SatSureLLM

Table of Contents

1. About the Team

Sourab Daparti

2. Solution Submission Attempts

The Version Control Sheet can provide a better understanding of how this model was developed and how the entire idea progressed.

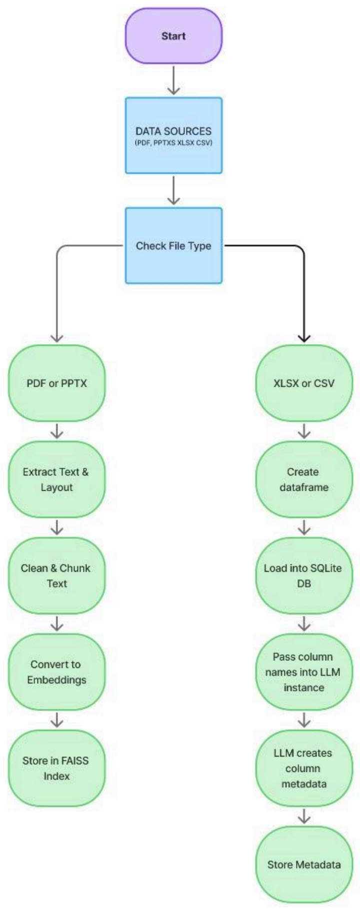
A much detailed version is attached here for a deeper understanding of the progression.

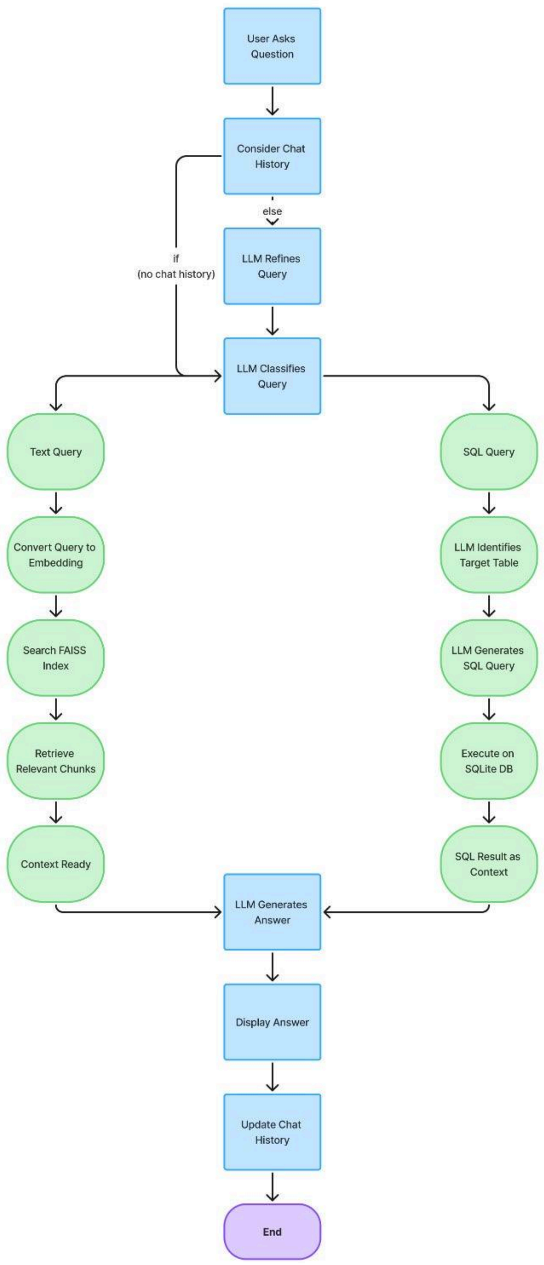
3. Problem Statement

The goal was to create an intelligent system that can understand and answer questions about SatSure's project proposal documents and related data. Users need to quickly find specific information, get summaries, compare details across different proposals (like PDFs and PowerPoint slides), and even get insights from data in spreadsheets (like CSV or Excel files). The system should be easy to use, provide accurate answers based *only* on the uploaded documents, and handle different types of questions – some needing direct information lookup and others requiring analysis of data tables.

