BYOSC Build Your Own Scalable Chatbots

Final Report

FEDERICO GIARRÈ MATTEO CIRCA
fgiarre@kth.se | circa@kth.se

February 5, 2024

School of Electrical Engineering and Computer Science

Contents

List of Acronyms and Abbreviations

LLM	Large Language Model
PEFT	Parameter Efficient Fine-Tuning
RAG	Retrieval-Augmented Generation
VectorDB	Vector database

1 Problem description

When preparing for university exams, having a partner has been proven to be essential to discover knowledge gaps and clarifying specific doubts on the topic treated during classes. While chatbots based on LLMs such as ChatGPT, Phind and Clod are providing help to students already, they cannot provide a lecture/material-specific help on the students' university courses. We propose to create a system to fine-tune chatbots on specific material of specific courses. Thanks to this, we will create study buddies for the courses of a typical university student, able to answer doubts, generate questions and more.

2 Tools

The tools involved in this project are as follows:

- Programming Language: Python everywhere.
- Platform: Jupyter Notebook for testing and Python files for the final deliverable.
- Batch Processing: pandas library in Python.
- Storage: locally for testing, a Cloud service for the final deliverable (Google Drive).
- UI: Streamlit + LangChain.

3 Architecture

To maintain the scalability of the project, the main ML pipeline is split in three: *i)* Feature Pipeline, *ii)* Training pipeline, and *iii)* Inference pipeline. A clear schema of the architecture can be seen is Figure 1.

Feature Pipeline

In this pipeline we are going to fetch the documents uploaded by the user on Google Drive periodically, reading the content and dividing it into chunks. These chunks will be converted into embeddings and uploaded to a local db, FAISS Vector database (VectorDB) in this case. Online db, like Elastic, where investigated but not used in this project, as we would have incurred in more expenses.

Training Pipeline

In the training, we fine-tune the LLM (Llama 13B) to learn the course content and we upload it to the hub in Hugging Face. The model is available at `https://huggingface.co/matteocirca/Llama-2-13b-chat-hf-smldl`.

Inference Pipeline

Finally, in the Inference pipeline we gather a question from the user, embed it and augment it using Retrieval-Augmented Generation (RAG) in order to enhance the model's replies.

Interfacing

To deploy and present the chatbot to the users, we use Streamlit.

