Information Technology Institute

Dsquares
KEEPING YOUR CUSTOMERS LOYAL

Dsquares AI Assistant

Supervised by: Eng. Abdelrahman Ghallab

Team Members

Karim Abdelhalim

Laila Hamdy

Nagwa Ahmed

Salma Mahran

Data Science Track – Intake 44

2023-2024

" وَقُلْ رَبِّ زِدْنِي عِلْمًا "

(سورة طه (114

Acknowledgment

We would like to express our deepest appreciation to our supervisor, Eng. Abdelrahman Ghallab, for his unwavering guidance, support, and mentorship throughout our graduation project. His expertise and dedication to the project have been instrumental in helping us successfully complete this project.

We would like to express our sincere gratitude to two esteemed supervisors who have guided and supported us during this stage in our career:

Eng. Toaa Gamal Eng. Eshraq Saeed

We would like also to thank Eng. Rana Dahish for her continuous support and guidance throughout the training program.

Table of Contents

Contents

Introduction

In a competitive world where time is money and efficiency is the key, companies search hard for the fastest and most efficient to do their tasks. While companies struggle with tabular data, unstructured data are more tricky and harder to deal with. Since many corporates have tremendous amounts of documents for their different project, employees may need to search for specific project documentation in a huge pool of documents for hours to find just a small piece of information. Dsquares choice to overcome this challenge is to utilize one of the most transformative technologies that have been making headlines which are LLMs (i.e., Large Language Models). Here is our contribution, we have developed a chatbot that utilizes the knowledge of an LLM model to search these documents and provide the user with answers that depend on RAG (i.e., Retrieval Augmented Generation) to build this knowledge base.

Problem Statement:

In today's data-driven business environment, Dsquares employees face significant challenges in efficiently engaging with and extracting relevant information from a vast corpus of project documents. These documents often contain critical insights, historical data, project specifications, and other valuable information necessary for informed decision-making and successful project execution. However, the sheer volume and complexity of the data make it difficult for employees to locate and utilize the information they need promptly.

This inefficiency not only hampers productivity but also increases the risk of overlooking important details, leading to potential errors and missed opportunities. The current methods of manual data extraction and analysis are time-consuming, prone to human error, and often insufficient to keep up with the fast-paced demands of the industry.

To address these challenges, there is a pressing need to develop and implement advanced solutions that can streamline the process of engaging with project documents. Such solutions should leverage state-of-the-art technologies to facilitate quick and accurate information retrieval, enabling Dsquares employees to make more informed decisions and improve overall project outcomes

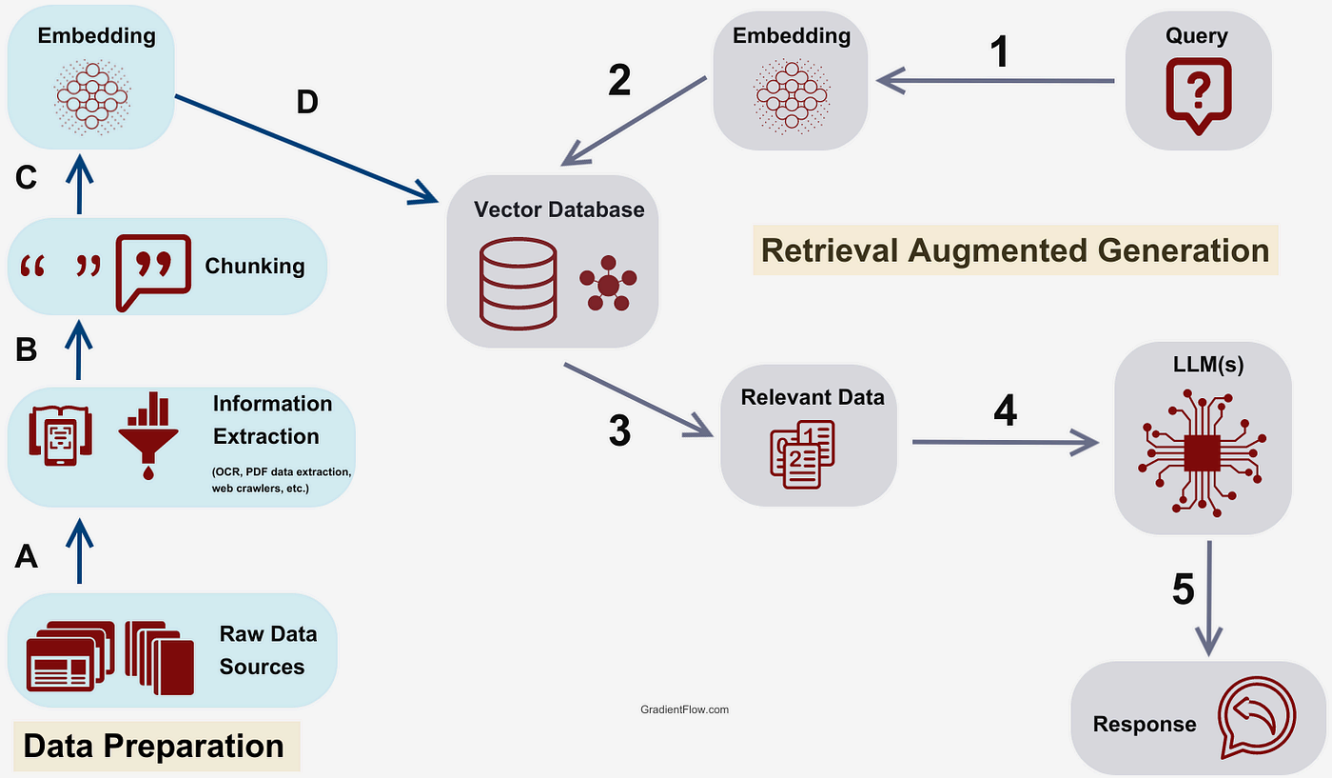
Objective:

