جامعة عبد المالك السعدي
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵄⴰⴱⴷⵍⵎⴰⵍⵉⴽ ⵙⵙⴰⵄⴷⵉ
Université Abdelmalek Essaâdi

كلية العلوم و التقنيات بطنجة
ⵜⴰⵖⵣⵍⵓⵜ ⵏ ⵜⵜⵓⵙⵙⵏⵉ ⴰ ⵜⵓⵣⵔⵜⵉ ⵏ ⵟⴰⵏⵊⴰ
Faculté des Sciences et Techniques de Tanger

A RAG AND FINE-TUNED LLM ON CUSTOM CONTEXT: A CASE STUDY AT FSTT

MST: Artificial Intelligence and Data Science

Prepared by:
ABDELMAJID BENJELLOUN
Ahmed SAMADY
Mohammed Amine Fakhre-Eddine
Chibani FAHD
Supervised by:
Pr. Lotfi EL AACHAK

Table of contents

1 Introduction

Artificial intelligence has shown extremely rapid development over the past ten years, transforming most industries. The most striking and impactful application is natural language processing, which allows machines to process, understand, and generate human language, thus enabling a more natural and seamless interaction between humans and computers. One of the most significant recent developments in this domain is the creation of so-called chatbots, which are capable of deep comprehension and generation of human-like text in their conversations. Such a development enables the wider adoption of chatbots in applications like customer service and technical support, as well as applications for personal use, like personal assistants and educational tools.

This project will focus on the design and development of a sophisticated chatbot build on a fine-tuned large language model (Llama3) on a specific context related to the Faculty of sciences and technologies of Tangier, as well as using RAG methods to extend its knowledge, with a large dataset of courses and related stuff. Fine-tuning the chatbot to adapt to the FSTT's academic environment and content will deliver a more relevant and appropriate response to the educational environment and better the experience for both students and faculty.

Deployment of chatbots within an educational setting offers numerous benefits. Chatbots can provide instant answers to questions for students, support in administrative tasks, language learning, and personalized tutoring. However, to achieve these benefits, it is essential that the chatbot is well fitted into the specific academic environment within which it will be put to use. This can be done by fine-tuning the chatbot on a specialized corpus that reflects the linguistic and thematic nuances of the educational materials and activities at FSTT.

2 Objectives

This project tries to create an intelligent and contextually aware chatbot using RAG and fine-tuning. The chatbot will be fine-tuned using a specialised French corpus and general information data collected from the faculty's website.

RAG, on the other hand, is a hybrid approach that combines retrieval- and generation-based techniques to produce more precise and contextually appropriate results. LangChain is a framework for seamlessly integrating and managing numerous language models when developing complicated NLP applications. Vector databases are utilised for efficient big data storage and retrieval, which is essential for managing massive amounts of training and operational data.

By using these technologies, the chatbot will respond accurately and contextually appropriately to a wide spectrum of requests regarding the academic environment at FSTT. It will considerably improve the end-user experience, making the chatbot a very useful tool, especially for new students and faculty members.

3 Motivations

Fine-tuning becomes one of the project's most important approaches since it is the act of deeply adapting pre-trained language models to the specifics of specialised domains or jobs. Unlike generic language models, fine-tuned models are modified in such a way that they are almost identical to the target domain's linguistic intricacies, terminologies, and contextual unique characteristics. In the context of Tangier's Faculty of Sciences and Techniques (FSTT), where such academic discourse encompasses a wide range of subjects and disciplines, fine-tuning is essential.

Due to an intense fine-tuning process of language models against this custom corpus harvested from the FSTT academic material, the chatbot develops exceptional proficiency in understanding and generating the kind of text demanded in the academic landscape of the specific institution. By undergoing this kind of granular customization, the chatbot is endowed with the ability not only to understand the underlying semantics of queries but also to respond in a manner that is more precise and relevant than off-the-shelf solutions can afford. This guarantees that the students, faculty, and staff using the chatbot will have a frictionless, engaging exchange with the chatbot, allowing them to proceed with a nuanced understanding and contextually relevant response.

