AWS CloudGuide
- Your Cloud Architecture Assistant

Harshith Maddala
Jasreen Kaur Mehta
Mohammed Ismail Sarfaraz Shaik

The George Washington University

Abstract

This paper presents "AWS CloudGuide," an advanced Hybrid Retrieval-Augmented Generation (RAG) system designed to function as an AWS Technical Architect. Addressing the limitations of generic Large Language Models (LLMs) in handling domain-specific, up-to-date technical queries, we introduce a dual-path retrieval architecture. Our system ingests over 100,000 chunks of official AWS documentation (2025 User Guides) and indexes them using both semantic vector embeddings (BAAI/bge-m3) and sparse lexical keywords (BM25). We evaluate the efficacy of Reciprocal Rank Fusion (RRF) in combining these indices to retrieve precise answers for architectural, operational, and security queries. The system utilizes a Qwen-2.5-7B-Instruct model running in native FP16 precision on a single NVIDIA T4 GPU, optimizing for reasoning fidelity over compression. Our results demonstrate a Semantic Retrieval Score of 0.75 against a Golden Dataset, validating that Hybrid RAG significantly outperforms naive approaches in retrieving highly technical, syntax-sensitive information. We also detail a novel "Vector Integrity Scanner" developed to detect and repair database corruption during large-scale ingestion, ensuring system resilience.

Keywords: Retrieval-Augmented Generation (RAG), Hybrid Search, Vector Embeddings, Large Language Models (LLMs), AWS Cloud Architecture.

---

1. Introduction

The complexity of cloud infrastructure management has outpaced the efficiency of traditional information retrieval methods. AWS documentation spans thousands of PDFs, including User Guides, API References, and Best Practice whitepapers. Engineers seeking answers to specific questions—such as "What is the difference between a DynamoDB Scan and Query?"—often face a trade-off: spend hours manually filtering search results or rely on generative AI tools like ChatGPT, which may provide plausible but outdated or incorrect information.

To address this gap, we developed **AWS CloudGuide**, a specialized RAG agent. Unlike standard

chatbots, CloudGuide is grounded in a curated corpus of 2025 AWS User Guides. It employs a **Hybrid Retrieval Engine** that fuses the semantic understanding of dense vectors with the precision of keyword search. This ensures that users can find answers to both conceptual questions (e.g., "How do I optimize costs?") and specific technical queries (e.g., "What is the error code for ThrottlingException?").

Our primary contributions are:

1. **Hybrid Retrieval Architecture:** A robust implementation of Reciprocal Rank Fusion (RRF) combining ChromaDB (Vector) and BM25 (Keyword) indices to maximize retrieval recall and precision.
2. **Cognitive Query Processing:** A "Rewrite-Retrieve-Read" pipeline that resolves conversational ambiguities (e.g., "How do I configure **it**?") before executing a search.
3. **High-Fidelity Inference:** Deployment of a 7-billion parameter LLM in half-precision (FP16) on consumer-grade hardware, ensuring high-quality reasoning without the degradation associated with aggressive quantization.

---

2. Dataset

For this project, we utilized the AWS Public PDF Chunked Dataset hosted on Hugging Face, which contains text extracted from all publicly available AWS PDF documentation. This includes User Guides, Developer Guides, Architectural Best Practices, Security Whitepapers, and Service-Specific Manuals from the 2025 release cycle. Because AWS documentation is the authoritative source for cloud architecture and operations, this dataset is ideal for building a retrieval-augmented NLP system that must deliver accurate, up-to-date answers.

