| Volume 02 | Issue 04 | September 2026| ISSN: 3139-1737

International Journal of Advances and Applications in Convergent Studies & Transdisciplinary (IJ-AACST)

Bridging Disciplines, Bridging Knowledge

Aligned with the Vision of NEP 2020 and UGC Guidelines

Website: https://aacstij.org || **Email:** journaleditor@aacstij.org

A Study: Architecture of Retrieval-Augmented Generation (RAG) Integrating Orchestration and Multi-Tiered Evaluation

DOI: https://doi.org/10.5281/zenodo.22338947

Ms. V. V. Thakare[1*], Dr. V. S. Tondre[2]

*Corresponding Author: [1]Brijlal Biyani Science College, Amravati.
Department of Computer Science & Application, Affiliated to SGBAU Amravati, Maharashtra, India.
thakarevaishali3@gmail.com

ORCID iD: https://orcid.org/0009-0001-1357-4527:

[2]Brijlal Biyani Science College, Amravati.
Department of Computer Science & Application, Affiliated to SGBAU Amravati, Maharashtra, India.
vstondre@gmail.com

ABSTRACT

Large Language Models (LLMs) exhibition innovative appropriate facility, their distribution in mission-critical applications is constrained by parametric vision and static data cutoff limits. Retrieval-Augmented Generation (RAG) avoids these limitations by vigorously training propagation interpreters on non-parametric suggestion removed from peripheral document stores. Though, current applications regularly smart from cascading architectural disappointment methods: coarse semantic chunking, retrieval noise propagation, prompt-level context dilution, and non-deterministic evaluation. This paper reviews an end-to-end operational framework that combines modular pipeline orchestration via LangChain with high-granularity observability and runtime execution tracing via LangSmith. By separating concerns diagonally text ingestion, vector indexing, cross-encoder reranking, and structured prompt synthesis, our architecture enables targeted component isolation. Additionally, validate a multi-tiered valuation classification that decouples retrieval diagnostics from generation reliability. The anticipated architecture offers an observed design for methodically evolving, benchmark, and continuing production-grade grounded language systems.

KEYWORDS: Retrieval-Augmented Generation (RAG), Pipeline Orchestration, LLM (Large Language Model) Observability, Distributed Tracing, Grounded Evaluation, Multi-stage retrieval, Vector database.

1. INTRODUCTION

Parametric memory in bottomless auto-regressive language models is fundamentally static and computationally exclusive to inform. After organized diagonally high-precision fields as well as procedural provision, initiative agreement, health information processing, and legal discovery normal representations display stochastic unpredictability, displayed as realistic manufacture, oversight of grave edge circumstances, and hallucinated source attribution [1].

Retrieval-Augmented Generation (RAG) resolves these boundaries by dissociating constant intellectual capability from field packing. By enquiring an exterior non-parametric corpus at interpretation time, RAG structures language models as conditional workstations over verified source indication [1].

Despite the intangible precision of RAG, real-world distribution offerings major engineering contests. RAG is not a lonely model, but a symbiotic, multi-stage circulated system. System disaster can stem from arbitrary passage boundary definitions, low-recall dense vector lookup, cross-attention degradation over long situations, or procreative unfaithfulness contempt correct relative injection [1], [2].

To address these challenges, this paper presents a systems-level architecture that integrates:

1. **Modular Pipeline Composition:** Implementing configurable pipelines for ingestion, structural partitioning, hybrid retrieval, and prompt assembly using LangChain [3].
2. **Diagnostic Runtime Observability:** Instrumenting execution chains with LangSmith to isolate intermediate latency, token consumption, and retrieval artifact propagation [4].
3. **A Multi-Tiered Evaluation Paradigm:** Decoupling retrieval precision from generative faithfulness to establish deterministic root-cause analysis for system failures.

2. END TO END MODULAR SYSTEM ARCHITECTURE

The proposed architecture organizes the RAG life cycle into distinct, loosely coupled layers, ensuring that individual components can be parameterized, benchmarked, and swapped without requiring downstream pipeline restructuring [3].

Stage 1: Document Ingestion and Normalization

Heterogeneous enterprise data—such as PDFs, structured records, Markdown documentation, and web pages—is extracted using format-specific parsers. Non-informative elements like repeated headers, formatting noise, and broken encodings are stripped, while structural metadata, including document hierarchy, creation dates, and authorization tags, is preserved [4].

Stage 2: Semantic Chunking Engine

Because language models operate with finite context windows, large documents must be divided into smaller passages. The architecture employs two primary strategies:

- **Recursive Character Splitting:** Text is segmented along natural semantic boundaries, such as paragraph breaks, sentence stops, and white-space, enforcing maximum token limits while preserving coherent ideas [4].
- **Structure-Aware Chunking:** Text is split according to internal document layout, such as section headings, bullet lists, or tables, embedding high-level headings into each chunk's metadata to preserve local context [5].

