International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(20s) 2026 pp. 710-720
https://doi.org/10.70917/ijcisim-2026-5213
Article

Retrieval-Augmented Generation: Architecting Trustworthy Knowledge Systems on Top of Large Language Models

Balakrishna Sreeram

Yahoo!, USA
ORCID ID: https://orcid.org/my-orcid?orcid=0009-0005-7367-3534

Abstract: Fluency and reliability are not the same property, and large language models have far more of the first than the second. A model's factual content is fixed at the point training concludes; its errors are distributed through billions of parameters rather than isolated in any inspectable location; and no signal in its output distinguishes a memorized fact from a well-phrased guess. These three properties — staleness, hallucination, and opacity — are not independent defects but different symptoms of the same design choice: knowledge stored implicitly in weights instead of explicitly in a retrievable structure. Retrieval-augmented generation (RAG) responds by splitting question answering into two stages, one that retrieves candidate evidence from an external, updatable corpus, and one that conditions generation on that evidence rather than on parametric memory alone. This review treats RAG as a systems-engineering discipline. Beginning from the specific failure modes that motivate retrieval augmentation, the discussion works through the component design of a production pipeline — ingestion, chunking, embedding, indexing, hybrid retrieval, and reranking — before turning to the evaluation methodology needed to determine whether a deployed system is actually behaving reliably. The literature surveyed spans dense passage retrieval, approximate nearest-neighbor search, vector database engineering, and automated RAG evaluation, and it converges on a claim central to this review: retrieval quality, not generator choice, is what governs whether a RAG system can be trusted. Consequences for high-stakes domains — finance, healthcare, law — are discussed, along with adaptive retrieval, agentic RAG, and evaluation standardization as directions likely to shape where the field goes next.
Keywords: retrieval-augmented generation; large language models; vector search; embeddings; hallucination mitigation; semantic retrieval; knowledge grounding; hybrid search

1. Introduction

A question about last quarter's regulatory filing gets the same confident, grammatically polished answer from a large language model as a question about a well-documented historical event. Nothing in the surface form of either response signals whether the underlying claim was recalled from memory, inferred through pattern completion, or invented to fill a gap. That indistinguishability is a minor nuisance in a trivia chatbot. It is disqualifying in a financial content pipeline, a clinical decision aid, or a legal research tool, where the cost of an unverifiable claim is not embarrassment but liability.

Better training data will not close this gap on its own, because the gap is not an incidental defect — it is a structural consequence of how these models are built. A transformer trained to predict the next token compresses everything it "knows" into learned weights, and once training stops, that knowledge is frozen, distributed across parameters rather than isolated in any auditable location, and stripped of any built-in trace of provenance. Three separate-sounding problems turn out to be one design choice viewed from different angles.

The first face of that choice is hallucination: fluent, internally consistent text produced by the identical mechanism that produces correct answers, unsupported by any actual source [9,10]. The second is staleness — a new regulation, a revised figure, a personnel change, none of it visible to a model whose only conventional update path, retraining, is too expensive to trigger for every incremental change in the world. The third is opacity: even a correct answer carries no indication of why it is correct, leaving independent verification effectively impossible without outside corroboration.

Retrieval-augmented generation targets all three at once. Rather than asking the model to answer from memory, a RAG system retrieves passages relevant to the query from an external corpus at inference time and hands those passages to the model as grounding context before generation begins [1,2,3]. Knowledge is no longer bounded by a training cutoff, since the corpus updates independently of the model. Provenance becomes traceable, at least in principle, back to the specific passages that informed a given answer. And because the generation task shifts from free recall toward synthesis over supplied evidence, the incentive structure of the task itself changes.

That shift opens its own engineering surface, and this surface — not the language model sitting at the end of the pipeline — is what this review treats as the central object of analysis. Retrieval quality is set by a long chain of decisions largely unrelated to model selection: how documents are segmented, how segments are embedded, how the resulting index is queried, how lexical and semantic signals are fused, and how candidates are reranked before reaching the generator. A more capable language model does little to correct a retrieval stage that surfaces irrelevant or incomplete evidence. It may do worse than nothing, since a stronger generator can dress up bad retrieval in more convincing prose.

Four contributions follow from that framing:

1. The structural failure modes of unaided language models are used to ground the case for retrieval augmentation, rather than a generic appeal to improved accuracy.

2. The retrieval pipeline is decomposed into its constituent stages — ingestion, chunking, embedding, indexing, hybrid retrieval, reranking — with the design trade-offs at each stage examined against the literature.

3. Evaluation methodology specific to retrieval-augmented systems is surveyed, separating metrics that assess retrieval from metrics that assess generation, since collapsing the two into a single accuracy figure hides where a system is actually failing.

