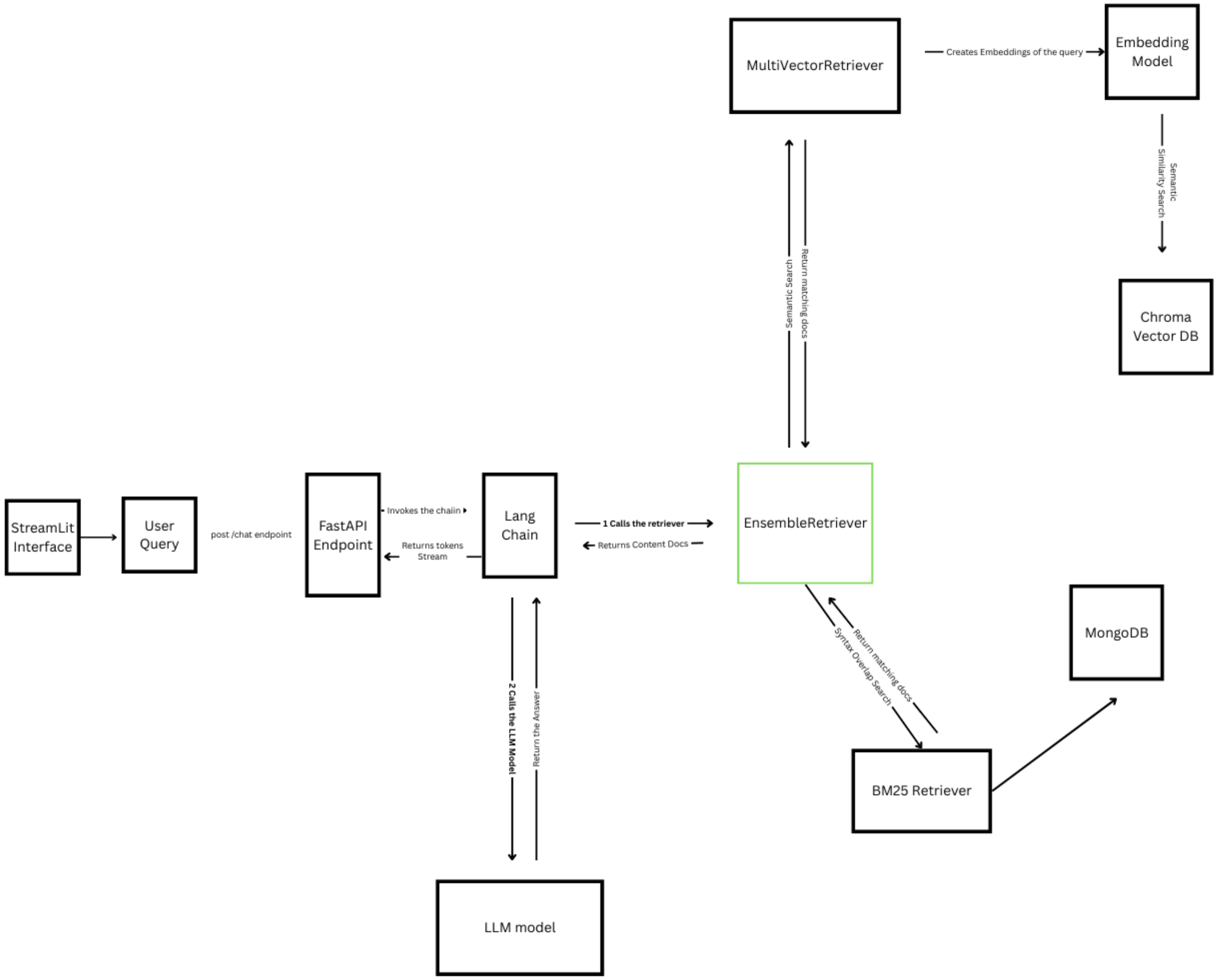
FoxBase task for Data Scientist role

General Overview of the System Pipeline:



An overview of the system inference flow:

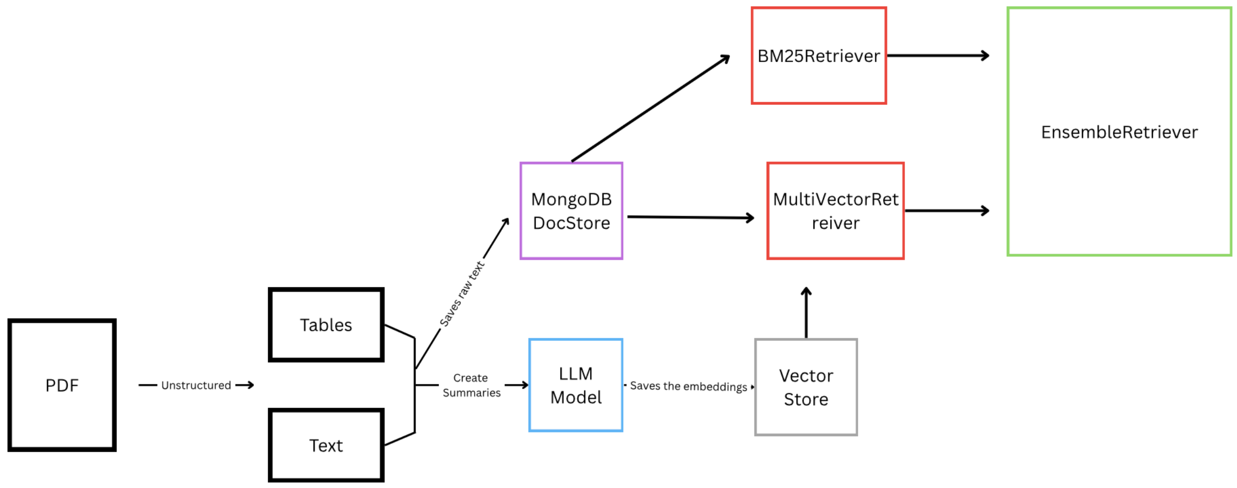
This RAG system integrates multiple retrieval methods to enhance response accuracy. The process begins with a Streamlit interface, where users submit queries, which are then sent to a **FastAPI** endpoint via a `/chat` post request.

