% execution efficiency, 2.84 s average execution time, and 92.6% recovery success, demonstrating improved reliability for autonomous LLM-driven task execution.

Keywords: Agentic AI, Large Language Models, Autonomous Agents, Tool Selection, Task Planning, Reasoning, Task Execution, AI Orchestration.

1. Introduction

Large Language Models (LLMs) have evolved from conversational text-generation systems into increasingly capable platforms for reasoning, planning, and autonomous problem solving. Traditional LLM interaction usually follows a prompt-response pattern, whereas complex real-world tasks often require multiple coordinated actions. Agentic AI extends LLM capability by combining language understanding with reasoning, planning, memory, external tools, and execution mechanisms. This enables an AI system to move beyond generating answers and instead perform structured workflows toward a defined objective.

A major requirement of agentic AI is autonomous decomposition of complex goals into smaller executable subtasks. Each subtask may require different tools, including search services, calculators, databases, code executors, document processors, and information-retrieval systems. Effective task completion therefore depends on selecting the appropriate tool, generating valid inputs, interpreting outputs, and determining subsequent actions. Dynamic tool selection is particularly important when task requirements change during execution or depend on intermediate results.

Existing prompt-only and rigid workflow approaches face several limitations, including weak planning, incorrect tool invocation, repeated actions, reasoning errors, and execution failures. These problems can propagate through multi-step tasks and reduce reliability. Static workflows also lack flexibility when an initial plan becomes invalid. A more robust agent requires continuous interaction between reasoning, execution, observation, verification, and replanning. Such a closed-loop process allows the system to detect failures, revise its strategy, select alternative tools, and continue execution without restarting the complete task.

This study proposes an Agentic AI Framework (AAIF) for autonomous reasoning, tool selection, and task execution using LLMs. The framework integrates task interpretation, goal decomposition, multi-step planning, context-aware tool selection, execution monitoring, and automatic error recovery within a unified architecture. Its major contributions include autonomous task processing, dynamic tool orchestration, closed-loop execution control, failure-driven replanning, and quantitative evaluation across simple, moderate, and complex tasks. AAIF is compared with Direct LLM, Fixed-Tool LLM, and Planning-Only Agent approaches to evaluate

improvements in task success, tool-selection accuracy, planning reliability, execution efficiency, and recovery performance.

2. Literature Review

Large Language Models have advanced substantially in instruction following, contextual understanding, and general-purpose text generation [1], [3]. Recent systems can perform reasoning-oriented tasks when guided through structured prompting, including chain-of-thought and stepwise decomposition [10]. These techniques improve transparency of intermediate reasoning and support more complex problem solving [11]. However, prompt-based reasoning alone remains limited when tasks require interaction with external environments, dynamic information retrieval, or execution of actions beyond language generation [2].

Agentic AI extends LLM capabilities by combining reasoning with planning, acting, memory, and autonomous decision making [7]. ReAct-style approaches integrate reasoning traces with action execution, allowing models to alternate between internal decision processes and external tool use. LLM-based planning has also been explored for decomposing complex goals into smaller executable steps [12]. External-tool-augmented systems further improve functionality through calculators, search engines, databases, code execution, function calling, and API orchestration [8], [14]. Retrieval-augmented generation complements these approaches by supplying external knowledge that can reduce unsupported responses and improve contextual relevance [4].

Current research also focuses on multi-step task execution, memory-based agents, reflection, and self-correction [9]. Memory mechanisms help agents preserve previous observations and maintain continuity across longer workflows, while reflection-based approaches enable agents to reconsider unsuccessful decisions [6]. Tool-selection strategies attempt to identify suitable tools from available registries, and execution-verification mechanisms assess whether returned outputs satisfy the current subtask [5]. Error-recovery methods may involve retries, alternative tool selection, plan revision, or regeneration of intermediate steps [13]. Despite these advances, many existing architectures remain sensitive to incorrect actions, repeated tool calls, long reasoning chains, and accumulated execution errors [15].

