Design Document: Question-Answer System for Unstructured and Semi-Structured Data.

Siddhika Sriram

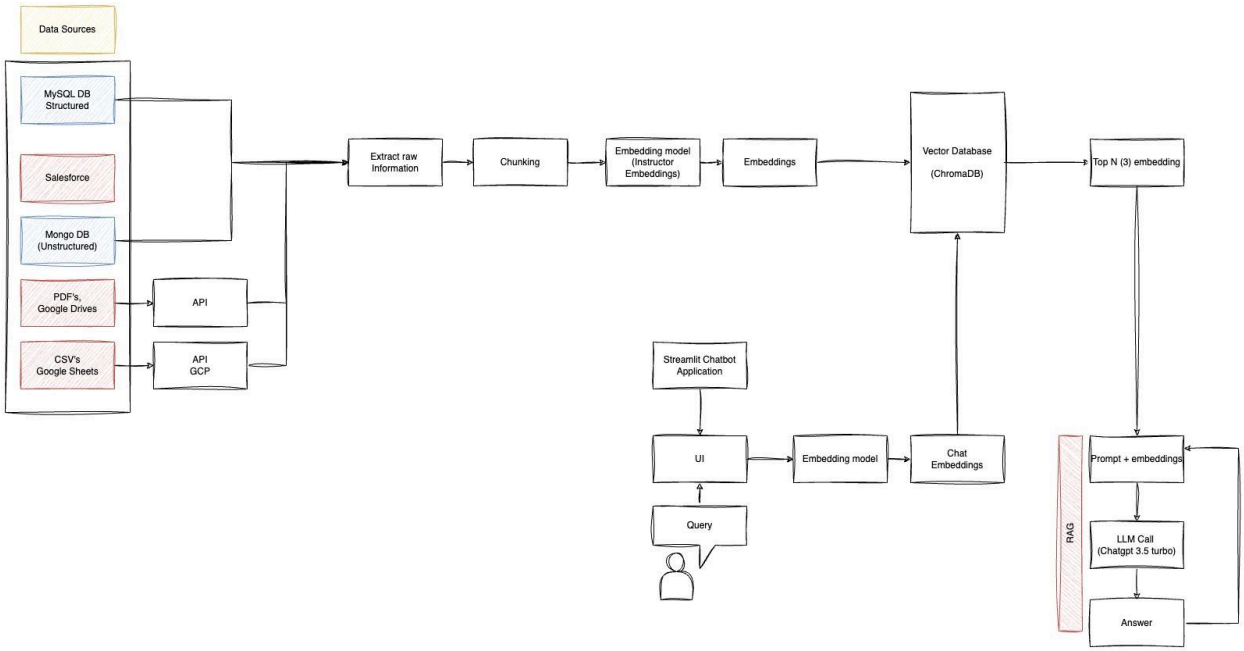
Introduction:

In today's enterprise landscape, vast amounts of data are generated across numerous platforms and applications. However, much of this data remains underutilized due to its unstructured or semi-structured format, creating challenges in extracting actionable insights. This project aims to address this issue by developing an intelligent **Question-Answer (QA) system** that can process queries in real time, leveraging data from various sources such as **file storage (Drive, SharePoint)**, **ticketing systems (Zendesk, JIRA)**, and **sales platforms (Salesforce)**.

The core objective of the system is to deliver **accurate, contextually relevant responses** to user queries, even when questions are unclear or span multiple applications. This capability is crucial for businesses seeking to optimize decision-making processes by utilizing historical data and previous interactions. By combining the power of natural language processing (NLP) with advanced machine learning algorithms, this system will transform raw data into useful information for **immediate and data-driven decisions**.

High-level Architecture

Here's an overview of the system's architecture:



Data Integration: The system collects data from a variety of sources, including cloud drives, internal collaboration tools (such as SharePoint), customer support platforms (such as Zendesk), and human resource management systems (such as Workday). The data is then analyzed to extract useful information. For this particular proof of concept, we have taken data from a single source but with different formats to represent different sources.

Data Integration and Fetching Mechanism

The system has integrated data from various enterprise sources, including cloud drives, collaboration tools like SharePoint, customer support platforms such as Zendesk, and HR systems like Workday. For the proof of concept, data from a single source was used to represent different formats, showcasing how diverse unstructured and semi-structured data can be processed to extract insights.

