# Reasoning RAG via System 1 or System 2: A Survey on Reasoning Agentic Retrieval-Augmented Generation for Industry Challenges

**Jintao Liang[1], Gang Su[2], Huifeng Lin[3], You Wu[4], Rui Zhao[5,6], Ziyue Li[7]***,
[1]Beijing University of Posts and Telecommunications, [2]University of Georgia,
[3]South China University of Technology, [4]Hong Kong University of Science and Technology,
[5]SenseTime Research, [6]Qingyuan Research Institute, Shanghai Jiaotong University,
[7]Department of Operations and Technology, and Munich Data Science Institute,
and Heilbronn Data Science Center, Technical University of Munich
ljt2021@bupt.edu.cn, {gangsuedu,huifeng.work}@gmail.com, ywuiu@connect.ust.hk
zhaorui@sensetime.com, ziyue.li@tum.de

## Abstract

Retrieval-Augmented Generation (RAG) has emerged as a powerful framework to overcome the knowledge limitations of Large Language Models (LLMs) by integrating external retrieval with language generation. While early RAG systems based on static pipelines have shown effectiveness in well-structured tasks, they struggle in real-world scenarios requiring complex reasoning, dynamic retrieval, and multi-modal integration. To address these challenges, the field has shifted toward *Reasoning Agentic RAG*, a paradigm that embeds decision-making and adaptive tool use directly into the retrieval process. In this paper, we present a comprehensive review of Reasoning Agentic RAG methods, categorizing them into two primary systems: *predefined reasoning*, which follow fixed modular pipelines to boost reasoning, and *agentic reasoning*, where the model autonomously orchestrates tool interaction during inference. We analyze representative techniques under both paradigms, covering architectural design, reasoning strategies, and tool coordination. Finally, we discuss key research challenges and propose future directions to advance the flexibility, robustness, and applicability of reasoning agentic RAG systems. Our collection of the relevant researches has been organized into a GitHub Repository.

## 1 Introduction

Large Language Models (LLMs) (Singh, 2023; Zhao et al., 2023; Zhu et al., 2024) have demonstrated remarkable capabilities in natural language understanding and generation, enabling a wide array of applications. However, LLMs rely on static training data, making them prone to hallucinations and limiting their ability to provide accurate, up-to-date information in dynamic or knowledge-intensive tasks (Rawte et al., 2023; Zhang et al., 2023; Huang et al., 2025). Retrieval-Augmented Generation (RAG) (Chen et al., 2024; Lewis et al., 2020; Gao et al., 2023) has attracted significant attention as a promising approach to overcome the knowledge limitations of LLMs resulting from static pretraining. By integrating relevant information from external knowledge bases or search engines, RAG enhances factual accuracy and broadens the model's temporal and domain coverage (Zhao et al., 2024; Li et al., 2024a). Traditional RAG methods perform strongly with well-formed queries and readily available necessary information in the retrieved context.

Despite the effectiveness of basic RAG methods, they often struggle when applied to real-world, industrial-scale applications involving complex and heterogeneous data. For example, in multi-document scenarios, relevant information is spread across sources, requiring not just retrieval but also coherent synthesis (Wang et al., 2025, 2024b). Naively concatenating retrieved passages can lead to fragmented or contradictory responses, particularly in domains where multi-hop reasoning is critical. Additionally, most RAG systems are limited to text-only processing and cannot handle multi-modal inputs such as tables, charts, or images (Ma et al., 2024; Yu et al., 2025). This limits their ability in data-rich environments like enterprise intelligence, scientific reporting, or technical support, where visual and structured data play a central role (Lin et al., 2023a; Yu et al., 2024).

To address these limitations of basic RAG in handling complex, real-world tasks, recent research has turned to *Agentic RAG* (Ravuru et al., 2024), a paradigm that tightly integrates retrieval with reasoning and decision-making(see Appendix A.2 for relevant researches). Figure 1 shows the evolution trajectory of Reasoning Agentic RAG. Unlike static pipelines, Agentic RAG treats retrieval as a dynamic, context-sensitive operation guided by the model's reasoning process. This reasoning-centric perspective is crucial for applications demanding

* Corresponding author.



*Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 1954–1966

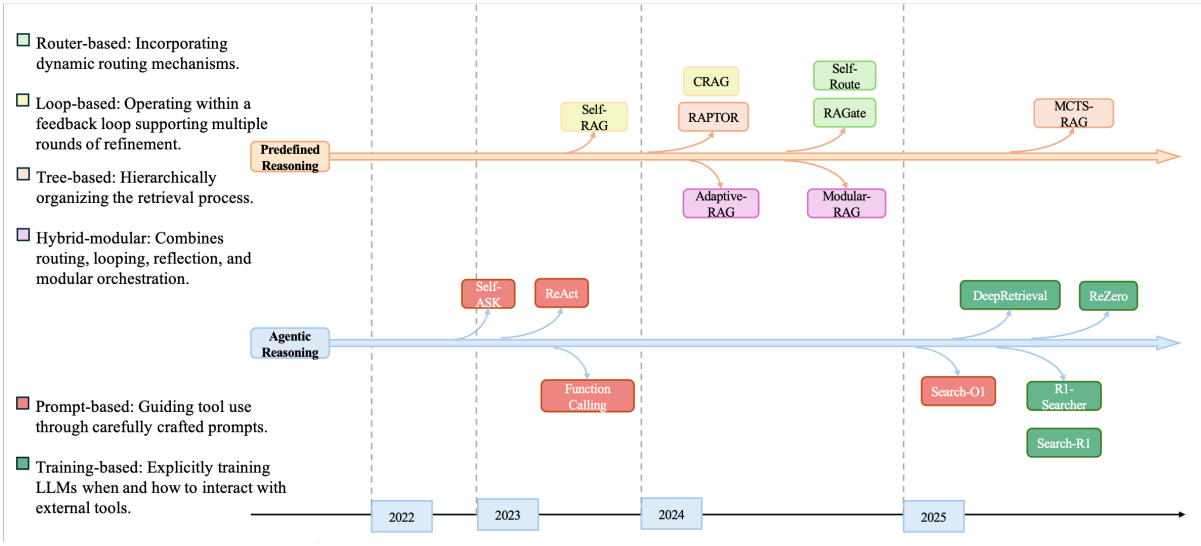
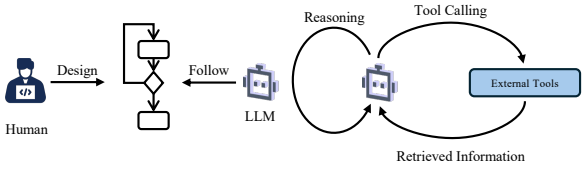



