A Survey of Retrieval-Augmented Generation: From Searchable Sources to Supported Answers

Abstract

Retrieval-augmented generation (RAG) lets a large language model answer from an external collection instead of relying only on what it learned during training. Information passes from sources through an index, retrieval, context construction and generation; a controller may repeat the search. A relevant passage can omit a needed fact, and a generator can ignore a fact that reaches its input. Even a supported answer can be wrong when its source is outdated. We survey these stages across text, graph and multimodal sources, comparing methods by what they keep, discard or add. Small, self-contained units suit fact lookup, while longer units preserve connections across a document. Further searches help when earlier results reveal a missing fact. Checking support for each required step reduces searches while preserving accuracy in most tested settings. Compression saves input tokens but can sever the link between a claim and its original source. When sources disagree, repetition and the model's memory can outweigh reliable evidence. Because these failures arise at different stages, evaluation must measure relevance, sufficiency, support and correctness separately, intervene on individual stages to locate failures, and count preparation as well as query costs. Keeping derived indexes consistent with changing sources, making retrieval decisions reliable and reproducing evaluations remain open problems.

1 Introduction

Large language models (LLMs) learn to generate text by predicting *tokens*, units such as words or word fragments, in large training collections. Training stores patterns of language and factual associations in the model's numerical parameters. Given a question or an instruction, the same model can then explain a concept, summarize supplied text or compose an answer, without a separate model trained for each task. This ability makes language models useful as conversational assistants and interfaces to specialized information [7]. For factual questions, generation draws on both learned knowledge and whatever information the input supplies. The learned knowledge is not a searchable copy of the training collection: the model can state a fact without identifying where it learned it [8]. It can also produce a plausible falsehood, and its stored knowledge stays fixed between training updates. Answering questions about changing events or private documents therefore requires information beyond the model's parameters.

External retrieval supplies that information without retraining the model. A searchable collection holds documents that can be updated or replaced, a retriever selects relevant passages, and the language model reads them before answering [8]. The answer can then be checked against identifiable sources, making this approach useful for search assistants and question answering over company documents [7]. For "Which country has the largest population?", a 2023 report of United Nations estimates supplies India where an older model may recall China. Searching once with the question works when the retrieved passages contain what the answer needs. A harder variant, "Who wrote the national anthem of the most populous country?", may retrieve population reports but nothing about anthem authors. If the country is initially unknown, the system must identify India before it can search for the author. Interleaving retrieval with reasoning helps gather evidence for such dependent steps [3].

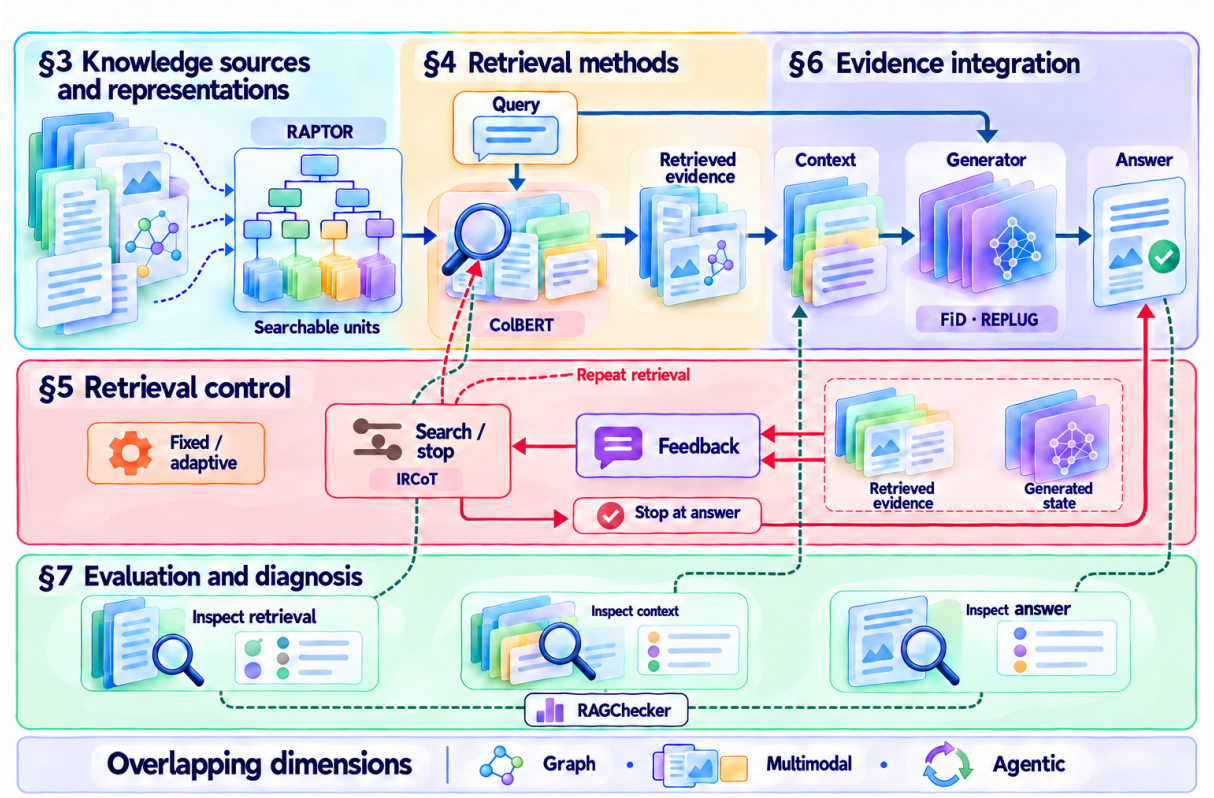


Figure 1: RAG separates source indexing (§3), evidence retrieval (§4) and evidence use (§6), with a controller that can repeat the search (§5). Evaluation checks retrieval, context and answers separately (§7). Graph, multimodal and agentic designs cross these stages rather than replacing the pipeline [1–6].

Retrieval and generation can be combined at different points. Some systems train the retriever and generator together [8, 9]; others train a generator to use a fixed retriever [10] or add retrieved text to an already trained model's input [11]. The model can also choose later searches from earlier results [3, 12]. We call this family *retrieval-augmented generation* (RAG): a language model generates output using material retrieved from an external collection during inference. RAG extends retrieve-then-read question answering [13] from extracting an answer span to generating an answer that combines several passages [4]. Giving the model control of search produces the agentic branch of RAG [14]. Enlarging its input window offers a different route to external information: a long-context model can read a whole supplied document, while retrieval selects evidence from collections too large to read in full [15, 16].

Retrieval adds two difficulties before the model begins writing. (i) *Indexing decides what can be found.* Splitting or rewriting sources can cut a fact off from its context: "it overtook China in 2023" leaves the country unnamed [17, 18]. (ii) *Relevant passages can leave a required step unanswered.* Population reports do not name anthem authors, so the system must search again using what it has learned [3, 19]. Two further difficulties extend the language model's existing problem of generating reliable text. (iii) *The model can ignore or misread its input.* Retrieval adds distractions and conflicting sources, and the model may follow repetition or its memory instead of the evidence [20–22]. (iv) *Support and correctness can diverge.* An outdated source can justify a false answer, while the model can supply a true answer from memory without support [6, 19]. We therefore follow the path from searchable sources (§3), through query transformation, matching and selection (§4), repeated search (§5) and evidence use (§6), to evaluation that distinguishes these failures and counts their cost (§7; Figure 1).

Earlier surveys already compare RAG components. Gao et al. organize methods around retrieval, generation and augmentation [23], while a broader account also compares training

strategies and applications [7]. Specialized surveys examine evaluation [24], graph RAG [25] and agentic RAG [14]. Component categories help locate a method, but an architectural label alone does not explain why its answers improve. A gain can come from finding different passages, preserving more of the same passages in the context, or changing how the generator uses them. An iterative system can also compensate for weak initial retrieval through later searches. Distinguishing these causes requires tracing what reaches the generator and comparing systems under matched conditions. Our contributions are:

- A formulation that separates retrieved evidence, constructed context and generated output, with four criteria (relevance, sufficiency, support and correctness) that distinguish evidence quality from answer quality (§2).
- A comparison across text, graph and multimodal sources that states what methods keep, discard or add, and separates evidence about whole pipelines from comparisons that isolate one change.
- An account of evaluation that measures each criterion separately, uses interventions to locate failures and counts the full cost of a system (§7), together with the open problems that these distinctions expose (§8).

2 Foundations

RAG combines finding material in an external collection with generating an answer from it. Retrieved material may be shortened or reorganized before generation, and an adaptive system can repeat the search. We separate these stages by what each receives and produces (§2.1), then trace the field's development into the design choices compared in the body sections (§2.2).

2.1 Retrieval-augmented generation

A language model can answer a query q directly from its learned parameters. Writing G for the generator and y for its output, generation without external evidence is:

$$y \sim G(\cdot \mid q). \tag{1}$$

RAG supplies additional information from an external collection of text documents, knowledge-graph records, page images, or a mix of these. The system builds a searchable index over units such as passages, sentences or graph paths. A text unit is often called a *chunk*. Index construction may rewrite, summarize or link the units and encode them for matching. An *embedding* represents a unit or query as a numerical vector, allowing retrieval to compare their content. When a question arrives, a retriever searches the index, and the system places what it finds in the generator's input. We call what the retriever returns *evidence*: material offered for answering, which may or may not be true.

Retrieved evidence can guide the generator only if it reaches its input, so we separate retrieval from context construction. Given the query and an index I, the retriever R returns ranked evidence E (2a). The context constructor A orders, truncates, compresses or formats that evidence into the context C (2b). The generator G then produces the output y from the query and context (2c):

$$E = R(q, I), \tag{2a}$$
$$C = A(q, E), \tag{2b}$$
$$y \sim G(\cdot \mid q, C). \tag{2c}$$

Compared with Eq. (1), the generator now receives an explicit context C in addition to the query. Retrieval and context construction determine what that context contains. The output

may be a short answer, a longer text with citations, or a continuation. The context is not always a single prompt: some generators encode each passage separately and attend to all of them while decoding, and others combine the predictions they make from each passage (§6.2). Evidence that retrieval found can be truncated before it reaches the generator. Separating E from C reveals both that loss and the effects of presenting the same evidence differently (§7.3).

Standard RAG follows this path once: it searches with the question, keeps the top k units as evidence and answers. An *adaptive* system lets a controller choose whether to search, what query to issue and when to answer. The controller reads the question and any previous queries, returned evidence or intermediate text. Suppose the model initially knows neither the most populous country nor its anthem's author. A first search returns a current population report naming India. The controller then asks who wrote India's national anthem, retrieves a passage naming "Jana Gana Mana" and Rabindranath Tagore, and answers from both sources. Such questions, whose answers require connecting several facts, are called *multi-hop* questions. A model that already knows the country can search directly for the author; one that knows both facts can answer without searching. The controller may follow fixed rules, use a prompted language model, or learn a policy from training (§5).

A system can succeed at one stage of this path and fail at the next, and four criteria tell the stages apart. Evidence is *relevant* when it bears on the question, as the population reports bear on the anthem question. Relevant evidence can still lack a fact the answer needs, so a context is *sufficient* only if it contains enough information to give a definite answer, whether or not the generator then uses it [19]. A context containing only population reports is insufficient, since none mentions an anthem. Even a sufficient context can be ignored or misread, so an output is *supported* when the context justifies its claims, and *correct* when its claims are true.

Support and correctness separate in both directions. An old report naming China, paired with a passage about China's anthem, supports an answer about the wrong country's author. A model that answers "Tagore" from memory after reading only population reports is correct but unsupported. Across the tested settings, models answer correctly in 35–62% of cases with insufficient context [19]. The four criteria concern different objects in Eq. (2): relevance concerns the evidence E, sufficiency the context C, support the output y relative to C, and correctness the claims in y. Evaluation must therefore check the evidence, the constructed context and the answer separately (§7.2).

2.2 System design dimensions

Early work explored answer-supervised retrieval [13], generation from several retrieved passages [4, 8] and learning retrieval during language-model pretraining [26]. Later pretraining used much larger retrieval collections [10], while few-shot answering jointly trained retrieval and generation [9]. From about 2022, four difficulties drew work toward different parts of the pipeline (Figure 2). Fixed-length passages can separate a fact from its context, so indexes began storing self-contained facts, longer documents and explicit connections through summary trees or graphs; page-image indexes also preserved information lost in text extraction (§3; [1, 17, 27–29]). A question's wording can miss relevant passages, and individually relevant passages can repeat the same information, motivating query rewriting and selection of complementary evidence (§4; [30–33]). Dependent questions require facts that the initial query cannot name, so controllers interleaved search with reasoning and later learned when and how to search (§5; [3, 12, 34–37]). Finally, hosted models made retrieval useful without retraining the generator, through prompts or per-passage predictions. Long, conflicting inputs and the need to check generated claims then motivated compression, conflict handling and source attribution (§6; [5, 11, 38–42]).

Two further choices affect the whole pipeline. (i) *Training.* A retriever can learn to find different evidence, a controller to choose different searches, and a generator to use the context differently. The components trained determine which parts of a system must be open to updates.

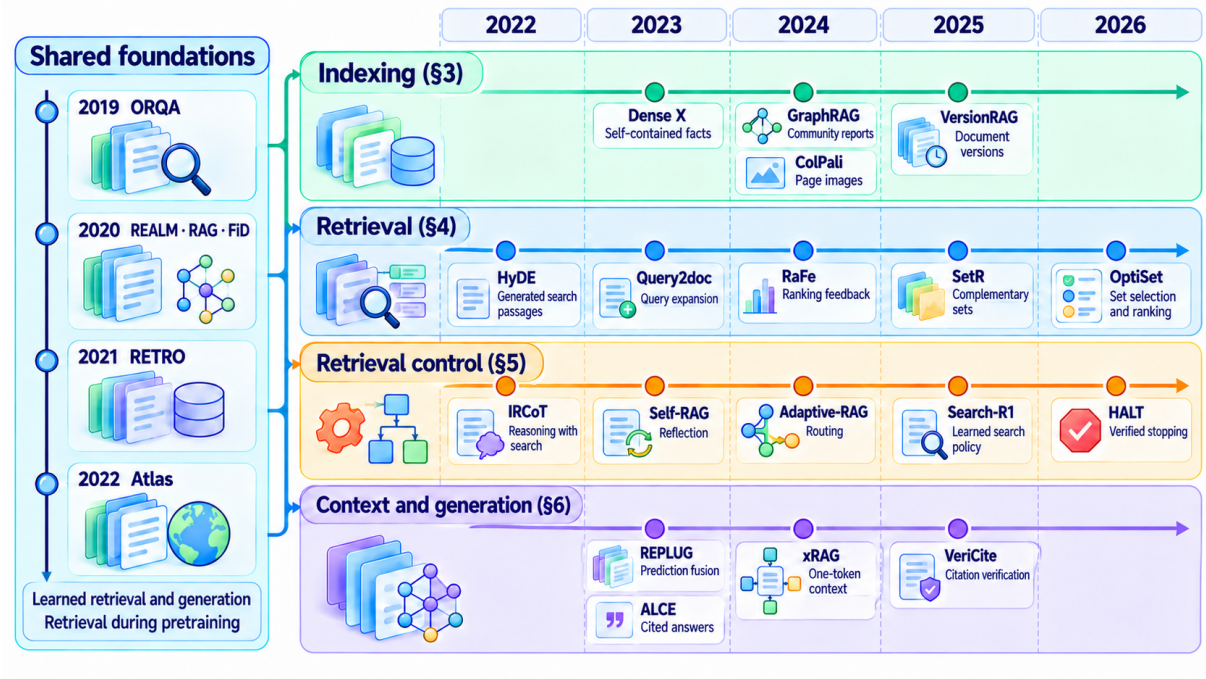


Figure 2: RAG developed from learned retrieval and generation into four parallel lines: indexing (§3), retrieval (§4), retrieval control (§5), and context construction and generation (§6) [3–5, 8–10, 12, 13, 17, 26, 28–31, 33, 36–39, 42–46].

(ii) *Access.* A text interface permits prompt changes but hides token probabilities, internal representations and attention. Some retrieval triggers, fusion methods and attribution methods require that access (§5.1, §6.2, §6.4). A remote search service can likewise hide how it ranks results and updates its index. A method's access requirements determine whether it can be added to a hosted model or needs a model the developer can inspect or train.

Graph, multimodal and agentic designs each change several of these choices at once. Graph structure can change indexing, matching and retrieval control [47]. Images change source representation and the generator's input [48], while agentic control changes search sequences and the reasoning between searches [49]. A multimodal system, for instance, still has to choose how to match and rerank candidates [50].

3 Knowledge sources and representations

Indexing determines which parts of a source a retriever can find and what context accompanies them. A sentence-sized unit beginning "It overtook China" loses the country's name; a transcription of a population chart may keep its labels but lose the crossing curves. A retriever cannot recover facts absent from its index. We compare three ways of preserving them: text units with surrounding context or metadata (§3.1), graphs that record connections explicitly (§3.2), and multimodal indexes that keep visual content or align different media (§3.3). Table 1 shows what each index stores and what remains available to the generator.

3.1 Documents and text indexes

A text index splits documents into units, and each unit has two jobs. It must match the questions it can answer, and it must make sense when read on its own. Small units match precisely but lose the text around them; large units keep that text but give the generator more to read. Most methods therefore try to give a unit the context it lacks without simply making it larger.

Table 1: Retrieval units, preserved context and construction requirements across text, graph and multimodal indexes.