A key research gap is that reasoning, planning, tool use, verification, and self-correction are frequently treated as separate or loosely connected capabilities. Few frameworks provide a unified closed-loop process that can dynamically decompose goals, select tools, execute actions, observe outcomes, detect failures, and autonomously replan within the same workflow. This limitation reduces reliability in complex multi-tool tasks. The proposed Agentic AI Framework addresses this gap by integrating autonomous reasoning, task planning, dynamic tool orchestration, execution-state monitoring, and recovery into a single coordinated architecture.

3. Methodology

3.1 Overall Agentic AI Architecture

The proposed Agentic AI Framework (AAIF) uses the LLM as the central reasoning and orchestration engine. As shown in Figure 1, the workflow follows User Request → Task Interpreter → Goal Decomposition → Reasoning and Planning Engine → Tool Selection Module → Tool Execution → Observation and Verification → Replanning/Error Recovery → Final Response. The framework also uses short-term memory, long-term memory, and a knowledge base to support contextual reasoning and execution history.

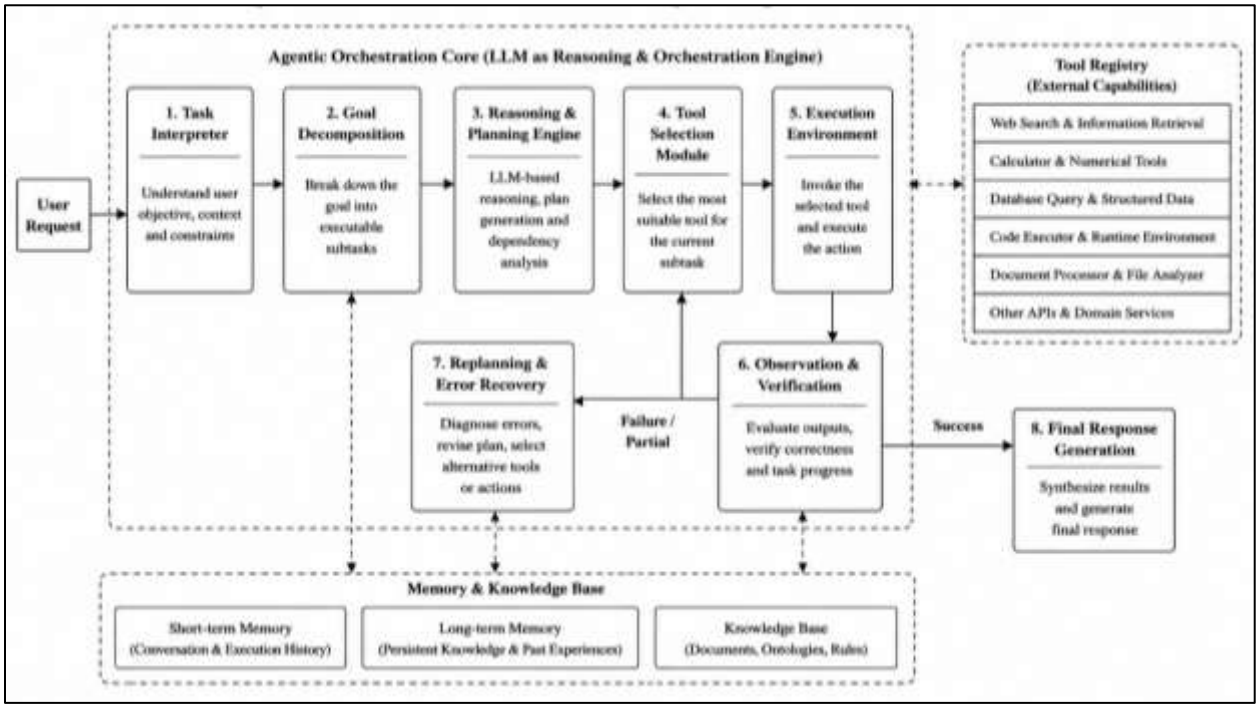

Figure 1. Overall Architecture of the Proposed Agentic AI Framework

The input task is represented as

$$Q = \{x, C, G\} \quad , \_\_\_\_\_ \quad (1)$$

where $x$is the user instruction, $C$is contextual information, and $G$is the required goal. The planning module converts the overall goal into executable subtasks:

$$P = \{s_1, s_2, \dots, s_n\} \quad , \_\_\_\_\_ \quad (2)$$

where $s_i$represents the i^{th}executable subtask.

