NexusRAG

NexusRAG is a production-grade Retrieval-Augmented Generation (RAG) system engineered for scientific literature synthesis. The system addresses critical limitations in conventional RAG implementations, including hallucination, poor source attribution, and context fragmentation. By implementing a multi-stage retrieval pipeline with self-correction mechanisms, NexusRAG achieves accurate, verifiable responses grounded entirely in source documents. The architecture prioritizes local execution, ensuring complete data privacy while maintaining competitive performance on resource-constrained hardware.

1. Introduction

Large Language Models (LLMs) demonstrate remarkable natural language capabilities but suffer from hallucination, generating plausible yet factually incorrect information. Retrieval-Augmented Generation mitigates this by grounding responses in retrieved documents. However, standard RAG implementations face challenges: irrelevant retrieval, context window limitations, and poor citation accuracy.

NexusRAG introduces a self-correcting architecture that evaluates retrieval quality before synthesis, implements hybrid search combining semantic and lexical matching, and enforces strict citation verification. The system operates entirely offline, processing sensitive research documents without external data transmission.

1.1 Design Objectives

1. **Accuracy:** Eliminate hallucination through strict source grounding
2. **Traceability:** Every claim linked to specific source passages
3. **Privacy:** 100% local execution with no external dependencies
4. **Efficiency:** Optimized for 8GB RAM consumer hardware
5. **Extensibility:** Modular architecture supporting component replacement

2. System Architecture

The system implements a six-layer architecture with clear separation of concerns. Each layer operates independently, communicating through well-defined interfaces.

Layer	**Components**	**Responsibility**
Presentation	FastAPI, Web Frontend	HTTP endpoints, user interface
Orchestration	Pipeline, Orchestrator	Workflow coordination
Agents	Synthesizer, Verifier	Response generation, validation
Retrieval	Dense, Sparse, Hybrid	Document search, ranking
Ingestion	Parser, Chunker, Embedder	Document processing
Storage	LanceDB, Document Store	Persistence, indexing

Table 1: System Architecture Layers

3. Core Algorithms

3.1 Hybrid Retrieval

The retrieval subsystem combines dense semantic search with sparse lexical matching. Dense retrieval uses sentence embeddings to capture semantic similarity, while sparse retrieval (BM25) excels at exact keyword matching. Results are fused using Reciprocal Rank Fusion (RRF):

$RRF(d) = \Sigma [w_i / (k + rank_i(d))]$

Where k=60 is a constant preventing division by zero, $rank_i(d)$ is the position of document d in result list i, and w_i is the weight assigned to each retrieval method (default: dense=0.7, sparse=0.3).

3.2 Semantic Chunking

Document segmentation follows a hierarchical strategy that preserves semantic coherence:

6. **Section Detection:** Identify document structure via heading patterns
7. **Paragraph Segmentation:** Split sections at paragraph boundaries
8. **Size Normalization:** Merge small segments, split oversized ones
9. **Context Injection:** Prepend section headers to each chunk

Target chunk size is 1000-1500 characters with 300-character overlap. Sentence boundaries are always respected—chunks never split mid-sentence.

3.3 Self-Correction Mechanism

Inspired by Corrective RAG (CRAG), the system evaluates retrieval quality before synthesis. Each retrieved chunk receives a relevance grade:

Grade	Criteria	Action
CORRECT	Directly answers query	Include in synthesis
AMBIGUOUS	Partially relevant	Include with lower weight
INCORRECT	Not relevant to query	Discard, reformulate query

Table 2: Retrieval Quality Grades

4. Response Synthesis

4.1 Context Assembly

Retrieved chunks are formatted into a structured context window. Each source is clearly demarcated with document name, section title, and numeric identifier for citation reference:

```
[Source 1] Document: paper.pdf, Section: Methods
"Extracted passage text from the document..."
```

4.2 Citation Enforcement

The synthesis prompt enforces strict citation discipline. The LLM is instructed to:

- Use ONLY information explicitly stated in provided sources
- Cite every factual claim with source numbers [1], [2], etc.
- Explicitly state when information is not available in sources
- Never hallucinate or use external knowledge

4.3 Post-Processing Verification

Generated responses undergo verification to validate citation accuracy. The system extracts all citations, confirms each references a valid source, and flags any unsupported claims. A confidence score is computed based on source coverage and citation density.

5. Performance Optimization

5.1 Memory Efficiency

The system is optimized for 8GB RAM environments through several mechanisms:

- **Lazy Model Loading:** Models initialized only when first needed
- **Batch Processing:** Reduced batch sizes (16) prevent memory spikes
- **Garbage Collection:** Explicit cleanup after heavy operations
- **Model Selection:** Default to 3B parameter model (2GB vs 5GB)

5.2 Model Specifications

Model	Size	RAM	Latency	Quality
llama3.2:3b (default)	2.0 GB	8 GB	3-5 sec	Good
llama3.1:8b	4.9 GB	16 GB	8-12 sec	Better
all-MiniLM-L6-v2	90 MB	< 1 GB	< 100 ms	384-dim

Table 3: Model Specifications

6. API Specification

The system exposes a RESTful API through FastAPI with the following endpoints:

Method	Endpoint	Description
POST	/api/upload	Ingest document (PDF, DOCX, TXT)
POST	/api/query	Query documents, return cited response
GET	/api/documents	List indexed documents with metadata
GET	/api/status	System health, model status, statistics
DELETE	/api/documents/{id}	Remove document and associated chunks

Table 4: REST API Endpoints

7. Conclusion

NexusRAG demonstrates that production-quality RAG systems can operate effectively on consumer hardware while maintaining strict accuracy standards. The combination of hybrid retrieval, self-correction mechanisms, and citation enforcement significantly reduces hallucination compared to naive RAG implementations.

The modular architecture enables future enhancements including knowledge graph integration, multi-agent orchestration, and streaming response generation. The system serves as both a practical tool for researchers and a reference implementation for RAG best practices.

References

1. Lewis, P. et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS.
2. Yan, S. et al. (2024). Corrective Retrieval Augmented Generation. arXiv:2401.15884.
3. Robertson, S. & Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval.
4. Reimers, N. & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. EMNLP.
5. Cormack, G. V., Clarke, C. L. A., & Buettcher, S. (2009). Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods. SIGIR.
6. Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2024). RAGAS: Automated Evaluation of Retrieval Augmented Generation. arXiv:2309.15217.
7. Gao, Y. et al. (2024). Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997.