Rillor

EXPLAINER

Agentic RAG and how it differs from standard RAG

Agentic RAG vs RAG: standard RAG retrieves once and answers, while agentic RAG lets the model plan searches, grade what it found and search again, at a higher cost.



Rillor

Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

Retrieval-augmented generation (RAG) gives a language model passages from a document index before it answers, and agentic RAG hands that retrieval to an agent that plans searches, judges what comes back and searches again [2, 19, 26].

- Standard retrieval-augmented generation (RAG) retrieves once, using the question as the query, then answers [2]. Agentic RAG lets the model plan searches, judge the results and search again before answering [5, 6, 19].
- One retrieval pass falls short on multi-hop questions, where each lookup depends on an earlier one. Interleaving retrieval with reasoning raised retrieval by up to 21 points and answers by up to 15 on four such datasets [6].
- Retrieval quality still sets the ceiling. On the FRAMES benchmark, one commercial model rose from 0.41 accuracy without retrieval to 0.66 with its own repeated searches, against 0.73 with every needed article in the prompt [16].
- The loop costs time. Averaged over six datasets with one commercial model, multi-step retrieval scored 50.87 F1 against 46.99 for one retrieval step, at 3.33 times the time per query [13].
- Search agents misjudge when to search. One trained agent could have answered correctly without searching in 27.7% of its search steps, and more searches made another model worse at saying an answer was absent [23, 24].
- A small corpus can skip retrieval. A model developer advises putting a knowledge base under 200,000 tokens, about 500 pages, straight into the prompt [25]. Long contexts have their own failure: models use facts in the middle of the context less well than facts at the start or end [8].
- Agentic RAG is a choice made per task. A survey of agentic RAG, revised in April 2026, finds modular pipelines sufficient for simple fact retrieval and well-scoped queries, and advises bounded planning and explicit stopping criteria where agents run the retrieval [19].

Background

Retrieval-augmented generation since 2020

Lewis et al. introduced retrieval-augmented generation in May 2020, in a paper accepted at NeurIPS 2020. Their models "combine pre-trained parametric and non-parametric memory for language generation": a pre-trained sequence-to-sequence model, and a dense vector index of Wikipedia read through a pre-trained neural retriever [2]. The input is embedded, a maximum inner product search finds the top documents, and the generator writes from them. The index held the December 2018 Wikipedia dump in 100-word chunks, 21 million documents, and the models were trained with 5 or 10 retrieved documents [2]. In both of the paper's variants, retrieval starts from the input [2].

Surveys name the stages that followed. Gao et al. describe a progression from Naive RAG to Advanced RAG and Modular RAG [10]. Singh et al.'s survey of agentic RAG, first posted in January 2025 and in its fourth version since 1 April 2026, adds Graph RAG and Agentic RAG. It describes Naive RAG as a "simple retrieve-read workflow" with keyword retrieval such as TF-IDF and BM25, and Advanced RAG as adding dense retrieval, reranking and multi-hop retrieval mechanisms [19].

What makes RAG agentic

The survey starts from a limit: traditional RAG systems "are constrained by static workflows and lack the adaptability required for multi-step reasoning and complex task management" [19]. Agentic RAG embeds autonomous agents in the pipeline, and those agents use "reflection, planning, tool use, and multi-agent collaboration to dynamically manage retrieval strategies" [19]. The survey sorts these systems by agent cardinality, control structure, autonomy and knowledge representation [19].

A model developer's December 2024 engineering post draws the line in plainer terms. "Workflows are systems where LLMs and tools are orchestrated through predefined code paths", while agents "dynamically direct their own processes and tool usage" [26]. A fixed retrieve-then-answer pipeline is a workflow, and a model that chooses its next search from what it has read is an agent. The same developer's June 2025 post contrasts static retrieval, which fetches the chunks most similar to the input query, with "a multi-step search that dynamically finds relevant information, adapts to new findings, and analyzes results" [27].