Stage 3: Vector Indexing and Semantic Representation

Normalized text chunks are transformed into continuous semantic vector representations using dense embedding models. These vectors are indexed in high-performance vector databases using approximate nearest-neighbor search graphs, enabling rapid semantic similarity matching across large document repositories.

Stage 4: Hybrid Retrieval and Cross-Encoder Reranking

To balance broad keyword matching with conceptual semantic similarity, the retrieval layer executes a two-stage search strategy:

- **Candidate Pool Generation:** The system combines dense vector similarity search with sparse keyword indexing to return a wide candidate pool of potentially relevant passages.
- **Neural Cross-Encoder Reranking:** A cross-encoder model evaluates the query alongside each candidate passage simultaneously, assessing deep contextual relevance to rank the most pertinent passages at the top of the candidate set [5].

Stage 5: Context-Aware Prompt Construction

The highest-ranked passages are collected into an organized prompt model together with the user's initial query [3]. The prompt integrates severe operative orders, lacking the model to originate its answers right from the providing source channels and explicitly announce when the suggestion is inadequate to answer the query.

Stage 6: Conditional Generation

The language model produces the accumulated circumstances into a easy, grounded response. Rather than replication raw channels, the generator précises, cross-references, and presentations the retrieved facts into a direct answer [4].

Stage 7: Telemetry and Observability Layer

Every implementation step is captured by an observability layer power-driven by LangSmith. This layer records query transformations, retrieved chunk identifiers, similarity scores, token counts, latency metrics, and model outputs.

3. SYSTEM EXECUTION FLOW

The workflow below illustrates the sequential progression of data from raw document intake through observability-backed inference:

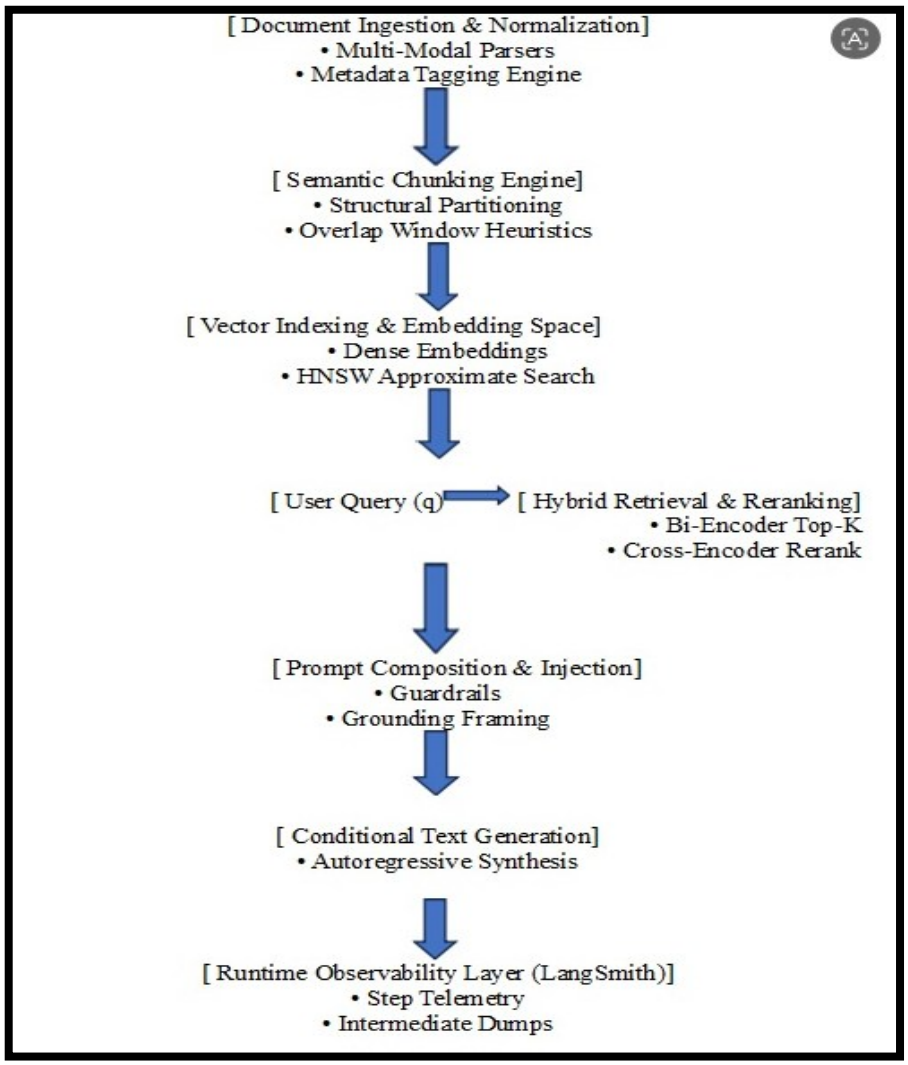

Fig 3.1: RAG System Data Execution Workflow [5].