The FastAPI server invokes a LangChain pipeline, which orchestrates the retrieval and response generation. The retrieval process is managed by an EnsembleRetriever, which calls two different retrievers: **MultiVectorRetriever** and **BM25Retriever**.

The MultiVectorRetriever leverages an embedding model to generate vector embeddings of the query and searches for relevant content in the **Chroma Vector DB**. Meanwhile, the **BM25Retriever** fetches relevant documents from **MongoDB** using traditional keyword-based search.

The retrieved documents are then combined and returned to LangChain, which forwards them to the LLM model for response generation. The **LLM** processes the input and returns a response **stream** back to the user via FastAPI and Streamlit. This hybrid retrieval strategy ensures both semantic and keyword-based search effectiveness, improving the quality of responses.

General Overview of the document processing architecture



The PDF gets passed to Unstructured library to extract Text and Table Elements, then creating a summarization using LLM, and embedding it to the Chroma DB Vector Store. While keeping the raw data in the MongoDB Doc Store. The MultiVectorRetriever uses the docstore and vectorstore for retrieval. Moreover, the BM25 Retriever (keyword matching) gets initialized with the documents from the docstore. And passing both to the EnsembleRetriever for the hybrid search.

System Components Overview

This Retrieval-Augmented Generation (RAG) pipeline is designed for processing Technical Manual PDF with high precision, extracting structured and unstructured content, and enabling hybrid retrieval (semantic + lexical). The pipeline leverages:

- **LLaMA 3.1 8B** (Open Source LLM) for summarization and response generation
- **Multilingual MiniLM-L12-v2** (Open Source Emb. Model) for dense vector embedding
- **Unstructured Library Integration** for high-resolution PDF parsing
- **MongoDB Docstore (raw texts/tables) + ChromaDB Vectorstore (summarized texts and tables)** for flexible retrieval
- **Hybrid Ensemble Retriever** combining **BM25** and **semantic retrieval**

- **FastAPI for REST API endpoint for the system,** by streaming and asynchronous processing.
- **Streamlit Interface** for easy interaction with the system.
- **RAGAS + Langfuse** evaluation of the system.

PDF Parsing & Chunking Strategy

Parser

- **Library**: `unstructured` (via LangChain)
- **Function**: `partition_pdf`

The **chunking strategy** in this RAG system is based on a high-resolution layout analysis using the **`by_title`** approach from LangChain's **`unstructured`** integration. This strategy segments the PDF content semantically by detecting section headers (e.g., titles and subtitles) and ensures structural fidelity by preserving layout elements like tables and images.

To maintain manageable context sizes, chunks are constrained with a **`max_characters`** limit of 4000, with soft splitting triggered at 3800 characters, and smaller fragments below 1000 characters are merged. The **`multipage_sections`** flag is set to **`False`**, meaning sections are split per page even if the same title spans multiple pages.. This results in clean, focused units of text optimized for summarization and retrieval.

This method results in having Chunks that are representing the Text Elements and Table Elements separately from the PDF. Moreover, I utilize the metadata provided from parsing the **tables** to parse it as an **HTML**. Thus, ensuring a better understanding of the tables for the LLM model used for summarization.

Unstructured uses layout extraction with YOLOX involves using the YOLOX (You Only Look Once X) model, a state-of-the-art object detection algorithm, to identify and extract the layout or structure of documents. **YOLOX** can detect and localize different elements within a document, such as text blocks, tables, images, and other visual components, by training it on a labeled dataset that includes these layout features. This technique can be particularly useful for automated document processing, where understanding the structure of a document is crucial for further analysis or data extraction tasks.

Prompt Engineering

The main prompt for the LLM inference is the following:

“ Answer the question based only on the following context, which can include text, and tables.
Context: {context_text}
Question: {user_question}

Always reply in German
Explain in paragraph the answer and the values the table provided in the context.

Reference the context used to answer between double quotations.

If the Question is not related to the content, reply saying that you can not answer this question.”