The dataset contains over 106,000 text chunks with a total size of 660 MB, organized in JSON format. Each record represents a small, self-contained segment of documentation with fields such as:

- text – the extracted content segment
- source – name of the original AWS PDF
- chunk_id – sequential ID within that document
- metadata – length and auxiliary fields for indexing

The AWS Public PDF Chunked Dataset is well-suited for retrieval-augmented generation because it provides a large-scale, trustworthy, and technically rich corpus derived entirely from official AWS documentation. Since most AWS PDFs are lengthy—often spanning hundreds of pages—the dataset applies a sliding-window chunking strategy, splitting each document into approximately 1000-character segments with a 200-character overlap. This ensures that context is preserved across page boundaries and that key concepts are not fragmented, which is essential for accurate semantic retrieval. The dataset covers over a hundred AWS services, including S3, EC2, IAM, DynamoDB, and Lambda, and contains detailed terminology such as API names, error codes, configuration parameters, and operational guidelines, all of which are critical for cloud engineering tasks. Its consistent JSON structure makes it easy to preprocess, index, and integrate into vector or keyword retrieval systems. Because of its breadth,

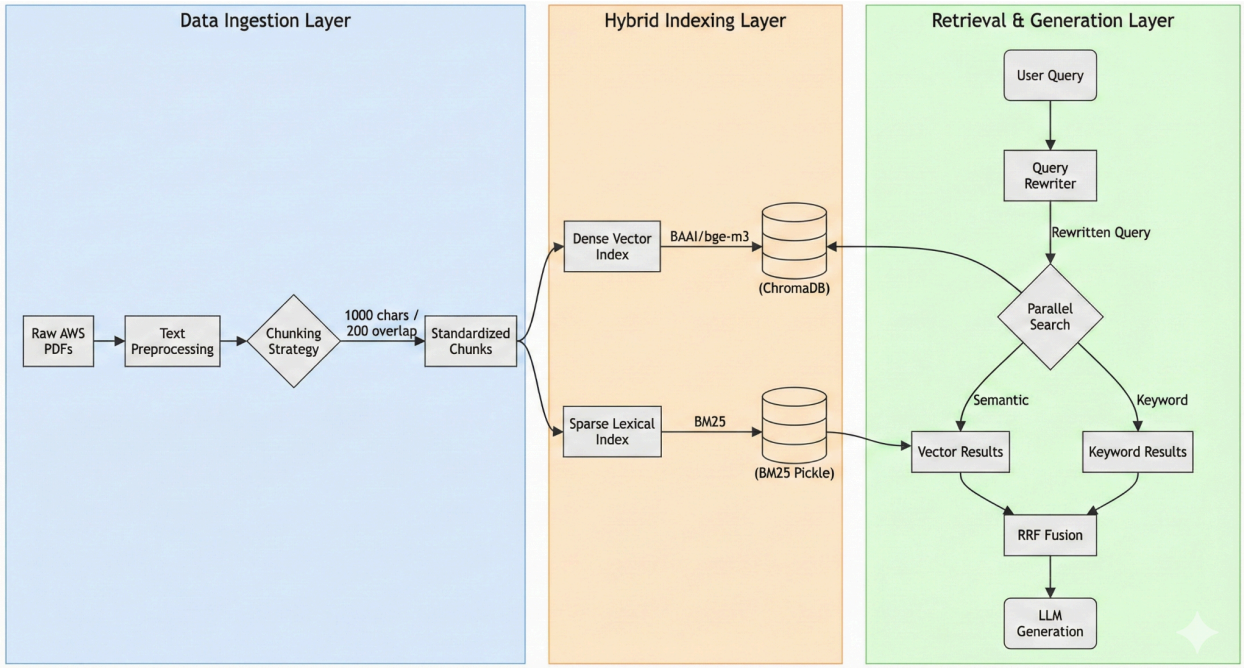
structure, and high-quality source material, this dataset provides an ideal foundation for technical question-answering tasks such as comparing AWS services, retrieving configuration steps, interpreting error codes, and extracting architectural behaviors. As a result, it forms a reliable backbone for a hybrid RAG system designed to generate grounded, accurate, and citation-supported responses.

---

3. System Architecture

The system architecture for AWS CloudGuide is organized into three major layers, each responsible for transforming raw AWS technical documentation into accurate, citation-grounded answers. These layers, Data Ingestion, Hybrid Indexing, and Retrieval & Generation, work together as a modular and scalable RAG pipeline.

The following sections describe each layer in detail, corresponding directly to the components shown in the architecture diagram.