4. Broader implications extend the discussion to domains where an unverifiable answer is disqualifying rather than merely inconvenient.

Adaptive and agentic retrieval close the article as likely directions for where this argument goes next.

2. Related Work: Why Unaided Language Models Fall Short

2.1 The Structural Nature of Hallucination

"Making things up" is the informal description usually attached to hallucination, and it understates how ordinary the underlying mechanism actually is. Surveys of the phenomenon separate intrinsic hallucination, where generated content contradicts a supplied source, from extrinsic hallucination, where content cannot be checked against any source because none was ever provided [9]. A single generative process produces both categories: the next token predicted is whatever is statistically probable given the training distribution and current context, with no separate step verifying the claim against an authoritative record. Confidence in the continuation has nothing to do with whether the continuation is true.

Instruction-tuned and retrieval-augmented systems do not escape this pattern; more recent surveys find that hallucination changes shape rather than disappearing once external context enters the picture [10]. A model conditioned on retrieved passages can still ignore that evidence in favor of parametric memory, conflate details drawn from multiple passages, or extrapolate past what any single passage actually supports. For evaluation, this matters

directly: grounding context reduces the opportunity for hallucination without eliminating the mechanism that produces it, so faithfulness to retrieved context has to be measured on its own terms rather than assumed as an automatic side effect of retrieval.

None of this makes hallucination a problem that can be patched at the model layer in isolation. Early open-domain question-answering work showed that a generator supplied with passages retrieved through a dense bi-encoder produced measurable gains over generation from parametric memory alone, establishing retrieval as a constraint on what the model is asked to produce rather than a supplement bolted onto an otherwise unchanged process [2,3]. Dense passage retrieval demonstrated specifically that a retriever trained to align question and passage embeddings in a shared vector space could outperform classical sparse retrieval on open-domain benchmarks — a result that reframed retrieval quality as a first-class design variable rather than a preprocessing detail [2]. Subsequent architectures pushed the coupling further, training retriever and generator jointly or semi-jointly so the retrieval signal adapts to what generation actually requires [1,8]. What runs through this literature is a consistent pattern: grounding narrows the surface area for hallucination, but it does not remove the need to verify, at the output level, that the model actually used what it was handed.

2.2 Knowledge Staleness and the Cost of Retraining

Describing staleness is easier than solving it. A model's factual content is fixed at the point its training corpus was assembled, and any event or publication after that point simply does not exist for the model unless reintroduced through further training. Ordinary use rarely exposes this limitation for stable, well-documented facts. Financial disclosures, regulatory guidance, product documentation, security advisories — domains where the underlying knowledge moves weekly or daily — expose it constantly, and correcting it through retraining scales poorly against the pace of change.

Retraining a large model on a weekly cadence is not realistic; the computational cost rules it out short of a major version release, and continual fine-tuning risks catastrophic forgetting, degrading performance on previously learned tasks in exchange for currency on new ones. What retrieval augmentation does instead is decouple what the model knows how to do from what it knows about the world. The corpus supplying grounding context can be updated by appending, replacing, or removing documents — an operation closer to a database write than a training run — while the underlying generator stays fixed [1,6]. Open-domain question answering and retrieval-augmented pretraining approaches built to exploit external context at both training and inference time have demonstrated this decoupling across a range of settings [6,8].

Reframing where "knowledge" needs to live changes what a generator is actually for. If an external index can supply facts and figures on demand, the generator's job shifts toward synthesis, reasoning over supplied evidence, and language production — properties that are comparatively stable and far less sensitive to how fresh any particular fact happens to be. Choosing a generator on the strength of its reasoning rather than its raw factual coverage follows directly from this shift, since factual coverage is exactly what retrieval is meant to supply from outside the model. Less obviously, it argues for treating the corpus itself, not the model, as the asset deserving the heaviest engineering investment: curation, versioning, and maintenance discipline applied to a retrieval corpus can let a modest generator outperform a stronger one paired with a poorly indexed source.

3. Architecture of a Retrieval-Augmented System

Distinct stages, each with its own failure modes and its own tuning surface — that is the shape of a retrieval-augmented pipeline once it is taken apart. Table 1 summarizes the stages and their principal design concerns, and Figure 1 first contrasts this two-stage architecture against unaided direct generation before Figure 2 lays out the complete pipeline. What follows walks through the pipeline in the order data actually moves through it.