Aspect	Standard RAG	Agentic RAG
Query	The question, once [2]	Written by the model at each step from what it has found [6, 16]
Retrieval calls	One per question [2, 7]	Several, until an answer or a step cap [5, 6]
Retrieved passages	Used as returned [9]	Graded, with a rewrite or a fallback when they miss [12, 29]

Aspect	Standard RAG	Agentic RAG
Whether to retrieve	Always [9]	Decided by the model or a router [9, 13]
Control flow	A predefined code path [26]	Directed by the model [26]

Table 1. Agentic RAG vs RAG, as the sources reviewed describe the two.

Graph RAG sits on a different axis. GraphRAG uses an LLM to derive an entity knowledge graph from the documents and to pregenerate summaries of related entities, for global questions such as "What are the main themes in the dataset?", on which its authors find that RAG fails [18]. Graph RAG changes the index, and agentic RAG changes who controls retrieval, so the two combine: the survey lists agentic RAG for graph RAG as a category of its own [19].

Method

Question. How does agentic RAG differ from standard RAG in design, measured results and cost, and when is a simpler design enough?

Sources. 29 sources, each opened on 8 November 2026: datasets and benchmarks [1, 3, 14, 16, 21], method papers [2, 4, 5, 6, 7, 9, 12, 13, 15, 17, 18, 20], studies of failures and efficiency [8, 11, 22, 23, 24], two surveys [10, 19], three engineering posts from one model developer [25, 26, 27], one cloud search service's documentation [28] and one agent framework's tutorial [29]. Venues were checked on their venue pages: the ACL Anthology for IRCoT and Lost in the Middle, and the ICLR 2024 proceedings for Self-RAG.

How. Figures come from each paper's abstract, tables or text. Models are described by size and kind. Figures from a vendor's own tests or worked examples are labelled. Numbers computed for this report are labelled.

What counts as a gain. No source reviewed gives one figure for how much agentic RAG improves on standard RAG across tasks, and the survey reports none [19]. Every gain below belongs to one method, one set of datasets and one model.

Findings

1. The single-pass pipeline and where it fails

Multi-hop questions expose the single pass. HotpotQA, from 2018, holds 113,000 Wikipedia-based question-answer pairs that "require finding and reasoning over multiple supporting documents to answer" [1]. MuSiQue's authors note that earlier multi-hop benchmarks are known to be "largely solvable via shortcuts", and built 25,000 questions of two to four hops in which "one reasoning step critically relies on information from another" [3]. A single-hop model scored 30 points lower in F1 on them than on existing datasets [3].

The IRCoT authors state the problem for retrieval: "what to retrieve depends on what has already been derived, which in turn may depend on what was previously retrieved" [6]. A query written from the question alone cannot ask for a fact the question does not name.

Models also fail to compose facts they hold. Press et al. measured a compositionality gap, how often a model answers every sub-question correctly and still misses the overall answer. In one family of models it stayed at a roughly constant 40% across sizes [4]. Their self-ask prompt has the model ask and answer its own follow-up questions first, a structure that "lets us easily plug in a search engine to answer the follow-up questions, which additionally improves accuracy" [4]. Self-ask with search is an early form of the agentic loop: break the question down, look up each part, then answer.

Engineering reports find the same gaps in built systems. Barnett et al. drew seven failure points from three case studies in research, education and biomedicine (2024) [11]:

Failure point	What happens
FP1 Missing content	The documents cannot answer the question
FP2 Missed the top ranked documents	The answer sits below the top K returned
FP3 Not in context	Retrieved, then lost when results were combined
FP4 Not extracted	Too much noise or contradicting information in the context
FP5 Wrong format	The answer ignores the requested format
FP6 Incorrect specificity	Too general or too specific
FP7 Incomplete	The answer leaves out information that was in the context

Table 2. Failure points of a single-pass RAG system, from [11].

For FP7 the paper gives its own remedy. For a question such as "What are the key points covered in documents A, B and C?", "A better approach is to ask these questions separately" [11]. That is the decomposition an agentic loop performs. FP1 is out of reach of any loop, because the answer is not in the corpus.

FLARE's authors sum up the baseline: most retrieval-augmented models use "a retrieve-and-generate setup that only retrieves information once based on the input" [7].

