



Enhancing Large Language Models with External Knowledge Using Retrieval-Augmented Generation: A Critical Review of Retrieval Design, Context Compression, Safety and Evaluation, with a Conceptual Governance-Aware Architecture

¹Auti Pallavi S., ²Prof.Bangar Abhishek P., ³Prof.Karad Vandana A., ⁴Prof.Bhosale Sachin B.

^{1,2,3,4,5}Department of Computer Engineering, Jaihind College of Engineering, Pune, Maharashtra, India

Abstract. Large language models (LLMs) depend on static parametric knowledge, leading to outdated answers, missing expertise, and hallucinations. Retrieval-Augmented Generation (RAG) addresses these issues by retrieving external evidence during inference, but the interactions between design choices are not well understood. This critical review explores configuration benchmarking, embedding quality, question-type-adaptive retrieval with context compression, comparison with fine-tuning, IAM-based permission filtering, and real-time, safety-critical graph-based RAG. A focused narrative design was used on a defined corpus, with foundational sources. The findings show that no single RAG configuration is best, as the optimal choice depends on the task and metrics. Retrieval quality strongly correlates with answer quality for extractive tasks but weakly for synthesis tasks. Adaptive retrieval and compression can reduce token usage but increase latency. Access control, freshness, and safety can be enforced outside the LLM, but this is only possible under explicit trust assumptions. However, most studies rely on Wikipedia-based question answering, small samples, single runs, and inconsistent metrics. Several studies also contain internal inconsistencies, which are documented here. This review contributes five key elements: a survey matrix, a comparative analysis that separates different experimental settings, an eight-layer conceptual architecture justified by literature, a minimum evaluation protocol for fair comparison, and a research gap agenda linked to evidence. These outputs provide researchers and practitioners with a foundation for building reliable, secure, and governable RAG systems in real-world, enterprise, and safety-critical applications.

Keywords: Retrieval-Augmented Generation; Large Language Models; Embeddings; Context Compression; Access Control; Knowledge Graphs; Evaluation; Critical Review

I. Introduction

Large language models (LLMs) show strong performance across many language tasks, yet their knowledge is fixed at training time. They cannot reliably answer questions about recent events, proprietary documents or specialised domains, and they may produce fluent but unfounded statements, commonly called hallucinations [1], [2]. Retrieval-Augmented Generation (RAG)



addresses this limitation by retrieving relevant passages from an external knowledge base at inference time and conditioning generation on them [2]. RAG has since become a dominant pattern for question answering, enterprise search, summarisation and domain assistants [3].

A RAG pipeline has five stages: chunking documents, embedding chunks as vectors, indexing the vectors for approximate nearest-neighbour (ANN) search, retrieving and optionally reranking candidates, and generating an answer from the retrieved context. Every stage involves design choices (chunk size, embedding model, index type, similarity metric, retrieval strategy, reranker, number of passages) that interact in ways that are hard to predict. Early work asked whether retrieval helps at all; recent work asks which choices help, when, at what cost and under what risks.

Surveys of RAG already exist [3], but a number of recent IEEE Access studies examine specific dimensions of the design space in detail: configuration benchmarking [4], embedding quality [1], question-type-adaptive retrieval with context compression [5], the comparison of RAG with fine-tuning [6], permission-aware retrieval [7] and real-time, safety-critical graph-based RAG [8]. Each reports quantitative evidence, but they use different datasets, generators and metrics, so their findings are rarely read together.

This review therefore has a deliberately critical scope. Its objectives are to: (1) describe the RAG pipeline and its main design choices; (2) summarise the experimental evidence in the six studies; (3) compare accuracy, latency, token cost and safety across them; (4) identify methodological inconsistencies and threats to validity; (5) propose a literature-supported conceptual architecture; (6) propose an evaluation protocol that avoids optimistic bias; and (7) set out a prioritised agenda of research gaps and future directions.

The main contributions are (i) a survey matrix of the six studies, (ii) a comparative analysis that keeps numbers from different experimental settings apart, (iii) a list of internal inconsistencies observed in the source papers, (iv) an eight-layer conceptual architecture that makes governance and freshness explicit, and (v) an evidence-linked research agenda.

II. Review Methodology

1. Source and search strategy

This is a focused narrative review of a defined corpus, not an exhaustive systematic review. The primary evidence consists of six full-text research articles published in IEEE Access between February 2025 and July 2026 and supplied for this study. No database-level search of IEEE Xplore, Scopus or Web of Science was logged, so identification counts for a wider literature are not reported and studies indexed elsewhere may have been missed. Foundational works (the original RAG paper, dense retrieval, benchmark datasets, evaluation frameworks) were taken from the reference lists of the six articles and are used for background only.

2. Eligibility criteria and screening

Each article was screened against the criteria in Table 1.

Table 1: Eligibility criteria applied during screening

Criterion	Included	Excluded
Topic	Uses external knowledge retrieval to augment an LLM, or compares against such systems	LLM prompting studies without a retrieval component



Criterion	Included	Excluded
Evidence	Reports quantitative experimental results	Position papers without experiments
Venue	Peer-reviewed journal article	Preprints as primary evidence
Period	2025–2026 (earlier work as background only)	Earlier primary studies
Language	English	Other languages

3. Study selection

All six articles met the criteria and none was excluded (Fig. 1). The reference lists contributed 18 further sources that are cited for background and are not treated as primary evidence.

