	Template	Code : DO-PFE-01
		Indice de
		Edition : 07/2022

Rapport de Stage Obligatoire d'Eté

Filière : Génie Logiciel
Niveau : 4^{ième} Année

Sujet :

Medical Chatbot Development using a fine-tuned LLM, RAG & NER

Réalisé par : **Fendri Adam**

Entreprise d'accueil :

Pura Solutions

Année Universitaire : 2023/2024

	Template	Code : DO-PFE-01
		Indice de
		Edition : 07/2022

Rapport de Stage Obligatoire d'Eté

Filière : Génie logiciel
Niveau : 4ème Année

Sujet :

Medical Chatbot Development using a fine-tuned LLM, RAG & NER

Réalisé par : **Adam Fendri**

Entreprise d'accueil :

Pura Solution

Responsable à l'entreprise:	***Avis de la commission des stages***
Mme Jawaher Chatbri	

1 Introduction:

The rapid advancement of artificial intelligence (AI) has had a transformative impact on the healthcare industry, particularly through the development of intelligent systems that can assist in patient care and diagnosis. This report documents the work accomplished during my internship at Pura Solutions, focusing on the development of a medical chatbot system. The primary objective was to build an AI-driven assistant capable of understanding and analysing medical inquiries to provide relevant and reliable information to patients.To achieve this, the project involved fine-tuning a large language model (LLM), Gemma2 2B, specifically adapted for medical contexts. The chatbot also integrates a Named Entity Recognition (NER) model based on RoBERTa to extract critical information, such as a patient's age, weight, and height. Furthermore, a Retrieval-Augmented Generation (RAG) system was implemented to enhance the chatbot's ability to provide contextually appropriate answers. Finally, a summarization model, T5, continuously updated the conversation summary, ensuring coherent and concise records of patient interactions. Through these integrated AI components, the chatbot aims to assist in preliminary medical assessments and guide patients effectively.

2 Presentation of the host company

2.1 Company Overview :

PURA Solutions is a pioneering company established in 2021, specializing in the development of advanced AI-driven hardware and software solutions for the biotechnological and medical sectors. Their mission is to revolutionize healthcare through technological innovation, offering tailored products and services that address critical challenges in the medical industry.

Sector of Activities:

PURA Solutions operates at the intersection of biotechnology and medicine, focusing on:

- **Innovative Product Development**: Designing and developing state-of-the-art medical devices and software solutions, including AI-driven innovations and educational medical equipment like the PURA-DefibrillateurEdu.
- **Biotechnological Advancements**: Leveraging scientific research and cutting-edge technology to create solutions that address equipment shortages and enhance healthcare standards.
- **Client-Centric Services**: Providing customized solutions tailored to the unique needs of each client, ensuring adaptability and alignment with the operational requirements of healthcare institutions.

