Applied Generative AI : LLM Application Development

Dates: 12 & 19 September 2025

Lab 4: Simple RAG (Text-Only)

Upload → Chunk → Embed → Retrieve → Generate

Learning Objectives

- Understand how chunk size & overlap shape retrieval
- Measure RAG quality: retrieval vs. generation metrics
- Recognize failure modes (fragmentation, duplication, dilution)
- Know where to modify code to improve a RAG system
- frameworks

Simple RAG

- Indexing: Text files → chunks → embeddings → FAISS
- Retrieval: query → vector search (top-K)
- Generation: LLM answers using only retrieved context
- Your controls: chunk_size, overlap, k, prompt

Chunking Fits

- Loader → Chunker → Embedder → VectorStore
- Chunker output: id, text, source_id, start, end
- Step size s = chunk_size − overlap
- If L ≤ chunk_size: n = 1; else n = ceil((L − chunk_size)/s) + 1

Chunk Size & Overlap

- Smaller chunks: precise matches, but fragmentation risk
- Larger chunks: better recall, but potential dilution
- More overlap: fewer boundary misses, but more duplicates & index size
- Heuristic start: 500–1200 chars, 10–20% overlap (tune per corpus)

Debugging

- Chunk Preview: counts, boundaries (start, end, length)
- chunk.log: exact splits for discussion
- Retrieved Context: duplicates & boundary misses
- With vs Without RAG: grounded vs hallucinated answers

Measuring RAG Quality — Retrieval

- Hit@K: any gold chunk present in top-K?
- Recall@K: #gold retrieved / #gold total
- MRR/nDCG: rank-sensitive retrieval quality
- Coverage: % questions whose support is retrieved
- Dedup rate: % near-identical items in top-K

Experiments

- Fix a query set; sweep chunk_size: 300, 800, 1200, 1600 (overlap 10–20%)
- Observe Hit@K, Recall@K, duplicates, groundedness
- Sweep overlap: 0 → 50% of chunk size; watch duplication vs recall
- Sweep k: 3, 5, 8, 10; look for noise beyond a threshold

Common Failure Modes

- Boundary miss: facts split across chunks
- Dup-bias: high overlap crowds top-K
- Dilution: huge chunks add noise
- Vocabulary drift: normalize casing/Unicode in loader

To Improve in the Code

- Chunking (lab4/rag/pipeline_text.py): token-based length; semantic split; adaptive overlap
- Embedder (embedder.py): try stronger models; ensure L2-normalize
- Retrieval (vectorstore_faiss.py): MMR; hybrid BM25+dense; reranker
- Prompting (streamlit_app_rag_text.py): concise rules; cite chunk IDs

Deeper Research

- Adaptive chunking by content density
- Graph-augmented retrieval (entities/relations)
- Query planning/decomposition; fusion (RRF)
- Context compression (selective quoting / summarization)
- Learning-to-retrieve; Hallucination guards

Lab 4: Simple RAG

Step 1: Activate Virtual Environment

Windows (PowerShell):

```
.\llm\Scripts\activate
```

macOS/Linux:

```
source llm/bin/activate
```

Step 2: Install Required Packages

- Download lab4_requirements.txt from bhattaraprot.com/lab4_requirements.txt
- Make sure the file is saved in your working directory.
- Install the required packages by running:

```
pip install -r lab4_requirements.txt
```

