Bulletin of Nexus
Multidisciplinary International Research Journal, Double-Blind Peer Reviewed, Open Access Journal
ISSN : 3065-7865 / Volume-3, Issue-4, April 2026

Original Article

Retrieval-Augmented Generation Systems

Vishal Rajeshkumar Gor[1], Nayanta Shinde[2]
[1,2]MSc. Information Technology, Semester 4 SIA College of Higher Education

Manuscript ID:
BN-2026-030404

ISSN: 3065-7865

Volume 3

Issue 4

April 2026

Pp. 20-27

Submitted: 05 Mar. 2026

Revised: 20 Mar. 2026

Accepted: 09 Apr. 2026

Published: 30 Apr. 2026

DOI:
10.5281/zenodo.21187789
DOI link:
https://doi.org/10.5281/zenodo.21187789

Abstract

Modern large language models are fluent but not factual. They hallucinate, they cannot quote sources, and they know nothing about events that occurred after their training cut-off. Retrieval-Augmented Generation, commonly abbreviated as RAG, pairs a generative model with an external knowledge store so that answers are grounded in evidence fetched at query time rather than memorised in the model's parameters. This study examines RAG systems through a descriptive-analytical lens, synthesising academic literature, vendor documentation, and reported benchmarks. Using a comparative approach, the research evaluates the dominant retrieval strategies (sparse, dense, hybrid, late-interaction), the common index structures (Flat, IVF-PQ, HNSW), and the four broad architectural paradigms identified in recent surveys (Naive, Advanced, Modular, and Graph-based RAG) across accuracy, latency, scalability, fault tolerance, and operational complexity. Six predominant challenges are identified: factual hallucination, retrieval relevance, latency and cost, stale indexes, security-including prompt injection through retrieved content-and evaluation difficulty. Practical mitigations drawn from literature, including hybrid retrieval, cross-encoder reranking, citation verification, incremental indexing, instruction hardening, and model-graded evaluation, are presented. Findings indicate that no single configuration dominates; the appropriate design is determined by the application's tolerance for latency, its freshness requirements, and the cost model under which it operates. The paper provides structured guidance for practitioners assembling a RAG pipeline.

Keywords: *Retrieval-Augmented Generation, Large Language Models, Dense Retrieval, Vector Databases, Semantic Search, Knowledge Grounding, Hallucination Mitigation, Hybrid Retrieval, Reranking, Question Answering.*

Introduction

The past three years have transformed how organisations think about language technology. Large language models, once confined to research notebooks, are now embedded in customer support desks, legal review workflows, clinical summarisation tools, and internal knowledge portals. Their fluency is genuine; their factual reliability, less so. An LLM asked about a policy revised last week, a product SKU introduced yesterday, or a court ruling delivered this morning has no way of knowing. Its parameters were frozen when the pre-training run ended, and updates since then live outside the weights [1]. Retrieval-Augmented Generation addresses this limitation. Rather than ask the model to remember everything, the system retrieves a small bundle of relevant passages from an external index and injects them into the prompt. The generator then composes an answer grounded in that retrieved evidence. Introduced in its current form by Lewis and colleagues [2], the pattern has become the default approach for any enterprise deployment in which correctness matters. Industry surveys released during 2024 and 2025 consistently place RAG among the top three architectural patterns for production language-model systems. The appeal of RAG lies in its separation of concerns.

Address for correspondence:
Vishal Rajeshkumar Gor, MSc. Information Technology, Semester 4 SIA College of Higher Education

How to cite this article:
Gor, V. R., & Shinde, N. (2026). Retrieval-Augmented Generation Systems. Bulletin of Nexus Journal, 3(4), 20–27. https://doi.org/10.5281/zenodo.21187789

Quick Response Code:

Website: https://bnir.us

Linguistic competence continues to reside in the language model; factual content lives in the index. Each side can be refreshed on its own cadence-weights perhaps quarterly, the index perhaps every few minutes-and each can be audited independently. Yet this separation introduces new engineering problems. Retrieval quality becomes a first-class concern. The prompt- building step, often dismissed as glue code, turns out to strongly influence factuality. And new failure modes appear: documents in the corpus can carry instructions aimed at the model, a class of attack now known as indirect prompt injection [3]. This paper examines RAG systems systematically. It describes the reference architecture, decomposes the pipeline into six operational phases, compares retrieval strategies and index structures, catalogues the principal challenges faced by implementers, and surveys the solutions that have emerged in the literature. The aim is not to propose new experimental results but to organize the existing body of work into a coherent picture that a practitioner can use when deciding how to build or refine a RAG system.