2. Interleaving retrieval with reasoning

ReAct, published at ICLR 2023, has the model alternate reasoning traces and actions [5]. On HotpotQA and FEVER, the actions were search, lookup and finish calls to a simple Wikipedia interface, which the authors describe as much weaker than leading lexical or neural retrievers. Runs were capped at 7 steps on HotpotQA and 5 on FEVER, because more steps did not improve results [5]. With a 540B-parameter model, ReAct alone scored 27.4 exact match on HotpotQA against 29.4 for chain-of-thought reasoning, and combining the two with self-consistency reached 35.1 [5]. On FEVER, ReAct beat chain of thought, 60.9 against 56.3 [5].

The authors labelled sampled HotpotQA trajectories by hand. Among ReAct's failures, 47% were reasoning errors, 23% search result errors and none hallucinations [5]. Among chain-of-thought failures, 56% were hallucinations [5]. "Non-informative search, which counts for 23% of the error cases, derails the model reasoning and gives it a hard time to recover and reformulate thoughts" [5]. The model also at times "repetitively generates the previous thoughts and actions" [5]. The loop traded hallucination for dependence on what each search returned.

IRCoT, published at ACL 2023, makes the reasoning itself the query [6]. It starts with the question as the query, then alternates two steps: extend the chain of thought by one sentence, and retrieve with that sentence as the next query. It stops when the sentence contains "answer is:" or after 8 steps [6]. The retriever was BM25 keyword search throughout, so the gains came from better queries [6]. Across HotpotQA, 2WikiMultihopQA, MuSiQue and IIRC, IRCoT "substantially improves retrieval (up to 21 points) as well as downstream QA (up to 15 points)", without additional training [6].

Dataset	11B open model	175B-parameter model
HotpotQA	9.4	7.1
2WikiMultihopQA	15.3	13.2
MuSiQue	5.0	7.1
IIRC	2.5	No gain

Table 3. Gain in answer F1 points from interleaved retrieval over one-step retrieval, from IRCoT's Figure 4 [6].

A 3B model with interleaved retrieval outperformed one-step retrieval with a model 58 times larger [6]. On 40 questions each, IRCoT reduced factual errors by 50% on HotpotQA and 40% on 2WikiMultihopQA [6].

FRAMES, published at NAACL 2025, tests 824 multi-hop questions that require information from multiple sources [16].

Setup (one commercial long-context model)	Accuracy
No retrieval	0.408
BM25, top 2 articles	0.452
BM25, top 4 articles	0.474
Multi-step: 5 steps, 5 queries a step, 10 documents	0.66
Oracle: every gold article in the prompt	0.729

Table 4. FRAMES results, from its Table 3 and section 3.2 [16].

In the multi-step setup the model wrote search queries, BM25 returned articles that were appended to the context, and after the last step the model answered [16]. With plain instructions, accuracy "still remains quite low even after five iterations", because of "the lack of diversity in the queries generated by the model": it "goes in the wrong direction in search retrievals and never corrects itself" [16]. With example query sequences and an instruction not to repeat queries, accuracy reached 0.66, close to the oracle's 0.73 [16].

Since 2025 the search behaviour itself is trained. Search-R1's authors found prompting reasoning models to use search engines "often suboptimal", and trained a model with a simple outcome-based reward to write multiple search queries while it reasons [20]. On seven question-answering datasets it improved performance over RAG baselines by 41% for a 7B open model and 20% for a 3B open model [20].

A search tool alone does not make an agent that searches well. On BrowseComp, 1,266 questions that require "persistently navigating the internet in search of hard-to-find, entangled information", a chat model's accuracy rose from 0.6% to 1.9% with browsing, and a research agent trained for this kind of task reached 51.5% [21].

3. Deciding when to retrieve and grading what comes back

A loop can also decide whether to retrieve at all, and check what retrieval returned.

Retrieval on low confidence. FLARE, published at EMNLP 2023, decides "when and what to retrieve across the course of the generation" [7]. It predicts the upcoming sentence and, if it "contains low-confidence tokens", uses it as a query and regenerates the sentence from what it finds. On four long-form generation tasks it matched or beat the baselines [7].

Retrieval trained into the model. Self-RAG, published at ICLR 2024, argues that retrieving a fixed number of passages "regardless of whether retrieval is necessary, or passages are relevant" can lead to unhelpful answers [9]. It trains one model, at 7B and 13B parameters, to retrieve on demand and to emit reflection tokens [9].