First, fine-tuning facilitates a symbiotic relationship between the language model and the domain experts involved in the project. Through a loop of iterative refinement, FSTT stakeholders can actively shape and enhance the chatbot's capabilities, ensuring that it remains attuned to evolving academic needs and preferences. This collective enterprise not only enhances the chatbot's effectiveness but also inculcates a sense of ownership and investment among the FSTT community, engendering trust and engagement with the AI-driven tool.

In essence, then, fine-tuning becomes a strategic enabler for the chatbot project, allowing it to transcend the limitations of generic language processing solutions and become a truly bespoke, contextually aware assistant, attuned specifically to the academic domain of FSTT.

4 Data Collection

The data collection process is a crucial step in this project, forming the foundation for fine-tuning the chatbot. This involves gathering relevant academic materials and information from two primary sources: the website of the Faculty of Sciences and Techniques of Tangier (FSTT) and various PDF documents provided by the professors. The data will be processed and organized to facilitate the fine-tuning of the language models and the implementation of Retrieval-Augmented Generation (RAG).

Data from the Website To gather data from the FSTT website, we utilized web scraping techniques, specifically leveraging Beautiful Soup, a Python library for parsing HTML and XML documents. The primary goal was to extract information from various sections of the website, including:

- DEUST, Licence, Master, Engineer Cycle: Information regarding different academic programs offered at FSTT. News: Updates and announcements related to academic and extracurricular activities.
- Clubs: Details about student clubs and organizations within the faculty.
- Coordinators: Information about academic coordinators and their roles.
- Departments: Details of the various academic departments within FSTT.
- Important Links: Key links to resources and external websites relevant to the students and faculty. Information about the Faculty: General information about FSTT, including its history, mission, and vision.

The process of scraping the website involved the following steps:

1. Accessing the Website: We accessed the FSTT website at https://fstt.ac.ma/Portail2023/.
2. Parsing HTML Content: Using Beautiful Soup, we parsed the HTML content of the website pages.
3. Extracting Relevant Data: Specific sections of the HTML were targeted to extract the required information.
4. Structuring the Data: The extracted data was then organized into a structured format, primarily JSON.
5. Generated from this latter another JSON file that comprises two main components: instructions (questions or prompts) and responses (answers or information related to the prompts). The JSON structure allows for efficient storage and retrieval of data, making it easier to use for training and fine-tuning the language models.
6. Data from PDFs: In addition to web data, we collected PDF documents from professors at FSTT. These documents include:
7. Course Materials: Lecture notes, syllabi, and other teaching materials.
8. Practical Works (TP): Instructions and guidelines for practical sessions.

9. Tutorials (TD): Problem sets and exercises for tutorial sessions.

10. Projects: Documentation and guidelines for academic projects.

5 Retrieval-Augmented Generation (RAG)

To enhance the chatbot's ability to provide accurate and contextually relevant responses, we employed Retrieval-Augmented Generation (RAG). RAG combines the strengths of retrieval-based and generation-based approaches, enabling the chatbot to fetch relevant information from a large corpus and generate coherent and contextually appropriate responses.

5.1 Populating the database

Our Corpus contains nearly everything related to the DEUST MIPC, Licence Génie Informatique, Master IASD (S1 and S2), including courses, exercises, projects and many more, the total size of data is nearly 1.01GB of PDFs, MS Word, and MS PowerPoint documents. In order to create embeddings for this large dataset we used a model called "OrdalieTech/Solon-embeddings-large-0.1" from HuggingFace which ranks 4th on the MTEB leaderboard for french language retrieval, this choice was made because first, text embeddings play a huge role in retrieving relevant context to the query, and second because the other three models are paid or they require some kind of paid API Access. there are other parameters which we should mention such as chunk size which we set equal to 800, this parameter controls the size of the chunk taken from a document, and chunk overlap is set to zero, to make sure that our chunks don't overlap since we need accurate embeddings. After generating said Embeddings they are stored in a ChromaDB database fro later retrieval tasks.