Work	Source	Retrieval unit	Context or structure encoded	Offline construction	Generator input, if evaluated
Dense X [17]	Text	Proposition	Atomic fact with resolved references	LLM rewriting into propositions; unit embeddings	Propositions
LongRAG [27]	Text	Document or document group	Whole-document and linked-document context	Grouping; each unit scored by its best chunk	Concatenated long units
Late Chunking [18]	Text	Text chunk	Surrounding text in chunk embeddings	Long-context encoding, then chunk pooling	Not evaluated
RAPTOR [1]	Text	Chunk or summary node	Semantic clusters at multiple levels	Recursive clustering, summaries, embeddings	Selected node texts
GraphRAG [28]	Text	Community report	Extracted entity relations and community hierarchy	Graph extraction; graph clustering; reports	Community reports, answered one by one and merged
GNN-RAG [51]	Existing knowledge graph	Reasoning path	Graph connectivity around question entities	Train a graph neural network to score answer nodes	Shortest paths written as text
KET-RAG [52]	Text	Entity, relation, text chunk	Graph from core chunks; keyword–text links	Select core chunks; partial graph extraction; keyword graph	Graph descriptions and text chunks
DSE [53]	Screenshots	Screenshot	Text, images, and page layout	Contrastive training; one vector per screenshot	Not evaluated
ColPali [29]	Page images	Page	Contextualized image patches	Contrastive training; patch vectors per page	Not evaluated
M3DocRAG [54]	PDF images	Page	Within-page visual content	ColPali page vectors; optional approximate index	Page images to a multimodal generator
UniIR [55]	Text, images, pairs	Multimodal item	Aligned text and image features	Instruction tuning; item embeddings	Not evaluated

Granularity and segmentation A *proposition* expresses one fact with enough context to stand alone. Rewriting passages into propositions can turn "It overtook China in April 2023" into "India overtook China as the most populous country in April 2023" (Dense X [17]). With retrievers trained without labeled question–passage pairs, indexing propositions instead of passages raises the share of questions whose answer appears in the top five results by 9–12 percentage points. The gain is largest for rarely mentioned entities. Propositions fit more answer-bearing text into a fixed token budget, though the gain in answer accuracy is smaller. Narratives need a different boundary: splitting them at topic changes keeps passages coherent and ranks evidence better than recursive splitting by length (LumberChunker [56]). Small units help fact lookup when each unit still carries the context needed to understand it.

Larger units keep context but shift work to the generator. Grouping related Wikipedia

documents into units of several thousand tokens shrinks the index from 22 million units to 600 thousand and raises the share of fact-lookup questions from Natural Questions whose answer appears in the top retrieved unit from 52% to 72% (LongRAG [27]). The generator then has to read tens of thousands of tokens, and beyond a few units, adding more makes its answers worse. No single size suits every collection. Concise factual answers favor small chunks, technical explanations favor large ones, and the best size also shifts with the embedding model [57]. The unit should match where the answer lives, and its size has to be tuned for each collection and retriever.

Contextual embeddings A unit can carry context in its embedding without changing its text. *Late chunking* encodes the whole document before averaging the token vectors within each chunk [18]. The embedding of "It overtook China" can then reflect the earlier mention of India. Late chunking preserves source wording and avoids the extra LLM calls needed to write a context summary for every chunk. Over chunks embedded independently, the method gains about two points of nDCG on average.[1] The benefit depends on meaningful surrounding text: synthetic tasks that hide a relevant sentence in unrelated text gain nothing. Repeating text through overlap is less selective. In a controlled study, a 20% overlap enlarged the index by a quarter without improving answers [58]. Encoding useful context can improve matching without duplicating the text the generator later reads.

Hierarchies and metadata Hierarchies let a question match either a summary or a detailed passage. They can be built by repeatedly clustering and summarizing chunks, with the originals kept as leaves (RAPTOR [1]), or by linking a document's table of contents to an entity graph [59]. The search strategy then matters. Searching all levels of a summary tree at once works better than descending one level at a time [1]. In another tree, each node stores only information absent from its parent's summary. Within a supplied document, descent matches flat retrieval's answer quality with about 30% fewer context tokens. When search must first choose the document from a collection, routing accuracy falls to about half that of flat retrieval [60]. A root summary about demographic trends may omit the country name needed to select its report. These results favor descent within a known document and direct access to detailed passages when the document itself must be found.

Metadata records a unit's source, date or other attributes beyond its main text. In annual filings with repeated boilerplate, adding the company and fiscal year to chunk representations helps distinguish otherwise similar passages. Embedding the metadata separately also permits updates without re-embedding the text [61]. Dates help distinguish the population rankings in the 2021 and 2023 reports, though a system must check when the reported estimates apply as well as when the report appeared. Versioned documents need links between states too. An index that records document versions and their changes answers 90% of questions on a version-specific benchmark correctly, against 58% for standard RAG and 64% for GraphRAG (VersionRAG [43]). Source dates and explicit version links give retrieval information that text similarity alone cannot recover.

3.2 Graph-structured knowledge

A knowledge graph stores entities and the relations between them explicitly, so a system can follow a chain of relations instead of hoping that one passage states the whole chain. A graph that links India to its national anthem and the anthem to its author answers the second half of the anthem question with a two-step path.

[1] Section 7.2 defines the ranking and answer metrics used in the method comparisons, including nDCG, Hit@k, exact match and F1.

Existing graphs When a graph is supplied, the design question is which part of it to give the generator: single facts, paths that connect the question's entities to candidate answers, or whole subgraphs. One method runs a graph neural network over a subgraph around the question's entities to score candidate answers, then gives the generator the shortest paths from the question's entities to the best candidates, written out as text (GNN-RAG [51]). Others retrieve a connected subgraph and keep the text of its nodes and edges alongside its structure [62, 63]. Selecting facts one at a time can break these connections. Searching the graph with a question's subquestions finds more relevant entities than searching with the whole question, but it returns disconnected pieces, and answer accuracy peaks when the search balances the two [64]. What the generator receives should keep the connections the question depends on, not only the facts that individually score highest.

Graphs constructed from text When the collection is text, extracting a graph can miss facts stated across chunk boundaries. If one chunk reports that "Jana Gana Mana" was written by Tagore and the next says only that "India adopted it as its national anthem in 1950", neither alone links the song's title to India. A graph neural network can locate incomplete regions in an extracted graph and direct targeted extraction there [65]. Keeping the text offers another recovery route. In a cyber-threat intelligence system, graph retrieval improves multi-hop questions, while repairing failed graph queries and falling back on text makes it operationally reliable [66]. Source text lets a system recover facts that extraction left out of the graph.

Extraction is also expensive when a language model reads every chunk. Extracting relations only from central chunks and linking the rest through a keyword graph cuts indexing cost by more than an order of magnitude, with comparable or better retrieval (KET-RAG [52]). Parsers avoid model calls altogether. Syntax trees expose code structure directly [67], and dependency parsing reaches 94% of the answer performance of language-model extraction at much lower cost in one comparison [68]. A full LLM-extracted graph is therefore not always necessary: partial extraction or parsing can retain much of the measured benefit while leaving the source text available for reading.

Community evidence Some questions concern a whole collection rather than one fact, such as what a decade of population reports says about demographic change in Asia. No single passage or path answers them. One approach clusters the extracted entity graph into nested communities, writes a summary of each, and answers a collection-wide question by drawing a partial answer from each summary and merging the partial answers (GraphRAG [28]). Unlike the clusters of a summary tree, which group passages by similarity, these communities follow the connections between extracted entities. On news and podcast transcripts, judges rate its answers as more comprehensive and more diverse than those of standard RAG, while standard RAG wins on directness because its answers are shorter. The summaries are derived evidence: an extraction step and a summarization step stand between each source sentence and the text the generator reads. They suit questions about a whole collection, whereas a specific question, such as who wrote India's anthem, is better served by a path or by the original passage.

3.3 Multimodal and heterogeneous sources

Much of what documents say lies in charts, tables and layout, which a text index sees only through a transcription.

Visual pages and text conversion Optical character recognition (OCR), parsing and captioning make pages searchable by text retrievers, but they decide which visual facts survive. If the population ranking appears only as two crossing curves, an OCR transcription can keep the

axis labels while losing the crossing. Retrievers that embed the page image avoid the transcription. One encodes each screenshot as a single vector and outperforms OCR-based text retrieval by more than 15 points of nDCG on slide decks (DSE [53]); another keeps a vector for each image patch and matches each query token against the patches (ColPali [29]). Retrieving the page is half the job, since the generator must then read it. Passing the retrieved page images to a multimodal generator lets a system answer questions about charts and figures that OCR-based pipelines ignore (M3DocRAG [54]). Visual retrieval finds pages that text conversion misses, and reading those pages as images lets the generator use their charts.

Controlled transcription comparisons recover much of the apparent visual-retrieval advantage on the tested benchmarks. With the retriever fixed, better OCR and text normalization recover up to 9 points of Hit@5 for the keyword retriever BM25 (§4.1) on multilingual benchmarks, and short descriptions of figures add up to 31 points on figure-heavy pages [69]. Language-specific preprocessing, such as word segmentation for Japanese, changes accuracy by more than 10 points with the OCR held fixed. Text retrieval still trails on questions that are visual by nature, such as spatial relations or trends in a plot. Storing both the parsed text and the page image avoids choosing between them [70]. Before concluding that a visual retriever beats text retrieval, an evaluation has to check what the text index actually contained.

A whole page can also be too coarse a unit. Retrieving layout blocks, such as a chart together with its caption, instead of whole pages cuts the generator's input by up to 78% while improving answers. Blocks that follow the layout also retrieve better than a fixed grid of image patches, which splits tables and figures [71]. Linking text, images, tables and equations in graphs is another way to keep such elements connected [72]. Units that follow the document's own layout keep a figure together with the text that explains it.

Heterogeneous alignment An index holding several modalities must match the requested type of content as well as its topic. A request for the chart of India's and China's populations should return the chart rather than a paragraph describing it. Yet retrievers built on multimodal language models tend to return text even when an image is requested [73]. A shared embedding space can be trained to respect modality through query instructions (UniIR [55]) or on-topic training negatives in the wrong modality (MM-Embed [73]). Alternatively, a router can choose among separate collections for each modality and granularity (UniversalRAG [74]; §5.1). A shared index must learn to distinguish modalities during matching; separate indexes make that distinction through an explicit routing decision.

4 Retrieval methods

A retrieval step matches a query against the index and selects evidence for the generator. It must handle differences in wording and cover the facts the answer needs: five highly ranked population reports may repeat one fact while leaving the anthem's author unknown. We distinguish three choices within a step: how to match queries with indexed units (§4.1), how to transform the question into search requests (§4.2), and how to rerank candidates or select complementary evidence (§4.3). Later searches chosen after reading the results belong to retrieval control (§5).

4.1 Sparse, dense and hybrid retrieval

A retriever finds only what its score rewards. It can count shared words, compare one vector per text, or compare vectors token by token. Each kind of match misses different passages. On its own, none of them checks whether a passage supports the answer or merely shares its topic. The best retriever therefore depends on the questions and the collection, and combining two helps only when they miss different passages.

Lexical and semantic matching *Sparse retrieval* represents text through weights on a small subset of vocabulary terms. BM25, a lexical sparse retriever, scores query-word matches and gives rare words more weight. *Dense retrieval* encodes the query and each unit as vectors and compares them. Word overlap preserves names and technical terms, while learned embeddings can match paraphrases, such as "most populous" and "has more people than any other country". Their relative value depends on how questions were written. On questions asked without knowing the answer, a retriever trained jointly with its reader beats BM25 by 19 points of exact match on one benchmark. On questions written by people who knew the answer, whose wording tends to follow the evidence, it falls behind [13]. For entity-centric questions with correct entity annotations, linking names directly to articles also beats BM25 [75]. Lexical matching suits shared terminology, dense matching suits paraphrases, and entity linking suits questions whose entities can be identified reliably.

Hybrid retrieval *Hybrid retrieval* combines sparse and dense results so each can recover what the other misses. On cystic-fibrosis literature, a weighted sum of keyword and dense similarities outperforms either alone, including a dense model trained on scientific papers [76]. On a structured medical textbook, dense retrieval alone answers more questions correctly than the hybrid and needs less computation [77]. Combining scores does not repair a poor retrieval unit: in repetitive financial filings, all tested methods find the right filing far more often than its answer-bearing chunk [78]. Mixture weights can also be chosen per query from retrieval uncertainty or a model's judgment of the top results [79, 80]. Hybrid retrieval is useful when its component retrievers make complementary errors; adding a second retriever without testing that complementarity can increase cost and lower answer quality.

Late interaction A single dense vector can blur matches between particular query words and passage words. *Late interaction* instead compares token vectors after encoding the query and passage separately. For example, ColBERT [2] keeps a vector for every passage token, takes each query token's best match and sums the matches. Passages are encoded once, offline. On web-passage ranking tasks in MS MARCO, it matches the ranking accuracy of a *cross-encoder*, a reranker that reads each query–passage pair together, while cutting reranking time per query from about 11 seconds to 61 milliseconds. The scientific question-answering comparison in Table 4 also favors a late-interaction pipeline, though that pipeline changes the encoder as well [81]. Token-level matching can retain cross-encoder accuracy with much less computation per query, in exchange for storing many vectors per passage.

Evidence-oriented supervision and structure Topical relevance can favor the wrong fact. A passage about India's national song, "Vande Mataram", is on topic for a question about India's national anthem, yet names a different author. Training can target support directly through human-marked evidence rationales and on-topic negatives that do not support the answer [82]. Synthetic labels for which subquestions a passage answers can instead train coverage of distinct facts; one such retriever covers 10% more answer facts than strong dense baselines without losing ordinary ranking accuracy [83]. Graph methods pursue connected evidence by spreading query relevance along weighted edges [84]. This traversal remains inside a retrieval step, unlike the controller's later searches (§5). Evidence supervision and graph structure both help retrieve the connections an answer needs rather than only individually similar units.

4.2 Query transformation

Three transformations turn a question into a better search request. *Rewriting* restates it as a clearer request, *expansion* adds words the source is likely to use, and *decomposition* splits it into simpler questions.

Rewriting and expansion Rewriters can learn from the retrieved documents or answers their queries produce, rather than from human-written rewrites. Rewarding a small model by the answer quality of a frozen generator raises answer F1 on multi-hop Wikipedia questions in HotpotQA by about 5 points over searching with the original question (Rewrite-Retrieve-Read [32]). The trained rewriter also outperforms a much larger model prompted to rewrite. A reranker's document scores can replace the answer reward when answer labels are unavailable [44]. Conversational rewriting adds a shared-context problem: "And who wrote its anthem?" needs to become "Who wrote the national anthem of India?" Training a reformulator and an answer-based expansion jointly against target passages addresses both missing context and missing search terms [85]. Retrieval-based feedback teaches queries to suit the collection instead of merely resembling human rewrites.

Expansion goes further and lets the model write text that resembles the answer. One method generates a hypothetical passage that answers the question and searches with its embedding, so the search compares documents with a document instead of with a short question (HyDE [30]). Without any relevance labels, this raises an unsupervised retriever's nDCG@10 on the 2019 TREC deep-learning queries from 44.5 to 61.3, close to the 62.1 of the same retriever fine-tuned with relevance labels. Another method appends a generated passage to the query for sparse or dense retrieval; it helps BM25 most and helps less as the dense retriever gets stronger [31]. Expansion can also draw on the relations of a knowledge graph, grounded in an initial retrieval [86]. Generated expansions help most on question-like queries, while on specialized scientific collections, classic pseudo-relevance feedback, which adds terms from the top retrieved documents, remains competitive [87]. The generated text carries the model's own beliefs into the search. A model that still thinks China is the most populous country writes a hypothetical passage about China's anthem and retrieves real passages about the wrong song. Generated text can shape the query, but it is a guess about the answer and should not enter the context as evidence.

Decomposition and multiple formulations Independent subquestions can be searched together. Combining their results with those of the original question, then reranking the pool against that original question, ranks relevant documents for multi-hop questions much higher than standard RAG does [88]. Several rewrites carrying different amounts of information offer another way to obtain complementary results [89]. The anthem question shows where advance decomposition stops being enough. It splits into "Which country is the most populous?" and "Who wrote that country's national anthem?", but the second request still lacks a country name. If that name is initially unknown, the controller must wait for the first answer before issuing the second search (§5.2).

The gains from rewriting shrink against a strong pipeline. With dense retrieval followed by a cross-encoder, individual rewriting methods help little. Their combined results raise top-ten retrieval success on enterprise questions by about 12 percentage points because they rescue different questions. The same combination hurts on ambiguous questions, where the lists overlap [90]. Rewriting can also counter a retriever's preference for short, early or repetitive documents. Simple rewrites reduce each bias separately, but only expansion into a generated document corrects the preference when several biases are exploited together [91]. Rewriting earns its cost when it targets an observed matching failure or retrieves complementary results.

4.3 Reranking and evidence selection

A first-stage retriever returns a candidate pool; reranking or selection then chooses the evidence E that will enter context construction. Candidates can be scored for relevance, usefulness to the generator or what they add to one another. Cross-encoder reranking has the clearest gains when the first-stage retriever is weak: on the tested question-answering benchmarks, nearly every

cross-encoder improves a small embedding retriever, while only the largest improves a large one [92]. A gain over a weak first stage therefore gives little reason to add the same reranker after a stronger retriever.

Generator utility Relevance scores can disagree with answer quality: on Natural Questions, more highly rated relevance rankings tend to produce lower answer F1 [93]. *Utility-based selection* scores evidence by its benefit to a generator. For example, IGP [93] measures how much each candidate reduces uncertainty in a short trial generation and drops candidates that reduce it little. With token-probability access and one trial per candidate, it raises average answer F1 over BM25 while cutting the final generator input by about three quarters across five datasets. A smaller multimodal model can estimate image utility instead, improving slightly over relevance-based selection [94]. Selection can also move into training: choosing difficult but answerable evidence teaches a more robust generator without requiring the selector at inference [95]. Utility-based selection can improve the final answer, but its savings must include the work of estimating utility, and its choices depend on the generator being served.