Table 1. Stages of a Retrieval-Augmented Pipeline and Their Design Concerns

Pipeline Stage	Purpose	Key Design Concern

Ingestion & chunking	Convert raw source material into indexable, passage-sized retrievable units	Balancing chunk length and overlap against context preservation; handling messy source formats (tables, boilerplate, footnotes) without corrupting downstream retrieval
Embedding	Convert chunks into dense vector representations that place semantically related passages near each other	Embedding model choice and robustness under limited labeled supervision; capturing domain-specific terminology
Vector indexing	Make similarity search computationally tractable at production scale	Approximate nearest-neighbor algorithm choice (e.g., HNSW, GPU-based methods) and maintaining index freshness under continuous updates
Retrieval	Surface candidate passages relevant to a query by combining dense and sparse signals	Balancing dense/sparse fusion weighting; setting top-k so relevant passages are neither omitted nor buried
Reranking	Reorder the retrieved candidate set for precision before it reaches the generator	Cost/accuracy trade-off between cross-encoder and bi-encoder scoring; avoiding over-reliance on lexical overlap

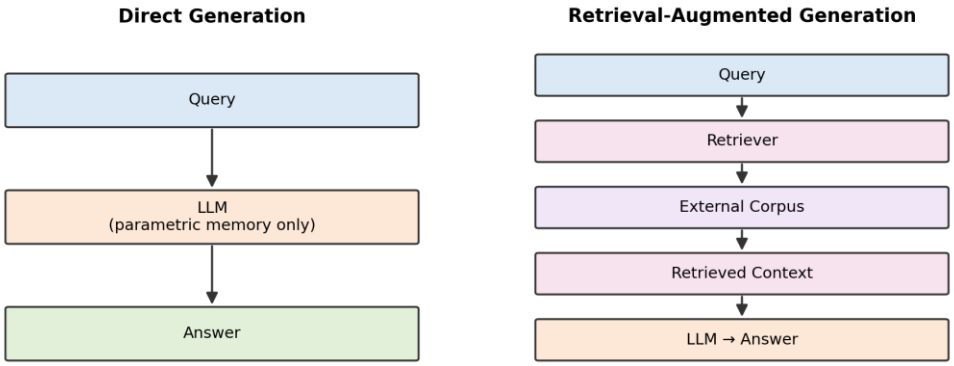

Fig. 1 - Direct generation from parametric memory alone (left) contrasted with retrieval-augmented generation, in which a retriever draws grounding context from an external corpus before the language model generates an answer (right).

3.1 Ingestion and Chunking

Source material has to be converted into an indexable form before any retrieval is possible. Ingestion covers extracting text from PDFs, HTML pages, structured database records, and internal wikis, then normalizing that text into a consistent representation. The stage absorbs far more engineering effort than its conceptual simplicity suggests, because production corpora arrive messy — tables, footnotes, boilerplate headers, inconsistent formatting — and all of it interferes with downstream chunking unless handled deliberately upfront.

Many of the practical failures in deployed RAG systems trace back to chunking, the step that divides ingested documents into passage-sized units for embedding and retrieval. Too short, and a chunk loses the context needed to stand on its own. Too long, and the specific information a query is targeting gets diluted by surrounding material the embedding model has to compress alongside it, weakening the resulting vector's discriminative power. Fixed-length chunking by token or character count remains the simplest and most widely used strategy, though it cuts across sentence and paragraph boundaries without regard for meaning, occasionally separating a claim from the qualification that immediately follows it. Semantic chunking instead segments along boundaries that preserve coherent units of

meaning, using similarity measures between adjacent sentences or passages to decide where a chunk should end, and this approach has been shown to improve the coherence of retrieved units relative to naive fixed-length splitting [17]. Chunk overlap, a related parameter, deliberately duplicates a small window of text between neighboring chunks so a fact spanning a boundary is not lost to either segment entirely.

How much of that retrieved context a generator can actually use is a separate question that interacts with chunking in ways easy to overlook. Long-context language models do not attend to all portions of a supplied context window equally; relevant information placed in the middle of a long context is used less reliably than information placed near the beginning or end, a pattern with direct consequences for how many chunks get retrieved and where the strongest candidates should sit in the assembled prompt [12]. Fewer, more precisely selected passages tend to outperform a larger number of loosely relevant ones as a result, since retrieving more context is no guarantee that a generator will attend to the passages that actually matter.

3.2 Embeddings and Vector Indexing

Dense vector representation is what each chunk becomes once chunking is finished, positioning semantically related passages near each other in a high-dimensional space regardless of whether they share surface vocabulary. This is what lets a query phrased differently from the source document still retrieve the right passage, a capability classical keyword search does not have by construction. Sentence-embedding architectures built on pretrained transformer encoders, refined to produce more semantically discriminative sentence-level vectors, form the standard building block for this stage [19,20], with cosine similarity between query and passage embeddings the typical relevance signal for both live retrieval and offline relevance analysis [16].