Figure 1: Illustration of the evolution trajectory of Reasoning Agentic RAG.

multi-step problem solving, adaptive information acquisition, and tool-assisted synthesis. Within this paradigm, as shown in Figure 2, two major types of reasoning agentic systems have emerged based on how control and decision-making are handled: *predefined reasoning*, which follow structured, rule-based plans with fixed pipelines to boost reasoning for retrieval and generation; and *agentic reasoning*, where the model actively monitors its reasoning process and determines when and how to retrieve or interact with external tools. These two workflows form the basis of *Reasoning Agentic RAG*, which unifies structured and autonomous approaches for more intelligent, context-aware retrieval-augmented reasoning. Reasoning Agentic RAG can be broadly categorized into two paradigms: *predefined reasoning* and *agentic reasoning*, as shown in Figure 3.




(a) Predefined Reasoning (b) Agentic Reasoning

Figure 2: Two major types of reasoning Agentic Systems.

**Predefined reasoning** adopts structured and modular RAG pipelines where the retrieval and reasoning steps are explicitly designed. These workflows typically decompose tasks into discrete components such as query reformulation, document retrieval, re-ranking, and answer synthesis, executed in a linear or orchestrated fashion. In general, predefined reasoning spans several architectural variants: *route-based* methods selectively trigger retrieval based on context or model uncertainty, such as low confidence scores or ambiguous intermediate outputs (Wang et al., 2024a); *loop-based* methods enable limited iteration through retrieval-feedback cycles, supporting multiple rounds of refinement (Asai et al., 2023; Yang et al., 2024b); *tree-based* methods organize information hierarchically to support structured exploration (Sarthi et al., 2024; Hu et al., 2025); and *hybrid-modular* frameworks compose specialized modules into a flexible but still rule-driven workflow (Jeong et al., 2024; Gao et al., 2024). These workflows prioritize control and modularity, fitting tasks requiring efficient computation and customization. However, their reasoning remains constrained by predesigned execution paths, limiting flexibility in evolving and open-ended tasks.

**Agentic reasoning** repositions the LLM as an active decision maker, that autonomously orchestrates retrieval and tool use throughout the reasoning process. Instead of executing a fixed plan, the model identifies knowledge gaps, formulates queries, retrieves external information via tools such as search engines or APIs, and integrates the retrieved contents into an evolving solution. This dynamic interplay of reasoning and tool use enables the system to tackle complex, multi-turn tasks that require iterative refinement and adaptive information synthesis. There are two primary methods for implementing agentic reasoning. The first is *prompt-based* methods, which leverages the in-context reasoning and instruction-following capa-

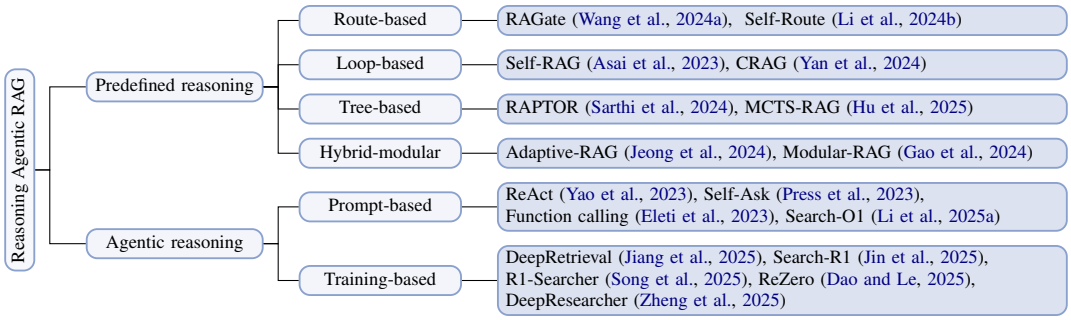



Figure 3: A taxonomy of Reasoning Agentic RAG.

bilities of LLMs (Yao et al., 2023; Press et al., 2023; Li et al., 2025a). In this setting, the model is guided by carefully crafted prompts or embedded control tokens that instruct it when to retrieve, what actions to take, and how to integrate external information. These methods require no additional training, making them lightweight and adaptable across tasks. The second paradigm is *training-based* methods, where models are explicitly optimized through reinforcement learning, to determine when and how to invoke external tools (Jiang et al., 2025; Jin et al., 2025; Zheng et al., 2025). This paradigm enables more fine-grained and strategic tool usage, enabling models to learn long-term planning and develop retrieval policies tailored to complex tasks. Owing to its autonomy and adaptability, agentic reasoning has shown strong performance in open-domain QA, scientific reasoning, and multi-stage decision-making scenarios.

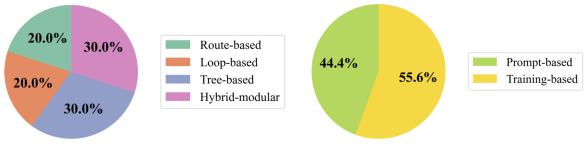
To further contextualize predefined and agentic reasoning within the dual-process theory of cognition, commonly referred to as System 1 and System 2 thinking (Yang et al., 2024a; Li et al., 2025b), we can draw an analogy between these RAG paradigms and human cognitive modes. Table 1 aligns predefined and agentic reasoning with the dual-system theory from cognitive science.

- Predefined reasoning resembles System 1 thinking: fast, structured, and efficient, relying on predefined heuristics and modular workflows that mirror habitual or rule-based cognition. While this enables rapid execution and predictable behavior, it often lacks the flexibility to adapt beyond its design.
- Agentic reasoning aligns more closely with System 2 thinking: slow, deliberative, and adaptive. Here, the LLM actively engages in reasoning, planning, and decision-making, dynamically leveraging external tools and retrieved knowledge to address complex, novel tasks. This reflective mode allows the model to identify gaps, reassess strategies, and adjust its behavior—traits characteristic of conscious, analytical human reasoning.

| System Type | Reasoning Workflow | Description |
|---|---|---|
| System 1 | Predefined Reasoning | Structured, modular, rule-based execution. |
| System 2 | Agentic Reasoning | Autonomous, adaptive, model-driven decision-making. |

Table 1: Cognitive system alignment of reasoning workflows.