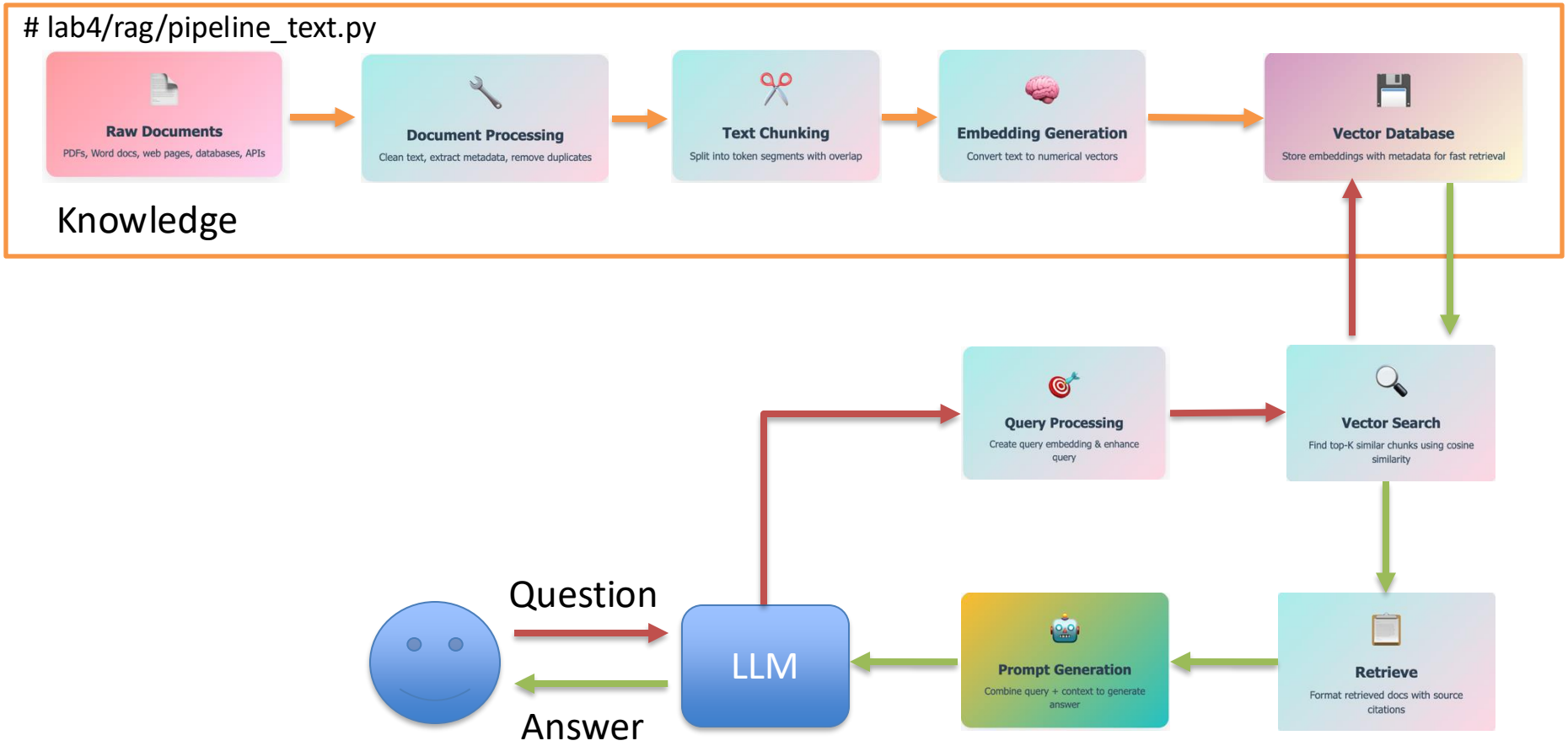
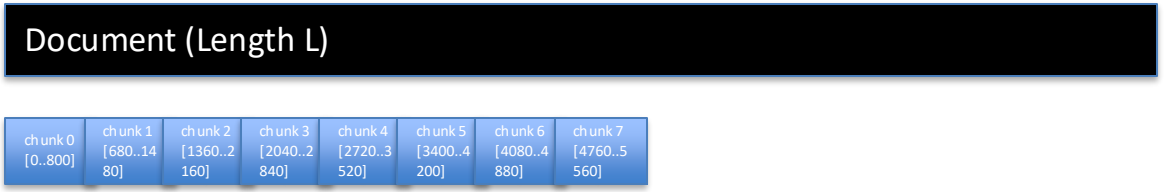

lab4/rag/pipeline_text.py
Raw Documents
PDFs, Word docs, web pages, databases, APIs
Document Processing
Clean text, extract metadata, remove duplicates
Text Chunking
Split into token segments with overlap
Embedding Generation
Convert text to numerical vectors
Vector Database
Store embeddings with metadata for fast retrieval
Knowledge
Query Processing
Create query embedding & enhance query
Vector Search
Find top-K similar chunks using cosine similarity
Question
LLM
Prompt Generation
Combine query + context to generate answer
Retrieve
Format retrieved docs with source citations
Answer

Diagram — Chunking Timeline (chunk_size & overlap)



Step size s = chunk_size − overlap. Larger overlap = more chunks & duplication; smaller overlap = fewer chunks but risk of boundary misses.

Lab 4:

Step 4: Source Installation

- Download lab4.zip from bhattaraprot.com/lab4.zip
- Extract the contents and place them into

  `llm/src/lab4`

- Navigate into the lab4 folder
- Verify the structure by listing files

  `ls –R or dir`

Lab 4:

Step 4: Run Streamlit

```
python -m streamlit run lab4/ui/streamlit_app_rag_text.py
```

Step 5: Upload text file

Upload logexam.txt

Lab 4:

Step 6:

System prompt:

"You are a security analyst. Use ONLY the provided CONTEXT chunks.
If the answer is not fully supported, say "I don't know from the context."
Always list the chunk IDs (or source_id + chunk_index) you used as citations.
Return concise bullet points."

