Review

Not peer-reviewed version

Large Language Model Agents: A Comprehensive Survey on Architectures, Capabilities, and Applications

Yiming Lei , Jiawei Xu , Chia Xin Liang , Ziqian Bi , Xiaoming Li , Danyang Zhang , Junhao Song , Zhenyu Yu *

Posted Date: 24 December 2025

doi: 10.20944/preprints202512.2119.v1

Keywords: large language models; autonomous agents; reasoning; tool use; multi-agent systems; memory augmentation; agent benchmarks

Review

Large Language Model Agents: A Comprehensive Survey on Architectures, Capabilities, and Applications

Yiming Lei [1], Jiawei Xu [2], Chia Xin Liang [3], Ziqian Bi [4], Xiaoming Li [5], Danyang Zhang [6], Junhao Song [7] and Zhenyu Yu [8,*]

[1] Georgia Institute of Technology, Atlanta, GA, USA
[2] Purdue University, West Lafayette, IN, USA
[3] National Chung Hsing University, Taichung, Taiwan
[4] Beijing University of Technology, Beijing, China
[5] University Canada West, Vancouver, BC, Canada
[6] Independent Researcher, USA
[7] Heriot-Watt University, Edinburgh, UK
[8] University of Malaya, Kuala Lumpur, Malaysia
* Correspondence: yuzhenyuyxl@foxmail.com

Abstract

Large Language Model (LLM) agents represent a paradigm shift in artificial intelligence, combining the remarkable reasoning capabilities of foundation models with the ability to perceive environments, make decisions, and take actions autonomously. This comprehensive survey provides an in-depth examination of LLM-based agents across multiple dimensions. We first establish a formal definition of LLM agents and trace their evolution from early language models to today's sophisticated autonomous systems. We then present a novel taxonomy that organizes the field into four fundamental categories: reasoning-enhanced agents that leverage chain-of-thought and tree-structured deliberation; tool-augmented agents that extend LLM capabilities through external APIs and knowledge bases; multi-agent systems that enable collaborative problem-solving through inter-agent communication; and memory-augmented agents that maintain persistent context across interactions. For each category, we analyze representative architectures, discuss key innovations, and evaluate their relative strengths and limitations. We further examine diverse applications spanning software engineering, scientific research, embodied robotics, and web automation, supported by systematic comparisons on established benchmarks including SWE-bench, WebArena, and AgentBench. Our analysis reveals that while current agents achieve impressive performance on structured tasks, significant challenges remain in areas such as long-horizon planning, hallucination mitigation, and safe deployment. We conclude by identifying promising research directions, including neuro-symbolic integration, multi-modal perception, and human-agent collaboration frameworks, providing a roadmap for advancing this rapidly evolving field.

Keywords: large language models; autonomous agents; reasoning; tool use; multi-agent systems; memory augmentation; agent benchmarks

1. Introduction

The emergence of Large Language Models (LLMs) has fundamentally transformed the landscape of artificial intelligence, enabling machines to understand, generate, and reason with natural language at unprecedented levels of sophistication [1–6]. Building upon these capabilities, a new paradigm has emerged: LLM-based agents that can autonomously perceive their environment, formulate plans, and execute actions to achieve complex goals. This evolution represents a significant departure from traditional language models that merely respond to queries, moving towards systems that actively interact with the world and accomplish tasks that previously required human intervention.

The concept of autonomous agents has long been a central pursuit in artificial intelligence research, with early work exploring rule-based systems, reinforcement learning agents, and planning algorithms [7]. However, the integration of LLMs as the cognitive backbone of these agents has catalyzed remarkable advances in recent years. The introduction of ChatGPT in November 2022 [8–10] and subsequent models such as GPT-4 [3,11,12] demonstrated that large-scale language models possess emergent capabilities including few-shot learning, chain-of-thought reasoning, and instruction following that make them particularly well-suited for agent applications. These developments have sparked an explosion of research into LLM agents, with seminal works such as ReAct [13], Toolformer [14–20], and Generative Agents [21,22] establishing foundational paradigms for reasoning, tool use, and social simulation respectively.

The rapid proliferation of LLM agent research has created a pressing need for systematic organization and analysis of this burgeoning field. While several surveys have examined specific aspects of LLM agents [23–28], the field continues to evolve at such a pace that comprehensive coverage remains challenging. Moreover, the diversity of approaches spanning reasoning enhancement, tool augmentation, multi-agent collaboration, and memory systems necessitates a unified framework for understanding the relationships between different methodologies and identifying opportunities for cross-pollination.

This survey aims to provide a comprehensive and structured overview of LLM-based agents that addresses these challenges. Our contributions can be summarized as follows. First, we establish a formal definition of LLM agents and present a novel taxonomy that organizes the field into four fundamental categories: reasoning-enhanced agents, tool-augmented agents, multi-agent systems, and memory-augmented agents. This taxonomy provides a principled framework for understanding the diverse landscape of agent architectures while highlighting the complementary nature of different approaches. Second, we conduct an in-depth analysis of representative methods within each category, examining their technical innovations, theoretical foundations, and empirical performance. Our analysis encompasses over 100 papers spanning the period from 2020 to 2024, with particular emphasis on developments since the introduction of instruction-tuned models. Third, we present a systematic comparison of agent capabilities across multiple benchmarks, including software engineering tasks (SWE-bench) [29,30], web automation (WebArena) [31], and general agent evaluation (AgentBench) [32,33]. This empirical analysis reveals both the impressive progress achieved and the substantial gaps that remain between current systems and human-level performance. Fourth, we identify key challenges facing the field, including issues of hallucination, long-horizon planning, safety, and scalability, and articulate promising research directions for addressing these limitations.

The remainder of this survey is organized as follows. Section 2 provides essential background on LLMs and establishes foundational concepts for understanding agent architectures. Section 3 presents our proposed taxonomy and provides an overview of the agent landscape. Sections 4 through 7 provide detailed analyses of reasoning-enhanced agents, tool-augmented agents, multi-agent systems, and memory-augmented agents respectively. Section 8 examines key application domains including software engineering, scientific research, and embodied AI. Section 9 presents benchmark comparisons and empirical analyses. Section 10 discusses current challenges and limitations, while Section 11 outlines future research directions. Finally, Section 12 concludes the survey with key takeaways and reflections on the trajectory of the field.

2. Background and Preliminaries

This section establishes the foundational concepts necessary for understanding LLM-based agents. We begin by reviewing the evolution of large language models, then formally define what constitutes an LLM agent, and finally introduce key evaluation metrics and paradigms.

2.1. Evolution of Large Language Models

The development of modern LLMs can be traced through several transformative milestones. The introduction of the Transformer architecture by Vaswani et al. [1,34] in 2017 established the selfattention mechanism as the foundation for scalable sequence modeling. Unlike recurrent architectures, Transformers enable parallel processing of input sequences and capture long-range dependencies through attention weights, making them particularly suitable for language understanding tasks. The subsequent development of BERT [2,35–37] demonstrated that bidirectional pre-training on large text corpora could yield powerful representations transferable to diverse downstream tasks.

The scaling of language models to billions of parameters marked a critical phase transition in capabilities [38–43]. GPT-3 with 175 billion parameters exhibited surprising emergent abilities including few-shot learning, where the model could adapt to new tasks from just a handful of examples provided in the prompt. This capability opened the door to in-context learning paradigms that would later prove essential for agent applications. The instruction-tuning revolution, exemplified by InstructGPT and ChatGPT [8,44,45], further enhanced model capabilities by training on human feedback to follow diverse instructions more reliably. These instruction-tuned models demonstrated markedly improved abilities to engage in multi-turn dialogue, follow complex instructions, and maintain coherent behavior across extended interactions.

The release of GPT-4 [3,46–49] in March 2023 represented another significant leap, achieving human-level performance on numerous professional examinations while exhibiting enhanced reasoning and reduced hallucination rates compared to predecessors. Concurrent developments in open-source models, particularly the Llama series [50] and Code Llama [51], have democratized access to powerful foundation models and spurred rapid innovation in the research community. The emergence of models specifically designed for safety through techniques like Constitutional AI [52] has also addressed critical concerns about deploying LLMs in autonomous agent settings. Recent advances in generative modeling [53,54] and natural language processing have further expanded the capabilities available to agent systems.

2.2. Definition of LLM Agents

We define an LLM agent as a system that employs a large language model as its core reasoning engine to perceive its environment, make decisions, and execute actions in pursuit of specified goals. Formally, an LLM agent can be characterized by the tuple $\mathcal{A} = (\mathcal{M}, \mathcal{E}, \mathcal{S}, \mathcal{A}, \pi)$, where $\mathcal{M}$ denotes the underlying language model, $\mathcal{E}$ represents the environment with which the agent interacts, $\mathcal{S}$ is the state space capturing relevant environmental and internal states, $\mathcal{A}$ is the action space available to the agent, and $\pi : \mathcal{S} \to \mathcal{A}$ is the policy function that maps states to actions.

The agent operates through an iterative perception-reasoning-action loop. At each timestep t, the agent receives an observation o_t from the environment, which is combined with the agent's memory m_t and task specification g to form the current state $s_t = (o_t, m_t, g)$. The LLM then processes this state, typically through prompting or fine-tuned inference, to produce a reasoning trace r_t and select an action $a_t = \pi(s_t)$. The action is executed in the environment, producing a new observation o_{t+1} and potentially updating the agent's memory to m_{t+1}. This cycle continues until a termination condition is satisfied or a maximum number of steps is reached.

Formally, the agent's policy π can be expressed as a conditional probability distribution over actions given the current context:

$$\pi(a_t|s_t) = P_{\mathcal{M}}(a_t|\text{prompt}(o_t, m_t, g)), \tag{1}$$

where $P_{\mathcal{M}}$ denotes the probability distribution induced by the language model $\mathcal{M}$, and $\text{prompt}(\cdot)$ is a function that formats the inputs into a textual prompt. The agent's trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$ is generated by iteratively sampling actions and observing state transitions:

$$\tau \sim \prod_{t=0}^{T-1} \pi(a_t|s_t) \cdot P(s_{t+1}|s_t, a_t), \tag{2}$$

where $P(s_{t+1}|s_t, a_t)$ represents the environment dynamics. The objective is typically to maximize expected cumulative reward:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)\right], \tag{3}$$

where $R(s_t, a_t)$ is the reward function and $\gamma \in [0, 1]$ is a discount factor.

This formulation distinguishes LLM agents from traditional LLM applications in several important ways. First, agents maintain persistent state across multiple interaction steps, enabling them to pursue multi-step goals rather than responding to isolated queries. Second, agents possess the ability to take actions that modify their environment, whether through tool invocation, code execution, or physical manipulation in embodied settings. Third, agents typically employ explicit reasoning mechanisms that make their decision-making process more interpretable and controllable than end-to-end neural approaches.

2.3. Core Agent Components

Modern LLM agents typically comprise several interconnected components that collectively enable autonomous behavior. The perception module is responsible for processing environmental observations and converting them into representations suitable for the LLM. In text-based environments, this may involve parsing structured outputs from tools or APIs, while in visual domains, it may require integration with vision encoders or multimodal models. The reasoning module, implemented through the LLM itself, processes the current state and generates plans, decisions, and verbal reasoning traces. Advanced reasoning strategies such as chain-of-thought prompting [55] and tree-of-thought exploration [56] enhance this component's ability to handle complex, multi-step problems.

The memory module maintains information across interaction steps, enabling the agent to learn from experience and maintain context over extended horizons. Memory systems range from simple conversation histories stored in the LLM's context window to sophisticated architectures with hierarchical storage and retrieval mechanisms [57]. The action module translates the LLM's decisions into concrete operations, which may include generating text responses, invoking external tools, executing code, or sending commands to physical actuators. Finally, the planning module orchestrates high-level strategy by decomposing complex goals into manageable subgoals and coordinating their execution over time.

Figure 1 illustrates the general architecture of an LLM agent, showing the interconnections between core components and the cyclic nature of agent operation.

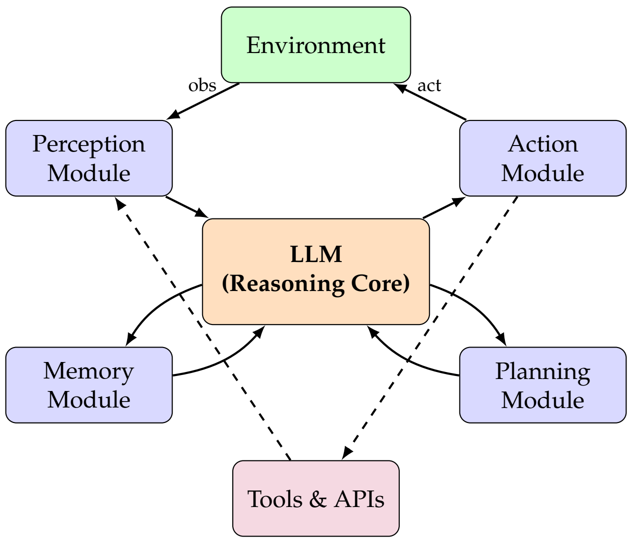

Figure 1. General architecture of an LLM agent. The LLM serves as the central reasoning core, coordinating perception, memory, planning, and action modules in a cyclic interaction with the environment.

2.4. Evaluation Paradigms

Evaluating LLM agents presents unique challenges compared to traditional NLP benchmarks due to the open-ended nature of agent tasks and the difficulty of defining success criteria for complex goals. Several evaluation paradigms have emerged to address these challenges. Task completion metrics assess whether agents successfully achieve specified objectives, typically measured through binary success rates or partial completion scores. For software engineering tasks, success may be determined by whether generated code passes test suites [29], while for web navigation, success depends on reaching target states in the browser environment [31].

Efficiency metrics capture the resources consumed by agents during task execution, including the number of interaction steps, API calls, or tokens processed. These metrics are particularly important for practical deployment where computational costs and latency must be balanced against task performance. Behavioral metrics assess qualitative aspects of agent behavior, such as the coherence of reasoning traces, the appropriateness of tool selections, and the safety of generated actions. Human evaluation remains essential for many of these aspects, though efforts to develop automatic behavioral evaluation are ongoing.

Multi-dimensional benchmarks such as AgentBench [32] evaluate agents across diverse environments including operating systems, databases, web browsers, and games, providing a more comprehensive picture of general agent capabilities. These benchmarks reveal that current agents exhibit substantial variation in performance across domains, with strong results on some tasks coexisting with failures on others that humans find straightforward.

3. Taxonomy and Overview

This section presents our proposed taxonomy for organizing the diverse landscape of LLM-based agents. We categorize existing approaches into four fundamental dimensions based on their primary mechanisms for extending LLM capabilities: reasoning enhancement, tool augmentation, multi-agent collaboration, and memory augmentation. While many practical systems combine multiple dimensions, this decomposition provides a principled framework for understanding the core innovations driving the field.

3.1. Taxonomy Overview

Figure 2 illustrates our proposed taxonomy, which organizes LLM agents along four complementary dimensions. Reasoning-enhanced agents focus on improving the quality of LLM decision-making through structured prompting strategies and deliberative search algorithms. Tool-augmented agents extend the action space of LLMs by enabling interaction with external resources including APIs, databases, code interpreters, and web browsers. Multi-agent systems distribute tasks across multiple LLM instances that collaborate through communication protocols, role-playing, and debate. Memory-augmented agents address the limited context windows of LLMs by implementing external memory systems that enable long-term information retention and retrieval.

These four dimensions are complementary rather than mutually exclusive, and state-of-the-art agent systems typically integrate capabilities from multiple categories. For instance, the ReAct framework [13] combines reasoning traces with action execution, while systems like AutoGPT [58] and LangChain [59] integrate all four dimensions into unified architectures. Understanding each dimension in isolation, however, provides crucial insights into the design space and enables principled composition of capabilities.