The paper systematically reviews and analyzes the current research approaches and future development paths of Reasoning Agentic RAG, summarizing them into two primary technical paradigms, among which the distribution of different works is shown in Figure 4. The remainder of the paper is organized as follows: Section 2 and Section 3 dive into the two types of reasoning workflows within Agentic RAG: *Predefined Reasoning* and *Agentic Reasoning*. Section 4 outlines future research directions and Section 5 concludes the paper.




(a) Predefined Reasoning (b) Agentic Reasoning

Figure 4: Distributed Works of Reasoning Agentic RAG.

## 2 Predefined Reasoning

Agents and RAG are increasingly integrated in advanced AI systems. By augmenting LLMs with external knowledge retrieval, RAG enables agents to ground their reasoning in relevant information. In turn, agent-based reasoning which includes planning, tool use and self-reflection, enhances RAG by guiding the model on what information to retrieve and how to incorporate it into the reasoning process. This synergy supports a predefined reasoning, as illustrated in Figure 5, where the agent iteratively queries external sources (e.g., a local database or

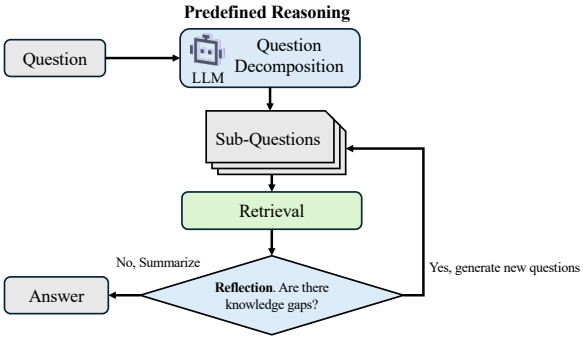



Figure 5: Overview of the Predefined Reasoning. The LLM decomposes a question into sub-questions, retrieves relevant evidence from external sources, and reflects on whether knowledge gaps remain. If gaps are found, new queries are generated; otherwise, the final answer should be summarized.

web search) and refines its reasoning based on the retrieved evidence. We categorize predefined RAG reasoning workflows into four broad types based on their structural and reasoning characteristics.

**Route-based Approaches:** RAG incorporates dynamic routing mechanisms that direct queries along different retrieval or reasoning paths based on predefined conditions—such as query type, model uncertainty, or confidence estimation—while still operating within a fixed architecture. RAGate (Wang et al., 2024a) uses the conversation context and model confidence to route dialogues truly requiring external knowledge to a RAG process. This ensures the system can bypass retrieval for straightforward prompts while invoking it for knowledge-intensive queries, exemplifying conditional RAG in dialogue. Self-Route (Li et al., 2024b) introduced dynamically routes queries to either RAG or Long-Context (LC) models based on the model's confidence-based routing. This method significantly reduces computation cost while maintaining performance comparable to LC models.

**Loop-based Approaches:** RAG operates within a feedback loop that supports multiple rounds of refinement. The system can self-reflect, critique intermediate outputs, and iteratively update retrieval inputs to improve generation quality. Self-RAG (Asai et al., 2023) is a foundational example of this controlled reasoning loop. In the Self-RAG workflow, a single LLM agent engages in self-reflection during generation to improve its output. Instead of relying on a fixed retrieved context, the model can decide mid-generation to fetch additional information or to critique its own draft answer. CRAG (Yan et al., 2024) introduced loop-based corrective feedback mechanism into the retrieval process. In the CRAG workflow, a lightweight retrieval evaluator assigning the confidence scores about the quality of the retrieved chunks/documents — categorized as correct, incorrect, or ambiguous. When retrieval quality is deemed suboptimal, the system activates corrective strategies such as query rewriting or external web search to gather better evidence. The system refines the retrieved content into a focused context and iteratively improves retrieval until a satisfactory output is generated.

**Tree-based Approaches:** RAG organizes the retrieval process hierarchically, using recursive structures such as trees to support multi-hop reasoning or document summarization. RAPTOR (Sarthi et al., 2024) introduces a recursive tree structure from documents, allowing for more efficient and context-aware information retrieval. This approach enhances RAG by creating a summary tree from text chunks, providing deeper insights to overcome limitations of short, contiguous text retrieval. MCTS-RAG (Hu et al., 2025) integrates a Monte Carlo Tree Search loop into the RAG process for complex reasoning tasks. MCTS-RAG dynamically integrates retrieval and reasoning through an iterative decision-making process. Unlike standard RAG, which retrieves information independently from reasoning and thus integrates knowledge suboptimally, or conventional MCTS reasoning, which depends solely on internal model knowledge without external facts, MCTS-RAG combines structured reasoning with adaptive retrieval.

**Hybrid-modular Approaches:** RAG in its most flexible form combines routing, looping, reflection, and modular orchestration. Tasks are divided among specialized components, coordinated by an agent that can reconfigure the workflow according to the query or reasoning context. Adaptive-RAG (Jeong et al., 2024) extends the Self-RAG framework by introducing routing mechanisms that enable dynamic path selection. In addition to allowing the model to interleave retrieval and generation steps, it equips the agent with a decision-making router that selects appropriate retrieval strategies or reasoning pathways based on the query characteristics or the agent's own uncertainty. Rather than simply determining whether to retrieve more information, the agent can choose which retrieval method to apply, what type of information to prioritize, or which downstream modules to engage. Modular-RAG (Gao et al., 2024) is the most advanced incarnation that transform RAG into a LEGO-like modular framework, breaking the RAG process into an orchestrated pipeline of specialized modules.

Rather than a single agent handling everything, a Modular-RAG architecture compartmentalizes tasks, e.g., one module for query reformulation, one for document retrieval, another for ranking or filtering results, and another for answer synthesis, all chained together in a composable workflow. The pipeline is composed by an agent that coordinates modular components, each of which can be optimized or swapped independently.