Retrieval quality downstream of anything happening in generation is shaped directly by which embedding model gets chosen. Contrastive training objectives, pulling semantically related passages together in vector space while pushing unrelated ones apart, have been shown to improve retrieval performance even with limited labeled supervision, extending dense retrieval quality into domains and languages where labeled question-passage pairs are scarce [6]. That matters for the systems-engineering argument at the center of this review: an organization deploying RAG against a specialized or proprietary corpus rarely has the large labeled retrieval dataset that early dense-retrieval benchmarks depended on, making robustness to limited supervision a practical engineering requirement rather than an academic footnote.

Indexing comes next, making similarity search computationally tractable at scale once passages are embedded, since comparing a query against every vector in a meaningfully sized corpus exhaustively is not viable at query-time latencies. Approximate nearest-neighbor algorithms trade a small amount of accuracy for large gains in speed, and two families currently dominate practice. Graph-based indexes — the hierarchical navigable small-world (HNSW) structure being the representative example — build a layered graph over the vector space so a query navigates toward its nearest neighbors through local hops rather than a global scan, and this approach has demonstrated a strong combination of search speed and recall across vector search benchmarks [5]. GPU-accelerated similarity search libraries extend the same idea by parallelizing distance computation across large vector collections, making search against corpora of a scale impractical on CPU-only infrastructure feasible [4]. Around these indexing techniques a broader ecosystem of vector database systems has emerged, handling metadata filtering, hybrid scoring, and index maintenance under continuous updates in addition to core nearest-neighbor search, and recent survey work catalogs the trade-offs among these systems across consistency, scalability, and query expressiveness [7]. An index in a production RAG deployment is never a static artifact built once and queried indefinitely — it absorbs additions, deletions, and updates continuously without degrading latency or search quality, which puts index maintenance inside the engineering scope of the system rather than treating it as a one-time offline step.

3.3 Hybrid Retrieval and Reranking

Dense vector retrieval captures semantic similarity well, but it is not uniformly superior to classical lexical retrieval. Specific identifiers, technical terms, product codes, and proper nouns are handled more reliably by sparse,

near-exact term matching than by an embedding model that may compress such terms into a vector representation without preserving the precision a lexical match would offer. Hybrid retrieval architectures exploit this complementarity, retrieving candidates from both a vector index and a traditional inverted index and merging the two ranked lists into one result set. Combining those two signals is itself a nontrivial decision: fusion functions that weight and merge dense and sparse relevance scores have been analyzed systematically, and findings indicate that the choice of fusion strategy measurably affects final retrieval quality, with no single approach dominating uniformly across query types [18].

A second retrieval pass, applied after an initial candidate set has been assembled, is what reranking introduces, typically using a more computationally expensive but more accurate relevance model to reorder a comparatively small candidate list before it reaches the generator. Operating over a few dozen candidates rather than an entire corpus lets reranking afford cross-encoder architectures that jointly encode the query and each candidate passage together, capturing interaction effects a bi-encoder comparison cannot represent, at a cost that would be prohibitive applied to the full corpus at initial retrieval time. Some approaches invert the conventional matching direction entirely, generating candidate questions from indexed passages and matching those against the user's actual query, specifically to improve alignment between how a corpus is indexed and how users phrase questions in practice, reporting improvements in retrieved-passage relevance for question-answering tasks over more conventional passage-matching baselines [13].

Domain-specific engineering pays off most visibly at the retrieval and reranking stages, since embedding and ranking models tuned on open-domain benchmarks do not automatically carry their performance characteristics over to specialized corpora. Public-facing web deployments built around retrieval-augmented question answering have reported that careful tuning of retrieval and prompt-assembly parameters produced more reliable answers than swapping in a larger underlying language model [14]. Human health risk assessment, a domain demanding that retrieved evidence trace back to specific regulatory or scientific sources, has produced comparable findings: grounding answers in retrieved documents materially improves reliability relative to an ungrounded baseline, provided retrieval is tuned to the vocabulary and document structure specific to the domain [15]. Open-domain conversational response generation built on retrieval augmentation has reported a similar pattern — the quality of retrieved knowledge, more than adjustments to the response generator, governs whether generated responses stay grounded over a multi-turn exchange [21]. Figure 2 depicts the complete pipeline described across this section, from offline document ingestion, chunking, embedding, and indexing through query-time retrieval, reranking, context assembly, and generation. What recurs across these deployments is consistent with the central claim running through this review: incremental gains in retrieval and reranking quality translate into system-level reliability gains that upgrading the generator alone struggles to reproduce.