The major functions, inputs, and outputs of the AAIF modules are summarized in **Table 1**. The task interpreter identifies the goal, the planning engine organizes subtasks, the tool selector chooses suitable tools, and the executor performs the actions. The verifier checks outputs, while the recovery module revises failed actions. Memory stores execution history, and the response generator produces the final verified response.

Table 1. Core Components of the Proposed AAIF

Component	Function	Input	Output
Task Interpreter	Understands user goal	User request	Structured task
Goal Decomposition	Splits task into subtasks	Structured task	Subtask sequence
Planning Engine	Generates action plan	Subtasks, context	Execution plan
Tool Selector	Chooses suitable tool	Current subtask	Selected tool
Executor	Performs action	Tool command	Execution result
Verifier	Checks task outcome	Tool output	Success/failure
Recovery Module	Revises failed actions	Failure state	Updated plan
Memory Module	Stores context and history	Execution history	Updated memory
Response Generator	Produces final answer	Verified results	Final response

3.2 Autonomous Reasoning and Task Planning

The autonomous reasoning and planning module enables the LLM to interpret each task, identify constraints, retrieve relevant context, and generate an executable sequence of subtasks. As shown in Figure 2, the process begins with task-goal interpretation and constraint analysis, followed by context retrieval, candidate-step generation, dependency checking, and execution-plan formation. Each planned step is then executed and verified before the agent proceeds to the next action.

For each subtask s_i, the reasoning state is defined as

$$r_i = f_\theta(s_i, C, H_i), \quad (3)$$

where f_θ represents the LLM reasoning function, C is the task context, H_i denotes execution history, and r_i is the reasoning state. The planning process supports sequential subtasks, conditional branches, dependencies between actions, intermediate-result checking, and dynamic plan modification. If the verification stage indicates incomplete or invalid progress, the agent revises the plan and generates alternative actions using updated context and execution history.