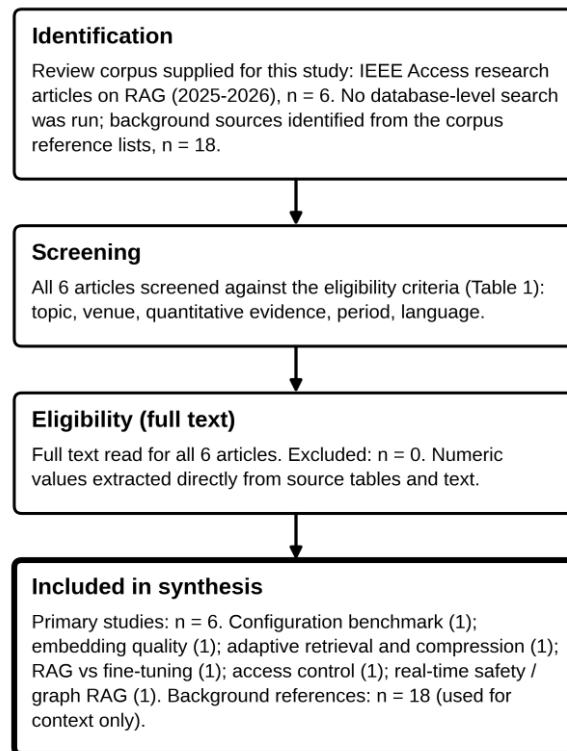


Figure 1: Study identification and selection flow for this review. Counts reflect the supplied corpus; no database-level search was performed.

4. Data extraction and synthesis

For each study we extracted the objective, the pipeline components studied, datasets, models (embedding, generator, reranker, vector store), metrics, principal quantitative results and stated limitations. Numeric values are copied from the source tables and text. Where a source is internally inconsistent we flag the inconsistency instead of resolving it silently. Studies were coded by the pipeline stage they target, their quality and cost measures, and their validity (dataset diversity, baselines, sample size,

reproducibility). Because the studies differ in datasets, generators and metrics, synthesis is qualitative and values are not pooled.

5. Research questions

- RQ1. Which design choices in the RAG pipeline most influence retrieval and answer quality?
- RQ2. What trade-offs in accuracy, latency, token cost and resources do these choices introduce?
- RQ3. How are access control, freshness and safety handled in RAG systems?
- RQ4. What gaps in evidence and methodology limit generalisation?

III. Background: The RAG Pipeline and Its Design Space

1. Pipeline and dense retrieval

Given a query q , a retriever selects a set of passages $R(q)$ from a corpus D and a generator produces an answer conditioned on q and $R(q)$. Dense retrieval maps queries and passages into a shared vector space [9]. Similarity is computed by cosine similarity, inner product or L2 distance. ANN indexes such as HNSW, IVF and ScaNN make search scalable, and rerankers (bi-encoders such as BGE, cross-encoders such as MiniLM) refine the candidate list [4].

2. Retrieval strategy variants

Beyond single-query dense retrieval, systems fuse ranked lists from several retrievers, for example by reciprocal rank fusion (RRF, Eq. 1), retrieve coarsely and then rescore (hierarchical retrieval), or generate a pseudo-answer to enrich the query (HyDE) [4].

$$\text{RRF}(d) = \sum_i 1 / (c + r_i(d)) \quad (1)$$

where $r_i(d)$ is the rank of document d in the i -th list and c is a smoothing constant (typically 60). Adaptive methods route queries by complexity [10] or by question type [5], and self-reflective methods trigger retrieval on demand [11]. Graph-based variants organise knowledge as entity–relation graphs [12] and, more recently, as time-decaying state graphs [8].

3. Context handling

Supplying many passages can add noise, and relevant evidence in long contexts may be overlooked [13]. Prompt and context compression [14], [15] reduces token cost, and SAC-RAG [5] pairs compression with type-specific retrieval.

4. Evaluation

Retrieval is evaluated with Recall@k, MRR, nDCG and coverage. Answers are evaluated with Exact Match (EM) and F1 on extractive tasks [16], [17], with lexical metrics (BLEU, ROUGE, METEOR), embedding-based metrics (BERTScore, cosine similarity) and LLM-assisted measures such as faithfulness and answer relevancy [18]. Operational evaluation adds latency, token consumption, cost and memory. General embedding benchmarks [19], [20] evaluate retrieval but not its effect on generation.

5. Governance and safety

Zero Trust principles [21] require explicit, least-privilege authorisation for every resource access, which conventional RAG over a shared index does not provide [7]. RAG also creates privacy risks [22] and new attack surfaces such as indirect prompt injection through retrieved content [23].



