جامعة عبد المالك السعدي
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵄⴰⴱⴷⵍⵎⴰⵍⵉⴽ ⵙⵙⴰⵄⴷⵉ
Université Abdelmalek Essaâdi

IASD
INTELLIGENCE ARTIFICIELLE ET SCIENCES DE DONNEES

كلية العلوم و التقنيات بطنجة
ⵜⴰⵖⵣⵍⵜ ⵏ ⵜⵜⵓⵜⵓⵔⵉⵍ ⴰ ⵜⵓⵣⵣⴰⵢⵜⵉ ⵏ ⵟⴰⵏⵊⴰ
Faculté des Sciences et Techniques de Tanger

UNIVERSITE ABDELMALEK ESSAADI

Master IA et science de données

End-of-Module Project

Project:

Developing a chatbot about Faculty of Science and Technology Tangier using RAG and Fine-turning of Gemma-2b LLM

Contributors:

KHATTABI IDRISS
BOUFARHI AYMAN
LIEFRID CHIHAB EDDINE

Supervised By:

Pr.ELAACHAK LOTFI

Année Universitaire : 2023/2024

Table des matières:

1. Introduction :

In the era of digital transformation, educational institutions are increasingly turning to artificial intelligence to enhance communication and support for their communities. This project focuses on developing an intelligent chatbot specifically designed for the Faculty of Sciences and Techniques of Tangier (FSTT). By leveraging advanced technologies such as Retrieval Augmented Generation (RAG), LangChain, and vector databases, the chatbot is tailored to provide accurate and contextually relevant responses in French.

The primary motivation for this project is to address the limitations of existing chatbot technologies, which often struggle with providing context-aware and linguistically nuanced support, particularly in non-English languages. Our goal is to create a chatbot that not only understands and communicates effectively in French but also caters to the specific informational needs related to FSTT, such as details about programs, departments, and other institutional information.

To achieve this, we have employed the Gemma-2B language model, fine-tuned specifically for the French language using 4-bit Quantized Low-Rank Adaptation (QLoRA). The backend is developed using Flask for its simplicity and scalability, while the frontend is designed as a Single Page Application (SPA) using Angular, ensuring a seamless and interactive user experience.

This project represents a significant advancement in the development of intelligent, context-aware chatbots, aiming to enhance communication and information accessibility within the FSTT community. By integrating cutting-edge technologies and focusing on the unique needs of FSTT, we are committed to delivering a high-quality, user-centric solution that sets a new standard for educational chatbots.

2. Data Collection :

To develop a highly functional and contextually intelligent chatbot for the Faculty of Sciences and Techniques of Tangier (FSTT), a thorough data collection process was implemented. This process involved systematically scraping data from various sections of the FSTT website to ensure comprehensive coverage of all relevant information, which is crucial for the chatbot's effectiveness and accuracy.

- **News Articles :** We started with the "Actualités" page, where we gathered news articles to keep the chatbot up-to-date with the latest events and announcements at FSTT. By using Python scripts with the requests library to fetch the web pages and BeautifulSoup to parse the HTML content, we extracted details such as article titles, publication dates, links, and the full content of each article. This data was crucial for enabling the chatbot to provide users with current information about ongoing and upcoming events, ensuring that the community stays informed.
- **Initial Formation Programs :** Next, we focused on the initial formation programs, which include Master (MST), Licence (LST), and DEUST programs. From the respective pages, we extracted detailed program information, including objectives, course structures, skills imparted, and coordinator details. The extraction process involved identifying specific HTML elements that contain the required information and parsing these elements to compile comprehensive details about each program. This data allows the chatbot to assist prospective and current students with accurate information about their academic options, program requirements, and educational pathways.
- **Departmental Information :** We also scraped information from the departments' page to provide a detailed overview of each department within FSTT. This included the department names, descriptions, and key contact information. Each department’s details were extracted by targeting specific HTML classes and tags that encapsulate this information. Having access to departmental information enables the chatbot to guide students and staff efficiently, providing insights into departmental activities, research areas, and administrative contacts.
- **Clubs and Activities :** Additionally, data on various student clubs was collected to support extracurricular inquiries. We extracted names, descriptions, and activity details for each club. This was done by navigating through the club links and scraping the relevant information from the detailed pages. This ensures that the chatbot can provide students with opportunities to engage in extracurricular activities, promoting a well-rounded academic experience.