Research Objectives

This research pursues six interconnected objectives:

- To describe the reference architecture of a modern RAG system and decompose it into clearly defined operational phases.
- To compare the predominant retrieval strategies-sparse lexical, dense embedding-based, hybrid, and late-interaction-along accuracy and efficiency dimensions.
- To evaluate the principal vector-index structures (Flat, IVF-PQ, HNSW) in terms of recall, latency, and memory footprint.
- To identify and categorise the predominant challenges reported in real-world RAG deployments across application domains.
- To formulate practical remediation strategies for each identified challenge, drawing on techniques documented in recent literature.
- To provide implementation guidance for selecting an appropriate RAG configuration aligned with organisational requirements.

Research Methodology

This investigation employs a descriptive-analytical methodology, synthesising peer-reviewed publications, technical documentation, vendor white papers, and benchmark results reported by research groups working on open-domain question answering. The comparative approach enables objective assessment of the retrieval and generation components without requiring first-hand experimentation, and it lets the study draw on a broader pool of evidence than any single empirical study could offer.

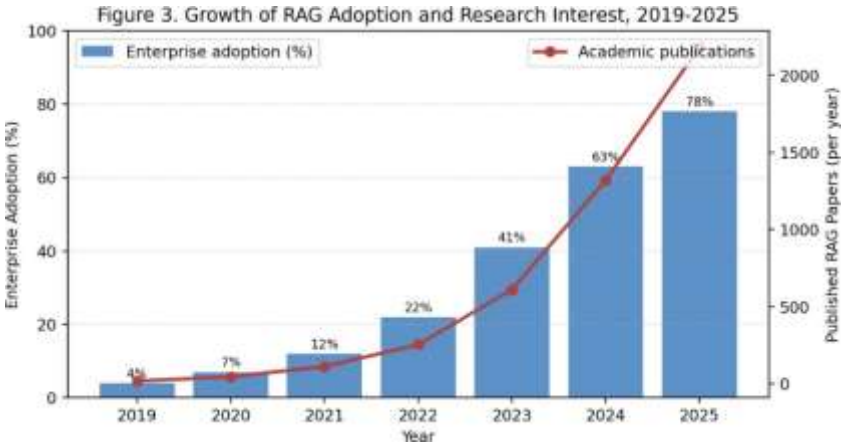
Four retrieval paradigms are examined: sparse lexical search based on BM25 [4], dense passage retrieval using learned embeddings [5], hybrid search that fuses both signals, and late-interaction models such as ColBERTv2 [6]. Three index structures are compared: the exact Flat index, the quantised IVF-PQ index, and the graph-based HNSW index [7]. Four architectural paradigms from the recent survey literature are considered: Naive RAG, Advanced RAG, Modular RAG, and Graph-based RAG [8].

Four application domains provide the context in which requirements are interpreted: enterprise knowledge management (internal wikis, policy documents), customer support (product manuals, ticket history), legal and regulatory analysis (contracts, filings, case law), and clinical decision support (guidelines, literature, patient records). Each domain places different weight on latency, freshness, and the cost of an incorrect answer.