IV. Review of Existing Research

1. Configuration benchmarking

Elkiran and Rasheed [4] benchmark 162 configurations (3 dataset sizes of 10k, 30k and 100k SQuAD documents, 3 indexes, 3 similarity metrics, 3 retrievers and 2 rerankers) using text-embedding-3-large, Milvus and gpt-4o-mini. HNSW with inner product, Fusion retrieval and a MiniLM reranker gives the best semantic quality (coverage 0.942, correctness 0.909, faithfulness 0.970, relevance 0.959), while IVF with L2 and Hierarchical retrieval gives the lowest latency (reported as 1.736). Fusion collapses under L2 (Recall@1 about 0.02–0.05), HyDE is most accurate but slowest and costliest, index choice mainly affects speed, and reranker differences are modest.

2. Embedding quality

Amaral Cejas et al. [1] compare 12 embeddings (22M–600M parameters) on SQuAD (10,570 pairs) and a private 1,000-pair Telecom dataset with top-k = 1 and two 3B-parameter generators. Qwen3-Embedding-0.6B gives the highest retrieval accuracy on both datasets (about 71% and 77%) and the best answers. Retrieval accuracy tracks exact-match answer quality strongly on extractive SQuAD but weakly on synthesis-style Telecom answers scored by cosine similarity.

3. Adaptive retrieval and compression

SAC-RAG [5] classifies each question as Fact, Definition or Reasoning, applies hybrid BM25 plus dense retrieval with BGE reranking, pure dense retrieval, or sub-question decomposition respectively, and then compresses the context with an LLM. A four-way ablation on HotpotQA, 2WikiMultiHopQA and MuSiQue shows that adaptive retrieval drives quality gains and compression drives token savings: tokens fall 38–58% relative to Adaptive RAG for a 0.5–1.6 point EM drop, while per-instance time rises from about 9.6 s to about 27.8 s on HotpotQA. The classifier reaches 92.0% accuracy on 300 labelled questions.

4. RAG versus fine-tuning

Budakoglu and Emekci [6] compare RAG, fine-tuning and a hybrid on SQuAD, MS MARCO and SQL CREATE TABLE data using a 12,500-triplet subset, bge-small-en embeddings, ChromaDB and GPT-3.5 Turbo. Fine-tuning scores higher on ROUGE and METEOR for SQuAD and MS MARCO; the hybrid is best on cosine similarity (0.89 on SQL) and on faithfulness and answer correctness for SQuAD and SQL; RAG has the lowest initial CPU, data and training cost.

5. Permission-aware retrieval

Jeong and Lee [7] retrieve from a central vector store and then validate every retrieved document against the owning system's native IAM (GCP, AWS, Keycloak), avoiding policy merging. On HotpotQA (13,783 documents), a control group with a manually merged policy omitted one attribute and leaked a restricted document that the delegated approach correctly blocked. All eight user profiles achieved perfect permission matching. Answer quality rises with permission coverage (EM 0.24 to 0.47; F1 0.31 to 0.61). IAM latency grows from about 65 ms (top-1) to about 204 ms (top-20) with 20 workers and falls from about 421 ms to about 127 ms as workers increase from 1 to 15.

6. Real-time, safety-critical RAG

HSK-GraphRAG [8] stores fleet state as a graph whose edge weights decay as in Eq. 2, where S is a clamped signal-quality factor derived from RSSI, λ is a decay constant and τ is the time of the last update.

$$w(t) = S \cdot \exp(-\lambda(t - \tau)) \quad (2)$$



A deterministic firewall injects safety constraints (automatic mode, no emergency stop, FRESH or VALID tier) into LLM-generated Cypher queries. Reported results are about 3.8 ms mean graph-query latency (P99 7.7 ms; 0.53 ms at N = 500), zero ghost selections versus 24.5% for vector RAG and 8.2% for context stuffing, and roughly 65× fewer tokens than context stuffing at N = 50. The authors scope the guarantees to trusted telemetry and the modelled predicates.

Table 2: Literature survey matrix

Ref	Focus	Datasets	Key components	Primary metrics	Headline finding
[4]	Configuration benchmark	SQuAD (10k/30k/100k docs)	HNSW/IVF/ScaNN; Cos/IP/L2; Fusion/Hier./HyDE; BGE/MiniLM; gpt-4o-mini	R@k, MRR, nDCG, correctness, faithfulness, latency, tokens, cost	No universal winner; HNSW-IP-Fusion-MiniLM best quality; L2 breaks Fusion
[1]	Embedding quality	SQuAD; private Telecom	12 embeddings; FAISS; Llama-3.2-3B, Qwen2.5-3B	Retrieval accuracy (top-1); EM; cosine similarity	Largest embedding wins; correlation strong on SQuAD, weak on Telecom
[5]	Adaptive retrieval + compression	HotpotQA, 2Wiki, MuSiQue	LLM classifier; BM25+dense; BGE rerank; LLM compressor; gpt-5-mini	EM, F1, Recall@10, tokens	Adaptive drives quality; compression cuts tokens; latency rises
[6]	RAG vs fine-tuning vs hybrid	SQuAD, MS MARCO, SQL CREATE TABLE	bge-small-en; ChromaDB; GPT-3.5 Turbo	ROUGE, BLEU, METEOR, BERTScore, cosine, RAGAS-style metrics, resources	Hybrid best semantic and faithfulness; fine-tuning best lexical; RAG cheapest
[7]	Access control	HotpotQA (13,783 docs)	FAISS; GCP/AWS/Keycloak IAM adapters; gpt-4o-mini	Match (FP+FN = 0), EM, F1, latency	Exact policy enforcement; merged-policy baseline leaks
[8]	Real-time safety	Simulated 50–500 robots; EuRoC replay	Memgraph; decay weights; Cypher firewall; GPT-4-turbo	Latency, SVR, GSR, tokens	Deterministic filtering gives zero ghost selection at ms-level latency

V. Comparative Analysis

1. Quality evidence

Table 3 places the principal quality results side by side. The values come from different datasets and generators and must not be pooled.