PURA

Driven by innovation

Figure1: Pura Solutions logo

2.2 Organizational Structure:

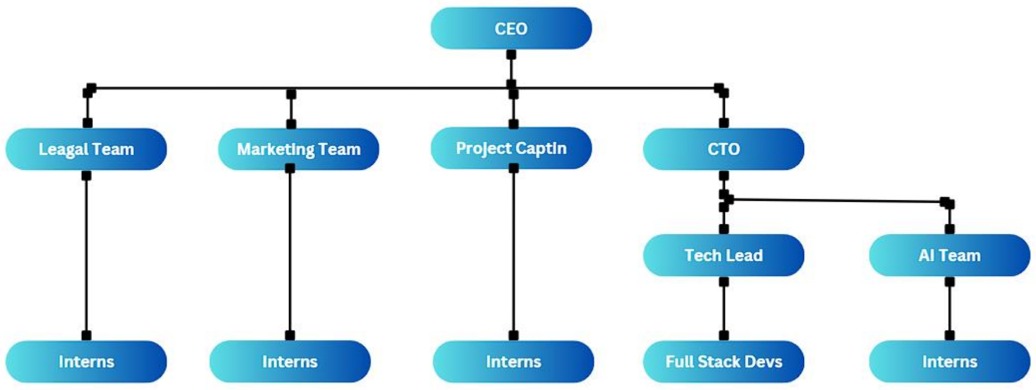


Figure 2: Pura Solutions organizational structure

2.3 Services Offered:

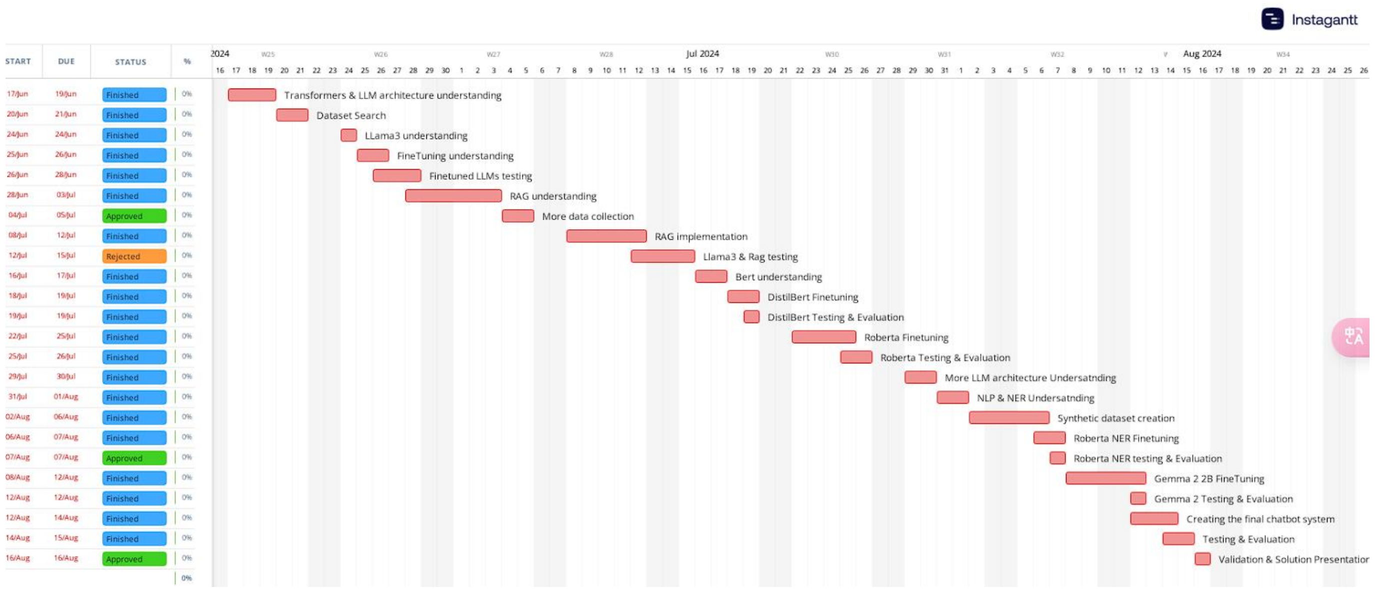
PURA Solutions offers a comprehensive range of services, including:

- **Conception and Development**: Crafting innovative hardware and software solutions tailored to the specific needs of the biotechnological and medical sectors.
- **Guidance and Consulting**: Assisting clients in shaping customized products aligned with their unique requirements.
- **After-Sales Support**: Offering robust after-sales services to guarantee seamless integration and operation of solutions within client environments.
- **Sales Services**: Ensuring a smooth and efficient procurement process for flagship products.
  For more information, visit their website: PURA Solutions[1].

3 Objectives(Project Scope):

The objective of this medical chatbot project is to transform patient analysis and illness guidance through the use of advanced AI technologies. By automating routine tasks traditionally managed by human medical assistants, the chatbot efficiently supports doctors by extracting critical patient details, analysing symptoms, and providing relevant medical insights that can assist in clinical decision-making. One of its key features is conversation summarization, which continuously compiles patient interactions into concise and coherent summaries, allowing healthcare professionals to quickly review cases and focus on patient care without having to sift through extensive dialogue. This functionality not only enhances efficiency but also reduces the risk of overlooking crucial information, ensuring a comprehensive understanding of the patient's condition. The ultimate goal is to integrate this intelligent assistant into a website, making it widely accessible to both patients and healthcare providers, thereby democratizing access to quality medical guidance. As a research and development (R&D) initiative, the project aims to push the boundaries of medical AI, exploring innovative methods to improve healthcare delivery while addressing significant challenges. It is important to note that one of the key requirements for this project was to ensure that all technologies used were fully open source, a constraint imposed to minimize costs and maximize transparency. Financial resources were severely limited, adding pressure to optimize performance within a strict budget. Additionally, the scarcity of high-quality medical data posed a considerable challenge, necessitating creative solutions to achieve robust and reliable model performance despite these limitations. By advancing the capabilities of AI in healthcare, the project aspires to set a new standard in patient care, blending cutting-edge technology with practical, real-world application