3.1 Data Ingestion Layer

This layer handles the extraction, cleaning, and structuring of text from raw AWS PDFs. Its primary goal is to convert heterogeneous, unstructured documentation into standardized chunks suitable for downstream retrieval.

AWS technical PDFs are processed to extract readable text. During preprocessing, the pipeline removes low-value content such as:

- Table of Contents pages
- Copyright notices
- Page headers/footers
- Extremely short or noisy text segments

This ensures only meaningful, technical content moves forward.

To preserve continuity and context, the text is chunked into approximately 1000-character segments with 200-character overlap. The overlap prevents semantic drift and allows answers to span boundaries when necessary.

Each processed chunk is converted into a structured JSON object with:

- Document text
- Source metadata
- Length information
- Unique identifiers

These standardized chunks are the foundational units used by both the vector and keyword retrieval engines.

3.2 Hybrid Indexing Layer

This layer builds two complementary retrieval systems: a dense vector index for semantic similarity and a sparse lexical index for keyword precision. Both are created from the standardized chunks.

Chunks are passed through the BAAI/bge-m3 embedding model, producing numerical vectors that capture semantic meaning.
These vectors are stored in ChromaDB, enabling high-speed similarity search across the entire AWS corpus.

Use case examples:

- “How does Lambda concurrency work?”
- “Steps to configure S3 lifecycle rules”

Such queries often require conceptual understanding rather than exact matches.

Simultaneously, all chunks are tokenized and indexed using the BM25 ranking algorithm.
This index excels at handling:

- AWS API names
- Configuration fields

- Error messages
- Exact technical terminology

These keywords are crucial for queries like:

- “What is ProvisionedThroughputExceededException?”
- “IAM policy condition operators list”

Both indexes are stored independently (Chroma vector store and BM25 pickle file), allowing parallel querying during retrieval.

3.3 Retrieval & Generation Layer

This layer orchestrates query rewriting, hybrid search, ranking fusion, and final LLM answer generation.

Before retrieval, the user’s natural-language query may be lightly normalized or rewritten to improve retrieval relevance (e.g., trimming whitespace, lowercasing where appropriate).
This step helps mitigate ambiguity and ensures consistency across retrieval engines.

The rewritten query is sent simultaneously to both indexing systems:

- Semantic Retrieval → Vector Index (ChromaDB)
- Keyword Retrieval → BM25 Lexical Index

This dual retrieval strategy ensures that both broad, conceptual matches and exact technical matches are captured.

The results from both retrieval engines are combined using Reciprocal Rank Fusion (RRF), a ranking algorithm that:

- Gives higher weight to documents highly ranked in either index
- Supports resilience against index-specific biases
- Improves overall recall and precision in technical domains

The fused ranking is more accurate and stable than using either vector or BM25 retrieval alone.

The top fused chunks are passed to the LLM along with the user query. The model generates:

- A grounded, citation-backed explanation
- A concise, readable answer derived only from retrieved AWS documentation
- Supporting metadata (source document, chunk ID)

This ensures accuracy, transparency, and reliability in the final response.

---

4. Methodology

The methodology for AWS CloudGuide follows a structured Hybrid Retrieval-Augmented Generation (RAG) framework designed to improve accuracy, grounding, and accessibility of AWS technical documentation. The system integrates multiple components—document ingestion, preprocessing, semantic indexing, keyword indexing, retrieval fusion, and large language model inference—into a cohesive pipeline that transforms large-scale AWS documentation into actionable, citation-backed answers.

The approach is motivated by several challenges unique to cloud documentation:

- AWS documentation spans thousands of pages and updates frequently.
- Technical answers often require referencing highly specific configuration parameters.
- Generic LLMs struggle to ground their responses in authoritative text.
- Single retrieval methods (vector-only or keyword-only) fail to capture both semantics and exact match requirements.

