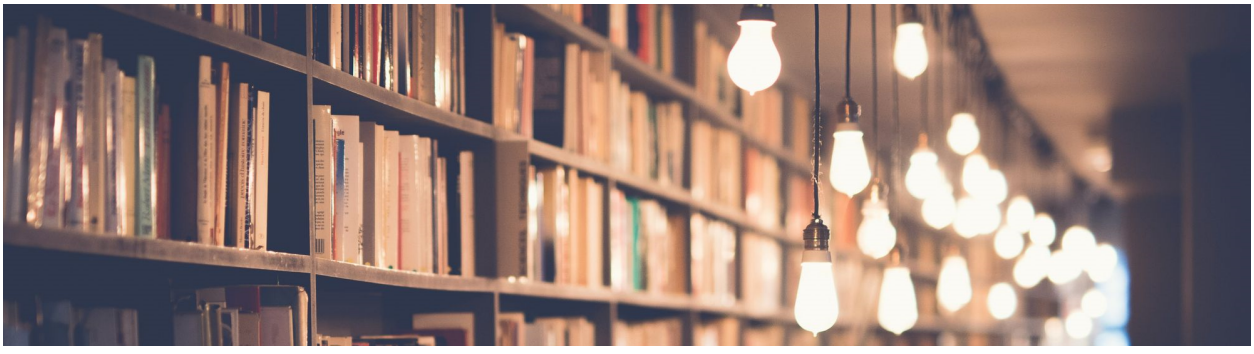
Project 3

Report: Development of a Multimodal AI Retrieval Chatbot capable of extracting knowledge from videos.

Author: Pedro Traba

1. Introduction

This report details the development of a multimodal AI chatbot designed to handle video download/upload, transcription, diarization, and question answering. The chatbot processes video inputs, extracts and transcribes audio, and stores the processed data in a vector store. In this project multiple AI models and libraries, including LangChain, LangSmith, Whisper, and Pyannote, are utilized.



2. Objectives

The primary objective of this project was to develop a chatbot that could:

- Handle both YouTube and local video uploads.
- Transcribe and diarize video content.
- Answer user queries about the video content.
- Summarize videos and extract specific parts.
- Translate video content into different languages.

3. Methodology

3.1 Functional Programming Approach

To manage the complexity of the application, a functional programming approach was adopted. This method ensured that each component of the system was developed as a distinct function, facilitating easier integration and maintenance.

3.2 Video Processing Functions

The initial phase focused on creating functions to download YouTube videos, given that most users would likely use them rather than uploading their own. The functions for local video uploads were developed much later into the project. A lot of work was required to handle the data properly since the very first step.

3.3 Audio and Metadata Extraction

The core functionality involves extracting audio and metadata from the videos. Metadata extraction was implemented to gather useful information, while the audio extraction was necessary for the transcription process. Handling empty Metadata was also necessary, since there was a possibility that not all videos would contain the same.

3.4 Transcription and Diarization

Transcription was performed using the Whisper model, known for its accuracy in understanding spoken language. Diarization, the process of segmenting the audio based on different speakers, was implemented using Pyannote. This dual approach ensured high-quality transcriptions with accurate speaker identification.

3.5 Data Integration

All extracted and processed data were integrated into a vector store using Chroma. Initially, Pinecone was considered for this purpose, but after going through various issues, Chroma ended up being the better option and the chosen one. All available documents (metadata, transcription, and diarization) data was combined into a Chroma database.

3.6 Retrieval and Augmentation

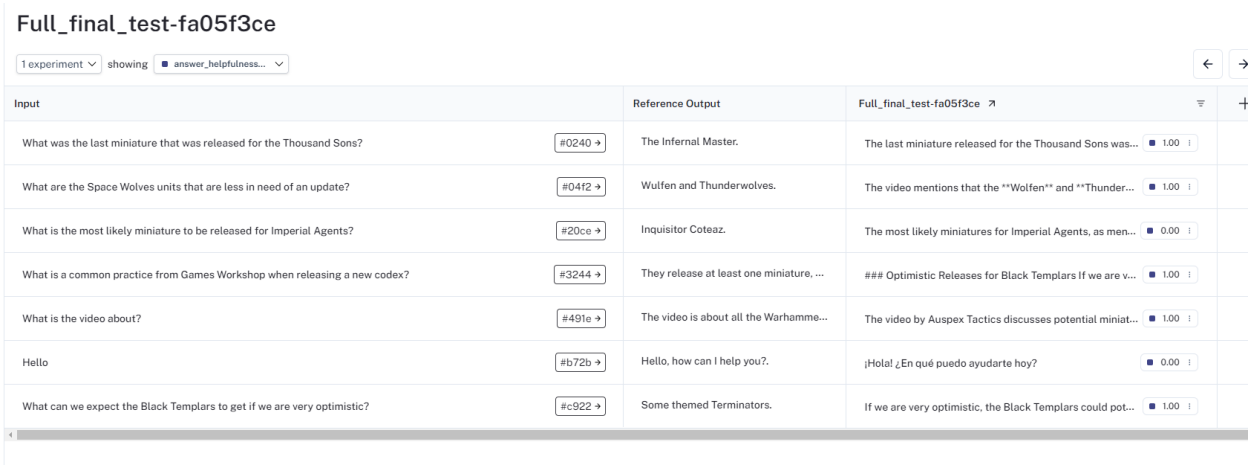
A retriever was developed to handle user queries, retrieving from the ChromaDB. This allowed the chatbot to perform retrieval augmentation.

