Advanced Offline Retrieval-Augmented Generation (RAG)

A Privacy-First, Deterministic System for Document Summarization and Question Answering

Abstract

This report presents an Offline Retrieval-Augmented Generation (RAG) system designed for privacy-preserving document understanding, extractive summarization, and grounded question answering. The system operates fully offline, combining dense neural embeddings, FAISS-based similarity search, sentence-level relevance scoring, and deterministic answer composition.

Unlike generative-only systems, the architecture prioritizes interpretability, reproducibility, and correctness by separating document summarization from question answering and avoiding hallucination-prone generation in the core pipeline. An optional local LLM is provided for rewriting, without affecting retrieval or ranking. The system is implemented in Python with Streamlit for interaction and supports PDFs/TXT ingestion, index persistence, export/import, and exploratory analysis.

System Overview

The Offline RAG system is designed around a clear separation of concerns:

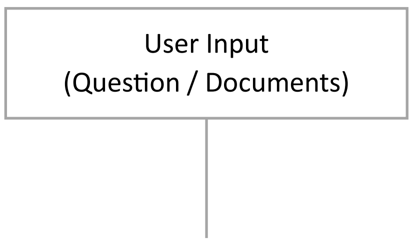
1. **Indexing (offline, batch)**: Documents are chunked, embedded, and indexed using FAISS.
2. **Retrieval (neural)**: Query embeddings retrieve top-k relevant chunks via cosine similarity.
3. **Answer Composition (extractive, deterministic)**: Relevant sentences are selected, filtered by question intent, deduplicated, and composed directly without generative summarization.
4. **Summarization (document-level)**: A graph-based extractive summarizer provides document overviews, separate from QA.
5. **Optional LLM Augmentation**: A local abstractive model may rewrite final outputs for readability, without influencing retrieval or ranking.

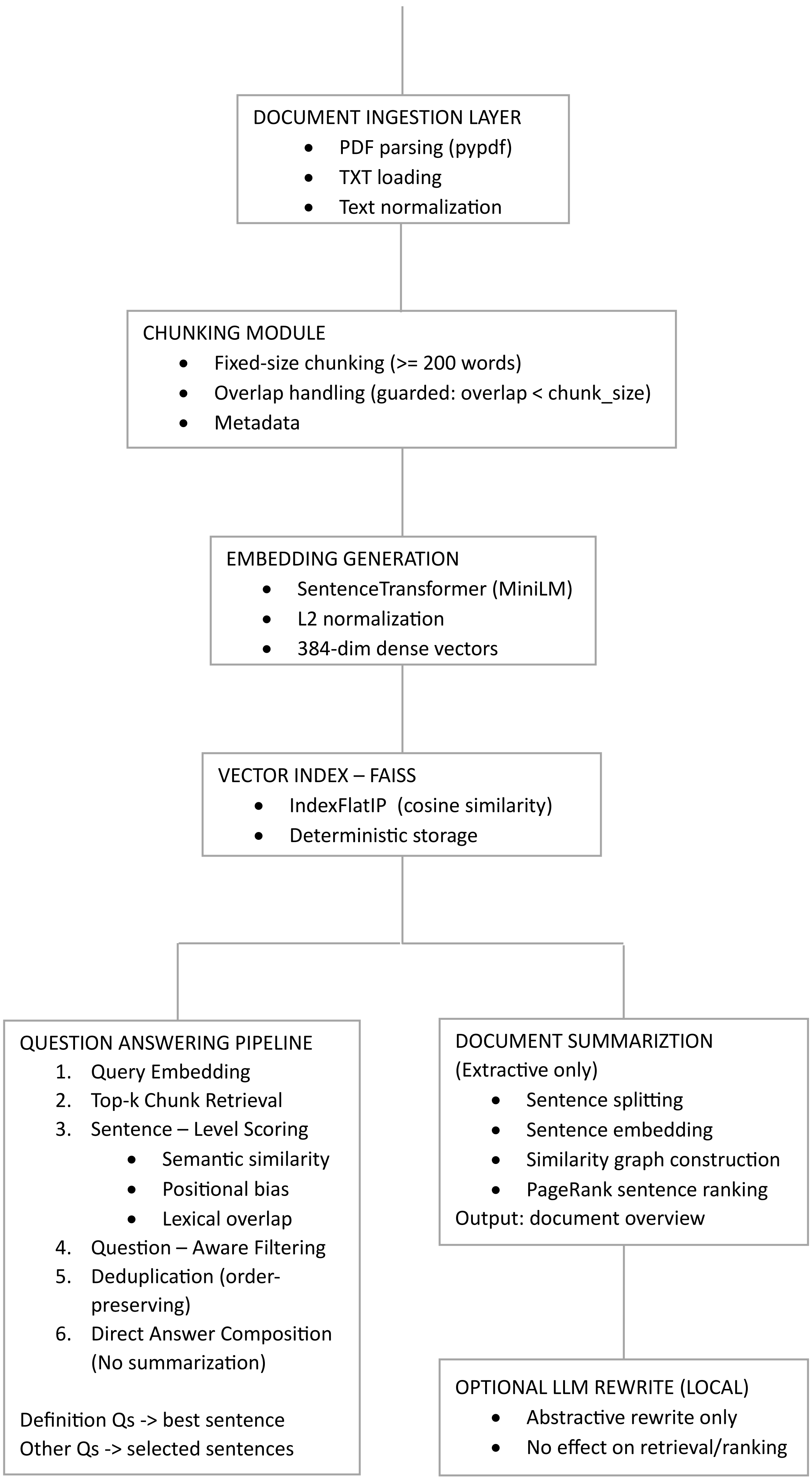
Core components:

