Text Splitters and Chunking:

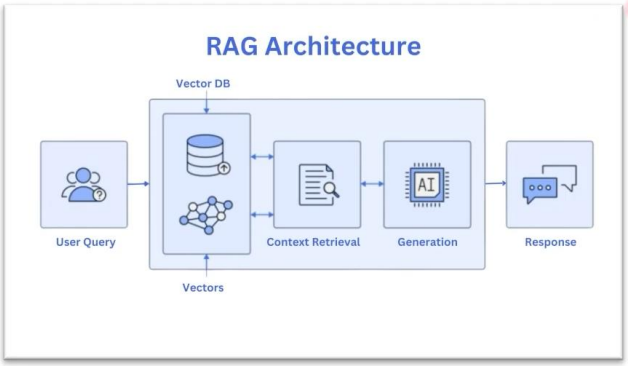
Text Splitters & Chunking

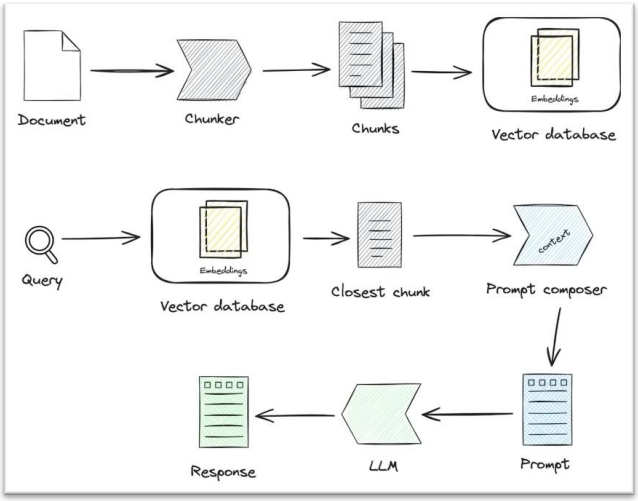
Text splitters divide large documents into smaller, meaningful chunks suitable for retrieval and generation. Chunking ensures that the language model receives focused, token-efficient context, improving accuracy in RAG workflows.

01

Text Splitters & Chunking

Text Splitters are techniques used to divide large text documents into smaller, manageable pieces called **chunks**. These chunks are then embedded into vector databases and used for semantic search or retrieval-augmented generation (RAG).

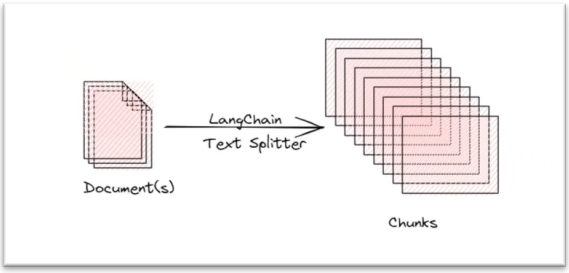




02

Why Splitting and Chunking Matter

- **Context Window Limit**: LLMs (like GPT-4) have a token limit (~8k to 128k). You can't embed or send entire books at once.
- **Search Relevance**: Smaller chunks help find the most relevant piece of information.
- **Chunk Overlap** ensures the context doesn't break awkwardly between ideas, paragraphs, or sentences.
- **Improved Retrieval Quality** in vector stores like FAISS, ChromaDB, or Pinecone.



Types of Text Splitters

Type	Description
Character Splitter	Splits text by a character count (e.g., every 500 characters)
Recursive Splitter	Smart splitting: tries sentences → paragraphs → words → characters
Token Splitter	Splits text by number of LLM tokens (e.g., 512-token chunks)
Sentence Splitter	Uses sentence boundaries to chunk logically
Markdown/HTML Splitter	Specialized for structured formats (e.g., titles, bullet points)

Chunking Strategy Parameters

Parameter	Purpose
chunk_size	Size (in tokens or characters) of each chunk
chunk_overlap	Number of overlapping tokens between chunks
separator	Optional character/symbol to split by (e.g., \n, .)

Example: With chunk_size=200, chunk_overlap=50 → Each chunk overlaps the previous by 50 tokens. This helps preserve context between chunks.

Best Practices for Chunking

- ❑ Use chunk_overlap to avoid losing semantic meaning between chunks.
- ❑ Align chunk size with your embedding model (e.g., 384 tokens for MiniLM, 1024 for e5-large).
- ❑ Use domain-specific logic: titles, bullet points, and headers if chunking structured documents.
- ❑ Don't make chunks too small — context gets lost (ideal: 200–800 tokens).
- ❑ Evaluate chunks using vector search recall or retriever metrics.

Use in RAG Pipelines

- Chunked documents → embedded into vector DBs (Pinecone, Chroma, FAISS).
- On query, the top relevant chunks are retrieved based on vector similarity.
- LLM uses these chunks as context to generate accurate, grounded answers.