4. MULTI-TIERED EVALUATION FRAMEWORK

The evaluation architecture separates retrieval efficacy from generation fidelity to independently diagnose stage-specific bottlenecks.

A. Retrieval Performance Assessment

Retrieval metrics evaluate whether the search system gathers complete and accurate supporting evidence before generation begins:

- **Context Precision:** Assesses the proportion of retrieved passages that contain genuine supporting facts relative to irrelevant or noisy content.

- **Context Recall:** Measures whether the search process succeeded in surfacing all necessary reference materials required to answer the query completely.
- **Rank Accuracy:** Evaluates whether the most critical and relevant evidence is placed at the top of the context block, preventing positional bias in language models [5], [6].

B. Generation Fidelity Assessment

Generation metrics verify whether the synthesized response remains strictly faithful to the retrieved documents:

- **Attributed Faithfulness:** Measures the degree to which every claim within the model output can be traced back to the retrieved source text, identifying ungrounded claims and hallucinations.
- **Answer Relevance:** Checks whether the generated response directly addresses the user's intent without drifting into tangential topics.
- **Negative Rejection:** Verifies that the model correctly refuses to guess or fabricate answers when the retrieved context lacks sufficient information [5].

C. Diagnostic Failure Triage

Decoupling these evaluation tiers enables clear operational diagnosis:

- **High Retrieval Precision with Low Faithfulness:** Indicates the retrieval stage functioned correctly, but the generative model hallucinated or ignored the context, pointing to needed adjustments in system prompts or decoding parameters [7].
- **Low Retrieval Precision with Low Faithfulness:** Points to an upstream information retrieval failure, requiring improvements in document parsing, chunking strategy, or vector search.
- **High Retrieval Precision with Low Relevance:** Indicates that although the system found the right facts, the prompt formatting failed to guide the model toward the user's core intent.

5. DISCUSSION AND ENGINEERING CONSIDERATIONS

Designing enterprise RAG systems requires balancing competing system constraints:

1. **Passage Granularity versus Context Integrity:** Extremely short text chunks isolate precise facts and improve semantic vector search, but risk separating related sentences. Larger chunks maintain broader context but introduce irrelevant text into the prompt window, which can dilute model focus [6].
2. **Search Depth versus Runtime Latency:** Incorporating multiple search layers, such as hybrid retrieval and neural reranking, improves context quality but increases query latency [4]. Production systems must balance search depth with response speed through semantic caching and asynchronous processing.

3. **Continuous Grounding through Observability:** Instrumenting execution chains with platforms like LangSmith adds operational transparency, allowing engineering teams to audit failure logs, monitor latency bottlenecks, and evaluate prompt adjustments systematically [8].

6. LIMITATIONS

While modular RAG architectures improve language model reliability, several engineering limitations remain:

- **Dependency on Corpus Integrity:** RAG systems inherit errors, conflicting statements, or outdated details present in the source documents.
- **Multi-Document Deductive Reasoning:** Standard retrieval pipelines struggle with complex questions that require multi-step reasoning across dispersed documents unless guided by iterative query rewriting or knowledge graph integration [7].

Evaluation Variance: Automated evaluation workflows that use secondary language models to grade outputs can introduce their own scoring inconsistencies, requiring ongoing human review in specialized domains.

7. CONCLUSION

Retrieval-Augmented Generation provides an effective foundation for deploying large language models within specialized knowledge domains. By combining modular pipeline composition through LangChain with deep runtime observability via LangSmith, the proposed architecture establishes an inspectable, maintainable, and verifiable system design. Furthermore, decoupling retrieval diagnostics from generation quality provides a practical methodology for systematically diagnosing pipeline failures. Future work will explore augmenting vector indices with structured knowledge graphs and agentic self-reflection routines to improve complex multi-step reasoning.

8. REFERENCES

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 9459–9474, 2020.

[2] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, “REALM: Retrieval-augmented language model pre-training,” in *Proc. 37th Int. Conf. Machine Learning (ICML)*, pp. 3929–3938, 2020.

[3] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, “Dense passage retrieval for open-domain question answering,” in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, 2020.

[4] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark *et al.*, “Improving language models by retrieving from trillions of tokens,” in *Proc. 39th Int. Conf. Machine Learning (ICML)*, pp. 2206–2240, 2022.

[5] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave, "Atlas: Few-shot learning with retrieval augmented language models," *Journal of Machine Learning Research (JMLR)*, vol. 24, no. 251, pp. 1–43, 2023.

[6] A. Asai, Z. Wang, J. Hwang, P. Singh, X. V. Lin, X. Chen, C. Xiong, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2024.

[7] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Transactions of the Association for Computational Linguistics (TACL)*, vol. 12, pp. 157–173, 2024.

[8] LangChain, "LangChain Framework Documentation," [Online]. Available: [https://docs.langchain.com/] (https://docs.langchain.com/) 2026.