This progression of predefined reasoning workflows reflects a broader shift from static retrieval pipelines to dynamic, agent-driven reasoning systems. Modern predefined reasoning increasingly integrates planning, tool use, and decision-making components that allow flexible orchestration of retrieval and reasoning strategies. Rather than predefining rigid retrieval steps, these systems empower agents to determine what information to seek, how to use it, and when to adapt their approach—marking a move toward more autonomous and intelligent knowledge integration. A summary of the representative works and open-source industrial implementations across these predefined RAG workflow types is provided in Table 2.

## 3 Agentic Reasoning

Beyond the predefined reasoning mentioned above, a more dynamic paradigm has emerged: the *Agentic Reasoning*. In this setting, the LLM serves as an autonomous agent that not only generates text, but also actively manages retrieval. With advances in reasoning and instruction-following capabilities, the model can identify knowledge gaps, determine when and what to retrieve, and interact with external tools such as search engines or APIs. This tight integration of reasoning and tool use enables iterative decision-making, enabling the system to refine its responses based on newly retrieved information. As a result, agentic reasoning supports more flexible and adaptive problem-solving, extending RAG beyond basic QA to complex tasks such as scientific inquiry, multi-step reasoning, and strategic decision-making. Agentic reasoning approaches can be broadly categorized by how the LLM learns to use tools:

- **Prompt-Based Approaches:** These methods leverage the instruction-following, in-context learning and reasoning capabilities of pre-trained LLMs, guiding tool use through carefully crafted prompts or built-in functionalities without additional training.
- **Training-Based Approaches:** These methods involve explicitly training LLMs via reinforcement learning, to learn when and how to interact with external tools effectively.

A summary of representative agentic reasoning approaches and their characteristics is provided in Table 2. The following sections examine representative and techniques within each approach.

### 3.1 Prompt-Based Approaches

Prompt-based approaches harness the remarkable capabilities already present in pre-trained LLMs to enable agentic behavior. Instead of updating model weights, these methods use advanced prompting, few-shot examples, or built-in tool interfaces to guide LLMs in interacting with external tools such as search engines.

**Function-Calling-Based**: A core prompt-based method for agentic behavior, and one way to implement function calling, is ReAct (Reason+Act) (Yao et al., 2023). ReAct aims to create a synergy between the reasoning processes and action-taking capabilities within an LLM. Its core mechanism involves prompting the LLM to generate outputs in an interleaved sequence of Thought, Action, and Observation. ReAct employs few-shot prompting, providing the LLM with examples that demonstrate this Thought-Action-Observation trajectory for solving similar tasks. the frozen model learns how to structure its reasoning, invoke tools, and move toward a goal. The framework demonstrated significant advantages in grounding the LLM's reasoning. By allowing the model to actively seek and incorporate external information via actions, ReAct can mitigate the hallucination and error propagation issues observed in purely internal reasoning methods like Chain-of-Thought (Wei et al., 2023). The explicit reasoning traces ("Thoughts") in ReAct enhance the interpretability and transparency of the model's decision-making. Within RAG, ReAct offers a natural agentic reasoning pipeline: the LLM's "Thought" process can identify a knowledge gap, leading to a search "Action," with the retrieved results forming the "Observation" that informs subsequent reasoning. A related method, Self-Ask (Press et al., 2023), encourages step-by-step problem decomposition by prompting LLM to generate and answer simpler follow-up questions. These intermediate steps involve search actions, enabling the model to gather relevant information before attempting to answer the main question.

**Predefined Reasoning**

| Approach | Strategy | Control Type | Reasoning Complexity | Code |
|---|---|---|---|---|
| RAGate (Wang et al., 2024a) | Route-based | Adaptive | Medium | Link |
| self-RAG (Asai et al., 2023) | Loop-based | Agentic | Medium | Link |
| CRAG (Yan et al., 2024) | Loop-based | Adaptive | Medium | Link |
| MCTS-RAG (Hu et al., 2025) | Tree-based | Agentic | High | Link |
| RAPTOR (Sarthi et al., 2024) | Tree-based | Fixed | Medium | Link |
| Adaptive-RAG (Jeong et al., 2024) | Hybrid-modular | Adaptive | Medium | Link |
| Modular-RAG (Gao et al., 2024) | Hybrid-modular | Fixed | Low | N/A |
| DeepSearcher | Industry | Adaptive | Medium | Link |
| RAGFlow | Industry | Adaptive | Medium | Link |
| Haystack | Industry | Adaptive | Medium | Link |
| Langchain-Chatchat | Industry | Adaptive/Agentic | Medium | Link |
| LightRAG | Industry | Adaptive | Medium | Link |
| R2R | Industry | Agentic | High | Link |
| FlashRAG | Industry | Adaptive | Medium | Link |

**Agentic Reasoning**

| Approach | Strategy | Training environment | Reward design | Code |
|---|---|---|---|---|
| ReAct (Yao et al., 2023) | Prompt-based | N/A | N/A | Link |
| Self-Ask (Press et al., 2023) | Prompt-based | N/A | N/A | Link |
| Function calling (Eleti et al., 2023) | Prompt-based | N/A | N/A | N/A |
| Search-O1 (Li et al., 2025a) | Prompt-based | N/A | N/A | Link |
| Search-R1 (Jin et al., 2025) | Training-based | Local retrieval system | Answer reward | Link |
| R1-Searcher (Song et al., 2025) | Training-based | Local retrieval system | Retrieval reward, format reward, answer reward | Link |
| ReZero (Dao and Le, 2025) | Training-based | Local retrieval system | Retrieval reward, format reward, answer reward, retry reward | Link |
| DeepRetrieval (Jiang et al., 2025) | Training-based | Restricted real-world search engine | Retrieval reward, format reward | Link |
| DeepResearcher (Zheng et al., 2025) | Training-based | Real-world search engine | Format reward, answer reward | Link |

Table 2: A summary of reasoning agentic rag.