Complementary sets Scoring candidates one at a time cannot tell that the fifth report on the population ranking adds nothing to the first four, or that none of them mentions an anthem. *Set selection* judges candidates together. One method first lists the pieces of information the question requires and then selects a set that covers them, improving multi-hop answering over rerankers that score passages one by one (SetR [33]). Another learns set preferences from how easily the generator produces the reference answer from each candidate set, and it answers more accurately than the first method while selecting fewer passages (OptiSet [45]). Structure gives a third route. Reserving part of a fixed budget for documents linked in a graph to the most relevant ones recovers complete evidence chains more often than spending the whole budget on a flat ranking, because the later steps of a chain often score low on their own [96]. Other selectors make the decision explicit, as a rationale for each admitted passage [97] or as an objective that trades relevance against coverage and redundancy [98]; the rationales make the choice interpretable, and the objective makes its trade-offs adjustable. Selecting sets pays off when the answer needs several distinct facts, as the two halves of the anthem question do.

Better evidence scores have to be checked against final answers. Replacing topical-relevance labels with answer-support labels changes which retrievers rank best, yet does not reliably improve downstream training or answer quality [99]. The effect also depends on opportunities to search again. For report generation, coverage of distinct facts in the retrieved set predicts coverage in the report for one-pass pipelines, but not for iterative pipelines that can gather missing facts later [100]. Complete selection matters most when one retrieval step must supply every fact. In an iterative system, selection and later searches have to be evaluated together.

5 Retrieval control

Adaptive retrieval uses the question and previous results to choose whether to search, what to ask next and when to stop. A model may know who wrote India's anthem but not know that India is now the most populous country. It then needs evidence for the country rather than for the author; skipping retrieval risks an outdated answer, while searching repeatedly can add cost and distractions. We organize control around three questions: how to route a question to no retrieval or a retrieval strategy (§5.1), how to use feedback during a search workflow and decide when to stop (§5.2), and how to train the components that make those decisions (§5.3).

5.1 Retrieval necessity and adaptive routing

Before any search, a router decides whether a question needs retrieval and, if it does, how much. It has no evidence yet, so it must guess what the model already knows. Useful signs include how well known the question's subject is, which strategy worked on similar questions, and how confident the model seems. A router pays off when it skips only searches the model did not need and costs less to run than the searches it skips.

Query-based routing Models know more about popular entities. Retrieving only for questions about less popular ones, as measured by how often their Wikipedia pages are viewed, keeps the benefit of retrieval for rare facts while avoiding passages that can mislead the model about facts it already knows [101]. Counting how often a subject and a relation appear together, instead of how popular the subject is, describes a fact more precisely [102]. Both signals estimate what a model knows from statistics about the world, so they work only for facts whose frequency can be looked up.

Classifying the question avoids frequency lookups. A router can choose no retrieval, standard RAG or iterative retrieval (Adaptive-RAG [36]). Its labels come from which strategy answered each training question correctly and from the question's dataset. They describe the available strategies' performance, so changing those strategies changes what the classifier needs to predict. On multi-hop questions, the router matches the accuracy of always iterating while running 40–50% faster than the iterative baseline. For the tested strategies, choosing which one suffices retains iterative retrieval's accuracy without paying its cost on every question.

Uncertainty-based routing A router can estimate how sure the model is, using signals permitted by its interface. With access to hidden states and several sampled generations, uncertainty can guide retrieval, document selection and reasoning strategy (SeaKR [103]). Using these uncertainty signals improves multi-hop answer F1 by 6 points over the strongest adaptive baseline in its evaluation. Text-only routers instead use disagreement among sampled answers [104] or confidence stated by the model [105]. A model can also learn to emit a retrieval token (§5.3). Neither popularity nor confidence necessarily detects a changing fact. India and its anthem remained well known when the population ranking changed in 2023; a model that learned the old ranking might still name China without hesitation. Treating time sensitivity as a separate routing decision addresses that possibility directly [106]. The question gives a reason to search even when the model is confident.

Turning confidence into a decision requires a cutoff, usually tuned for the model on held-out data. In one text-only router, this improves mean F1 slightly and retrieves about 20% fewer passages than always retrieving, across six datasets and three models. Total token use nevertheless rises because obtaining confidence takes an extra generation [105]. A preliminary answer can instead guide how many passages to fetch, saving latency on known questions while reducing distractors [107]. More information for the router is not always worth gathering: in a matched multimodal comparison, features from an initial retrieval did not reliably improve routing over the query alone [108]. Routing should be assessed by answer quality and total work, including any generation or retrieval performed just to make the decision.

5.2 Search workflows and feedback

The value of a later search depends on what earlier ones revealed, so a fixed list of reformulations, written before any result arrives, cannot follow the dependency between the two steps of the anthem question. Adaptive workflows differ in what the controller reads before its next search: its own reasoning, a draft of the answer, or a judgment of the evidence. Figure 3 contrasts

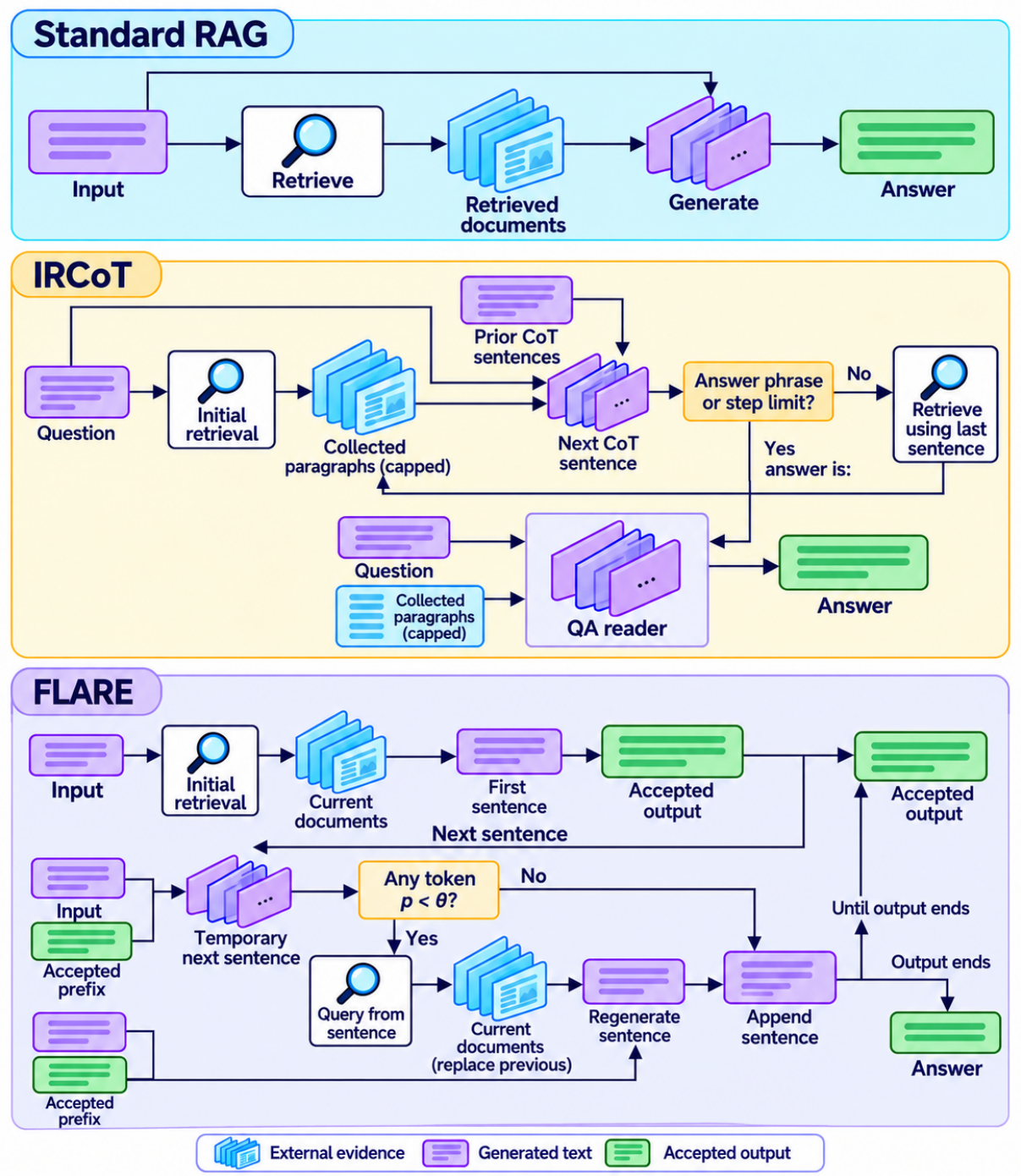


Figure 3: Retrieval can occur once (top), after each reasoning step (middle), or when a draft sentence contains uncertain tokens (bottom). CoT denotes chain-of-thought reasoning; the QA reader is the final answering step. The bottom workflow searches with the draft before regenerating and accepting it [3, 35].

standard RAG with two of these workflows. One searches after every step of its reasoning; the other searches only when its draft of the next sentence is uncertain.

Reasoning and answer feedback Search can follow intermediate reasoning rather than wait for a complete draft. For example, IRCoT [3] writes one sentence of *chain-of-thought* reasoning from the collected passages, searches with that sentence and repeats until an answer appears or a step limit is reached. A separate answering step reads the collected passages. Across multi-hop benchmarks, interleaving raises gold-evidence recall (§7.2) by 11–21 percentage points

over standard RAG and answer F1 by up to 15 points. Explicit follow-up questions offer a more targeted request than a reasoning sentence (Self-Ask [34]). At the other extreme, searching with an entire draft and regenerating it needs fewer calls and answers about six percentage points more HotpotQA questions correctly than the follow-up-question method after two rounds [109]. The useful feedback unit can therefore be a reasoning sentence, an intermediate answer or a whole draft; searching after every small step is not inherently more effective.

Generation-triggered acquisition Long answers raise a different question: when during generation is another search useful? With access to token probabilities, a system can draft the next sentence and search if it contains an uncertain token (FLARE [35]). Suppose the draft names the author of India's anthem but gives the name low probability. Searching with the draft while omitting the uncertain name lets retrieval supply the author before the sentence is accepted. A matched query ablation on multi-hop questions raises exact match from 39.0 to 48.8 when the query uses the upcoming sentence instead of the previous one. Searching for what the model is about to claim targets information that past text may never mention.

Evidence and environment feedback Judging retrieved evidence lets a controller change course even when the model is confident. An evaluator can rate the initial documents as adequate, inadequate or ambiguous, then refine them, replace them with web results or combine both sources (CRAG [110]). A local-first policy instead keeps a curated collection as the default and turns to the web when that collection is inadequate (PrefRAG [111]). Both make source choice depend on evidence quality. For the population question, detecting an old report could justify seeking a current one. In a knowledge graph, evidence judgments guide which paths remain open as the controller selects relations and neighboring entities (Think-on-Graph [112]). Feedback can thus change both where a system searches and which search paths it continues.

Finding the missing evidence is not enough if the system then ranks it too low. In iterative retrieval over graphs, the gains are clearest on bridge questions, whose intermediate entity reveals the missing link. Yet the documents about that intermediate entity, which later searches find, often end up ranked below the top ten, where the generator does not use them [113]. Retrieving more documents does not fix this, since the extra documents add noise and hurt accuracy, especially on simple questions. Issuing two complementary queries and moving those documents to the top of the context does. A new search helps only if what it finds reaches the generator.

Stopping needs its own signal because finding no new evidence does not tell the controller whether the missing facts are obtainable. Trace studies find that most steps in long searches add no new gold evidence (§7.3). A judge attached to a trained search policy saves about 4% of searches at a loss of 0.6 points of exact match, while adding computation of its own [114]. Checking support for each expected step gives a more explicit stopping rule (HALT [46]). It reduces search loops while preserving accuracy in five of six tested settings; predicting the required steps from the question also preserves accuracy, with smaller search reductions. For the anthem question, a report identifying India leaves the author unchecked, while a passage about the anthem leaves the current population ranking unchecked. The coverage rule requires both before stopping.

Extra deliberation must justify its cost as well as extra retrieval. In one orchestrator, a reflection loop raises latency up to sixfold on a multi-hop benchmark while lowering answer quality [115]. Together with the small savings from a separate stopping judge, this shows why counting avoided searches alone is inadequate. A workflow improves efficiency only when its decisions save more work than their own reasoning and verification consume.

5.3 Learning retrieval policies

Training decides which controller decisions receive feedback and which component changes. A retrieval trigger can learn directly from labels for when the model needed evidence. Multi-step search is harder to supervise because answer correctness is observed only after all the searches. A final-answer reward scores the whole sequence without identifying which query helped.

Supervised decisions and reflection The simplest supervision labels when a particular model needs retrieval. Questions it answers correctly without context become direct-answer examples; failures become examples of emitting a retrieval token and answering from retrieved context (Adapt-LLM [116]). These labels depend on the model that produced them. Richer supervision teaches evidence use too. A critic can annotate whether retrieval is needed, a passage is relevant, the output is supported and the answer is useful (Self-RAG [12]). The generator learns to produce the corresponding *reflection tokens* through next-token prediction. Weighting those tokens at inference adjusts citation behavior without retraining. In the reported open-domain, reasoning and fact-verification tasks, the trained model outperforms retrieval-augmented chat baselines. Supervision can therefore teach both the decision to search and the judgments needed to use its results.

Outcome rewards and training environments *Reinforcement learning* (RL) can replace labels for individual search decisions with a reward for the final answer. The model explores sequences of reasoning and searches and learns which lead to correct answers [37, 117]. Retrieved documents are supplied observations rather than actions the model chose to write. Masking them from the training loss updates the policy only for its reasoning, queries and answers (Search-R1 [37]). A live engine makes training costly and exposes it to uncontrolled changes in results. A simulated engine can instead return useful or deliberately noisy documents during training, with a real engine used at inference (ZeroSearch [118]). Policies can likewise learn to search graphs [119] or choose among tools [120]. Final-answer rewards remove the need for labeled search traces, but leave the contribution of each search unspecified.

Credit, efficiency, and controller placement A final-answer reward cannot distinguish a useful search from an irrelevant one followed by a correct guess. Answering "Tagore" from memory earns the same reward as answering from retrieved support. More targeted feedback can separate planning from execution or score the evidence path itself. APEX-Searcher [121] rewards correct decompositions and trains execution separately, while PathRouter [122] weights answer rewards by evidence-path quality and adds teacher guidance for poor searches. Efficiency needs feedback too. Training on a mix of questions answerable from memory and questions requiring search teaches a policy to skip unnecessary retrieval; on knowledge tasks it already handles, it searches for fewer than 2% of questions (SEM [123]). Stage-specific and evidence-based rewards give a policy reasons to search well, while mixed training questions teach when search is unnecessary.

The system's constraints also decide where learning can happen. Training the generator end to end rules out a frozen or proprietary model. One alternative trains only a separate search agent and rewards it by how much its evidence improves a frozen generator's answers over standard RAG. Averaged over six general question-answering benchmarks, it outperforms baselines trained on 70 times more data, using about 2.4 thousand training examples (s3 [124]). Computing its reward still requires running the frozen generator in every training episode. Learning can also move into the retriever: one method uses reinforcement learning to train only the embedder, which learns to choose successive chunks of a long document, so the language model never has to learn to write queries (Q-RAG [125]). A trained controller can sit in the generator, in a separate

search agent or in the retriever, and the choice decides which models must be open to training and which can stay frozen.

6 Evidence integration

The evidence E helps only if it reaches the generator in a usable form and the generator uses it. Integration covers the context constructor A and the generator G in Eq. (2), and each can lose what retrieval found. A compressor can drop the date that tells the 2021 population report from the 2023 one, a generator can side with its memory over the retrieved report, and an answer can cite a source that does not support it. We organize this section by what happens to the evidence: what of it reaches the context (§6.1), how the generator combines several pieces (§6.2), how it handles evidence that conflicts with its memory or with other sources (§6.3), and how its claims are tied back to their sources (§6.4).

6.1 Context construction and compression

Concatenating the selected passages keeps their wording but spends the context budget on useful facts and surrounding text alike. Compression shortens the context by keeping source text, rewriting it, or replacing it with learned vectors. Each choice decides what survives and whether it can still be traced to its source.

Extractive and abstractive compression *Extractive compression* keeps original sentences, so retained text can be traced directly to its source. Sentence selection can use a classifier that reads the sentence in its document context (EXIT [40]), a reranker [126], or the generator's attention when its internals are accessible [127]. The document-aware classifier raises average exact match by about 4 points over uncompressed documents with a large generator. A separate comparison with a smaller generator favors extraction over rewriting: the extractive pipeline answers in under a second where an abstractive compressor needs more than eight. Selecting sentences retains source wording without paying for a newly generated summary.

Abstractive compression rewrites evidence into shorter text. It can merge repeated facts or make evidence easier for the target model to read by combining a summarizer's predictions with that model's predictions [128]. Iterative systems can also replace previously read page images with compact notes (VISOR [129]). Rewriting, however, can hide a loss of support. In one comparison, answers built from abstractive summaries have citation precision (§7.2) of 0.86 against those summaries but only 0.12 against the original sources; extractive compression retains 0.43–0.49 against the originals [130]. If a summary keeps the population ranking but drops "according to United Nations estimates from April 2023", a citation to the summary cannot recover the missing date. For answers that require verifiable citations, extraction preserves a more reliable link to source wording.

Latent compression and structured rendering *Latent compression* replaces text with learned vectors. An encoder can turn a long context into a few memory slots for a frozen language model (ICAE [131]), or a projection can map the retriever's document vector to one generator input embedding (xRAG [39]). These methods shorten the input, but the vectors contain no quotable spans. Verifying source wording requires retaining the original text alongside them. Structured rendering instead controls which explicit facts fit in the context. A graph ordered by distance from the question's entities preserves answer-bearing edges that a naive rendering loses to truncation [132]. Selecting regions from a hierarchy over multimodal evidence similarly cuts input by 14–28% relative to the same iterative pipeline without compression [133]. Vector

compression reduces the tokens the generator reads; structured selection also keeps readable evidence for checking what survived.

Formatting changes what the generator can use even when the content is fixed. Converting passages into atomic statements in JSON lowers performance on multi-hop and temporal reading tasks, while Markdown removes that penalty and uses 33–46% fewer tokens [134]. Metadata helps where the question needs it. Adding the period during which each fact holds improves accuracy on questions that filter facts by date, which is what the population question requires. Splitting a paragraph whose events depend on one another into separate statements, however, hurts questions about how those events relate in time. A context format therefore has to suit the task, since the format that helps date filtering hurts temporal reasoning.