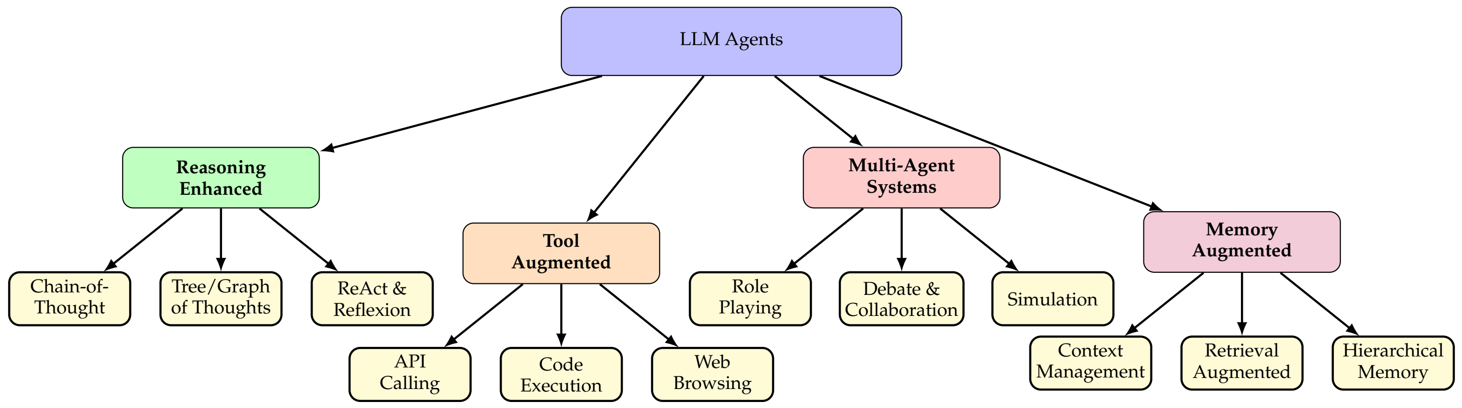

Figure 2. Taxonomy of LLM-based agents. The proposed classification organizes existing methods into four fundamental categories based on their primary mechanisms for extending LLM capabilities. Each category encompasses distinct subcategories that address specific aspects of agent design.

3.2. Reasoning-Enhanced Agents

Reasoning-enhanced agents improve LLM decision-making by structuring the generation process to elicit more deliberate and accurate responses. The chain-of-thought (CoT) prompting paradigm [55] demonstrated that including step-by-step reasoning examples in prompts substantially improves performance on arithmetic, symbolic, and commonsense reasoning tasks. Zero-shot variants [60] achieve similar benefits through simple trigger phrases like "Let's think step by step," suggesting that LLMs possess latent reasoning capabilities that can be activated through appropriate prompting.

Beyond linear chains, tree-structured and graph-structured approaches enable exploration of multiple reasoning paths. Tree of Thoughts [56] maintains a tree of partial solutions that can be expanded through breadth-first or depth-first search, with the LLM itself serving as both the generator and evaluator of candidate thoughts. Graph of Thoughts [61] further generalizes this framework by allowing arbitrary connections between thought nodes, enabling operations such as aggregation and refinement that are not possible in tree structures. Self-consistency [62] takes a complementary approach by sampling multiple reasoning chains and selecting the most consistent answer through majority voting.

The ReAct paradigm [13] synergizes reasoning and acting by interleaving verbal reasoning traces with environment actions in an alternating fashion. This approach enables the model to ground its reasoning in observations while using reasoning to inform action selection, achieving strong performance on question answering and interactive decision-making tasks. Reflexion [63] extends this paradigm by incorporating self-reflection, where agents verbally analyze their failures and use these reflections to improve subsequent attempts without any gradient updates.

3.3. Tool-Augmented Agents

Tool-augmented agents extend the capabilities of LLMs by enabling them to invoke external tools and services. This approach addresses fundamental limitations of pure language models, including their inability to access current information, perform precise calculations, and interact with external systems. Toolformer [14] pioneered the self-supervised approach to tool use, training models to autonomously decide when and how to invoke tools such as calculators, search engines, and translation systems.

API-calling capabilities have been substantially enhanced through dedicated training and evaluation frameworks. ToolLLM [64] introduces a comprehensive dataset of over 16,000 real-world APIs along with a training recipe that enables open-source models to achieve competitive tool-use performance. Gorilla [65] specifically targets API documentation retrieval and accurate API call generation, demonstrating that retrieval-augmented approaches can reduce hallucination in tool invocation. The introduction of function calling capabilities in commercial APIs such as OpenAI's function calling [66] has further standardized the interface between LLMs and external tools.

Code execution represents a particularly powerful form of tool augmentation, enabling agents to leverage the full expressivity of programming languages. Program-aided language models (PAL) [67] decompose reasoning problems into programmatic solutions that are executed by interpreters, achieving state-of-the-art results on mathematical reasoning benchmarks. WebGPT [68] and subsequent systems demonstrate that web browsing capabilities enable agents to access current information and verify factual claims, substantially reducing hallucination rates on knowledge-intensive tasks.

3.4. Multi-Agent Systems

Multi-agent systems distribute problem-solving across multiple LLM instances that interact through structured communication protocols. This approach offers several advantages including task decomposition, diversity of perspectives, and emergent collaborative behaviors. The CAMEL framework [69] introduced role-playing as a mechanism for autonomous agent cooperation, where agents assume complementary roles such as instructor and assistant to collaboratively solve tasks without human intervention.

Generative Agents [21] demonstrated the potential for LLM-based agents to exhibit believable social behaviors in simulated environments. By equipping agents with memory systems, reflection capabilities, and planning modules, the researchers created a small town populated by 25 agents that autonomously formed relationships, spread information, and coordinated activities. This work highlighted the emergent properties that can arise from interactions between multiple LLM agents and opened new avenues for social simulation research.

Software development has emerged as a compelling application domain for multi-agent collaboration. MetaGPT [70] organizes multiple agents into a simulated software company with roles including product manager, architect, and engineer, achieving strong performance on code generation benchmarks. ChatDev [71] similarly structures agents according to the software development lifecycle, enabling collaborative design, coding, testing, and documentation. AutoGen [72] provides a flexible framework for defining custom multi-agent conversations, supporting diverse patterns including sequential chat, group discussion, and hierarchical delegation.

3.5. Memory-Augmented Agents

Memory-augmented agents address the fundamental limitation of finite context windows by implementing external memory systems that persist across interactions. Retrieval-augmented generation (RAG) [7] established the foundational paradigm by combining parametric knowledge stored in model weights with non-parametric knowledge retrieved from external databases. This approach enables models to access current information and domain-specific knowledge without requiring retraining.

MemGPT [57] takes inspiration from operating system memory hierarchies to implement a tiered memory architecture for LLM agents. The system distinguishes between main context (analogous to RAM) and external context (analogous to disk storage), with the agent autonomously managing data movement between tiers through function calls. This enables effectively unbounded context while maintaining the responsiveness of limited context windows for immediate interactions.

Generative Agents [21] implement a sophisticated memory architecture that includes a memory stream for recording observations, a retrieval mechanism that combines recency, importance, and relevance, and a reflection process that synthesizes higher-level insights from accumulated memories. This architecture enables agents to maintain coherent personalities and remember past interactions over extended simulation periods, demonstrating the importance of structured memory for believable agent behavior.

Table 1 provides a comprehensive comparison of representative methods across the four taxonomic categories, summarizing their key characteristics including primary mechanisms, training requirements, and notable capabilities.

Table 1. Comprehensive Comparison of Representative LLM Agent Methods.

Method	**Category**	**Mechanism**	**Training**	**LLM Calls**	**Key Strength**
Chain-of-Thought	Reasoning	Linear decomposition	None	1	Simple, broadly applicable
Tree of Thoughts	Reasoning	Tree search	None	O(b^d)	Deliberative exploration
Self-Consistency	Reasoning	Multiple sampling	None	k samples	Robust via voting
ReAct	Reasoning	Reason + Act	None	Per step	Grounded reasoning
Reflexion	Reasoning	Self-reflection	None	Multi-trial	Learning from failures
Toolformer	Tool	Self-supervised	Fine-tune	1	Autonomous tool use
Gorilla	Tool	Retrieval + API	Fine-tune	1	Accurate API calls
ToolLLM	Tool	DFSDT search	Fine-tune	Multi-step	Large API scale
HuggingGPT	Tool	Model orchestration	None	Multi-model	Multimodal composition
CAMEL	Multi-agent	Role-playing	None	Dialogue	Autonomous cooperation
MetaGPT	Multi-agent	SOP workflow	None	Multi-role	Structured outputs
Gen. Agents	Multi-agent	Social simulation	None	Per agent	Emergent behavior
AutoGen	Multi-agent	Flexible chat	None	Configurable	General framework
RAG	Memory	Retrieval + Gen.	Optional	1	External knowledge
MemGPT	Memory	Virtual context	None	Per operation	Unbounded context

3.6. Integration and Frameworks

The practical deployment of LLM agents typically requires integration across multiple taxonomic dimensions. LangChain [59], launched in October 2022, emerged as one of the first comprehensive frameworks for building agent applications, providing abstractions for chains, tools, memory, and agent executors. The framework's modular design enables flexible composition of components while standardizing interfaces between different subsystems.

AutoGPT [58] garnered significant attention in early 2023 as one of the first demonstrations of fully autonomous agent behavior, automatically decomposing goals into subtasks and iteratively working toward completion. While limited by tendency toward loops and high API costs, AutoGPT demonstrated the potential for end-to-end autonomous systems and inspired numerous follow-up projects. More recent frameworks such as TaskWeaver [73] adopt code-first approaches that leverage the expressivity of programming languages for agent orchestration, while AgentVerse [74] provides infrastructure for both task-solving and simulation applications of multi-agent systems.

4. Reasoning-Enhanced Agents

Reasoning-enhanced agents represent a fundamental category of LLM agents that improve decision-making quality through structured prompting strategies and deliberative search algorithms [75,76]. This section provides a comprehensive analysis of reasoning paradigms, beginning with chain-of-thought approaches and progressing through increasingly sophisticated tree and graph-based methods.

4.1. Chain-of-Thought Reasoning

The chain-of-thought (CoT) paradigm introduced by Wei et al. [55] demonstrated that prompting large language models with step-by-step reasoning examples substantially improves performance on complex reasoning tasks. The key insight is that intermediate reasoning steps, rather than directly predicting final answers, enable models to decompose problems and apply learned reasoning patterns more effectively. Empirical evaluation on arithmetic reasoning benchmarks such as GSM8K showed that CoT prompting with PaLM-540B achieved state-of-the-art accuracy of 56.9%, compared to only 17.9% without chain-of-thought, representing a remarkable improvement attributable solely to prompting strategy.

Zero-shot chain-of-thought [60,77] extended this paradigm by showing that the simple prompt "Let's think step by step" can elicit reasoning behavior without task-specific examples. This finding suggested that large language models possess latent reasoning capabilities that require only minimal triggering to activate. The zero-shot approach increased accuracy on MultiArith from 17.7% to 78.7% and on GSM8K from 10.4% to 40.7% using InstructGPT, demonstrating broad applicability across reasoning domains. Subsequent work identified that the effectiveness of trigger phrases varies across tasks, motivating research into automatic prompt optimization.

Self-consistency [62] introduced a complementary enhancement by sampling multiple diverse reasoning paths and aggregating their conclusions through majority voting. This approach leverages the intuition that complex problems typically admit multiple valid solution strategies, and consistency across different approaches provides stronger evidence for correctness than any single chain. Formally, given a question q, self-consistency samples k reasoning chains $\{(r_1, a_1), \ldots, (r_k, a_k)\}$ and selects the most frequent answer:

$$\hat{a} = \arg\max_{a} \sum_{i=1}^{k} \mathbf{1}[a_i = a], \tag{4}$$

Self-consistency improved CoT performance by 17.9% on GSM8K, 11.0% on SVAMP, and 12.2% on AQuA, establishing it as a standard component in reasoning agent architectures. The method requires no additional training and works with any language model capable of chain-of-thought reasoning.

Least-to-most prompting [78] addresses the challenge of generalizing to problems more complex than those seen in prompts. The approach decomposes complex problems into simpler subproblems that are solved sequentially, with each solution informing subsequent steps. On the SCAN benchmark for compositional generalization, least-to-most prompting achieved 99.7% accuracy on the length generalization split, compared to only 16% for standard chain-of-thought, demonstrating superior ability to handle novel problem complexities. Plan-and-solve prompting [79] similarly addresses multi-step reasoning by first devising a plan that divides the task into subtasks, then executing the plan step by step, consistently outperforming zero-shot CoT across diverse benchmarks.

4.2. Tree and Graph-Based Deliberation

While chain-of-thought produces linear reasoning sequences, many problems benefit from exploring multiple solution paths and comparing alternatives [80]. Tree of Thoughts (ToT) [56] generalizes CoT by maintaining a tree of partial solutions that can be expanded through breadth-first or depth-first search. Recent work on knowledge-augmented planning [81] addresses planning hallucination by incorporating explicit action knowledge. Each node in the tree represents an intermediate reasoning state, and the LLM serves dual roles as both generator of candidate expansions and evaluator of their promise. This enables deliberate exploration and backtracking when initial attempts prove unfruitful.

The ToT framework can be formalized as a search over a tree $\mathcal{T} = (V, E)$ where each node $v \in V$ represents a partial solution state. Given a state s, the LLM generates candidate thoughts $\{t_1, \ldots, t_b\}$ with branching factor b, and evaluates each through a value function $V(s, t)$:

$$V(s,t) = P_{\mathcal{M}}(\text{“promising”}|s,t,\text{eval\_prompt}), \tag{5}$$

The search algorithm (BFS or DFS) expands nodes based on these values until reaching a solution or exhausting the search budget.

ToT demonstrated dramatic improvements on tasks requiring strategic planning and exploration. On the Game of 24, where the objective is to combine four numbers using arithmetic operations to obtain 24, GPT-4 with standard prompting solved only 4% of problems while ToT achieved 74% success rate. Similarly, on creative writing tasks requiring coherent paragraph generation with constraints, ToT produced outputs that human evaluators rated significantly higher than those from chain-of-thought or standard prompting. The framework's flexibility allows customization of search algorithms and evaluation criteria for different problem domains.

Graph of Thoughts (GoT) [61] further extends the deliberation paradigm by representing reasoning as an arbitrary directed graph rather than a tree. This enables operations not possible in tree structures, including aggregation of multiple thought branches into unified conclusions and refinement of earlier thoughts based on subsequent discoveries. GoT improved sorting quality by 62% over ToT while reducing computational costs by over 31%, demonstrating that more flexible reasoning topologies can simultaneously improve quality and efficiency. The framework has been applied to diverse tasks including document merging, set operations, and keyword extraction.

Self-ask [82] introduced a related approach where the model explicitly asks and answers follow-up questions before addressing the main query. This compositional structure naturally decomposes multi-hop reasoning into single-hop subquestions that can be answered individually. The structured format also enables seamless integration with search engines, where follow-up questions can be routed to external information sources rather than relying solely on the model's parametric knowledge. This combination of self-decomposition and retrieval augmentation achieved strong results on multi-hop question answering benchmarks.

Figure 3 illustrates the structural differences between Chain-of-Thought, Tree of Thoughts, and Graph of Thoughts reasoning paradigms.

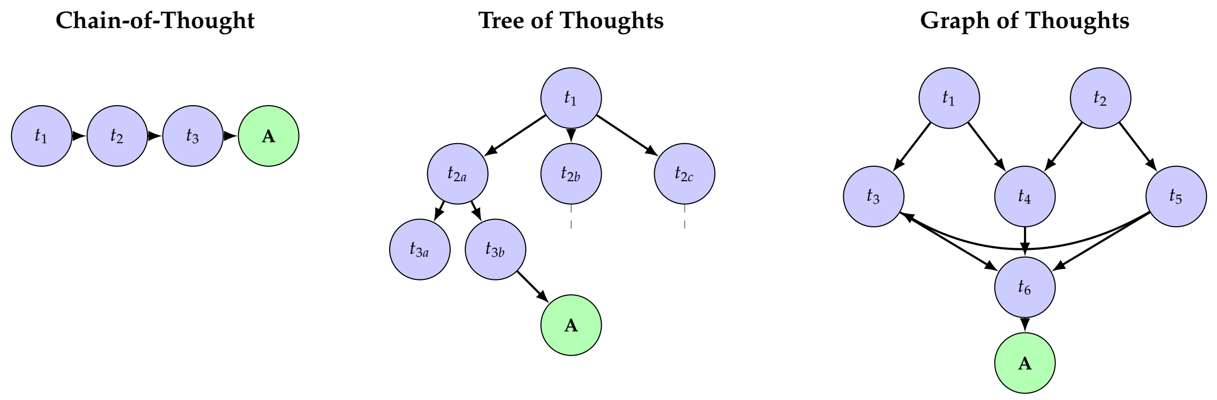

Figure 3. Comparison of reasoning topologies. Chain-of-Thought follows a linear sequence; Tree of Thoughts enables branching exploration with backtracking; Graph of Thoughts allows arbitrary connections including aggregation and refinement loops.

4.3. Reasoning with Action

The ReAct paradigm [13] represents a pivotal advancement that synergizes reasoning and acting in an interleaved fashion. Rather than separating reasoning and action into distinct phases, ReAct agents alternate between generating verbal reasoning traces that interpret observations and plan next steps, and taking actions that query external sources or modify the environment. This tight coupling enables reasoning to be grounded in concrete observations while actions are informed by deliberate reasoning.

Figure 4 illustrates the ReAct interaction loop, showing how thought, action, and observation interleave to solve complex tasks.