4 Internship diary:



START	DUE	STATUS	%	
17/Jun	19/Jun	Finished	0%	Transformers & LLM architecture understanding
20/Jun	21/Jun	Finished	0%	Dataset Search
24/Jun	24/Jun	Finished	0%	LLama3 understanding
25/Jun	26/Jun	Finished	0%	FineTuning understanding
26/Jun	28/Jun	Finished	0%	Finetuned LLMs testing
28/Jun	03/Jul	Finished	0%	RAG understanding
04/Jul	05/Jul	Approved	0%	More data collection
08/Jul	12/Jul	Finished	0%	RAG implementation
12/Jul	15/Jul	Rejected	0%	Llama3 & Rag testing
16/Jul	17/Jul	Finished	0%	Bert understanding
18/Jul	19/Jul	Finished	0%	DistilBert Finetuning
19/Jul	19/Jul	Finished	0%	DistilBert Testing & Evaluation
22/Jul	25/Jul	Finished	0%	Roberta Finetuning
25/Jul	26/Jul	Finished	0%	Roberta Testing & Evaluation
29/Jul	30/Jul	Finished	0%	More LLM architecture Undersatnding
31/Jul	01/Aug	Finished	0%	NLP & NER Undersatnding
02/Aug	06/Aug	Finished	0%	Synthetic dataset creation
06/Aug	07/Aug	Finished	0%	Roberta NER Finetuning
07/Aug	07/Aug	Approved	0%	Roberta NER testing & Evaluation
08/Aug	12/Aug	Finished	0%	Gemma 2 2B FineTuning
12/Aug	12/Aug	Finished	0%	Gemma 2 Testing & Evaluation
12/Aug	14/Aug	Finished	0%	Creating the final chatbot system
14/Aug	15/Aug	Finished	0%	Testing & Evaluation
16/Aug	16/Aug	Approved	0%	Validation & Solution Presentatio
			0%	

Figure 3 : Gantt diagram

For a more detailed high resolution image, check out the github repo.

5 Work Done:

5.1 Understanding the Transformers architecture:

The foundation of our medical chatbot system is built upon the Transformer architecture, which has fundamentally changed the landscape of natural language processing (NLP). Transformers have addressed the limitations of earlier models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs), both of which struggled with capturing long-term dependencies in text data and were prone to issues such as vanishing gradients during training. In contrast, Transformers can model dependencies over long sequences and have proven to be highly effective for a wide range of NLP tasks.

- **The Architecture Overview:**

The Transformer model is composed of two main components: the Encoder and the Decoder. Each of these components is made up of a stack of identical layers, typically six or more, depending on the size of the model.

1. **Encoder**:
    - The Encoder is a stack of layers, each containing two main sub-layers: a **self-attention mechanism** and a **feed-forward neural network**. The Encoder's job is to process the input text and extract meaningful features that capture both the content and the relationships between words.
    - Each word in the input sequence is converted into an embedding vector, which is then processed by the self-attention mechanism to learn which words in the sequence are most relevant to each other. The output of the self-attention layer is then fed into a fully connected feed-forward network to further transform the information. Residual connections and layer normalization are applied around each sub-layer to stabilize training.
2. **Decoder**:
    - The Decoder is also a stack of layers, but it has an additional sub-layer called the **encoder-decoder attention mechanism**. This mechanism allows the Decoder to focus on relevant parts of the input sequence generated by the Encoder. The Decoder's main purpose is to generate the output text, such as a translated sentence or, in our case, a medical response, one word at a time.
    - Like the Encoder, each Decoder layer includes a self-attention sub-layer and a feed-forward neural network, but it also incorporates the encoder-decoder attention to align the generated output with the input features.

- **Self-Attention Mechanism:**

  At the heart of the Transformer architecture is the **self-attention mechanism**, which allows the model to weigh the importance of each word in a sentence relative to every other word. This ability is critical for understanding the context and relationships between words, especially in complex medical dialogues where certain symptoms or terms may depend on one another for accurate interpretation.

    1. **How Self-Attention Works**:
        - The self-attention mechanism uses three main components for each word in the input sequence: **Queries (Q)**, **Keys (K)**, and **Values (V)**. These are derived from the input embeddings using learned weight matrices.
        - For each word in the input, the model computes a dot product between the Query vector of that word and the Key vectors of all other words in the sequence. This dot product gives a measure of how much attention one word should pay to another. The scores are then scaled and normalized using a softmax function to ensure they add up to 1, creating a distribution of attention weights.
        - Finally, each word's Value vector is multiplied by its corresponding attention weight, and these weighted values are summed to produce the final output for that word. This process allows the model to focus more on words that are contextually relevant and less on those that are not.