5.2 Retrieval Mechanism and Inference

When a query is received, the vector store retrieves the most relevant documents or text snippets from the indexed corpus, after much experimentation we've concluded that a good number for top-K documents is 7, since lower that 7 is too little context for large document, and choosing a bigger number make the LLM confused and not sure which information in context to use for inference. The retrieved context is then passed to Llama3-8B for inference and generating response.

6 Fine-Tuning

Fine-tuning involved adapting pre-trained language models to understand and generate text specific to the academic context of FSTT. This process included several critical steps to ensure the chatbot's effectiveness and relevance:

1. **Scraping Data**: This step involved scraping data from the faculty web site to have all the general information needed for the LLM to learn.

2. **Formatting Data**: Once we had our desired data, we ended up with a heavily nested JSON file which an LLM can't use for its learning, so we had to format it into

multiple formats including plane text, markdown, and a JSON file of of instructions and responses, this simplifies our data format for rfurther use by the LLM.

3. **Model Selection**: Choosing a model depends on multiple factors, including our needs, the computational resources available, and the desired performance. We selected the Llama 3 8B instruct model for its small size compared to the 70b model, and its great performance in the academic and educational context.

4. **Preparing data for training**: To train the model on our instructions and responses, we had to format each row into a text with a system prompt ate the top, we also added the raw markdown text to the dataset to help the model understand the context of the text.

5. **Training**: The training was done using the unsloth library, which automates the training process, makes it easier to fine-tune the model, and provides an even lighter and faster version of our chosen model. Due to the small size of our dataset and the limited computational resources, we opted for the Low-Rank Adaptation (LoRA) method, which is a more efficient way to fine-tune the model, reducing the number of parameters to be trained and memory usage.

Through this fine-tuning process, the chatbot developed a decent understanding of the academic content specific to FSTT, enabling it to provide accurate and contextually appropriate responses to users' queries.

7 Tools and Technologies

In order to create a smart chatbot for the Faculty of Sciences and Techniques of Tangier (FSTT), we used a wide range of tools and technologies in this project. From data collection and model optimisation to deployment and user engagement, every tool was essential to the project at different points.

- **Hugging Face**: Hugging Face provided a comprehensive library of pre-trained models and tools for natural language processing.

- **PyTorch**: PyTorch was the deep learning framework used for training and fine-tuning the models.

- **Llama 3 8B:Instruct**: An instruct fine-tuned version of the 8B model that is optimized for specific tasks. For instance, it can be used to create educational tools that explain complex subjects.. It provided:

  - High Accuracy: Leveraging its large parameter size for nuanced understanding.
  - Contextual Relevance: Generating contextually appropriate responses for academic queries.

- **Kaggle and Google Colab**: Kaggle and Google Colab were utilized for their GPU resources to speed up the training and fine-tuning processes. They provided:

  - GPU Access: Leveraging powerful GPUs for faster computation.

  - Collaborative Environment: Facilitating code sharing and collaboration.

- **Unsloth**: Unsloth simplifies the training process while providing a light weight model to be used.

- **Langserve**: Langserve is a language model serving tool that was used to deploy the fine-tuned models efficiently. It helped in:

  - Serving Models: Providing an interface for querying the fine-tuned models.
  - Scaling: Managing multiple instances to handle concurrent requests.

- **Ollama**: Ollama was used to run our LLMs locally.

- **SvelteKit**: SvelteKit was used for developing the front-end interface of the chatbot. It offered:

  - Interactive UI: Creating a responsive and user-friendly interface.
  - Single Page Application: Ensuring smooth navigation and fast load times.

- **Chroma DB (Vector Database)**: Chroma DB served as the vector database for storing and retrieving large volumes of academic text. It facilitated:

  - Efficient Indexing: Storing text in a format suitable for fast retrieval.
  - Similarity Searches: Enabling the retrieval of contextually relevant information.

- **Docker**: Docker was used to containerize the application, ensuring consistent environments across different stages of development, testing, and deployment. It facilitated:

  - Environment Consistency: Ensuring that the application runs identically on different machines.
  - Scalability: Simplifying the scaling process for handling more users.

8 User interface

The user interface is implemented with sveltekit, its similar to any chatbot interface, with a text input field and a send button. In addition to conversation and history persistence, the user can hop between the fine-tuned model and the RAG chain model.