- *core.py*: Indexing, retrieval, summarization, and QA logic
- *streamlit_app.py*: User interface and orchestration
- *requirements.txt*: Dependency specification

---

1. High-Level Architecture




DOCUMENT INGESTION LAYER
• PDF parsing (pypdf)
• TXT loading
• Text normalization
CHUNKING MODULE
• Fixed-size chunking (>= 200 words)
• Overlap handling (guarded: overlap < chunk_size)
• Metadata
EMBEDDING GENERATION
• SentenceTransformer (MiniLM)
• L2 normalization
• 384-dim dense vectors
VECTOR INDEX – FAISS
• IndexFlatIP (cosine similarity)
• Deterministic storage
QUESTION ANSWERING PIPELINE
1. Query Embedding
2. Top-k Chunk Retrieval
3. Sentence – Level Scoring
• Semantic similarity
• Positional bias
• Lexical overlap
4. Question – Aware Filtering
5. Deduplication (order-preserving)
6. Direct Answer Composition (No summarization)
Definition Qs -> best sentence
Other Qs -> selected sentences
DOCUMENT SUMMARIZTION (Extractive only)
• Sentence splitting
• Sentence embedding
• Similarity graph construction
• PageRank sentence ranking
Output: document overview
OPTIONAL LLM REWRITE (LOCAL)
• Abstractive rewrite only
• No effect on retrieval/ranking

High-level architecture of the Advanced Offline RAG system. The pipeline separates document summarization from question answering to avoid double compression and preserve answer coherence. Neural embeddings are used exclusively for retrieval, while answer composition remains extractive and deterministic. Optional LLM components operate strictly as rewriters and do not affect retrieval or ranking.

---

2. Indexing and Embedding Pipeline

2.1 Document Ingestion and Chunking

Documents (PDF or TXT) are converted to raw text and segmented into overlapping chunks. Chunking parameters are expressed in words, with a minimum size of 200 words and configurable overlap. A guard enforces overlap < chunk_size to prevent infinite loops and ensure deterministic behavior.
Design rationale:

- **200-word minimum** preserves paragraph-level semantic coherence.
- Overlap enables context continuity across boundaries.
- Batch rebuilding avoids duplication.

2.2 Embedding Generation

Each chunk is embedded using a SentenceTransformer bi-encoder (default: all-MiniLM-L6-v2). Embeddings are L2-normalized and stored as 32-bit floating-point vectors.
This model was selected because it offers:

- Strong sentence-level semantic similarity
- Low computational cost
- Stable behavior in offline settings

2.3 Vector Index Construction

Embeddings are indexed using FAISS IndexFlatIP, enabling cosine similarity via inner product on normalized vectors. The index is rebuilt deterministically during ingestion or when the embedding model changes.
Integrity checks enforce:

- Alignment between chunks and metadata
- Consistent dimensionality across embeddings

This design intentionally favors correctness and simplicity over incremental updates.

---

3. Retrieval Mechanism

Given a user query:

1. The query is embedded using the same SentenceTransformer.
2. FAISS performs a brute-force similarity search.
3. The top-k chunks are returned with similarity scores.

Retrieval is purely neural. No symbolic rules or heuristics are applied at this stage, ensuring recall-oriented behavior.

---

4. Sentence-Level Answer Composition

4.1 Motivation

Initial experiments showed that applying extractive summarization to QA outputs caused coherence loss due to double compression. To address this, the system adopts **sentence selection without summarization** for QA.

4.2 Sentence Scoring

For each retrieved chunk:

- Sentences are split using regex-based segmentation.
- Sentence embeddings are compared with the query embedding.
- Scores incorporate:
    - Cosine similarity
    - Positional bias (earlier sentences favored)
    - Lexical overlap with query tokens

4.3 Question-Aware Filtering

A critical improvement ensures that only sentences with lexical overlap with the query are retained. This removes:

- Application examples when asking for definitions

4.4 Definition Handling

For definition-style questions (e.g., "What is X?"), the system returns the single highest-scoring sentence globally. This avoids topic drift and produces crisp, textbook-style answers.