Table 3: Principal quality results (as reported in the source papers)

Ref	Setting	Best configuration / result	Weakest configuration / result
[4]	SQuAD, 162 configs	HNSW-IP-Fusion-MiniLM: correctness 0.909, faithfulness 0.970, relevance 0.959, coverage 0.942	SCaNN-L2-Fusion-BGE (small): coverage 0.267, correctness 0.484



Ref	Setting	Best configuration / result	Weakest configuration / result
[1]	SQuAD / Telecom, top-1	Qwen3-Embedding-0.6B: retrieval about 71% / 77%; EM 54.05% (Llama), 56.82% (Qwen)	msmarco-distilbert-base-tas-b: retrieval 43.02% (SQuAD)
[5]	HotpotQA	Adaptive RAG: EM 0.611, F1 0.751; SAC-RAG: EM 0.595, Recall@10 0.881	Naive RAG: EM 0.560, Recall@10 0.787
[6]	SQuAD / MS MARCO / SQL	Hybrid cosine 0.54 / 0.58 / 0.89; hybrid answer correctness 0.82 (SQuAD), 0.80 (SQL)	RAG-only answer correctness 0.50 (SQuAD), 0.48–0.53 (MS MARCO)
[7]	HotpotQA, 8 users	Admin (100% coverage): EM 0.47, F1 0.61	No-access user: EM 0.24, F1 0.31
[8]	Simulated fleet	Ghost selection 0.0%; 0 safety violations within modelled predicates	Vector RAG: ghost selection 24.5%; context stuffing 8.2%

2. Cost and latency evidence

Table 4 lists cost and latency evidence. The figures measure different layers of the system and are not like-for-like.

Table 4: Cost and latency evidence

Ref	What is measured	Reported values	Notes
[8]	Graph-query layer only (excludes LLM)	Mean about 3.8 ms; P99 7.7 ms; P99 0.53 ms at N = 500; about 230 vs about 15,000 tokens at N = 50	Context-stuffing baseline measured at N = 5 (412 ms), extrapolated to about 4.1 s at N = 50
[4]	Latency per configuration	Lowest 1.736 (IVF-L2-Hierarchical); HyDE slowest, about 1.7× Fusion cost	Units inconsistent in the paper (ns in abstract and text, s in one section)
[5]	Per-instance wall-clock (HotpotQA)	Naive about 9.6 s; Adaptive about 21.5 s; SAC-RAG about 27.8 s	Extra LLM calls for classification, decomposition, compression
[5]	Average tokens (HotpotQA / 2Wiki / MuSiQue)	Naive 1,645 / 1,883 / 2,159; Adaptive 2,407 / 3,002 / 3,111; SAC-RAG 1,087 / 1,264 / 1,918	Savings vs Adaptive 54.8% / 57.9% / 38.3%
[7]	IAM permission checks	65–204 ms with 20 workers (top-1 to top-20); 421 to 127 ms as workers go 1 to 15	Caching disabled; E2E latency about 0.7–0.9 s
[6]	Resource profile (qualitative)	RAG: minimal initial CPU/memory, no training cost; fine-tuning and hybrid: high	Vector database cost applies to RAG and hybrid
[1]	Scalability (qualitative)	Larger embeddings raise index memory, indexing time and GPU footprint	No measured latency reported

A derived observation: relative to the naive baseline, SAC-RAG's token savings are about 34% (HotpotQA), 33% (2Wiki) and 11% (MuSiQue), because adaptive retrieval itself enlarges the context. The quoted 38–58% reduction therefore depends on the baseline chosen.

3. Findings by research question

RQ1 (design choices). The similarity metric and retriever interact strongly: Fusion reaches Recall@1 of about 0.72–0.75 with inner product but only 0.02–0.05 with L2, while HyDE stays between about 0.57 and 0.63 across metrics [4]. Retrieval strategy, not compression, drives Recall@10 gains of up to about nine points [5]. Embedding choice shifts retrieval accuracy by tens of



points, although small semantic-search models come within a few points of the largest [1]. Index choice mainly affects speed, and reranker choice is second-order [4]. Task type moderates all of this [1], [6].

RQ2 (trade-offs). Quality-oriented choices (HyDE, decomposition, cross-encoder reranking, LLM compression) add model calls and therefore latency and cost [4], [5]. Compression can lose distinguishing details: in one documented case it removed cues that separated two similar entities and flipped a correct answer [5]. Fine-tuning and hybrid approaches buy alignment at higher training and data cost [6]. Permission filtering adds a per-document round trip that parallelism largely recovers [7].