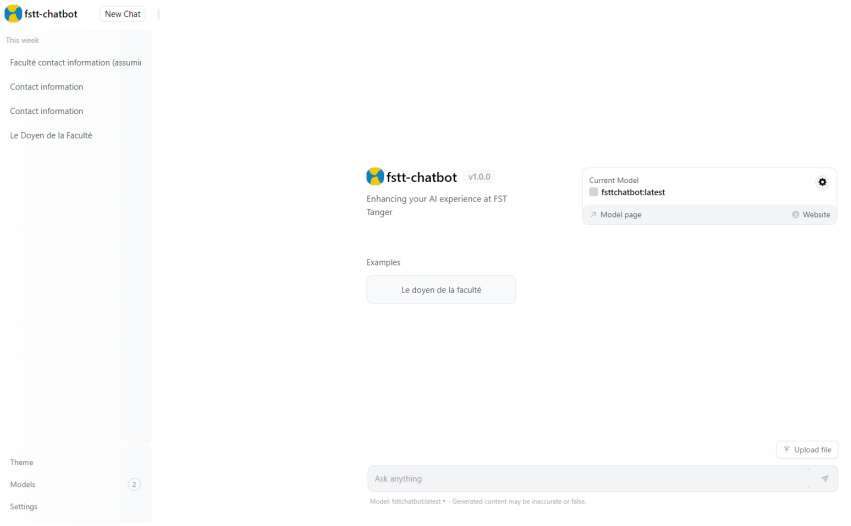

Figure 1: Default chat interface

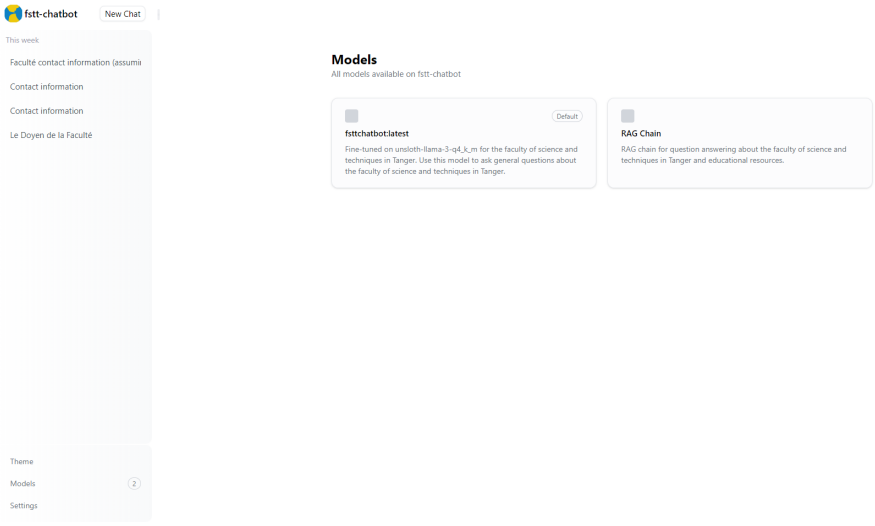

Figure 2: Choosing the model

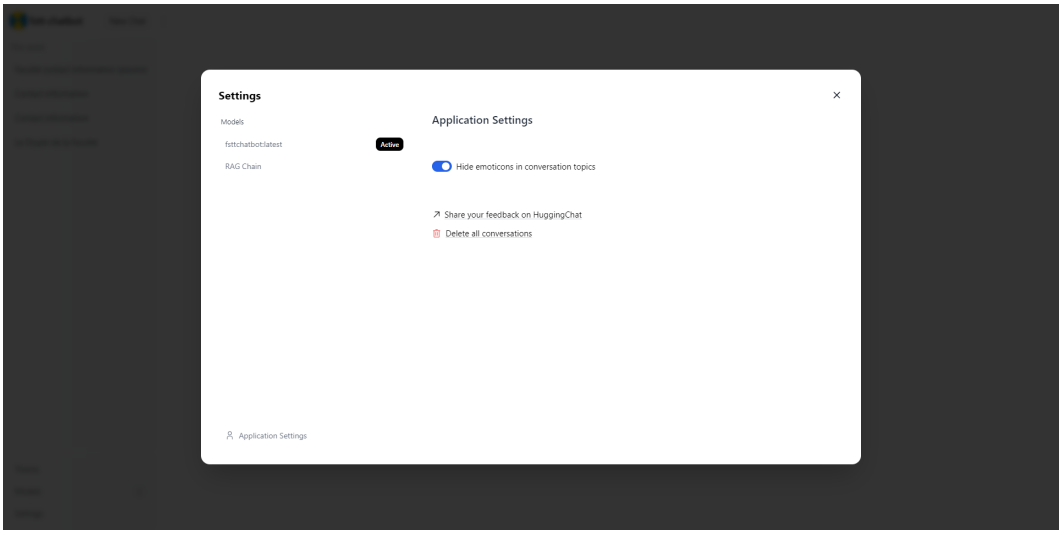

Figure 3: Application settings

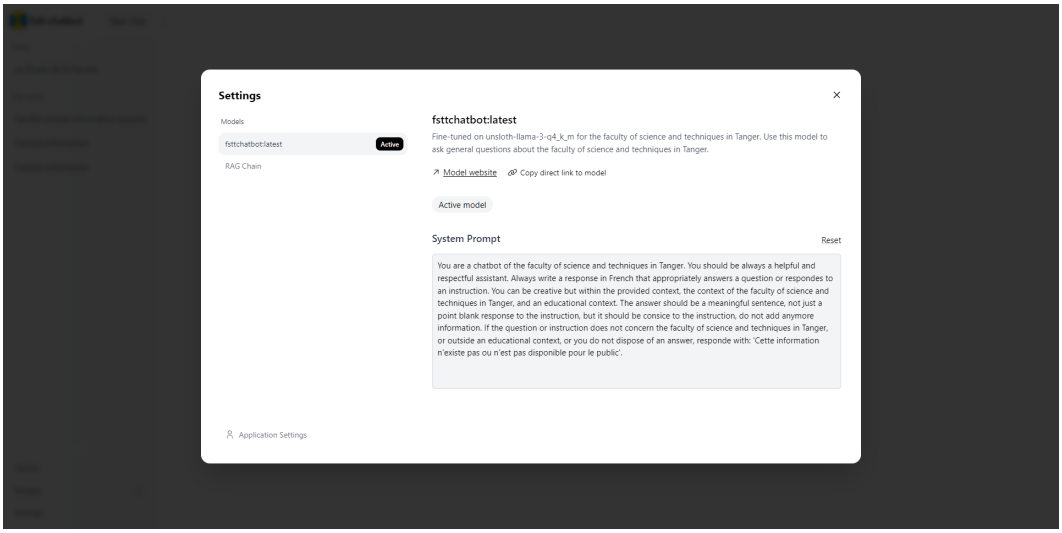

Figure 4: Model settings

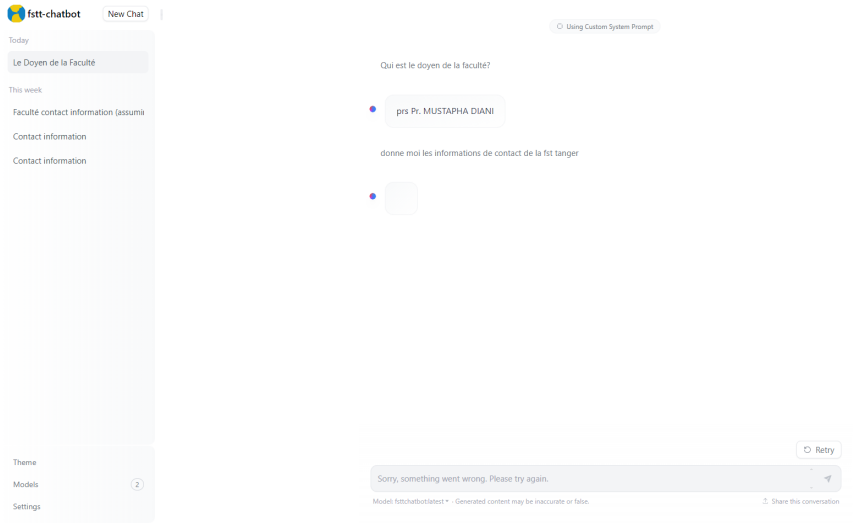

Figure 5: Conversation example

9 API

Our API is developed using Langserve, which is based on FastAPI. It wrappes a Langchain chain that gets embeddings from the chromaDB collection based on similarity to the query, then it feeds the context to a Llama 3 model which generates a response based on that context.