Exact-match / ID prompts

- “List every event mentioning CVE-2025-0175. Include date/time, server, product/component, and action taken.”
- “Show all lines with /orders/view_order.php and list the CVE numbers involved.”
- “Find ‘Path traversal in file upload’ incidents. For each, give date/time, server, and product (e.g., GitLab).”
- “Which entries mention '' OR 1=1 --'? List date/time, server, and targeted component.”
- “List all ‘Patch applied’ events. Include CVE, product, and which service was restarted.”

Semantic / paraphrase prompts

- “Summarize attempts to break out of containers. Which CVEs and servers are involved?”
- “Where do we see privilege-escalation attempts in the kernel? Summarize by CVE and server.”
- “Group all OGNL/Struts remote-code-execution attempts: dates, servers, and whether mitigated.”
- “Which logs indicate XSS payloads in ‘link’ fields? List products and dates.”
- “Where does the log talk about SSRF via data source configuration? Summarize by product and CVE.”

Aggregation / timeline prompts

- “For CVE-2025-0172, report earliest and latest timestamps, affected servers, and all described vectors.”
- “Count events per CVE and show the top 5 by frequency; include any Severity labels you find.”
- “Make a timeline for CVE-2025-0201: scans found, exploits attempted, and patches applied.”
- “Which products or systems are tied to CVE-2025-21610? List each with the evidence lines.”
- “Extract all endpoints/paths mentioned (e.g., /admin/deleteuser.php) and map them to CVEs.

Hybrid & MMR “challenge” prompts

- Hybrid test (ID + concept): “For CVE-2025-0176, combine exact ID matches with any paraphrased mentions (e.g., ‘kernel escalation’ or ‘container runtime’) and summarize.”

- MMR effect: “Retrieve top evidence about SQL-injection attempts across products; avoid near-duplicate chunks and keep diverse examples.”

Tuning tasks (run each prompt with different settings)

- Top-K: run at K=3 → 5 → 8.
  - Watch: Hit@K, duplicate chunks, answer confidence.

- Chunk size / overlap: try 300/20, 800/120, 1200/200.
  - Watch: boundary misses (small chunks), dilution (large chunks), near-dup crowding (high overlap).

RAG Limits & Your Solutions

- From your Lab 4
  - Identify a **limitation** you observe (e.g., boundary miss, duplicate top-K, dilution, missed IDs, stale evidence).

RAG: Limitations

Limitations:

- Retrieval recall gaps (missed evidence, bad chunking/boundaries)
- Evidence quality/staleness & conflicts across sources
- Context window pressure → dilution or truncation
- Redundancy/ranking errors; wrong query intent
- Domain shift & multilingual edge cases
- Hallucinations even with context; weak attribution
- Privacy/compliance (PII, access control, auditability)
- Latency, cost, and update cadence of the index
- Evaluation difficulty: labels, faithfulness, human review

RAG Research ideas

RAG Variant	Example Apps	Domains
Adaptive, structure-aware chunking	Policy Q&A copilot; Contract clause finder; SOP navigator for hospitals; Compliance checklist explainer	Legal/compliance, university regulations, healthcare guidelines, procurement/TORs, technical standards (ISO/NIST)
Hybrid retrieval (BM25 + dense) with RRF + MMR	CVE/incident triage bot; API/code search assistant; Product/SKU catalog search; Financial filing finder	Cybersecurity, developer docs, e-commerce catalogs, finance/regulatory filings
Freshness-aware RAG (time/recency signals)	Regulatory update advisor; Vulnerability/patch dashboard; Price/stock tracker Q&A	News/policy updates, security advisories, gov circulars, market/pricing data, release notes/Changelogs
Self-RAG (retrieve → generate → critique)	Cautious clinical guideline explainer; Legal research assistant with self-check; Student support FAQ with verification	Healthcare (guidelines/SOPs), legal/policy research, education knowledge bases, internal IT KBs
GraphRAG for narrative private data	Multi-hop policy reasoning (e.g. Who must approve); Root-cause analysis assistant; Impact analysis across systems	Hospital SOPs, enterprise architecture docs, pharma/biomed literature, complex program governance
Cross-lingual RAG (Thai–English)	Bilingual PDPA policy Q&A; TH-EN helpdesk search; Cross-lingual procurement assistant; Mixed-language course catalog Q&A	Public sector policies, university governance, procurement, legal/compliance, education admin
RAG evaluation harness for regulated domains	Faithfulness/groundedness scoring dashboard; Red-team hallucination tester; Retrieval quality monitor (Hit@K/nDCG)	Healthcare, finance, government, university admin—any place needing auditability and evidence trails
Continual-learning RAG (when to re-embed vs re-rank)	Change-aware support copilot; Docs drift detector; Auto-refresh indexing policy engine	SaaS product docs, internal IT/DevOps runbooks, security KBs, university websites/announcements