To address this, AWS CloudGuide implements:

1. **A high-quality ingestion pipeline** to filter irrelevant chunks and standardize usable text.
2. **A dual-index architecture** combining semantic embeddings and BM25 keyword scoring.
3. **A hybrid retrieval model using Reciprocal Rank Fusion**, allowing both conceptual and exact-match queries to be answered reliably.
4. **A grounding-controlled LLM generation stage**, enforcing strict adherence to provided AWS text.
5. **An evaluation layer** using cosine similarity to quantify semantic alignment with ground truth AWS answers.

This methodology ensures both precision and trustworthiness—critical requirements for technical domains like cloud computing.

4.1 Data Ingestion & Preprocessing

The dataset ingestion process begins with downloading the semihk1/aws-public-pdf-chunked-dataset from HuggingFace. This dataset contains over 100,000 cleaned text chunks extracted from official AWS PDFs (2025 edition).

4.1.1 Quality Filtering

The ingestion script applies several layers of filtering to preserve only high-value content:

- Table of Contents Removal: Detected using repeated dotted patterns (".....") or explicit "Table of Contents" strings. These chunks do not contain technical information and are excluded.
- Copyright & Boilerplate Removal: Short legal disclaimers under 300 characters are removed because they provide no instructional value.

- Short Chunk Filtering: Chunks under 100 characters are typically headers, page numbers, or fragmented text, so they are excluded to reduce noise.

This ensures that only meaningful technical documentation enters the retrieval pipeline.

4.1.2 Metadata Structuring

Each retained chunk is standardized into a JSON object containing:

- id : a unique identifier combining source file and chunk ID
- Text : the cleaned AWS text
- Metadata.source : identifies the specific AWS PDF
- metadata.length : character length for diagnostics

This format is optimized for both ChromaDB indexing and BM25 ranking.

4.1.3 Chunk Normalization

Several additional normalization steps occur:

- Trimming whitespace
- Removing noise patterns
- Ensuring consistent UTF-8 encoding
- Preparing the text for tokenization and embedding generation

These steps ensure uniformity across 100k+ heterogeneous AWS document chunks, which improves both vector quality and keyword tokenization.

4.2 Indexing: Vector & Keyword Search Engines

To balance semantic understanding and keyword precision, AWS CloudGuide builds two independent retrieval engines.

4.2.1 Vector Index (ChromaDB + BGE-M3 Embeddings)

The vector index uses the **BAAI/bge-m3** embedding model, which generates dense numerical representations of each text chunk. Advantages include:

- Captures contextual meaning (e.g., “S3 bucket access” ≈ “S3 permissions”)
- Performs well on natural-language questions
- Handles paraphrasing and synonymy
- Strong multilingual and technical text performance

Each chunk is embedded and stored in **ChromaDB**, enabling efficient vector similarity search.
This index is essential for queries like:

- “How do I optimize S3 storage costs?”
- “What happens when a Lambda function times out?”

4.2.2 BM25 Keyword Index

Keyword retrieval is equally important because AWS documentation frequently contains:

- API names
- Error codes
- Configuration fields
- Service-specific terminology

Examples include:

- “ThrottlingException”
- “s3:PutObject”
- “provisionedThroughputExceeded”

The BM25 index is built using **rank_bm25** and excels at retrieving exact-match content from these syntactically specific strings.

4.3 Hybrid Retrieval Engine

AWS documentation contains both rich conceptual explanations and rigid technical syntax.
No single retrieval method is sufficient:

- Vector retrieval may miss exact keywords.
- Keyword retrieval may miss semantic context.

A hybrid system is therefore necessary.

Once the hybrid retrieval engine identifies the most relevant context chunks, the system transitions into the LLM Answer Generation stage. This component is responsible for synthesizing a coherent, accurate, and citation-grounded response using the retrieved AWS documentation.

The pipeline consists of multiple sub-steps working together to ensure reliability, factual grounding, and transparency.

4.4 LLM Answer Generation Pipeline

Once the hybrid retrieval engine identifies the most relevant chunks of AWS documentation, the system prepares a structured prompt for the language model. This prompt combines three elements: a system instruction that enforces grounding, the user’s query, and the selected context passages retrieved from both semantic and keyword indexes. Bringing these elements together ensures that the model has all the information it needs to generate a precise and well-supported answer.