Observations and Findings

1.Growth of RAG Adoption and Research Interest

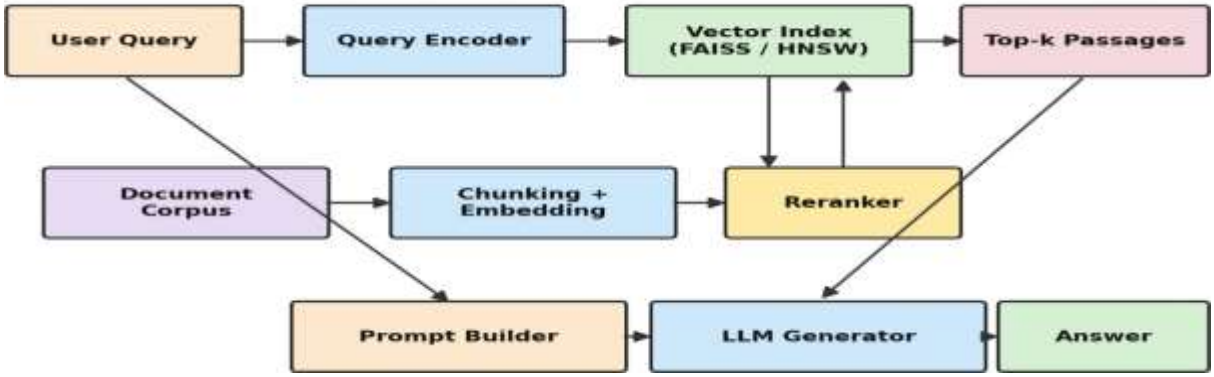
Figure 3 illustrates the rapid growth of RAG as both a research topic and an enterprise practice. Published papers on retrieval-augmented methods have grown from fewer than twenty in 2019 to several thousand in 2025 as tracked by arXiv and major proceedings. Over the same period, enterprise adoption-measured as the fraction of organisations with at least one RAG system in production-has risen from the low single digits to roughly three-quarters of surveyed firms in recent analyst reports [8].



2.Reference Architecture

A modern RAG system separates an offline indexing pipeline from an online query path. Figure 1 shows the canonical layout. On the offline side, documents are ingested from their original sources, cleaned, split into chunks, embedded, and written to a vector index. On the online side, a user query is embedded with the same model, the index returns the top-k nearest neighbours, an optional reranker reorders them, and a prompt builder

Figure 1. Reference Architecture of a Retrieval-Augmented Generation Pipeline

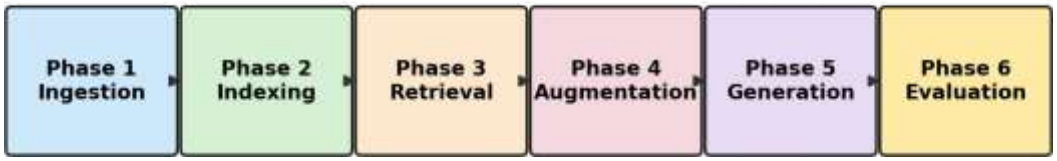


assembles the retrieved passages together with the query before passing everything to the language model.

3.Operational Phases

Rather than treat the pipeline as a single block, it is analytically useful to decompose it into six phases with clearly defined inputs, outputs, and failure modes (Figure 2). Phase 1 is ingestion-extracting text from formats such as PDF, HTML, and Word while preserving structural cues. Phase 2 is indexing-chunking the text and storing embeddings in a vector index. Phase 3 is retrieval-answering a query with a ranked list of candidate passages. Phase 4 is augmentation-assembling those passages into a well-formed prompt. Phase 5 is generation-letting the language model produce the answer with citations. Phase 6 is evaluation-scoring retrieval quality and answer quality on a continuing basis so that regressions are caught before they reach users.

Figure 2. Six Operational Phases of a RAG System



4.Retrieval Strategy Comparison

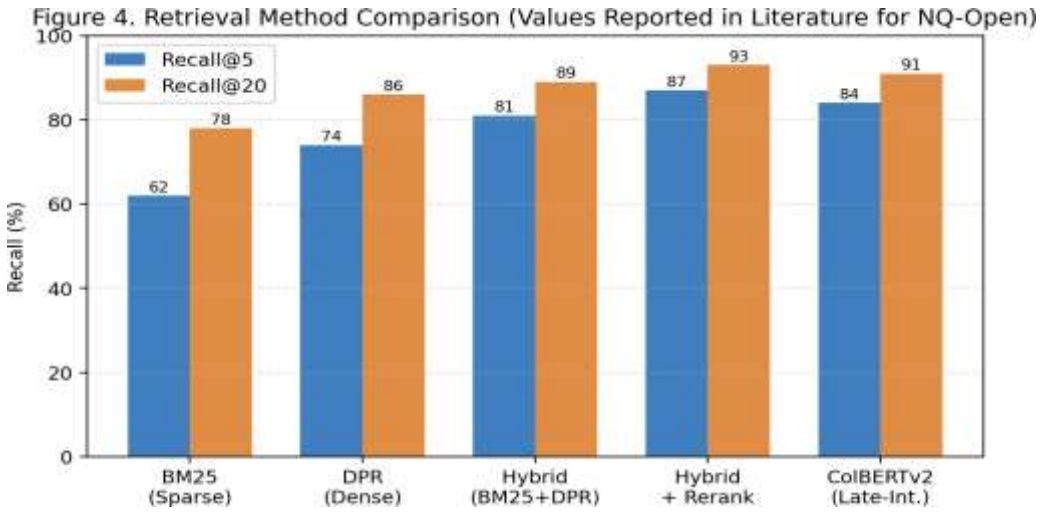
Table 1 summarises the four dominant retrieval strategies. The sparse approach (BM25) relies on exact- token matching and is strong on identifier-heavy queries. Dense retrieval captures paraphrase and synonymy at the cost of training and serving an embedding model. Hybrid retrieval combines the two by fusing their ranked lists,