To enhance employee interaction with project documents, we have developed an advanced chatbot designed to address the specific challenges faced by Dsquares employees in extracting and utilizing information from a large corpus of data. This chatbot leverages cutting-edge Large Language Models (LLM) and Retrieval Augmented Generation (RAG) technology to provide precise and contextually relevant answers to employee queries.

By integrating LLM and RAG, the chatbot is capable of understanding complex questions and retrieving the most pertinent information from vast and diverse project documents. This solution aims to significantly reduce the time and effort required for employees to locate critical information, thereby improving efficiency, accuracy, and overall productivity.

The implementation of this chatbot will empower employees to make faster, more informed decisions, minimize the risk of errors, and enhance their ability to successfully execute projects. Ultimately, this innovative approach will contribute to achieving better project outcomes and driving the overall success of Dsquares.

RAG Overview:



1. Data Preparation (A):
    - The process begins with gathering and organizing raw data sources. This can include various formats like text documents, web pages, databases, etc.
2. Information Extraction (B):
    - Relevant information is extracted from the raw data sources using techniques like Optical Character Recognition (OCR), PDF data extraction, and web crawling.
3. Chunking (C):
    - The extracted information is divided into smaller, manageable chunks to improve efficiency and accuracy during the retrieval process.
4. Embedding (D):
    - Each chunk is converted into a numerical representation called an embedding. This allows for efficient similarity-based search within the vector database.

5. Vector Database (E):

    - The embeddings of the chunks are stored in a vector database, which enables fast retrieval of relevant information based on query similarity.

6. Query Embedding (1):

    - A user query is also converted into an embedding.

7. Retrieval (2):

    - The query embedding is compared to the embeddings in the vector database, and the most similar chunks are retrieved.

8. Relevant Data (3):

    - The retrieved chunks are presented as relevant data to the language model (LLM).