Another prominent prompt-based approach involves leveraging the function calling capabilities that have been explicitly built into or fine-tuned into certain LLMs, such as versions of GPT (Eleti et al., 2023), Llama, and Gemini. This feature allows the LLM to interact reliably with predefined external tools or APIs based on natural language instructions. Function calling significantly expands the capabilities of LLMs beyond text generation, enabling them to access real-time, dynamic information, interact with external systems and databases, automate tasks, and reliably convert natural language requests into structured API calls or database queries. In contrast to the more open-ended "thought-action-observation" cycle of ReAct, function calling often bypasses explicit intermediate reasoning steps. The LLM directly identifies the relevant tool and generates the necessary parameters based on its training to recognize and format specific function calls. This more direct approach relies on the model's pre-existing knowledge of available tools and their required inputs. Furthermore, the format and capabilities of the tools accessible via function calling are typically predefined and have been integrated into the model's training or prompt design. For Agentic RAG, function calling provides a straightforward and structured way for the LLM agent to invoke a search API when its internal analysis determines that external information is required to answer a prompt accurately.

**Large Reasoning Model-based**: A growing trend in Agentic RAG workflow involves directly utilizing LLMs that possess inherently strong reasoning capabilities, often referred to as Large Reasoning Models (LRMs). These models, sometimes developed through techniques like large-scale reinforcement learning (e.g., models analogous to OpenAI's o1 (OpenAI et al., 2024), DeepSeek-R1 (DeepSeek-AI et al., 2025)), are designed to excel at complex, multi-step reasoning tasks. The underlying premise is that an LLM with superior intrinsic reasoning abilities will be better equipped to manage the complexities of an Agentic RAG workflow, including decomposing challenging queries, planning information-gathering steps, assessing the relevance and utility of retrieved information, and synthesizing knowledge effectively. In essence, leveraging LRMs within RAG represents a prompt-based agentic strategy where the model's powerful inherent reasoning capabilities drive the process, implicitly deciding when and how to retrieve information to support its complex thought processes.

However, effectively managing the retrieved context is another significant challenge. LLMs with extremely long context windows can suffer from a "lost-in-the-middle" problem, where information presented in the middle of a long input receives less attention. Furthermore, retrieved documents, whether in long-context models or standard RAG, often contain verbose, noisy or contradictory content that can disrupt the coherence of the LLM's reasoning process. Mitigating this challenge requires

more precise retrieval strategies and adaptive context management mechanisms. As shown in Figure 6, the Search-o1 framework (Li et al., 2025a) is specifically designed to enhance LRMs by tackling knowledge insufficiency during long, step-by-step reasoning chains. It integrates two core components: an Agentic RAG Mechanism where the LRM dynamically triggers search queries based on self-assessed knowledge gaps, and a Reason-in-Documents Module that processes retrieved content to distill relevant information into a refined format, thereby minimizing noise and maintaining the LRM's reasoning integrity. Search-o1 exemplifies a sophisticated prompt-based agentic approach focused on maintaining reasoning integrity in the face of external information retrieval.

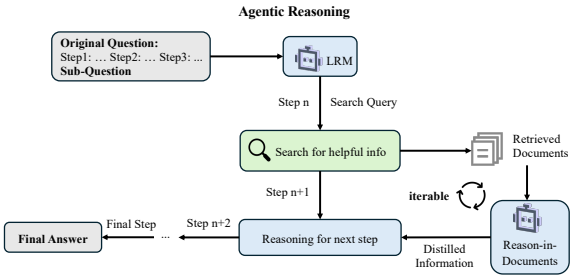



Figure 6: Overview of Agentic Reasoning. The LRM iteratively identifies missing knowledge at each reasoning step, issues targeted search queries, and invokes a Reason-in-Documents module to extract and distill relevant information from retrieved content. The distilled results are incorporated into the next reasoning step until a final answer is derived.

### 3.2 Training-Based Approaches

While prompt-based methods leverage the inherent capabilities of LLMs, their performance in complex tool-use scenarios can be inconsistent. Achieving highly reliable and optimized behavior, especially in deciding when and how to interact with tools like search engines, often benefits from explicit training. Training-based approaches, particularly those utilizing Reinforcement Learning (RL), enable the LLM agent to learn sophisticated strategies through trial and error, directly optimizing its actions towards specific goals such as maximizing retrieval effectiveness or overall task success. RL enables agents to develop more robust and strategic interaction patterns than prompting alone.

**Interacting with local retrieval systems**: Search-R1 (Jin et al., 2025) tackles a different aspect of agentic search: training the LLM to autonomously decide when and what to search for during a multi-step reasoning process. It extends RL-based reasoning frameworks (like DeepSeek-R1) by integrating search engine interaction directly into the learning loop. In the Search-R1 framework, the search engine is modeled as part of the RL environment. The LLM agent learns a policy to generate a sequence of tokens that includes both internal reasoning steps (often enclosed in `<think>` tags) and explicit triggers for search actions. These triggers are special tokens, `<search>` and `</search>`, which encapsulate the generated search query. This design allows for flexible, multi-turn interactions where the LLM can interleave reasoning, searching, processing retrieved information (presented within `<information>` tags), and further reasoning or searching as needed. The framework utilizes a simple outcome-based reward function, typically based on the correctness of the final answer generated by the LLM (within `<answer>` tags) compared to a ground truth, avoiding the complexity of designing intermediate process rewards. A crucial technique employed is retrieved token masking. During the calculation of the RL loss (using algorithms like PPO or GRPO (Shao et al., 2024)), the tokens corresponding to the content retrieved from the search engine (i.e., within the `<information>` tags) are ignored or masked out, which stabilizes the training process. Search-R1 has shown significant performance improvements over various RAG baselines on question-answering datasets. Its core contribution is training the LLM to learn an optimal policy for interacting with the search engine as an integrated part of its reasoning flow, enabling dynamic, context-aware search decisions. The related R1-Searcher (Song et al., 2025) framework also proposes a similar two-stage, outcome-based RL approach for enhancing search capabilities.