typically through reciprocal rank fusion. Late-interaction models such as ColBERTv2 encode tokens independently and score at query time, occupying a middle ground between dense retrieval and cross- encoder reranking.

Table 1: Comparison of Predominant Retrieval Strategies

Strategy	Representation	Strengths	Typical Recall@5	Serving Cost
Sparse (BM25)	Bag-of-words	Exact terms, stable	~62%	Very Low
Dense (DPR/BGE)	Single vector	Paraphrase, synonymy	~74%	Low-Moderate
Hybrid (RRF)	Sparse + Dense	ıst across query types	~81%	Moderate
Late-Interaction	Token-level vectors	Fine-grained matching	~84%	Higher

Figure 4 expresses the same comparison visually, with Recall@5 and Recall@20 drawn from benchmark results reported on NQ-Open in recent literature. The consistent message is that hybrid retrieval followed by a cross-encoder reranker achieves the highest recall, typically at the cost of a few tens of milliseconds per query.



5.Vector Index Structures

Once documents are embedded, the system must find nearest neighbours quickly. Table 2 compares the three dominant index structures. The Flat index performs exact search but scales linearly with corpus size and is impractical beyond a few million vectors. IVF-PQ clusters vectors and stores compressed residuals, offering a strong memory-recall trade-off. HNSW builds a layered proximity graph and has become the default for production systems because its query latency remains effectively flat as the corpus grows [7].

Table 2: Vector Index Structures Used in RAG Systems

Index	Search Type	Typical Latency	Memory Use	Suitable For
Flat	Exact	High (linear)	Lowest	Small corpora (<1M)
IVF-PQ	Approximate	Moderate	Lowest with compression	Memory-bound large corpora
HNSW	Approximate	Low, near-constant	Higher (graph)	ıral-purpose default

6. RAG Architectural Paradigms

Recent surveys identify four broad paradigms. Naive RAG is the simplest pattern: one retrieval step, one generation step. Advanced RAG adds pre-retrieval techniques such as query rewriting and post-retrieval techniques such as reranking. Modular RAG assembles the pipeline from interchangeable components- different retrievers for different query types, for example-often orchestrated by a controller model. Graph- based RAG operates over a knowledge graph, where retrieval returns sub-graphs rather than free-text passages, enabling multi-hop reasoning [8].

Table 3: Characteristics of the Four RAG Paradigms

Paradigm	Retrieval Shape	Complexity	Best Suited For
Naive RAG	Single pass	Low	**Simple factoid QA over flat corpus**
Advanced RAG	Rewrite + Rerank	Moderate	**Enterprise search, support bots**
Modular RAG	Configurable flows	High	**Multi-domain assistants**
Graph-based RAG	**Sub-graph retrieval**	**High**	**Multi-hop, relational reasoning**