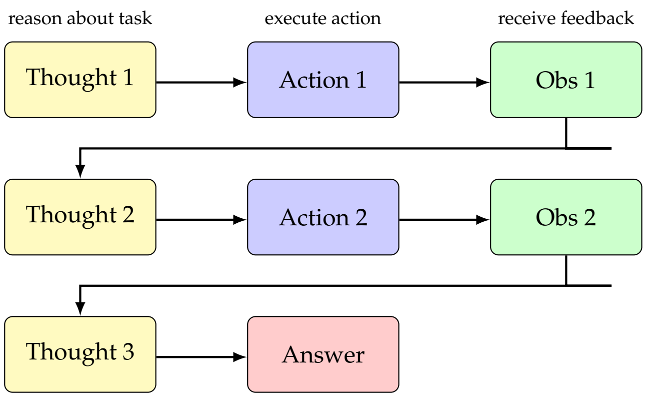

Figure 4. The ReAct interaction loop. Agents alternate between generating thoughts (reasoning), taking actions, and receiving observations, enabling grounded problem-solving.

ReAct achieved notable top 5% recognition at ICLR 2023 and demonstrated strong performance across diverse benchmarks. On HotPotQA [83], ReAct with GPT-3 outperformed both pure reasoning (chain-of-thought) and pure acting (action generation without reasoning traces) approaches, achieving a 6% improvement in exact match score over the best baseline. The framework proved particularly effective when actions could provide new information relevant to the reasoning process, creating a productive feedback loop between thought and observation.

Reflexion [63] extends ReAct by incorporating explicit self-reflection after task completion or failure. Rather than immediately attempting tasks again, Reflexion agents generate verbal analyses of what went wrong and how to improve, storing these reflections in an episodic memory buffer. On subsequent attempts, the agent conditions its behavior on accumulated reflections, enabling learning from experience without gradient updates. This "verbal reinforcement learning" achieved significant improvements on code generation benchmarks, including state-of-the-art performance on HumanEval with 91% pass@1 accuracy.

Self-Refine [84] demonstrates that iterative self-improvement can enhance outputs across diverse tasks. The approach generates an initial output, then uses the same LLM to provide feedback and refine the output, repeating this cycle until convergence or a maximum iteration count. Across seven tasks ranging from dialogue generation to mathematical reasoning, Self-Refine improved task performance by approximately 20% absolute compared to single-shot generation. The method requires no additional training data and works with any sufficiently capable language model, making it broadly applicable.

CRITIC [85] addresses the observation that LLMs alone cannot reliably verify their own outputs without external grounding. The framework enables models to validate and revise outputs through interaction with external tools such as search engines and code interpreters. On question answering tasks, CRITIC with ChatGPT achieved 7.7 F1 improvement across three benchmarks, while on mathematical reasoning, it provided 7.0% absolute gains. The results highlight the critical importance of external feedback for enabling reliable self-correction in LLM agents.

4.4. *Agent Fine-Tuning for Reasoning*

While prompting-based approaches avoid the computational cost of training, fine-tuning can yield agents with more robust and efficient reasoning capabilities. FireAct [86] investigates fine-tuning language models on agent trajectories generated by more capable models such as GPT-4. The approach collects successful reasoning-action traces across multiple tasks and uses them to fine-tune smaller models in a distillation-like setup. Fine-tuning Llama2-7B with just 500 agent trajectories led to 77% improvement on HotPotQA while reducing inference costs by 70% compared to prompting GPT-4 directly.

The benefits of fine-tuning extend beyond performance to include improved robustness and reduced sensitivity to prompt variations. FireAct showed that fine-tuned models maintained strong performance even when observations were noisy or partially missing, with only 5.1% drop compared to 28.0% for prompted models in adversarial settings. Multi-method fine-tuning, where training data combines trajectories from different prompting strategies, can further improve generalization, though optimal combinations vary across base models. These findings suggest that agent fine-tuning represents a promising direction for developing practical agent systems with predictable behavior.

4.5. *Summary and Comparative Analysis*

Table 2 summarizes the key characteristics of reasoning-enhanced agent approaches. Chain-of-thought methods provide accessible improvements through prompting alone but are limited to linear reasoning. Tree and graph-based approaches enable more sophisticated exploration but incur higher computational costs. ReAct-style methods excel in interactive settings where reasoning can be grounded in observations, while fine-tuning approaches offer the best combination of performance and efficiency for deployment.

Table 2. Comparison of Reasoning-Enhanced Agent Approaches.

Method	Structure	Training	Search
Chain-of-Thought	Linear	None	None
Self-Consistency	Linear	None	Sampling
Tree of Thoughts	Tree	None	BFS/DFS
Graph of Thoughts	Graph	None	Custom
ReAct	Interleaved	None	None
Reflexion	Episodic	None	Retry
FireAct	Linear	Distillation	None

5. Tool-Augmented Agents

Tool-augmented agents extend the capabilities of large language models by enabling them to interact with external tools, APIs, and computational resources. This section examines the key paradigms for tool augmentation, including self-supervised tool learning, API integration frameworks, code execution engines, and web browsing capabilities.

5.1. *Foundations of Tool Use*

The integration of external tools addresses several fundamental limitations inherent to pure language models. First, LLMs possess only parametric knowledge frozen at training time, making them unable to access current information or domain-specific databases. Second, while LLMs can approximate arithmetic and symbolic reasoning, they remain prone to errors that even simple calculators would never make. Third, language models cannot directly interact with external systems such as databases, file systems, or web services. Tool augmentation addresses all three limitations by enabling LLMs to delegate appropriate subtasks to specialized tools while orchestrating the overall problem-solving process.

Toolformer [14] established the paradigm of self-supervised tool learning, where models learn when and how to invoke tools without explicit supervision. The approach trains models to insert API calls into text in positions where doing so improves perplexity on subsequent tokens. By optimizing for this self-supervised objective, the model learns to use tools including calculators, search engines, translation systems, and calendars in contextually appropriate ways. Toolformer achieved substantially improved zero-shot performance across diverse tasks, often matching or exceeding much larger models that lack tool access.

The mechanism underlying Toolformer involves first sampling potential API call positions and arguments from a prompted model, then filtering these candidates based on whether including the API response reduces perplexity. Formally, a tool call c at position i is retained if:

$$L_i(c) = L(x_{i:n}|x_{1:i-1}, c, r_c) - L(x_{i:n}|x_{1:i-1}) \geq \tau \tag{6}$$

where L denotes the language model loss, r_c is the tool response, and τ is a threshold. The surviving examples are used to fine-tune the model to generate tool calls autonomously. This self-supervised approach requires only a handful of demonstrations for each tool and can be extended to new tools without architectural changes. The key insight is that language models possess sufficient world knowledge to determine when tool assistance would be beneficial, and this knowledge can be distilled into tool-using behavior through targeted fine-tuning.

5.2. API Integration and Function Calling

The practical deployment of tool-augmented agents requires robust mechanisms for discovering, selecting, and invoking APIs. OpenAI's function calling capability [66], introduced in June 2023, established a standardized interface where developers describe available functions in JSON schema format, and the model outputs structured JSON arguments for function invocation. This approach separates the concerns of function selection and argument generation from actual execution, enabling flexible integration with arbitrary backend services while maintaining safety through application-controlled execution.

Gorilla [65] specifically targets the challenge of accurate API call generation, recognizing that hallucination of incorrect API names, parameters, or usage patterns represents a significant barrier to reliable tool use. The system combines fine-tuning on curated API documentation with retrieval augmentation, enabling adaptation to documentation changes at test time without retraining. Gorilla achieved state-of-the-art performance on API call accuracy, surpassing GPT-4 on the APIBench dataset while maintaining the flexibility to incorporate new APIs through document retrieval. The retriever-aware training approach ensures that the model learns to leverage retrieved documentation effectively rather than relying solely on memorized API knowledge.

ToolLLM [64] addresses the challenge of scaling tool use to thousands of real-world APIs. The framework constructs ToolBench, a comprehensive dataset covering over 16,000 APIs from the RapidAPI marketplace, along with automatically generated instructions and solution paths. To enable efficient navigation of large API spaces, ToolLLM introduces a depth-first search-based decision tree algorithm that enables models to explore multiple reasoning traces and backtrack from unsuccessful attempts. The resulting ToolLLaMA model demonstrates strong generalization to unseen APIs, achieving comparable performance to ChatGPT on tool-use tasks while enabling deployment with open-source models.

5.3. Code Execution and Program Synthesis

Code execution represents a particularly powerful modality for tool augmentation, as it enables agents to leverage the full expressivity of programming languages for computation, data manipulation, and system interaction. Program-aided language models (PAL) [67] pioneered the approach of generating programs as intermediate reasoning steps, with actual computation delegated to an interpreter. This separation of concerns allows the LLM to focus on problem decomposition and program structure while avoiding arithmetic errors that plague pure language model reasoning.

PAL demonstrated dramatic improvements on mathematical reasoning benchmarks. On GSM8K, PAL with Codex outperformed PaLM-540B using chain-of-thought by an absolute 15%, despite using a substantially smaller model. The key advantage is that generated programs can be verified for syntactic correctness and tested on examples, providing feedback mechanisms not available for free-form text reasoning. Moreover, programmatic solutions are inherently more precise, avoiding the ambiguity and approximation that can accumulate in natural language reasoning chains.

HuggingGPT [87], also known as JARVIS, extends code execution to orchestrate diverse AI models as tools. The system uses ChatGPT as a controller that plans task decomposition, selects appropriate models from the Hugging Face ecosystem, executes selected models on task inputs, and generates integrated responses. This meta-learning approach enables handling of complex multimodal tasks by composing specialized models for subtasks including image generation, speech recognition, and object detection. HuggingGPT successfully integrates hundreds of models across 24 task types, demonstrating the potential for LLMs to serve as general-purpose AI orchestrators.

TaskWeaver [73] adopts a code-first philosophy for agent orchestration, converting user requests into executable code that coordinates plugins and manages stateful computation. Unlike approaches that treat code as one tool among many, TaskWeaver uses code as the primary representation for plans and actions, leveraging programming language constructs for control flow, error handling, and data transformation. This design enables handling of complex data structures and domain-specific operations while maintaining interpretability through generated code artifacts.

Figure 5 illustrates the general pipeline for tool-augmented agent operation, showing how the LLM orchestrates tool discovery, selection, invocation, and result integration.

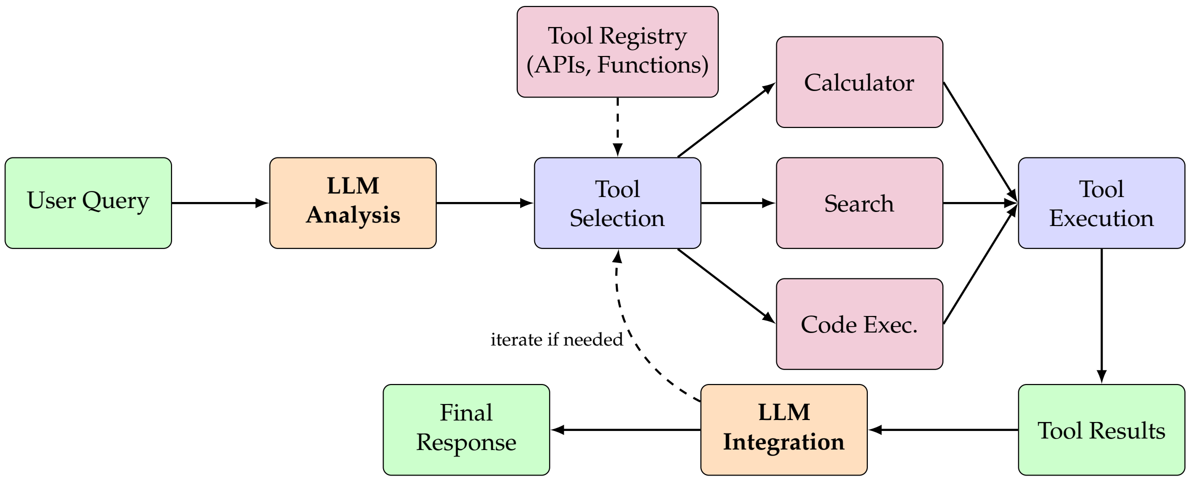

Figure 5. Tool-augmented agent pipeline. The LLM analyzes user queries, selects appropriate tools from a registry, executes them, and integrates results. An iterative loop enables multi-step tool use when initial results are insufficient.

5.4. Web Browsing and Information Retrieval

Web browsing capabilities enable agents to access current information, verify claims against authoritative sources, and interact with web-based services. WebGPT [68] established foundational approaches by fine-tuning GPT-3 to operate a text-based web browser for answering questions. The system learns to issue search queries, navigate search results, and extract relevant information through imitation learning on human demonstrations. Subsequent optimization with human feedback improved answer quality to the point where model responses were preferred over human-written answers 56% of the time.

WebGPT demonstrated substantial improvements in factual accuracy through its browsing capability. On TruthfulQA, WebGPT models outperformed all GPT-3 variants on both truthfulness and informativeness metrics. Critically, the percentage of truthful and informative answers increased with model size for WebGPT, unlike base GPT-3 where scaling provided minimal improvements. This finding suggests that grounding in retrieved information enables more beneficial scaling of factual knowledge capabilities.

Mind2Web [88] introduced the first large-scale benchmark for generalist web agents, comprising over 2,000 tasks across 137 real websites spanning 31 domains. The benchmark evaluates agents on their ability to follow natural language instructions to accomplish diverse web tasks including form filling, navigation, and information extraction. Evaluation revealed significant gaps between current models and human performance, with the best approaches achieving only around 50% accuracy on element selection compared to human levels exceeding 90%. This benchmark has catalyzed research into more capable web agents.

WebArena [31] advances web agent evaluation by providing a realistic, reproducible environment with self-hosted websites representing e-commerce, social forums, software development, and content management domains. The benchmark includes 812 tasks designed to emulate authentic human web use, with evaluation based on functional correctness rather than surface-level metrics. Experiments revealed that even GPT-4-based agents achieved only 14.41% task success compared to 78.24% human performance, highlighting substantial room for improvement. The controlled environment enables reproducible evaluation and systematic analysis of failure modes.

5.5. Multimodal Tool Integration

Recent advances have extended tool augmentation to multimodal settings where agents must process and generate content across text, images, audio, and video. Visual language models such as GPT-4V enable agents to perceive visual information directly, reducing reliance on separate vision tools. However, the integration of specialized tools for tasks such as image generation, video editing, and 3D modeling remains important for achieving high-quality outputs in specialized domains.

The orchestration of multimodal tools introduces unique challenges including modality alignment, format conversion, and quality assessment across different content types. Systems like HuggingGPT [87] address these challenges by maintaining explicit representations of input-output modalities for each available tool, enabling automatic planning of tool chains that respect modality constraints. The success of such systems demonstrates that LLMs can effectively coordinate diverse specialized tools to accomplish complex multimodal objectives.

5.6. Summary

Tool-augmented agents substantially extend the capabilities of language models by enabling access to external computation, information, and services. Table 3 summarizes key approaches across different tool modalities. The field continues to evolve rapidly, with ongoing work addressing challenges including tool discovery, composition, and verification.

Table 3. Comparison of Tool-Augmented Agent Approaches.

Method	Tool Type	Training	API Scale
Toolformer	Mixed	Self-sup.	5 tools
Gorilla	REST APIs	Fine-tune	1,600+
ToolLLM	REST APIs	Fine-tune	16,000+
PAL	Interpreter	None	1 tool
WebGPT	Browser	RLHF	1 tool
HuggingGPT	AI Models	None	100s

6. Multi-Agent Systems

Multi-agent systems distribute problem-solving across multiple LLM instances that interact through structured communication protocols [89,90]. This paradigm offers several advantages over single-agent approaches, including task decomposition through role specialization, diversity of perspectives that can improve solution quality, and emergent collaborative behaviors that mirror human teamwork. This section examines the foundational frameworks, communication protocols, and application patterns that characterize LLM-based multi-agent systems.

6.1. Role-Playing and Autonomous Cooperation

The CAMEL framework [69] introduced role-playing as a mechanism for enabling autonomous cooperation between LLM agents without human intervention. The system assigns complementary roles to different agents, such as an AI user that provides instructions and an AI assistant that executes them, and uses inception prompting to guide their interaction toward task completion. A task specifier agent first elaborates an initial idea into a well-defined task, which the user and assistant agents then collaboratively solve through multi-turn dialogue.

CAMEL revealed several challenges inherent to autonomous multi-agent cooperation. Conversation deviation occurs when agents gradually drift from the original task objective, potentially exploring tangential or irrelevant topics. Role flipping happens when agents confuse their assigned roles, with the assistant providing instructions or the user attempting to execute tasks. Establishing appropriate termination conditions proved difficult, as agents may declare completion prematurely or continue indefinitely without reaching solutions. These observations have informed subsequent work on more robust multi-agent coordination.