Reflection token	Values	What it judges
Retrieve	yes, no, continue	Whether to retrieve
IsRel	relevant, irrelevant	Whether a passage provides useful information
IsSup	fully supported, partially supported, no support	Whether the statement is supported by the passage
IsUse	5 to 1	How useful the response is

Table 5. Self-RAG's reflection tokens, from its Table 1 [9].

Retrieval fires when the probability of Retrieve=Yes passes a threshold, 0.2 for most tasks and 0 for a benchmark that requires citations [9]. As the threshold moves, retrieval frequency changes dramatically, and the cost of retrieving less depends on the task: small on a health claims dataset, larger on an open-domain question set [9]. The paper reports significant gains in factuality and citation accuracy for long-form answers against two chat models [9].

A separate grader. Corrective RAG (CRAG), a 2024 preprint, uses "a lightweight retrieval evaluator", a small fine-tuned model, to score the retrieved documents for a query [12]. Above an upper threshold the action is Correct, below a lower one Incorrect, and between them Ambiguous. On Incorrect, the documents are discarded and the system turns to web search, and a "decompose-then-recompose" step filters what it keeps [12].

Routing by complexity. Adaptive-RAG, published at NAACL 2024, finds that existing approaches "either handle simple queries with unnecessary computational overhead or fail to adequately address complex multi-step queries" [13]. A smaller classifier model sends each question to no retrieval, single-step retrieval or iterative retrieval. All methods used the same BM25 retriever, on 500 test questions from each of three single-hop and three multi-hop datasets [13].

Strategy (commercial instruction-tuned model)	F1	Retrieval steps	Time per query, relative to one step
No retrieval	48.56	0	0.71
Single-step	46.99	1	1.00
Adaptive-RAG	50.91	1.03	1.46
Multi-step	50.87	2.81	3.33

Table 6. Averages over six datasets, from Adaptive-RAG's Table 1 [13].

With an open 3B model, single-step retrieval scored 44.31 F1 and multi-step 48.85, at 8.81 times the time [13]. With the commercial model, the router scored 50.91 F1, level with multi-step retrieval's 50.87, at about 44% of multi-step's time (computed for this report from Table 6). With that model, no retrieval scored above single-step retrieval, 48.56 against 46.99 [13]. How much accuracy routing keeps depends on how well the classifier sorts the questions.

Rewrite and retry. An agent framework's tutorial builds the pattern as a graph. The model answers directly or calls the retriever tool, and "That choice is what makes the system agentic rather than a fixed retrieve-then-generate pipeline" [29]. A grader then checks the documents and either moves on to the answer or rewrites the question and tries again, so the agent "recovers from a weak first retrieval instead of stopping or hallucinating an answer" [29].

4. Planning several searches at once

Some questions split into parts that can be searched at the same time. A model developer's June 2025 post describes an orchestrator-worker design in which "a lead agent coordinates the process while delegating to specialized subagents that operate in parallel" [27]. The lead agent starts 3 to 5 subagents, each using 3 or more tools in parallel, and the developer reports that this cut research time by up to 90% for complex queries (vendor-reported) [27]. Its prompts scale effort to the question: 1 agent with 3 to 10 tool calls for simple fact-finding, 2 to 4 subagents with 10 to 15 calls each for direct comparisons, and more than 10 subagents for complex research [27].

One cloud search service's documentation, last updated 17 September 2026, offers a form of this as a setting: LLM-based query planning that can "break down a complex query into smaller, focused subqueries", runs them in parallel and reranks each one's results [28]. Reasoning effort defaults to low, and at minimal effort the planning step is skipped. As of that page, query planning is part of a preview version of the service's API [28].

Our explainer on when a multi-agent system beats a single agent (https://rillor.com/insights/single-agent-vs-multi-agent-systems) covers the controlled studies of when splitting work helps and when it hurts.