Ordering and context length Generators can use the same passages differently depending on their order. In a controlled reproduction, two open models answer nearly equally well wherever an answer-bearing document is placed, but their end-to-end RAG accuracy still changes with document count and ordering [135]. Long answers also tend to follow the order of their source documents [136]. Ordering therefore needs to be tested on retrieved sets and final answers, even when moving one known answer-bearing passage has little effect.

Context length poses a separate choice. Reading a whole supplied document does better overall than retrieving from it, especially for Wikipedia articles and stories [16]. For larger collections, retrieval lets a short-window model match the same model with a fourfold longer window, and also improves the extended model [15]. Full-document reading suits a known, self-contained source; retrieval remains useful when relevant evidence is scattered through more text than the model should read at once.

6.2 Fusion architectures

When several passages are retrieved, the generator must combine them, and the point at which it combines them decides how much access the system needs, even when the model itself stays frozen. Table 2 compares the mechanisms by what the generator reads, how the evidence is combined, what access and training each needs, and what extra work it adds at inference. Combining passages in a shared prompt needs only text access, combining predictions made from each passage needs token probabilities, and combining internal representations needs access to embeddings, attention caches or encoder and decoder internals.

Prompt and probability fusion Prompt fusion places retrieved text in an unchanged model's input. It needs only a text interface and can improve language modeling without retraining (In-context RALM [11]). More frequent retrieval helps further in that evaluation, while requiring the prefix to be recomputed each time. With access to token probabilities, *probability fusion* can instead combine predictions from separate model runs. Retrieval-weighted averaging runs the model once per document (REPLUG [5]); related methods combine passage-conditioned probabilities for a whole output or separately at each token [8]. Another mixes predictions from the full prompt with those from retrieved text alone to favor the evidence [140]. Separate predictions can be combined without letting the model compare the sources. If four equally weighted old reports favor China and one new report favors India, averaging their predictions can preserve the outdated majority. Resolving that disagreement requires comparing what the reports say and when it applies.

Representation fusion *Representation fusion* lets the generator attend jointly to internal representations of several passages. For example, FiD [4] encodes each passage separately with the question, then lets the decoder read all the resulting states. Encoding cost grows linearly

Table 2: Evidence fusion at the prompt, prediction or internal-representation level, with its access and training requirements.

Method	Evidence at generator	Fusion	Inference access	Learned components	Online work
RAG-Sequence [8]	Passage text	Average passage-conditioned probabilities for the whole output	Token probabilities	Query encoder; generator	Candidate generation per passage; sequence scoring
RAG-Token [8]	Passage text	Average passage-conditioned probabilities at each token	Token probabilities	Query encoder; generator	Per-passage next-token prediction
FiD [4]	Passage encoder states	Decoder attends to all passages	Encoder and decoder internals	Encoder; decoder	Independent encodings; joint decoding
In-context RALM [11]	Passage text	Prepend to input prefix	Text prompt	Optional reranker	Prefix recomputation after retrieval
REPLUG [5]	Passage text	Average token predictions, weighted by retrieval score	Token probabilities	Retriever (optional)	One generator pass per passage
RETRO [10]	Encoded retrieved chunks and the text after each	Chunked cross-attention	Decoder internals	Encoder for retrieved chunks; decoder (retriever frozen)	Chunk retrieval; encoding; cross-attention
xRAG [39]	Dense retrieval vector	Project to one input embedding	Input embeddings	Projection layer	Projection; one evidence token
KV-Fusion [137]	Stored attention keys and values for each passage	Join these representations before decoding	Attention caches; decoder internals	Answer decoder	Passage representations computed in parallel; joint decoding
MuRAG [138]	Image patches; text embeddings	Joint multimodal encoding; decoder attention	Multimodal model internals	Multimodal encoder; decoder	Visual encoding; joint reading
VisRAG (multi-image) [139]	Retrieved page images	Joint multi-image prompting	Multi-image prompt	Retriever	Image processing; multi-image decoding

with passage count, and answer accuracy improves as more passages are added in its experiments. The decoder can connect the population report to the anthem passage even though neither alone answers the question. Other designs insert retrieved chunks through internal attention [10], join stored attention keys and values from separately encoded passages [137], or learn how much retrieved information to admit into the model's hidden states [141]. These mechanisms require access to the internals and usually training. A text-only alternative generates separate answers and votes among them when joint prompting returns no answer [142]. Voting can select a complete answer from one passage; joint reading can construct an answer from facts spread across passages.

Multimodal reading Multimodal generators can read retrieved media jointly or combine separate answers. Joint encoding lets a decoder attend to retrieved images and text together with the query (MuRAG [138]); multi-image prompting gives it several retrieved page images directly (VisRAG [139]). With a single-image generator, the latter pipeline can join pages into one image or answer per page and select by confidence and retrieval score. A staged approach first records evidence from each image and then reasons over those records, focusing attention on supporting regions (EVisRAG [143]). Joint reading enables comparison across pages; separate answering suits single-image models when one page can supply a complete answer.

6.3 Grounding and conflicting evidence

Once evidence reaches the generator, it must decide which claims to accept. Conflicting sources call for comparison, a draft calls for verification, and missing support may call for abstention. These responses address different failures: following an outdated memory, accepting a claim that the evidence does not justify, or guessing when a required fact is absent.

Parametric knowledge and source disagreement Controlled conflict studies find two biases. Models favor retrieved passages that agree with what they memorized, and they tend to follow the position that more passages state [21, 22]. A model shown a single coherent passage that contradicts its memory usually accepts it, but once a passage agreeing with its memory is added, one model returns to its prior answer about two times in three [22]. Repetition reinforces the effect: paraphrases of one claim shift answers more than independent pieces of evidence do [144]. Suppose the population retrieval returns mostly old reports and the model still recalls China as the leader. Source counts and remembered knowledge then reinforce the same outdated answer. Resolving the conflict requires checking dates and source dependence instead of treating repetition as independent confirmation.

Explicit conflict handling Conflicts can be handled during token prediction or through an explicit report before answering. *Contrastive decoding* compares predictions made with evidence against those made from memory or misleading context, then suppresses the competing signal (CD2 [21]). It needs access to token predictions and improves answer recall by about 2 points in the reported conflict test. An explicit report instead records the model's own view, refines retrieved statements and summarizes their disagreements for the final generator (CARE-RAG [41]). The full pipeline improves exact match over standard RAG across tested models and datasets. Token-level correction avoids a separate report-generation pass; reporting adds work but exposes the disagreement to both the final generator and a reader.

The right way to resolve a conflict depends on why the sources disagree. The 2021 and 2023 reports disagree because the world changed; a fabricated page naming a different author for India's anthem disagrees because someone planted it. One method decides from the context whether a conflict reflects change over time, deliberate manipulation or unresolved uncertainty, and only then decides how to present the evidence (EvoTrustRAG [145]). The credibility of each source is another signal. Giving the generator a reliability rating for each publisher addresses the case in which misleading sources outnumber reliable ones [146]. When a conflict cannot be resolved, a system can keep it explicit: it can turn the evidence into a measure of uncertainty and choose to answer, to explain the disagreement or to abstain [147]. Questions with several legitimate readings call for a qualified answer instead. Neither counting sources nor deferring to the newest-looking one handles all of these cases, because resolving a conflict requires knowing whether the sources describe different times, different readings of the question, or a falsehood.

Verification and revision Checking a draft against explicit premises can catch claims its evidence does not justify. One method extracts question-relevant premises, drafts an answer and revises it when a support scorer finds weak support (PAVE [148]). With retriever and model fixed, this raises accuracy by 33 percentage points over standard RAG on a subset of SQuAD. The premises must retain the conditions on each claim. A list that drops "according to 2023 estimates" can make a timeless population ranking appear supported, just as a compressed summary can hide missing support (§6.1). Verification should therefore preserve a route from each premise to the original evidence, including its dates and qualifications.

Selective use and answering Training can teach a model to use evidence selectively. Fine-tuning on a mix of relevant and distracting documents placed in the prompt teaches it to ignore what does not help [20, 149]. When the context lacks what the answer needs, the right response may be to abstain. When a classifier's judgment of sufficiency decides when to abstain, the share of correct answers among those the model gives rises by 2–10 points across model families [19]. Training can teach the same boundary directly. One method orders evidence from unsupported through partial and minimally sufficient to redundant, and trains the model to abstain until the evidence becomes sufficient [150]. Another trains the model to tell missing knowledge from ambiguity in the question, and to withhold answers in uncertain or conflicting cases [151]. Selective answering gives fewer answers in exchange for more reliable ones. Abstaining whenever the context is insufficient would also withhold "Tagore" after the system has retrieved only the 2023 report, although many models know the answer. The right policy therefore depends on whether an application accepts correct but unsupported answers.

6.4 Attribution and citations

Citations let a reader check an answer, but only if each citation points to a source that supports the claim beside it. A sentence stating that Tagore wrote India's anthem, cited to the 2023 population report, has a citation but no support. Benchmarks for generation with citations give the model numbered passages, ask it to cite them statement by statement, and check each citation with a model that judges whether the cited passage supports the statement [38]. Giving the model summaries or snippets of the passages instead of their full text makes its answers more often correct but lowers the quality of its citations. Shortening the evidence thus trades citation quality for correctness, so the two have to be measured together.

Before and during generation Citations can be fixed before the text is written. One method selects fine-grained source spans first, plans how to use them, and then generates sentences grounded in those spans, so each sentence's support exists before its wording does (Attribute First, then Generate [152]). Planning with a sequence of questions that the text will answer improves attribution in the same way [153]. Inline methods cite while generating. One requires a document identifier and a verbatim snippet after every claim. It either enforces this format with constrained decoding or lets the model write an approximate snippet and afterwards matches it to the closest exact span in the source (FullCite [154]). The second approach raises the token F1 (§7.2) between quoted and source snippets on ASQA from about 13 to 62 for one model. Exact quotation makes every cited span easy to locate and check.

Post-hoc support alignment A located span still has to support its claim, and post-hoc methods check and repair citations after generation. They differ in what they may change: one refines only the citations of a finished answer [155], while another checks the draft's claims against the extracted evidence and can rewrite what the answer says (VeriCite [42]). Citations can also be made more precise, pointing to the shortest sub-sentence span that suffices [156]. Repair

has to check the original sources. As Section 6.1 showed, citations checked against compressed summaries can look accurate while most of them cannot be traced to the original text, so aligning citations to a summary hides exactly the loss the check should catch.

Influence and visual evidence A supporting citation can be attached to an answer produced from memory. *Influence* asks whether the source changed the model's predictions, rather than whether it justifies the claim. One method identifies answer tokens whose probabilities shift with the context and traces them to source words through gradients (MIRAGE [157]). Another removes random subsets of sources, measures changes in answer probability and fits a sparse linear model to estimate each source's contribution (ContextCite [158]). These methods estimate dependence on the evidence through internal computations or repeated model runs. They complement citation checks by asking what changed the prediction, not merely what could support it.

For charts and scanned pages, an attribution must also say where on the page the support lies. One method generates the answer together with a bounding box around the supporting region of the original screenshot (VISA [159]), and another trains each step of its reasoning to name the page and region it relies on (Look-As-You-Think [160]). If the population ranking appears only in a chart, a box around the crossing curves lets a reader check a claim that a citation to the whole page would leave vague. Region-level attribution makes visual evidence as checkable as quoted text.

7 Evaluation and diagnosis

Evaluation asks whether each stage did its job: whether retrieval found relevant evidence, whether the context was sufficient, whether the output is supported by the context and correct, and what all of this cost. Because these criteria concern different objects in Eq. (2), a single answer score cannot say which stage failed. Suppose a system names the author of China's anthem in reply to the anthem question. Retrieval may have returned an outdated report, or the generator may have ignored a current one; either way the answer is scored wrong. If the benchmark's reference answer itself dates from before 2023, the same answer is scored right. We organize this section around what an evaluation has to specify: the task, the evidence it supplies and the controls that make comparisons fair (§7.1); the measurements for each criterion and the judges who apply them (§7.2); the interventions that locate failures and the attacks and defenses that test robustness and privacy (§7.3); and the resources a system spends (§7.4).

7.1 Tasks, benchmarks and experimental controls

An evaluation supplies a question and access to evidence, then judges whether the answer meets the task's requirements. Four task families need different references. (i) *Fact lookup* asks for one localized fact, usually checked against an answer string. (ii) *Multi-hop answering* connects several facts, such as the country and author in the anthem question. Benchmarks connect Wikipedia articles (2WikiMultiHopQA [161]) or news articles (MultiHop-RAG [162]) and annotate *gold evidence*, the facts or passages designated as supporting the reference answer. (iii) *Long-form answering* requires coverage of several points or interpretations. ASQA supplies ambiguous questions and reference answers covering their readings [163]; citation benchmarks add support checks (ALCE [38]), while Long^{2}RAG measures coverage of document key points [164]. (iv) *Collection-wide synthesis* asks what many sources say about a theme, so success requires breadth rather than one answer-bearing passage [28].

Benchmarks also differ in what evidence the system receives, which decides the stages they test. Some fix a collection that the system must search, such as Wikipedia for knowledge-intensive

tasks [165] or a fixed, human-verified set of web documents for deep-research questions [166]. Others supply the context, as a long text or as a multi-page document with charts and evidence spread across its pages [167, 168]. A benchmark can also mix source types, such as text, tables and knowledge-graph records, and annotate which source each question needs [169]. Table 3 compares representative resources by the evidence the system receives, what they measure and how they score it. A score on a benchmark that supplies the context says nothing about whether the system could have found that context.

Matched conditions A diagnostic comparison changes the evidence condition while keeping the question, generator and decoding settings fixed. One protocol uses four conditions: no evidence, retrieved evidence, the full source and only the gold supporting evidence [171]. The differences reveal how much retrieval recovers and how much the generator uses. Comparing two retrievers requires additional controls on evidence budget and presentation, so a longer input or a different ordering does not masquerade as better retrieval. The collection snapshot, source versions and possible training exposure must also be recorded. Reference answers age along with sources: a reference from 2021 would name the author of China's anthem for the running question. A fixed collection removes live-search variation [166]; recording when the collection and references were fixed makes the comparison reproducible.

7.2 Component and answer quality

Each criterion of Section 2.1 has its own measurements. Retrieval metrics score the evidence E, sufficiency ratings score the context C, support metrics compare the output y with C, and answer metrics compare y with a reference. Figure 4 places the measurements along the evidence path. Many are applied by judges, human or automatic, and Table 4 shows several of them side by side.

Retrieved evidence and sufficiency Ranking evaluation receives a question, a retrieved list and relevance annotations. Success means placing relevant units early. *Hit@k* is the fraction of questions with at least one relevant unit in the top k. *Mean reciprocal rank* (MRR) averages the reciprocal rank of the first relevant unit, assigning zero when none is found. A first relevant unit at rank three contributes 1/3. *Normalized discounted cumulative gain* (nDCG) sums relevance gains with rank r weighted by $1/\log_2(r+1)$, then divides by the sum for an ideal ranking. Gains commonly use $2^{\text{rel}} - 1$ for a graded relevance label rel. The @k variants stop at rank k; these scores range from zero to one and are often reported as percentages.

Multi-hop evaluation also checks completeness. *Precision* P is the fraction of predicted units matching the reference, and *recall* R is the fraction of reference units recovered. *F1* combines them as $F_1 = 2PR/(P+R)$, with zero when both are zero. *Gold-evidence recall* applies recall to annotated supporting units recovered by retrieval. *Support F1* compares selected evidence with annotated supporting sentences or passages. For the anthem question, ranking the population report first gives perfect MRR, but retrieving no anthem passage recovers only half of a two-passage gold set. Joint task-and-evidence scores enforce both requirements.[2]

Gold-evidence recall still counts units, not information. Sufficiency instead asks whether the context C allows a definite answer, which an LLM rater can judge without a reference answer [19]. A rater must detect missing information rather than topical overlap. One lightweight scorer is built to ignore passage count and word overlap. Tested on pairs of contexts in which one member lacks a needed fact, it ranks the sufficient member higher in 84% of pairs on average [173]. Earlier sufficiency raters that run locally do worse, and so does a large commercial model

[2]KILT credits a task output only when the full annotated set of gold pages is retrieved [165]. Reader-interface diagnostics can further require every gold unit to reach the generator's input [172].

Table 3: Evaluation resources by the evidence the system receives, what they measure, and how they score it.

Resource and kind	Task / evidence unit	Evidence setting	Measured target	Scoring / annotation
KILT [165] Benchmark	Knowledge-intensive tasks; Wikipedia pages	Fixed collection search	Task output; retrieval of gold pages; output credited only with all gold pages	Task metrics; gold pages
MultiHop-RAG [162] Dataset	Multi-hop questions; news chunks	Fixed collection search	Evidence retrieval; answer accuracy	Gold evidence per query
ASQA [163] Dataset	Ambiguous questions; long answers	Collection search; runs without retrieval as controls	Coverage of interpretations; answer quality	Human references; word overlap and a question-answering model
ALCE [38] Benchmark	Long-form or list questions; cited passages	Collection search	Fluency; correctness; citation support	Task metrics; model checks that each citation supports its statement
LongBench [167] Benchmark	Long text; questions, summaries, code and other tasks	Supplied context	Long-context task performance	Task-specific automatic metrics
MMLongBench-Doc [168] Benchmark	Questions on long documents; visual pages	Supplied document	Answer accuracy; unanswerable detection	Reference answers; accuracy and F1
mmRAG [169] Benchmark	Questions over text, tables and knowledge-graph records	Unified collection search	Routing; retrieval; answer quality	Pooled LLM relevance labels
BrowseComp-Plus [166] Benchmark	Deep-research questions; web documents	Fixed collection search	Retrieval; answer accuracy; search calls	Human-verified evidence documents
Sufficient Context [19] Diagnostic protocol	Questions; query–context pairs	Supplied or retrieved context	Sufficiency; correctness; abstention	LLM sufficiency and answer raters
RAGChecker [6] Framework	Long-form questions; atomic claims	Retrieved context and output	Recall of reference claims; correctness; support by context	Reference answer; LLM checks of each claim
GroUSE [170] Judge benchmark	Grounded questions; reference–answer tests	Supplied test cases	Judge calibration; failure detection	Curated unit tests; expected score ranges

prompted as a rater. Scores of this kind order contexts by sufficiency, and a system that abstains on them still has to choose a threshold. Because these raters need no reference answer, a system can run them before it generates, to decide whether to answer or search again.