3.7 Agent and tools

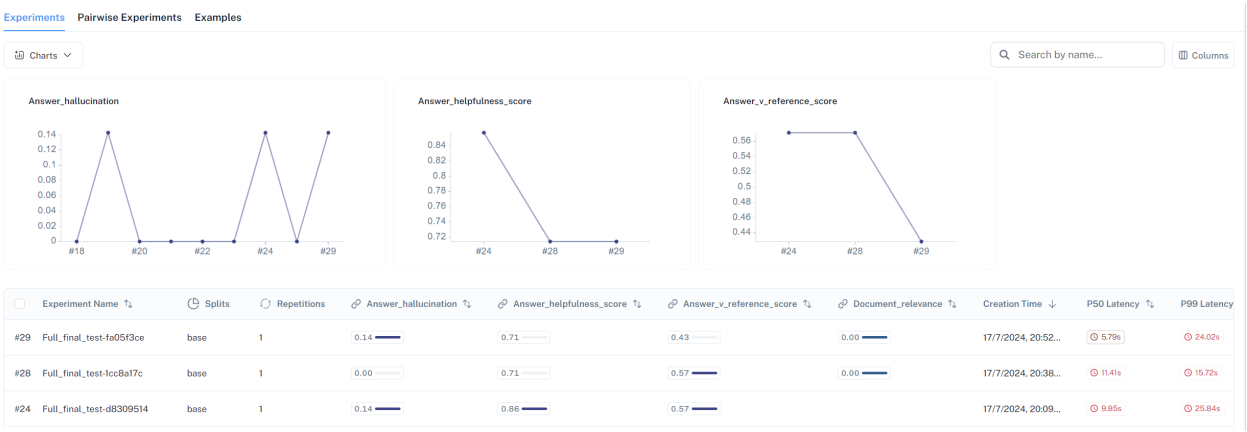
A ReAct agent was created to become the interactive Chatbot. It was added memory, a prompt and a tool to use the retriever whenever necessary.

3.8 Evaluation and Testing

https://smith.langchain.com/public/e76364d8-24fe-433e-b340-473040c840d3/d



While a lot of what was done involved human testing of the Agent’s output, LangSmith was used for the evaluation and testing of the chatbot's performance. An evaluation dataset containing 7 questions and answers from a particular youtube video was created. Hallucination and Document Relevancy couldn’t be correctly evaluated since I couldn’t manage to provide the evaluators with the Agent’s retrieval context.



4. Implementation Details

4.1 Whisper Model Selection

Different Whisper models were employed based on the length of the video and the desired balance between speed and accuracy. Longer videos utilized lighter models to expedite processing, while shorter videos leveraged more accurate but slower models.

4.2 Memory and Conversation Handling

Memory was implemented to allow the chatbot to maintain context over a conversation. This enabled the chatbot to retrieve information from the conversation memory or the retrieval agent as needed, enhancing the user experience.

4.3 Handling Multiple Languages

The chatbot was tested with videos in both English and Spanish. Despite initial concerns about transcription quality in Spanish, the retrieval performance was satisfactory, demonstrating the chatbot's multilingual capabilities.

4.4 Deployment Challenges

Deployment was attempted using Docker and Streamlit. However, issues related to API keys and configuration complexities hindered successful deployment. As a result, the final demonstration was conducted using Jupyter Notebook.

5. Results

The chatbot demonstrated impressive performance in terms of transcription accuracy, speaker diarization, and retrieval capabilities. It successfully answered queries and provided summaries for various test videos. However, some issues with language consistency and deployment remain to be addressed.

6. Challenges and Solutions

Several challenges were encountered during development:

- **Pinecone Integration**: Initial issues with Pinecone led to the adoption of Chroma for the vector store.
- **Whisper Installation**: Compatibility issues with libraries required troubleshooting and reinstallation of components.
- **Data Flow Management**: Ensuring data was in the correct format for each processing step was challenging but resolved through iterative testing.
- **Deployment**: Difficulties with Docker and Streamlit highlighted the need for further refinement in deployment strategies.

7. Next Steps

Future enhancements for the project include:

- **Improved Deployment**: Resolving deployment issues to enable wider accessibility.
- **Computer Vision Integration**: Adding functionality to analyze visual content in videos.
- **Speech-to-Text and Text-to-Speech**: Implementing these features for better accessibility and user interaction.

- **Performance Tuning**: Fine-tuning model selection thresholds and runtime configurations for optimized performance.
- **Enhanced Multilingual Support**: Improving language handling to ensure consistent responses in the user's preferred language.
- **Runtime Config**: Implement Config changes to the ReAct agent during runtime based on user actions. This could be a way to enforce the agent to use a specific language.

8. Conclusion

There was a lot of enthusiasm and effort invested on this project. The result, while overshadowed by the problems with deployment, is quite satisfactory because of its possibilities and power.