Lab - RAG youtuber

Purpose of this lab is to use the knowledge of AI engineering and data engineering to create a RAG chatbot on real world scenario. The scenario is a youtuber that teaches data engineering, which wants the followers of his youtube channel to be able have an interface for asking questions about the content of the videos. Idea is that this will enhance the overall learning experience.

The Youtuber, (look at his T-shirt and see his nerdy humour)

Data description

The Youtuber have collected data in the form of transcripts when recording the videos. These are just a small fraction of all the videos he has created, but should suffice for a proof of concept (PoC), to show that the idea is viable. The transcripts are collected using speech to text technologies during recording, which have its limitations. Especially it has problems with technical terminologies, and maybe some wordings due to The Youtuber not being clear enough. Further there are some post processing done to the transcript such as filler word removals.

Github repo

Create a separate public github repository for this lab.

Tasks

As The Youtuber is knowledgeable in the data engineering field, he has divided into several tasks that you should do. You should watch the videos, work through and deeply understand the lectures in 07-11, they will help you a lot. Keep the base structure of your code as in lecture 10, but you can of course add scripts when needed.

Task 0-3 and 5-6 are mandatory for G, task 4 is bonus if you aim for VG.

> **Tips:** use a jupyter notebook to explore different parts when you are working with the different tasks

Task 0 - data ingestion

First step is to understand the dataset and then setup a vector database (lancedb) that is our knowledge base. Then ingest the data into lancedb and also store the vector embeddings there.

Task 1 - build the RAG

Build a retrieval augmented generation (RAG) using PydanticAI that answers questions based on the transcripts. Try to make his personality similar to The Youtubers, so there is a nice feeling that the user is interacting with The Youtuber.

Task 2 - serve the RAG

Create an API using FastAPI to serve the RAG chatbot. Consume this API using either Streamlit or Taipy.

Task 3 - serverless deployment of the API

Deploy your API to Azure Function and connect the deployment to streamlit locally.

Task 4 - turn it into MVP (BONUS)

In this task you get to go further, and explore more things than have been covered in the lectures. We'll go from proof of concept (PoC) to minimum viable product (MVP), which The Youtuber have specified should be:

- deploy the frontend to Azure Web App
- add memory to the rag (it's okay that the memory stays for a single session)
- you should have a history endpoint in the API to keep track of the history
- make it like a chatbot where all questions and answers can be viewed
- The Youtuber wants to have an endpoint to get youtube description, which is a summary of the video
- The Youtuber also wants to have an endpoint to get 20-40 keywords about a particular video which he can paste into Youtube as tags
    - format of the tags are `keyword1,keyword2,keyword3,...`

Don't need to do, but if you have more time over you can explore

- multimodal inputs (image, audio)
- audio conversation with the RAG bot

Task 5 - a nice README

Create a nice README about your project. It should be short, but contain some highlights of the lab. Explain the following

- how to use the code to get it up and running
- some interesting screenshots
- some overview of how things are connected

Task 6 - video pitch for stakeholder presentation

Create a video 5-10 min long where you record the screen and you do a demo of your PoC/MVP, go through your repository and solutions to the taks. Your target audience should be The Youtuber and its data team. Explain with correct computer science terminology but keep it simple.

The Youtuber and the team have limited amount of time, so keep the time limit stated above. If you have done task 4, you should present it in the video as well both a demo and short explanation of the code structure.

For simplicity use microsoft teams and open up a meeting with yourself, then share screen and record. Afterwards download the video file and upload to your learning platform. If the file size is too big then you could upload it to youtube but keep it unlisted. Or use google drive, onedrive, dropbox or similar, but make sure that anyone with the link can view the file.

LLM usage

LLMs for coding are allowed, for smaller parts, for generating ideas, but not for solving entire parts. Very important is that you understand the code. If there are parts where you use LLMs for coding, it's important that you make a comment that it is LLM generated.

Submission

Hand in the following to your learning platform:

- a link to your public github repository for this lab
- a video file or link to the video

Grading

If you have taken ideas of codes from someone or found them online, it is **important** that you state the source and understand how the codes work. Write a comment next to these codes.

Criteria for G:

- solved tasks 0, 1, 2, 3 and 5-6 correctly
- the video is clear and easy to follow
- several relevant commits with descriptive commit messages

Criteria for VG:

- also solved task 4 correctly
- the coding is well structured and holds overall high quality
- DRY principle is followed and code is modularized well
- also your video, you use correct computer science, data engineering and AI engineering terminology