Retrieval Augmented Generation

Module 1 of LLM Theory to Practice Series

CATAPANG JASPER KYLE

Tokyo University of Foreign Studies

2025/08/20

Agenda

1. Why plain LLMs fall short
2. RAG 101: what it is and when to use it
3. Baseline: Naive RAG
4. Key decisions: *what/when/how* to retrieve
5. Core components (naive RAG)
6. Advanced RAG
7. Additional components (advanced RAG)
8. Evaluation: retrieval & generation metrics; error analysis
9. Stack & practices; trends
10. Hands-on setup & exercise

Why Plain LLMs Fall Short

- **Make things up**: fluent, confident, but not grounded in sources.
- **Go stale**: training cutoffs; cannot reflect new facts or policies.
- **Hard to update**: adding knowledge via retraining is slow and costly.
- **Opaque**: weak provenance—hard to verify or audit answers.

Think

Goal: answers you can *verify*, not just believe.

What a Good RAG System Delivers

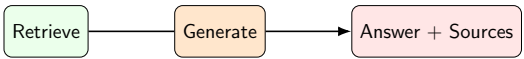
- **Grounded**: every claim links to a source snippet.
- **Fresh**: quick ingest via re-chunk, re-embed, and upserts.
- **Efficient**: predictable latency; fits a fixed context budget.
- **Safe & private**: guardrails plus retrieval-time access control.
- **Observable**: log query $\rightarrow$ retrieved docs $\rightarrow$ prompt $\rightarrow$ answer.

Application Requirements

- Source-grounded answers.
- Frequent updates without retraining.
- Provenance and explainability.
- Cost control and privacy.

Definition

- Retrieve relevant evidence from an external knowledge base, then generate the answer.
- Avoid task-specific retraining; keep a live, swappable corpus.
- Suited to knowledge-intensive tasks: definitional QA, fact checking, grounded summaries.



RAG vs. Fine-Tuning (and Hybrids)

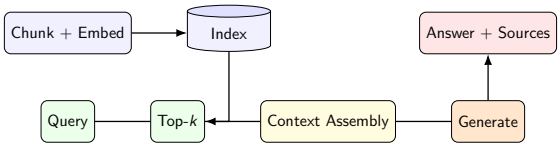
Approach	**Strengths**	**Watch-outs**
RAG	Up-to-date; transparent sources; no retraining	Index freshness; chunk quality; context budget
Fine-tuning	Style/format control; tool-use patterns	Ongoing data curation; drift; higher cost
Hybrid	Structure + freshness; robust routing	Complexity; evaluation overhead

When to Switch Between Fine-Tuning and RAG

- **Switch to RAG** when:
  - Knowledge changes frequently $\rightarrow$ avoids constant retraining.
  - Corpus is large or domain-specific $\rightarrow$ retrieval targets only what's needed.
  - Queries require citations or traceability $\rightarrow$ surface sources.
  - Latency/compute budget can tolerate retrieval step.
- **Switch to Fine-Tuning** when:
  - Domain knowledge is stable $\rightarrow$ update model rarely.
  - Responses require fluency and stylistic consistency $\rightarrow$ embed patterns in weights.
  - Retrieval is impractical (privacy, cost, or infra limits).
  - Inputs are predictable and narrow in scope.
- **Hybrid** when:
  - Core concepts baked in via fine-tuning, volatile facts fetched via RAG.

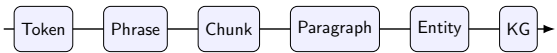
Naive RAG

- Index once, retrieve top-k, stuff into prompt, generate.
- Strength: simplicity and speed.
- Limitation: redundant context; sensitive to chunking.



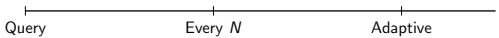
What to retrieve? (granularity)

- Token/phrase/chunk/paragraph; entity-level; knowledge graph.
- Broader recall vs. precision, storage, compute.
- Start with chunks; move finer/coarser as needed.



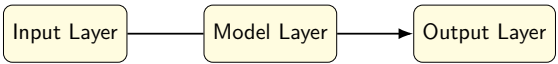
When to retrieve? (frequency)

- Once per query; periodic (every N tokens); adaptive (on uncertainty).
- Efficiency vs. coverage: adaptive can help but adds complexity.



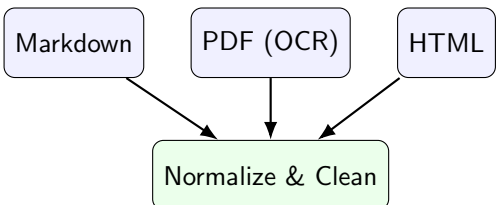
How to use retrieved info? (integration layer)

- Input layer: prompt stuffing; simplest baseline.
- Model layer: fusion modules; strongest but needs training.
- Output layer: citation/verification for faithfulness.



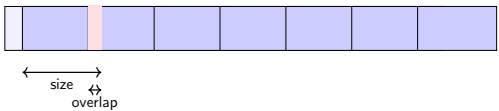
Documents (sources)

- Normalize text; preserve headings; strip boilerplate.
- Attach minimal metadata (title, section, date).
- Keep source of truth close to plain text.