The DERA framework [91] specifically targets knowledge-intensive tasks by structuring dialog between two specialized agents: a Researcher that processes information and identifies crucial problem components, and a Decider that integrates the Researcher's findings to make final judgments. This division of labor mirrors expert consultation processes, where information gathering and synthesis are separated from decision-making. On medical question answering tasks, DERA demonstrated significant improvements over single-agent baselines, achieving GPT-4 level performance on the MedQA dataset with explicit reasoning traces that enhance interpretability.

6.2. Software Development with Multiple Agents

Software development has emerged as a compelling domain for multi-agent collaboration due to its inherent structure of specialized roles and well-defined workflows. MetaGPT [70] organizes multiple agents into a simulated software company with roles including product manager, architect, project manager, and engineer. The framework encodes standardized operating procedures (SOPs) into prompt sequences, guiding agents through structured workflows that mirror professional software development practices. Critically, MetaGPT requires agents to produce structured outputs such as requirements documents, design artifacts, and interface specifications, which serve as intermediate artifacts that reduce ambiguity and error propagation.

MetaGPT achieved state-of-the-art performance on code generation benchmarks, with 85.9% and 87.7% Pass@1 on HumanEval and MBPP respectively. The framework demonstrated 100% task completion rate on tested scenarios, highlighting the robustness enabled by structured workflows. The use of intermediate documents proved particularly valuable, as it allowed later-stage agents to work from well-specified requirements rather than ambiguous natural language descriptions. This structured approach substantially reduced the hallucination and inconsistency problems that plague less constrained multi-agent systems.

ChatDev [71] similarly structures agents according to the software development lifecycle phases of design, coding, testing, and documentation. The chat chain architecture decomposes each phase into atomic chat interactions between pairs of agents, enabling fine-grained control over communication while maintaining the benefits of natural language collaboration. The framework employs communicative dehallucination techniques where agents cross-check each other's outputs, reducing errors through social verification mechanisms. ChatDev demonstrated strong performance on software generation tasks while providing interpretable logs of the development process.

AutoGen [72] provides a more flexible framework for defining custom multi-agent conversations without prescribing specific organizational structures. The framework supports diverse conversation patterns including two-agent chat, sequential multi-agent chat, group discussion, and hierarchical delegation. Agents can be customized with different LLM backends, system prompts, tool access, and code execution capabilities. This flexibility has enabled applications across mathematics, coding, question answering, operations research, and entertainment domains, demonstrating the generality of the multi-agent conversation paradigm.

6.3. Simulation and Emergent Behavior

Generative Agents [21] demonstrated that LLM-based agents can exhibit believable social behaviors in simulated environments. The system populated a small village with 25 agents, each possessing a unique identity, occupation, and set of relationships. Agents were equipped with memory systems for recording and retrieving experiences, reflection capabilities for synthesizing higher-level insights, and planning modules for coordinating daily activities. Without explicit programming of social behaviors, agents spontaneously formed relationships, spread information through the community, and coordinated group activities such as parties.

The emergent behaviors observed in Generative Agents included information diffusion, where knowledge introduced to one agent propagated through the social network via conversations, and relationship formation, where agents developed friendships based on shared experiences and compatible personalities. Agents demonstrated coherent personalities over extended periods, remembering past interactions and adjusting behavior accordingly. Human evaluators rated agent behaviors as significantly more believable than ablated versions lacking memory or reflection capabilities, validating the importance of these architectural components.

AgentVerse [74] provides infrastructure for both task-solving and simulation applications of multi-agent systems. The framework supports dynamic adjustment of agent group composition through recruitment and dismissal based on task requirements. Experiments demonstrated that multi-agent groups consistently outperformed single agents across diverse tasks, with the most significant improvements on complex problems requiring diverse expertise. The emergence of social behaviors including collaboration, competition, and specialization was observed even in task-solving scenarios, suggesting that multi-agent dynamics provide benefits beyond simple task decomposition.

6.4. Debate and Collective Intelligence

Beyond cooperative problem-solving, multi-agent systems can leverage adversarial dynamics to improve solution quality. Debate frameworks pit agents against each other in argumentation, with the expectation that rigorous examination of opposing positions will surface errors and strengthen conclusions. This approach mirrors human practices such as peer review and courtroom proceedings, where adversarial processes are valued for their ability to expose weaknesses.

ProAgent [92] investigates proactive cooperation in multi-agent settings, where agents must anticipate and adapt to the behaviors of teammates without explicit communication. The framework uses LLMs to reason about teammate intentions and dynamically adjust behavior to complement observed actions. In cooperative game environments like Overcooked-AI, ProAgent achieved superior performance in both AI-AI and AI-human cooperation scenarios, with over 10% improvement over baselines. The ability to cooperate effectively with unknown partners represents an important capability for deployable multi-agent systems.

6.5. Communication Protocols and Coordination

The design of communication protocols significantly impacts multi-agent system performance [93]. Unstructured natural language communication offers flexibility but can lead to redundancy, ambiguity, and information loss. Structured protocols that define message formats and turn-taking rules can improve efficiency but may constrain the range of expressible interactions. Hybrid approaches that combine structured coordination with natural language content appear promising for balancing these concerns.

Figure 6 illustrates three fundamental communication topologies employed in multi-agent systems: sequential handoff, hierarchical delegation, and fully connected collaboration.

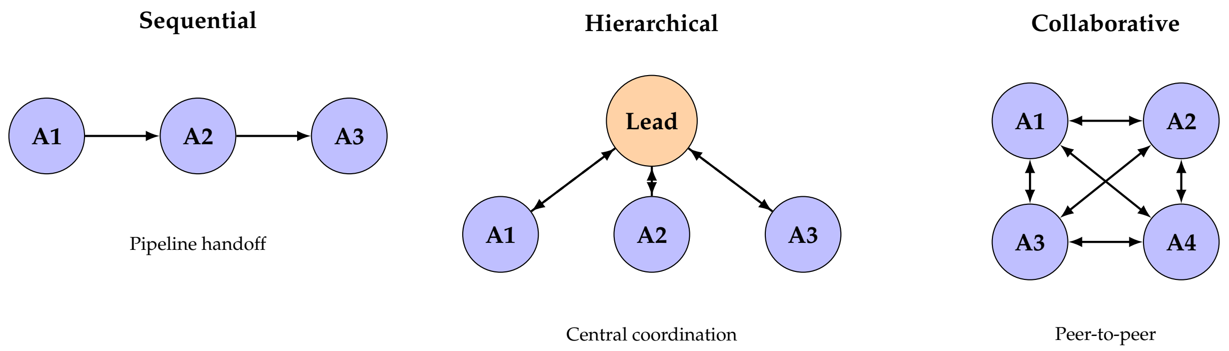

Figure 6. Multi-agent communication topologies. Sequential patterns pass information along a chain; hierarchical patterns use a leader for coordination; collaborative patterns enable peer-to-peer communication among all agents.

Memory and context management present particular challenges in multi-agent settings. Individual agents have limited context windows that may be insufficient for tracking extended multi-party conversations. Shared memory systems can provide common ground but introduce synchronization and consistency challenges. Selective attention mechanisms that filter relevant information for each agent based on role and task state represent an active area of research.

6.6. Summary

Multi-agent systems leverage the power of LLMs through coordination and collaboration, enabling capabilities that exceed those of individual agents. Table 4 summarizes key multi-agent frameworks and their characteristics. The field continues to evolve toward more sophisticated coordination mechanisms and larger-scale deployments.

Table 4. Comparison of Multi-Agent System Frameworks.

Framework	Agents	Domain	Coordination
CAMEL	2-3	General	Role-playing
MetaGPT	5+	Software	SOP workflow
ChatDev	4+	Software	Chat chain
AutoGen	2+	General	Flexible chat
Gen. Agents	25	Simulation	Autonomous
AgentVerse	Variable	General	Dynamic

7. Memory-Augmented Agents

Memory-augmented agents address the fundamental limitation of finite context windows in large language models by implementing external memory systems that persist across interactions [94–96]. While modern LLMs can process increasingly long contexts, with some models supporting over 100,000 tokens, practical constraints including computational cost and attention degradation over long sequences motivate the development of explicit memory architectures. This section examines the design principles, implementations, and applications of memory systems for LLM agents.

7.1. Context Window Limitations

The context window of a language model defines the maximum amount of text that can be processed in a single forward pass, fundamentally constraining the temporal horizon over which agents can maintain coherent state. Early transformer models were limited to 512 or 2048 tokens, making extended conversations and document analysis impractical without memory augmentation. While recent models have dramatically expanded context lengths, with Claude supporting 100K tokens and GPT-4 supporting 128K tokens, several limitations persist even at these scales.

First, attention complexity scales quadratically with sequence length in standard transformer architectures, making very long contexts computationally expensive and slow to process. Second, empirical studies have demonstrated that LLMs exhibit degraded performance on information located in the middle portions of long contexts, a phenomenon termed the “lost in the middle” effect. Third, the cost of processing long prompts through API-based services can be prohibitive for applications requiring frequent updates or many concurrent sessions. These factors motivate memory architectures that maintain essential information in compact representations rather than raw context.

7.2. Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) [7] established the foundational paradigm for combining parametric knowledge stored in model weights with non-parametric knowledge retrieved from external databases. The approach augments the model's input with documents retrieved based on query similarity, enabling access to information not present in training data without requiring model updates. RAG models demonstrated substantial improvements on knowledge-intensive NLP tasks including question answering, fact verification, and open-domain dialogue.

The RAG architecture comprises a retriever component that identifies relevant documents and a generator component that conditions on retrieved content to produce outputs. The retriever typically employs dense passage representations computed by neural encoders, with similarity measured through inner products in embedding space. Documents are retrieved at inference time based on query embeddings, and the top-k results are concatenated with the original query to form the generator input. This design enables seamless integration of dynamic knowledge bases with frozen language model generators.

Subsequent developments have refined RAG architectures along multiple dimensions. Iterative retrieval interleaves retrieval and generation steps, allowing the model to formulate follow-up queries based on initial results. Reranking stages improve retrieval precision by applying more expensive models to rank retrieved candidates. Query expansion and reformulation techniques enhance recall by generating multiple query variants. These enhancements have established RAG as a standard component in knowledge-intensive agent systems.

7.3. Hierarchical Memory Architectures

MemGPT [57] introduced a hierarchical memory architecture inspired by operating system memory management. The system distinguishes between main context (analogous to RAM) comprising the LLM's active context window, and external context (analogous to disk storage) comprising persistent databases. The agent autonomously manages data movement between tiers through function calls that read from and write to external storage, conceptually similar to page faults and cache management in traditional computing.

Figure 7 illustrates the hierarchical memory architecture employed in modern memory-augmented agents, showing the relationship between immediate context, working memory, and long-term storage.

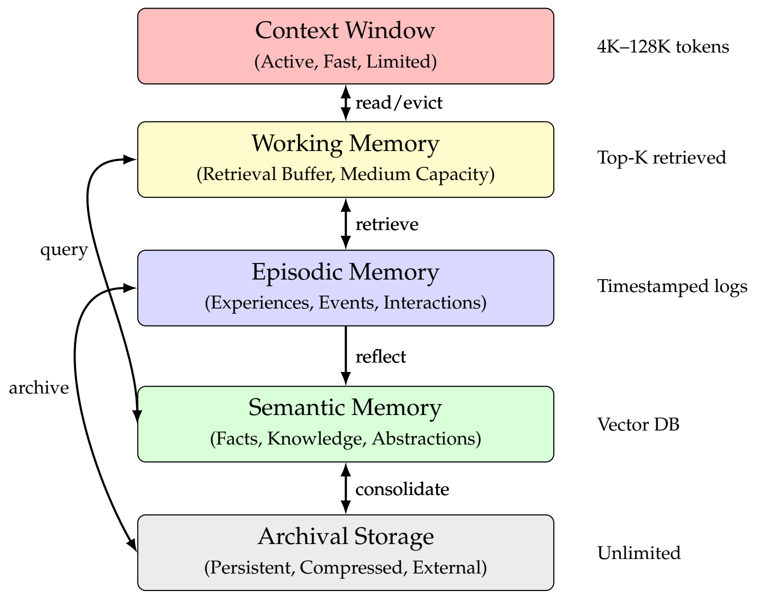

Figure 7. Hierarchical memory architecture for LLM agents. Information flows between tiers through retrieval, reflection, and consolidation operations, with capacity increasing and access speed decreasing at lower tiers.

This architecture enables effectively unbounded context while maintaining responsive interaction through the limited main context. When relevant information resides in external storage, the agent can retrieve it into main context for processing. When main context becomes full, the agent can evict less immediately relevant information to external storage while retaining retrieval capability. MemGPT was evaluated on conversational agents and document analysis tasks, demonstrating that virtual context management enables performance competitive with models having much larger native context windows.

The design leverages the function calling capabilities of modern LLMs, where the agent can invoke memory operations as tools alongside other actions. Memory operations include reading specific entries by key, searching external storage for relevant content, and writing new observations or synthesized knowledge. The agent learns when to invoke these operations through standard prompting or fine-tuning on memory management demonstrations. This approach treats memory management as an integral part of agent behavior rather than a separate preprocessing step.

7.4. Memory in Generative Agents

Generative Agents [21] implemented a sophisticated memory architecture designed to support believable social behavior over extended time horizons. The architecture comprises three interconnected components: a memory stream that records observations in natural language, a retrieval function that selects relevant memories based on multiple criteria, and a reflection process that synthesizes higher-level insights from accumulated experiences.

The memory stream maintains a comprehensive record of the agent's experiences, with each entry containing a description, creation timestamp, and most recent access timestamp. Retrieval considers three factors: recency, with more recent memories weighted higher; importance, with significant events weighted higher; and relevance, with memories semantically similar to the current context weighted higher. The retrieval score for memory m given query q is computed as:

$$\text{score}(m, q) = \alpha \cdot \text{recency}(m) + \beta \cdot \text{importance}(m) + \gamma \cdot \text{relevance}(m, q), \tag{7}$$

where recency decays exponentially with time since last access:

$$\text{recency}(m) = \exp(-\lambda \cdot (t_{\text{now}} - t_{\text{access}}(m))), \tag{8}$$

and relevance is computed as cosine similarity between embedding vectors:

$$\text{relevance}(m, q) = \frac{\mathbf{e}_m \cdot \mathbf{e}_q}{\|\mathbf{e}_m\| \|\mathbf{e}_q\|}. \tag{9}$$

The combination of these factors determines which memories surface during recall, enabling contextually appropriate behavior while preserving episodic coherence.

The reflection process addresses the challenge of extracting meaningful patterns from numerous individual memories. Periodically, agents aggregate recent memories and prompt the LLM to generate higher-level observations about patterns, relationships, and insights. These reflections are themselves stored in the memory stream and can be retrieved like primary observations, enabling hierarchical abstraction over experience. Human evaluations confirmed that agents with full memory architectures exhibited significantly more believable and coherent behavior than ablated versions lacking memory or reflection capabilities.

7.5. Episodic and Semantic Memory

Drawing on cognitive science distinctions, some memory architectures explicitly separate episodic memory capturing specific experiences from semantic memory capturing general knowledge. Episodic memories preserve contextual details including when and where events occurred, supporting recall of specific interactions and experiences. Semantic memories abstract away contextual details to capture generalizable facts and relationships, supporting reasoning about categories and typical properties.

This separation enables different retrieval strategies suited to different query types. Questions about specific past interactions benefit from episodic retrieval that considers temporal and contextual similarity. Questions about general facts or typical patterns benefit from semantic retrieval that ignores contextual details in favor of conceptual similarity. Hybrid systems can route queries to appropriate memory stores based on query characteristics or retrieve from both stores and aggregate results.

The formation of semantic memories from episodic experiences mirrors human learning processes where repeated experiences give rise to generalized knowledge. Mechanisms for semantic memory formation include frequency-based abstraction, where commonly occurring patterns are extracted as general rules, and analogy-based transfer, where similarities across episodes support formation of abstract categories. These mechanisms remain active areas of research in memory-augmented agent design.

7.6. Memory Consolidation and Forgetting

As agents accumulate experiences over time, memory systems must address storage growth and the potential for outdated information to interfere with current behavior. Memory consolidation processes compress detailed episodic memories into more compact semantic representations, reducing storage requirements while preserving essential information. Forgetting mechanisms remove or down-weight memories that are outdated, contradicted by newer information, or simply irrelevant to current objectives.

Intelligent forgetting presents significant challenges, as determining which memories can be safely discarded requires understanding their potential future relevance. Time-based decay provides a simple heuristic but may prematurely remove infrequently accessed but important memories. Access-based policies retain frequently used memories but may preserve outdated information that happens to be commonly queried. Importance-based approaches require reliable estimation of memory significance, which itself may depend on future contexts that are unknowable. These tradeoffs parallel fundamental challenges in database design and information retrieval.

7.7. Summary

Memory augmentation enables LLM agents to maintain coherent state and leverage past experience over extended time horizons. Table 5 summarizes key memory architecture approaches and their characteristics. The design of effective memory systems remains an active research area with important implications for agent capabilities.

