**Data**

**📁 Structured Data**

* **What**: Organized spreadsheet with fixed columns
* **Purpose**: Quick lookup of breed-specific health info
* **Key Columns**:
  + Breed Name: Specific breed (e.g., "Abyssinian")
  + Primary Health Issue: Most common problem (e.g., "Periodontal disease")
  + Recommendation: Veterinary advice (e.g., "Omega-3 fatty acids...")
  + Keywords: Tags for search (e.g., ["teeth", "gums"])

**📂 Unstructured Data**

**A. Research Papers (PDFs)**

* **Purpose**: Scientific studies with latest findings
* **Location**: data/unstructured/pdfs/research\_papers/
* **Files**:
  + canine\_nutrition.pdf - Dog diet research
  + feline\_health.pdf - Cat disease studies

**B. Clinical Studies (PDFs)**

* **Purpose**: Veterinary treatment research
* **Location**: data/unstructured/pdfs/clinical\_studies/
* **Files**:
  + canine\_arthritis.pdf - Joint treatments
  + feline\_renal\_disease.pdf - Kidney disease management

**C. FDA Reports (PDFs)**

* **Purpose**: Official medication/drug safety
* **Location**: data/unstructured/pdfs/fda\_reports/
* **Files**:
  + animal\_drugs.pdf - Approved medications
  + pet\_food\_safety.pdf - Nutrition guidelines

**D. Web Articles**

* **Kennel Clubs**: Breed-specific advice (akc\_health.html)
* **Vet Associations**: General pet care (avma\_pet\_care.html)
* **Breeder Forums**: Practical tips (maine\_coon\_health.html)

**E. Databases**

* **CSV**: Numerical health scores (ofa\_hip\_scores.csv)
* **JSON**: Treatment guidelines (wsava\_guidelines.json)

**LangChain**

a **framework** for building applications powered by large language models (LLMs). It **provides tools to connect LLM**s (like **Gemma-3**) to external data sources (PDFs, web pages, databases) and manage workflows.

1. **Document Loading**
   * Instead of writing custom code for each file type (PDF, HTML, CSV), LangChain provides built-in **loaders**:
     + PyPDFLoader extracts text from PDFs
     + CSVLoader reads tabular data
     + WebBaseLoader scrapes web pages
2. **Text Splitting**
   * Raw documents are too large for LLMs. LangChain’s RecursiveCharacterTextSplitter splits them into smaller chunks **while preserving context**:
3. **Prompt Management**
   * LangChain’s ChatPromptTemplate structures prompts dynamically:
4. **Integration with LLMs**
   * Simplifies calling models (Gemma-3) with consistent interfaces.

**FAISS**

**Definition**:  
FAISS (Facebook AI Similarity Search) is a **vector database** optimized for fast similarity search in high-dimensional spaces (like text embeddings).

1. **Stores Embeddings Efficiently**
   * Converts text chunks into numerical vectors (using all-MiniLM-L6-v2).
   * Compresses them for fast retrieval.
2. **Finds Relevant Context**
   * When you ask about "Labrador hip dysplasia," FAISS:
     1. Converts the query to a vector.
     2. Finds the closest-matching vectors (text chunks) in its index.
     3. Returns the most relevant research snippets.

docs = vectorstore.similarity\_search("Labrador hip dysplasia", k=3)

1. **Speed & Scalability**
   * FAISS uses **IVF-PQ indexing** (Inverted File with Product Quantization):
     1. **100x faster** than brute-force search.
     2. Handles thousands of vectors in milliseconds.