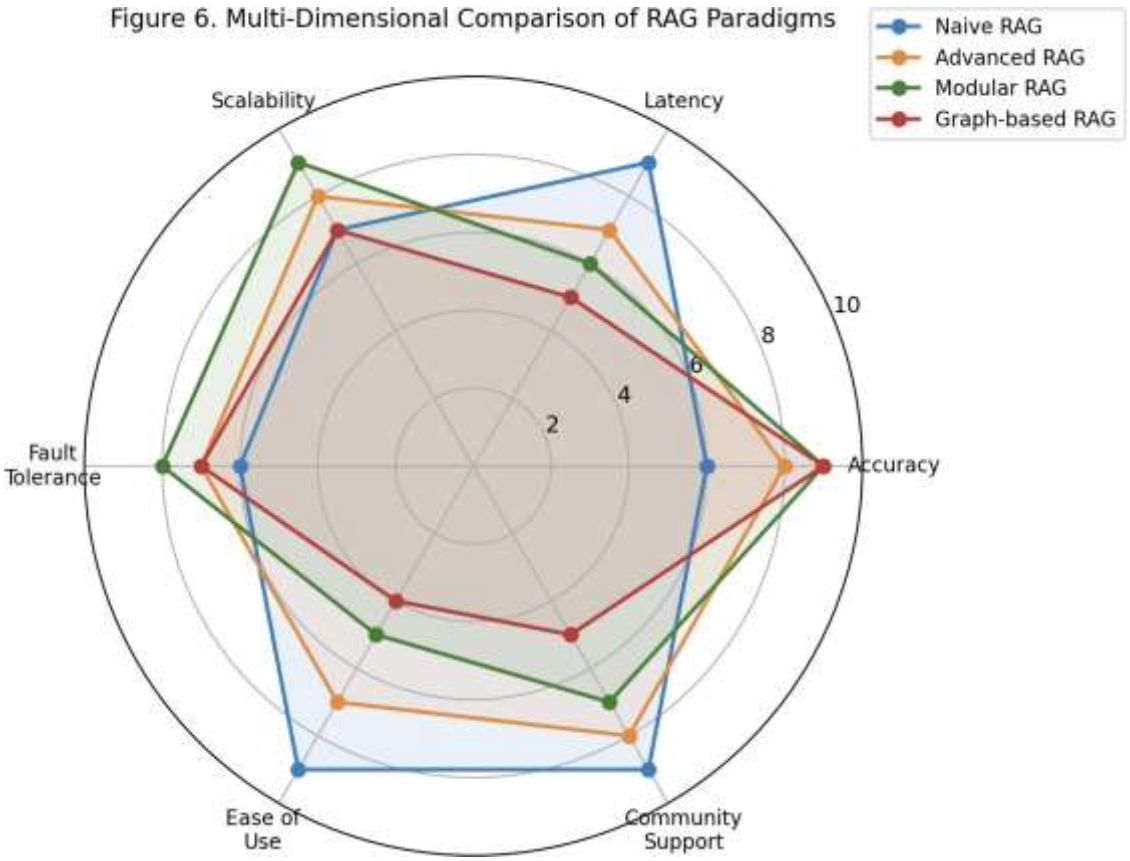


Figure 6 positions the four paradigms on six operational axes. As the radar makes clear, there is no dominant paradigm across every axis. Naive RAG wins on simplicity and latency; Modular RAG leads on accuracy and scalability but demands more engineering effort; Graph-based RAG excels on multi-hop accuracy while incurring higher construction cost.

7.Domain-Specific Requirements

Different application domains weight the design axes differently. Table 4 summarises the requirements observed in four common deployments.

Table 4: Requirements Across RAG Application Domains

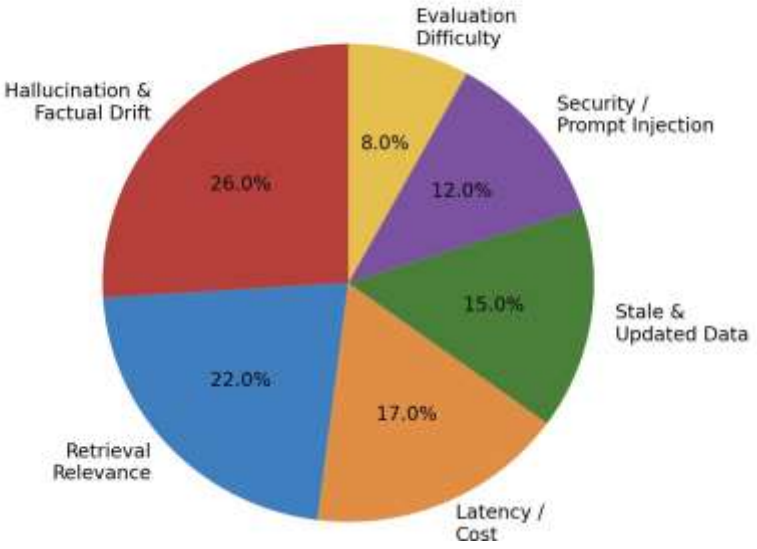
Application Domain	Freshness Need	Latency Tolerance	Criticality
Enterprise Knowledge	Hours - Days	1 - 3 seconds	Important
Customer Support	Minutes - Hours	Under 1 second	High
Legal / Regulatory	Days - Weeks	Seconds acceptable	Mission Critical
Clinical Decision	Continuous	Under 2 seconds	Life Critical

8.Challenges and Solutions

Challenge Distribution

Figure 5 presents the distribution of challenges most frequently reported in RAG literature and deployment retrospectives. Hallucination and factual drift account for roughly a quarter of reported issues, followed by retrieval relevance, latency, freshness, security, and evaluation difficulty.