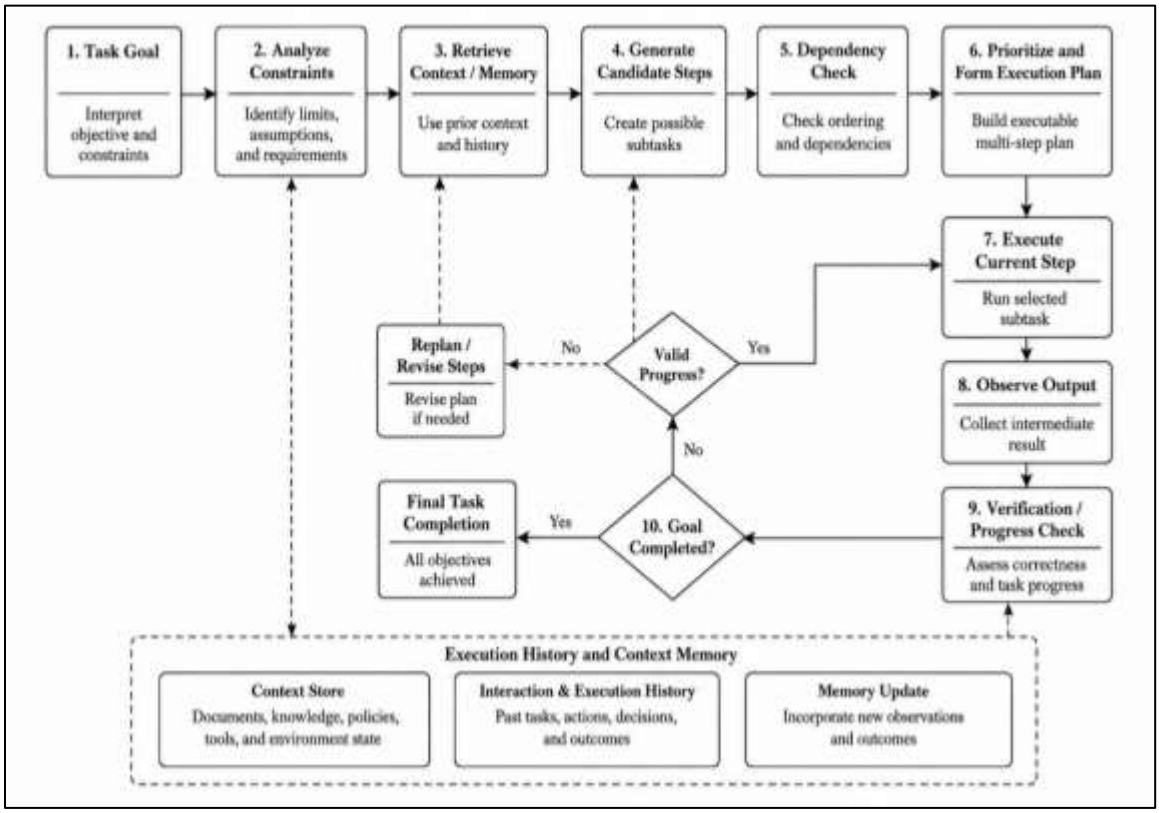

Figure 2. Autonomous Reasoning and Multi-Step Task Planning Workflow

3.3 Dynamic Tool Selection and Execution

The dynamic tool-selection module enables AAIF to choose the most appropriate external capability for each subtask. As summarized in Table 2, the tool registry includes web search, calculator, database query, code execution, document processing, and API-based services. Each tool receives a suitability score according to its relevance, compatibility, historical reliability, and execution efficiency:

$$S(t_j \mid s_i) = w_1 R_j + w_2 C_j + w_3 A_j + w_4 E_j, \quad (4)$$

where R_j is tool relevance, C_j is compatibility, A_j is historical success or reliability, and E_j is execution efficiency. The weights satisfy $w_1 + w_2 + w_3 + w_4 = 1$.

The tool with the highest suitability score is selected as

$$t_i^* = \arg\max_{t_j \in T} S(t_j \mid s_i), \quad (5)$$

where T represents the available tool set. Before execution, the framework verifies tool availability and input validity to reduce unsupported calls and execution errors.

Table 2. Tool Registry and Agent Capabilities

Tool	Primary Capability	Input	Output
Web Search	Retrieves external information	Search query	Relevant information

Calculator	Performs numerical operations	Mathematical expression	Numerical result
Database Tool	Retrieves structured data	Database query	Data records
Code Executor	Runs computational tasks	Code/instruction	Execution output
Document Processor	Extracts and analyzes documents	Document/query	Processed information
API Service	Accesses external services	API request	Service response

3.4 Execution Monitoring, Verification, and Error Recovery

The AAIF uses a closed-loop monitoring process to evaluate each executed action before proceeding to the next step. The framework examines execution status, output relevance, task progress, detected errors, and whether replanning is required. The execution state is defined as

$$z_i \in \{\text{Success}, \text{Partial}, \text{Failure}\}$$

If $z_i = \text{Failure}$, the agent identifies the error, determines whether it originates from planning, tool selection, or tool execution, revises the current subtask, selects an alternative tool or action, and retries within a predefined limit.

The computational evaluation configuration is summarized in Table 3. The framework is evaluated using 500 tasks across five categories, covering simple, moderate, and complex task levels. The setup uses a maximum of 10 planning steps, 3 retries per failed action, 6 available tools, and 5 experimental repetitions. Performance is assessed using task success rate, tool-selection accuracy, planning success, execution efficiency, execution time, and recovery success rate.

Table 3. Computational Evaluation Configuration

Parameter	Setting
Task Categories	5
Total Evaluation Tasks	500
Tasks per Category	100
Complexity Levels	Simple, Moderate, Complex
Maximum Planning Steps	10
Maximum Retries per Failed Action	3
Available Tools	6
Baseline Methods	Direct LLM, Fixed-Tool LLM, Planning-Only Agent
Proposed Method	AAIF
Experimental Repetitions	5
Evaluation Metrics	Task success, tool-selection accuracy, planning success, execution efficiency, execution time, recovery success

4. Results And Discussion

4.1 Overall Autonomous Task-Completion Performance

The proposed AAIF achieved the strongest overall performance across the evaluated task categories. It obtained a 94.2% task success rate, compared with 73.8% for Direct LLM, 80.6% for Fixed-Tool LLM, and 86.7% for the Planning-Only Agent. For complex multi-step tasks, AAIF maintained an 89.4% completion rate, while the baseline methods showed larger reductions. As summarized in Table 4, AAIF also produced the lowest incomplete-task rate (3.7%) and failure rate (2.1%). These results indicate that integrated goal decomposition, dynamic tool selection, execution monitoring, and automatic recovery improve reliable autonomous task completion.