5. Retrieval quality still sets the ceiling

The survey is direct: "Agentic reasoning cannot compensate for consistently poor retrieval", and "Investing in robust retrieval pipelines and high-quality indexing before introducing agentic complexity remains essential" [19]. The measured results agree. On FRAMES, the tuned loop reached 0.66 against 0.73 with perfect retrieval [16]. In IRCoT, retrieval on IIRC improved by 21 points while answer F1 with the 175B-parameter model did not improve [6].

Retrieval can be improved before any loop is added. A model developer's September 2024 post describes contextual retrieval: before each chunk is embedded and added to a BM25 index, a short chunk-specific context, "usually 50-100 tokens", is prepended to it [25]. BM25 helps with "unique identifiers or technical terms" [25]. On the developer's own tests, measured as 1 minus recall@20, the share of relevant chunks missing from the top 20 results:

Method (developer's own tests)	Top-20 retrieval failure rate	Reduction
Embeddings	5.7%	
Contextual embeddings	3.7%	35%
Contextual embeddings and contextual BM25	2.9%	49%
Both, plus reranking	1.9%	67%

Table 7. Retrieval failure rates reported in [25].

More passages are not free. In an open-domain case study, Liu et al. found that "reader model performance saturates long before retriever performance saturates". More than 20 retrieved documents improved accuracy by only 1.5% and 1% for two commercial chat models, while raising input length, latency and cost, and the authors suggest reranking or retrieving fewer documents [8]. Jin et al. (2024) found that for many long-context models output quality "initially improves first, but then subsequently declines as the number of retrieved passages increases", which they traced to near-miss "hard negatives" [17]. Reordering the retrieved passages helped without training [17]. A loop that keeps adding passages can lose accuracy the same way.

6. Long context as an alternative

When the corpus is small, retrieval can be skipped. The contextual retrieval post advises: "If your knowledge base is smaller than 200,000 tokens (about 500 pages of material), you can just include the entire knowledge base in the prompt that you give the model, with no need for RAG or similar methods" [25].

Long contexts have their own failure. Liu et al. found a "U-shaped performance curve": accuracy was higher with the relevant information at the start or end of the context, and it "significantly degrades when models must access relevant information in the middle of long contexts, even for explicitly long-context models" [8]. For one commercial chat model, accuracy with the relevant passage in the middle fell below its closed-book accuracy with no documents, 56.1% [8]. RULER, published at COLM 2024, tested 17 long-context models on 13 tasks [14]. Despite near-perfect scores on a plain needle-in-a-haystack test, "almost all models exhibit large performance drops as the context length increases". All claimed 32K tokens or more, and only half maintained satisfactory performance at 32K [14]. These findings describe 2024 models. Long contexts also take GPU memory, because the KV cache grows with context, as our explainer on how much GPU memory LLM inference needs (https://rillor.com/insights/gpu-memory-for-llm-inference) sets out.

Li et al., in a study presented at EMNLP 2024, compared the two directly. "When resourced sufficiently, LC consistently outperforms RAG in terms of average performance. However, RAG's significantly lower cost remains a distinct advantage" [15]. The two gave identical predictions for over 60% of queries [15]. Their Self-Route method gives the model the retrieved chunks first and lets it answer "unanswerable", and only those queries go to the full context [15]. One model scored 46.83 against 47.04 for long context while using 61% of the tokens, and cost fell by 65% and 39% for two commercial long-context models [15]. The authors suggest that bringing chain-of-thought reasoning into RAG may help with the multi-step questions RAG missed [15], the case the agentic loop targets.

7. Cost, latency and stopping rules

Model calls. Each round is at least one more model call. IRCoT's gains "come with an additional computational cost", because it "makes a separate call to an (L)LM for each sentence of CoT" [6]. FRAMES's multi-step setup "requires six non-parallelizable inference calls (five for retrieval + one for final answering)" [16]. In Adaptive-RAG's averages, multi-step retrieval took 3.33 times as long per query as one step with a commercial model, and 8.81 times with an open 3B model [13].

Latency. Yang et al. (2025) found that the interleaved paradigm "introduces substantial efficiency bottlenecks": exact search adds retrieval time, coarse retrieval adds reasoning steps, and small retrieval delays cascade into end-to-end latency [22]. Their system, SearchAgent-X, reports up to 3.4 times higher throughput and 5 times lower latency against standard serving and search setups [22].