Table 5. Comparison of Memory-Augmented Agent Architectures.

System	Structure	Retrieval	Reflection
RAG	Flat	Similarity	None
MemGPT	Hierarchical	Function call	None
Gen. Agents	Stream	Multi-factor	Periodic

8. Applications

LLM agents have been deployed across a diverse range of application domains, demonstrating their versatility and practical utility. This section examines key application areas including software engineering, scientific research, embodied AI, web automation, and enterprise applications, analyzing both the successes achieved and the challenges encountered in each domain.

8.1. Software Engineering

Software engineering represents one of the most active and consequential application domains for LLM agents [97,98]. The structured nature of code, availability of execution feedback, and clear success criteria through test suites make software tasks particularly amenable to agent-based approaches. Early work focused on code generation from natural language specifications, with models demonstrating impressive ability to synthesize functions and complete code snippets. Recent work has explored both agent-based and agentless approaches [99] to software engineering tasks.

The introduction of Devin by Cognition AI [100] in March 2024 marked a significant milestone as a claimed first fully autonomous AI software engineer. Devin operates with access to a shell, code editor, and browser within a sandboxed environment, enabling it to plan and execute complex engineering tasks from natural language specifications. On the SWE-bench benchmark [29], Devin resolved 13.86% of issues end-to-end, substantially exceeding the previous state-of-the-art of 1.96%. However, subsequent independent evaluations revealed significant limitations, with only 3 of 20 tested tasks completed satisfactorily, highlighting the gap between benchmark performance and practical reliability.

SWE-agent [101] introduced the concept of agent-computer interfaces (ACIs) specifically designed for LLM interaction with software systems. Unlike human-oriented interfaces with graphical elements and free-form navigation, ACIs provide structured commands and outputs optimized for language model consumption. This interface design philosophy substantially improved agent performance on software engineering tasks, achieving state-of-the-art results on SWE-bench when combined with capable base models. The work demonstrated that thoughtful interface design can be as important as model capability for agent effectiveness.

OpenHands [102], evolved from the OpenDevin project, provides an open-source platform for AI software developers as generalist agents. The platform supports diverse software engineering workflows including bug fixing, feature implementation, and code review, with evaluation across 15 benchmarks spanning different aspects of software development. The open-source nature has enabled rapid community contributions and research into agent architectures, tool designs, and evaluation methodologies.

8.2. Scientific Research

LLM agents are increasingly being applied to accelerate scientific research across domains including chemistry, biology, materials science, and mathematics. These applications leverage agents' ability to process scientific literature, design experiments, analyze data, and generate hypotheses. The combination of domain knowledge in training data with tool access to computational resources and databases enables agents to contribute meaningfully to research workflows.

In chemistry, agents have been developed to assist with synthesis planning, property prediction, and laboratory automation. Systems that combine language models with specialized chemistry tools such as molecular simulators and reaction databases can propose synthetic routes for target compounds and predict reaction outcomes. The ability to reason about molecular structures in natural language while accessing quantitative computational chemistry tools represents a powerful combination for accelerating discovery.

Mathematical reasoning represents another significant application area, building on the strong performance of LLMs on mathematical benchmarks. Agents that combine natural language reasoning with formal proof assistants can tackle problems that neither approach handles well alone. The natural language component enables intuitive problem understanding and strategy formulation, while formal tools provide rigorous verification and search capabilities. This combination has achieved notable results on competition mathematics and theorem proving tasks.

8.3. Embodied AI and Robotics

Embodied AI extends LLM agents into physical environments where they must perceive the world through sensors and act through physical actuators. This domain presents unique challenges including the need for real-time responsiveness, handling of continuous state spaces, and ensuring safe behavior in physical environments. Despite these challenges, LLM-based approaches have demonstrated promising results in robot manipulation, navigation, and human-robot interaction.

Voyager [103] demonstrated the potential for LLM agents in virtual embodied environments by creating an agent that continuously explores, learns, and accomplishes diverse goals in Minecraft. The system comprises an automatic curriculum for exploration, a skill library for storing and retrieving learned behaviors, and an iterative prompting mechanism that incorporates environmental feedback. Voyager obtained 3.3x more unique items and unlocked technology milestones up to 15.3x faster than baselines, while demonstrating generalization to new Minecraft worlds with novel task specifications.

Inner Monologue [104] investigated the use of language models for embodied reasoning in robotic settings. Recent surveys on robot intelligence with LLMs [105] and embodied multi-modal agents [106] have further advanced this area. The approach enables robots to form internal monologues that interpret observations, plan actions, and adapt to feedback through natural language. By grounding language generation in physical observations and action outcomes, robots can exhibit more robust and adaptive behavior than purely feedforward approaches. The system demonstrated capabilities including multilingual instruction following and interactive problem solving when faced with obstacles.

The HuggingGPT system [87] demonstrates how LLM orchestration can enable complex multimodal tasks that combine perception, reasoning, and generation. By routing subtasks to appropriate specialized models from the Hugging Face ecosystem, the system can handle requests spanning image understanding, speech processing, and video analysis. This orchestration paradigm provides a path toward general-purpose embodied agents that leverage diverse perceptual and actuation capabilities.

8.4. Web Automation

Web automation applies LLM agents to navigate websites, fill forms, extract information, and complete transactions on behalf of users. The web represents a natural environment for agent deployment due to its structured nature, well-defined action space of clicks and typing, and immediate feedback through page updates. However, the complexity and diversity of real websites present significant challenges for robust automation.

Mind2Web [88] established the first large-scale benchmark for generalist web agents, revealing substantial gaps between current capabilities and practical requirements. Subsequent work including WebVoyager [107], AutoWebGLM [108], and WebAgent [109] has advanced web navigation capabilities through multimodal understanding and reinforcement learning. The benchmark comprises tasks across 137 websites in 31 domains, with evaluation on element selection accuracy and task completion. Best methods achieved approximately 50% element selection accuracy compared to human levels above 90%, indicating significant room for improvement. The benchmark has catalyzed research into more capable web understanding and interaction.

WebArena [31] advances evaluation by providing reproducible self-hosted web environments representing realistic domains including e-commerce, forums, and content management. The 812 included tasks emulate authentic human web activities with functional correctness evaluation. GPT-4-based agents achieved only 14.41% task success compared to 78.24% human performance, demonstrating that current agents struggle with the full complexity of web interaction. The controlled environment enables systematic analysis of failure modes and iterative improvement.

OSWorld [110] extends agent evaluation to general computer use across operating systems including Ubuntu, Windows, and macOS. The benchmark comprises 369 tasks involving real applications, file operations, and multi-application workflows. State-of-the-art models achieved only 12.24% success compared to 72.36% human performance, with primary challenges including GUI element identification and operational knowledge. This benchmark highlights the substantial gap remaining before agents can serve as general computer assistants.

Table 6 provides a comprehensive comparison of LLM agent applications across different domains, summarizing key characteristics, representative systems, and current performance levels.

Table 6. Comparison of LLM Agent Application Domains.

Domain	**Key Tasks**	**Representative Systems**	**Best Perf.**	**Human Perf.**	**Main Challenges**
Software Eng.	Bug fixing, coding	Devin, SWE-agent	22%	85%+	Long-horizon, context
Web Automation	Navigation, forms	WebArena agents	14.4%	78.2%	Dynamic content, UI
OS Interaction	File ops, apps	OSWorld agents	12.2%	72.4%	GUI understanding
Scientific	Experiments, analysis	ChemCrow	Varies	–	Domain knowledge
Embodied AI	Navigation, manip.	Voyager, PaLM-E	3.3x better	–	Real-time, safety
Enterprise	QA, document proc.	Custom agents	–	–	Accuracy, security

8.5. Enterprise and Business Applications

Enterprise applications represent a significant opportunity for LLM agent deployment, with potential to automate knowledge work, enhance decision-making, and improve customer service. Customer service chatbots leveraging LLM capabilities can handle a broader range of queries with more natural conversation than traditional rule-based systems. Document processing agents can extract, summarize, and route information from diverse business documents.

Data analysis and business intelligence represent particularly promising application areas. Agents that combine natural language understanding with database querying and visualization tools can enable non-technical users to explore data and generate insights. The ability to translate natural language questions into SQL queries, execute them against databases, and present results in interpretable formats democratizes access to data-driven decision making.

However, enterprise deployment raises significant concerns around accuracy, security, and governance. Hallucination risks are particularly consequential in business contexts where incorrect information can lead to financial or reputational damage. Integration with sensitive data systems requires robust access controls and audit capabilities. Regulatory compliance in domains such as finance and healthcare imposes additional requirements on agent behavior and documentation.

8.6. Timeline of Major Developments

Figure 8 illustrates the evolution of key LLM agent developments from 2022 to 2024. The timeline shows the rapid acceleration of progress following the introduction of ChatGPT and GPT-4, with foundational paradigms established in 2023 and increasingly sophisticated systems emerging in 2024.

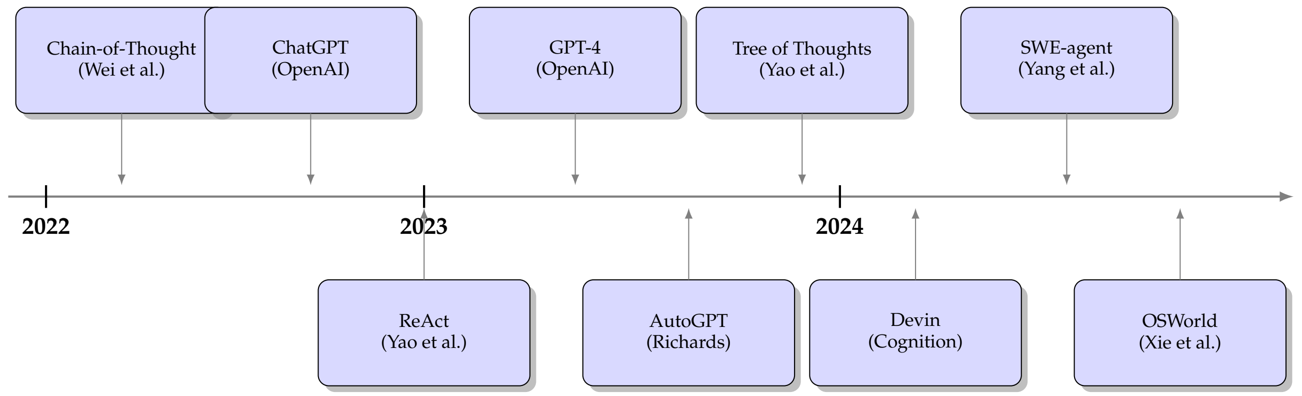

Figure 8. Timeline of major developments in LLM agents. Key milestones are shown from the introduction of chain-of-thought prompting through recent advances in software engineering and computer use agents.

9. Benchmarks and Evaluation

The systematic evaluation of LLM agents requires benchmarks that capture the multi-faceted nature of agent capabilities including reasoning, tool use, planning, and interaction with complex environments. This section surveys the major benchmarks developed for agent evaluation, presents comparative analyses of agent performance, and discusses evaluation methodologies and their limitations.

9.1. General Agent Benchmarks

AgentBench [32] provides the first comprehensive benchmark designed to evaluate LLMs as autonomous agents across diverse environments. Recent work has introduced TheAgentCompany [111] for evaluating agents on consequential real-world tasks, while comprehensive surveys on agent evaluation [112,113] have systematized evaluation methodologies. The benchmark comprises eight distinct environments spanning operating systems, databases, knowledge graphs, digital card games, lateral thinking puzzles, house-holding simulation, web browsing, and web shopping. Each environment requires different combinations of skills including command generation, query formulation, strategic planning, and interactive navigation.

Table 7 presents the performance of representative models on AgentBench. The results reveal substantial performance gaps between commercial and open-source models, with GPT-4 achieving 4.82 overall score compared to 0.56 for Llama2-70B-Chat. The benchmark also exposes uneven performance across environments, where models that excel at database queries may struggle with operating system commands, highlighting the challenge of developing truly generalist agents.

Table 7. Performance Comparison on AgentBench (Overall Score).

Model	OS	DB	Web	Overall
GPT-4	35.1	32.0	2.5	4.82
GPT-3.5-turbo	25.6	24.5	1.2	3.49
Claude-2	21.8	18.6	1.0	2.78
Llama2-70B	12.4	8.2	0.3	0.56

The InterCode benchmark [114] focuses on interactive coding with execution feedback, providing environments for Bash, SQL, and Python programming. Unlike static code generation benchmarks, InterCode evaluates agents on their ability to iteratively refine solutions based on execution errors and test results. This interactive paradigm more closely mirrors real software development where developers incrementally debug and improve their code based on feedback.

9.2. Software Engineering Benchmarks

SWE-bench [29] has emerged as the primary benchmark for evaluating agents on real-world software engineering tasks. The benchmark comprises 2,294 issues from 12 popular Python repositories including Django, Flask, and scikit-learn, with each task requiring agents to generate patches that resolve described problems. Evaluation is based on whether generated patches pass the associated test suites, providing objective measures of functional correctness.

Table 8 summarizes agent performance on SWE-bench. Early systems achieved very low success rates, with Claude 2 resolving only 1.96% of issues. The introduction of specialized agent architectures substantially improved performance, with Devin achieving 13.86% and subsequent systems exceeding 20% on the full benchmark. SWE-bench Verified, a human-validated subset of 500 problems, has seen even higher success rates exceeding 50% for the best systems as of late 2024.

Table 8. Agent Performance on SWE-bench.

Agent	SWE-bench Full (%)	SWE-bench Verified (%)
Claude 2 (baseline)	1.96	–
Devin	13.86	–
SWE-agent + GPT-4	12.47	23.8
SWE-agent + Claude 3.5	18.0	33.6
OpenHands + Claude 3.5	22.0	41.0

Despite impressive progress, the gap between current agents and human performance remains substantial. Human developers can resolve the majority of SWE-bench issues given sufficient time, while even the best agents fail on most problems. Analysis of failure modes reveals that agents struggle with tasks requiring deep understanding of code context, multi-file changes, and complex debugging workflows that humans handle through experience and intuition.

9.3. Web Automation Benchmarks

Mind2Web [88] provides a large-scale benchmark for generalist web agents comprising over 2,000 tasks across 137 real websites in 31 domains. Tasks include form filling, navigation, information extraction, and transaction completion, requiring agents to interpret natural language instructions and execute appropriate sequences of clicks and inputs. Evaluation assesses both element selection accuracy and overall task completion.

WebArena [31] advances web agent evaluation by providing reproducible self-hosted environments representing e-commerce, forums, software development, and content management domains. The 812 included tasks evaluate functional correctness rather than surface-level metrics, requiring agents to actually accomplish specified objectives in working web environments. Table 9 presents agent performance on WebArena.

Table 9. Agent Performance on WebArena.

Agent	Task Success Rate (%)
Human	78.24
GPT-4 + CoT	14.41
GPT-4 + SoM	11.08
GPT-3.5-turbo	7.20

The substantial gap between human performance (78.24%) and the best agents (14.41%) on WebArena highlights fundamental challenges in web automation. Agents frequently struggle with dynamic content, multi-step workflows, and ambiguous instructions. The controlled environment enables detailed analysis of failure modes, revealing that perceptual errors in identifying interactive elements and planning errors in action sequencing are primary contributors to failures.

Figure 9 provides a visual comparison of agent performance across major benchmarks, illustrating the substantial gap between current agent capabilities and human-level performance.

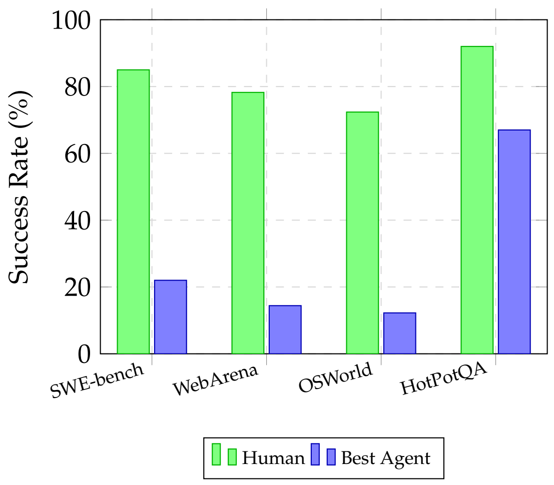

Figure 9. Performance comparison between human baselines and best-performing agents across major benchmarks. Significant gaps persist across all domains.

OSWorld [110] extends evaluation to general computer use across operating systems including Ubuntu, Windows, and macOS. The benchmark comprises 369 tasks involving real applications, file operations, and multi-application workflows that reflect authentic computer use scenarios. Performance remains low across all evaluated systems, with the best achieving only 12.24% success compared to 72.36% human performance. Primary challenges include GUI element identification and operational knowledge about application functionality.