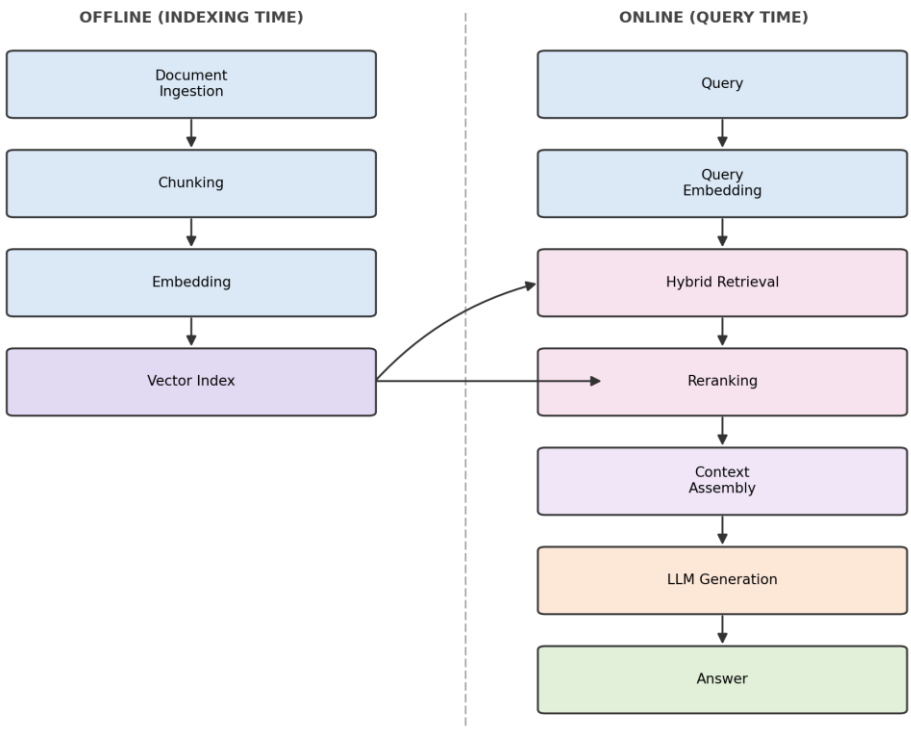

Fig. 2 - The complete retrieval-augmented pipeline, separating offline indexing-time stages (document ingestion, chunking, embedding, vector indexing) from online query-time stages (query embedding, hybrid retrieval, reranking, context assembly, and LLM generation).

4. Evaluation Methodology for Retrieval-Augmented Systems

4.1 Why Stage-Separated Evaluation Is Essential

Two structurally different failures can produce the identical visible symptom in a retrieval-augmented system: an answer that is simply wrong. Retrieval can fail to surface the passages that actually contain the answer, leaving the generator to work with insufficient or irrelevant material that no amount of prompting sophistication will recover. Or retrieval can succeed entirely — surfacing exactly the passages needed — while generation fails anyway, ignoring supplied evidence in favor of parametric memory, misreading a passage, or overreaching past what the retrieved material actually supports. End-to-end answer correctness alone cannot tell these two failure modes apart, and without that distinction, an engineering team has no way to know whether the next unit of effort belongs in the corpus, the chunking strategy, and the retriever, or in prompt design and generation parameters instead.

Stage-separated evaluation is the methodological answer to that ambiguity, formalized in automated RAG evaluation frameworks that decompose system quality into retrieval-side and generation-side metrics computable without hand-labeled ground-truth answers for every query, using a reference language model as evaluator over retrieved context, generated answer, and original question together [11]. The case for separating stages only strengthens once hybrid retrieval and reranking layers enter the pipeline, since every additional stage is a fresh candidate location for failure — an ingestion problem, a chunking problem, an embedding mismatch, an indexing lag, a fusion-function imbalance, a reranking error, or a generation-side failure to use good context correctly could each equally explain a poor end-to-end accuracy score. Diagnosing which one is responsible, absent decomposed metrics, reduces to manual inspection of individual failure cases, an approach that does not scale as a system moves from prototype into production.

Why generation-side evaluation cannot simply be inferred from retrieval-side success is reinforced by the long-context finding that models underuse information sitting in the middle of a supplied context window [12]. A retriever might correctly surface the passage containing the answer — an evaluation checking only retrieval recall would register that as a clean success — while the generator, receiving that passage buried among several others, fails to draw on it and produces an unsupported answer regardless. Only a metric examining specifically whether the generated answer is faithful to the retrieved context would catch this, which is exactly why faithfulness has to stand as its own evaluation dimension rather than something inferred from retrieval recall.