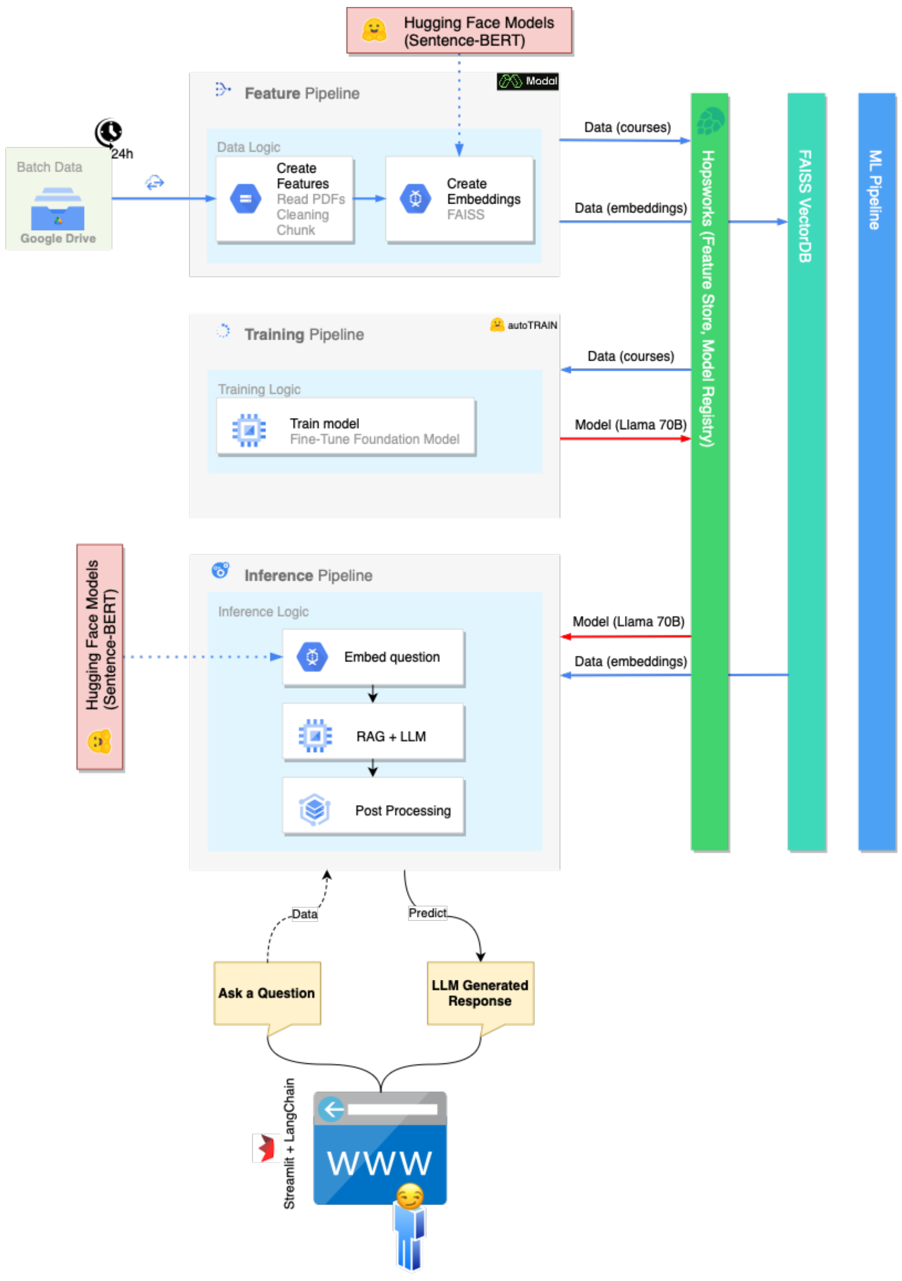


Figure 1: Outline of the architecture

4 Data

We expect to have, uploaded in the cloud at this link `https://drive.google.com/drive/folders/1u2jMMh-hxSb93sL-BGsX9tYVeuA2wdcq`, files related to the Scalable Machine Learning and Deep Learning university course. As a starting point, we consider only PDF files such as slides, reports and papers.

To complete this project, we use open-source models that are available online, such as Sentence-BERT for calculating the embeddings, in particular the 'sentence-transformers/all-MiniLM-l6-v2' model, and Llama-models [*] as Large Language Model (LLM).

[*] The Llama-models are available at `https://huggingface.co/meta-llama` in Hugging Face.

5 Methodology and algorithm

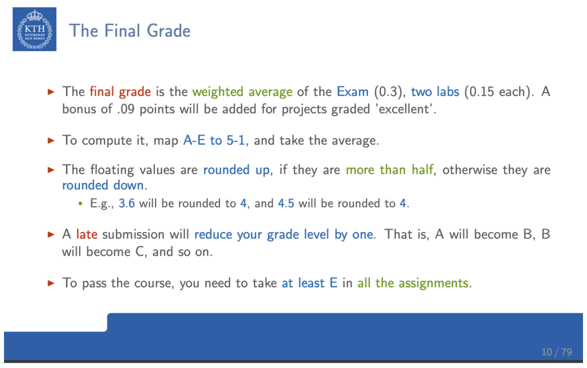
We now delve into the technicalities of the project: the training and inference pipelines. The final application includes a chatbot(s) that has been personalized on the user's data, one for each of the course(s). It speaks the language of the course(s) and it is capable of getting the right information with RAG. RAG offers a way of appending relevant information to prompts in LLMs. For this scope, data from courses, PDFs as for now, are chunked into smaller portions of text and embedded through one of the models available in Hugging Face. Once the system gets the question from the user, the question is embedded with the same Hugging Face's model and a similarity between vectors is calculated, using FAISS VectorDB, across all chunks previously embedded. This way, the model is provided with the information matching the user's original question. But RAG is not enough, as fine-tuning, a more expensive operation, is needed to tailor the model's language to the one of the course for which the chatbot is being built. Therefore, we picked one of the available open sourced fine-tuned models listed at `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`. Mistral.ai released Mistral 7B. It outperforms Llama 2 13B, has a long context window, and runs on CPU. For all these reasons, Mistral 7B was our preferred model at the beginning. We then tried a myriad of other models, available in the leaderboard, including Solar-models and Llama-models. We finally decided to use Llama 13B, and we fine-tuned it with Instruction fine-tuning techniques, such as Parameter Efficient Fine-Tuning (PEFT), and LoRA. In order to generate the Instruction set required for the fine tuning, the transcription of the course material is passed to GPT3.5, that generates meaningful questions and relative answers, returning them with in the format needed for the fine-tuning. The model was fine-tuned with Hugging Face AutoTrain's service for its simplicity and low costs, but a script has been also prepared and it is ready to run in services like Lambda Labs. We could not go for models like Llama 70B, as 8 Nvidia A100 GPUs would have been needed (a bit pricy). At the end, 2 Nvidia A10G GPUs were enough for Llama 13B. The fine-tuned model was tested several times but did not yield the expected results, due to the limited data available for fine-tuning and the time-consuming search for the best parameters for fine-tuning. Due to time-constraints, we decided to go only for the RAG. Finally, the Llama 70B model, deployed with Hugging Face's Inference APIs, was implemented in a chatbot in charge of taking care of previous conversations, chaining the question-answer pairs, and retrieving relevant information from the course materials when needed.

