Report on Project 3 - Business case
Multimodal AI Chatbot for YT Video QA
Veritasium-Chatbot

By Faisal Hammad

Introduction

The chatbot project was developed to enhance user interaction with the content of the YouTube channel Veritasium. This report outlines the steps taken to develop the chatbot, the agents involved, the memory system, and the results of its deployment and evaluation. The primary goal was to provide users with quick access to relevant information, summaries, and external context without the need to watch entire videos.

Project Steps

1. Data Acquisition and Initial Categorization

The first step in the project involved acquiring data using the Google Cloud Console, specifically the YouTube Data API (YouTube Data V3). Data for approximately 350 Veritasium videos was collected. This data, comprising titles and descriptions, was then categorized using GPT based on the information provided in the titles and descriptions. The initial categorization helped in organizing the videos into relevant groups, making subsequent steps more manageable.

2. Adding Transcriptions

Transcriptions for the videos were added using yt_dlp and OpenAI's Whisper-medium model. This step ensured that the chatbot could access the full textual content of the videos, enabling more accurate categorization and summarization and paving the way for the main functionality of the Chatbot.

3. Recategorization, Preprocessing, and Vectorization

With transcriptions in place, the data was recategorized using GPT based on the more comprehensive content now available. The data was then preprocessed and cleaned to ensure consistency and accuracy. The following steps were taken to chunk the data and add it to Pinecone:

1. **Custom Text Splitting**: Transcriptions were split into chunks using a custom text splitter function that ensures chunks have some overlap, maintaining context continuity.
2. **Summarization**: Each chunk was summarized using a BART LLM model to generate concise and informative summaries.
3. **Metadata Enrichment**: Metadata, including video ID, title, description, URL, category, transcription chunk, summary, and publication date, was added to each chunk.
4. **Embeddings Generation**: The OpenAI embeddings model was used to generate vector embeddings for each chunk.

5. **Data Cleaning**: Non-alphanumeric characters were removed from the transcriptions, titles, and descriptions to ensure clean and uniform data.
6. **Batch Processing**: Chunks were processed in batches to avoid memory issues and ensure efficient handling of the data.
7. **Upsert to Pinecone**: The chunks, along with their embeddings and metadata, were uploaded to Pinecone in batches. Each chunk was assigned a unique ID and stored in Pinecone, facilitating efficient retrieval.

4. Agent Development and Testing

The next phase involved creating individual agents and testing them, each with a specific purpose:

a. Main Agent (Retriever with Context Integration)

- **Purpose**: To answer user queries by retrieving relevant information from Veritasium videos and integrating it with the context for accurate and comprehensive responses.
- **Strength**: Ensures users get detailed and educational responses by leveraging a combination of retrieved content and existing context.
- **Function**: It retrieves relevant documents from Pinecone using a QA chain, combines the retrieved texts with the current context, and formulates a response.

b. Fetch Agent

- **Purpose**: Utilizes metadata to retrieve information such as video URLs based on keywords, categories, or descriptions.
- **Strength**: Essential for quickly locating specific videos and ensuring users can easily access relevant content.
- **Function**: For instance, if a user requests a video recommendation from the physics category, this agent fetches relevant videos and provides the URLs.

c. Video Summarizer Agent

- **Purpose**: Provides summaries or key points from videos.
- **Strength**: Useful for users who want a quick overview of video content without watching the entire video.
- **Function**: When a user requests a summary, the agent pulls all relevant chunks with the same prefix (video ID), joins the transcriptions in order and generates a comprehensive summary via GPT.

d. External Knowledge Retrieval Agent

- **Purpose**: Provides answers to questions not covered by the Veritasium video documents.
- **Strength**: Ensures users get accurate and comprehensive answers even when the required information is outside the current dataset.
- **Function**: If the requested information is not in the vector store, the agent informs the user and retrieves information from external sources like Wikipedia.

e. Allocate Agent (Part of Orchestration)

- **Purpose**: Decides which agent should process the query based on similarity using sentence transformers.
- **Strength**: Ensures that users receive the appropriate response tailored to their request.
- **Function**: Processes the query and decides the appropriate agent to handle the task. It also checks if the answer is already in memory and responds accordingly if it is found.

5. Memory

- **Purpose**: The memory system was integrated to keep context throughout the interaction and to reduce token usage, thereby making the chatbot more efficient and user-friendly.
- **Type of Memory**: The type of memory used is Conversation Summary Buffer Memory. This system retains summaries of past interactions, allowing the chatbot to refer back to previous conversations, ensuring continuity and relevance in responses.