ReZero (Retry-Zero) (Dao and Le, 2025) introduces another dimension to RL-based agentic search by specifically focusing on incentivizing persistence. It addresses the common scenario where an initial search query might fail to retrieve the necessary information, potentially causing the LLM agent to halt prematurely or generate a suboptimal response. ReZero aims to teach the agent the value of "trying one more time." The framework operates within a standard RL setup (using GRPO is mentioned) where the LLM interacts with a search environment. The novelty lies in its modified reward function, which includes a specific component termed reward retry. This component provides a positive reward signal whenever the LLM issues a `<search>` query after the initial search query within the same reasoning trajectory. Crucially, this reward for retrying is conditional upon the agent

successfully completing the task, indicated by generating a final answer enclosed in <answer> tags. This conditionality prevents the agent from accumulating rewards simply by retrying indefinitely without making progress. By directly rewarding the act of persistence (when productive), ReZero encourages the LLM to explore alternative queries or search strategies if the first attempt proves insufficient. This contrasts with methods that might only implicitly reward persistence through eventual task success. ReZero positions itself as complementary to frameworks like DeepRetrieval; while DeepRetrieval focuses on optimizing a single refined query, ReZero emphasizes the value of making multiple retrieval attempts when needed.

**Interacting with real-world search engines**: DeepRetrieval (Jiang et al., 2025) focuses specifically on improving the quality of the search queries generated by the LLM agent. It frames the task of query generation or rewriting as an RL problem, training the LLM to transform an initial user query into a more effective query for downstream retrieval systems. The core mechanism involves the LLM generating an augmented or rewritten query based on the input query. DeepRetrieval employs RL algorithms like Proximal Policy Optimization (PPO) (Schulman et al., 2017) to train this query generation process. A key innovation lies in its reward signal: instead of relying on supervised data (e.g., pairs of original and "gold" rewritten queries), DeepRetrieval uses the performance of the generated query in the actual retrieval system as the reward. Metrics such as recall@k, Normalized Discounted Cumulative Gain (NDCG), or evidence-seeking retrieval accuracy (Hits@N) obtained from executing the generated query against a restricted real search engine (like PubMed) or document collection are used to provide feedback to the LLM. The model learns, through trial and error, to generate queries that maximize these retrieval metrics. To structure the generation, the model often produces reasoning steps within <think> tags before outputting the final query in an <answer> tag. This approach offers significant advantages. By directly optimizing for the end goal (retrieval performance), it bypasses the need for expensive and potentially suboptimal supervised query datasets. Compared to other RL methods, DeepRetrieval's primary focus is on optimizing the content and formulation of the search query itself.

DeepResearcher (Zheng et al., 2025) pushes the boundaries of training-based Agentic RAG by moving beyond controlled environments or static corpora to perform end-to-end RL training directly within real-world web. It aims to equip LLM agents with the capabilities for complex research tasks that require navigating the noisy, unstructured and dynamic nature of the open web. This addresses a key limitation of many existing agents, whether prompt-engineered or trained in simulated/static RAG settings, which struggle with the complexities of real-world web interaction. The framework employs RL (specifically GRPO with an F1 score-based reward for answer accuracy) to train agents that interact with web search APIs and browse actual webpages. DeepResearcher utilizes a specialized multi-agent architecture to handle the complexities of web interaction. This includes a reasoning module for invoking web search, and dedicated "browsing agents" responsible for extracting relevant information from the diverse structures of webpages encountered. Training in this realistic setting was found to foster several emergent cognitive behaviors not typically observed in agents trained under more constrained conditions. These include the ability to formulate initial plans and dynamically adjust them during research process, cross-validate information retrieved from multiple web sources, engage in self-reflection when retrieved information seems contradictory or insufficient leading to refined search strategies, and exhibit honesty by declining to provide an answer when definitive information cannot be found. DeepResearcher demonstrated substantial performance improvements over prompt-engineering baselines and RAG-based RL agents trained on static corpora, particularly on open-domain research tasks. The results suggest that end-to-end training in realistic web environments is crucial for developing capable research agents, moving closer to the capabilities hinted at by proprietary systems like OpenAI's Deep Research (OpenAI, 2025) or Grok's DeeperSearch.

The progression for the training-based methods, from optimizing the decision process of when and what to query (Search-R1), to fostering persistence (ReZero), optimizing query formulation (DeepRetrieval), and managing real-world research workflows (DeepResearcher) reflects the growing sophistication of RL in agentic search. It reflects a growing appreciation that effective information seeking by an agent involves a confluence of factors: query quality, strategic timing, resilience to failure, and adeptness in navigating realistic information environments and so on. Future advance-

ments in RL-based Agentic RAG will likely need to integrate these facets more holistically, perhaps through more complex reward structures, multi-objective optimization, or architectures that explicitly model these different dimensions of the search process, to achieve truly human-like research and problem-solving capabilities.

## 4 Future Research Directions

**Enhancing tool interaction through advanced configuration.** Current agentic reasoning often utilizes search tools with relatively basic interfaces, primarily focused on generating text queries. Future work should enable agents to exploit more advanced configurations offered by external APIs and tools. This could involve training agents to understand and utilize options like result filtering (e.g., by date, source type), sorting criteria, specifying search domains, or interacting with structured databases via complex queries. Granting finer control would support more targeted, efficient, and strategic retrieval aligned with task demands.

**Developing finer-Grained and process-oriented reward functions.** Simple outcome-based rewards like exact match may not offer adequate guidance for complex RAG tasks that require multi-step reasoning or detailed responses. Future research should develop fine-grained reward functions that assess both final answer correctness and intermediate steps such as document relevance, reasoning coherence, information cross-validation, and effective problem decomposition. These signals are vital for training agents to handle queries requiring more than short factual answers.

**Improving Efficiency in Retrieval.** The approaches mentioned above primarily focus on the accuracy of the final answer, but enhancing the efficiency of the retrieval process itself is also critical. Agents trained to interact with potentially vast information sources, must learn to perform retrievals strategically. Future research should focus on techniques that help agents avoid excessive or unnecessary search queries, select the most promising sources, and know when sufficient information has been gathered. Developing strategies to prevent agents from getting stuck in loops of unproductive searching or performing redundant retrievals is vital for practical and scalable Agentic RAG.

**Enhancing Generalization and Robustness in Dynamic Environments.** Robust generalization to new queries, unseen tools (e.g., sparse or dense retrieval), and changing environments remains a major challenge. While training in realistic conditions (as in DeepResearcher) improves resilience, agents still struggle with tool failures and shifting knowledge availability. Future work should explore adaptive training methodologies and architectures that ensure robust performance in unfamiliar or dynamic settings.

