COMS 4995: Applied Machine Learning Final Project

Multi-Document Retrieval-Augmented Generation (RAG) Assistant

Cheng Wu (cw3729) Pranav Jain (pj2459) Jaewon Cho (jc6618) Winston Li (wl3062)

Abstract

Retrieval-Augmented Generation (RAG) improves factual grounding by conditioning LLM outputs on retrieved evidence, but multi-document questions remain challenging due to distributed evidence, inconsistent document structure, and long-context degradation (e.g., "lost-in-the-middle"). We build an end-to-end multi-document RAG assistant over PDF collections, including offline ingestion (PDF loading, chunking, SBERT embeddings, FAISS indexing) and an online reasoning layer that routes queries by intent (synthesis/comparison/extraction), groups retrieved evidence by document, and enforces citation-aware structured prompts. We evaluate the system through four complementary analyses (E1–E4). Results show that structured prompts yield more complete cross-document answers and better document coverage than flat baselines under the same retrieval setting, and that our prompt-building pipeline provides a clean interface for mitigation strategies such as relevance-based reordering.

1 Introduction and Motivation

LLMs can answer many questions fluently, but they frequently hallucinate when asked to reference specific documents. RAG mitigates this by retrieving relevant text chunks and conditioning generation on those chunks. However, **multi-document** settings introduce additional failure modes: (i) relevant evidence is distributed across sources, (ii) retrieved chunks can be noisy or unbalanced across documents, and (iii) long concatenated contexts degrade generation quality and coverage.

Our project targets three common multi-document query intents: **synthesis** (aggregate themes across sources), **comparison** (contrast methods or conclusions), and **extraction** (identify assumptions, definitions,constraints). Instead of scaling models or retrieval, we focus on a **reasoning-aware prompt construction layer** that makes document structure explicit and encourages evidence-grounded integration.

Contributions.

- An end-to-end multi-PDF RAG system with reproducible ingestion, retrieval, and evaluation scripts.
- A lightweight `MultiDocReasoner` that performs query-type routing and document-grouped structured prompting.
- A targeted evaluation suite (E1–E4) designed to surface multi-document and long-context failure modes.
- A Gradio demo UI that supports PDF upload, indexing, and citation-aware question answering.

2 Related Work

RAG systems combine dense retrieval with conditional generation to improve factual grounding. Common building blocks include embedding-based retrieval (e.g., SBERT-style encoders) and vector indexes such as FAISS. Beyond retrieval quality, prior work highlights that **prompt structure** and **context**

organization strongly affect model behavior, especially under long-context degradation where models over-attend to early/late evidence. Our approach emphasizes reasoning scaffolds (routing + grouping + citation constraints) to improve multi-document synthesis without changing the retrieval backbone.

3 System Design and Methodology

Our system consists of an **offline ingestion/indexing stage** and an **online retrieval + reasoning stage**. Figure 1 illustrates the architecture.

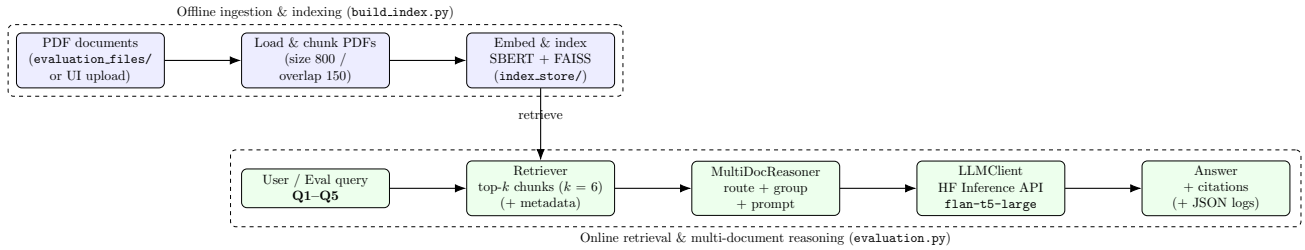


Figure 1: End-to-end pipeline: offline ingestion/indexing (top) and online retrieval + reasoning (bottom).