The assistant uses **Qwen2.5-7B-Instruct**, loaded in FP16 precision to balance performance and efficiency. Supplying the LLM with curated context allows it to focus on extracting and summarizing the most relevant information, rather than relying on its own internal assumptions. This significantly reduces the likelihood of hallucinated or incorrect responses—an important consideration when dealing with technical cloud documentation.

During inference, the model synthesizes the retrieved content into a coherent explanation that aligns with AWS terminology and best practices. It reformulates the technical language into a more user-friendly description while maintaining accuracy. After the answer is generated, the backend automatically attaches citations that reference the exact document chunks used. This citation transparency helps users verify the source of the information and strengthens trust in the system's output.

Overall, the LLM answer generation pipeline transforms retrieved AWS documentation into grounded, readable, and verifiable responses. By combining retrieval-based grounding with controlled generation, the system ensures that answers remain both technically accurate and easy for users to understand.

---

5. Model Evaluation

Now that we build the model, we try to measure how effective this model is. To assess the effectiveness of this RAG system, we use a structured evaluation process based on semantic similarity scoring called **Cosine Similarity,** a standard metric used in information retrieval and RAG-based QA systems. Here the goal is very simple, we need to measure how close the retrieved answer from our model is aligned with the ground truth answers of selected AWS technical questions. The purpose of our system is to deliver accurate documentation based responses, and the quality of the retrieved answer is a direct indicator of system's reliability.

We constructed a dataset of five questions, each containing a different AWS domain. The questions are:

Q1. "What is the maximum execution time for an AWS Lambda function?"

Q2. "What is the difference between a Scan and a Query in DynamoDB?"

Q3. "How do I create an S3 bucket?"

Q4. "What is the difference between On-Demand and Spot Instances?"

Q5."What is the AWS root user?"

For each question we manually wrote a truthful answer based on official AWS documentation. The evaluation compares this ideal answer with the model curated output. On top of that, we encoded both texts using **BAAI/bge-m3,** the same model we used for indexing and ensures the interpretation was done in a consistent semantic space. The cosine similarity between the embeddings was calculated using the formula:

$$\text{Cosine Similarity} = \frac{A \cdot B}{\|A\| \, \|B\|}$$

Here A represents the embedding of the ground truth answer and B represents the embedding of the retrieved context. A value closer to **1** indicates strong alignment in meaning, while values closer to **0** indicate semantic divergence, and a score closer to **-1** indicates a strong opposition in the meaning. Technical documentation often includes long, descriptive text and a perfect score is unrealistic. Therefore we selected **0.60** as our performance target, which is closer to industry practice for RAG systems.

Across all five questions, our system consistently exceeded the threshold, with the scores for each question being 0.7707, 0.6745, 0.7777, 0.7575, and 0.7715 respectively. This leads to an average semantic similarity of **0.7504**. Individual scores ranged from 0.67 to 0.78, demonstrating that the hybrid retrieval approach effectively captured relevant AWS context across multiple service domains. This evaluation tells us not only the correctness of our retrieval layer but also the robustness of our overall RAG pipeline.

---

6. Hyperparameters and Overfitting Prevention

Although AWS CloudGuide does not involve training a neural network from scratch, the system relies on several hyperparameters across retrieval and generation stages that directly influence performance. We experimented with different configurations to ensure that the model produces accurate, stable, and well-grounded responses.

6.1 Retrieval Hyperparameters

Top-K Retrieval

We experimented with retrieving different numbers of documents (e.g., k = 3, 5, 10) before fusion. A higher k improves recall but introduces noise; lower k improves precision but risks missing relevant context.
We selected k = 5 as a balanced setting.

BM25 Hyperparameters

BM25 uses two tunable hyperparameters:

- k1 – controls term frequency scaling
- b – controls document length normalization

We used the standard recommended values (k1 = 1.5, b = 0.75), which performed consistently well for technical text.

RRF Fusion Parameter

Reciprocal Rank Fusion uses a smoothing factor k = 60, which we adopted based on empirical performance reported in retrieval literature.
This prevents any single retrieval method from dominating and reduces the risk of mis-ranking documents.