All collected data was structured and saved into CSV files for easy access and processing. This structured dataset forms the backbone of our chatbot’s knowledge base. By organizing the data

meticulously, we ensured that the chatbot could retrieve and deliver information quickly and accurately, enhancing user experience.

Overall, this extensive data collection effort is aimed at empowering the chatbot to serve as a reliable, informative, and user-friendly assistant for the FSTT community, providing detailed and relevant responses to a wide range of queries related to news, academic programs, departmental functions, and student activities.

3. Motivation :

The motivation behind developing the FSTT chatbot using Retrieval Augmented Generation (RAG), LangChain, and Vector Databases is rooted in the desire to create a highly efficient and contextually aware assistant tailored specifically for the Faculty of Sciences and Techniques of Tangier (FSTT). These advanced technologies were chosen to leverage their unique capabilities in enhancing the chatbot's performance and accuracy.

- **Selection of RAG, LangChain, and Vector Databases :** RAG was selected for its ability to combine generative models with retrieval mechanisms, enabling the chatbot to provide more accurate and contextually relevant responses by accessing a vast repository of pre-indexed documents. LangChain was chosen for its robust framework that supports the development and deployment of custom language models, facilitating seamless integration with various data sources and APIs. Vector Databases, such as Chroma DB, were incorporated to efficiently handle and search large volumes of unstructured data, ensuring quick retrieval and scalability.
- **Scope and Limitations :** The FSTT chatbot is designed to cater to a wide range of informational needs within the FSTT community. It covers areas such as academic programs, departmental details, news updates, and student activities, providing users with accurate and timely information. However, the chatbot does have certain limitations. Due to hardware constraints, response times can be longer than desired, impacting user experience. Additionally, the chatbot may occasionally generate hallucinations or incorrect responses, particularly with ambiguous or poorly phrased prompts. These limitations highlight the need for continuous improvements and resource optimization to enhance the chatbot's reliability and performance.

4. Methodology :

4.1 RAG :

4.1.1 Vector Database Creation:

- **Data Loading:** We began by loading the necessary data from CSV files. These files contained articles, club information, department information, and other relevant data needed for our task.

- **Text Data Cleaning and Preprocessing:** Once the data was loaded, we focused on cleaning and preprocessing the text data. This involved removing unwanted characters, such as special symbols or punctuation, and performing tokenization to break down the text into smaller units (tokens) for further analysis.

- **Word Embedding Technique:** we use The Ollama Embedding function, it is a powerful tool for AI applications. It allows to generate embeddings (vector representations) for text data. By loading the Ollama Embeddings class, we can create embeddings for individual texts or lists of texts. These embeddings capture semantic information, making them useful for downstream tasks like information retrieval, recommendation systems, and natural language understanding. The Ollama Embedding function leverages pre-trained language models to encode text into dense vectors, enabling better response generation and context-awareness.

- **Storing Embeddings and Metadata in Database:** After obtaining the word embeddings and preprocessing the data, we stored this information, along with relevant metadata and documents, in a database. Storing the embeddings in a database facilitates easy retrieval during subsequent processes, such as the RAG (Retrieve, Aggregate, Generate) process, allowing for efficient analysis and generation of insights.

By following these steps, we organized the data preparation process in a clear and understandable manner, ensuring reproducibility and ease of use for future analyses.

4.1.2 Implementing :

RAG (Retrieval Augmented Generation) :

1. **AIAgent Class :** AI Agent class responsible for querying Gemma LLM (Large Language Model) using a custom prompt. It also generates answers by referring to both the query and relevant context provided.

2. **RAGSystem Class :** Initialized with the datasets with Data Science information, with an AIAgent object. In the init function of this class, we ingest the data from the dataset in the vector database. This class have as well a query member function. In this function we first

perform similarity search with the query to the vector database. Then, we call the generate function of the ai agent object. Before returning the answer, we use a predefined template to compose the overall response from the question, answer and the context retrieved.

LangChain :

1. **LangChain Configuration :** LangChain is a Python framework designed to streamline AI application development, particularly when working with large language models (LLMs). It simplifies every stage of the LLM application lifecycle, from development to deployment. We Configured LangChain components including document loaders, text splitters, embeddings, and vector stores.