4.2 The Four Quality Dimensions

Four metrics recur across the RAG evaluation literature, two aimed at the retrieval stage and two at the generation stage [11]. Context recall asks whether the retrieved passage set actually contains what is needed to answer the query, typically assessed by checking whether claims in a reference answer can be attributed to at least one retrieved passage. A low score here points to a retrieval or corpus problem specifically: either the needed information is absent from the indexed corpus entirely, or it exists and the retriever failed to surface it — a distinction that matters for deciding whether the fix belongs in corpus curation or retriever tuning.

The cost of over-retrieval is what context precision captures instead, measuring the proportion of retrieved passages that are actually relevant to the query. A retriever chasing strong context recall by returning a large number of passages can achieve it while burying the useful ones among irrelevant material, and given the long-context attention pattern discussed above, that burying can itself degrade generation quality even though the needed information was technically retrieved. Context recall and context precision together characterize the retrieval stage on its own terms, isolated from anything the generator subsequently does with the material it receives — which is exactly what lets an engineering team attribute a quality problem to retrieval specifically before looking anywhere else.

Faithfulness turns the question toward generation: are the claims in a generated answer actually supported by the retrieved context supplied to the model, regardless of whether the answer happens to be true in some external sense. A faithful answer can still be wrong if the retrieved context itself was flawed or incomplete, and an answer can be accurate by coincidence while remaining unfaithful to what was actually supplied — a distinction that matters because faithfulness is the property the generation stage can genuinely be held accountable for improving. Answer relevance, the fourth dimension, checks whether the generated answer addresses the question actually asked, independent of faithfulness; an answer can be entirely faithful to retrieved context and still miss the question, for instance by restating retrieved material rather than synthesizing it into a direct response.

Localizing failures is what these four metrics accomplish together — context recall and context precision pointing to the retrieval stage, faithfulness and answer relevance pointing to generation — giving an evaluation report the diagnostic resolution needed to direct engineering effort where it will actually move outcomes. Domain-specific RAG deployments, including retrieval-augmented health-risk question answering and web-facing retrieval applications, report that measuring these four dimensions separately surfaces retrieval-stage weaknesses that end-to-end accuracy metrics alone would obscure, reinforcing that evaluation methodology is itself part of the systems-engineering discipline argued for here, not an afterthought bolted onto a system once it is otherwise complete [14,15].

5. Discussion — Broader Implications

What accountability means for a deployed language-model application changes once retrieval quality is treated as the primary determinant of trustworthiness. An incorrect answer from a purely parametric model offers no natural point of intervention short of retraining or discarding the response, since the error sits somewhere among billions of learned parameters with no way to isolate, correct, or audit it in place. A retrieval-augmented system that produces an incorrect answer, in contrast, offers two inspectable artifacts: the passages supplied as context, and the generated answer's relationship to those passages. Accountability infrastructure, rather than an opaque prediction engine, is what this structural difference makes possible — not because the underlying language model has become more

interpretable, but because the surrounding system has been designed to expose a verifiable chain from source document to retrieved passage to generated claim.

That chain of verifiability carries the most weight in domains where an unsupported answer disqualifies a system rather than merely inconveniencing a user. A generated summary of a company's earnings disclosure or regulatory filing in financial services has to trace back to the specific filing language it summarizes, because a fabricated or misattributed figure carries direct compliance and reputational consequences; a retrieval-augmented pipeline surfacing the exact source passage alongside its summary gives a reviewer something concrete to check, where a purely parametric summary offers nothing equivalent. Clinical and regulatory sources grounding retrieval-augmented question answering in healthcare have been proposed for the same reason: a generated answer can be checked against the specific document it claims to draw from, a property directly relevant to risk-assessment and decision-support contexts where an unverifiable answer cannot responsibly be acted upon [15]. Statutes, case law, and internal policy documents in legal research and compliance workflows follow the same logic — an answer's value depends less on how fluently it reads than on whether the source it claims to rely on actually says what the answer asserts.

Eliminating the need for human review in these domains is not what retrieval augmentation does; changing what that review consists of is. A reviewer equipped with the retrieved passages evaluates whether generation accurately represented what those passages say, a narrower and considerably more tractable task than verifying an ungrounded claim from first principles. Organizational investment shifts accordingly: a system architected around retrieval invites ongoing investment in corpus curation, source vetting, and index maintenance, rather than treating generator selection as the primary lever for improving reliability. Web-facing and domain-specific RAG deployments reported in the literature consistently frame retrieval and prompt-assembly tuning, not generator swaps, as the intervention that actually moved reliability outcomes, a pattern consistent with the broader claim that trustworthiness in these systems is substantially an information-architecture problem rather than a model-selection problem [14,15,21].