Sufficient context improves answers only when the generator uses it. On three question-answering benchmarks, models answer more accurately when the context is sufficient, yet they still often answer wrongly instead of abstaining [19]. Checking sufficiency can identify whether

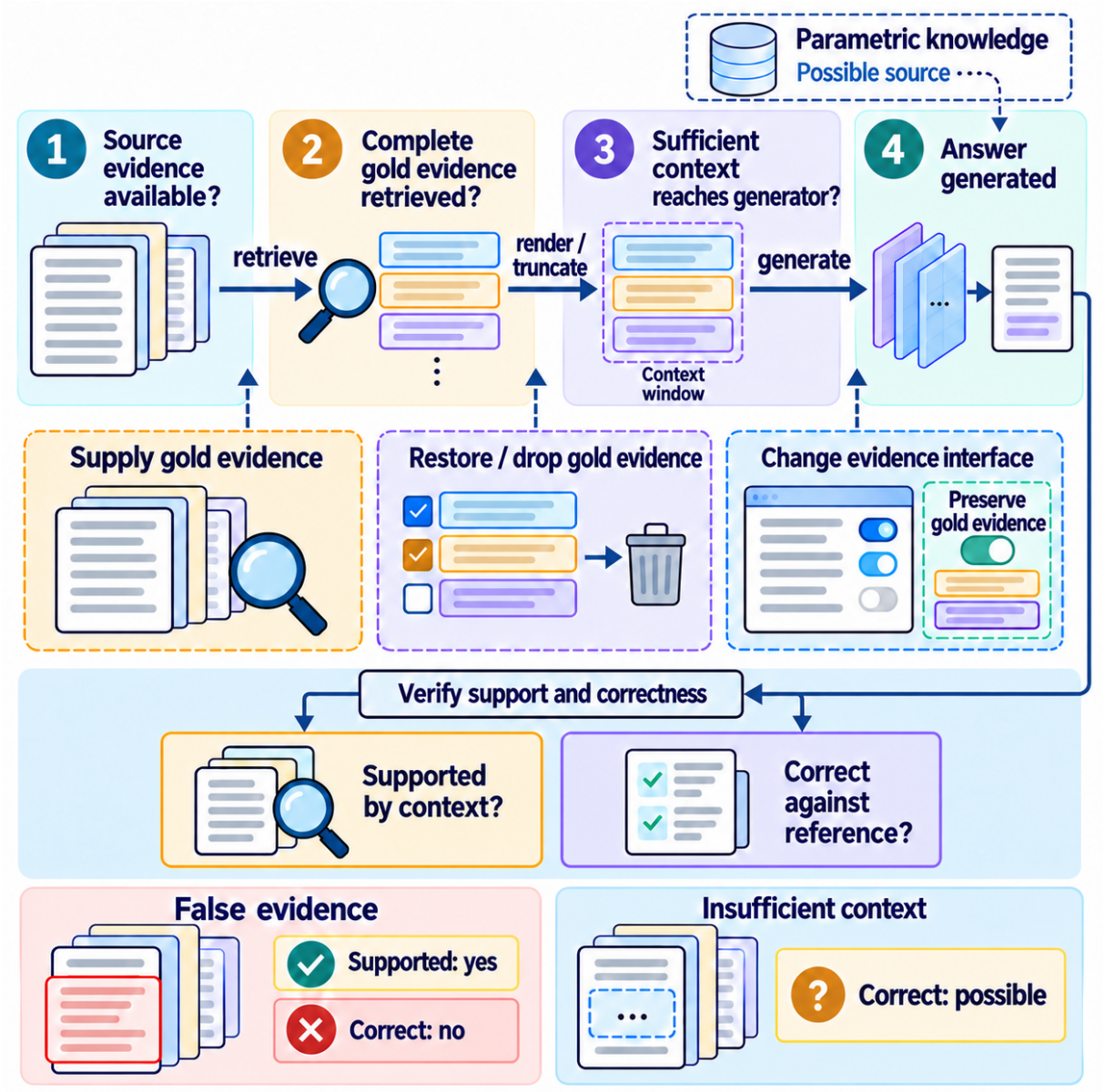


Figure 4: Evidence can be lost in retrieval, context construction or generation. Supplying gold evidence, restoring or dropping it, and changing its presentation test these stages separately. The lower panels distinguish support for an answer from agreement with a reference [6, 19, 165, 172].

more evidence is needed, but answer and support checks are still needed to detect mistakes made after that evidence arrives.

Claims, citations, and abstention Answer evaluation compares the generated output y with a reference. *Exact match* (EM) gives one when the normalized prediction equals an accepted reference and zero otherwise. *Answer F1* applies the preceding calculation to token occurrences in the prediction and reference. It gives partial credit to "Rabindranath Tagore" against "Tagore", while exact match rejects the longer form unless the benchmark accepts it as an alias. *Judged accuracy* is the fraction of answers an evaluator accepts as correct, permitting equivalent wording beyond string matching.

Support evaluation instead compares the answer with its context C. Reference-free checks judge whether answer statements follow from the context [174]. Claim-level frameworks also compare atomic answer claims with the reference, so correctness and context support receive

separate scores [6]. For citations, *citation recall* is the fraction of statements fully supported by the passages they cite. *Citation precision* is the fraction of citations contributing to that support [38]. These checks establish justification rather than influence (§6.4). Inserting a supporting statement into a previously uncited document makes one model cite that document in up to 57% of cases [175]. A plausible citation alone therefore cannot establish that the source shaped the answer.

Abstention needs a stated policy before it can be scored. One evaluation model treats a system as successful when it answers correctly after retrieval succeeds or abstains after retrieval fails [176]. On one benchmark, systems with nearly identical accuracy differ threefold in how often they follow this policy, because some of them guess when retrieval fails. Accuracy among attempted answers hides such differences. A *risk–coverage curve* keeps them visible by plotting the error rate among answered questions against the share of questions answered, so that two systems can be compared at the same share.

The unit of support must also match the representation. For graph-based systems, evaluation can compare paths and communities in a knowledge graph built from the answer and the context, instead of isolated facts [177]. For multimodal sources, one framework splits the answer into atomic subclaims and checks each against text, image, audio or video sources, since text-only metrics adapted to video correlate poorly with human judgments [178]. Text overlap cannot show whether a chart, an image region or a stretch of audio supports a claim, so multimodal evaluation has to check the claim against the medium itself.

Judges and human assessment A language-model judge must be validated for the judgment it is asked to make. On a benchmark of controlled answer variations with expected score ranges, existing evaluation frameworks miss important failure modes [170]. Open evaluator models fail many of its individual tests even though their scores correlate strongly with those of a large commercial judge. For support judgments in a shared evaluation of RAG systems, an independent expert agreed more with a language-model judge than with the original human annotator [179]. Humans who post-edited the model's labels agreed with them far more often than humans who judged from scratch, 72% against 56%, so editing anchors the human judgment. People differ among themselves too. For long-form answers, crowd workers favor concise answers, whereas domain experts penalize answers that omit important content, and standard automatic metrics correlate poorly with the experts [180]. Failure detection, system ranking, individual support labels and overall preference are different properties, so a judge validated for one cannot be trusted for another.

Whole-pipeline comparison Scientific question answering illustrates the difference between choosing a pipeline and isolating a mechanism (Table 4). Six pipelines answer synthetic questions about the problem or method described in a paper abstract [81]. They share the collection and generator; successful retrieval returns the paper used to write the question. Hit@3 and MRR@3 measure that success. A separate model assigns overall answer ratings from one to five using only the question and answer, so it does not check support against the retrieved papers.

The configurations change more than one component: reranking increases the candidate pool, and late interaction changes the encoder as well as matching. Under those conditions, the late-interaction pipeline retrieves the target among its top three results about twice as often as the baseline and receives higher answer ratings. Rank fusion over query rewrites and model-directed tool calling both fall below the baseline on the paired answer-score comparison. For these synthetic scientific questions, the tested late-interaction configuration gives the strongest retrieval and answer ratings.

Table 4: Retrieval and answer scores for six scientific question-answering pipelines that share a collection, a generator and a judge.

Pipeline	Hit@3 (%)	MRR@3 (%)	Answered mean (1–5)	Paired Δ [95% CI]
Problem queries ($n = 9{,}742$); study: [81]				
Baseline	46.3	39.3	3.71	Reference
Query rephrased	39.7	33.3	3.69	−0.023 [−0.040, −0.006]
Rephrased + reranked	47.5	41.6	3.84	+0.126 [+0.110, +0.142]
Rank fusion (RRF)	38.6	30.6	3.64	−0.093 [−0.111, −0.076]
Tool call	45.7	39.0	3.57	−0.170 [−0.190, −0.150]
Late interaction (ColBERT)	92.8	89.4	3.96	+0.272 [+0.255, +0.290]
Method queries ($n = 9{,}742$); study: [81]				
Baseline	46.8	40.6	3.50	Reference
Query rephrased	45.9	39.7	3.55	+0.059 [+0.039, +0.079]
Rephrased + reranked	52.8	48.2	3.71	+0.231 [+0.212, +0.251]
Rank fusion (RRF)	44.2	36.4	3.47	−0.057 [−0.078, −0.036]
Tool call	48.5	42.4	3.51	−0.050 [−0.073, −0.027]
Late interaction (ColBERT)	94.8	92.2	3.92	+0.467 [+0.446, +0.487]

The answered mean averages overall judge ratings only where an answer was produced. Paired Δ averages per-question score differences from the baseline over all questions, counting non-answers as zero. CI denotes the study's 95% confidence interval for that difference. The different treatment of non-answers explains why the two columns can favor different pipelines. The common generator also wrote the questions; all pipelines search 463,971 arXiv records.

7.3 Failure analysis and robustness

Diagnosis tests how answers change when evidence or search steps are altered. Missing or reordered evidence probes retrieval and context use; changes to an intermediate search probe error propagation. Adversarial tests then ask whether a system retrieves and adopts planted material, while privacy tests ask what a query or answer discloses. Each test needs a defined intervention and an observed consequence rather than a judgment of the final answer alone.

Evidence interventions Removing a found gold passage, or restoring a missed one, tests whether the generator's answer depends on it (Figure 4). On multi-hop benchmarks, missing gold evidence at the generator's input costs 22–41 points of answer F1; restoring it raises F1 by a similar amount. Moving that evidence to the front of an otherwise unchanged context adds 3–7 points [172]. These interventions distinguish missing information from poorly presented information. A comparison of retrievers must keep presentation fixed because ordering alone can move answer F1 by several points.

Distractor tests must specify the starting evidence, the number of added passages and their similarity to the question. One study reports that near-topic passages without the answer distract more than random passages, and that adding random passages can improve accuracy [181]. A later replication adds five non-gold passages from each similarity band to gold or retrieved evidence. No tested band improves correctness above the gold-only baseline, and the pattern persists with the earlier study's generator [182]. Random text is therefore not a dependable remedy for poor evidence; any benefit must be demonstrated for the model and context being used.

Targeted perturbations reveal failures that random damage can miss. In knowledge graphs, deleting answer-path edges hurts much more than deleting a fifth of all edges at random [183]. Ordering tests also need enough questions: the apparently best order can change between small test samples [135]. A robustness result should therefore state which evidence was changed and how the tested questions were sampled, so random perturbations do not hide dependence on a

particular fact or order.

Search trajectories and causal attribution A *search trajectory* records queries, returned evidence and intermediate outputs in order. Comparing accumulated evidence with the gold set shows when useful material arrived and which steps added none. Across six agents searching a fixed collection, answer accuracy correlates strongly with accumulated gold-evidence recall ($r = 0.99$) and weakly with final context size ($r = 0.16$). Between 78% and 94% of steps add no new gold evidence, and most useful evidence arrives early. A retrospective estimate puts accuracy retained by a 20-search cap at 86–92% of full-run accuracy [184]. These observations identify costly repetition and favor measuring evidence gathered over simply measuring context length.

Locating a causal failure requires changing the suspected step. Suppose an old report directs the search toward China. Repairing that step and rerunning the continuation tests whether the correction recovers the right answer. In a benchmark that deliberately corrupts intermediate evidence, repair-based probing identifies the faulty step in two thirds of 18 failed cases at depth two, whereas diagnosis from final evidence coverage identifies none [185]. The small test supports intervention as a diagnostic tool when later errors hide the initial fault.

Written reasoning can be tested in the same way. Explicit reasoning about conflicting evidence improves answer accuracy, but that gain alone does not show that the written rationale caused the answer [186]. Injecting errors into the rationale changes one model's answers in most cases and barely changes another's. Comparing original and perturbed continuations therefore tests a dependence that final answer scores and plausible reasoning traces leave unresolved.

Adversarial exposure and privacy Adversarial tests place deliberately misleading material in the collection and measure whether it is retrieved and believed. An attack can optimize a passage to rank highly across a topic while poisoning no more than a millionth of the collection [187]. In another benchmark, planted passages rank first for more than 83% of questions [188]. A false page crediting India's anthem to another author has reached the model if retrieved; it changes the answer only if the generator adopts it. Exposure and adoption must therefore be measured separately.

Generator tests measure the answer change after contaminated evidence is supplied. A small chat model invents answers for 96% of questions with answer-free contexts, and adding one adversarial document removes more than half of its correct answers [189]. Training against contamination provides a direct defense test: one RL method rewards correct extraction without adopting planted claims or following planted instructions [190]. Adoption falls from 73% to 69%, a statistically inconclusive change, and obedience to planted instructions does not fall. Robustness tests must measure both ranking exposure and the generator's response, since a defense at either stage can leave the other failure intact.

Privacy evaluation first specifies the observer and the information to hide. A remote index service can learn a user's interests from the query. Private retrieval protocols restrict that disclosure at the cost of computation and communication. A client can select a document cluster locally from public cluster centers, then fetch it using encryption that permits computation without revealing the plaintext (PIR-RAG [191]). Alternatively, a perturbed query vector can narrow the search before encrypted exact-distance computation and private document fetching (RemoteRAG [192]). These protocols address what the retrieval server observes; protecting the collection from users requires checking generated answers as well.

The generator can still reveal the collection. Prompts can make a RAG system reproduce private documents from its retrieval database [193]. An attacker can also infer whether a particular document is in the database from the answers alone, or from token probabilities where they are exposed [194]. Defenses must be checked at the stage they claim to protect. One audit examined three RAG implementations meant to provide *differential privacy*, a guarantee limiting

what their outputs reveal about any single document [195]. Their apparent protection came from changes at retrieval, while their hooks at generation were inactive placeholders. *Canary strings*, deliberately inserted text unlikely to occur naturally, test whether generated answers disclose source content. A defense that changes one part of the system protects only that part.

7.4 Efficiency and resource trade-offs

Resource evaluation measures the work needed to prepare a system and answer its queries at a stated quality. Preparation includes index construction, refreshes and model training. Query work includes encoding, searching, reranking, compression and generation. In the anthem example, choosing a second search requires intermediate reasoning as well as another retrieval, and the final generation reads evidence from both. Search and token counts capture parts of that work. We distinguish stage-by-stage runtime costs, work moved into preparation and storage, and budgets that limit expensive queries.

The resource measures count different things. *Latency* is elapsed time to an answer; *time to first token* measures only the wait for generation to begin. *Energy per query* is the energy consumed by the measured hardware while answering, reported in joules. *Monetary cost* totals charges for model calls, tools and source access. These measurements require stated boundaries, including whether preparation is counted and whether requests run individually or in batches.

Stage profiles and hardware Profiling tests whether a stage saves more downstream than it consumes. In a system retrieving ten documents for an eight-billion-parameter generator, reranking reduces time to first token from 322 to 165 milliseconds, offsetting its own 24-millisecond overhead. Combining reranking and compression cuts energy per query by two thirds [196]. On an edge device, however, removing only a tenth of the context raises total energy use by up to 13% because compression costs more than it saves [197]. Shorter prompts improve efficiency only when the measured generation savings exceed selection or compression overhead.

Retrieval accuracy also trades against memory and latency. On financial and technical documents, a late-interaction page-image retriever ranks evidence best among the tested pipelines, but needs ten times the memory of a single-vector visual retriever and more than 20 seconds per query. Dense pipelines finish in under a second [198]. The ranking gain therefore suits applications that can afford slower queries and larger indexes; a latency-constrained application must weigh that gain against its response-time budget.

Preparation, storage, and communication Precomputing passage representations moves work out of each query while adding storage. A small query-time encoder can adapt those representations and match full retrieval-augmented reading at a fraction of its query computation (LUMEN [199]). Graph extraction makes preparation costly in a different way. Limiting extraction and using graph elements to point back to source text retains answer quality at a fraction of full graph-construction cost [200]. The comparison must therefore include both what is precomputed and how often that preparation will be reused.

Dialogue memory makes the importance of reuse concrete. In one reproduction, building a memory costs 73–120 thousand tokens for each dialogue history, while reducing the context read per question from about 1.0 to 0.7 thousand tokens [201]. Heavy compression answers 57% of questions correctly, below the 67% from retrieving raw turns, while milder compression reaches 71%. Changing only the retriever over a fixed memory store moves accuracy from 58% to 76%. A short final prompt therefore establishes neither a low total cost nor a good memory design. Preparation must be amortized over the questions that reuse it, and compression must be tested with the retriever that will read the memory.

Private retrieval adds communication as well as computation, through cluster downloads or

encrypted distance calculations (§7.3). Reporting final prompt length alone misses that work. Cost reports should separate preparation, query computation, storage and transferred data so applications can compare the resources they actually pay for.