3.1 Ingestion, Metadata, and Index Persistence

We load PDFs, extract page-level text, and chunk documents with **chunk size 800 characters** and **overlap 150**. A key implementation choice is preserving **traceable metadata** per chunk: `(source_pdf, page_range, chunk_id)`. This metadata is carried through retrieval and later used by the reasoning layer to (i) group evidence by document, and (ii) format citations as `[DocumentName.pdf]`.

To support reproducibility and efficient iteration, the FAISS index is persisted to disk (`index_store/`). This avoids re-embedding all chunks after each run, enabling both UI usage and scripted evaluation to share the same index artifact.

Why these hyperparameters? Chunk size and overlap represent a trade-off between semantic coherence and index granularity. A moderate chunk size (800) captures multi-sentence claims while still allowing retrieval to localize evidence, and overlap (150) reduces boundary loss for definitions or assumptions that span chunk edges.

3.2 Retrieval Interface and Evidence Export

At query time, we retrieve the top-k chunks with $k = 6$ and retain similarity scores. We additionally export retrieved chunks to `evaluation_outputs/evaluation_chunks.json` so that evaluation can inspect the *same evidence* used by the model. This is important: E3/E4 isolate prompt structure effects under identical retrieval, rather than conflating improvements with changing retrieved context.

Component	Setting
Chunking	size 800 chars, overlap 150
Embeddings	SBERT `all-MiniLM-L6-v2`
Vector store	FAISS (persisted as `index_store/`)
Retriever	cosine similarity, top-$k = 6$
LLM backend	HF API, `google/flan-t5-large`
Citations	enforced format `[DocumentName.pdf]`

Table 1: Core system configuration used by both demo and evaluation.

3.3 Multi-Document Reasoning Layer

The central contribution is a lightweight reasoning module placed between retrieval and generation. It enforces:

- **Query-type routing**: classify queries as synthesis, comparison, or extraction.
- **Document-aware context**: group retrieved chunks by source document.
- **Structured prompting**: task-specific instructions plus explicit citation rules.

Query-type routing

We use an interpretable rule-based classifier (keywords for comparison vs. assumptions/limitations vs. default synthesis). This choice is deliberate: routing is a *control-plane* decision that should be transparent and debuggable, especially in a systems project.

Prompt contracts and "citation discipline"

Rather than treating retrieved chunks as raw context, prompts enforce a contract:

- **Document blocks**: retrieved evidence is displayed under headings like `--- doc1.pdf ---`.
- **Answer schema**: synthesis prompts request themes; comparison prompts request common ground + differences; extraction prompts request lists.
- **Citation discipline**: every non-trivial claim should include at least one bracket citation `[docX.pdf]`.

Prompt skeletons (high-level, no code). To make the design concrete, Table 2 summarizes the core structure of each prompt type.

Type	Instruction + structure
Synthesis	**Goal:** integrate themes across sources. **Steps:** (1) identify shared themes, (2) list document-specific additions, (3) write a combined summary. **Rule:** cite each theme with `[Document.pdf]`.
Comparison	**Goal:** contrast methods/conclusions. **Steps:** (1) common ground, (2) key differences, (3) contradictions/edge cases. **Rule:** differences should cite both sides (at least two docs).
Extraction	**Goal:** extract structured items. **Steps:** produce bullet list of assumptions/definitions/limitations grouped by document. **Rule:** each bullet includes a bracket citation.

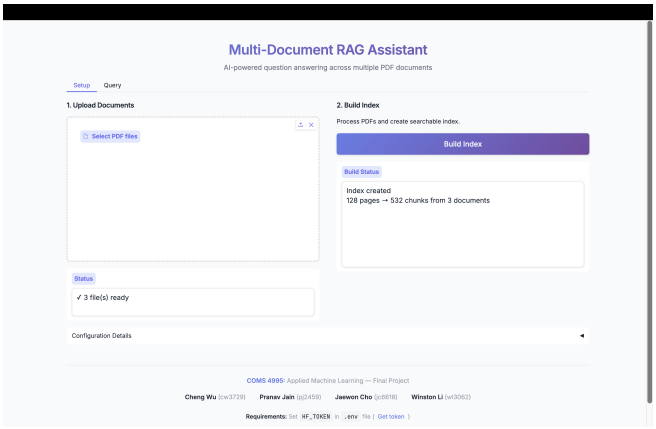
Table 2: High-level prompt skeletons used by the reasoning layer.