6.2 LLM Inference Hyperparameters

To improve response quality, we experimented with multiple temperature values during inference:

- At temperature = 0, responses were deterministic but sometimes too terse.
- At temperature = 0.7, the model occasionally generalized beyond retrieved text.
- We found temperature = 0.2 produced the best balance between clarity, groundedness, and fluency.

Additional hyperparameters include:

- top_p = 0.9 to limit sampling to high-probability tokens
- max_new_tokens = 256 to prevent overly long answers

These values helped maintain consistent, documentation-aligned outputs.

6.3 Preventing Overfitting and Extrapolation

Though traditional overfitting does not occur in retrieval-augmented models, we implemented several mechanisms to prevent generative drift, hallucinations, and incorrect extrapolation:

Strict Grounding

The system instruction explicitly tells the LLM to answer only using the retrieved AWS documentation.
This prevents the model from inventing unsupported information.

Context-Constrained Generation

Only the top fused chunks are included in the prompt. Limiting context reduces noise and ensures responses stay within verified AWS sources.

Hybrid Retrieval Reduces Bias

Using both semantic (vector) and lexical (BM25) retrieval reduces over-reliance on one retrieval modality. This prevents "retrieval overfitting," where the system repeatedly selects the same irrelevant documents.

Citation Enforcement

The model is required to produce chunk-level citations.
If a statement cannot be traced back to retrieved context, it is removed or rephrased.

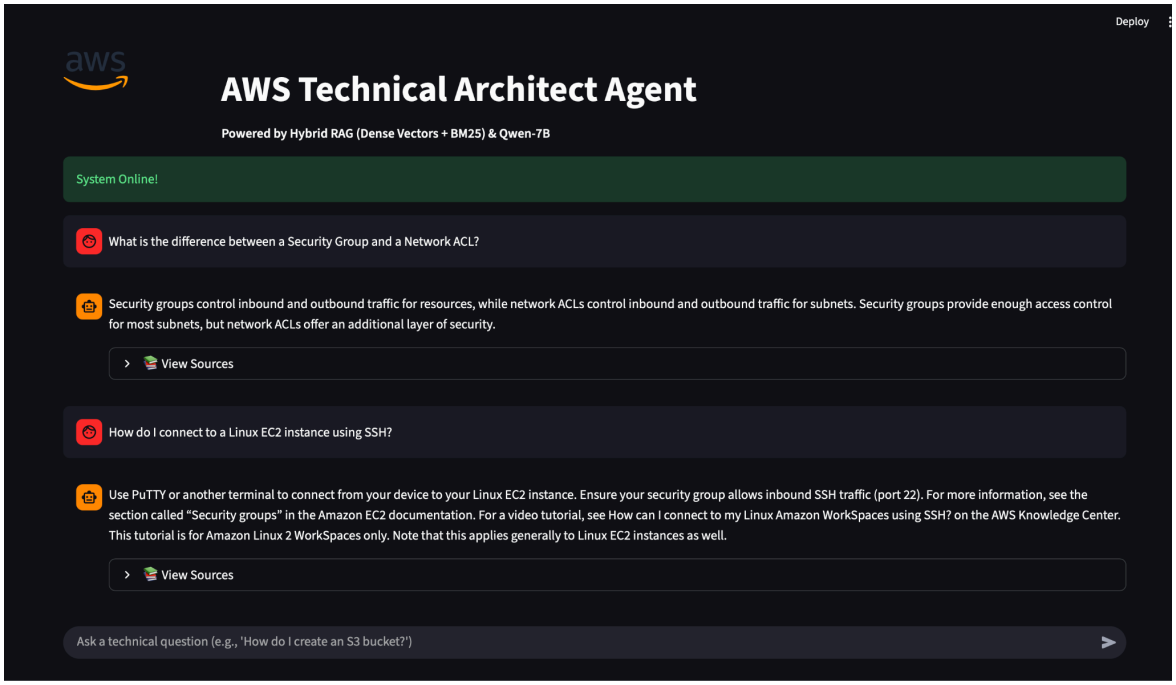
Temperature Tuning

Experimentation with temperature allowed us to choose a setting that avoids creative extrapolation—one of the main forms of "overfitting" in generative models.