Budgets and tails A small number of expensive queries can dominate resource use. In one hybrid-retrieval evaluation, an exact top-20 ranking takes fewer than 311 document accesses for most queries but more than 3.5 million for a difficult one. Limiting accesses directly retains at least 95% of nDCG@20 while avoiding much of that work [202]. Search caps offer a related control for agents, with the retrospective accuracy trade-off described in Section 7.3. A budget should bound the operation that consumes resources, such as document accesses or search calls. Reports should then show the resulting quality and the distribution of actual costs, including expensive queries that an average conceals.

8 Open problems

RAG systems must remain reliable as their sources, questions and models change. New population estimates can leave old summaries and graph edges in the index, alter whether another search is needed and invalidate a reference answer. We group the remaining problems by what must stay consistent: evidence across sources and versions (§8.1), retrieval decisions under changing conditions (§8.2), and evaluations that others can reproduce (§8.3).

8.1 Dynamic and heterogeneous evidence

Freshness requires more than adding new documents. When population estimates are revised, every representation derived from the old reports, from propositions and summaries to extracted graph edges and community reports, can keep the old ranking. Questions about earlier states, such as which country was the most populous in 2020, need the old versions kept as well. Version-aware indexes record which statements supersede others (§3.1). An index-free alternative searches the raw documents at query time, trading preparation for computation on every query [203]. Tracing each part of a question to the passages that answer it and to their verbatim source text keeps derived evidence checkable [204]. The open problem is to keep derived representations consistent with changing sources, and to preserve the history needed for questions about the past, without rebuilding everything at every change.

Gathering complete evidence becomes harder when support crosses repositories, languages or media. Ranking several repositories by one score can favor the largest or the one whose wording best matches the question. Allocating the context by estimated source relevance helps retain evidence from others [205]. Yet a conclusion may require a table, a qualification elsewhere and a later revision together. Selection and updates must preserve those dependencies as well as individual facts.

Cross-language search shows how much evidence can be lost before reading begins. In a benchmark that translates evidence for English deep-research questions into eleven other languages, supplying gold evidence raises answer accuracy by roughly 65–75 percentage points [206]. Finding evidence is the larger obstacle, though reading remains harder than with the original English sources even when gold evidence is supplied. The next test is to preserve support across native multilingual sources that change over time, including claims whose evidence spans languages, modalities and versions.

8.2 Reliable retrieval control

A controller needs to estimate what another search could change. Repeatedly producing the same answer is weak evidence for stopping when every sample sees the same wrong or empty retrieval. Separate checks of the answer, evidence and retrieval state detect errors hidden by that agreement, but one tested rule accepts only about 8% of answers [207]. The problem is to identify inadequate evidence without either endorsing a repeated retrieval error or abstaining on almost everything. The decision must also remain reliable when sources, questions or models change. A policy trained against one engine, including a simulated engine (§5.3), has to recognize a different engine's omissions and ranking errors.

Stopping makes errors irreversible within a run: the controller cannot recover missing evidence after it has ended the search. Coverage-based rules already reduce searches while preserving answer accuracy in several settings, whereas a separate learned judge saves relatively few searches (§5.2). The remaining question is when verification reduces total cost after its own computation is included, and whether the rule still works when the collection or search engine changes.

Detecting a problem must also lead to a useful action. In multi-turn tests with contradictory evidence, models acknowledge the conflict and still give unsafe recommendations; among the tested small models, acknowledgment is statistically unrelated to safe resolution [208]. A controller must choose whether to search again, revise or abstain, rather than merely describe its uncertainty. Conflicting population reports, for example, call for checking which estimates apply to the requested date. Training needs to reward that corrective action and test its effect on later searches, including cases where an early mistake has already changed the query sequence.

8.3 Reproducible system evaluation

Benchmark averages can hide parts of a collection that are barely tested. Entity-based clustering reveals topic regions with few questions and systematically weaker retrieval [209]. More questions from the same skewed distribution can leave those regions underrepresented. Generating questions to fill the gaps introduces dependence on the question-writing model, as in the scientific comparison of Section 7.2. Reproducible evaluation needs to record both how questions were produced and which parts of the collection they cover.

Evidence dependence also needs tests that separate source use from memorization. Fictional news timelines make the supplied evidence necessary by asking about invented events [210]. Real dated claims retain natural questions, while restricting their evidence to earlier documents prevents later information from leaking into the collection [211]. These designs address different problems: invented events test reliance on evidence, and dated collections test what could have been known at the time. A benchmark must state which of those abilities its construction tests.

The evaluator can introduce another source of variation. In a specialized industrial domain, an LLM judge rates answer relevance up to twice as highly as experts; statistical correction anchored in expert labels is needed to use its judgments [212]. Together with the judge disagreements in Section 7.2, this makes validation part of the evaluation itself. Releasing the judge's instructions and expert checks helps others distinguish system differences from differences in how answers were rated.

Reproducing a diagnosis requires the search history as well as the final answer. Removing, replacing or duplicating evidence in a one-pass pipeline tests dependence on individual items [213]; in an adaptive pipeline the same change may alter every later query. A recorded trace and a regenerated continuation therefore answer different questions. Cost comparisons likewise need the preparation, controller work and hardware measurements of Section 7.4. Shared records of inputs, search steps, continuations and costs would let an evaluator repeat the intervention and compare systems under the same resource budget.

9 Conclusion

A RAG system must preserve evidence from its source to the answer. Indexing can separate a fact from its context, retrieval can return relevant passages that leave a required step unanswered, and generation can ignore the needed fact even after receiving it. The anthem example distinguishes these failures: losing the country's name in segmentation, retrieving population reports without an author, or following old reports despite a current estimate. Relevance, sufficiency, support and correctness check different parts of that path. Keeping them separate lets evaluation identify a failure and test a repair.

The resulting design choices depend on the task. Fact lookup benefits from small, self-contained units, while questions spanning a document need longer context or explicit connections. Source dates, versions and links to original text keep evidence interpretable as collections change. Later searches help when they fill an identified gap, and support checks for each required step offer a concrete stopping rule. Compression must be judged by what support survives, and conflicting sources by why they disagree. Comparisons must then include the work of making these decisions, along with preparation, storage and generation. Evaluations that follow the evidence through these stages can explain both an answer's quality and the resources needed to obtain it.

References

[1] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. arXiv preprint arXiv:2401.18059, 2024.

[2] O. Khattab and M. Zaharia. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020.

[3] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. arXiv preprint arXiv:2212.10509, 2023.

[4] Gautier Izacard and Edouard Grave. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. arXiv preprint arXiv:2007.01282, 2020.

[5] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. REPLUG: Retrieval-Augmented Black-Box Language Models. arXiv preprint arXiv:2301.12652, 2023.

[6] Dongyu Ru, Lin Qiu, Xiangkun Hu, et al. RAGChecker: A Fine-grained Framework for Diagnosing Retrieval-Augmented Generation. arXiv preprint arXiv:2408.08067, 2024.

[7] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024.

[8] Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems 33 (NeurIPS 2020), 2020.

[9] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Few-shot Learning with Retrieval Augmented Language Models. arXiv preprint arXiv:2208.03299, 2022.

[10] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, et al. Improving language models by retrieving from trillions of tokens. arXiv preprint arXiv:2112.04426, 2022.

[11] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-Context Retrieval-Augmented Language Models. arXiv preprint arXiv:2302.00083, 2023.

[12] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. arXiv preprint arXiv:2310.11511, 2023.

[13] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent Retrieval for Weakly Supervised Open Domain Question Answering. Annual Meeting of the Association for Computational Linguistics, 2019.

[14] Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei, and Athanasios V. Vasilakos. Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG. arXiv preprint arXiv:2501.09136, 2025.

[15] Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. Retrieval meets Long Context Large Language Models. arXiv preprint arXiv:2310.03025, 2023.

[16] Xinze Li, Yixin Cao, Yubo Ma, and Aixin Sun. Long Context vs. RAG for LLMs: An Evaluation and Revisits. arXiv preprint arXiv:2501.01880, 2024.

[17] T Chen, H Wang, S Chen, W Yu, K Ma, X Zhao, D Yu, et al. Dense X Retrieval: What Retrieval Granularity Should We Use? arXiv preprint arXiv:2312.06648, 2023.

[18] Michael Günther, Isabelle Mohr, Daniel James Williams, Bo Wang, and Han Xiao. Late Chunking: Contextual Chunk Embeddings Using Long-Context Embedding Models. arXiv preprint arXiv:2409.04701, 2024.

[19] Hailey Joren, Jianyi Zhang, Chun-Sung Ferng, Da-Cheng Juan, Ankur Taly, and Cyrus Rashtchian. Sufficient Context: A New Lens on Retrieval Augmented Generation Systems. arXiv preprint arXiv:2411.06037, 2024.

[20] Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. Making Retrieval-Augmented Language Models Robust to Irrelevant Context. arXiv preprint arXiv:2310.01558, 2023.

[21] Z Jin, P Cao, Y Chen, K Liu, X Jiang, J Xu, Q Li, and J Zhao. Tug-of-war between knowledge: Exploring and resolving knowledge conflicts in retrieval-augmented language models. arXiv preprint arXiv:2402.14409, 2024.

[22] Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive Chameleon or Stubborn Sloth: Revealing the Behavior of Large Language Models in Knowledge Conflicts. arXiv preprint arXiv:2305.13300, 2023.

[23] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv preprint arXiv:2312.10997, 2023.

[24] Aoran Gan, Hao Yu, Kai Zhang, Qi Liu, Wenyu Yan, Zhenya Huang, Shiwei Tong, and Guoping Hu. Retrieval augmented generation evaluation in the era of large language models: A comprehensive survey. arXiv preprint arXiv:2504.14891, 2025.

[25] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph Retrieval-Augmented Generation: A Survey. ACM Transactions on Information Systems, 2026.

[26] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-Augmented Language Model Pre-Training. arXiv preprint arXiv:2002.08909, 2020.

[27] Ziyan Jiang, Xueguang Ma, and Wenhu Chen. LongRAG: Enhancing Retrieval-Augmented Generation with Long-context LLMs. arXiv preprint arXiv:2406.15319, 2024.

[28] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv preprint arXiv:2404.16130, 2024.

[29] Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. ColPali: Efficient Document Retrieval with Vision Language Models. arXiv preprint arXiv:2407.01449, 2024.

[30] Luyu Gao, Xueguang Ma, Jimmy J. Lin, and Jamie Callan. Precise Zero-Shot Dense Retrieval without Relevance Labels. Annual Meeting of the Association for Computational Linguistics, 2022.

[31] Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query Expansion with Large Language Models. arXiv preprint arXiv:2303.07678, 2023.

[32] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query Rewriting for

Retrieval-Augmented Large Language Models. arXiv preprint arXiv:2305.14283, 2023.

[33] Dahyun Lee, Yongrae Jo, Haeju Park, and Moontae Lee. Shifting from Ranking to Set Selection for Retrieval Augmented Generation. arXiv preprint arXiv:2507.06838, 2025.

[34] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and M. Lewis. Measuring and Narrowing the Compositionality Gap in Language Models. Conference on Empirical Methods in Natural Language Processing, 2022.

[35] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active Retrieval Augmented Generation. arXiv preprint arXiv:2305.06983, 2023.

[36] S Jeong, J Baek, S Cho, SJ Hwang, and JC Park. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. arXiv preprint arXiv:2403.14403, 2024.

[37] Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. arXiv preprint arXiv:2503.09516, 2025.

[38] Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling Large Language Models to Generate Text with Citations. arXiv preprint arXiv:2305.14627, 2023.

[39] Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xRAG: Extreme Context Compression for Retrieval-augmented Generation with One Token. arXiv preprint arXiv:2405.13792, 2024.

[40] Taeho Hwang, Sukmin Cho, Soyeong Jeong, Hoyun Song, SeungYoon Han, and Jong C. Park. EXIT: Context-Aware Extractive Compression for Enhancing Retrieval-Augmented Generation. arXiv preprint arXiv:2412.12559, 2024.

[41] Juan Chen, Baolong Bi, Wei Zhang, Jingyan Sui, Xiaofei Zhu, Yuanzhuo Wang, Lingrui Mei, and Shenghua Liu. Rethinking All Evidence: Enhancing Trustworthy Retrieval-Augmented Generation via Conflict-Driven Summarization. arXiv preprint arXiv:2507.01281, 2025.

[42] Haosheng Qian, Yixing Fan, Jiafeng Guo, Ruqing Zhang, Qi Chen, Dawei Yin, and Xueqi Cheng. VeriCite: Towards Reliable Citations in Retrieval-Augmented Generation via Rigorous Verification. arXiv preprint arXiv:2510.11394, 2025.

[43] Daniel Huwiler, Kurt Stockinger, and Jonathan Fürst. VersionRAG: Version-Aware Retrieval-Augmented Generation for Evolving Documents. arXiv preprint arXiv:2510.08109, 2025.

[44] Shengyu Mao, Yong Jiang, Boli Chen, Xiao Li, Peng Wang, Xinyu Wang, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. RaFe: Ranking Feedback Improves Query Rewriting for RAG. arXiv preprint arXiv:2405.14431, 2024.

[45] Yi Jiang, Sendong Zhao, Jianbo Li, Bairui Hu, Yanrui Du, Haochun Wang, and Bing Qin. OptiSet: Unified Optimizing Set Selection and Ranking for Retrieval-Augmented Generation. arXiv preprint arXiv:2601.05027, 2026.

[46] Daeyoung Roh and Donghee Han. HALT: Verification-Aware Stopping for Retrieval-Augmented Search Agents. arXiv preprint arXiv:2608.02009, 2026.

[47] Zulun Zhu, Tiancheng Huang, Kai Wang, Junda Ye, Xinghe Chen, and Siqiang Luo. Graph-based Approaches and Functionalities in Retrieval-Augmented Generation: A Comprehensive Survey. arXiv preprint arXiv:2504.10499, 2025.

[48] Mohammad Mahdi Abootorabi, Amirhosein Zobeiri, Mahdi Dehghani, Mohammadali Mohammadkhani, Bardia Mohammadi, Omid Ghahroodi, Mahdieh Soleymani Baghshah, and Ehsaneddin Asgari. Ask in Any Modality: A Comprehensive Survey on Multimodal Retrieval-Augmented Generation. arXiv preprint arXiv:2502.08826, 2025.

[49] Yuelyu Ji, Zhuochun Li, Rui Meng, and Daqing He. Retrieval–Reasoning Processes for Multi-hop Question Answering: A Four-Axis Design Framework and Empirical Trends. arXiv preprint arXiv:2601.00536, 2026.

[50] Chan-Wei Hu, Yueqi Wang, Shuo Xing, Chia-Ju Chen, Suofei Feng, Ryan Rossi, and Zhengzhong Tu. mRAG: Elucidating the Design Space of Multi-modal Retrieval-Augmented Generation. arXiv preprint arXiv:2505.24073, 2025.

[51] Costas Mavromatis and George Karypis. GNN-RAG: Graph Neural Retrieval for Efficient Large

Language Model Reasoning on Knowledge Graphs. Findings of the Association for Computational Linguistics: ACL 2025, 2025.

[52] Yiqian Huang, Shiqi Zhang, and Xiaokui Xiao. KET-RAG: A Cost-Efficient Multi-Granular Indexing Framework for Graph-RAG. arXiv preprint arXiv:2502.09304, 2025.

[53] Xueguang Ma, Sheng-Chieh Lin, Minghan Li, Wenhu Chen, and Jimmy Lin. Unifying Multimodal Retrieval via Document Screenshot Embedding. arXiv preprint arXiv:2406.11251, 2024.

[54] Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. M3DocRAG: Multi-modal Retrieval is What You Need for Multi-page Multi-document Understanding. arXiv preprint arXiv:2411.04952, 2024.

[55] Cong Wei, Yang Chen, Haonan Chen, Hexiang Hu, Ge Zhang, Jie Fu, Alan Ritter, and Wenhu Chen. UniIR: Training and Benchmarking Universal Multimodal Information Retrievers. arXiv preprint arXiv:2311.17136, 2023.

[56] André V. Duarte, João Marques, Miguel Graça, Miguel Freire, Lei Li, and Arlindo L. Oliveira. LumberChunker: Long-Form Narrative Document Segmentation. arXiv preprint arXiv:2406.17526, 2024.

[57] Sinchana Ramakanth Bhat, Max Rudat, Jannis Spiekermann, and Nicolas Flores-Herr. Rethinking Chunk Size For Long-Document Retrieval: A Multi-Dataset Analysis. arXiv preprint arXiv:2505.21700, 2025.

[58] Sofia Bennani and Charles Moslonka. A Systematic Analysis of Chunking Strategies for Reliable Question Answering. arXiv preprint arXiv:2601.14123, 2026.

[59] Shu Wang, Yingli Zhou, and Yixiang Fang. BookRAG: A Hierarchical Structure-aware Index-based Approach for Retrieval-Augmented Generation on Complex Documents. arXiv preprint arXiv:2512.03413, 2025.

[60] Junaid Farooq. Semantic Compression Trees: Multi-Resolution Knowledge Retrieval via Hierarchical Semantic Residuals. arXiv preprint arXiv:2608.21610, 2026.

[61] Raquib Bin Yousuf, Shengzhe Xu, Mandar Sharma, Andrew Neeser, Chris Latimer, and Naren Ramakrishnan. Utilizing Metadata for Better Retrieval-Augmented Generation. arXiv preprint arXiv:2601.11863, 2026.

[62] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering. arXiv preprint arXiv:2402.07630, 2024.

[63] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. GRAG: Graph Retrieval-Augmented Generation. arXiv preprint arXiv:2405.16506, 2024.

[64] Valentin Six, Evan Dufraisse, and Gaël de Chalendar. The Structure-Content Trade-off in Knowledge Graph Retrieval. arXiv preprint arXiv:2506.13380, 2025.

[65] Jiaming Zhang, Yibo Zhao, Jing Yu, Jianxiang Yu, and Xiang Li. Beyond Chunk-Local Extraction: Cross-Chunk Graph Augmentation for GraphRAG. arXiv preprint arXiv:2605.28004, 2026.