By addressing key areas such as improving agent control over tools, designing more sophisticated reward signals, increasing efficiency, and enhancing generalization, the field can move toward building more capable, reliable, and widely applicable Agentic RAG systems. These advancements are essential for transitioning agentic AI from research prototypes to practical systems that can effectively support humans in complex information tasks.

## 5 Conclusions

As language models are increasingly deployed in complex, knowledge-intensive applications, the limitations of static RAG pipelines have become apparent. Reasoning Agentic RAG offers a promising path forward by integrating retrieval with model-driven planning, self-reflection, and tool use. This paper surveyed the landscape of reasoning workflows within Agentic RAG, distinguishing between predefined reasoning with fixed orchestration, and agentic reasoning that enables dynamic, autonomous decision-making. We reviewed key methods across both paradigms, highlighting their strengths, limitations, and use-case applicability. To advance the field, we identify several crucial directions for future research, including fine-grained reward design, enhanced tool control, automated data synthesis, and robust training in dynamic environments. These innovations will be essential for realizing intelligent, context-aware RAG systems capable of addressing real-world challenges with greater adaptability, transparency, and reliability.

## Limitations

While this paper provides a comprehensive review of Reasoning Agentic RAG methods, it still has limitations in terms of resource coverage. The study mainly focuses on representative published literature and open-source projects in recent years, excluding closed-source solutions from the analysis scope. This may lead to an incomplete understanding of the practical application landscape of Reasoning Agentic RAG.

## References


Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *Preprint*, arXiv:2310.11511.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762.

Alan Dao and Thinh Le. 2025. Rezero: Enhancing llm search ability by trying one-more-time. *Preprint*, arXiv:2504.11001.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Atty Eleti, Jeff Harris, and Logan Kilpatrick. 2023. Function calling and other api updates.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2:1.

Yunfan Gao, Yun Xiong, Meng Wang, and Haofen Wang. 2024. Modular rag: Transforming rag systems into lego-like reconfigurable frameworks. *Preprint*, arXiv:2407.21059.

Yunhai Hu, Yilun Zhao, Chen Zhao, and Arman Cohan. 2025. Mcts-rag: Enhancing retrieval-augmented generation with monte carlo tree search. *Preprint*, arXiv:2503.20757.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *Preprint*, arXiv:2403.14403.

Pengcheng Jiang, Jiacheng Lin, Lang Cao, Runchu Tian, SeongKu Kang, Zifeng Wang, Jimeng Sun, and Jiawei Han. 2025. Deepretrieval: Hacking real search engines and retrievers with large language models via reinforcement learning. *Preprint*, arXiv:2503.00223.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *Preprint*, arXiv:2503.09516.

Yilun Kong, Jingqing Ruan, Yihong Chen, Bin Zhang, Tianpeng Bao, Shi Shiwei, Du Qing, Xiaoru Hu, Hangyu Mao, Ziyue Li, and 1 others. 2024. Tptu-v2: Boosting task planning and tool usage of large language model-based agents in real-world industry systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 371–385.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Jiarui Li, Ye Yuan, and Zehua Zhang. 2024a. Enhancing llm factual accuracy with rag to counter hallucinations: A case study on domain-specific queries in private knowledge-bases. *arXiv preprint arXiv:2403.10446*.

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

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, and 1 others. 2025b. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.

Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024b. Retrieval augmented generation or long-context llms? a comprehensive study and hybrid approach. *Preprint*, arXiv:2407.16833.

Weizhe Lin, Jinghong Chen, Jingbiao Mei, Alexandru Coca, and Bill Byrne. 2023a. Fine-grained late-interaction multi-modal retrieval for retrieval augmented visual question answering. *Advances in Neural Information Processing Systems*, 36:22820–22840.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, and 1 others. 2023b. Ra-dit: Retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315.

Zi-Ao Ma, Tian Lan, Rong-Cheng Tu, Yong Hu, Heyan Huang, and Xian-Ling Mao. 2024. Multi-modal retrieval augmented multi-modal generation: A benchmark, evaluate metrics and strong baselines. *arXiv preprint arXiv:2411.16365*.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, and 244 others. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

OpenAI. 2025. Deep research system card.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. *Preprint*, arXiv:2210.03350.

Chidaksh Ravuru, Sagar Srinivas Sakhinana, and Venkataramana Runkana. 2024. Agentic retrieval-augmented generation for time series analysis. *arXiv preprint arXiv:2408.14484*.

Vipula Rawte, Swagata Chakraborty, Agnibh Pathak, Anubhav Sarkar, SM_Towhidul Islam Tonmoy, Aman Chadha, Amit Sheth, and Amitava Das. 2023. The troubling emergence of hallucination in large language models-an extensive definition, quantification, and prescriptive remediations. Association for Computational Linguistics.

Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Hangyu Mao, Ziyue Li, Xingyu Zeng, Rui Zhao, and 1 others. 2023. Tptu: Task planning and tool usage of large language model-based ai agents. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. *Preprint*, arXiv:2401.18059.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Aditi Singh. 2023. Exploring language models: A comprehensive survey and analysis. In *2023 International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE)*, pages 1–4. IEEE.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *Preprint*, arXiv:2503.05592.

Han Wang, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. 2025. Retrieval-augmented generation with conflicting evidence. *arXiv preprint arXiv:2504.13079*.

Xi Wang, Procheta Sen, Ruizhe Li, and Emine Yilmaz. 2024a. Adaptive retrieval-augmented generation for conversational systems. *Preprint*, arXiv:2407.21712.

Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024b. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19206–19214.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *Preprint*, arXiv:2401.15884.

Cheng Yang, Chufan Shi, Siheng Li, Bo Shui, Yujiu Yang, and Wai Lam. 2024a. Llm2: Let large language models harness system 2 reasoning. *arXiv preprint arXiv:2412.20372*.

Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Gui, Ziran Jiang, Ziyu Jiang, and 1 others. 2024b. Crag-comprehensive rag benchmark. *Advances in Neural Information Processing Systems*, 37:10470–10490.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Qinhan Yu, Zhiyou Xiao, Binghui Li, Zhengren Wang, Chong Chen, and Wentao Zhang. 2025. Mramg-bench: A beyondtext benchmark for multimodal retrieval-augmented multimodal generation. *arXiv preprint arXiv:2502.04176*.