6. Orchestration

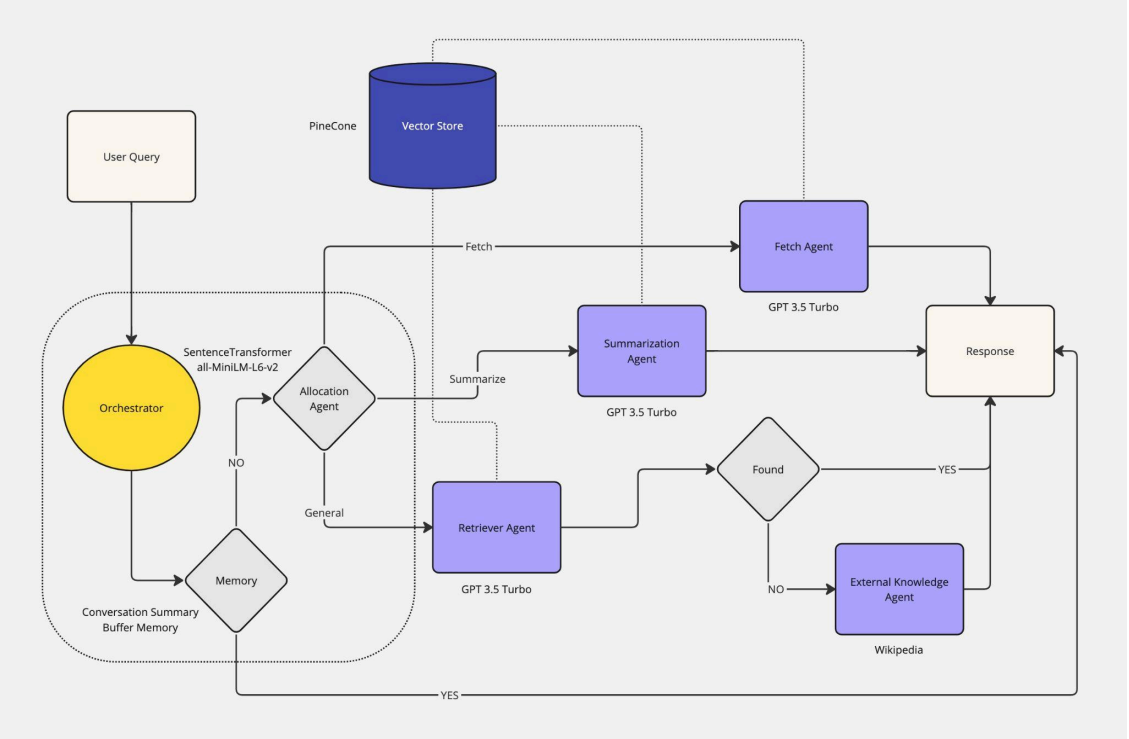
The individual agents were integrated through orchestration, managed by the Orchestration Agent. This setup ensured seamless interaction between the agents and memory, providing a cohesive user experience. The Orchestration Agent determines the appropriate agent to handle each query based on the nature of the request, optimizing the response process. Here's a detailed explanation of how the orchestration works:

- **Orchestration Agent Functionality**: The Orchestration Agent is designed to allocate user queries to the appropriate agent (Retriever Agent, Fetch Agent, Video Summarizer Agent, or External Knowledge Agent) based on the query's context and content. The agent also integrates memory functionality to maintain context and save on token usage.

7. Deployment Testing and Evaluation

The chatbot was deployed locally with Flask & Docker for initial testing and debugging. Following successful local deployment, it was deployed on Railway for broader access and further evaluation. The final step involved testing of the deployed both via langchain’s QAEvalChain to ensure stability and accuracy. Average Success Rate (response vs reference) was at 72.38%

Flowchart of the Chatbot



Conclusion

The development of this chatbot involved deep planning and execution, from data acquisition to deployment and evaluation. The various agents developed served distinct purposes, collectively enhancing the user's ability to interact with and retrieve information from Veritasium video content efficiently.

This project showcases the potential of combining advanced ML & AI models alongside cloud-based services to create a powerful tool for managing and interacting with video content from the Veritasium YouTube channel.

Room for Improvement

Future enhancements could include:

- Improving the Fetch and Summarization speed for better UX.
- Employing an agent to deal with ambiguous user queries.
- Enhancing the orchestration/allocation logic for even more precise agent allocation.
- Improving memory usage and adding mechanisms that would override responding from memory when user requests.
- Upgrading HTML and Style for a nicer UI.
- Using more advanced and accurate Evaluation techniques.