[66] Dzenan Hamzic, Florian Skopik, Max Landauer, Markus Wurzenberger, and Andreas Rauber. Beyond RAG for Cyber Threat Intelligence: A Systematic Evaluation of Graph-Based and Agentic Retrieval. arXiv preprint arXiv:2604.11419, 2026.

[67] Manideep Reddy Chinthareddy. Reliable Graph-RAG for Codebases: AST-Derived Graphs vs LLM-Extracted Knowledge Graphs. arXiv preprint arXiv:2601.08773, 2026.

[68] Congmin Min, Sahil Bansal, Joyce Pan, Abbas Keshavarzi, Rhea Mathew, and Amar Viswanathan Kannan. Towards Practical GraphRAG: Efficient Knowledge Graph Construction and Hybrid Retrieval at Scale. arXiv preprint arXiv:2507.03226, 2025.

[69] M Asenov, K Benkirane, D Goldwater, and A Ghodsi. Retrieval or Representation? Reassessing Benchmark Gaps in Multilingual and Visually Rich RAG. arXiv preprint arXiv:2603.04238, 2026.

[70] Wang Chen, Wenhan Yu, Guanqiang Qi, Weikang Li, Yang Li, Lei Sha, Deguo Xia, and Jizhou Huang. CMRAG: Co-modality-based visual document retrieval and question answering. arXiv preprint arXiv:2509.02123, 2025.

[71] Yifan Zhu, Yu Mi, Yue Lu, Yanchu Guan, and Zhixuan Chu. LFRAG: Layout-oriented Fine-grained Retrieval-Augmented Generation on Multimodal Document Understanding. arXiv preprint arXiv:2605.22829, 2026.

[72] Zirui Guo, Xubin Ren, Lingrui Xu, Jiahao Zhang, and Chao Huang. RAG-Anything: All-in-One RAG Framework. arXiv preprint arXiv:2510.12323, 2025.

[73] Sheng-Chieh Lin, Chankyu Lee, Mohammad Shoeybi, Jimmy Lin, Bryan Catanzaro, and Wei Ping. MM-Embed: Universal Multimodal Retrieval with Multimodal LLMs. arXiv preprint arXiv:2411.02571, 2024.

[74] Woongyeong Yeo, Kangsan Kim, Soyeong Jeong, Jinheon Baek, and Sung Ju Hwang. UniversalRAG: Retrieval-Augmented Generation over Corpora of Diverse Modalities and Granularities. arXiv preprint arXiv:2504.20734, 2025.

[75] HS Shavarani and A Sarkar. Entity Retrieval for Answering Entity-Centric Questions. arXiv preprint arXiv:2408.02795, 2024.

[76] Priyanka Mandikal and Raymond Mooney. Sparse Meets Dense: A Hybrid Approach to Enhance Scientific Document Retrieval. arXiv preprint arXiv:2401.04055, 2024.

[77] Nusrat Sultana, Abdullah Muhammad Moosa, Kazi Afzalur Rahman, and Sajal Chandra Banik. A Systematic Study of Retrieval Pipeline Design for Retrieval-Augmented Medical Question Answering. arXiv preprint arXiv:2604.07274, 2026.

[78] A Kobeissi and P Langlais. Decomposing Retrieval Failures in RAG for Long-Document Financial Question Answering. arXiv preprint arXiv:2602.17981, 2026.

[79] Hoin Jung and Xiaoqian Wang. Uncertainty-Aware Hybrid Retrieval for Long-Document RAG. arXiv preprint arXiv:2606.13550, 2026.

[80] Hsin-Ling Hsu and Jengnan Tzeng. DAT: Dynamic Alpha Tuning for Hybrid Retrieval in Retrieval-Augmented Generation. arXiv preprint arXiv:2503.23013, 2025.

[81] Bhagyesh Rathi, Eshan Chawla, and William B. Andreopoulos. Re-ranking and Late Interaction Drive Retrieval Quality: A Controlled Comparison of RAG Strategies for Scientific Question Answering. arXiv preprint arXiv:2609.38473, 2026.

[82] Francielle Vargas, João Robiatti, Diego Alves, Lucas Pascotti Valem, Maximilian Seeth, Sebastián Ferrada, Ameeta Agrawal, Daniel Pedronette, and André Freitas. Beyond Topical Similarity: Contrastive Evidence Retrieval with Interpretable Attention Alignment in RAG. arXiv preprint arXiv:2606.01482, 2026.

[83] Jia-Huei Ju, Eugene Yang, Trevor Adriaanse, Suzan Verberne, and Andrew Yates. Search for Coverage: Learning Coverage-Aware Retrieval with Augmented Sub-Question Answerability. arXiv preprint arXiv:2605.28522, 2026.

[84] Zhuoping Zhou, Davoud Ataee Tarzanagh, Sima Didari, Wenjun Hu, Baruch Gutow, Oxana Verkholyak, Masoud Faraki, Heng Hao, Hankyu Moon, and Seungjai Min. Query-Aware Flow Diffusion for Graph-Based RAG with Retrieval Guarantees. arXiv preprint arXiv:2605.18775, 2026.

[85] Fengran Mo, Kelong Mao, Yutao Zhu, Yihong Wu, Kaiyu Huang, and Jian-Yun Nie. ConvGQR: Generative Query Reformulation for Conversational Search. arXiv preprint arXiv:2305.15645, 2023.

[86] Yu Xia, Junda Wu, Sungchul Kim, Tong Yu, Ryan A. Rossi, Haoliang Wang, and Julian McAuley. Knowledge-Aware Query Expansion with Large Language Models for Textual and Relational Retrieval. arXiv preprint arXiv:2410.13765, 2024.

[87] Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. Query Expansion by Prompting Large Language Models. arXiv preprint arXiv:2305.03653, 2023.

[88] Paul J. L. Ammann, Jonas Golde, and Alan Akbik. Question Decomposition for Retrieval-Augmented Generation. arXiv preprint arXiv:2507.00355, 2025.

[89] Zhicong Li, Jiahao Wang, Zhishu Jiang, Hangyu Mao, Zhongxia Chen, Jiazhen Du, Yuanxing Zhang, Fuzheng Zhang, Di Zhang, and Yong Liu. DMQR-RAG: Diverse Multi-Query Rewriting for RAG. arXiv preprint arXiv:2411.13154, 2024.

[90] Sara Shanian, Xiaoqin Yi, Pavlo Ruban, and Kurt MacDonald. Better Together: Complementary Query Rewriting Under a Strong RAG Baseline. arXiv preprint arXiv:2609.05637, 2026.

[91] Agam Goyal, Koyel Mukherjee, Apoorv Saxena, Anirudh Phukan, Eshwar Chandrasekharan, and Hari Sundaram. Masking or Mitigating? Deconstructing the Impact of Query Rewriting on Retriever Biases in RAG. arXiv preprint arXiv:2604.06097, 2026.

[92] Gabriel de Souza P. Moreira, Ronay Ak, Benedikt Schifferer, Mengyao Xu, Radek Osmulski, and Even Oldridge. Enhancing Q&A Text Retrieval with Ranking Models: Benchmarking, fine-tuning

and deploying Rerankers for RAG. arXiv preprint arXiv:2409.07691, 2024.

[93] Z Song, Y Zhou, X Kong, J Jiao, X Bao, X You, X Shi, et al. Less is More for RAG: Information Gain Pruning for Generator-Aligned Reranking and Evidence Selection. arXiv preprint arXiv:2601.17532, 2026.

[94] Weiqing Luo, Zongye Hu, Xiao Wang, Zhiyuan Yu, Haofeng Zhang, and Ziyi Huang. Utility-Oriented Visual Evidence Selection for Multimodal Retrieval-Augmented Generation. arXiv preprint arXiv:2605.13277, 2026.

[95] Jiashuo Sun, Pengcheng Jiang, Saizhuo Wang, et al. Rethinking the Reranker: Boundary-Aware Evidence Selection for Robust Retrieval-Augmented Generation. arXiv preprint arXiv:2602.03689, 2026.

[96] Thomson D. Nguy. More Context, Same Budget: Dual-Bounded Relational Recall Beyond Top-K Retrieval. arXiv preprint arXiv:2608.18448, 2026.

[97] Yash Saxena, Ankur Padia, Mandar S Chaudhary, Kalpa Gunaratna, Srinivasan Parthasarathy, and Manas Gaur. Ranking Free RAG: Replacing Re-ranking with Selection in RAG for Sensitive Domains. arXiv preprint arXiv:2505.16014, 2025.

[98] Rahul Singh and Madhav Vadlamani. QUBO-Optimized Evidence Selection for Retrieval-Augmented Question Answering with Unconventional Solvers. arXiv preprint arXiv:2607.12334, 2026.

[99] Utshab Kumar Ghosh, Debayan Mukhopadhyay, and Shubham Chatterjee. Assessing the Downstream Utility of Evidence-Aware Retrieval in RAG. arXiv preprint arXiv:2608.26379, 2026.

[100] Saron Samuel, Alexander Martin, Eugene Yang, Andrew Yates, Dawn Lawrie, Ian Soboroff, Laura Dietz, and Benjamin Van Durme. Beyond Relevance: On the Relationship Between Retrieval and RAG Information Coverage. arXiv preprint arXiv:2603.08819, 2026.

[101] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust LMs: Investigating effectiveness of parametric and non-parametric memories. ACL, 2023.

[102] Seiji Maekawa, Hayate Iso, Sairam Gurajada, and Nikita Bhutani. Retrieval Helps or Hurts? A Deeper Dive into the Efficacy of Retrieval Augmentation to Language Models. arXiv preprint arXiv:2402.13492, 2024.

[103] Z Yao, W Qi, L Pan, S Cao, L Hu, W Liu, L Hou, and J Li. SeaKR: Self-aware Knowledge Retrieval for Adaptive Retrieval Augmented Generation. arXiv preprint arXiv:2406.19215, 2024.

[104] KD Dhole. To Retrieve or Not to Retrieve? Uncertainty Detection for Dynamic Retrieval Augmented Generation. arXiv preprint arXiv:2501.09292, 2025.

[105] Chandan Kumar Sah, Li Zhang, and Xiaoli Lian. Beyond Self-Knowledge: Propagating Uncertainty Across Reasoning and Retrieval in LLMs. arXiv preprint arXiv:2607.25600, 2026.

[106] Qinyuan Cheng, Xiaonan Li, Shimin Li, Qin Zhu, Zhangyue Yin, Yunfan Shao, Linyang Li, Tianxiang Sun, Hang Yan, and Xipeng Qiu. Unified Active Retrieval for Retrieval Augmented Generation. arXiv preprint arXiv:2406.12534, 2024.

[107] Zhe Dong, Fang Qin, Manish Shah, and Yicheng Wang. Know Before You Fetch: Calibrated Retrieval-Budget Allocation for Retrieval-Augmented Generation. arXiv preprint arXiv:2606.29959, 2026.

[108] Qiaomu Li, Qiuyuan Zhang, and Nong Ming. Beyond the Query: Do Retrieval Signals Improve Adaptive Multimodal RAG Routing? arXiv preprint arXiv:2609.12437, 2026.

[109] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy. arXiv preprint arXiv:2305.15294, 2023.

[110] SQ Yan, JC Gu, Y Zhu, and ZH Ling. Corrective Retrieval Augmented Generation. arXiv preprint arXiv:2401.15884, 2024.

[111] Qingfei Zhao, Ruobing Wang, Yukuo Cen, Daren Zha, Shicheng Tan, and Jie Tang. PrefRAG: Preference-Driven Multi-Source Retrieval Augmented Generation. arXiv preprint arXiv:2411.00689, 2024.

[112] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Sai Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, H. Shum, and Jian Guo. Think-on-Graph: Deep and Responsible Reasoning of Large Language

Model on Knowledge Graph. International Conference on Learning Representations, 2023.

[113] Kai Guo, Xinnan Dai, Shenglai Zeng, Harry Shomer, Haoyu Han, Yu Wang, and Jiliang Tang. Beyond Static Retrieval: Opportunities and Pitfalls of Iterative Retrieval in GraphRAG. arXiv preprint arXiv:2509.25530, 2025.

[114] Weimeng Luo. When Should Multi-Round RAG Stop? Structured Stopping Judgments and Retrieval Reduction in Search-R1. arXiv preprint arXiv:2608.13237, 2026.

[115] Anuj Maharjan, Devinder Kaur, and Richard Molyet. Agent-Orchestrated Adaptive RAG: A Comparative Study on Structured and Multi-Hop Retrieval. arXiv preprint arXiv:2606.05658, 2026.

[116] Tiziano Labruna, Jon Ander Campos, and Gorka Azkune. When to Retrieve: Teaching LLMs to Utilize Information Retrieval Effectively. arXiv preprint arXiv:2404.19705, 2024.

[117] Mingyang Chen, Tianpeng Li, Haoze Sun, et al. ReSearch: Learning to Reason with Search for LLMs via Reinforcement Learning. arXiv preprint arXiv:2503.19470, 2025.

[118] Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. ZeroSearch: Incentivize the Search Capability of LLMs without Searching. arXiv preprint arXiv:2505.04588, 2025.

[119] Haoran Luo, E. Haihong, Guanting Chen, et al. Graph-R1: Towards Agentic GraphRAG Framework via End-to-end Reinforcement Learning. arXiv preprint arXiv:2507.21892, 2025.

[120] Qi Luo, Xiaonan Li, Yuxin Wang, Tingshuo Fan, Yuan Li, Xinchi Chen, and Xipeng Qiu. MARAG-R1: Beyond Single Retriever via Reinforcement-Learned Multi-Tool Agentic Retrieval. arXiv preprint arXiv:2510.27569, 2025.

[121] Kun Chen, Qingchao Kong, Zhao Feifei, and Wenji Mao. APEX-Searcher: Refining Credit Assignment with Subgoaling for Agentic Retrieval-Augmented Generation. arXiv preprint arXiv:2603.13853, 2026.

[122] Bo Wang, Heyan Huang, Yaolin Li, Wei Tang, Yuan Zhang, Wenbo Li, Mingze Gao, Ge Shi, and Chong Feng. PathRouter: Aligning Rewards with Retrieval Quality in Agentic Graph Retrieval-Augmented Generation. arXiv preprint arXiv:2606.16409, 2026.

[123] Zeyang Sha, Shiwen Cui, and Weiqiang Wang. SEM: Reinforcement Learning for Search-Efficient Large Language Models. arXiv preprint arXiv:2505.07903, 2025.

[124] Pengcheng Jiang, Xueqiang Xu, Jiacheng Lin, Jinfeng Xiao, Zifeng Wang, Jimeng Sun, and Jiawei Han. s3: You Don't Need That Much Data to Train a Search Agent via RL. arXiv preprint arXiv:2505.14146, 2025.

[125] Artyom Sorokin, Nazar Buzun, Alexander Anokhin, Oleg Inozemcev, Egor Vedernikov, Petr Anokhin, Mikhail Burtsev, Trushkov Alexey, Yin Wenshuai, and Evgeny Burnaev. Q-RAG: Long Context Multi-step Retrieval via Value-based Embedder Training. arXiv preprint arXiv:2511.07328, 2025.

[126] N Chirkova, T Formal, V Nikoulina, and S Clinchant. Provence: efficient and robust context pruning for retrieval-augmented generation. arXiv preprint arXiv:2501.16214, 2025.

[127] Lvzhou Luo, Yixuan Cao, and Ping Luo. AttnComp: Attention-Guided Adaptive Context Compression for Retrieval-Augmented Generation. arXiv preprint arXiv:2509.17486, 2025.

[128] Dongwon Jung, Qin Liu, Tenghao Huang, Ben Zhou, and Muhao Chen. Familiarity-Aware Evidence Compression for Retrieval-Augmented Generation. arXiv preprint arXiv:2409.12468, 2024.

[129] Yucheng Shen, Jiulong Wu, Jizhou Huang, Dawei Yin, Lingyong Yan, and Min Cao. VISOR: Agentic Visual Retrieval-Augmented Generation via Iterative Search and Over-horizon Reasoning. arXiv preprint arXiv:2604.09508, 2026.

[130] Deepanshu Mody. The Attribution-Compression Frontier in Retrieval-Augmented Generation. arXiv preprint arXiv:2609.14245, 2026.

[131] Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context Autoencoder for Context Compression in a Large Language Model. arXiv preprint arXiv:2307.06945, 2023.

[132] Jayakumar Manoharan and Yamini Sehgal. Seed-Anchored Budget-Bounded Graph Rendering for Question Answering on Industry-Standard Power-Grid Information and Exchange Models. arXiv preprint arXiv:2609.02011, 2026.

[133] Hyojeong Yun, Jueun Kim, and Wook-Shin Han. CANOPY: Adaptive-Granularity Evidence Compression for Multimodal RAG. arXiv preprint arXiv:2610.00923, 2026.

[134] Saber Zerhoudi, Michael Granitzer, and Jelena Mitrovic. Metadata, Structure, or Strategy? A Decomposition of RAG Context Enrichment. arXiv preprint arXiv:2606.29645, 2026.

[135] Jorge Gabín, Anxo Perez, and Javier Parapar. Lost in the Evidence? Reproducing Document Position and Context Size Effects in RAG. arXiv preprint arXiv:2605.27105, 2026.

[136] Hung-Ting Chen, Fangyuan Xu, Shane Arora, and Eunsol Choi. Understanding Retrieval Augmentation for Long-Form Question Answering. arXiv preprint arXiv:2310.12150, 2023.

[137] Philhoon Oh, Jinwoo Shin, and James Thorne. Parallel Key-Value Cache Fusion for Position Invariant RAG. arXiv preprint arXiv:2501.07523, 2025.

[138] Wenhu Chen, Hexiang Hu, Xi Chen, Pat Verga, and William W. Cohen. MuRAG: Multimodal Retrieval-Augmented Generator for Open Question Answering over Images and Text. Conference on Empirical Methods in Natural Language Processing, 2022.