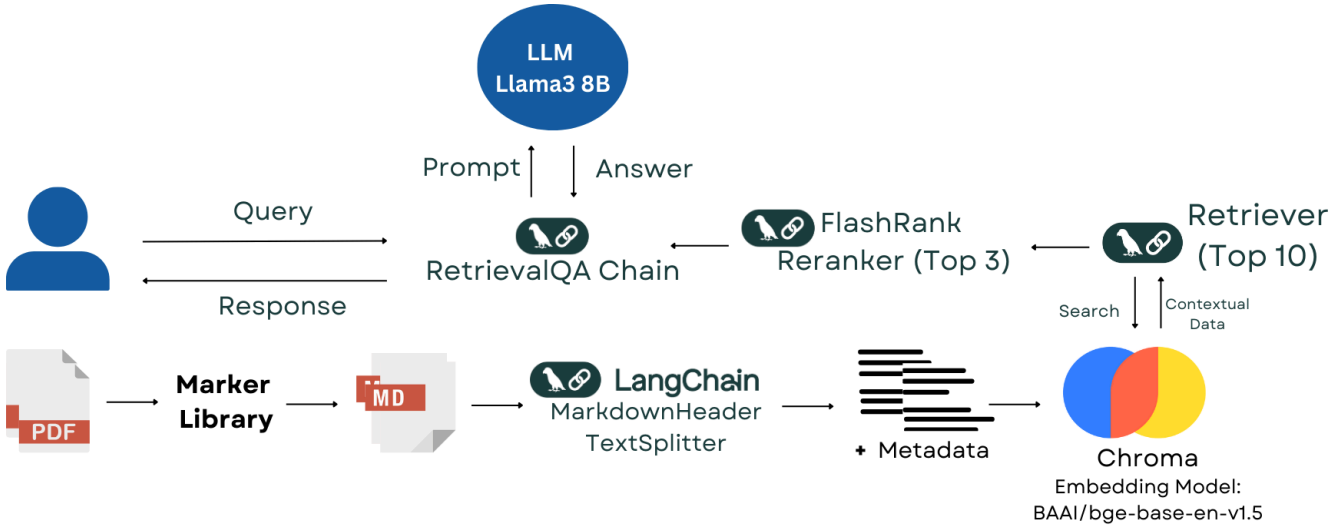
9. LLM(s) (4):

    - The LLM processes the retrieved data along with the original query to generate a comprehensive and informative response.

10. Response (5):

    - The final response is presented to the user.

Overview of Our System



The solution shown in the diagram illustrates our information retrieval system which uses Llama3 8B, a large language model (LLM). A user's query is processed through a RetrievalQA Chain, which uses a retriever to search contextual data stored in a Chroma database. The retriever initially selects the top 10 relevant documents, which are then refined to the top 3 by a FlashRank reranker. These documents, enriched with metadata, originate from a marker library containing PDF files that have been processed using LangChain's MarkdownHeader and TextSplitter tools. The final ranked documents are fed back to the LLM to generate a comprehensive answer to the user's query. This integration ensures accurate and contextually relevant responses by combining robust document retrieval and advanced language understanding capabilities. Followingly, we will explain each of the parts in detail.

Data Preprocessing

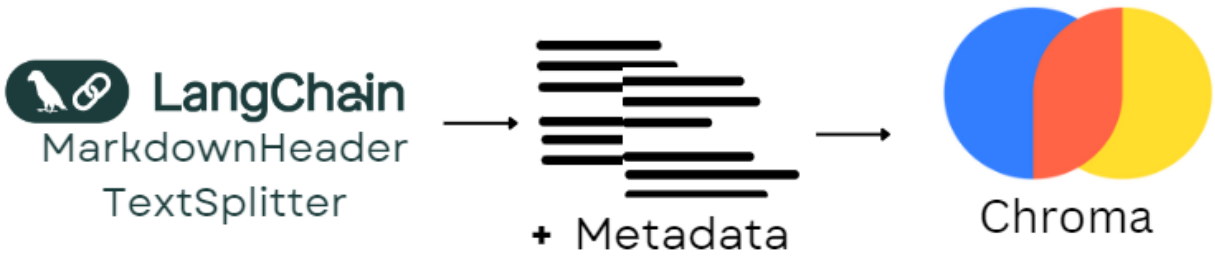
All projects documents were converted to PDF file format and then we utilized the "marker" library to convert these documents to markdown files therefore we can utilize "*MarkdownHeaderTextSplitter*" to split each file into chunks (nodes) of data. The idea behind using a markdown splitter is to let each chunk represent a section of the file not so we have chunks of different sizes but each one represents a complete piece of information not just a sentence with no meaning or just a few characters. Each node has its own metadata which consists of (header name and file name) which will help our model when retrieving these chunks to answer questions.

Since text data can't be understood by models, each chunk of data is represented by a vector of numbers called an "embedding vector". There are various embedding models but we have chosen to use "*BAAI/bge-base-en-v1.5*" to embed our text data from "*FastEmbed*" library since it's a lightweight and fast as well as accurate library for text embedding generation.

.

Not only documents are converted into vectors but also the query or the question of the user is converted to a vector.