4.5 Deduplication and Composition

Selected sentences are deduplicated while preserving order, then concatenated directly to form the final answer. No further summarization is applied.
This pipeline ensures:

- Grounded answers
- Coherent flow
- Zero hallucination risk

---

5. Document Summarization

Summarization is treated as a **separate task** from QA.

5.1 Method

A graph-based extractive summarizer (TextRank-style) is used:

- Sentences are embedded
- A similarity graph is constructed
- PageRank identifies central sentences

Selected sentences are reordered by original position to produce an overview.

5.2 Intended Use

This summarizer is designed for:

- Document overviews
- High-level understanding
- Pre-reading and navigation

It is explicitly **not used** in QA due to known coherence limitations.

---

6. Optional LLM Augmentation

The system optionally supports a local abstractive summarizer (e.g., DistilBART) for:

- Rewriting extractive summaries
- Improving readability of QA outputs

Key constraints:

- LLM outputs never affect retrieval or ranking
- All core logic remains deterministic
- LLM use is explicitly user-controlled

This design prevents hallucination from influencing factual correctness.

---

7. Streamlit Application Layer

The Streamlit UI provides three modes:

1. **Question Answering**
    - Query input
    - Top-k retrieval control
    - Evidence visualization
    - Highlighted sentences
2. **Summarize All**
    - File-wise extractive summaries
    - Optional LLM rewrite
    - Keyword extraction
3. **Explore Index**
    - Chunk inspection
    - File-level statistics
    - Knowledge graph visualization

Index persistence and export/import are supported for reproducibility.

---

8. System Strengths

- **Privacy-first**: Fully offline by default
- **Deterministic**: No stochastic generation in core logic
- **Interpretable**: Evidence and sentence highlights exposed
- **Modular**: Embeddings, summarizer, and UI can be swapped
- **Research-aligned**: Clear separation of summarization and QA

---

9. Limitations

- **Extractive summarization coherence**
  The document summarization module is extractive and graph-based, prioritizing factual coverage over narrative flow. As a result, summaries may be semantically correct but lack coherence.
- **Limited intent discrimination in retrieval**
  The system relies on bi-encoder embeddings for retrieval, which favor recall over fine-grained intent understanding. This necessitates heuristic filtering to prevent topic drift in question answering.
- **Absence of cross-encoder reranking**
  The current pipeline does not employ cross-encoder models for reranking, limiting its ability to model deep query–sentence interactions at the final selection stage.
- **Scalability issue**
  The use of FAISS IndexFlatIP and batch index rebuilding is optimized for moderate-sized corpora and is not efficient for very large document collections without additional indexing strategies.
- **No explicit contradiction or uncertainty modelling**
  The system does not detect conflicting evidence across documents or provide confidence estimates, which limits its ability to signal ambiguity in complex or disputed topics.

---

10. Future Work

- **Cross-encoder reranking for QA**
  Integrating a lightweight cross-encoder could improve sentence-level relevance estimation by jointly modelling queries and candidate sentences, reducing reliance on heuristic filters.
- **Scalable and incremental indexing**
  Future versions could adopt approximate nearest neighbour indices (e.g., IVF or HNSW) and support incremental updates while maintaining metadata consistency.
- **Discourse-aware summarization**
  Incorporating discourse modelling or hybrid extractive–abstractive summarization techniques could improve narrative coherence while preserving interpretability.
- **Adaptive chunking strategies**
  Chunking could be made structure-aware by leveraging document layout, section boundaries, or semantic segmentation instead of fixed word counts.
- **Contradiction detection and uncertainty estimation**
  Adding mechanisms to identify conflicting evidence and quantify uncertainty would improve transparency and trustworthiness in ambiguous scenarios.

11. Conclusion

This project presents an offline Retrieval-Augmented Generation system designed with a focus on correctness, clarity, and reliability rather than unrestricted text generation. By clearly separating document summarization from question answering, the system avoids common problems such as over-compressing information, mixing unrelated topics, and generating unsupported answers. All responses are grounded directly in the source documents, which helps maintain accuracy and trustworthiness.
The system is built with a clear design approach where each component serves a specific purpose. Batch indexing is used to keep the system stable and consistent, bi-encoder retrieval helps capture relevant content efficiently, and simple filtering rules improve answer precision. Optional local language models are used only to rewrite final outputs for readability and do not influence retrieval or ranking. The system's limitations are openly recognized, and its modular structure makes it easy to extend with more advanced techniques such as better reranking, scalable indexing, or improved summarization in the future.