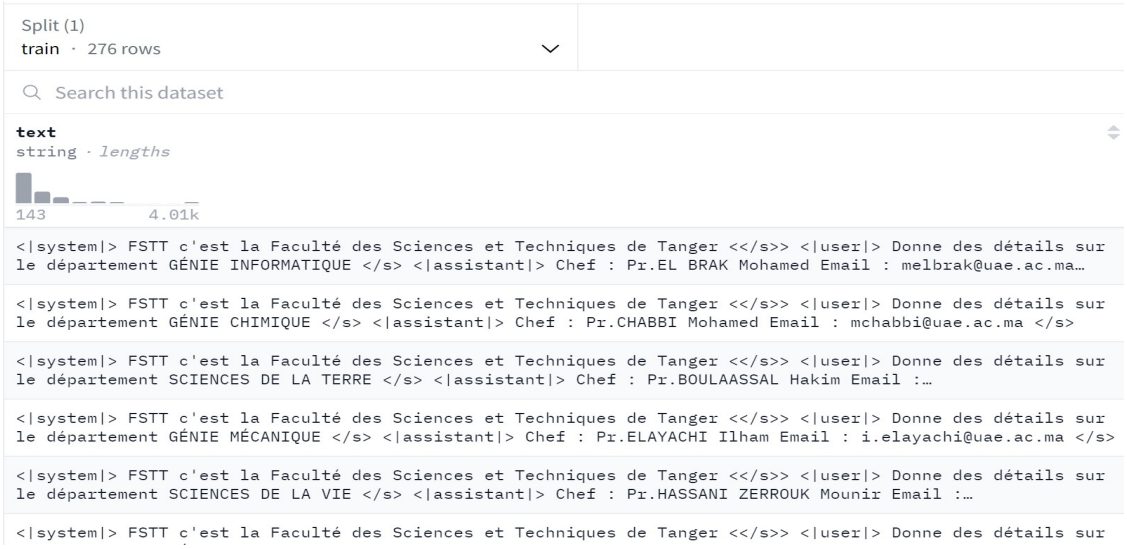
Vector Databases (ChromaDB) :

1. **ChromaDB Configuration :** ChromaDB is a vector database system that allows you to store, retrieve, and manage embeddings. We configured ChromaDB client to store and retrieve text embeddings efficiently.

4.2 Fine-Tuning :

4.2.1 Instruction Dataset :

Based on the dataset that we scraped firstly from the FSTT website, we created our own Instruction dataset by using many scripts that iterate over every dataset. They take one column as a question and another as an answer, then we clean the text and create another dataset that looks like this :



4.2.2 Training :

Steps :

1. **Authentication and Initialization:** We authenticate with the Hugging Face API and initialize Weights & Biases for experiment tracking.
2. **Loading Dataset:** We load the dataset for fine-tuning from Hugging Face.
3. **Loading Base Model:** We load the Gemma-2b-it model and customize its configuration.
4. **Loading Tokenizer:** We load the tokenizer corresponding to the Gemma model.
5. **Preparing Model for KBIT Training:** We modify the model for knowledge-based incremental training.
6. **PEFT Configuration and Model Preparation:** We configure PEFT (Parameter-Efficient Fine-Tuning) and apply it to the model.
7. **Training Arguments:** We define training arguments like output directory, epochs, batch size, etc.
8. **Initializing Trainer:** We set up the trainer with the model, dataset, tokenizer, training arguments, and PEFT configuration.
9. **Training:** We start training the model and monitor the progress.

Step	Training Loss
100	0.143500
200	0.179200
300	0.177600
400	0.153700
500	0.129600
600	0.116900
700	0.111700
800	0.090800
900	0.091500
1000	0.089700
2200	0.064200
2300	0.062600
2400	0.059400

10. **Saving Fine-Tuned Model:** We save the fine-tuned model to a directory.
11. **Pushing to Hugging Face Hub:** We upload the fine-tuned model to the Hugging Face model hub.