10 Architecture

10.1 Containerization

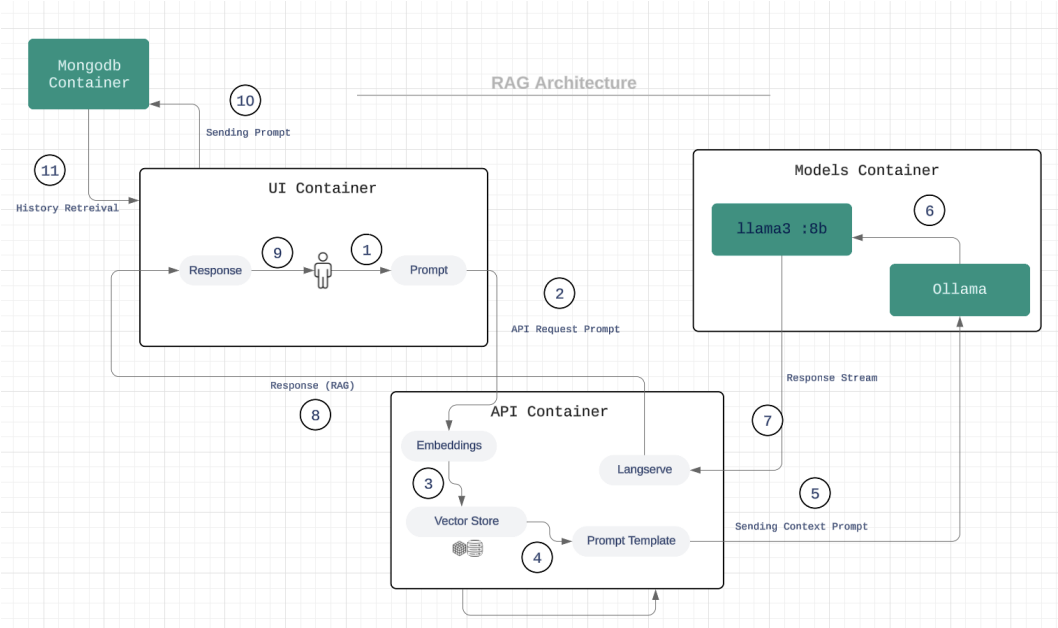

Figure 6: The RAG Architecture

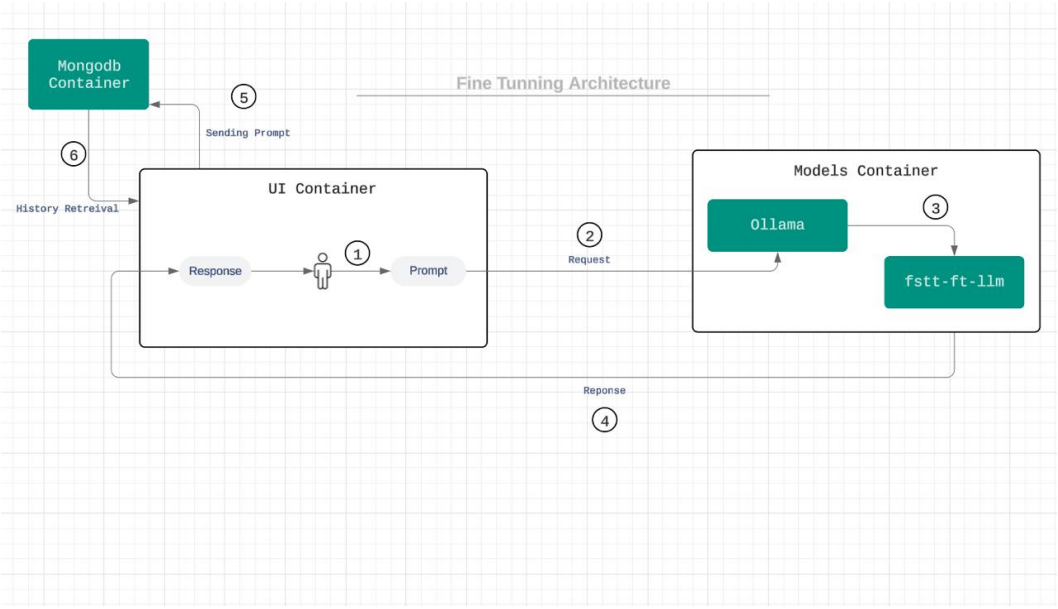

Figure 7: The fine-tuning Architecture

The three Docker containers that make up the architecture's system are essential to the chatbot's operation:

1. **User Interface (UI) Container** This container serves as the entry point for user interaction. Users initiate the process by entering their query (prompt) through this interface.

2. **API Container:** Upon receiving the user's prompt from the UI container, the API container initiates an API request. This request encapsulates the user's prompt and facilitates its transmission to the subsequent stages within the architecture.

    - **Embedding and Vector Store**
    - **LangServe:** The LangServe component serves as the interface for the frontend utilizing the concept of chains implemented in Langchain.

3. **Model Container:** This container houses the core computational power of the RAG system.

    - **Ollama:** This component plays a vital role in managing and optimizing the language model (Llama3) utilized within the system.
    - **Llama3**
    - **fsttchatbot:** The core of the fine-tuning architecture is represented by this part, a fine-tuned Llama 3 8B instruct model on general information concerning the FSTT.

This containerization approach ensures that the application can be run on any machine that has docker installed. Moreover, adding kubernetes can make the application scalable and fault tolerant.

10.2 Storage

We use a MongoDB database to store app specific data, such conversations and history, this is done on the sveltekit level. We could use amongoDB container for this purpose, but we settled for a local instance due compuational constraints.

11 Conclusion

In conclusion, the creation of this chatbot involved the utilization of two main techniques: retrieval-augmented generation (RAG) and fine-tuning. The RAG technique is employed to extract information from PDF files, such as course materials or descriptions. This is achieved by using an embedding model, referred to as "OrdalieTech/Solon-embeddings-large-0.1", which transforms the extracted information into a format that can be processed. When a user poses a question, the context is derived from these embeddings and fed into the Lama3 model, which has 8 billion parameters, to generate a response.