4. Modelling Approach & Document Processing

Our solution uses a "Retrieval-Augmented Generation" (RAG) approach combined with a smart query understanding system. Here's how different document types are handled:





- **Document Ingestion & Initial Processing:**
  1. **PDFs (.pdf):**
     - The system reads text from each page. It also tries to identify and extract tables, converting them into a text format.
     - **Structure Awareness:** It attempts to understand the document's layout by identifying potential sections like "title page," "table of contents," "introduction," or "appendix" using keywords and page position. This helps provide better context later.
     - The extracted text and table information is then broken into smaller, overlapping "chunks." These chunks are tagged with their source page and identified section.
  2. **PowerPoint Presentations (.pptx):**
     - Text is extracted from all slide shapes (like text boxes and placeholders) and from the presenter notes section of each slide.
     - **Slide Merging:** A smart heuristic is used to combine slides that are just titles with their following content slide. This makes the information flow more naturally. For example, if Slide 5 is "Project Goals" and Slide 6 details those goals, their text might be merged into a single piece of information.
     - This combined text is then also broken into smaller "chunks," tagged with their source slide information.
  3. **CSV Files (.csv) & Excel Spreadsheets (.xlsx):**
     - These files are treated as structured data tables.
     - The system reads the data from each CSV file or each sheet within an Excel file directly into a temporary, organized database (SQLite).
     - **Understanding Table Columns:** An AI (SambaNova) analyzes a sample of the data and the column headers to understand what each column represents (e.g., "Project ID," "Task Description," "Budget Amount") and its likely data type (text, number, etc.). This "metadata" is stored alongside the table in the database.
     - *Unlike PDFs/PPTX, the full content of CSVs/XLSX is primarily queried via the database, not by breaking their entire content into text chunks for general text search.*
  4. **Creating Searchable Knowledge (for Text Documents):**

     For the text chunks from PDFs and PowerPoints, the system uses an "embedding model" (sentence-transformers/all-MiniLM-L6-v2) to convert each chunk into a numerical representation (an embedding). This allows the system to understand the *meaning* of the text.

    These numerical representations are stored in a special, fast-search index (FAISS).

- **Smart Query Handling (When you ask a question):**
    1. **Chat Context:** If you're having a conversation, the system first uses an AI (SambaNova) to make your latest question clear and complete by looking at previous messages.
    2. **Query Type Analysis:** It then uses the AI to figure out what kind of question you're asking:
        - Is it about information found in the text of PDFs/PPTs?
        - Does it require calculations or looking up specific data in the tables from CSVs/XLSX files?
    3. **Information Retrieval:**
        - **For Text Questions (PDFs/PPTs):** The system searches its FAISS index to find the text chunks most similar in meaning to your (refined) question.
        - **For Data Table Questions (CSVs/XLSX):** The AI identifies the most relevant table in its database (using the column understanding it gained earlier) and then writes a specific database query (SQL) to get the exact data needed from that table.
    4. **Answer Generation (Using SambaNova LLM):**
        - **Text Answers:** The most relevant text chunks (from PDFs/PPTs) are given to the AI, along with your question and chat history. The AI then writes a natural language answer based *only* on this information.
        - **Data Table Answers:** The result from the database query (e.g., a sum, a list of items) is given to the AI to explain it in a clear, easy-to-understand sentence or paragraph, also considering the chat history.
    5. **User Interface:** A web-based chat interface (built with Streamlit) allows users to upload documents, ask questions, and see the answers along with insights into how the system arrived at them.

This approach allows the system to use the best method for the type of question and the type of document, aiming for accurate and relevant answers.

Note: The metadata generation according to the directory is not implemented yet due to the lack of a central database, hence not mentioned in this document. Ideas and workflow will be proposed during the presentation.

