Documentation

* developed by utkarsh kuamr

Q. steps taken to create this workflow

a. based on the project requirements, an early prototype was developed in main.py, where cli was used to test the calling of mistral 7b model from hugging face spaces and check the quality of text generated. further development led to use of Langchain framework to create a pipeline of rag application, where pdf text was extracted, pre-processed and stored in faiss vector database. later, addition of kbc was established to add the possibility of checking the current level of understanding. fast Api served as an endpoint for backend and stream lit served as the frontend option. finally, using sarvam.ai API key the generated text was translated INTO HINDI and converted into audio file.

Q. future development

A. making the system more robust to queries is to ESTABLISH so that to skip the possibility of prompt injections, frontend can be more fleshed, maybe using js or react language. frontend can be more gamified in the kbc setup, adding the possibility of levels, scores, timers to engage the child as a friendly learning app, not another text only platform.

1. document\_loader.py

Dedicated to the loading and processing of documents, such as PDFs. It allows extracting the needed content and metadata so that essential information does not get lost but is divided into manageable pieces. All this will serve to make future interactions easier but at the same time enhance the performance of information retrieval at the later stages of the application.

2. embeddings.py

This module generates text embeddings and indexes a dataset to support rapid search with the FAISS library. This transforms text into vector representations to help in the fast retrieval of similar texts in order to make handling user queries easier. The ability to produce correct contextual context mappings is also critical for checking the relevance of questions against a baseline set as it maps user interactions in the correct context of the document.

3. main.py

This module acts as an entry point to the application and governs the workflow at a general level. It is the part responsible for the initial loading of documents and embeddings, which subsequently allows for user interactions. Being a command-line interface, it enables users to query what they might want to know about a given topic, such as sound and quietly exits when required to ensure excellent UX.

4. query\_handler.py

This module analyses user queries over the content of the documents. It is responsible for the analysis of the user's questions with respect to the document pieces, and it retrieves information in summarized answers. Through this, the approach fosters the ability of the user to make the appropriate conclusions from the materials offered, thereby improving on the experience in interaction.

5. student\_kbc.py

This module generates quiz questions from the document content with an educational engagement focus. It measures comprehension and retention of the material by developing multiple-choice questions from appropriate text chunks. This feature is very useful to people who want to test their knowledge through interactive assessments.

6. api.py

Made with Fast API, this module makes communication between the client-backend service very smooth. It provides different endpoints for handling queries, generating quiz questions, and text translation. This structure does ensure efficient data retrieval but supports very diverse functions to enhance the overall interactivity of this application.

7. speech.py

It incorporates text-to-speech functionality, such that the application can listen to answers provided because of generating from the users' queries. It offers a great way through which users may enjoy relating with their content if presented in the audio form for the text responses. The feature is quite very helpful to those who learn by audition as well as enhances the usability of the application through accommodation for preferences in different learning ways.

8. frontend.py

The module is built keeping the user interface as friendly as possible; it applies Streamlit to allow for easy interaction. Inbuilt functionalities also include searching for notes, text-to-speech conversion, and the capacity to formulate quizzes, all on one platform that allows seamless interaction with the application. Such attributes as instant feedback make it possible for users to access more features and engage much more intensely with the system for the same amount of time.