For the second part of the process, the LoRA method is utilized to fine-tune the Llama3 8B instruct model. This allows for the model's weights to be frozen, meaning they remain unchanged, while the model is trained on data that has been scraped from the faculty's resources. By doing so, the model can be customized to better suit the specific data and likely queries it will encounter.

In the final stage, both models are integrated into a single interface. This provides users with the flexibility to choose between the two models based on their preferences or needs. Such an interface enhances the user experience by offering a tailored interaction, whether they require detailed information retrieval or more nuanced conversational engagement. The end result is a sophisticated chatbot capable of handling a variety of tasks, from answering straightforward questions to providing in-depth explanations, all while maintaining a user-friendly approach.

References

[1] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, and B. Bossan, "Peft: State-of-the-art parameter-efficient fine-tuning methods," https://github.com/huggingface/peft, 2022.

[2] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[3] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[4] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021.

[5] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2017.

[6] P. QoChuk, Benjamin, "Basics of quantization in machine learning (ml) for beginners." [Online]. Available: https://iq.opengenus.org/basics-of-quantization-in-ml/

[7] M. A. Fakhre-Eddine, "al0new0lf/unsloth-llama-3-q4_k_m · hugging face," 2024. [Online]. Available: https://huggingface.co/aL0NEW0LF/unsloth-llama-3-q4_k_m

[8] ——, "al0new0lf/unsloth-llama-3-lora · hugging face," 2024. [Online]. Available: https://huggingface.co/aL0NEW0LF/unsloth-llama-3-lora