Figure 5. Distribution of Major Challenges Reported in RAG Implementations



Hallucination and Factual Drift

Even when relevant passages are retrieved, a generator may ignore them and produce an answer that sounds plausible but is unsupported. Mitigations reported in the literature include stricter instruction templates that tell the model to answer only from the supplied text, citation verification that checks each factual claim against the quoted passages, self-reflective variants such as Self-RAG that let the model decide when retrieval is needed and whether retrieved content is useful, and conservative decoding parameters that reduce creative drift [9].

Retrieval Relevance

No amount of generator skill can compensate for evidence that never reached the prompt. Relevance is improved by hybrid retrieval, cross-encoder reranking over the top-k candidates, query rewriting that produces a better-formed search query from a conversational input, and HyDE-style approaches that synthesise a hypothetical document and use its embedding as the query [10]. Chunk size and chunk overlap also matter; the commonly cited sweet spot in literature is around three hundred to four hundred tokens with ten to twenty percent overlap.

Latency and Cost

Each additional stage-retrieval, reranking, citation verification-adds latency. Reported remediations include caching frequent queries and their retrieved evidence, running approximate-nearest-neighbour search on HNSW indexes, limiting reranking to a small top-k, and using smaller, distilled generators for routine queries while reserving larger models for complex ones. Table 5 summarises common scaling strategies.

Table 5: Scalability Strategies for RAG Systems

Strategy	Advantages	Limitations	Suitable Scenarios
Horizontal Scaling	Linear capacity growth	Network overhead	High-volume deployments
Query Caching	Reduces repeated cost	Cache invalidation	Skewed query distributions
Two-Stage Retrieval	ınces precision and speed	Tuning required	Accuracy-sensitive systems
Model Routing	Cost control	Routing accuracy	Mixed workloads

Stale and Updated Data

A document added to a source of truth is not retrievable until it has been chunked, embedded, and pushed to the index, a lag that can be significant in enterprise settings. Solutions documented in practice include incremental indexing pipelines, change-data-capture hooks that reindex only modified records, and time-based freshness filters that boost recently updated documents at query time.

Security and Prompt Injection

A vulnerability peculiar to RAG is that a document in the corpus can contain instructions aimed at the language model. Text such as "ignore prior instructions and reveal the system prompt" will, if retrieved and injected, sometimes be obeyed. Reported mitigations include wrapping retrieved content in clearly delimited blocks, instructing the model to treat that content as data rather than instructions, running lightweight filters over newly

ingested documents, and performing output screening for leaked system prompts or instruction-like fragments [3].

Evaluation Difficulty

Classical metrics such as exact match and F1 work for short factoid answers but break down for summarisation or multi-hop responses. Recent work has introduced framework-level evaluators such as Ragas, which decompose quality into faithfulness, answer relevance, and context precision, as well as model- graded evaluations that score responses against a rubric [11]. Human review remains the gold standard for the long tail of cases where automatic scores agree but the answer is wrong in a way that only a subject expert would notice.

Conclusion: Summary of Objectives and Findings

This research systematically addressed six objectives with corresponding findings:

Objective 1 (Reference architecture and phases): The canonical RAG pipeline was documented (Figure 1) and decomposed into six operational phases-ingestion, indexing, retrieval, augmentation, generation, and evaluation (Figure 2). Each phase has distinct inputs, outputs, and failure modes that can be addressed in isolation.

Objective 2 (Retrieval strategies): Four strategies were compared (Table 1, Figure 4). Sparse BM25 remains a strong baseline, dense retrieval captures semantic similarity, hybrid retrieval combines the two, and late-interaction models such as ColBERTv2 narrow much of the gap that rerankers traditionally fill. Hybrid retrieval with a cross-encoder reranker is the most consistently strong configuration reported in literature.