While Workato or MergeAPI were considered for stateful, real-time data integration, the prototype relied on static files stored in Drive due to computational constraints, handling one file at a time. In a full-scale implementation, Workato would be preferred for its ability to efficiently fetch and integrate data across multiple sources in real-time.

Data Processing and Chunking: After the data is retrieved, it is split down into smaller, more manageable pieces. Given the variety of data formats, specialized document parsers are used to read and analyze material for each individual source.

Embedding Model:

The collected data chunks are processed through an embedding model, generating vector embeddings that represent each word based on its context.

Options:

- BERT-based models (BERT, DistilBERT, and SBERT).
- GPT models (GPT-2, GPT-3, and GPT-4).
- Instructor Embeddings: Universal Sentence Encoder (USE)
- FastText/Word2Vec

Choice: Instructor embeddings (Hugging Face) are chosen because they can handle task-specific questions with greater precision. They are optimized for scenarios such as question-answering and semantic search by using specific training instructions. This allows for a better comprehension of the query's intent and improves semantic matching between inquiries and responses.

Vector Database for Embedded Storage:

A vector database is required to store the embeddings after the data has been segmented into chunks.

Options:

- Chroma DB.
- Facebook AI Similarity Search (FAISS)
- Milvus

Choice: Chroma DB was chosen because it is simple, easy to integrate, and suitable for real-time querying. It has a developer-friendly interface and facilitates rapid prototyping, making it an excellent fit for this system. FAISS may be considered for larger-scale installations because it has powerful search capabilities and GPU support, albeit at a higher complexity.

Retrieval-Augmented Generation:

To generate intelligent, context-aware answers, ChatGPT-3.5 Turbo is used to perform Retrieval-Augmented Generation (RAG). The model is used for real-time LLM calls, allowing the system to understand user inquiries and respond appropriately depending on the stored embedded data. This integration enables the appropriate handling of conversational and contextual inquiries across many data sources.

Constraints and Considerations.
Several restrictions were considered throughout the system design:

Training is Expensive: Because building models from scratch can be resource-intensive, pre-trained models such as instructor embeddings and GPT-3.5 Turbo are utilized to strike a balance between performance and cost.

Statelessness is problematic: A stateless system can cause context loss, particularly during multi-turn talks. This problem is addressed by leveraging embedding-based retrieval, which allows the system to track the conversation flow and respond to user inquiries coherently.

Ambiguous Queries: When a user submits an ambiguous inquiry, the system uses Natural Language Generation (NLG) techniques to continue the conversation. It asks the user for clarification or rephrases the response to promote more interaction.

Real-Time Queries: The system is intended for real-time querying, which means that replies are returned within seconds of user input. This is especially crucial in corporate settings where timely decision-making is necessary.

Multi-Application requests: User requests frequently include data from numerous applications. The system is capable of retrieving information from several sources and combining it to provide a comprehensive response.

Conversational or computational inquiries: The system can handle both conversational (for example, "What was the revenue last quarter?") and computational inquiries (for example, "Calculate the average sales over the past 6 months."). This guarantees that people engage naturally with the technology.

System Workflow Data Chunking: Different data sources necessitate different methods for accessing and storing content. After retrieving data, it is divided into digestible chunks to improve system performance during embedding production and querying.

Embedding Generation: The technology converts processed data pieces into vector representations using instructor embeddings. These embeddings are saved in Chroma DB, making them easy to retrieve during the querying phase.

Querying and User Interaction: The frontend is developed with Streamlit, which is a simple and straightforward web interface. Here is the process:

A user enters a query into a text input window in Streamlit.

Query Embedding: The system creates an embedding for the user's input using the pre-trained Instructor embedding model.
Search: The query embedding is compared to the stored embeddings in Chroma DB, with the top three most comparable embeddings (answers) returned.
Result Display: The system shows the user the top three most relevant replies, giving quick, meaningful solutions based on the requested data.

Conclusion:
This question-and-answer system is intended to deliver real-time, intelligent responses to queries based on enterprise data. The system attempts to provide high-quality replies by utilizing Instructor embeddings for task-specific learning, ChatGPT-3.5 Turbo for context-aware responses, Chroma DB for efficient embedding storage, and Streamlit for user interaction. To maintain a consistent user experience, the restrictions of real-time queries, statelessness, and dealing with ambiguous questions have been carefully considered. For future scalability, FAISS may be used to handle large-scale nearby searches, while more real-time data sources can be incorporated for a more comprehensive solution.