9.4. Reasoning and Knowledge Benchmarks

While not specifically designed for agents, several reasoning benchmarks have been widely adopted for evaluating agent capabilities. HotPotQA [83] evaluates multi-hop question answering requiring reasoning across multiple documents, a capability essential for agents that must gather and synthesize information from diverse sources. ReAct and related methods have achieved strong results on HotPotQA by interleaving reasoning with information retrieval actions.

Mathematical reasoning benchmarks including GSM8K and MATH evaluate agents' ability to solve quantitative problems requiring multi-step calculation and logical deduction. Chain-of-thought and program-aided approaches have achieved impressive results on these benchmarks, with some configurations exceeding 90% accuracy on GSM8K. However, performance on more challenging mathematical reasoning remains limited, and the relationship between benchmark performance and practical mathematical assistance capabilities remains unclear.

Table 10 provides comprehensive statistics for major agent benchmarks, including dataset sizes, task characteristics, and evaluation metrics.

Table 10. Statistics of Major Agent Evaluation Benchmarks.

Benchmark	Tasks	Domains	Metric	Environment	Year
AgentBench	8 env.	OS/DB/Web/Game	Weighted score	Simulated	2023
SWE-bench	2,294	Software repos	Pass@1	Real codebases	2024
SWE-bench Verified	500	Software repos	Pass@1	Real codebases	2024
WebArena	812	4 web domains	Task success	Self-hosted	2024
Mind2Web	2,350	137 websites	Element acc.	Real websites	2023
OSWorld	369	3 OS platforms	Task success	Virtual machines	2024
HotPotQA	113K	Wikipedia	EM/F1	Text retrieval	2018
GSM8K	8.5K	Math word	Accuracy	Calculator	2021
ToolBench	16,464	16K APIs	Pass rate	API simulation	2023

9.5. Multimodal and Embodied Benchmarks

Recent benchmarks have extended agent evaluation to multimodal settings requiring integration of visual perception with language understanding. VisualWebArena builds on WebArena by incorporating visual elements into web tasks, evaluating whether agents can understand screenshots and visual layouts in addition to HTML structure. Performance drops substantially when visual understanding is required, revealing gaps in current multimodal agent capabilities.

Embodied benchmarks evaluate agents in simulated physical environments requiring navigation, manipulation, and object interaction. While primarily designed for reinforcement learning research, these benchmarks have been adapted for LLM agent evaluation. Results demonstrate that LLM agents can provide high-level planning and reasoning for embodied tasks, but integration with low-level motor control remains challenging.

Figure 10 presents a radar chart comparing the capabilities of different foundation models across key agent-relevant dimensions, illustrating the varying strengths and weaknesses that inform model selection for specific applications.

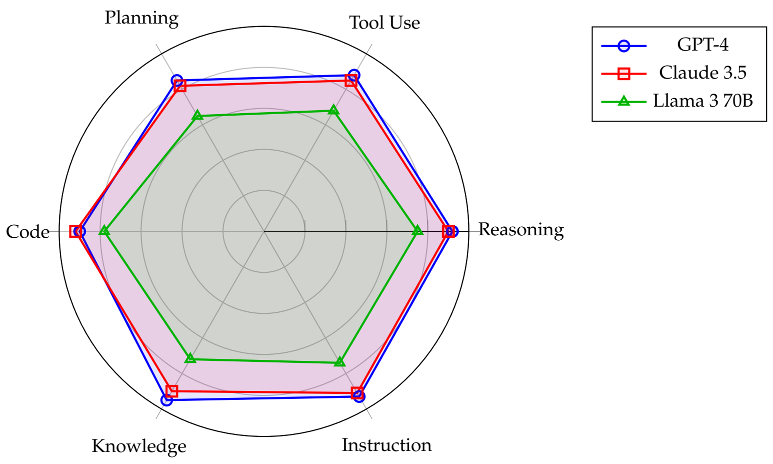

Figure 10. Radar chart comparing foundation model capabilities across agent-relevant dimensions. Scores are normalized percentages based on benchmark performance.

9.6. Evaluation Methodology Considerations

Several methodological considerations affect the interpretation of benchmark results. Many benchmarks were designed for evaluating models in isolation and may not fully capture the interactive, multi-turn nature of agent behavior. The choice of prompting strategy, tool configurations, and hyperparameters can substantially affect performance, making fair comparisons difficult. Contamination concerns arise when evaluation data may have appeared in model training sets, potentially inflating performance estimates.

The development of robust evaluation methodologies for LLM agents remains an active research area. Proposals include multi-dimensional evaluation frameworks that assess different aspects of agent competence independently, behavioral testing that examines agent responses to diverse scenarios beyond task completion, and human evaluation protocols that capture subjective assessments of agent usefulness and reliability. The relationship between cognitive capabilities of LLMs and their practical utility as agents has been explored in recent work examining similarities between language model reasoning and human cognition . Recent surveys on multimodal LLMs and contrastive learning provide complementary perspectives on the capabilities underlying agent performance. The role of parallel computing in enabling efficient agent inference has also been examined .

10. Challenges and Limitations

Despite remarkable progress, LLM agents face substantial challenges that limit their practical deployment and reliability. This section examines the key challenges spanning hallucination and factual accuracy, planning and reasoning limitations, safety and security concerns, and efficiency and scalability constraints.

10.1. Hallucination and Factual Accuracy

Hallucination, the generation of plausible but factually incorrect or unsupported content, represents a fundamental challenge for LLM agents. Unlike static generation where hallucinations may be merely embarrassing, agent hallucinations can lead to incorrect actions with real-world consequences. An agent that hallucinates an API endpoint may attempt calls that fail or, worse, succeed with unintended effects. An agent that fabricates information during decision-making may pursue suboptimal or harmful courses of action.

The causes of hallucination in LLMs are multifaceted and not fully understood [115,116]. Training on internet text inevitably exposes models to misinformation, which may be reproduced during generation. The pressure to produce fluent, confident outputs can lead models to fabricate details rather than acknowledge uncertainty. The parametric nature of model knowledge means that information is compressed and abstracted in ways that may introduce distortions. Recent work on semantic entropy [117] provides principled approaches to detecting confabulations. Current approaches to hallucination mitigation include retrieval augmentation to ground generation in verified sources [7], self-consistency checks to identify unreliable outputs [62], and external verification through tools like CRITIC [85].

Recent work has examined the relationship between LLM capabilities and potential for misuse, including concerns about adversarial attacks, prompt injection, and jailbreaking that can compromise agent behavior [118]. The security of LLM agents against malicious manipulation represents an ongoing research challenge, particularly as agents are granted broader capabilities for interacting with external systems.

10.2. Long-Horizon Planning

LLM agents exhibit significant limitations in long-horizon planning, where goals must be decomposed into extended sequences of subgoals and actions. While agents can successfully complete tasks requiring a handful of steps, performance degrades substantially as task complexity increases. Several factors contribute to this limitation.

Error accumulation presents a fundamental challenge where small errors in early steps compound through subsequent decisions. An incorrect tool selection or misinterpreted observation early in a task can lead the agent down paths from which recovery is difficult. The probability of task completion can decrease exponentially with the number of required steps if each step has any probability of failure.

Context limitations constrain the amount of information agents can maintain about task state, history, and goals. While memory augmentation addresses some aspects of this challenge, current memory systems remain imperfect in their retention and retrieval of relevant information. Long tasks may require more contextual information than can be effectively managed, leading to forgotten goals or repeated actions.

Credit assignment becomes increasingly difficult in long-horizon settings. When a task ultimately fails, identifying which of many preceding decisions was responsible is challenging for both human evaluators and potential learning mechanisms. This difficulty impedes both human debugging of agent behavior and automated approaches to improving agent policies.

10.3. Generalization and Robustness

Current agents exhibit limited generalization across tasks, domains, and environments. An agent that performs well on one set of benchmarks may fail on superficially similar tasks that differ in subtle ways. This brittleness limits the practical utility of agents, as users cannot reliably predict which tasks will be handled successfully.

Distribution shift between training/development conditions and deployment scenarios poses particular challenges. Agents evaluated on curated benchmarks encounter more diverse and potentially adversarial inputs in real-world deployment. The gap between controlled evaluation and open-ended use remains substantial for current systems.

Robustness to input variations is often limited. Small changes to task phrasing, observation formatting, or environmental conditions can lead to dramatically different agent behavior. This sensitivity makes agent performance unpredictable and limits user trust. Recent surveys on multimodal LLMs have examined how visual and language understanding capabilities interact, with implications for robust agent design [119].

10.4. Safety and Alignment

Deploying autonomous agents raises significant safety concerns that extend beyond those of passive language models [120,121]. Agents that can take actions in the world may cause harm through incorrect, unintended, or misaligned behavior. Recent work has examined vulnerabilities in safety alignment [122]. Ensuring that agents pursue intended goals without negative side effects is a challenge of increasing importance as agent capabilities expand.

Goal misspecification occurs when the objectives provided to an agent do not fully capture the user's true intentions. An agent that optimizes a proxy metric may achieve high scores while failing to satisfy the underlying human need. Sandbagging, where agents may underperform on evaluations while preserving capabilities for other purposes, represents an emerging concern.

Unintended side effects arise when agents pursue goals through means that have negative consequences not explicitly prohibited. An agent tasked with cleaning up disk space might delete important files if not carefully constrained. Defining comprehensive constraints that prevent all harmful actions while preserving useful capabilities is extremely challenging.

The development of agent systems that are transparent, interpretable, and aligned with human values has been addressed through various approaches including constitutional AI [52] and instruction tuning . Explainability remains an important concern, as users need to understand agent reasoning to appropriately calibrate trust . The broader ethical implications of deploying autonomous AI agents have received increasing attention [123].

10.5. Efficiency and Scalability

Practical deployment of LLM agents faces significant efficiency and scalability constraints. The computational cost of LLM inference, particularly for large frontier models, limits the number of interaction steps that can be economically supported. Complex tasks requiring hundreds of model calls may incur substantial costs, making agents impractical for many applications.

Latency presents challenges for interactive applications where users expect responsive feedback. Multi-step agent processes that require sequential model calls introduce delays that degrade user experience. Parallel processing of independent subproblems can help but is not always possible given task dependencies.

Memory and context management become bottlenecks as task complexity scales. Maintaining comprehensive records of observations, actions, and reasoning traces consumes limited context capacity. External memory systems introduce additional overhead for storage and retrieval. The optimization of memory architectures for different use cases represents an active area of investigation .

The computational demands of reasoning approaches vary substantially, with more sophisticated techniques like tree-of-thought search incurring multiplicative cost increases over basic prompting . Balancing reasoning quality against computational efficiency requires careful consideration of application requirements.

10.6. Reproducibility and Evaluation

The evaluation of LLM agents faces reproducibility challenges stemming from multiple sources. Commercial API-based models may change behavior over time as providers update their systems, making longitudinal comparisons difficult. Stochastic generation means that repeated runs can produce different outcomes, requiring statistical analysis that is often lacking in agent evaluations.

Benchmark contamination concerns arise because large-scale pretraining corpora may include evaluation data, artificially inflating performance estimates. Determining whether specific evaluation examples appeared in training data is often impossible given the opacity of training processes for commercial models.

The gap between benchmark performance and real-world utility remains poorly understood. High scores on curated evaluations do not necessarily translate to practical value for users, and metrics that better predict real-world usefulness are needed. The development of more comprehensive evaluation frameworks that assess multiple dimensions of agent competence represents an important research direction. Recent work on uncertainty quantification [124] and trustworthy AI provides foundations for more robust evaluation methodologies.

Table 11 summarizes the major challenges facing LLM agents along with current mitigation approaches and remaining open problems.

Table 11. Summary of Major Challenges in LLM Agent Development.

Challenge	Description	Current Approaches	Open Problems
Hallucination	Generation of factually incorrect content	RAG, self-consistency, external verification	Reliable detection, root cause
Long-horizon planning	Performance degrades with task length	Hierarchical planning, subgoal decomposition	Error accumulation, credit assignment
Generalization	Limited transfer across domains	Multi-task training, in-context learning	Distribution shift, robustness
Safety	Unintended harmful behaviors	Constitutional AI, RLHF, guardrails	Goal misspecification, side effects
Efficiency	High computational costs	Model distillation, caching, pruning	Real-time interaction, cost reduction
Reproducibility	Inconsistent evaluation results	Standardized benchmarks, controlled environments	API changes, contamination

11. Future Directions

The field of LLM agents is rapidly evolving, with numerous promising research directions poised to address current limitations and unlock new capabilities. This section outlines key areas for future investigation including architectural innovations, training paradigms, human-agent collaboration, and emerging application domains.

11.1. Advanced Reasoning Architectures

Future agent architectures will likely incorporate more sophisticated reasoning mechanisms that go beyond current chain-of-thought and tree-search approaches. Neuro-symbolic integration offers a promising direction, combining the flexibility of neural language models with the precision and verifiability of symbolic reasoning systems. Hybrid architectures that use LLMs for high-level planning while delegating formal reasoning to specialized engines could achieve the best of both paradigms.

World models that enable agents to simulate the consequences of actions before executing them represent another frontier. By maintaining predictive models of environment dynamics, agents could evaluate potential action sequences through mental simulation, reducing costly interactions with real environments. The development of accurate world models for complex domains such as software systems and web interfaces remains challenging but would substantially enhance agent planning capabilities.

Hierarchical reasoning architectures that operate at multiple levels of abstraction could address limitations in long-horizon planning. High-level modules could decompose goals into subgoals while lower-level modules focus on detailed execution, with bidirectional communication enabling adaptation to unexpected situations. Such architectures mirror human cognitive organization and could enable more robust handling of complex tasks.

The transfer and optimization of reasoning capabilities across models has received initial attention , with potential for developing more efficient agents that leverage reasoning patterns learned from larger teacher models. Mixture-of-experts architectures may enable more efficient agent systems that activate specialized capabilities as needed. The automation of research processes through agent-based systems represents another promising direction that could accelerate scientific discovery.

11.2. Agent Training and Learning

Current agents primarily rely on prompting and few-shot learning, with limited use of training specifically for agent behavior. Future work will likely develop more sophisticated training paradigms that directly optimize agent performance on downstream tasks. Reinforcement learning from human feedback (RLHF) has proven effective for general instruction following [125,126], with recent work exploring RLAIF as a scalable alternative [127]. Extensions specifically targeting agent capabilities remain underexplored.

Learning from interaction experience represents a natural extension of current approaches. Agents that can improve through autonomous exploration, learning from both successes and failures without continuous human supervision, would be substantially more practical for deployment. Key challenges include credit assignment over long horizons, sample efficiency in sparse reward settings, and safe exploration that avoids harmful actions during learning.

Self-improvement mechanisms that enable agents to identify and address their own weaknesses could lead to continuously improving systems. An agent that monitors its own performance, identifies recurring failure patterns, and develops strategies to address them would exhibit a form of meta-learning valuable for general-purpose deployment. Initial work on self-reflection and self-refinement provides foundations for such capabilities [63,84].

Fine-tuning methodologies for adapting large models to agent tasks efficiently continue to advance . Low-rank adaptation and related parameter-efficient approaches may enable broader deployment of specialized agent capabilities without the computational costs of full model training.

11.3. Human-Agent Collaboration

The future of LLM agents likely involves deep integration with human workflows rather than full automation. Designing effective human-agent collaboration requires understanding how to partition tasks between human and machine contributors, how to communicate agent uncertainty and limitations, and how to enable human oversight without introducing prohibitive overhead.

Adaptive autonomy systems that adjust the level of agent independence based on task difficulty, user preference, and confidence estimates could optimize the tradeoff between efficiency and human control. Agents might operate fully autonomously on routine tasks while seeking human input for novel or high-stakes decisions. Developing robust confidence estimates that accurately predict when human involvement would be beneficial remains an open challenge.

Explanation and transparency capabilities will be essential for human-agent collaboration. Users need to understand agent reasoning to provide effective guidance and appropriately calibrate trust. Current agents often operate as black boxes, with reasoning traces providing limited insight into actual decision processes. Improving the interpretability and faithfulness of agent explanations is a priority for human-centered design .

Personalization represents another dimension of human-agent collaboration, where agents adapt to individual user preferences, expertise levels, and working styles. Personalized agents could learn communication patterns that particular users find helpful, adjust their level of initiative based on user preferences, and remember user-specific context across sessions.

11.4. Multimodal and Embodied Agents

The extension of LLM agents to multimodal settings incorporating vision, audio, and potentially other sensory modalities opens vast new application spaces. Multimodal agents that can understand documents with complex layouts, interpret visual interfaces, and process multimedia content would be substantially more capable than text-only systems. Recent advances in multimodal large language models provide foundations for such capabilities . The development of affective agents that can perceive and respond to emotional cues represents an emerging direction with applications in customer service and marketing .