[139] Shi Yu, Chaoyue Tang, Bokai Xu, et al. Visrag: Vision-based retrieval-augmented generation on multi-modality documents. arXiv preprint arXiv:2410.10594, 2024.

[140] Ibne Farabi Shihab, Fariya Afrin, Sanjeda Akter, and Anuj Sharma. Grounded Decoding: Retrieval-Anchored Probability Fusion for Faithful RAG. arXiv preprint arXiv:2606.00432, 2026.

[141] Yixin Chen, Ying Xiong, Shangyu Wu, Xiangrui Ke, Nan Guan, and Chun Jason Xue. ReFilter: Improving Robustness of Retrieval-Augmented Generation via Gated Filter. arXiv preprint arXiv:2602.12709, 2026.

[142] Ye Liu, Semih Yavuz, Rui Meng, Meghana Moorthy, Shafiq Joty, Caiming Xiong, and Yingbo Zhou. Modeling Uncertainty and Using Post-fusion as Fallback Improves Retrieval Augmented Generation with LLMs. arXiv preprint arXiv:2308.12574, 2023.

[143] Yubo Sun, Chunyi Peng, Yukun Yan, Shi Yu, Zhenghao Liu, Chi Chen, Zhiyuan Liu, and Maosong Sun. VisRAG 2.0: Evidence-Guided Multi-Image Reasoning in Visual Retrieval-Augmented Generation. arXiv preprint arXiv:2510.09733, 2025.

[144] Atharv Naphade. Rational Synthesizers or Heuristic Followers? Analyzing LLMs in RAG-based Question-Answering. arXiv preprint arXiv:2601.06189, 2026.

[145] Xi Nie, Hongwei Li, Shenghao Wu, Wenshu Fan, Qiyang Song, and Wenbo Jiang. EvoTrustRAG: Evolution-Aware Conflict Attribution and Evidence Handling for Reliable Retrieval-Augmented Generation. arXiv preprint arXiv:2608.07933, 2026.

[146] Ziyu Ge, Yuhao Wu, Daniel Wai Kit Chin, Roy Ka-Wei Lee, and Rui Cao. Resolving Conflicting Evidence in Automated Fact-Checking: A Study on Retrieval-Augmented LLMs. arXiv preprint arXiv:2505.17762, 2025.

[147] S M Asif Hossain, Ruksat Khan Shayoni, and M. F. Mridha. EvidentialRAG: Quantifying and Mitigating Information Conflict in Multi-Source Retrieval-Augmented Generation via Evidential Deep Learning. arXiv preprint arXiv:2607.10491, 2026.

[148] Tianyi Huang, Caden Yang, Emily Yin, Eric Wang, and Michael Zhang. PAVE: Premise-Aware Validation and Editing for Retrieval-Augmented LLMs. arXiv preprint arXiv:2603.20673, 2026.

[149] Tianjun Zhang, Shishir G. Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. RAFT: Adapting language model to domain specific RAG. arXiv preprint arXiv:2403.10131, 2024.

[150] Haruto Sato, Yuki Tanaka, Ren Nakamura, Aoi Kobayashi, and Mei Ito. Learning Evidence Sufficiency Boundaries for Selective Answering in Grounded Multi-Hop QA. arXiv preprint arXiv:2609.01687, 2026.

[151] Sunguk Shin, Meeyoung Cha, Byung-Jun Lee, and Sungwon Park. ERA: Evidence-based Reliability Alignment for Honest Retrieval-Augmented Generation. arXiv preprint arXiv:2604.20854, 2026.

[152] Aviv Slobodkin, Eran Hirsch, Arie Cattan, Tal Schuster, and Ido Dagan. Attribute First, then Generate: Locally-attributable Grounded Text Generation. arXiv preprint arXiv:2403.17104, 2024.

[153] Constanza Fierro, Reinald Kim Amplayo, Fantine Huot, Nicola De Cao, Joshua Maynez, Shashi Narayan, and Mirella Lapata. Learning to Plan and Generate Text with Citations. arXiv preprint arXiv:2404.03381, 2024.

[154] Anar Yeginbergen, Amelie Wührl, Anna Rogers, and Rodrigo Agerri. Explicit Evidence Grounding via Structured Inline Citation Generation. arXiv preprint arXiv:2606.07130, 2026.

[155] Haosheng Qian, Yixing Fan, Ruqing Zhang, and Jiafeng Guo. On the Capacity of Citation

Generation by Large Language Models. arXiv preprint arXiv:2410.11217, 2024.

[156] Guo Chen, Qiuyuan Li, Qiuxian Li, Hongliang Dai, Xiang Chen, and Piji Li. Concise and Sufficient Sub-Sentence Citations for Retrieval-Augmented Generation. arXiv preprint arXiv:2509.20859, 2025.

[157] Jirui Qi, Gabriele Sarti, Raquel Fernández, and Arianna Bisazza. Model Internals-based Answer Attribution for Trustworthy Retrieval-Augmented Generation. arXiv preprint arXiv:2406.13663, 2024.

[158] Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. ContextCite: Attributing Model Generation to Context. arXiv preprint arXiv:2409.00729, 2024.

[159] Xueguang Ma, Shengyao Zhuang, Bevan Koopman, Guido Zuccon, Wenhu Chen, and Jimmy Lin. VISA: Retrieval Augmented Generation with Visual Source Attribution. arXiv preprint arXiv:2412.14457, 2024.

[160] Shuochen Liu, Pengfei Luo, Chao Zhang, Yuhao Chen, Haotian Zhang, Qi Liu, Xin Kou, Tong Xu, and Enhong Chen. Look as You Think: Unifying Reasoning and Visual Evidence Attribution for Verifiable Document RAG via Reinforcement Learning. arXiv preprint arXiv:2511.12003, 2025.

[161] Xanh Ho, A. Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. International Conference on Computational Linguistics, 2020.

[162] Y Tang and Y Yang. MultiHop-RAG: Benchmarking Retrieval-Augmented Generation for Multi-Hop Queries. arXiv preprint arXiv:2401.15391, 2024.

[163] Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. ASQA: Factoid Questions Meet Long-Form Answers. Conference on Empirical Methods in Natural Language Processing, 2022.

[164] Zehan Qi, Rongwu Xu, Zhijiang Guo, Cunxiang Wang, Hao Zhang, and Wei Xu. Long^{2}RAG: Evaluating Long-Context & Long-Form Retrieval-Augmented Generation with Key Point Recall. arXiv preprint arXiv:2410.23000, 2024.

[165] Fabio Petroni, Aleksandra Piktus, Angela Fan, et al. KILT: a Benchmark for Knowledge Intensive Language Tasks. NAACL, 2021.

[166] Zijian Chen, Xueguang Ma, Shengyao Zhuang, et al. BrowseComp-Plus: A More Fair and Transparent Evaluation Benchmark of Deep-Research Agent. arXiv preprint arXiv:2508.06600, 2025.

[167] Yushi Bai, Xin Lv, Jiajie Zhang, et al. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. arXiv preprint arXiv:2308.14508, 2023.

[168] Yubo Ma, Yuhang Zang, Liangyu Chen, et al. MMLongBench-Doc: Benchmarking Long-context Document Understanding with Visualizations. arXiv preprint arXiv:2407.01523, 2024.

[169] C Xu, Q Chen, Y Feng, and G Cheng. mmRAG: A Modular Benchmark for Retrieval-Augmented Generation over Text, Tables, and Knowledge Graphs. arXiv preprint arXiv:2505.11180, 2025.

[170] Sacha Muller, António Loison, Bilel Omrani, and Gautier Viaud. GroUSE: A Benchmark to Evaluate Evaluators in Grounded Question Answering. arXiv preprint arXiv:2409.06595, 2024.

[171] Haizhou Xia. Diagnosing Evidence Utilization in Long-Context and Retrieval-Augmented Language Models under Matched Evidence Conditions. arXiv preprint arXiv:2606.06758, 2026.

[172] Junchi Liao, Jiawen Deng, and Fuji Ren. Evidence Interfaces Shape How Retrieval-Augmented Readers Use Support. arXiv preprint arXiv:2607.17108, 2026.

[173] Suting Chen, Peichun Hua, and Yunming Xiao. Relevance Is Not Sufficient Evidence: Detecting Evidence Gaps Before Generation in RAG. arXiv preprint arXiv:2609.37469, 2026.

[174] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated Evaluation of Retrieval Augmented Generation. arXiv preprint arXiv:2309.15217, 2023.

[175] Jonas Wallat, Maria Heuss, Maarten de Rijke, and Avishek Anand. Correctness is not Faithfulness in RAG Attributions. arXiv preprint arXiv:2412.18004, 2024.

[176] Pius von Däniken, Felix Matthias Saaro, Mark Cieliebak, and Jan Deriu. The RAT: A Unified Bayesian Model for RAG Evaluation. arXiv preprint arXiv:2608.24753, 2026.

[177] Sicheng Dong, Vahid Zolfaghari, Nenad Petrovic, and Alois Knoll. Knowledge-Graph Based RAG System Evaluation Framework. arXiv preprint arXiv:2510.02549, 2025.

[178] Alexander Martin, William Walden, Reno Kriz, Dengjia Zhang, Kate Sanders, Eugene Yang, Chihsheng Jin, and Benjamin Van Durme. Seeing Through the MiRAGE: Evaluating Multimodal Retrieval Augmented Generation. arXiv preprint arXiv:2510.24870, 2025.

[179] Nandan Thakur, Ronak Pradeep, Shivani Upadhyay, Daniel Campos, Nick Craswell, and Jimmy Lin. Support Evaluation for the TREC 2024 RAG Track: Comparing Human versus LLM Judges. arXiv preprint arXiv:2504.15205, 2025.

[180] Fangyuan Xu, Yixiao Song, Mohit Iyyer, and Eunsol Choi. A Critical Evaluation of Evaluations for Long-form Question Answering. arXiv preprint arXiv:2305.18201, 2023.

[181] Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. The Power of Noise: Redefining Retrieval for RAG Systems. arXiv preprint arXiv:2401.14887, 2024.

[182] Alexandria Leto, Cecilia Aguerrebere, Ishwar Bhati, Ted Willke, Mariano Tepper, and Vy Ai Vo. Toward Optimal Search and Retrieval for RAG. arXiv preprint arXiv:2411.07396, 2024.

[183] D Zhou, Y Zhu, Y He, J Chen, E Kharlamov, and S Staab. Evaluating Knowledge Graph Based Retrieval Augmented Generation Methods under Knowledge Incompleteness. arXiv preprint arXiv:2504.05163, 2025.

[184] Qi Liu, Jiaxin Mao, Fengbin Zhu, and Tat-Seng Chua. Diagnosing Search Behavior and Failure Modes in Long-Horizon Search Agents. arXiv preprint arXiv:2608.01913, 2026.

[185] Lauren Pothuru. When Failures Propagate: Causal Failure Attribution in Agentic Retrieval-Augmented Generation. arXiv preprint arXiv:2608.20627, 2026.

[186] Yihang Chen, Pin Qian, Su Wang, Sipeng Zhang, Huan Xu, Shuhuai Lin, and Xinpeng Wei. Does RAG Know When Retrieval Is Wrong? Diagnosing Context Compliance under Knowledge Conflict. arXiv preprint arXiv:2605.14473, 2026.

[187] Matan Ben-Tov and Mahmood Sharif. GASLITEing the Retrieval: Exploring Vulnerabilities in Dense Embedding-based Search. arXiv preprint arXiv:2412.20953, 2024.

[188] Jinyan Su, Jin Peng Zhou, Zhengxin Zhang, Preslav Nakov, and Claire Cardie. Towards More Robust Retrieval-Augmented Generation: Evaluating RAG Under Adversarial Poisoning Attacks. arXiv preprint arXiv:2412.16708, 2024.

[189] SI Park and JY Lee. Toward Robust RALMs: Revealing the Impact of Imperfect Retrieval on Retrieval-Augmented Language Models. arXiv preprint arXiv:2410.15107, 2024.

[190] Yanyu Chen, Yue Li, Yongyi Cui, Dongsheng Shi, and Lichang Dai. Reinforcement Learning for Large Language Model Selective Evidence Adoption from Contaminated Retrieval Results. arXiv preprint arXiv:2607.20090, 2026.

[191] Baiqiang Wang, Qian Lou, Mengxin Zheng, and Dongfang Zhao. PIR-RAG: A System for Private Information Retrieval in Retrieval-Augmented Generation. arXiv preprint arXiv:2509.21325, 2025.

[192] Yihang Cheng, Lan Zhang, Junyang Wang, Mu Yuan, and Yunhao Yao. RemoteRAG: A Privacy-Preserving LLM Cloud RAG Service. arXiv preprint arXiv:2412.12775, 2024.

[193] Shenglai Zeng, Jiankun Zhang, Pengfei He, et al. The Good and The Bad: Exploring Privacy Issues in Retrieval-Augmented Generation (RAG). arXiv preprint arXiv:2402.16893, 2024.

[194] Maya Anderson, Guy Amit, and Abigail Goldsteen. Is My Data in Your Retrieval Database? Membership Inference Attacks Against Retrieval Augmented Generation. arXiv preprint arXiv:2405.20446, 2024.

[195] Yanhang Li, Zhichao Fan, and Zexin Zhuang. Mind the Hook: Source-Level Auditing of Privacy Defenses in Retrieval-Augmented Generation. arXiv preprint arXiv:2608.09001, 2026.

[196] Zlatan Feric, Amir Taherin, Bin Ren, Yanzhi Wang, Jennifer Dy, and David Kaeli. RAGMark: A Comprehensive Framework for Benchmarking Retrieval-Augmented Generation Systems. arXiv preprint arXiv:2609.05760, 2026.

[197] Zlatan Feric, Amir Taherin, Yanzhi Wang, and David Kaeli. From Retrieved Context to Runtime Control: Adaptive Compression for Edge-based RAG. arXiv preprint arXiv:2608.19535, 2026.

[198] Emre Kuru and Mehmet Onur Keskin. Evaluating Modern RAG: Textual, Multimodal, Dense, and Late Interaction Pipelines. arXiv preprint arXiv:2608.23176, 2026.

[199] Michiel de Jong, Yury Zemlyanskiy, Nicholas FitzGerald, Joshua Ainslie, Sumit Sanghai, Fei Sha, and William Cohen. Pre-computed memory or on-the-fly encoding? A hybrid approach to retrieval

augmentation makes the most of your compute. arXiv preprint arXiv:2301.10448, 2023.

[200] Yuzhong Zhang, Haoyang Ma, Chao Peng, Lionel Briand, Boxi Yu, and Jialun Cao. When Is Graph Structure Worth Its Cost? The Case for Structure Pricing in Retrieval-Augmented Generation. arXiv preprint arXiv:2609.18099, 2026.

[201] Yongjie Zhou, Shuai Wang, Bevan Koopman, and Guido Zuccon. Reproducing LightMem: Naive RAG Is Just as Good for Memory Management. arXiv preprint arXiv:2607.29104, 2026.

[202] Chunran Zhang. Top-K Is Not a Budget for Hybrid Retrieval. arXiv preprint arXiv:2609.15143, 2026.

[203] Xingjun Wang, Gongsheng Li, Qi Fan, Yunlin Mao, Luyan Su, and Yingda Chen. LENS: In-Context Search via Latent Evidence Exploration over Dynamic Raw Documents. arXiv preprint arXiv:2608.16185, 2026.

[204] Linyao Zheng, Xuhang Shi, Zhifang Mao, Sai Zhou, Shuaixian An, Xiuquan Hou, and Jinze Li. LineageRAG: Harnessing GraphRAG by Constructing Evidence Lineages with Source Grounding. arXiv preprint arXiv:2608.16004, 2026.

[205] Hridya Dhulipala, Rajesh Ombase, Michael Wang, and Tien N. Nguyen. W-RAG: Source-Aware Retrieval for Enterprise Document Generation from Heterogeneous Knowledge Bases. arXiv preprint arXiv:2608.22081, 2026.

[206] Yuheng Lu, Qingcheng Zeng, Heli Qi, Puxuan Yu, Fuheng Zhao, Rui Yang, Hitomi Yanaka, Naoto Yokoya, and Weihao Xuan. Beyond Monolingual Deep Research: Evaluating Agents and Retrievers with Cross-Lingual BrowseComp-Plus. arXiv preprint arXiv:2606.15345, 2026.

[207] Sahib Julka. When Confidence Takes the Wrong Path: Diagnosing Retrieval-State Lock-In in RAG. arXiv preprint arXiv:2606.22728, 2026.

[208] Zhe Yu, Wenpeng Xing, Chen Ye, Xuyang Teng, Bo Yang, Changting Lin, and Meng Han. Detecting Is Not Resolving: The Monitoring Control Gap in Retrieval Augmented LLMs. arXiv preprint arXiv:2605.27157, 2026.

[209] Andrew Klearman, Radu Revutchi, Rohin Garg, Rishav Chakravarti, Samuel Marc Denton, and Yuan Xue. Coverage, Not Averages: Semantic Stratification for Trustworthy Retrieval Evaluation. arXiv preprint arXiv:2604.20763, 2026.

[210] Max Glockner, Xiang Jiang, Leonardo F. R. Ribeiro, Iryna Gurevych, and Markus Dreyer. NEOQA: Evidence-based Question Answering with Generated News Events. arXiv preprint arXiv:2505.05949, 2025.

[211] Michael Schlichtkrull, Zhijiang Guo, and Andreas Vlachos. AVeriTeC: A Dataset for Real-world Claim Verification with Evidence from the Web. arXiv preprint arXiv:2305.13117, 2023.

[212] Grégoire Martinon, Alexandra Lorenzo de Brionne, Jérôme Bohard, Antoine Lojou, Damien Hervault, and Nicolas J-B. Brunel. Towards a rigorous evaluation of RAG systems: the challenge of due diligence. arXiv preprint arXiv:2507.21753, 2025.

[213] Siddharth Jain and Venkat Narayan Vedam. CUE-R: Beyond the Final Answer in Retrieval-Augmented Generation. arXiv preprint arXiv:2604.05467, 2026.