RQ3 (governance and safety). Only [7] and [8] treat governance and safety as first-class requirements, and both enforce policy deterministically outside the LLM: one by delegating authorisation to each data owner's IAM, the other by injecting safety predicates and freshness thresholds into queries. Both bound their guarantees by trust assumptions. The other four studies do not address access control, freshness or adversarial content.

4. Methodological observations

Reading the studies side by side revealed issues that readers should weigh. They are our observations from the source text and figures and should be re-checked against the originals.

Table 5 appraises the validity of each study on five points that matter for generalisation.

Table 5: Validity appraisal of the reviewed studies

Ref	Domain coverage	Comparison / base-lines	Repetition	Artefact availability	Main caveat
[4]	SQuAD only (three sizes)	162 internal configurations	Not stated in text reviewed	Not confirmed in text reviewed	Latency units; reranker labelling
[1]	SQuAD plus private Telecom set	12 embeddings, 2 generators	Averaged over 10 runs	Code repository stated	top-k = 1; 3B generators only
[5]	Three multi-hop QA datasets	Naive, Compress, Adaptive, SAC-RAG ablation	Single deterministic run	Not stated in text reviewed	5–26-item categories
[6]	SQuAD, MS MARCO, SQL (12,500-triplet subset)	RAG, fine-tuning, hybrid	Not stated in text reviewed	Not stated in text reviewed	Embedding, not LLM, fine-tuned
[7]	HotpotQA (13,783 docs)	Merged-policy control group	50 queries (latency); 1,000 questions (quality)	Public code repository	Ground truth defined by IAM; caching off
[8]	Simulation plus Eu-RoC replay	Context stuffing, vector RAG, Redis TTL	100 trials per endpoint; 50 per scenario	Raw logs on GitHub; engine promised	Extrapolated baseline; trusted telemetry

- **Units and tables in [4].** Latency is given in nanoseconds in the abstract and text but in seconds in one results section; nanoseconds are implausible for an end-to-end RAG query. BGE and MiniLM rows are numerically identical in the retrieval tables but differ in the reranking tables (e.g., Fusion on HNSW-cosine, medium dataset: R@1 0.733 versus 0.553

with BGE and 0.820 with MiniLM), which suggests the reranker label in the retrieval tables does not reflect an applied reranker.

- **Retrieval versus generation in [1].** The text says the lowest-retrieval embedding also gives the lowest exact match. As read from Fig. 2 and Table 2 of that paper, the lowest SQuAD retrieval belongs to msmarco-distilbert-base-tas-b (43.02%) whereas the lowest EM belongs to msmarco-distilbert-dot-v5 (37.26%), whose retrieval is moderate. The correlation is real but noisier than described. The study also uses top-k = 1 and 3B generators only.
- **Terminology and consistency in [6].** The fine-tuning step adapts the embedding model (bge-small-en) while generation uses GPT-3.5 Turbo, so it differs from fine-tuning the LLM. The abstract says the hybrid may not surpass fine-tuning, whereas the conclusion calls it superior. Only a 12,500-sample subset is used.
- **Sample sizes in [5].** Definition and Reasoning categories contain 5–26 questions per dataset, so per-category conclusions (e.g., Reasoning EM from 0.455 to 0.577 on HotpotQA) are fragile, and the Reasoning subsets differ between pipelines (22 versus 26). The comparison with other methods uses different backbone LLMs, and all results are single-run.
- **Ground truth in [7].** The Match metric compares IAM decisions with the policy used to configure the IAM, so perfect agreement mainly confirms correct integration. Caching, the main production latency tool, was disabled.
- **Baselines and scope in [8].** The context-stuffing baseline is extrapolated from $N = 5$; the hardware-in-the-loop study replays EuRoC micro-aerial-vehicle data rather than a ground-robot fleet; the reference implementation reportedly uses $\lambda = 2.0$ instead of the $\lambda = 5.0$ analysed; and the code is promised on acceptance.

5. Agreement and disagreement across studies

The studies agree that retrieval design matters: strategy and metric choices move recall far more than index choice [4], [5], and the embedding model changes retrieval accuracy materially [1]. They agree that extra pipeline stages cost time [4], [5], [7]. They are less consistent about where gains appear: adaptive retrieval improves EM by 3.3–5.1 points over naive retrieval [5], while embedding choice has a much smaller effect on synthesis-style answers [1]. On knowledge injection, [6] reports the hybrid as strongest on semantic and faithfulness metrics, whereas other work cited there [24] finds RAG stronger than fine-tuning for new knowledge; because the experimental designs differ, these results are not in direct conflict, but they show the question is open.

6. Selection guide

Table 6 is our synthesis of the evidence. It should be validated per use case, since both [1] and [4] stress case-by-case selection.