Tokens. The research-system developer reports that agents "typically use about 4x more tokens than chat interactions, and multi-agent systems use about 15x more tokens than chats" (vendor-reported) [27]. In its BrowseComp analysis, token usage alone explained 80% of the variance in performance [27].

Money. The cloud search documentation works through a cost example at assumed prices (a hypothetical reranking price and one model's token rates), noting that real costs could be lower: 2,000 agentic retrievals, with three subqueries per query plan and 50 chunks reranked per subquery, cost about $3.30 for reranking and $1.02 for query-planning tokens, $4.32 in all [28].

Step caps. No source reviewed gives a general rule for how many steps a loop should take. The papers set their own caps: 7 and 5 steps in ReAct, 8 in IRCoT and five rounds in FRAMES [5, 6, 16].

Over-search and under-search. Wu et al., at EMNLP 2025, name "over-search (retrieving redundant information) and under-search (failing to retrieve necessary information)" [23]. Two trained search agents could have answered correctly without searching in 20.2% and 27.7% of their search steps [23]. In steps where they did not search, the same two agents were wrong 63% and 34% of the time [23].

Xie et al., at EACL 2026, found that search "generally improves answer accuracy on answerable queries but harms abstention on unanswerable ones" [24]. Over-searching was more pronounced in complex reasoning models and deep research systems, worse with noisy retrieval, and it compounded across turns. Negative evidence in the results improved abstention [24]. For one commercial reasoning model, as its limit on search turns rose from 0 to 19, answer accuracy improved sharply from no search to one, peaked around 7 searches and then plateaued, while abstention accuracy fell and tokens per correct answer kept rising [24].

Unbounded effort. The research-system developer lists early failures such as "spawning 50 subagents for simple queries, scouring the web endlessly for nonexistent sources", and wrote effort rules into its prompts [27]. The survey warns that "Unrestricted autonomy risks excessive tool invocation, redundant reasoning loops, or misaligned objectives", and finds that effective systems use "bounded planning horizons, predefined tool access policies, and explicit stopping criteria" [19].

Citations. A loop that retrieves more has more citations to check. Self-RAG's IsSup token judges whether each statement is supported by its passage [9], and the research-system developer's judge rubric includes "citation accuracy (do the cited sources match the claims?)" [27]. Our note on how to check LLM citations against their sources (https://rillor.com/insights/checking-llm-citations-against-sources) sets out a claim-by-claim method.

Implications

For teams with a RAG prototype that misses multi-part questions. Check retrieval before adding a loop. Measure whether the needed passages reach the top results, combine keyword search with embeddings and rerank [25], and confirm that the answer is in the corpus at all, since no loop fixes missing content [11, 19]. If the corpus is under about 200,000 tokens, test the whole corpus in the prompt [25], and test where the key facts sit [8].

When the questions are multi-hop. Interleaved retrieval is the measured response, with gains on four multi-hop datasets [6] and on FRAMES [16]. Instructions on query diversity mattered [16], and trained search behaviour outperformed RAG baselines in Search-R1's tests [20].

For cost and evaluation. Route simple questions away from the loop [13, 15], cap the steps [5, 6, 16], and track time per query [13] and tokens per correct answer [24]. Score intermediate decisions as well as answers: the survey notes that "Existing benchmarks focus on output quality with limited visibility into intermediate decisions" [19], and over-search, under-search and abstention can be measured [23, 24].

Situation	Starting design	Sources
Corpus under about 200,000 tokens	The whole corpus in the prompt, with a test of where key facts sit	[8, 25]
Simple fact lookups and well-scoped queries	Single-pass RAG with keyword and embedding search and reranking	[19, 25]
Multi-hop questions over a large corpus	Interleaved retrieval with a step cap	[6, 16]
A mix of simple and complex questions	A router by complexity, or by whether the retrieved chunks suffice	[13, 15]
Parts that can be searched separately	Parallel subqueries or subagents	[27, 28]
The answer may be absent from the corpus	A test of abstention and a limit on searches	[11, 24]

Table 8. Starting designs by situation, drawn from the sources cited.