-> This Prompt ensures that the LLM only replies from the Content provided by the retriever, and reduces hallucinations.

Evaluation:

Based on the test set given, I conducted an extensive evaluation using Ragas and Langfuse. **RAGAS** (Retrieval-Augmented Generation for Assessment and Scoring) evaluates RAG (Retrieval-Augmented Generation) LLMs by assessing their ability to retrieve relevant information and generate accurate, context-aware responses based on external data. **LangFuse** is a platform designed for monitoring, analyzing, and improving the performance of language models (LMs) in production. It provides tools for tracking model interactions, detecting errors, and optimizing outputs, helping teams enhance the effectiveness of their LLM-powered applications.

Here are the evaluation results (**generated vs reference**):

The RAG Pipeline's performance is assessed using the following Ragas metrics:

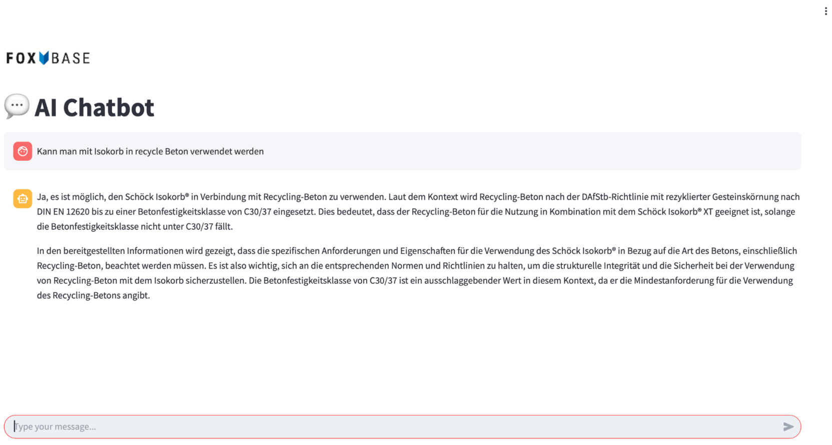
1. **Semantic Similarity (0.7679)**
   - This reflects how semantically close the generated answer is to an ideal response. A **0.77 score** indicates strong similarity but leaves room for improvement in making answers more precise and contextually aligned.
2. **Answer Relevancy (0.6341)**
   - This measures how well the generated answer aligns with the user's query. A score of **0.53** suggests moderate relevance but indicates room for improvement in ensuring responses are more directly tied to user intent.
3. **Faithfulness (0.8218)**
   - This assesses whether the generated answer stays true to the retrieved context. With a **high score of 0.82**, your system generally produces factually accurate responses without hallucinating information.
4. **LLM Context Precision Without Reference (0.7881)**
   - This metric evaluates how well the model extracts and utilizes relevant information from the provided context. A **0.78 score** indicates strong precision in retrieving and integrating useful details.
5. **Answer Accuracy (0.6559)**
   - **Answer Accuracy** measures how closely a model's response aligns with a reference ground truth for a given query. It is evaluated through two independent "LLM-as-a-judge" prompts, each returning a rating of 0, 2, or 4. These ratings are then scaled to a [0,1] range and averaged. Higher scores reflect stronger agreement between the model's answer and the reference.

Next Steps:

To overcome the limitations and improve this system, we need to experiment with the following:

- **Query Enhancement:** Adding a query enhancement component to the retrieval pipeline to ensure that the query is in the best form possible for the system.
- **Utilizing the metadata:** Utilizing the metadata of the documents for retrieval (page numbers, and sections, etc.)
- **Reranking the components:** The retrieved documents get reranked according to relevance.
- **Experimenting with other models:** Check whether other models can improve the performance.
- **Test other OCR methods:** such as Misteral OCR, etc. for better PDF parsing.
- **Perform summarization for Images and embed to the VectorStore.**
- **Dockerization**

Streamlit UI Interface:



Kann man mit Isokorb in recycle Beton verwendet werden

Ja, es ist möglich, den Schöck Isokorb® in Verbindung mit Recycling-Beton zu verwenden. Laut dem Kontext wird Recycling-Beton nach der DAfStb-Richtlinie mit rezyklierter Gesteinskörnung nach DIN EN 12620 bis zu einer Betonfestigkeitsklasse von C30/37 eingesetzt. Dies bedeutet, dass der Recycling-Beton für die Nutzung in Kombination mit dem Schöck Isokorb® XT geeignet ist, solange die Betonfestigkeitsklasse nicht unter C30/37 fällt.

In den bereitgestellten Informationen wird gezeigt, dass die spezifischen Anforderungen und Eigenschaften für die Verwendung des Schöck Isokorb® in Bezug auf die Art des Betons, einschließlich Recycling-Beton, beachtet werden müssen. Es ist also wichtig, sich an die entsprechenden Normen und Richtlinien zu halten, um die strukturelle Integrität und die Sicherheit bei der Verwendung von Recycling-Beton mit dem Isokorb sicherzustellen. Die Betonfestigkeitsklasse von C30/37 ist ein ausschlaggebender Wert in diesem Kontext, da er die Mindestanforderung für die Verwendung des Recycling-Betons angibt.