6. Conclusions

Retrieval-augmented generation, this review has argued, belongs on the drafting table as a systems-design problem rather than a question settled by picking the strongest available language model. Frozen parametric knowledge, hallucination generated by the identical mechanism that produces correct answers, and an absence of internal provenance are the structural limits of unaided language models, and retrieval augmentation addresses all three by externalizing knowledge into a maintained, updatable, and inspectable corpus [1,2,3,9,10]. System reliability tracks the quality of that external layer — chunking strategy, embedding and indexing choices, how dense and sparse retrieval signals are combined and reranked — more directly than it tracks generator choice, a pattern recurring across the architecture and evaluation literature surveyed here [4,5,6,7,11,12,13,14,15,17,18]. Stage-separated evaluation using context recall, context precision, faithfulness, and answer relevance gives that argument a diagnostic footing, distinguishing retrieval-stage failures from generation-stage failures rather than collapsing both into a single opaque accuracy figure [11].

A handful of directions look likely to carry this argument further. Adaptive retrieval, where a system decides dynamically whether retrieval is even needed for a given query and how many passages to pull, may cut down the waste of retrieving a fixed passage count regardless of query complexity. Agentic RAG architectures — issuing multiple retrieval queries iteratively, reformulating the search based on intermediate results before a final answer gets generated — extend the retrieval-generation separation argued for here into something closer to a deliberative process than a single retrieve-then-generate pass. Evaluation standardization remains unfinished business: frameworks already exist that decompose retrieval and generation quality, but comparing retrieval-augmented systems across organizations and domains still depends on protocols that are not yet uniformly adopted [11]. Whichever of these directions gains ground first, the central claim of this review is likely to hold regardless: trustworthiness in a retrieval-

augmented system is built primarily at the retrieval and corpus layer, with the generator's role reduced to synthesizing faithfully from what that layer supplies.

Acknowledgments: Not applicable.

Funding: This research received no external funding.

Author Contribution: The author is solely responsible for the conception, literature review, analysis, and writing of this article.

Conflict of Interest: The author declares no conflict of interest.

Data Availability: No new data were generated; all sources analyzed are cited in the reference list.

Biographical Sketch

Balakrishna Sreeram is a Senior Backend Engineer (AI) at Yahoo Finance, where his work centers on backend services and AI/LLM-powered content platforms. He brings over a decade of backend and data-engineering experience across financial and enterprise platforms, having held engineering roles at Strategic Staffing Solutions (Venerable), Harvey Nash (Vanguard), PDS Technologies (Bayer), BCforward, and Axis Bank. He holds an MS in Computer Science from San Francisco Bay University and a BS in Computer Science from Chhatrapati Shahu Ji Maharaj University. His current work applies retrieval-augmented and large-language-model techniques to financial content and data systems.

REFERENCES

1. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W. Yih, T. Rocktaschel, S. Riedel, D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems 33, pp. 9459-9474, 2020. Available: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html
2. V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, W. Yih, "Dense Passage Retrieval for Open-Domain Question Answering," Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pp. 6769-6781, 2020. DOI: 10.18653/v1/2020.emnlp-main.550 Available: https://aclanthology.org/2020.emnlp-main.550/
3. G. Izacard, E. Grave, "Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering," Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pp. 874-880, 2021. DOI: 10.18653/v1/2021.eacl-main.74 Available: https://aclanthology.org/2021.eacl-main.74/
4. J. Johnson, M. Douze, H. Jegou, "Billion-Scale Similarity Search with GPUs," IEEE Transactions on Big Data, Vol. 7, No. 3, pp. 535-547, 2021. DOI: 10.1109/TBDATA.2019.2921572 Available: https://ieeexplore.ieee.org/document/8733051/
5. Y. A. Malkov, D. A. 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. DOI: 10.1109/TPAMI.2018.2889473 Available: https://ieeexplore.ieee.org/document/8594636/
6. G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, E. Grave, "Unsupervised Dense Information Retrieval with Contrastive Learning," Transactions on Machine Learning Research, 2022. Available: https://openreview.net/forum?id=jKN1pXi7b0
7. J. J. Pan, J. Wang, G. Li, "Survey of Vector Database Management Systems," The VLDB Journal, Vol. 33, pp. 1591-1615, 2024. DOI: 10.1007/s00778-024-00864-x Available: https://link.springer.com/article/10.1007/s00778-024-00864-x
8. G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, E. Grave, "Atlas: Few-shot Learning with Retrieval Augmented Language Models," Journal of Machine Learning Research, Vol. 24, Article 251, pp. 1-43, 2023. Available: https://www.jmlr.org/papers/v24/23-0037.html
9. Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, D. Chen, W. Dai, H. S. Chan, A. Madotto, P. Fung, "Survey of Hallucination in Natural Language Generation," ACM Computing Surveys, Vol. 55, No. 12, Article 248, pp. 1-38, 2023. DOI: 10.1145/3571730 Available: https://dl.acm.org/doi/10.1145/3571730
10. L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, T. Liu, "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions," ACM Transactions on Information Systems, Vol. 43, No. 2, pp. 1-55, 2025. DOI: 10.1145/3703155 Available: https://dl.acm.org/doi/10.1145/3703155
11. S. Es, J. James, L. Espinosa Anke, S. Schockaert, "RAGAs: Automated Evaluation of Retrieval Augmented Generation," Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System