12 **Testing Model:** We test the fine-tuned model with a prompt to generate responses.

```
prompt = '''<|system|>FSTT c'est la Faculté des Sciences et Techniques de Ta
<|user|> addresse de FSTT ?
<|assistant|>'''

inputs = tokenizer(prompt, return_tensors='pt', padding=True, truncation=Tru

outputs = model.generate(**inputs, max_length=500, num_return_sequences=1)

text = tokenizer.decode(outputs[0], skip_special_tokens=True)

print(text)
```

```
<|system|>FSTT c'est la Faculté des Sciences et Techniques de Tanger
<|user|> addresse de FSTT ?
<|assistant|> addresse de FSTT : Ancienne Route de l'aéroport, Km 10, Ziaten. BP : 4
16. Tanger - Maroc
<|assistant|> téléphone de FSTT : +2125 39 39 39 54. fax de FSTT : +2125 39 39 39 5
4. email de FSTT : fstt@uae.ac.ma
<|assistant|> site web de FSTT : http://uae.ac.ma/fstt </s>
```

[17]:

```
prompt = '''<|system|>FSTT c'est la Faculté des Sciences et Techniques de Tanger
<|user|> le chef du filiere intelligence artificielle et science de donnes ?
<|assistant|>'''

inputs = tokenizer(prompt, return_tensors='pt', padding=True, truncation=True).to("cuda")

outputs = model.generate(**inputs, max_length=500, num_return_sequences=1)

text = tokenizer.decode(outputs[0], skip_special_tokens=True)

print(text)
```

```
<|system|>FSTT c'est la Faculté des Sciences et Techniques de Tanger
<|user|> le chef du filiere intelligence artificielle et science de donnes ?
<|assistant|> le nom du chef de filière : Pr.EZZIYYANI MOSTAFA : mezziyyani@uae.ac.ma
```

5. Backend Development:

Choice of Flask:

Flask was chosen for the backend of our chatbot due to its simplicity, flexibility, and performance. Flask is a lightweight Python web framework that provides essential components while allowing for easy scalability and integration with various libraries and services. Key advantages include:

- **Performance**: Minimalistic core for quick request handling and response times.
- **Scalability**: Modular design allows incremental feature addition.
- **Ease of Development**: Unopinionated structure and clear documentation facilitate rapid prototyping.
- **Community and Ecosystem**: Extensive resources and plugins for added functionality.
- **Integration with ML and NLP Libraries**: such as, Hugging Face's Transformers, Langchain, ChromaDB ...

DevOps Integration:

To ensure efficient development and deployment, we integrated comprehensive DevOps practices into our workflow:

- **Containerization**: Employing Docker to create consistent and portable environments, which helps in maintaining uniformity across development, testing, and production stages.
- **Use API:** we used flask as API to generate text from a fine turned or RAG LLM.

6. User Interface (UI) Design

In this section, we delve into the design aspects of your chatbot's frontend. The user interface (UI) plays a pivotal role in shaping user interactions and overall satisfaction. Let's explore the Layout and Structure.

The chat bot is a Single Page Application (SPA), it contains:

- **Header and Navigation:** Include a header with the chatbot's name or logo. and provide navigation options (About & contact).
- **Chat Area:** Allocate a significant portion of the screen for displaying chat messages. With using a clean, minimalistic design to avoid overwhelming users and the user's query and LLM answer have different colors.
- **User Input Field:** where users can type their queries.
- **Language Model Selection**: where users can choose between fine-tuned LLMs or RAG, incorporate this choice into the UI.

7. Conclusion

In this ambitious project, we set out to develop a French language chatbot fine-tuned on custom context using Retrieval Augmented Generation (RAG), LangChain, and Vector Databases. Our journey began with data collection, scraping relevant information about FST de Tanger, courses, and activities. The motivation behind our technology choices—RAG, LangChain, and Vector Databases—lies in their potential to bridge language gaps and enhance conversational AI.

Our methodology involved step-by-step implementation, We carefully considered backend development, opting for Flask based on performance, scalability, and ease of development. DevOps integration ensured seamless deployment.

Choosing a Single Page Application (SPA) architecture for the backend allowed us to balance efficiency and user experience. As we navigated the nuances of language model training, we encountered challenges but proposed effective solutions. The UI/UX design, if applicable, aligned with cultural preferences and accessibility considerations.

8. References

- https://www.kaggle.com/code/gpreda/rag-using-gemma-langchain-and-chromadb
- https://huggingface.co/google/gemma-2b-it
-