Where Rillor works. Rillor builds research and analysis agents, in which a planner breaks a question into parts and gives each part to a specialist agent, and retrieval with checked citations, in which each citation in an answer must match a passage the system actually retrieved, or the claim is removed.

Limits of this analysis

- This is an explainer of published methods and results. It ranks no products and reports no pooled figure for agentic RAG over standard RAG, because no source reviewed gives one.
- Most results come from public question-answering benchmarks [1, 3, 16, 21], and the survey reads gains on open-domain tasks as more modest [19].
- Several results come from models released between 2022 and 2024, and RULER's findings describe 2024 models [14].
- Figures from the model developer and the cloud search service are their own tests or worked examples [25, 27, 28].
- CRAG, Search-R1, SearchAgent-X and Jin et al.'s long-context study list no venue on arXiv and are cited as preprints [12, 17, 20, 22].
- IRCoT's abstract gives retrieval gains of up to 21 points, and its section 5 reports a larger figure for one dataset. This report uses the abstract's figure [6].
- Access control for retrieval, which user may see which document, is outside the scope of this report.

Sources

1. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov and Christopher D. Manning, arXiv:1809.09600, 25 September 2018 (EMNLP 2018), viewed 8 November 2026. https://arxiv.org/abs/1809.09600

2. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni and others, arXiv:2005.11401, version 4, 12 April 2021 (NeurIPS 2020), viewed 8 November 2026. https://arxiv.org/abs/2005.11401

3. MuSiQue: Multihop Questions via Single-hop Question Composition. Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot and Ashish Sabharwal, arXiv:2108.00573, version 3, 5 May 2022 (TACL 2022), viewed 8 November 2026. https://arxiv.org/abs/2108.00573

4. Measuring and Narrowing the Compositionality Gap in Language Models. Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith and Mike Lewis, arXiv:2210.03350, version 3, 17 October 2023 (Findings of EMNLP 2023), viewed 8 November 2026. https://arxiv.org/abs/2210.03350

5. ReAct: Synergizing Reasoning and Acting in Language Models. Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan and Yuan Cao, arXiv:2210.03629, version 3, 10 March 2023 (ICLR 2023), viewed 8 November 2026. https://arxiv.org/abs/2210.03629

6. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot and Ashish Sabharwal, arXiv:2212.10509, version 2, 23 June 2023 (ACL 2023), viewed 8 November 2026. https://arxiv.org/abs/2212.10509

7. Active Retrieval Augmented Generation. Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan and Graham Neubig, arXiv:2305.06983, version 2, 22 October 2023 (EMNLP 2023), viewed 8 November 2026. https://arxiv.org/abs/2305.06983

8. Lost in the Middle: How Language Models Use Long Contexts. Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni and Percy Liang, Transactions of the Association for Computational Linguistics, Volume 12, 2024. arXiv:2307.03172, viewed 8 November 2026. https://arxiv.org/abs/2307.03172

9. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil and Hannaneh Hajishirzi, arXiv:2310.11511, 17 October 2023 (ICLR 2024), viewed 8 November 2026. https://arxiv.org/abs/2310.11511

10. Retrieval-Augmented Generation for Large Language Models: A Survey. Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia and others, arXiv:2312.10997, version 5, 27 March 2024, viewed 8 November 2026. https://arxiv.org/abs/2312.10997

11. Seven Failure Points When Engineering a Retrieval Augmented Generation System. Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly and Mohamed Abdelrazek, arXiv:2401.05856, 11 January 2024 (CAIN 2024), viewed 8 November 2026. https://arxiv.org/abs/2401.05856

12. Corrective Retrieval Augmented Generation. Shi-Qi Yan, Jia-Chen Gu, Yun Zhu and Zhen-Hua Ling, arXiv:2401.15884,

version 3, 7 October 2024, viewed 8 November 2026. https://arxiv.org/abs/2401.15884

13. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang and Jong C. Park, arXiv:2403.14403, version 2, 28 March 2024 (NAACL 2024), viewed 8 November 2026. https://arxiv.org/abs/2403.14403

14. RULER: What's the Real Context Size of Your Long-Context Language Models? Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang and Boris Ginsburg, arXiv:2404.06654, version 3, 6 August 2024 (COLM 2024), viewed 8 November 2026. https://arxiv.org/abs/2404.06654