Embodied agents that operate in physical environments through robotic platforms represent a longer-term frontier. While current robotic applications of LLMs focus on high-level planning, tighter integration with perception and control systems could enable more capable robotic assistants. Challenges include real-time responsiveness requirements, handling of physical uncertainty, and ensuring safe operation in human environments.

Cross-modal reasoning that leverages information from multiple modalities to inform decisions will become increasingly important. An agent that can read a technical manual (text), observe a demonstration (video), and then perform a task (embodied action) would exhibit a form of multimodal understanding currently beyond reach. Developing unified representations that support such cross-modal transfer is an active research direction.

11.5. Emerging Application Domains

Several application domains are poised for significant expansion of agent capabilities. Scientific research assistance, where agents help with literature review, hypothesis generation, experimental design, and data analysis, could substantially accelerate discovery processes . The depth of domain knowledge required for effective scientific agents presents challenges but also opportunities for developing specialized systems with expert-level capabilities.

Healthcare applications including clinical decision support, patient monitoring, and medical documentation could benefit from agent capabilities, though the high stakes of medical decisions impose stringent requirements for accuracy, reliability, and safety. The relationship between AI capabilities and clinical utility requires careful evaluation .

Education represents a domain where personalized agent tutors could provide individualized instruction at scale. Agents that adapt to student knowledge states, identify misconceptions, and provide targeted feedback could democratize access to high-quality instruction. Challenges include maintaining student engagement, accurately assessing understanding, and appropriately scaffolding learning progressions.

11.6. Safety and Governance

As agent capabilities expand, safety and governance considerations become increasingly urgent. The development of agent systems that reliably pursue intended goals without harmful side effects remains a fundamental challenge. Formal verification approaches that provide guarantees about agent behavior in specified circumstances could complement empirical testing and human oversight.

Governance frameworks for agent deployment must address questions of liability, accountability, and appropriate use. When an agent takes an action that causes harm, determining responsibility among developers, deployers, and users requires new legal and ethical frameworks. Proactive engagement with policymakers and the public will be essential for navigating these challenges.

Red-teaming and adversarial testing will become increasingly important for identifying agent vulnerabilities before deployment. Systematic approaches to discovering failure modes, security vulnerabilities, and potential for misuse can inform both system design and deployment decisions. The development of robust agent systems that maintain intended behavior under adversarial conditions represents a critical research priority .

Table 12 summarizes the key future research directions, their current status, and potential impact on agent capabilities.

Table 12. Summary of Future Research Directions for LLM Agents.

Direction	Key Approaches	Current Status	Timeline	Expected Impact
Neuro-symbolic	Hybrid LLM + formal reasoning	Early research	Medium-term	Verifiable reasoning
World Models	Predictive simulation	Emerging	Medium-term	Better planning
Agent Training	RL from interaction	Limited	Short-term	Improved efficiency
Human-Agent Collab.	Adaptive autonomy	Active research	Short-term	Practical deployment
Multimodal Agents	Vision + language + action	Rapid progress	Short-term	Broader applications
Safety & Alignment	Formal verification, red-teaming	Critical priority	Ongoing	Reliable deployment

12. Conclusion

This survey has provided a comprehensive examination of large language model agents, a rapidly evolving field that combines the remarkable capabilities of foundation models with autonomous action-taking in pursuit of complex goals. We have presented a novel taxonomy organizing the diverse landscape into four complementary dimensions: reasoning-enhanced agents that leverage structured prompting and deliberative search; tool-augmented agents that extend LLM capabilities through external resources; multi-agent systems that enable collaborative problem-solving; and memory-augmented agents that maintain context across extended interactions.

Our analysis of over 100 papers spanning 2020-2024 reveals both the impressive progress achieved and the substantial challenges that remain. On reasoning, we have traced the evolution from basic chain-of-thought prompting through increasingly sophisticated tree and graph-based deliberation, culminating in paradigms like ReAct and Reflexion that tightly couple reasoning with action and learning from experience. On tool use, we have examined how agents have progressed from simple API calling to orchestrating thousands of real-world services and executing arbitrary code. Multi-agent systems have demonstrated that collaboration among LLM instances can yield capabilities exceeding those of individual agents, while memory architectures have begun addressing the fundamental limitation of finite context windows.

The application domains for LLM agents continue to expand, with particularly notable progress in software engineering where agents can now resolve a meaningful fraction of real-world issues automatically. Yet benchmark evaluations consistently reveal substantial gaps between agent and human performance, whether on software tasks (SWE-bench), web automation (WebArena), or general computer use (OSWorld). These gaps underscore the challenges of hallucination, long-horizon planning, robustness, and safety that must be addressed for agents to achieve practical reliability.

Looking forward, we identify several promising research directions. Advanced reasoning architectures incorporating neuro-symbolic integration and world models could address current planning limitations. Training paradigms specifically designed for agent behavior, rather than general language modeling, could yield more capable and reliable systems. Deep integration of human-agent collaboration, with adaptive autonomy and transparent explanation, will be essential for real-world deployment. The extension to multimodal and embodied settings opens vast new application spaces while introducing additional challenges.

The development of LLM agents sits at the intersection of multiple AI research traditions, drawing on insights from natural language processing, reinforcement learning, robotics, and cognitive science. As the field matures, we anticipate increased integration of these perspectives and the emergence of unified frameworks that transcend current categorical distinctions. The ultimate goal of creating AI systems that can perceive, reason, plan, and act to accomplish human-specified objectives remains challenging but increasingly within reach.

We hope this survey provides a useful resource for researchers and practitioners entering the field, establishing a shared understanding of foundational concepts, key methods, and open challenges. The pace of progress in LLM agents is remarkable, and we anticipate that many of the limitations discussed here will be addressed in the coming years. At the same time, the challenges of safety, alignment, and beneficial deployment will only become more pressing as capabilities expand. Addressing these challenges thoughtfully and proactively is essential for realizing the potential of LLM agents to benefit humanity.

References

1. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is All You Need **2017**. *30*.
2. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2019, pp. 4171–4186.
3. OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774* **2023**.
4. Xin, Y.; Luo, S.; Liu, X.; Zhou, H.; Cheng, X.; Lee, C.E.; Du, J.; Wang, H.; Chen, M.; Liu, T.; et al. V-petl bench: A unified visual parameter-efficient transfer learning benchmark. *Advances in neural information processing systems* **2024**, *37*, 80522–80535.
5. Yu, Z. AI for science: A comprehensive review on innovations, challenges, and future directions. *International Journal of Artificial Intelligence for Science (IJAI4S)* **2025**, *1*.
6. Yu, Z.; Idris, M.Y.I.; Wang, P.; Qureshi, R. CoTextor: Training-free modular multilingual text editing via layered disentanglement and depth-aware fusion. In Proceedings of the NeurIPS Creative AI Track, 2025.
7. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.t.; Rocktäschel, T.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Proceedings of the Advances in Neural Information Processing Systems, 2020, Vol. 33, pp. 9459–9474.
8. OpenAI. Introducing ChatGPT. https://openai.com/blog/chatgpt, 2022.
9. Xin, Y.; Du, J.; Wang, Q.; Yan, K.; Ding, S. Mmap: Multi-modal alignment prompt for cross-domain multi-task learning. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2024, Vol. 38, pp. 16076–16084.
10. Wang, H.; Zhang, X.; Xia, Y.; Wu, X. An intelligent blockchain-based access control framework with federated learning for genome-wide association studies. *Computer Standards & Interfaces* **2023**, *84*, 103694.
11. Xin, Y.; Qin, Q.; Luo, S.; Zhu, K.; Yan, J.; Tai, Y.; Lei, J.; Cao, Y.; Wang, K.; Wang, Y.; et al. Lumina-dimoo: An omni diffusion large language model for multi-modal generation and understanding. *arXiv preprint arXiv:2510.06308* **2025**.
12. Xin, Y.; Zhuo, L.; Qin, Q.; Luo, S.; Cao, Y.; Fu, B.; He, Y.; Li, H.; Zhai, G.; Liu, X.; et al. Resurrect mask autoregressive modeling for efficient and scalable image generation. *arXiv preprint arXiv:2507.13032* **2025**.
13. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. ReAct: Synergizing Reasoning and Acting in Language Models. In Proceedings of the International Conference on Learning Representations, 2023.
14. Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Zettlemoyer, L.; Cancedda, N.; Scialom, T. Toolformer: Language Models Can Teach Themselves to Use Tools. In Proceedings of the Advances in Neural Information Processing Systems, 2023, Vol. 36.
15. Lin, S. Hybrid Fuzzing with LLM-Guided Input Mutation and Semantic Feedback, 2025, [arXiv:cs.CR/2511.03995].
16. Lin, S. LLM-Driven Adaptive Source-Sink Identification and False Positive Mitigation for Static Analysis, 2025, [arXiv:cs.SE/2511.04023].
17. Wu, X.; Zhang, Y.; Shi, M.; Li, P.; Li, R.; Xiong, N.N. An adaptive federated learning scheme with differential privacy preserving. *Future Generation Computer Systems* **2022**, *127*, 362–372.
18. Xin, Y.; Luo, S.; Zhou, H.; Du, J.; Liu, X.; Fan, Y.; Li, Q.; Du, Y. Parameter-efficient fine-tuning for pre-trained vision models: A survey. *arXiv e-prints* **2024**, pp. arXiv–2402.
19. Tian, Y.; Yang, Z.; Liu, C.; Su, Y.; Hong, Z.; Gong, Z.; Xu, J. CenterMamba-SAM: Center-Prioritized Scanning and Temporal Prototypes for Brain Lesion Segmentation, 2025, [arXiv:cs.CV/2511.01243].
20. Xin, Y.; Luo, S.; Jin, P.; Du, Y.; Wang, C. Self-training with label-feature-consistency for domain adaptation. In Proceedings of the International Conference on Database Systems for Advanced Applications. Springer, 2023, pp. 84–99.

21. Park, J.S.; O'Brien, J.C.; Cai, C.J.; Morris, M.R.; Liang, P.; Bernstein, M.S. Generative Agents: Interactive Simulacra of Human Behavior. In Proceedings of the Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, 2023, pp. 1–22.
22. Yu, Z.; Idris, M.Y.I.; Wang, P. Visualizing our changing Earth: A creative AI framework for democratizing environmental storytelling through satellite imagery. In Proceedings of the NeurIPS Creative AI Track, 2025.
23. Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. A Survey on Large Language Model based Autonomous Agents. *Frontiers of Computer Science* **2024**, *18*, 186345.
24. Xi, Z.; Chen, W.; Guo, X.; He, W.; Ding, Y.; Hong, B.; Zhang, M.; Wang, J.; Jin, S.; Zhou, E.; et al. The Rise and Potential of Large Language Model Based Agents: A Survey. *arXiv preprint arXiv:2309.07864* **2023**.
25. Wang, Z.; et al. A Survey on Large Language Model-based Autonomous Agents. *Frontiers of Computer Science* **2024**.
26. Sumers, T.; Yao, S.; Narasimhan, K.; Griffiths, T. Cognitive Architectures for Language Agents. *Transactions on Machine Learning Research* **2024**.
27. He, Y.; Li, S.; Li, K.; Wang, J.; Li, B.; Shi, T.; Xin, Y.; Li, K.; Yin, J.; Zhang, M.; et al. GE-Adapter: A General and Efficient Adapter for Enhanced Video Editing with Pretrained Text-to-Image Diffusion Models. *Expert Systems with Applications* **2025**, p. 129649.
28. Yang, C.; He, Y.; Tian, A.X.; Chen, D.; Wang, J.; Shi, T.; Heydarian, A.; Liu, P. Wcdt: World-centric diffusion transformer for traffic scene generation. In Proceedings of the 2025 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2025, pp. 6566–6572.
29. Jimenez, C.E.; Yang, J.; Wettig, A.; Yao, S.; Pei, K.; Press, O.; Narasimhan, K. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? In Proceedings of the International Conference on Learning Representations, 2024.
30. Liang, X.; Tao, M.; Xia, Y.; Wang, J.; Li, K.; Wang, Y.; He, Y.; Yang, J.; Shi, T.; Wang, Y.; et al. SAGE: Self-evolving Agents with Reflective and Memory-augmented Abilities. *Neurocomputing* **2025**, p. 130470.
31. Zhou, S.; Xu, F.F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; et al. WebArena: A Realistic Web Environment for Building Autonomous Agents. In Proceedings of the International Conference on Learning Representations, 2024.
32. Liu, X.; Yu, H.; Zhang, H.; Xu, Y.; Lei, X.; Lai, H.; Gu, Y.; Ding, H.; Men, K.; Yang, K.; et al. AgentBench: Evaluating LLMs as Agents. In Proceedings of the International Conference on Learning Representations, 2024.
33. Gao, B.; Wang, J.; Song, X.; He, Y.; Xing, F.; Shi, T. Free-Mask: A Novel Paradigm of Integration Between the Segmentation Diffusion Model and Image Editing. In Proceedings of the Proceedings of the 33rd ACM International Conference on Multimedia, 2025, pp. 9881–9890.
34. Wang, J.; He, Y.; Zhong, Y.; Song, X.; Su, J.; Feng, Y.; Wang, R.; He, H.; Zhu, W.; Yuan, X.; et al. Twin co-adaptive dialogue for progressive image generation. In Proceedings of the Proceedings of the 33rd ACM International Conference on Multimedia, 2025, pp. 3645–3653.
35. Zhou, Y.; He, Y.; Su, Y.; Han, S.; Jang, J.; Bertasius, G.; Bansal, M.; Yao, H. ReAgent-V: A Reward-Driven Multi-Agent Framework for Video Understanding. *arXiv preprint arXiv:2506.01300* **2025**.
36. Xin, Y.; Du, J.; Wang, Q.; Lin, Z.; Yan, K. Vmt-adapter: Parameter-efficient transfer learning for multi-task dense scene understanding. In Proceedings of the Proceedings of the AAAI conference on artificial intelligence, 2024, Vol. 38, pp. 16085–16093.
37. Lin, S. Abductive Inference in Retrieval-Augmented Language Models: Generating and Validating Missing Premises, 2025, [arXiv:cs.CL/2511.04020].
38. Agarwal, R.; Vieillard, N.; et al. Many-Shot In-Context Learning. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37.
39. Tanwar, P.; Bhandari, A.; et al. LLMs Are Few-Shot In-Context Low-Resource Language Learners. In Proceedings of the Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics, 2024.
40. Wu, X.; Wang, H.; Tan, W.; Wei, D.; Shi, M. Dynamic allocation strategy of VM resources with fuzzy transfer learning method. *Peer-to-Peer Networking and Applications* **2020**, *13*, 2201–2213.
41. Zhang, G.; Chen, K.; Wan, G.; Chang, H.; Cheng, H.; Wang, K.; Hu, S.; Bai, L. Evoflow: Evolving diverse agentic workflows on the fly. *arXiv preprint arXiv:2502.07373* **2025**.
42. Chen, K.; Lin, Z.; Xu, Z.; Shen, Y.; Yao, Y.; Rimchala, J.; Zhang, J.; Huang, L. R2I-Bench: Benchmarking Reasoning-Driven Text-to-Image Generation. *arXiv preprint arXiv:2505.23493* **2025**.