5. Software Version and Reproducibility

- **Python Version:** Python 3.x (e.g., 3.9+)
- **Key Software Components:**
    - **Embedding Model:** sentence-transformers/all-MiniLM-L6-v2
    - **LLM (Q&A, Analysis, SQL Gen, Metadata):** SambaNova (e.g., Meta-Llama-3.3-70B-Instruct via openai library)
    - **PDF Processing:** pdfplumber
    - **Spreadsheet/Data Handling:** pandas
    - **PowerPoint Processing:** python-pptx
    - **Vector Indexing:** faiss-cpu (or faiss-gpu)
    - **Database:** sqlite3
    - **Web Interface:** streamlit
- **Configuration File:** System behavior is primarily controlled by config1.ini.
- **Environment Variables:** API keys (e.g., SAMBANOVA_API_KEY) managed via a .env file.
- **Reproducibility:** Requires installation of listed libraries, correct config1.ini, API keys, and data in ./Data/.

6. Supporting Data and Code Libraries

- **Input Data:** Files in ./Data/ (PDF, XLSX, CSV, PPTX).
- **Core Python Libraries:** os, re, logging, json, configparser, time, sys, io, contextlib, torch, numpy, pandas, faiss, sentence-transformers, pdfplumber, python-pptx, streamlit, langchain-core (for prompt structuring), dotenv, sqlite3, openai. (Note: requests is imported but not directly used for core LLM calls in this SambaNova version).

7. Intermediate Result Files

- **Text Chunks (PDF/PPTX):** JSON files in ./Faiss_index/ (e.g., document_name.json).
- **Embeddings (PDF/PPTX):** NumPy arrays (.npy) in ./Faiss_index/ (e.g., document_name.npy).
- **FAISS Indexes (PDF/PPTX):** Index files in ./Faiss_index/ (e.g., document_name.index).
- **SQLite Database:** Stores tables from CSV/XLSX. If sqlite_db_path is :memory:, it's temporary.
- **Log File:** rag_system_sambanova.log.

8. Findings

- **Hybrid Approach Strength:** Combining text-based RAG (for PDFs/PPTX) with direct SQL querying of structured data (CSVs/XLSX, guided by LLM-generated metadata and SQL) provides a powerful and flexible solution for diverse information needs.
- **LLM as a Multi-Tool:** The SambaNova LLM is central to the system, used for understanding user intent (classification, refinement), generating structured database queries (SQL), and synthesizing human-readable answers from both text and structured data results, all while maintaining conversational context.
- **Structured Data Integrity:** Moving from Pandas code execution to an SQLite backend with LLM-generated SQL offers a more robust, secure, and standardized way to interact with tabular data.
- **Contextual Understanding:** Features like chat history awareness and PDF/PPTX structural analysis (section tagging, title merging) aim to provide richer, more relevant context to the LLM, leading to improved answer quality.
- **Configuration and Modularity:** The system is designed with configuration files and modular components, allowing for easier adaptation and maintenance.

9. The Flaws and Solutions

1. The bias towards using chat history as context is very high

   Solution: Can be adjusted with some tuning. Need some user inputs for adjusting this dependency

2. Unable to classify between SQL and Text queries when no context/keywords are provided

   Solution: Requires a lot more data in a structured repository

3. Not Robust for it to be useful in multiple use cases

   Solution: A temperature slider will be added to cater the requirements of the user. This ensures a balance between factual retrieval and creative responses.

4. Contains no context apart from the system prompt about the company itself for it to act according to user needs

   Solution: Requires a lot more data in a structured repository

5. This SQL querying process can fail during complex table joins especially when we have multiple spreadsheets in the same project of the database without relevant context.

   Solution: The intent of question must be understood and the response generation must happen from a local index of the relevant folder in the database

6. Cannot identify and according to spelling errors in the user prompts

   Solution: We can use an NLP model to identify but making a small change in the system prompt would be efficient. Need to land on an approach.

7. Struggles to join text based answers and SQL based answers together and present an answer in cohesion.

   Solution: Parsing the question and classifying if the question is a multi stage retrieval question and passing the both the outputs into the final LLM instance to generate a cohesive answer.