15. Retrieval Augmented Generation or Long-Context LLMs? A Comprehensive Study and Hybrid Approach. Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei and Michael Bendersky, arXiv:2407.16833, version 2, 17 October 2024 (EMNLP 2024 industry track), viewed 8 November 2026. https://arxiv.org/abs/2407.16833

16. Fact, Fetch, and Reason: A Unified Evaluation of Retrieval-Augmented Generation. Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay and Manaal Faruqui, arXiv:2409.12941, version 3, 24 January 2025 (NAACL 2025), viewed 8 November 2026. https://arxiv.org/abs/2409.12941

17. Long-Context LLMs Meet RAG: Overcoming Challenges for Long Inputs in RAG. Bowen Jin, Jinsung Yoon, Jiawei Han and Sercan O. Arik, arXiv:2410.05983, 8 October 2024, viewed 8 November 2026. https://arxiv.org/abs/2410.05983

18. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley and others, arXiv:2404.16130, version 2, 19 February 2025, viewed 8 November 2026. https://arxiv.org/abs/2404.16130

19. Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG. Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei and Athanasios V. Vasilakos, arXiv:2501.09136, version 4, 1 April 2026 (first version 15 January 2025), viewed 8 November 2026. https://arxiv.org/abs/2501.09136

20. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani and Jiawei Han, arXiv:2503.09516, version 5, 5 August 2025, viewed 8 November 2026. https://arxiv.org/abs/2503.09516

21. BrowseComp: A Simple Yet Challenging Benchmark for Browsing Agents. Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney and others, arXiv:2504.12516, 16 April 2025, viewed 8 November 2026. https://arxiv.org/abs/2504.12516

22. Demystifying and Enhancing the Efficiency of Large Language Model Based Search Agents. Tiannuo Yang, Zebin Yao, Bowen Jin, Lixiao Cui, Yusen Li, Gang Wang and Xiaoguang Liu, arXiv:2505.12065, 17 May 2025, viewed 8 November 2026. https://arxiv.org/abs/2505.12065

23. Search Wisely: Mitigating Sub-optimal Agentic Searches By Reducing Uncertainty. Peilin Wu, Mian Zhang, Xinlu Zhang, Xinya Du and Zhiyu Zoey Chen, arXiv:2505.17281, version 2, 9 October 2025 (EMNLP 2025), viewed 8 November 2026. https://arxiv.org/abs/2505.17281

24. Over-Searching in Search-Augmented Large Language Models. Roy Xie, Deepak Gopinath, David Qiu, Dong Lin, Haitian Sun, Saloni Potdar and Bhuwan Dhingra, arXiv:2601.05503, version 2, 11 March 2026 (EACL 2026), viewed 8 November 2026. https://arxiv.org/abs/2601.05503

25. Introducing Contextual Retrieval. Anthropic, 19 September 2024, viewed 8 November 2026. https://www.anthropic.com/engineering/contextual-retrieval

26. Building effective agents. Anthropic, 19 December 2024, viewed 8 November 2026. https://www.anthropic.com/engineering/building-effective-agents

27. How we built our multi-agent research system. Anthropic, 13 June 2025, viewed 8 November 2026. https://www.anthropic.com/engineering/multi-agent-research-system

28. Agentic retrieval in Azure AI Search. Microsoft Learn, last updated 17 September 2026, viewed 8 November 2026. https://learn.microsoft.com/en-us/azure/search/agentic-retrieval-overview

29. Build a custom RAG agent with LangGraph. LangChain documentation, viewed 8 November 2026. https://docs.langchain.com/oss/python/langgraph/agentic-rag

About this report

Published by Rillor on 11 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. Agentic RAG and how it differs from standard RAG. 11 November 2026. https://rillor.com/insights/agentic-rag-vs-rag

Notices

Superintelligence (SI) outputs can be wrong. Important decisions should include human review.

Work with Rillor

Tell us the workflow, the sources it draws on and the decisions it supports, and we scope the agents, tools and tests around them. Contact us at rillor.com/contact?type=agentic-systems or contact@rillor.com. More on agentic systems: rillor.com/si/agentic-systems