3.4 Lost-in-the-Middle Mitigation Hook

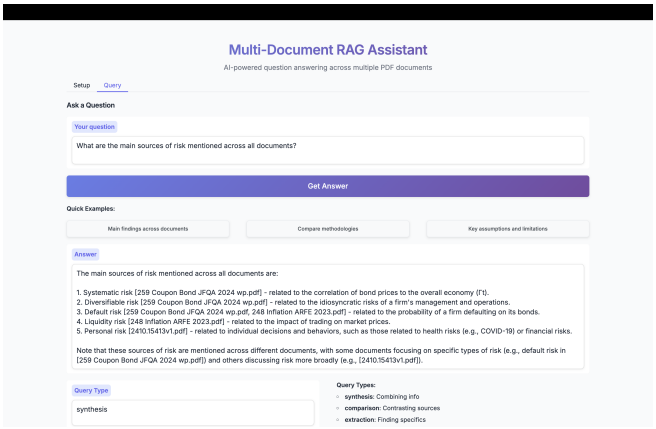
To expose long-context failure modes, we optionally reorder retrieved chunks by relevance score: move the highest-relevance chunks to the beginning and end of the prompt, and interleave across documents for balanced exposure. This is evaluated in E4 as a lightweight ablation. The goal is not to "solve" long-context issues fully, but to demonstrate the system provides a clean interface for reasoning-level interventions.

3.5 User Interface

We provide a Gradio interface to demonstrate end-to-end functionality: upload PDFs, build the index, and ask multi-document questions with citations. Figure 2 shows the setup and query screens.



(a) Setup: upload PDFs and build index



(b) Query: ask questions and view cited answers

Figure 2: Gradio demo UI. In one example run, an index was created from 3 documents (128 pages) producing 532 chunks.

Demo workflow. The UI enforces a two-step workflow aligned with our pipeline: (1) build the index once per document set, then (2) run many queries. This mirrors real-world usage patterns where indexing is the expensive offline stage and query-time latency is dominated by retrieval + generation.

4 Evaluation and Results

We evaluate using four complementary analyses:

E1 query-type classification accuracy of the reasoning module;
E2 prompt-structure sanity checks for synthesis/comparison/extraction prompts;
E3 baseline flat prompts vs. structured reasoning prompts;
E4 an ablation on "lost-in-the-middle" mitigation via relevance-based reordering.

All experiments operate on the same code paths as the full system (`MultiDocReasoner`, retriever, prompt builders, LLM client). Results are generated via `examples/evaluation.py` and stored in `evaluation_outputs/` as JSON for inspection.

4.1 Experimental Setup

Documents and indexing. For evaluation, we use a controlled PDF set located in `evaluation_files/` (e.g., `doc1.pdf`, `doc2.pdf`) to ensure reproducibility. Documents are chunked with size 800 and overlap 150; FAISS is built with SBERT embeddings.

Evaluation questions. We adopt five canonical multi-document questions:

Q1 Summarize the main ideas across the documents. (synthesis)
Q2 What are the main sources of risk discussed? (synthesis)
Q3 Compare how different documents describe the same concept or methodology. (comparison)
Q4 What are the key assumptions and limitations? (extraction)
Q5 How do the documents differ in conclusions or policy implications? (comparison)

LLM and practical constraints. We use `google/flan-t5-large` through the HuggingFace Inference API. During development, HF endpoint changes and rate limits reduced the number of long, repeated calls; therefore E3–E4 emphasize qualitative prompt and output comparisons (with saved JSON artifacts) rather than large-scale automatic scoring.

4.2 E1: Query-Type Classification Accuracy

Table 3 reports routing accuracy on Q1–Q5. The classifier achieves 100% accuracy, confirming prompt routing is reliable for our targeted intents.

Question	Gold Type	Predicted	Correct?
Q1	synthesis	synthesis	True
Q2	synthesis	synthesis	True
Q3	comparison	comparison	True
Q4	extraction	extraction	True
Q5	comparison	comparison	True
Accuracy		**1.00 (5/5)**	

Table 3: E1: Query-type routing accuracy (sanity check).