Table 6: Evidence-based configuration guidance

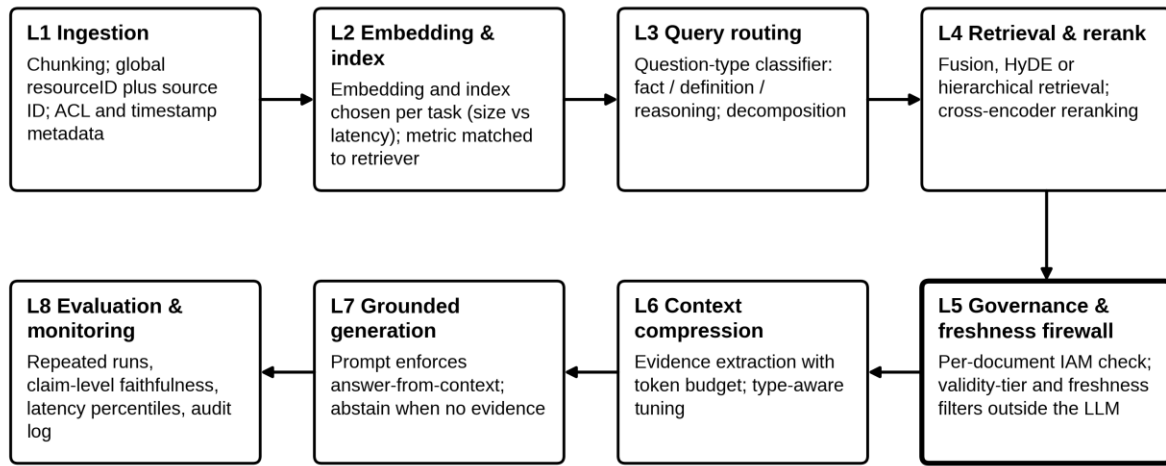
Requirement	Suggested choice	Supporting evidence
Maximum accuracy on extractive QA	High-capacity embedding; inner product or cosine; Fusion retrieval; cross-encoder reranker	[4], [1]
Lowest latency or cost	Smaller semantic-search embedding; IVF or ScaNN index; Hierarchical retrieval	[4], [1]
Sparse or underspecified queries	HyDE (accept higher cost)	[4]
Token-budget constrained	Adaptive retrieval plus compression; verify on target tasks	[5]
Structured or schema-like data	RAG or hybrid; expect higher scores	[6]

Requirement	Suggested choice	Supporting evidence
Multi-source, regulated data	Per-document IAM validation; avoid policy merging	[7]
Time-varying, safety-critical state	Freshness-weighted graph with deterministic constraint injection	[8]

VI. Proposed Conceptual System Architecture

Fig. 2 presents a conceptual eight-layer architecture that combines the components with the strongest evidence in the reviewed studies. It is a synthesis, not an experimentally validated system, and each layer should be treated as a hypothesis to be tested.

Conceptual eight-layer, governance-aware RAG architecture



Feedback: L8 results and audit logs recalibrate L2-L6 (index choice, routing, thresholds). Not validated experimentally; see Section 6.

Figure 2: Proposed conceptual eight-layer, governance-aware RAG architecture. Layers are justified by the studies discussed in Sections 4 and 5.

L1 Ingestion. Chunks carry a globally unique resource identifier, the source identifier and access-control and timestamp metadata, so that permissions and freshness can be checked later without re-indexing vectors [7]. **L2 Embedding and index.** The embedding and index are selected for the task, trading accuracy against latency, and the similarity metric is matched to the retriever because Fusion fails under L2 [4], [1]. **L3 Query routing.** A classifier routes questions by type and decomposes reasoning questions [5], with the caveat that evidence for Definition and Reasoning categories is thin.

L4 Retrieval and reranking uses Fusion, HyDE or Hierarchical retrieval with a cross-encoder reranker according to the accuracy–cost trade-off [4], [5]. **L5 Governance and freshness firewall** validates each retrieved item against its owner’s IAM and



applies validity filters before the LLM sees the content [7], [8]; this keeps safety decisions deterministic and outside the generator. L6 Context compression extracts evidence within a token budget, with type-aware intensity as future work [5]. L7 Grounded generation instructs the generator to answer only from the supplied context and to abstain when no evidence passes the firewall [1]. L8 Evaluation and monitoring logs retrieval, firewall and answer outcomes and feeds results back to recalibrate L2–L6 [18].

Three design principles follow from the evidence: enforce policy deterministically outside the LLM; keep policy and freshness metadata separate from embeddings so that they can change without re-indexing; and treat every added model call as a latency cost that must be justified by measured gains.

VII. Evaluation and Validation Framework

To avoid the optimistic bias visible in several reviewed studies, we propose the following minimum protocol for future RAG evaluations.

- Fix and report the generator. Compare retrieval configurations under an identical generator, prompt and decoding settings, and repeat with at least one additional generator.
- Separate retrieval and answer metrics. Report Recall@k and MRR alongside EM/F1, and add claim-level faithfulness checked against retrieved evidence [18].
- Use realistic k. Evaluate $k > 1$ and multi-hop questions; top-1 retrieval can exaggerate the role of the retriever [1].
- Repeat runs and report variance. Report means with confidence intervals over repeated runs and significance tests across configurations.
- Hold out documents and questions. Keep test questions and their source documents out of any tuning or fine-tuning data.
- Report cost end to end. Give latency percentiles (P50, P95, P99) including LLM inference, tokens per query and monetary cost, with caching conditions stated.
- Test governance and freshness. Include permission changes, revocation, IAM outages and stale-content cases, and adversarial inputs such as prompt injection through retrieved text [23].
- Publish artefacts. Release code, configurations and raw logs, and state all unit conventions.