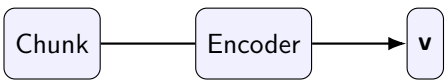
Chunking

- Aim for self-contained segments.
- Section-aware splits; maintain overlap.
- Avoid splitting tables/figures mid-structure.



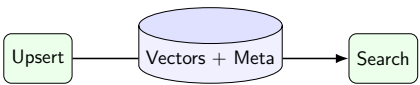
Embeddings

- Text $\rightarrow$ vector; normalize for cosine similarity.
- Store vector + minimal metadata.
- Choose model by domain; validate on retrieval tasks.



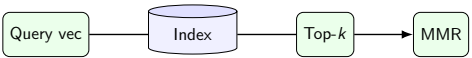
Vector Index

- Store each chunk as an embedding vector *plus* metadata (e.g., source, date, tags).
- Retrieve via ANN search; refine with metadata filters.
- Hybrid mode: combine vector similarity with keyword/BM25 for rare terms.
- Keep schema simple; rebuild or upsert when content changes.



Retriever

- Similarity (top-k) as baseline; MMR for diversity.
- Tune k to balance coverage vs. context budget.
- Optional: metadata filters (topic, date).

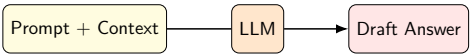


Prompt Template

- Task + constraints: “answer from context only.”
- Serialize chunks with source markers.
- Keep instructions concise; avoid chain-of-thought.

Generator

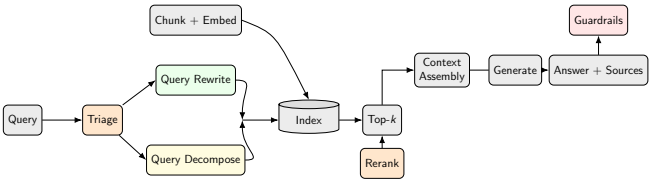
- Use conservative decoding: low temperature; cap output length.
- Encourage refusal when unsupported.
- Emphasize clarity and structure.



Citations & Answer Assembly

- Link claims to chunks; collapse duplicates.
- Show minimal, human-readable source IDs.
- Encourage quick verification.

Advanced RAG



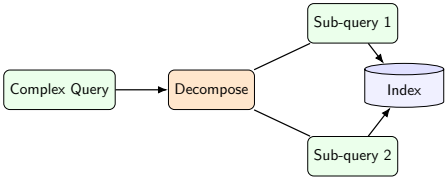
- Builds on basic RAG with pre-, mid-, and post-retrieval steps.
- Adds new modules for query handling, retrieval refinement, and output safety.
- Benefits: higher precision, multi-hop reasoning, safer responses.

Query Rewriting

- Reformulates the query to improve retrieval match.
- Expands abbreviations, fixes grammar, normalizes terms.
- Adapts phrasing to match corpus vocabulary and style.
- Helps with multilingual, noisy, or underspecified queries.

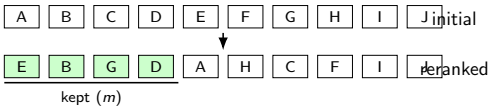
Query Decomposition

- Splits complex or multi-hop questions into smaller, answerable sub-queries.
- Executes each sub-query independently, then aggregates results.
- Supports reasoning over multiple documents or knowledge sources.
- Techniques: rule-based splitting, LLM-driven decomposition.



Reranking

- The retriever gives k possible matches, but the order may not be ideal.
- A stronger model or set of rules re-checks each match for true relevance.
- The most relevant pieces are moved to the front so they fit into the limited context window.
- Benefit: sharper answers with less irrelevant or repeated content.
- Trade-off: adds some extra processing time.



Guardrails

- Policy: refuse if unsupported; avoid speculation.
- Detect prompt injection; sanitize links.
- Keep prompts and system messages private.

Retrieval Metrics

- Recall@k, Precision@k, MRR, NDCG.
- Coverage across sources; diversity vs. redundancy.
- Diagnose failure modes: entity misses, long queries.

Recall@k

- Measures how many relevant items are retrieved in the top-k results.
- Formula: $\text{Recall@}k = \frac{\#\text{ relevant retrieved in top-}k}{\text{total }\#\text{ relevant in corpus}}$
- Higher recall means fewer misses, but may bring more noise.

Precision@k

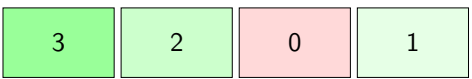
- Fraction of retrieved items in the top-k that are actually relevant.
- Formula:
  Precision@$k = \frac{\#\text{ relevant retrieved in top-}k}{k}$
- High precision means less noise, but may miss relevant items.

Mean Reciprocal Rank (MRR)

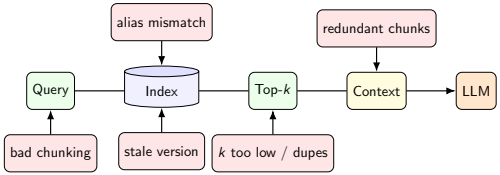
- Evaluates how far down the ranked list the first relevant item appears.
- Formula: $\text{MRR} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank of first relevant}}$
- High MRR means relevant items appear early in the ranking.