4.3 E2: Prompt-Structure Sanity Checks

We verify that prompt templates expose document structure and required instructions (grouping and citation constraints). All checks pass (Table 4), confirming templates provide correct structural signals to the LLM.

Check	Pass?
Synthesis prompt includes `doc1.pdf`	True
Synthesis prompt includes `doc2.pdf`	True
Comparison prompt groups chunks by document	True
Extraction prompt mentions "assumptions"	True

Table 4: E2: Prompt-template sanity checks.

4.4 E3: Baseline vs. Structured Reasoning Prompts

We compare:

- **Baseline prompt**: flat concatenation of retrieved chunks (no document grouping or explicit reasoning rules).
- **Structured prompt**: produced by `MultiDocReasoner` with document grouping, answer schema, and citation constraints.

Evaluation dimensions. Because large-scale automatic scoring was limited by API stability, we evaluate qualitatively along three system-level dimensions: **(i) document coverage**, **(ii) answer organization**, and **(iii) citation behavior**. These dimensions directly reflect the motivation of the reasoning layer.

Concrete qualitative evidence (examples). Representative observations from saved outputs (Q2/Q3/Q4) illustrate what changed when only prompt structure changed:

	Baseline (flat)	Structured (reasoner)
Document coverage	Often dominated by one source; weaker cross-doc integration	Better balance across docs due to grouping
Citations	Inconsistent, sometimes missing	Enforced `[Document.pdf]` usage
Answer structure	Less organized; mixing claims across docs	Theme/contrast/list schema aligned with intent
Failure visibility	Hard to diagnose (unordered context)	Easier to inspect (per-doc blocks + rules)

Table 5: E3: Observed differences under identical retrieval ($k = 6$).

- **Q2 (risk synthesis):** baseline outputs often list risks from a single document; structured prompts more frequently surface risks across documents and tag them with bracket citations.
- **Q3 (method comparison):** baseline responses tend to summarize one method; structured prompts more consistently separate "common ground" vs. "differences" and cite both documents when contrasting.
- **Q4 (assumptions/limitations):** structured extraction prompts produce more list-like outputs (assumption → citation), reducing narrative drift.

4.5 E4: Lost-in-the-Middle Ablation (Relevance Reordering)

Long contexts can cause models to overweight early/late evidence. We test a simple mitigation strategy: reorder chunks so that highest-relevance evidence appears near the beginning and end of the prompt.

Why this matters in multi-doc QA. In multi-document settings, losing one mid-context chunk can effectively mean losing an entire document's unique contribution. Reordering provides a lightweight way to (i) emphasize the most relevant evidence, and (ii) reduce sensitivity to arbitrary chunk ordering.

Observed effect. While not a full mitigation study, we observe that reordering makes outputs less sensitive to where high-salience evidence appears in the prompt, and it integrates naturally into our prompt-building pipeline without modifying retrieval or embeddings.

5 Comparison to Baselines and Alternative Designs

Our system modifies the *reasoning interface* between retrieval and generation. This design is orthogonal to scaling: under the same retriever and LLM, we aim to improve multi-document behavior by making document structure explicit and enforcing task-specific reasoning constraints.

5.1 System-level comparison

We compare three designs: (C1) naive multi-document RAG (top-k retrieval + flat concatenation), (C2) our structured reasoning RAG, and (C3) a scaling-only alternative that relies primarily on larger models or longer contexts without a reasoning layer.

5.2 Why prompt structure matters beyond retrieval

Even with correct retrieval, flat prompts can collapse document boundaries, making it easy for the model to (1) over-weight one source, (2) mix evidence without attribution, and (3) omit minority documents. By enforcing document blocks and answer schemas aligned with query intent, our system improves coverage and interpretability without changing embeddings, vector store, or k.

Dimension	C1: Naive RAG	C2: Ours	C3: Scale-only
Multi-doc awareness	Implicit	Explicit (grouped)	Mostly implicit
Query intent handling	None	Routed (3 types)	None
Prompt structure	Flat	Structured schemas	Flat
Citation discipline	Inconsistent	Enforced	Inconsistent
Failure diagnosis	Difficult	Interpretable	Difficult
Compute dependence	Low	Low	High
Extensibility hooks	Limited	Modular	Limited

Table 6: System-level comparison with common multi-document RAG alternatives.

