Retrieval-Augmented Generation (RAG) Architecture

This diagram outlines the flow of a RAG system that integrates HTML scraping, chunking, vector database storage, and response generation using a language model.

1. HTML Scraping

Use libraries like BeautifulSoup or Scrapy to extract raw text from web pages. This is the data ingestion step where structured or unstructured content is collected.

2. Text Chunking

Split the scraped text into smaller overlapping chunks (e.g., 500 tokens with 50-token overlap). This helps maintain context and improves retrieval accuracy.

3. Embedding & Vector Database

Convert each chunk into vector embeddings using models like SentenceTransformer. Store these embeddings in a vector database such as Chroma, Pinecone, or Weaviate for efficient similarity search.

4. Query Embedding

User inputs a query which is also converted into an embedding using the same model. This embedding is used to search the vector database.

5. Retrieval

The system retrieves the most relevant chunks from the vector database based on similarity to the query embedding.

6. Response Generation

Combine the retrieved chunks with the user query and pass them to a language model (e.g., GPT) to generate a context-aware response.