VIII. Research Gaps and Challenges

Table 7 summarises the gaps supported by the evidence. Priorities reflect our judgement of impact and tractability.

Table 7: Evidence-supported research gaps

ID	Gap	Evidence	Priority
G1	No common benchmark across pipeline stages	Six studies use different datasets, generators and metrics (Tables 3–4); general embedding benchmarks [19], [20] do not measure generation impact, as [1] notes	High
G2	Weak measurement of correctness and hallucination	[1] computes no hallucination rate and relies on cosine similarity; [6] relies partly on lexical metrics; judge reliability is not reported	High
G3	Retrieval-to-generation link unclear beyond extractive QA	Strong for SQuAD, weak for Telecom, with top-k = 1 [1]; recall gains translate unevenly into EM [5]	High
G4	End-to-end efficiency rarely reported consistently	LLM time excluded [8]; caching disabled [7]; inconsistent units [4]; latency roughly triples in [5]	Medium



ID	Gap	Evidence	Priority
G5	Security, privacy and adversarial robustness unexamined	Four studies ignore these threats [4], [1], [5], [6]; [8] and [7] assume trusted telemetry and identities; injection [23] and leakage [22] are known risks	High
G6	Limited domain diversity and small evaluation sets	Mostly Wikipedia-based QA; one private 1,000-pair set [1]; simulation [8]; 5–26-item categories [5]	Medium
G7	Confounded RAG vs fine-tuning comparisons	Embedding fine-tuned rather than the LLM and a small subset [6]; other work [24] reports RAG stronger for new knowledge	Medium
G8	Knowledge freshness and corpus dynamics ignored	Only [8] models time; [6] uses a static set; no index-update or revocation tests ([7] lists them as open)	Medium
G9	Reproducibility and statistical rigour	Single runs [5]; no variance [1]; code promised [8]; few confidence intervals	High

Future Research Directions

- Unified, generator-controlled benchmarks (G1, G4). Evaluate retrieval, compression, generation, latency, tokens and cost jointly, extending BEIR- and MTEB-style suites [19], [20] to end-to-end RAG.
- Automatic configuration selection (G1, G3). The 162-configuration space [4] suggests learning policies that choose index, metric, retriever and reranker from task and corpus features.
- Learned routing and type-aware compression (G6). SAC-RAG uses a hand-written mapping over three types and one compression prompt [5]; learned routers (compare [10]) and per-type compression remain to be tested.
- Better evaluation of correctness and hallucination (G2). Combine exact-match checks, claim-level faithfulness, human evaluation and judge calibration with variance reporting.
- Governance-aware retrieval at scale (G5, G8). Combine metadata pre-filtering with post-retrieval IAM validation, TTL caching with explicit revocation, and health checks with safe fallbacks [7].
- Freshness-aware retrieval beyond robotics (G8). Decay-weighted validity tiers [8] may generalise to policies, prices or incident status, but need domain-specific calibration.
- Security evaluation suites (G5). Test prompt injection [23], identity spoofing, replay and data exfiltration [22] against deterministic-firewall designs.
- Controlled RAG, fine-tuning and hybrid studies (G7). Match data budgets and generators, tune both retriever and generator, and measure knowledge injection and style adaptation separately [24].
- Broader domains and modalities (G6). Evaluate on regulated, multilingual, multimodal and long-document corpora.
- Efficient local deployment (G4). Use small local models for classification and compression to reduce the latency penalty seen in [5], and edge-deployed graph partitions [8].

Limitations of this Review and Unresolved Questions

The corpus is small and drawn from a single venue, so selection bias is possible, and no database-level search was logged. Findings from different papers are not numerically comparable, and we did not re-run any experiment; performance figures are reported as published. Several observations in Section 5.4 rest on our reading of figures and tables and should be checked against the originals. Reference identifiers (DOIs and arXiv numbers) were taken from the source papers and not re-resolved online. Open questions for future work include: (1) how far the reported gains transfer to larger generators and longer contexts; (2) how much of the retrieval–answer correlation survives realistic k ; (3) what overhead governance and freshness checks add at production scale; and (4) which of the reported configurations remain best on proprietary, multilingual or multimodal data.



IX. Conclusion

This research review enhancing LLMs with external knowledge through RAG. Four conclusions follow. First, RAG quality depends on interacting design choices: the pairing of retriever and similarity metric can change recall by an order of magnitude [4], and embedding choice shifts retrieval accuracy by tens of points on extractive tasks [1]. Second, improvements carry costs: adaptive retrieval and compression trade tokens for latency [5], permission checks add per-document overhead [7], and hybrid approaches add training and data cost [6]. Third, safety, access control and freshness can be enforced deterministically outside the LLM [7], [8], but only within explicit trust assumptions. Fourth, the evidence base is fragmented, relies heavily on Wikipedia-style QA and small samples, and contains inconsistencies that limit comparison. Progress will depend on unified benchmarks, stronger measurement of correctness and hallucination, security-aware testing and controlled comparisons of knowledge-injection strategies. The proposed architecture and evaluation protocol are offered as starting points for that work.

