Blog

Core Techniques Powering Enterprise Knowledge Assistants

2025-01-15 by Wierd van der Haar, 5 minute read

CHATBOT

To harness the potential of RAG, organizations need to master a few crucial building blocks.

***** This article is part of blog series. If you haven't read the previous article yet, be sure to check it out:** Building AI Knowledge Assistants for Enterprise PDFs: A Strategic Approach

1. Extracting from PDFs

Before you can feed your data into an RAG pipeline, you need to extract it f PDFs. This step sets the foundation for the entire workflow. The goal of y to present actual images, provide text-only responses, or gen directly impacts how you extract and process each PDF. For ir display or summarize images, you'll need dedicated mechanis retrieve them; if you're only interested in text, you can focus o

Text Extraction

body of text and store them separately as part of the document's metadata.

Image Detection

Some PDFs include images or diagrams that may hold critical information. Identifying these images is essential if you need a fully comprehensive pipeline that can reference not just text but also visual elements.

OCR (Optical Character Recognition)

OCR transforms the scanned images of text into machine-readable text, thereby creating a "digital text layer" where none existed before. This ensures you can index, extract, and analyze the content just like any other text-based PDF. The process can be resource-intensive (often requiring GPUs), but it's indispensable for processing large volumes of scanned documents.

Table Extraction

When PDFs contain data in tabular format, consider using specialized tools or libraries (e.g., Tabula, Camelot) to extract tables accurately. Tables often include important figures or text organized in rows and columns, which may otherwise be lost if parsed as standard text. Decide whether to keep the table structure (e.g., converting to CSV or HTML) or to summarize the data for downstream tasks like embedding or semantic search.

Metadata Collection

Don't forget about titles, authors, creation dates, and other metadata. These details help with advanced filtering and can also influence the retrieval steps later. In some cases, you might add header and footer data here if it provides contextual clues or helps distinguish versions of a document.

2. Chunking Extracted Data

Structure and Content-Aware Chunking: Considers sentences, paragraphs, sections, or chapters when chunking. Preserve logical boundaries, which can significantly improve retrieval quality.

Document-Based Chunking: With this chunking method, you split a document based on its inherent structure. This approach respects the natural flow of the content but may not be as effective for documents that lack a clear structure.

Hierarchical Chunking: Combines fixed-size and structure-awareness by chunking at multiple levels (document → chapter → paragraph) and linking them in a parent-child relationship.

Semantic Chunking: The main idea is to group text segments with similar meaning. You create embeddings for each segment, then compare those embeddings to see which ones are most closely related. This approach keeps similar ideas together, preventing arbitrary splits that could harm retrieval quality.

Agentic Chunking: This method empowers an LLM to dynamically decide how to split the text into chunks. We begin by extracting short, independent statements from the text and let an LLM agent determine if each statement should join an existing chunk or start a new one. Because the model understands context, it can produce more coherent chunks than fixed or structural methods.

3. Generating Embeddings

Once you have your chunks, each chunk needs a vector representation (an embedding) that captures its semantic meaning.

Choosing Embedding Models

Security Considerations: The classification of your documents might prohibit the use of online LLMs, pushing you toward an on-premise or self-hosted model. If confidentiality is a priority, local deployment can ensure that no data leaves your environment.

summaries.

- **Images:**
  - If your chatbot must search for images via text or by providing another image ("show me images similar to this"), you'll need to generate embeddings for the images
  - If you only need to display the original images without advanced search features, you may opt to store them directly in your database.
  - Multimodal models (e.g., CLIP or GPT-4 Vision) can handle both text and images, enabling semantic search across different data types.
- **Task Orientation:** Think about the end goal—text-to-text, image-to-text, image-to-image, or table-based queries. Different scenarios benefit from specialized embedding models.

Performance Considerations

Hardware Requirements: Embedding models often require GPUs or other accelerators for efficient batch processing.

Local vs. Cloud Deployment: Weigh the cost and convenience of a cloud solution against the benefits of total control and data sovereignty offered by an on-premises model.

Image & Table Processing: Generating embeddings for images or large tables typically requires more compute resources and sometimes specialized libraries or frameworks.

4. Storing the Data

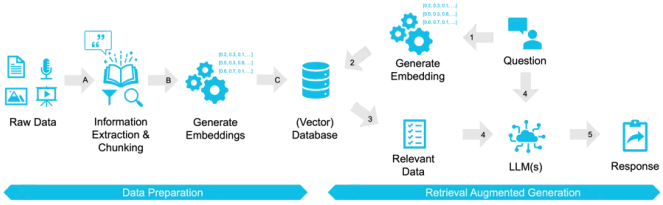
The diversity of data and the sophistication of AI models demand a flexible, powerful, and nuanced approach to data management. As AI continues to penetrate various sectors, the need for databases that can adapt to complex data landscapes becomes paramount.
The future of multi-model databases in AI shines—as an enabler of complex, context-rich, and real-time intelligent applications.

In a RAG workflow, you need to store:

offer **flexibility** (for structured, unstructured, or semi-structured data), **speed** (sub-second queries on large datasets), and **advanced search** functionalities.

*** **Continue reading:** Designing the Consumption Layer for Enterprise Knowledge Assistants

Related Posts

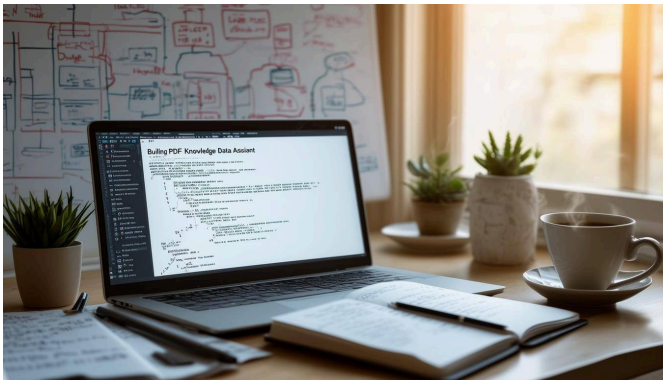


Building AI Knowledge Assistants for Enterprise PDFs: A Strategic Approach

2025-01-15

In today's increasingly data-driven world, many organizations are sitting on mountains of information locked away in PDFs. Whether it's business reports, regulatory documents, user manuals, or researc...

READ MORE



Step by Step Guide to Building a PDF Knowledge Assistant

2025-01-15

This guide outlines how to build a PDF Knowledge Assistant, covering: Setting up a project folder. Installing dependencies. Using two Python scripts (one for extracting data from PDFs, and one for cr...

Designing the Consumption Layer for Enterprise Knowledge Assistants

2025-01-15

Once your documents are processed (text is chunked, embedded, and stored) — read "Core techniques in an Enterprise Knowledge Assistant" — , you’re ready to answer user queries in real time. This stage...

READ MORE