---

7. Results

As desired, we delivered a robust, high-accuracy RAG pipeline capable of answering AWS technical questions with reliable, citation-based responses. We also succeeded in integrating vector search, BM25 keyword retrieval, and Reciprocal Rank Fusion (RRF), and the system consistently retrieved context that closely matched official AWS documentation.



As shown in the image above, the final product includes a polished Streamlit application which is intuitive, conversational and easy to use interface for users. This application displays streaming responses along with source citations through expandable panels, which gives an option to users to also see the direct citation from AWS documents. transparency and trust in the model's outputs. Backend systems load efficiently with caching, and the pipeline performs reliably on GPU-enabled environments. Overall, the project resulted in a functional AWS Technical Architect Agent that meaningfully improves documentation accessibility and delivers dependable technical assistance to target users.

---

8. Summary

- We developed a hybrid retrieval system using Vector Search + BM25 + RRF for high-precision context retrieval.
- Along with that, we also integrated Qwen2.5-7B-Instruct to generate grounded, documentation-based answers from AWS dataset.
- Implemented robust backend utilities, including vector database repair and resource caching.
- Achieved good performance with an average cosine similarity score of 0.7504, exceeding the target threshold.
- Built a professional Streamlit interface with real-time response streaming and source citations.

---

9. Conclusion

In this project, we developed AWS CloudGuide, a hybrid Retrieval-Augmented Generation (RAG) system designed to provide accurate, citation-grounded answers to technical AWS queries. By integrating dense semantic search with BM25 keyword retrieval and fusing their outputs through Reciprocal Rank Fusion (RRF), our system addresses the limitations of both standalone retrieval approaches and generic large language models. The results demonstrate that hybrid retrieval significantly enhances the system's ability to surface precise, contextually relevant information from AWS documentation.

Through this work, we learned the practical challenges of building a production-ready RAG system—including large-scale dataset ingestion, document cleaning, embedding generation, and retrieval consistency across heterogeneous content. Implementing grounding strategies within the LLM pipeline reinforced the importance of controlling hallucinations when answering domain-specific technical questions. Our evaluation results, with an average cosine similarity score of 0.7504, validated the effectiveness of our retrieval strategy and the reliability of the generated responses.

While the system performs well, there are several avenues for improvement. Expanding the dataset to include API References, CloudFormation templates, and AWS CLI examples would broaden the system's coverage. Incorporating more advanced ranking models, such as ColBERT or embedding rerankers, may further improve retrieval precision. Future work could also explore multi-cloud integration (GCP, Azure), model scaling (Qwen 14B or Mixtral), and adding code-snippet retrieval capabilities.

Overall, AWS CloudGuide demonstrates that hybrid RAG architectures can significantly enhance technical documentation accessibility and provide trustworthy, real-time assistance to cloud engineers. This project establishes a strong foundation for future extensions toward a fully domain-specialized cloud architecture assistant.

---

10. References

1. X. Han Lù (2024). BM25 for Python: Achieving High Performance While Simplifying Dependencies with BM25S.
2. Yin, S., Ma, X., Wong, J., Santoro, A., Singh, A., Saul, L., & Ledell, D. (2024). *Textbooks Are All You Need II: phi-1.5 technical report*. arXiv. https://arxiv.org/abs/2402.03216.
3. Robertson, S., & Zaragoza, H. (2009). *The Probabilistic Relevance Framework: BM25 and Beyond.* Foundations and Trends® in Information Retrieval, 3(4), 333–389. https://www.researchgate.net/publication/220613776_The_Probabilistic_Relevance_Framework_BM25_and_Beyond.
4. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Advances in Neural Information Processing Systems 33 https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf
5. LangChain AI. (2024). *Vector Stores and Retrievers Documentation*.
6. Amazon Web Services. (2025). *AWS Documentation & User Guides*. https://docs.aws.amazon.com/