Acknowledgements

We sincerely thank our respected guide and faculty members for their valuable guidance and support in completing this review paper. We are also thankful to Jaihind College of Engineering for providing the necessary resources. Finally, we thank all the researchers whose work helped us prepare this paper.

References

1. O. Amaral Cejas, Y. Guo, and Q. Tang, "From retrieval to response: Tracing the impact of embedding quality in RAG systems," *IEEE Access*, vol.13, pp. 212773–212781, 2025, doi: 10.1109/ACCESS.2025.3644595.
2. P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, pp. 9459–9474, 2020, doi: 10.5555/3495724.3496517.
3. Y. Gao et al., "Retrieval-augmented generation for large language models: A survey," 2023, doi: 10.48550/arXiv.2312.10997.
4. H. Elkiran and J. Rasheed, "EvaRAG: Evaluating advanced RAG techniques with indexing and distance metrics," *IEEE Access*, vol. 13, pp. 215724–215747, 2025, doi: 10.1109/ACCESS.2025.3646665.
5. D. Zhang, H. Yu, J. Huo, and M. Su, "SAC-RAG: Semantic adaptive context compression for retrieval-augmented generation," *IEEE Access*, vol. 14, pp. 117406–117418, 2026, doi: 10.1109/ACCESS.2026.3718025.
6. G. Budakoglu and H. Emekci, "Unveiling the power of large language models: A comparative study of retrieval-augmented generation, fine-tuning, and their synergistic fusion for enhanced performance," *IEEE Access*, vol. 13, pp. 30936–30951, 2025, doi: 10.1109/ACCESS.2025.3542334.
7. J. Jeong and S.-G. Lee, "Permission-aware RAG: Identity and access management (IAM)-based access filtering in multi-resource environments," *IEEE Access*, vol. 13, pp. 192819–192835, 2025, doi: 10.1109/ACCESS.2025.3628960.
8. A. Ibrahim, F. Al Machot, and K. Kyamakya, "Cyber-Physical GraphRAG: Safe, real-time, and scalable orchestration of large-scale autonomous fleets," *IEEE Access*, vol. 14, pp. 107177–107195, 2026, doi: 10.1109/ACCESS.2026.3706610.
9. V. Karpukhin et al., "Dense passage retrieval for open-domain question answering," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, pp. 6769–6781, 2020, doi: 10.18653/v1/2020.emnlp-main.550.
10. S. Jeong, J. Baek, S. Cho, S. J. Hwang, and J. Park, "Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity," in *Proc. NAACL-HLT*, pp. 7036–7050, 2024, doi: 10.18653/v1/2024.naacl-long.389.
11. A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024, doi: 10.48550/arXiv.2310.11511.



12. D. Edge et al., "From local to global: A graph RAG approach to query-focused summarization," 2024, doi: 10.48550/arXiv.2404.16130.
13. N. F. Liu et al., "Lost in the middle: How language models use long contexts," *Trans. Assoc. Comput. Linguistics*, vol. 12, pp. 157–173, Feb. 2024, doi: 10.1162/tacl_a_00638.
14. H. Jiang, Q. Wu, C.-Y. Lin, Y. Yang, and L. Qiu, "LLMLingua: Compressing prompts for accelerated inference of large language models," in *Proc. EMNLP*, pp. 13358–13376, 2023, doi: 10.18653/v1/2023.emnlp-main.825.
15. F. Xu, W. Shi, and E. Choi, "RECOMP: Improving retrieval-augmented LMs with compression and selective augmentation," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024, doi: 10.48550/arXiv.2310.04408.
16. P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proc. EMNLP*, pp. 2383–2392, 2016, doi: 10.18653/v1/D16-1264.
17. Z. Yang et al., "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in *Proc. EMNLP*, pp. 2369–2380, 2018, doi: 10.18653/v1/D18-1259.
18. S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "RAGAS: Automated evaluation of retrieval augmented generation," 2023, doi: 10.48550/arXiv.2309.15217.
19. N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), Datasets and Benchmarks Track*, 2021, doi: 10.5555/3540261.3540346.
20. N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "MTEB: Massive text embedding benchmark," 2022, doi: 10.48550/arXiv.2210.07316.
21. S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," NIST, Gaithersburg, MD, USA, Tech. Rep. SP 800-207, Aug. 2020, doi: 10.6028/NIST.SP.800-207.
22. S. Zeng et al., "The good and the bad: Exploring privacy issues in retrieval-augmented generation (RAG)," in *Proc. Findings Assoc. Comput. Linguistics (ACL)*, pp. 4505–4524, Aug. 2024, doi: 10.18653/v1/2024.findings-acl.267.
23. K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," in *Proc. 16th ACM Workshop Artif. Intell. Secur.*, Copenhagen, Denmark, pp. 79–90, Nov. 2023, doi: 10.1145/3605764.3623985.
24. O. Ovadia, M. Brief, M. Mishaali, and O. Elisha, "Fine-tuning or retrieval? Comparing knowledge injection in LLMs," 2023, doi: 10.48550/arXiv.2312.05934.