Normalized Discounted Cumulative Gain (NDCG)

- Considers graded relevance and penalizes lower-ranked hits.
- DCG: $\text{DCG@}k = \sum_{i=1}^{k} \frac{2^{rel_i}-1}{\log_2(i+1)}$
- NDCG: $\text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}$, where IDCG is the best possible DCG.



Retrieval Failure Modes (with quick fixes)

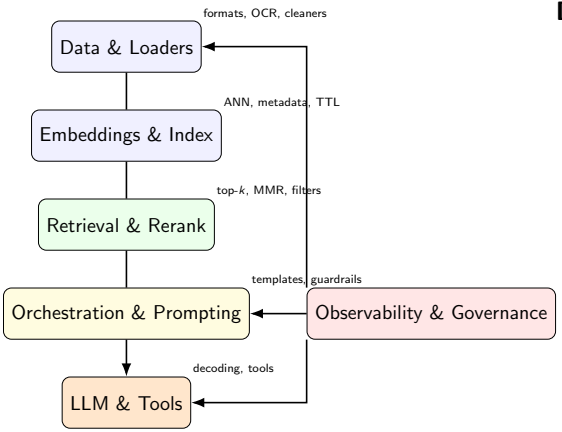


- **Alias mismatch**: "NYC DOE" vs "Dept. of Education" → *rewrite; synonyms; hybrid BM25+vec*.
- **Entity/number miss**: IDs, dates, codes → *keyword fields; exact match; BM25 boost*.
- **Chunking loss**: split evidence → *overlap; section-aware chunks*.
- **Stale hits**: outdated docs → *recency boost; TTL; versioning*.
- k **too low / dupes**: low diversity → *raise k; MMR; per-source caps*.
- **Noisy context**: wasted tokens → *rerank; dedupe; length cap*.
- **ACL misses**: relevant docs filtered → *retrieval-time ACLs*.
- **Coverage gaps**: missing ingestion → *audits; gap alerts*.

Generation Metrics & Methods

- Faithfulness (supported by context), factuality, citation correctness.
- Methods: human rubrics, small gold sets, careful LLM-as-judge.
- Track regression across index updates.

Stack (layered view & practices)



Design principles

- **Swap-ability**: keep retriever/embedder modular.
- **Latency**: aim retrieval 30–50%, generation 50–70%; cache hits.
- **Freshness**: re-chunk/re-embed on schedule; versioned index.
- **Privacy**: filter rows/fields; remove PII pre-index.
- **Observability**: log query $\rightarrow$ docs $\rightarrow$ prompt $\rightarrow$ answer.
- **Evaluation**: run canaries; A/B k, MMR, reranker.
- **Cost**: smaller models, lower k, early refuse low-confidence.

Trends (2024–2025)

- **Agentic / multi-agent rise**: planner→solver→verifier loops; RAG becomes *one tool* in the toolbox.
- **Hybrid search by default**: vector + BM25 + metadata filters; recency boosts.
- **Lightweight rerankers**: small cross-encoders; faster late-interaction scoring.
- **Long-context models**: fewer/smarter chunks; still use retrieval to focus.
- **Multimodal RAG**: tables/figures/diagrams; layout-aware chunking.
- **Structured outputs**: JSON w/ citations; verifiable summaries.
- **Observability**: tracing query→docs→prompt→answer; feedback loops.
- **Cost-savvy orchestration**: caching, dedupe, small local embedders.

Challenges & Open Questions

- **Agent reliability**: plan drift, determinism, guardrails across tools.
- **Retrieval quality vs. speed**: tail queries, synonyms, domain terms.
- **Freshness at scale**: upserts vs. rebuilds; TTL and versioning.
- **Safety**: prompt/indirect injection; link sanitization; model leaks.
- **Access control**: row/field-level ACLs at retrieval time.
- **Evaluation**: faithful *and* helpful; drift & regression tracking.
- **Multilingual & code**: cross-lingual and code-aware embeddings.
- **Answer assembly**: dedupe, cite correctly, avoid over-stuffing.
- **Business alignment**: optimize beyond Recall/NDCG to task KPIs.
- **When to prefer agents**: complex multi-step tasks where tools/flows dominate; keep RAG for grounding.

Key Takeaways

- RAG grounds answers in your corpus to reduce hallucination.
- Performance depends on chunking, retrieval, prompt constraints, and evaluation—not just the model.
- Start simple; iterate with measurement; keep modules swappable.

Hands-On Activity

- **Goal**: Build a small RAG system over a local folder of Markdown notes.
- **Plan**:
  1. Load & normalize documents; chunk with overlap.
  2. Create embeddings; store in a lightweight vector index.
  3. Retrieve top-k chunks; compare similarity vs. MMR.
  4. Prompt an LLM with "answer from context only" + citation format.
  5. Show answer with sources; refuse when unsupported.