6 Error Analysis and Failure Modes

A systems report should be explicit about where and why failures occur. We observed the following failure modes in multi-document QA:

6.1 Failure modes: baseline vs. structured reasoning

(F1) Single-document dominance. Under flat prompting, answers frequently anchor on the first coherent narrative, even when retrieval returns evidence from multiple documents. In contrast, grouped prompts reduce this effect by forcing the model to process evidence under per-document headings, making omissions more visible.

(F2) Context ordering sensitivity. Baseline prompts exhibit strong sensitivity to chunk order, especially when unique evidence appears mid-context. Our relevance-reordering ablation (E4) demonstrates that the structured pipeline naturally supports ordering interventions without changing retrieval.

(F3) Citation without grounding. Requiring citations improves traceability but does not guarantee true grounding. The model can still attach citations post hoc to vague claims. This motivates future claim–evidence alignment evaluation beyond bracket formatting.

(F4) Practical constraints from public endpoints. Public inference endpoints introduce variability (rate limits, timeouts) that affect evaluation throughput. Persisting indexes and exporting retrieved evidence to JSON partially mitigates this by decoupling retrieval artifacts from generation.

7 Scalability and System-Level Considerations

Our architecture separates offline indexing from online querying, which matches real usage patterns: build once, query many times.

Index-time cost. Indexing scales linearly with the number of chunks: PDF loading + chunking + embedding each chunk, followed by FAISS index construction. Because the index is persisted to `index_store/`, this cost is amortized across future queries and shared by both the UI demo and evaluation.

Query-time cost. Query latency is dominated by (i) FAISS similarity search and (ii) a single LLM generation call. The reasoning layer adds negligible overhead relative to generation, because it performs lightweight routing, grouping, and formatting. This design makes it easy to experiment with prompt strategies without recomputing embeddings or rebuilding indexes.

Scaling to more documents. As the number of documents grows, flat prompting becomes brittle due to context length limits and ordering sensitivity. Our design degrades more gracefully because document grouping preserves source boundaries even when not all evidence can fit into the prompt. In practice, the same reasoning interface can be paired with future retrieval upgrades (reranking, diversified top-k selection) to maintain multi-document balance.

8 Discussion and Limitations

What worked. Across E1–E4, the reasoning layer improves interpretability and multi-document coverage under identical retrieval. Grouping makes evidence boundaries explicit, and citation rules encourage traceability. The modular design enables extensions (reranking, routing upgrades, memory) while maintaining a stable ingestion backbone.

Limitations. Our evaluation is small and emphasizes qualitative comparisons rather than large-scale automatic metrics. We also rely on the public HuggingFace API, and rate limits/endpoint changes reduced the amount of repeated LLM testing. Finally, citation formatting does not guarantee true grounding; stronger evaluation (claim–evidence alignment, human judgment) would strengthen conclusions.

9 Conclusion and Future Work

We built a reproducible multi-document RAG assistant for PDF collections, combining FAISS-based retrieval with a lightweight reasoning module that routes queries, structures evidence by document, and enforces citation-aware prompts. Our evaluation shows that structured prompting improves cross-document synthesis and provides a clear interface for mitigation strategies targeting long-context degradation.

Future work.

- **Retrieval improvements:** reranking (cross-encoder), hybrid lexical+dense retrieval, and diversity-aware selection for multi-doc balance.
- **Routing improvements:** learnable query-type classification beyond keyword rules, or multi-label intents.
- **Evaluation:** claim-evidence alignment metrics and human evaluation for faithfulness/coverage.
- **System robustness:** caching, batching, and fallback models to reduce dependence on unstable inference endpoints.

Team Contributions

- **Cheng Wu**: Led and implemented the overall system, including document ingestion and retrieval, end-to-end system integration and debugging, as well as the complete evaluation design and report writing.
- **Pranav Jain**: Contributed to LLM logic exploration, including the design and testing of prompting strategies and multi-document reasoning experiments.
- **Jaewon Cho**: Led the development of the end-to-end demo interface, integrating the backend RAG pipeline into an interactive web-based UI for document upload and querying.
- **Winston Li**: Worked on document ingestion and retrieval components (PDF loading, chunking, embeddings, vector store).