Suggested Tools

Data Storage Options for RAG Challenge

When tackling the Retrieval-Augmented Generation (RAG) challenge, your choice of data storage architecture can significantly impact your workflow and performance. Depending on your approach—vector-based, relational (SQL), graph-based, or a combination—different technologies are recommended.

---

1. Vector Databases

Purpose: Semantic search on unstructured text

Recommended Tools: FAISS, ChromaDB

Features:

• Lightweight, open-source, and locally runnable

• Optimized for fast similarity search on embeddings

• Ideal for storing chunked HTML embeddings

Use When:

• You need semantic search across HTML or text

• Your pipeline includes OpenAI or other embedding models

---

2. Graph Databases

Purpose: Modeling relationships, entities, and linked data

Recommended Tool: Neo4j (Community Edition)

Features:

• Excellent for representing and querying complex relationships

• Uses Cypher, an intuitive and powerful query language

• Suitable for linking pages, concepts, and hierarchical structures

Use When:

• You extract or infer relationships from HTML content

• You want to analyze connections, links, or data hierarchies

---

3. Relational Databases

Purpose: Managing structured/tabular data

Recommended Tool: SQLite

Features:

• Lightweight and file-based; no setup required

• Great for local or embedded environments

• Works well with structured HTML elements (e.g., tables, attributes)

Use When:

• Your parsed HTML yields structured/tabular data

• You want a simple, reliable storage format for structured info

---

Additional Tools to Consider

• **Hugging Face:** Pre-trained models and tokenizers (can be used for embeddings)

• **LangChain:** Pipeline orchestration for RAG and agentic workflows. (Recommended to build the RAG pipeline)

• **Google Colab:** Recommended - Cloud-based compute resources (if not running locally)

Note:

👉 These are guidelines to help you get started—feel free to use any other tools or frameworks that best suit your project needs.

Product Component Breakdown

1. **Raw Data**

Description: Unstructured HTML content in json collected from company websites.

Role: Acts as the foundational dataset that contains all the necessary company information.

2. **Data Parsing and Storage**

Description: This stage involves cleaning and extracting relevant content from the raw HTML data.

Role: The extracted information is segmented into manageable chunks and then stored in a suitable database (vector-based, relational, or graph-based) designed to support efficient retrieval.

3. **Agent (RAG System)**

Description: The Retrieval-Augmented Generation (RAG) agent combines retrieval mechanisms (vector search, SQL queries, etc) with a generative language model.

Role: It leverages the stored data to identify the most relevant pieces of information in response to a user query, forming the backbone of the intelligent system.

4. **Answer Generation**

Description: Using a generative language model (such as GPT-4o), this component processes the retrieved information to generate coherent and contextually accurate answers.

Role: It converts technical data into insights and actionable recommendations tailored to the user's query and intent.

5. **User Interface (Basic)**

Description: A simple, user-friendly interface that allows users to input queries and view the generated responses.

Role: Ensures accessibility and ease of use, enabling the end user to interact with the system via a web portal or CLI, or inside a notebook, facilitating quick decision-making.