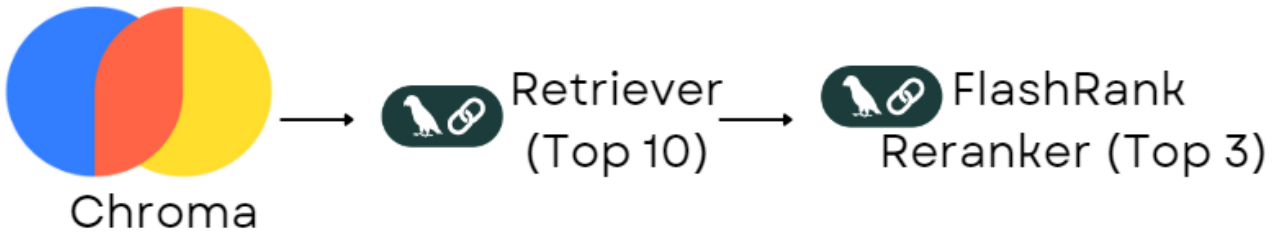
Vector Store Database



Usually, to perform a search on tabular data, we use some database engine that stores data and helps us to search using different algorithms for faster and better performance. The same applies to project document data, since each node as mentioned is represented by an embedding vector, we save the text and its embedding as well as its metadata in a database

called "*Vector Store Database*". As like embedding models, there are numerous vector store databases differing in what they save in addition to the embedding vector and the searching algorithms that they use. We have decided to use "*ChromaDB*" as our vector store database that will save our documents. ChromaDB performs embedding to documents and queries and performs the search to find top K relevant chunks (top 10 in our case) using some similarity score between the query and the chunks of data.

Retriever and Re-ranking



Although using similarity scores such as cosine similarity and other metrics is useful and fast to find relevant chunks of data, we could achieve better using an additional retrieval stage using reranking. Reranking involves using some deep learning model that takes the query and documents to calculate the relevance score. This score doesn't depend only on the similarity of words in each of query and document but also on semantic meaning of them even if they don't share many words. Since this process is time-consuming and computationally expensive, reranking is performed on retrieved nodes from the similarity search so we retrieve top 10 chunks using similarity and then perform reranking to find top 3 relevant chunks to pass them to the model. We have used "*FlashRank reranker*" from langchain.

Ollama

Regarding the model, we have used "*llama3 with 8 billion parameters*" and deployed it locally using the Ollama server. Llama3 belongs to the llama family of open-source models developed by Meta. Llama 3 is pre-trained on over 15T tokens that were all collected from publicly available sources

Streamlit App:

For the deployment of our solution, we used Streamlit to create a chatbot. Streamlit is a Python library that empowers developers to create web applications with minimal coding effort. Its strengths lie in building data-centric applications, making it ideal for rapid prototyping and iterative development. In this project, Streamlit serves as the front-end interface, providing a user-friendly platform to interact with the chatbot.

The Streamlit app is designed to create a conversational interface for users to interact with the chatbot while incorporating robust security measures and conversation history. It utilizes the underlying components, including the vector store, language model, and retrieval system, to deliver informative and relevant responses.

Key functionalities of the Streamlit app:

- User Authentication: To protect sensitive data and ensure authorized access, the app implements a secure user authentication process. This typically involves user registration, login, and password management.
- User Input: Authenticated users can input their questions into a designated text field.
- Chat Interface: A chat-like display presents the conversation history, including both user queries and chatbot responses. The app maintains a conversation history for each user to enable context-aware responses.
- Response Generation: Upon receiving a user query, the app processes the input, considering the conversation history, and interacts with the backend components to generate a response. The response is then displayed in the chat interface, and the conversation history is updated.
- New Chat Button: To initiate a new conversation, the app provides a "New Chat" button. Clicking this button clears the current conversation history for the user.

Future Work

Managing Various File Formats: Handling different file formats like markup and ppt files…etc.

Uploading files via GUI: The ability to upload files via GUI and the vector database will add this document only without performing embedding for previously added documents.

Exploring Different Embedding Models: using different embedding models with different dimensions may help improve the performance of the model

Experimenting with Alternative Vector Databases: using different vector databases that may be more suitable for our use case and comparing them to find the best. An example of them is the FAISS vector database which is known for its high performance.

Investigating LLMs with More Parameters: Usually models with a larger number of parameters perform better but they require more resources. We recommend experimenting with "*llama3: 70 billion parameters*".

Performing Evaluation: We have performed human evaluation. There are various evaluation metrics that can be used for RAG such groundness, context relevance, answer relevance, QAcorrectness…etc