43. Chen, H.; Peng, J.; Min, D.; Sun, C.; Chen, K.; Yan, Y.; Yang, X.; Cheng, L. MVI-Bench: A Comprehensive Benchmark for Evaluating Robustness to Misleading Visual Inputs in LVLMs. *arXiv preprint arXiv:2511.14159* **2025**.
44. Wu, X.; Wang, H.; Zhang, Y.; Zou, B.; Hong, H. A tutorial-generating method for autonomous online learning. *IEEE Transactions on Learning Technologies* **2024**, *17*, 1532–1541.
45. Wu, X.; Zhang, Y.T.; Lai, K.W.; Yang, M.Z.; Yang, G.L.; Wang, H.H. A novel centralized federated deep fuzzy neural network with multi-objectives neural architecture search for epistatic detection. *IEEE Transactions on Fuzzy Systems* **2024**, *33*, 94–107.
46. Qi, H.; Hu, Z.; Yang, Z.; Zhang, J.; Wu, J.J.; Cheng, C.; Wang, C.; Zheng, L. Capacitive aptasensor coupled with microfluidic enrichment for real-time detection of trace SARS-CoV-2 nucleocapsid protein. *Analytical chemistry* **2022**, *94*, 2812–2819.
47. Cao, Z.; He, Y.; Liu, A.; Xie, J.; Chen, F.; Wang, Z. TV-RAG: A Temporal-aware and Semantic Entropy-Weighted Framework for Long Video Retrieval and Understanding. In Proceedings of the Proceedings of the 33rd ACM International Conference on Multimedia, 2025, pp. 9071–9079.
48. Cao, Z.; He, Y.; Liu, A.; Xie, J.; Wang, Z.; Chen, F. PurifyGen: A Risk-Discrimination and Semantic-Purification Model for Safe Text-to-Image Generation. In Proceedings of the Proceedings of the 33rd ACM International Conference on Multimedia, 2025, pp. 816–825.
49. Xin, Y.; Yan, J.; Qin, Q.; Li, Z.; Liu, D.; Li, S.; Huang, V.S.J.; Zhou, Y.; Zhang, R.; Zhuo, L.; et al. Lumina-mgpt 2.0: Stand-alone autoregressive image modeling. *arXiv preprint arXiv:2507.17801* **2025**.
50. Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288* **2023**.
51. Rozière, B.; Gehring, J.; Gloeckle, F.; Sootla, S.; Gat, I.; Tan, X.E.; Adi, Y.; Liu, J.; Remez, T.; Rapin, J.; et al. Code Llama: Open Foundation Models for Code. *arXiv preprint arXiv:2308.12950* **2023**.
52. Bai, Y.; Kadavath, S.; Kundu, S.; Askell, A.; Kernion, J.; Jones, A.; Chen, A.; Goldie, A.; Mirhoseini, A.; McKinnon, C.; et al. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint arXiv:2212.08073* **2022**.
53. Cao, Z.; He, Y.; Liu, A.; Xie, J.; Wang, Z.; Chen, F. CoFi-Dec: Hallucination-Resistant Decoding via Coarse-to-Fine Generative Feedback in Large Vision-Language Models. In Proceedings of the Proceedings of the 33rd ACM International Conference on Multimedia, 2025, pp. 10709–10718.
54. Wu, X.; Dong, J.; Bao, W.; Zou, B.; Wang, L.; Wang, H. Augmented intelligence of things for emergency vehicle secure trajectory prediction and task offloading. *IEEE Internet of Things Journal* **2024**, *11*, 36030–36043.
55. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In Proceedings of the Advances in Neural Information Processing Systems, 2022, Vol. 35, pp. 24824–24837.
56. Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.L.; Cao, Y.; Narasimhan, K. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In Proceedings of the Advances in Neural Information Processing Systems, 2023, Vol. 36.
57. Packer, C.; Fang, V.; Patil, S.G.; Lin, K.; Wooders, S.; Gonzalez, J.E. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560* **2023**.
58. Richards, T.B. Auto-GPT: An Autonomous GPT-4 Experiment. https://github.com/Significant-Gravitas/AutoGPT, 2023.
59. Chase, H. LangChain. https://github.com/langchain-ai/langchain, 2022.
60. Kojima, T.; Gu, S.S.; Reid, M.; Matsuo, Y.; Iwasawa, Y. Large Language Models are Zero-Shot Reasoners. In Proceedings of the Advances in Neural Information Processing Systems, 2022, Vol. 35, pp. 22199–22213.
61. Besta, M.; Blach, N.; Kubicek, A.; Gerstenberger, R.; Podstawski, M.; Gianinazzi, L.; Gajda, J.; Lehmann, T.; Nyczyk, H.; Pyrka, P.; et al. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2024, Vol. 38, pp. 17682–17690.
62. Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; Zhou, D. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In Proceedings of the International Conference on Learning Representations, 2023.
63. Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; Yao, S. Reflexion: Language Agents with Verbal Reinforcement Learning. In Proceedings of the Advances in Neural Information Processing Systems, 2023, Vol. 36.

64. Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; et al. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. In Proceedings of the International Conference on Learning Representations, 2024.
65. Patil, S.G.; Zhang, T.; Wang, X.; Gonzalez, J.E. Gorilla: Large Language Model Connected with Massive APIs. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37.
66. OpenAI. Function Calling and Other API Updates. https://openai.com/blog/function-calling-and-other-api-updates, 2023.
67. Gao, L.; Madaan, A.; Zhou, S.; Alon, U.; Liu, P.; Yang, Y.; Callan, J.; Neubig, G. PAL: Program-aided Language Models. In Proceedings of the International Conference on Machine Learning. PMLR, 2023, pp. 10764–10799.
68. Nakano, R.; Hilton, J.; Balaji, S.; Wu, J.; Ouyang, L.; Kim, C.; Hesse, C.; Jain, S.; Kosaraju, V.; Saunders, W.; et al. WebGPT: Browser-assisted Question-answering with Human Feedback. *arXiv preprint arXiv:2112.09332* **2022**.
69. Li, G.; Hammoud, H.A.A.K.; Itani, H.; Khizbullin, D.; Ghanem, B. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In Proceedings of the Advances in Neural Information Processing Systems, 2023, Vol. 36.
70. Hong, S.; Zhuge, M.; Chen, J.; Zheng, X.; Cheng, Y.; Zhang, C.; Wang, J.; Wang, Z.; Yau, S.K.S.; Lin, Z.; et al. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In Proceedings of the International Conference on Learning Representations, 2024.
71. Qian, C.; Liu, W.; Liu, H.; Chen, N.; Dang, Y.; Li, J.; Yang, C.; Chen, W.; Su, Y.; Cong, X.; et al. ChatDev: Communicative Agents for Software Development. In Proceedings of the Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, 2024, pp. 15174–15186.
72. Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Zhang, S.; Zhu, E.; Li, B.; Jiang, L.; Zhang, X.; Wang, C. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. *arXiv preprint arXiv:2308.08155* **2023**.
73. Qiao, B.; Li, L.; Zhang, X.; He, S.; Kang, Y.; Zhang, C.; Yang, F.; Dong, H.; Zhang, J.; Wang, L.; et al. TaskWeaver: A Code-First Agent Framework. *arXiv preprint arXiv:2311.17541* **2024**.
74. Chen, W.; Su, Y.; Zuo, J.; Yang, C.; Yuan, C.; Qian, C.; Chan, C.M.; Qin, Y.; Lu, Y.; Xie, R.; et al. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. In Proceedings of the International Conference on Learning Representations, 2024.
75. Huang, X.; Liu, W.; Chen, X.; Wang, X.; Wang, H.; Lian, D.; Wang, Y.; Tang, R.; Chen, E. Understanding the Planning of LLM Agents: A Survey. *arXiv preprint arXiv:2402.02716* **2024**.
76. Sahoo, P.; Singh, A.K.; Saha, S.; Jain, V.; Mondal, S.; Chadha, A. A Survey of Prompt Engineering Methods in Large Language Models for Different NLP Tasks. *arXiv preprint arXiv:2407.12994* **2024**.
77. Bi, Z.; Chen, K.; Wang, T.; Hao, J.; Song, X. CoT-X: An Adaptive Framework for Cross-Model Chain-of-Thought Transfer and Optimization. *arXiv:2511.05747* **2025**.
78. Zhou, D.; Schärli, N.; Hou, L.; Wei, J.; Scales, N.; Wang, X.; Schuurmans, D.; Cui, C.; Bousquet, O.; Le, Q.; et al. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In Proceedings of the International Conference on Learning Representations, 2023.
79. Wang, L.; Xu, W.; Lan, Y.; Hu, Z.; Lan, Y.; Lee, R.K.W.; Lim, E.P. Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. In Proceedings of the Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, 2023, pp. 2609–2634.
80. Schulhoff, S.; Ilie, M.; Balepur, N.; et al. The Prompt Report: A Systematic Survey of Prompting Techniques. *arXiv preprint arXiv:2406.06608* **2024**.
81. Zhang, Y.; Gao, K.; Zhang, Q.; Liu, Q. KnowAgent: Knowledge-Augmented Planning for LLM-Based Agents. *arXiv preprint arXiv:2403.03101* **2024**.
82. Press, O.; Zhang, M.; Min, S.; Schmidt, L.; Smith, N.A.; Lewis, M. Measuring and Narrowing the Compositionality Gap in Language Models. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2023, 2023.
83. Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W.W.; Salakhutdinov, R.; Manning, C.D. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In Proceedings of the Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 2369–2380.
84. Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. Self-Refine: Iterative Refinement with Self-Feedback **2023**. *36*.
85. Gou, Z.; Shao, Z.; Gong, Y.; Shen, Y.; Yang, Y.; Huang, M.; Duan, N.; Chen, W. CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing. In Proceedings of the International Conference on Learning Representations, 2024.

86. Chen, B.; Shu, C.; Shareghi, E.; Collier, N.; Narasimhan, K.; Yao, S. FireAct: Toward Language Agent Fine-tuning. *arXiv preprint arXiv:2310.05915* **2023**.
87. Shen, Y.; Song, K.; Tan, X.; Li, D.; Lu, W.; Zhuang, Y. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 36.
88. Deng, X.; Gu, Y.; Zheng, B.; Chen, S.; Stevens, S.; Wang, B.; Sun, H.; Su, Y. Mind2Web: Towards a Generalist Agent for the Web. In Proceedings of the Advances in Neural Information Processing Systems, 2023, Vol. 36.
89. Tran, K.T.; Dao, D.; Nguyen, M.D.; Pham, Q.V.; O'Sullivan, B.; Nguyen, H.D. Multi-Agent Collaboration Mechanisms: A Survey of LLMs. *arXiv preprint arXiv:2501.06322* **2025**.
90. Guo, T.; Chen, X.; Wang, Y.; Chang, R.; Pei, S.; Chawla, N.V.; Wiest, O.; Zhang, X. Large Language Model based Multi-Agents: A Survey of Progress and Challenges. In Proceedings of the Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI), 2024.
91. Nair, V.; Schumacher, E.; Tso, G.K.F.; Kannan, A. DERA: Enhancing Large Language Model Completions with Dialog-Enabled Resolving Agents. *arXiv preprint arXiv:2303.17071* **2023**.
92. Zhang, C.; Yang, K.; Hu, S.; Wang, Z.; Li, G.; Sun, Y.; Zhang, C.; Zhang, Z.; Liu, A.; Zhu, S.C.; et al. ProAgent: Building Proactive Cooperative Agents with Large Language Models. *arXiv preprint arXiv:2308.11339* **2023**.
93. Zhang, Y.; Gu, R.; Diab, M.; et al. Chain-of-Agents: Large Language Models Collaborating on Long-Context Tasks. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37.
94. Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Wang, H. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint arXiv:2312.10997* **2024**.
95. Zhao, P.; Zhang, H.; Yu, Q.; Wang, Z.; Geng, Y.; Fu, F.; Yang, L.; Zhang, W.; Cui, B. Retrieval-Augmented Generation for AI-Generated Content: A Survey. *arXiv preprint arXiv:2402.19473* **2024**.
96. Fan, W.; Ding, Y.; Ning, L.; Wang, S.; Li, H.; Yin, D.; Chua, T.S.; Li, Q. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. *arXiv preprint arXiv:2405.06211* **2024**.
97. Jiang, J.; Wang, F.; Shen, J.; Kim, S.; Kim, S. A Survey on Large Language Models for Code Generation. *ACM Transactions on Software Engineering and Methodology* **2024**.
98. Tang, Y.; Luo, T.; et al. CodeAgent: Enhancing Code Generation with Tool-Integrated Agent Systems for Real-World Repo-level Coding Challenges. *arXiv preprint arXiv:2401.07339* **2024**.
99. Xia, C.S.; Deng, Y.; Dunn, S.; Zhang, L. Agentless: Demystifying LLM-based Software Engineering Agents. *arXiv preprint arXiv:2407.01489* **2024**.
100. Cognition AI. Introducing Devin, the First AI Software Engineer. https://cognition.ai/blog/introducing-devin, 2024.
101. Yang, J.; Jimenez, C.E.; Wettig, A.; Lieret, K.; Yao, S.; Narasimhan, K.; Press, O. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37.
102. Wang, X.; Chen, B.; Chen, Z.; Wang, B.; et al. OpenHands: An Open Platform for AI Software Developers as Generalist Agents. *arXiv preprint arXiv:2407.16741* **2024**.
103. Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; Anandkumar, A. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv preprint arXiv:2305.16291* **2023**.
104. Huang, W.; Xia, F.; Xiao, T.; Chan, H.; Liang, J.; Florence, P.; Zeng, A.; Tompson, J.; Mordatch, I.; Chebotar, Y.; et al. Inner Monologue: Embodied Reasoning through Planning with Language Models **2023**. pp. 1769–1782.
105. Wu, Y.; et al. A Survey of Robot Intelligence with Large Language Models. *Applied Sciences* **2024**, *14*, 8868.
106. Yang, S.; Liu, Z.; et al. Embodied Multi-Modal Agent Trained by an LLM from a Parallel TextWorld. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024.
107. He, H.; Yao, W.; Ma, K.; Yu, W.; et al. WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models. *arXiv preprint arXiv:2401.13919* **2024**.
108. Lai, H.; Liu, X.; Shen, T.; Shao, J.; et al. AutoWebGLM: Bootstrap And Reinforce A Large Language Model-based Web Navigating Agent. *arXiv preprint arXiv:2404.03648* **2024**.
109. Gur, I.; Furuta, H.; Huang, A.; Saber, M.; et al. A Real-World WebAgent with Planning, Long Context Understanding, and Program Synthesis. *arXiv preprint arXiv:2307.12856* **2024**.
110. Xie, T.; Zhang, D.; Chen, J.; Li, X.; Zhao, S.; Cao, R.; Hua, T.J.; Cheng, Z.; Shin, D.; Lei, F.; et al. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37.

111. Xu, F.F.; Zhou, Y.; Song, Y.; et al. TheAgentCompany: Benchmarking LLM Agents on Consequential Real World Tasks. *arXiv preprint arXiv:2412.14161* **2024**.
112. Yehudai, A.; Eden, L.; Li, A.; Uziel, G.; Zhao, Y.; Bar-Haim, R.; Cohan, A.; Shmueli-Scheuer, M. Survey on Evaluation of LLM-based Agents. *arXiv preprint arXiv:2503.16416* **2025**.
113. Mohammadi, M.; Li, Y.; Lo, J.; Yip, W. Evaluation and Benchmarking of LLM Agents: A Survey. *arXiv preprint arXiv:2507.21504* **2025**.
114. Yang, J.; Prabhakar, A.; Narasimhan, K.; Yao, S. InterCode: Standardizing and Benchmarking Interactive Coding with Execution Feedback. In Proceedings of the Advances in Neural Information Processing Systems, 2023, Vol. 36.
115. Huang, L.; Yu, W.; Ma, W.; Zhong, W.; Feng, Z.; Wang, H.; Chen, Q.; Peng, W.; Feng, X.; Qin, B.; et al. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *arXiv preprint arXiv:2311.05232* **2024**.
116. Tonmoy, S.M.; et al. A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models. *arXiv preprint arXiv:2401.01313* **2024**.
117. Farquhar, S.; Kossen, J.; Kuhn, L.; Gal, Y. Detecting Hallucinations in Large Language Models Using Semantic Entropy. *Nature* **2024**, *630*, 625–630.
118. Peng, B.; Chen, K.; Li, M.; Feng, P.; Bi, Z.; Liu, J.; Niu, Q. Securing large language models: Addressing bias, misinformation, and prompt attacks. *arXiv:2409.08087* **2024**.
119. Niu, Q.; Liu, J.; Bi, Z.; Feng, P.; Peng, B.; Chen, K.; Li, M.; Yan, L.K.; Zhang, Y.; Yin, C.H.; et al. Large language models and cognitive science: A comprehensive review of similarities, differences, and challenges. *BIO Integration 2025* **2024**.
120. Wang, K.; et al. A Comprehensive Survey in LLM(-Agent) Full Stack Safety: Data, Training and Deployment. *arXiv preprint arXiv:2504.15585* **2025**.
121. Yu, M.; Meng, F.; Zhou, X.; Wang, S.; Mao, J.; Pang, L.; Chen, T.; Wang, K.; Li, X.; Zhang, Y.; et al. A Survey on Trustworthy LLM Agents: Threats and Countermeasures. *arXiv preprint arXiv:2503.09648* **2025**.
122. Yi, J.; Guo, R.; Hong, Q.; et al. On the Vulnerability of Safety Alignment in Open-Access LLMs. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024, 2024.
123. Chen, S.; Wang, T.; Jing, B.; Yang, J.; Song, J.; Chen, K.; Li, M.; Niu, Q.; Liu, J.; Peng, B.; et al. Ethics and Social Implications of Large Models **2024**.
124. Wang, T.; Wang, Y.; Zhou, J.; Peng, B.; Song, X.; Zhang, C.; Sun, X.; Niu, Q.; Liu, J.; Chen, S.; et al. From Aleatoric to Epistemic: Exploring Uncertainty Quantification Techniques in Artificial Intelligence. *arXiv preprint arXiv:2501.03282* **2025**.
125. Kaufmann, T.; et al. A Survey of Reinforcement Learning from Human Feedback. *arXiv preprint arXiv:2312.14925* **2024**.
126. Dai, J.; Pan, X.; Sun, R.; et al. Safe RLHF: Safe Reinforcement Learning from Human Feedback. In Proceedings of the International Conference on Learning Representations, 2024.
127. Lee, H.; Phatale, S.; Mansoor, H.; et al. RLAIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. *arXiv preprint arXiv:2309.00267* **2024**.