Table 4. Autonomous Task-Completion Performance

Method	Task Success Rate (%)	Complex-Task Success (%)	Incomplete Tasks (%)	Failure Rate (%)
Direct LLM	73.8	58.6	16.4	9.8
Fixed-Tool LLM	80.6	68.3	12.1	7.3

Planning-Only Agent	86.7	77.8	8.5	4.8
Proposed AAIF	94.2	89.4	3.7	2.1

4.2 Tool-Selection and Reasoning Performance

AAIF achieved 96.1% tool-selection accuracy and a 93.4% planning success rate, indicating reliable identification of suitable tools and execution sequences. Invalid tool calls were reduced to 2.3%, while valid execution paths reached 92.8%. The context-aware tool-scoring mechanism improved performance over fixed tool assignment by considering subtask relevance, compatibility, historical reliability, and execution efficiency. The improvement was particularly evident in complex workflows requiring multiple tools and dependent execution steps.

4.3 Execution Efficiency and Error-Recovery Performance

AAIF achieved an execution efficiency of 91.8%, with an average execution time of 2.84 s per task. Although Direct LLM required less execution time, it showed considerably lower autonomous task completion. AAIF also achieved a 92.6% recovery success rate, demonstrating that most failed intermediate actions could be corrected through replanning or alternative tool selection. Figure 3 compares Direct LLM, Fixed-Tool LLM, Planning-Only Agent, and AAIF across task success, tool-selection accuracy, planning success, and execution efficiency.

$$\eta = \frac{N_{\text{successful actions}}}{N_{\text{total actions}}} \quad , \_\_\_\_\_ \quad (6)$$

where ηrepresents execution efficiency.