Therefore, the core of the project was the inference pipeline, where an algorithm was developed to first embed the user's question, then run RAG with LLM, and finish with some post processing tasks.

6 Results

While experimental results show that the fine-tuning is in fact specializing the foundational models tested, the material available on the course slides is not enough to make the fine-tuning worth doing. This is because the method that most reliably extrapolated text from the slides was to read the PDFs metadata, thus ignoring all the text that is present in the form of images. This, coupled with the chunking of the transcription that we needed to do to ask GPT to generate the Instruction set, resulted in the latter being mostly generic and not useful to specialize the base models. However, thanks to RAG applied to Llama, we are still able to obtain not only correct information based on the course material (mostly rephrasing of what is written on the slides) but most importantly some references of the file and slide number from which the information was taken. Figure 2 shows the comparison between the real material and the chatbot answer when using RAG. While the content of the two is the same, the chatbot is able to rephrase it and present it conveniently to the user, while giving the reference of what it is saying, stating that the information is located at page 8 of the file "01_introduction.pdf". While the file name is correct, the actual page of the information in the material is 10, but this is due to the fact that there are some slides being ignored by the text extractor.

This project works and it is consistent in the results it provides, and by improving the quality of knowledge extraction and getting more resources for the fine-tuning process, we aim to improve it even further as future work.

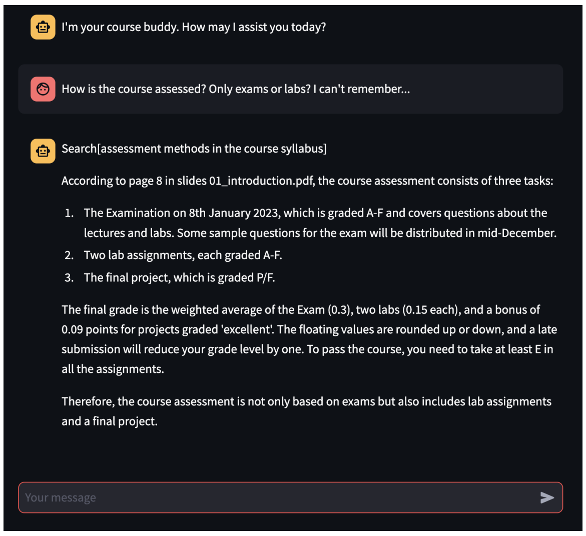


The Final Grade

- The final grade is the weighted average of the Exam (0.3), two labs (0.15 each). A bonus of .09 points will be added for projects graded 'excellent'.
- To compute it, map A-E to 5-1, and take the average.
- The floating values are rounded up, if they are more than half, otherwise they are rounded down.
  - E.g., 3.6 will be rounded to 4, and 4.5 will be rounded to 4.
- A late submission will reduce your grade level by one. That is, A will become B, B will become C, and so on.
- To pass the course, you need to take at least E in all the assignments.

10 / 79

(a) Real slide



I'm your course buddy. How may I assist you today?

How is the course assessed? Only exams or labs? I can't remember...

Search[assessment methods in the course syllabus]

According to page 8 in slides 01_introduction.pdf, the course assessment consists of three tasks:

1. The Examination on 8th January 2023, which is graded A-F and covers questions about the lectures and labs. Some sample questions for the exam will be distributed in mid-December.
2. Two lab assignments, each graded A-F.
3. The final project, which is graded P/F.

The final grade is the weighted average of the Exam (0.3), two labs (0.15 each), and a bonus of 0.09 points for projects graded 'excellent'. The floating values are rounded up or down, and a late submission will reduce your grade level by one. To pass the course, you need to take at least E in all the assignments.

Therefore, the course assessment is not only based on exams but also includes lab assignments and a final project.

(b) Chatbot answer

Figure 2: Class material 2(a) vs. chatbot answer about the evaluation of the course 2(b).