Demonstrations, pp. 150-158, 2024. DOI: 10.18653/v1/2024.eacl-demo.16 Available: https://aclanthology.org/2024.eacl-demo.16/
12. N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, P. Liang, "Lost in the Middle: How Language Models Use Long Contexts," Transactions of the Association for Computational Linguistics, Vol. 12, pp. 157-173, 2024. DOI: 10.1162/tacl_a_00638 Available: https://aclanthology.org/2024.tacl-1.9/
13. B. Saha, U. Saha, M. Z. Malik, "QuIM-RAG: Advancing Retrieval-Augmented Generation With Inverted Question Matching for Enhanced QA Performance," IEEE Access, Vol. 12, pp. 185401-185410, 2024. DOI: 10.1109/ACCESS.2024.3513155 Available: https://ieeexplore.ieee.org/document/10781379/
14. I. Radeva, I. Popchev, L. Doukovska, M. Dimitrova, "Web Application for Retrieval-Augmented Generation: Implementation and Testing," Electronics, Vol. 13, No. 7, Article 1361, 2024. DOI: 10.3390/electronics13071361 Available: https://www.mdpi.com/2079-9292/13/7/1361
15. W. Meng, Y. Li, L. Chen, Z. Dong, "Using the Retrieval-Augmented Generation to Improve the Question-Answering System in Human Health Risk Assessment: The Development and Application," Electronics, Vol. 14, No. 2, Article 386, 2025. DOI: 10.3390/electronics14020386 Available: https://www.mdpi.com/2079-9292/14/2/386
16. K. Venkatesh Sharma, P. R. Ayiluri, R. Betala, P. J. Kumar, K. Shirisha Reddy, "Enhancing Query Relevance: Leveraging SBERT and Cosine Similarity for Optimal Information Retrieval," International Journal of Speech Technology, Vol. 27, pp. 753-763, 2024. DOI: 10.1007/s10772-024-10133-5 Available: https://link.springer.com/article/10.1007/s10772-024-10133-5
17. C. Kiss, M. Nagy, P. Szilagyi, "Max-Min Semantic Chunking of Documents for RAG Application," Discover Computing, Vol. 28, Article 117, 2025. DOI: 10.1007/s10791-025-09638-7 Available: https://link.springer.com/article/10.1007/s10791-025-09638-7
18. S. Bruch, S. Gai, A. Ingber, "An Analysis of Fusion Functions for Hybrid Retrieval," ACM Transactions on Information Systems, Vol. 42, No. 1, Article 20, 2023. DOI: 10.1145/3596512 Available: https://dl.acm.org/doi/10.1145/3596512
19. B. Wang, C.-C. J. Kuo, "SBERT-WK: A Sentence Embedding Method by Dissecting BERT-Based Word Models," IEEE/ACM Transactions on Audio, Speech, and Language Processing, Vol. 28, pp. 2146-2157, 2020. DOI: 10.1109/TASLP.2020.3008390 Available: https://ieeexplore.ieee.org/document/9140343/
20. J. Seo, S. Lee, L. Liu, W. Choi, "TA-SBERT: Token Attention Sentence-BERT for Improving Sentence Representation," IEEE Access, Vol. 10, pp. 39119-39128, 2022. DOI: 10.1109/ACCESS.2022.3164769 Available: https://ieeexplore.ieee.org/document/9749108/
21. Y. Ahn, S.-G. Lee, J. Shim, J. Park, "Retrieval-Augmented Response Generation for Knowledge-Grounded Conversation in the Wild," IEEE Access, Vol. 10, pp. 131374-131385, 2022. DOI: 10.1109/ACCESS.2022.3228964 Available: https://ieeexplore.ieee.org/document/9982598/