SAMIR GENAI ACADEMY

PROFESSIONAL STUDY GUIDE SERIES

Phase 02 · RAG & Knowledge Systems

08

Advanced RAG Techniques

Self-RAG, Corrective RAG (CRAG), GraphRAG & hybrid search

Module 08 of 25 · **1 hour**

Instructor: **Samir Kumar** · Edition 2026

STUDY GUIDE

01 Overview

> Self-RAG, Corrective RAG (CRAG), GraphRAG & hybrid search. This guide gives you the core theory, the tools that matter, a hands-on build, working code, and the pitfalls to avoid — everything you need to learn the topic to a professional standard and apply it in real projects.

1 hour	4 + 4	4	08/25
LESSON LENGTH	CONCEPTS + BUILD STEPS	FRAMEWORKS & TOOLS	PHASE 02

02 Learning Objectives

- Apply hybrid (keyword + vector) search for robust retrieval.
- Implement self-reflective and corrective retrieval loops.
- Use GraphRAG for multi-hop, relationship-heavy questions.
- Build hierarchical indexes with RAPTOR.

03 Core Concepts

Hybrid search

Vector search captures meaning but misses exact terms (IDs, codes, names). Combine it with keyword/BM25 search and fuse rankings (e.g. reciprocal rank fusion). Hybrid retrieval is the most reliable default for production RAG.

Self-RAG & CRAG

Self-RAG lets the model decide when to retrieve and critique whether retrieved chunks are relevant before answering. Corrective RAG grades retrieval quality and, if poor, triggers a web/secondary search — adding a correction loop that prevents confidently wrong answers.

GraphRAG

For questions spanning many documents and relationships, build a knowledge graph of entities and links. GraphRAG traverses the graph to assemble multi-hop context that flat vector retrieval cannot reach.

Hierarchical indexing (RAPTOR)

RAPTOR recursively clusters and summarises chunks into a tree. Queries can retrieve at the right level of abstraction — fine detail or high-level summary — improving answers on large corpora.

04 Visual Diagram

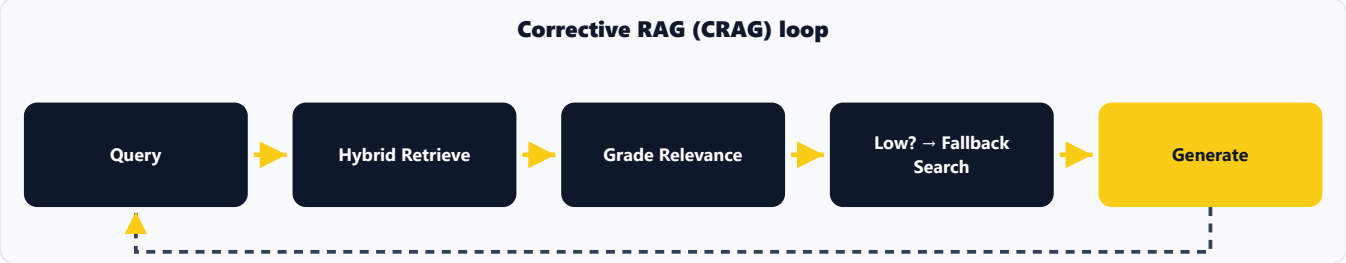

Figure 8 — A grading step decides whether retrieval is good enough; if not, the system corrects itself before answering.

05 Formula Spotlight

Reciprocal Rank Fusion (hybrid search)

$$\mathrm{RRF}(d) = \Sigma_i \; 1 \; / \; (\; k + \mathrm{rank}_i(d) \;)$$

d	a candidate document
$\mathrm{rank}_i(d)$	d's position in result list i (vector, BM25, ...)
k	a smoothing constant, typically 60
Σ_i	sum across every retriever's ranking

RRF fuses multiple ranked lists without needing comparable scores: a document ranked highly by several retrievers accumulates a large combined score. It is the simple, robust way to merge vector and keyword search.

06 Key Frameworks & Tools

LangGraph (control loops) · GraphRAG / Neo4j · Elasticsearch / OpenSearch (BM25) · RAPTOR

07 Hands-On Build

🛠 Hierarchical Indexing with RAPTOR

1. Add BM25 keyword search and fuse with vector results (hybrid).
2. Implement a relevance-grading step that can trigger a fallback search (CRAG).
3. Build a RAPTOR summary tree over a large corpus.
4. Compare flat vs hierarchical retrieval on multi-hop questions.

08 Code Walkthrough

Reciprocal rank fusion for hybrid search — PYTHON

```python
def rrf(rankings, k=60):
    scores = {}
    for ranking in rankings:          # each: list of doc_ids best-first
        for rank, doc_id in enumerate(ranking):
            scores[doc_id] = scores.get(doc_id, 0) + 1 / (k + rank + 1)
    return sorted(scores, key=scores.get, reverse=True)

fused = rrf([vector_ids, bm25_ids])
```

09 🧪 Lab Assistant

Lab 8 · Hybrid search with RRF

Prerequisites: A vector index · A BM25 index (e.g. rank_bm25 or OpenSearch)

1. **Get top-k ids from both vector and BM25 retrievers**
   > Expected: Two ranked id lists
2. **Fuse them with the RRF function**
   > Expected: A single merged ranking
3. **Compare hybrid vs vector-only on ID/code queries**
   > Expected: Hybrid wins on exact-term queries

Troubleshooting

- Exact terms missed → confirm BM25 list is actually contributing.
- No change → check both lists use the same document ids.

10 Pitfalls & Key Takeaways

⚠ COMMON PITFALLS

- Using pure vector search for queries full of exact identifiers.
- Adding correction loops without latency/cost budgets.
- Building a graph when hybrid search would already suffice.

✓ KEY TAKEAWAYS

- Hybrid search is the robust production default.
- Self/Corrective RAG add reflection to prevent confident errors.
- GraphRAG and RAPTOR unlock multi-hop and large-corpus reasoning.

📺 **Companion video:** Watch the module 08 lesson in your Student Dashboard, then use this guide to revise, copy the code, and complete the build. Download both for offline study.

11 Further Resources

Self-RAG & CRAG papers • Microsoft GraphRAG • RAPTOR paper