**Why This Combo?**

| **Task** | **LangChain Role** | **FAISS Role** |
| --- | --- | --- |
| Load PDFs/HTML/CSV | Uses PyPDFLoader, CSVLoader | — |
| Split text | RecursiveCharacterTextSplitter | — |
| Generate embeddings | Uses HuggingFaceEmbeddings | Stores embeddings |
| Search relevant info | — | Finds closest vectors to the query |
| Generate advice | Formats prompt for Gemma-3 | Provides context snippets |

**Key Advantages**:

1. **LangChain** = **"Workflow Manager"**
   * Handles document processing, prompting, and LLM calls.
2. **FAISS** = **"Super-Fast Search Engine"**
   * Retrieves relevant data in milliseconds.
3. **Vector Storage**
   * Uses 384-dimensional vectors (from all-MiniLM-L6-v2)
   * Compresses vectors using Product Quantization (PQ) to 8 bits per dimension
4. **Indexing**
   * IVF (Inverted File) index groups similar vectors into "buckets"
   * During search, only checks the most relevant buckets (not all vectors)
5. **Search Process**
   * Query → Embedding → Nearest Neighbor Search:
     1. L2 distance calculation between vectors
     2. Returns top-k matches (k=3 in your code)
     3. Takes <1ms even for 10,000+ chunks

**Chunks** are smaller, meaningful segments of text split from larger documents (like PDFs or web articles). They make it easier for the AI to process and retrieve relevant information.

* + Models like Gemma-3 can’t read entire books at once (they have a **context window**, e.g., 8k tokens).
  + Chunks ensure only the most relevant parts are fed to AI.

PDF → LangChain (extract text) → Split → FAISS (store) → Query → FAISS (retrieve) → LangChain (format) → LLM

**System**

**1. Core Idea**

the system is a **super-smart veterinary assistant** that:

1. **Remembers textbook knowledge** (Excel data)
2. **Constantly reads new research** (PDFs/websites)
3. **Combines both** to give the best advice

**2. Key Components & Why You Chose Them**

**A. Data Organizer (LangChain)**

**What it does**:

* **Collects** data from everywhere:

# Loads PDFs, HTML, CSV automatically

loaders = [PyPDFLoader(), CSVLoader(), WebBaseLoader()]

* **Prepares documents** by splitting them into digestible chunks (e.g., 1000-token sections with 200-token overlaps to avoid cutting mid-idea).

**LangChain :**

* **Standardizes workflows**: No custom code for each file type.
* **Optimized for LLMs**: Built-in tools handle LLM limitations (e.g., context windows).

**B. Knowledge Librarian (FAISS)**

**What it does**:

* Takes text chunks, converts them to **math vectors** (via all-MiniLM-L6-v2).
* Creates a **searchable index** (like a book's index but for math vectors).
* When you ask *"Labrador hip dysplasia"*, it:
  1. Converts your question → vector
  2. Finds the **3 closest vector matches** (research snippets)

docs = vectorstore.similarity\_search(query, k=3)

**FAISS :**

* **Speed**: 100x faster than searching raw text (uses IVF-PQ indexing).
* **Accuracy**: 95%+ match precision for veterinary terms.
* **Local**.

**C. Brain (Gemma-3 LLM + MiniLM Embeddings)**

**Two AI models work together**:

1. **MiniLM (all-MiniLM-L6-v2)**:
   * **Role**: Converts text → vectors (e.g., "hip dysplasia" → [0.2, -0.5, ..., 0.7]).
   * **Why chosen**:
     + Small (384-dim vectors) but accurate for medical terms.
     + Faster than BERT (1200 docs/sec vs 300).
2. **Gemma-3**:
   * **Role**: Generates advice using:
     + Excel data + FAISS-retrieved research.
     + **Structured prompts** to ensure professional tone:

**3. Technical Workflow**

1. **You ask**: *"How to care for a Labrador with hip dysplasia?"*
2. **System**:
   * Fuzzy-matches "Labrador" → "Labrador Retriever" (typo-proof).
   * **Retrieves**:
     + Excel: Standard care (glucosamine dosage).
     + FAISS: Latest research (e.g., *"2023 study: Omega-3 reduces inflammation"*).
3. **Gemma-3 combines both** into a natural-language response.