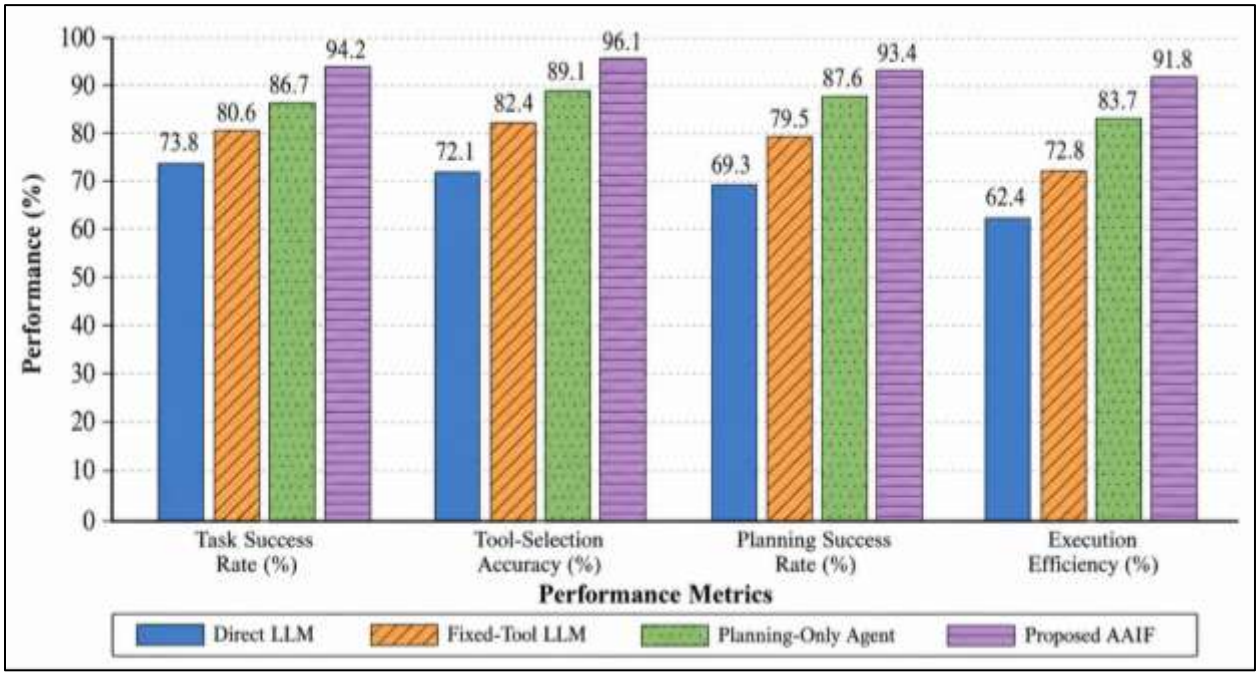

Figure 4. Performance Comparison of Direct LLM, Fixed-Tool LLM, Planning-Only Agent, and Proposed AAIF

4.4 Complexity Analysis, Ablation Study, and Overall Discussion

AAIF maintained stronger performance as task complexity increased, particularly for complex multi-tool workflows. As summarized in Table 5, the ablation analysis showed that removing individual components reduced task success. AAIF without autonomous planning achieved 82.5%, while removing dynamic tool selection resulted in 87.9%, and removing error recovery achieved 90.6%. The Full AAIF obtained the highest task success rate of 94.2%. These results confirm that autonomous planning contributes most strongly to performance, while dynamic tool selection and error recovery provide additional improvements in execution reliability.

Table 5. Ablation Analysis of AAIF

Configuration	Task Success Rate (%)
AAIF without Autonomous Planning	82.5
AAIF without Dynamic Tool Selection	87.9
AAIF without Error Recovery	90.6
Full AAIF	**94.2**

5. Conclusion

This study addressed the challenge of transforming Large Language Models from passive text-generation systems into autonomous agents capable of executing complex multi-step tasks. Conventional direct prompting and rigid tool-use strategies remain limited by weak planning, inappropriate tool selection, execution errors, and limited recovery capability. To overcome these limitations, the proposed Agentic AI Framework (AAIF) integrates autonomous goal decomposition, multi-step reasoning, dynamic tool selection, external-tool execution, execution-state verification, and automatic error recovery within a unified closed-loop architecture. The framework achieved a 94.2% task success rate, 96.1% tool-selection accuracy, 93.4% planning success rate, 91.8% execution efficiency, and 92.6% recovery success rate, demonstrating strong performance, particularly for complex multi-tool workflows. Although AAIF introduces additional computational and reasoning overhead, with an average execution time of 2.84 s per task, this cost is balanced by improved reliability and autonomous completion. The study is limited by controlled computational evaluation and dependence on the reliability of the underlying LLM and external tools. Future work should evaluate AAIF in larger real-world task environments with heterogeneous tools, stronger security controls, permission-aware execution boundaries, long-term memory, and adaptive planning mechanisms.

References

1. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877-1901.
2. Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., ... & Neubig, G. (2023, July). Pal: Program-aided language models. In *International conference on machine learning* (pp. 10764-10799). PMLR.
3. Lee, J. S. (2025). InstructPatentGPT: training patent language models to follow instructions with human feedback. *Artificial Intelligence and Law*, *33*(3), 739-782.
4. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, *33*, 9459-9474.
5. Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., ... & Tang, J. (2024, May). Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations* (Vol. 2024, pp. 52989-53046).
6. Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., ... & Tang, J. (2024, May). Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations* (Vol. 2024, pp. 52989-53046).
7. Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., ... & Clark, P. (2023). Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, *36*, 46534-46594.
8. Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023, October). Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology* (pp. 1-22).
9. Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023, October). Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology* (pp. 1-22).
10. Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., ... & Sun, M. (2024, May). Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations* (Vol. 2024, pp. 9695-9717).
11. Rajan, C. (2026). Congestion- and fault-severity-aware escape routing with local recovery for multicore networks-on-chip. *VSF Journal of Electronics and Communication Engineering, 1*(1), 30–37.
12. Shen, Y., Song, K., Tan, X., Li, D., Lu, W., & Zhuang, Y. (2023). Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, *36*, 38154-38180.
13. Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., & Yao, S. (2023). Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, *36*, 8634-8652.

14. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, *35*, 24824-24837.
15. Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., & Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, *36*, 11809-11822.