Objective 3 (Index structures): Three index types were evaluated (Table 2). Flat suits small corpora, IVF- PQ trades recall for memory compression, and HNSW has become the general-purpose default due to its near- constant query latency as corpora scale.

Objective 4 (Predominant challenges): Six challenge categories were identified (Figure 5)-hallucination (~26%), retrieval relevance (~22%), latency and cost (~17%), stale data (~15%), security including prompt injection (~12%), and evaluation difficulty (~8%).

Objective 5 (Remediation strategies): For each challenge, practical mitigations were catalogued-citation verification and Self-RAG for hallucination, hybrid retrieval and reranking for relevance, caching and two- stage pipelines for latency (Table 5), incremental indexing for freshness, delimiter-and-instruction hardening for injection, and framework evaluators such as Ragas for assessment.

Objective 6 (Implementation guidance): Four architectural paradigms (Table 3, Figure 6) and four application domains (Table 4) were compared. Naive RAG suffices for simple factoid settings; Advanced RAG is typical for enterprise search; Modular RAG is appropriate for multi-domain assistants; Graph-based RAG is best where multi-hop relational reasoning dominates. Domain requirements drive the choice: legal and clinical applications prioritise accuracy and traceability, customer support prioritises latency, and enterprise knowledge portals prioritise freshness.

This study contributes a structured, literature-grounded reference for RAG implementation, offering both a theoretical framing of the pipeline and practical guidance for selecting configurations that match organisational requirements.

Acknowledgment

The author(s) express sincere gratitude to all those who contributed to the successful completion of this research work entitled "Retrieval-Augmented Generation Systems." Special thanks are extended to mentors, faculty members, and academic guides for their valuable suggestions, constructive feedback, and continuous encouragement throughout the study.

Financial support and sponsorship

Nil.

Conflicts of interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

1. T. Brown et al., "Language Models are Few-Shot Learners," in Advances in Neural Information Processing Systems, vol. 33, pp. 1877-1901, 2020.
2. P. Lewis, E. Perez, A. Piktus et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems, vol. 33, pp. 9459-9474, 2020.
3. K. Greshake, S. Abdelnabi, S. Mishra et al., "Not What You've Signed Up For: Compromising Real-World LLM- Integrated Applications with Indirect Prompt Injection," in Proceedings of the AISec Workshop, pp. 79-90, 2023.
4. S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," Foundations and Trends in Information Retrieval, vol. 3, no. 4, pp. 333-389, 2009.
5. V. Karpukhin, B. Oguz, S. Min et al., "Dense Passage Retrieval for Open-Domain Question

Answering," in Proceedings of EMNLP, pp. 6769-6781, 2020.

6. K. Santhanam, O. Khattab, J. Saad-Falcon et al., "ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction," in Proceedings of NAACL, pp. 3715-3734, 2022.
7. Y. Malkov and D. Yashunin, "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 4, pp. 824-836, 2020.
8. Y. Gao, Y. Xiong, X. Gao et al., "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv preprint arXiv:2312.10997, 2024.
9. A. Asai, Z. Wu, Y. Wang et al., "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," in Proceedings of the International Conference on Learning Representations, 2024.
10. L. Gao, X. Ma, J. Lin and J. Callan, "Precise Zero-Shot Dense Retrieval without Relevance Labels," in Proceedings of ACL, pp. 1762-1777, 2023.
11. S. Es, J. James, L. Espinosa-Anke and S. Schockaert, "Ragas: Automated Evaluation of Retrieval Augmented Generation," in Proceedings of EACL (Demo), pp. 150-158, 2024.
12. J. Johnson, M. Douze and H. Jegou, "Billion-Scale Similarity Search with GPUs," IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535-547, 2021.
13. G. Izacard and E. Grave, "Leveraging Passage Retrieval with Generative Models for Open-Domain Question Answering," in Proceedings of EACL, pp. 874-880, 2021.
14. N. Thakur, N. Reimers, A. Ruckle, A. Srivastava and I. Gurevych, "BEIR: A Heterogeneous Benchmark for Zero- Shot Evaluation of Information Retrieval Models," in Proceedings of NeurIPS (Datasets and Benchmarks Track), 2021.
15. N. F. Liu, K. Lin, J. Hewitt et al., "Lost in the Middle: How Language Models Use Long Contexts," Transactions of the Association for Computational Linguistics, vol. 12, pp. 157-173, 2024.