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, and 1 others. 2024. Visrag: Vision-based retrieval-augmented generation on multi-modality documents. *arXiv preprint arXiv:2410.10594*.

Bin Zhang, Hangyu Mao, Jingqing Ruan, Ying Wen, Yang Li, Shao Zhang, Zhiwei Xu, Dapeng Li, Ziyue Li, Rui Zhao, and 1 others. Controlling large language model-based agents for large-scale decision-making: An actor-critic approach. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, and 1 others. 2023. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *Preprint*, arXiv:2504.03160.

Yizhang Zhu, Shiyin Du, Boyan Li, Yuyu Luo, and Nan Tang. 2024. Are large language models good statisticians? *arXiv preprint arXiv:2406.07815*.

# A Related Work

## A.1 Basic RAG

*Retrieval-Augmented Generation* (RAG) was introduced to overcome the static knowledge limitations of LLMs by integrating external retrieval mechanisms during inference (Chen et al., 2024; Gao et al., 2023). Naive RAG methods represent the earliest implementations, typically using sparse retrieval techniques like BM25 (Robertson et al., 2009) to fetch documents based on keyword overlap (Ma et al., 2023). While efficient for simple factoid queries, these approaches offered limited semantic understanding, thus often retrieving noisy or redundant content and failing to reason across multiple sources.

The emergence of Advanced RAG and Modular RAG was aimed at addressing key limitations of the Naive RAG, particularly in terms of retrieval precision, information integration, and system flexibility (Gao et al., 2023). Advanced RAG improves retrieval quality through techniques such as dense semantic matching, re-ranking, and multi-hop querying, while also introducing refined indexing strategies like fine-grained chunking and metadata-aware retrieval. Modular RAG rethinks the Naive RAG by breaking down the end-to-end process of indexing, retrieval, and generation into discrete, configurable modules. This design allows for greater architectural flexibility and enables system developers to incorporate diverse techniques into specific stages, such as enhancing retrieval with fine-tuned search modules (Lin et al., 2023b). In response to specific task demands, various restructured and iterative module designs have also emerged. As a result, modular RAG has increasingly become a dominant paradigm in the field, supporting both serialized pipeline execution and end-to-end learning across modular components.

Despite their effectiveness, basic RAG workflows are limited by static control logic and lack the ability to reflect, adapt, or assess the sufficiency of retrieved information. These constraints reduce their suitability for tasks requiring iterative reasoning, tool use, or multi-modal integration. Thus, Agentic RAG has proposed to embed reasoning and decision-making into the retrieval process. This work focuses on reasoning Agentic RAG approaches that enable more autonomous and context-aware information processing.

## A.2 Reasoning Agentic RAG

The year 2025 is marked as the year of agentic AI, with applications emerging such as agentic LLMs and so on (Ruan et al., 2023; Kong et al., 2024; Zhang et al.). Recent advances in RAG have seen a shift from static, rule-driven retrieval pipelines toward dynamic, reasoning-driven architectures, collectively referred to as *Reasoning Agentic RAG*. Figure 1 illustrates the evolution trajectory of Reasoning Agentic RAG. These systems embed decision-making into the retrieval process, enabling models to actively determine when, what, and how to retrieve based on their internal reasoning trajectory. As shown in Figure 3, Reasoning Agentic RAG approaches can be broadly categorized into two paradigms: *predefined reasoning* and *agentic reasoning*.

Predefined reasoning depends on structured, rule-based pipelines where the retrieval and reasoning stages are modularized and fixed in advance. These workflows often include components for query reformulation, document retrieval, re-ranking, and response generation, coordinated by static control logic. RAGate (Wang et al., 2024a) exemplifies route-based designs, where retrieval is conditionally triggered based on the context or model con-

fidence, enabling the system to skip unnecessary operations and focus on knowledge-intensive inputs. Self-RAG (Asai et al., 2023) introduces loop-based reasoning by enabling the model to self-reflect and iteratively refine its responses, while RAPTOR (Sarthi et al., 2024) leverages a recursive tree structure to hierarchically summarize and organize retrieved content, supporting multi-hop and abstractive reasoning. Building on these foundations, more advanced frameworks like Adaptive-RAG (Jeong et al., 2024) combine dynamic routing and retrieval adaptation, enabling models to select optimal reasoning paths. Modular-RAG (Gao et al., 2024) extends this idea by dividing the RAG pipeline into interoperable modules like retrievers, rerankers and generators, which can be flexibly composed into hybrid workflows. These designs enabling more flexible orchestration while still operating under predefined execution paths.

Agentic reasoning empowers the LLM to act as an autonomous agent, dynamically deciding how to interact with external tools based on its current reasoning state. These workflows tightly couple reasoning with tool use, enabling the model to issue retrieval queries, assess results, and iteratively adapt its actions. Two main implementation strategies have emerged: *prompt-based* and *training-based* approaches. Prompt-based methods leverage the instruction-following abilities of pretrained LLMs to drive agentic behavior without additional training. For example, ReAct (Yao et al., 2023) interleaves reasoning steps with tool use to guide retrieval based on emerging knowledge gaps. Other methods like Self-Ask (Press et al., 2023) and Search-o1 (Li et al., 2025a) support decomposition into sub-questions or trigger retrieval mid-generation. Additionally, function calling mechanisms (Eleti et al., 2023) built into commercial LLMs such as GPT and Gemini offer structured interfaces for tool use, further enabling prompt-based agentic control. In parallel, training-based approaches aim to explicitly teach LLMs to reason and retrieve in a unified, goal-driven manner by leveraging reinforcement learning (RL) to optimize tool-use behavior. DeepRetrieval (Jiang et al., 2025) trains models to reformulate queries by maximizing retrieval metrics. Search-R1 (Jin et al., 2025) and R1-Searcher (Song et al., 2025) both adopt a two-stage, outcome-driven RL framework that enables LLMs to learn when and what to search within a reasoning trajectory. ReZero (Dao and Le, 2025)incentivizes persistence, rewarding effective retry strategies. DeepResearcher (Zheng et al., 2025) pushes further by training agents in open web environments, enabling robust search and synthesis across diverse, unstructured sources.