2. **Mathematical Representation**:
   - The self-attention output for each word can be mathematically represented as:

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

   - Here, **Q** represents the Queries, **K** represents the Keys, **V** represents the Values, and dk is the dimension of the Key vectors. The scaling factor $\sqrt{dk}$ prevents the dot products from becoming too large, stabilizing the gradients during training.

- **Multi-Head Attention**

To enhance the model's ability to capture different types of relationships between words, Transformers use **multi-head attention**. Instead of computing a single set of attention scores, the model creates multiple sets of Queries, Keys, and Values, each with different learned weight matrices. The output of these attention heads is then concatenated and transformed through a linear layer. This allows the model to consider multiple perspectives or "attention heads" simultaneously, making it better at understanding complex language structures.

- **Positional Encoding**

Since Transformers do not inherently process data sequentially, they need a way to understand the order of words in a sentence. This is where **positional encoding** comes into play. Positional encodings are added to the input embeddings to give the model information about the position of each word in the sequence. These encodings are designed so that the model can distinguish between words and understand their order, which is especially crucial in medical contexts where the sequence of symptoms or events can change the meaning of a conversation. The positional encodings use a combination of sine and cosine functions to generate patterns that vary with the position, enabling the model to learn relationships based on the order of words.

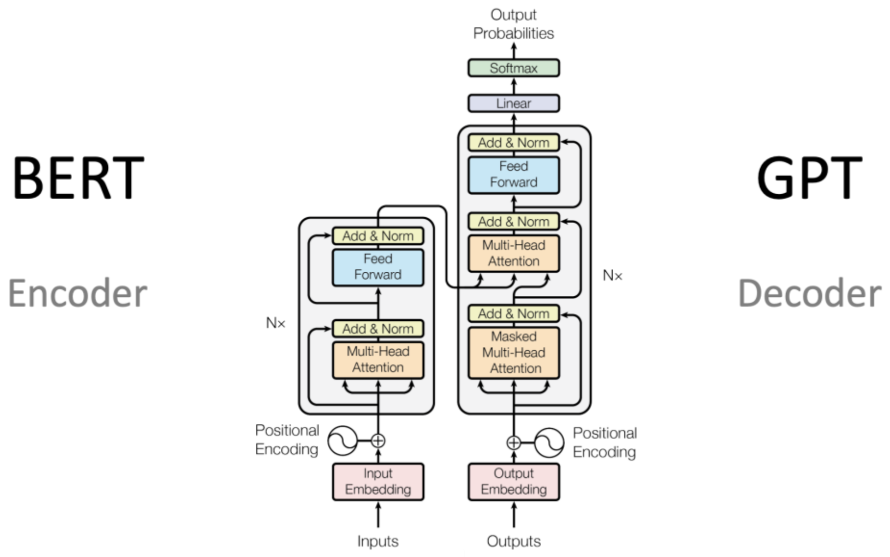


Figure 4 : Transformers Architecture

Overall, the Transformer architecture, with its self-attention and positional encoding mechanisms, provides a powerful framework for handling complex language tasks. In the context of our medical chatbot, this architecture enables the system to interpret and respond to medical queries with precision, taking into account the nuances of language and the relationships between different medical terms and symptoms.

5.2 NER Task:

- The doctors ability to understand and interact meaningfully with patients heavily depends on Named Entity Recognition (NER), a crucial natural language processing (NLP) task. NER is used to identify and classify key pieces of information, referred to as "entities" within a given text. In the context of a medical chatbot, NER enables the system to extract patient-specific details such as age, weight, and height from user queries, which are essential for accurate medical analysis and guidance but most importantly this task was used to make a record for the patient that a doctor usually use to track his patients.
- **What is Named Entity Recognition (NER)?**
  1. **Definition and Purpose**:
     - Named Entity Recognition (NER) is an NLP task that involves identifying and categorizing words or phrases in a text into predefined classes. These classes can include proper nouns (like names of people, organizations, and places), dates, quantities, medical terms, or any other entities relevant to the application.
     - In medical applications, NER is critical for extracting specific data points that can influence diagnoses or treatment recommendations. For instance, knowing a patient's age and weight can help determine if certain symptoms are age-related or if a specific dosage of medication is appropriate.
  2. **How NER Works**:
     - NER models scan a sentence and classify each word or token into one of the predefined entity categories or mark it as "O" (outside of any entity category). For example, in the sentence: **"I am 30 years old and weigh 75 kilograms."**
     The NER model would label "30 years" as an **AGE** entity and "75 kilograms" as a **WEIGHT** entity.
     - The classification is usually performed using a tagging scheme called **B-I-O**:

B: Indicates the beginning of an entity (e.g., "**B-AGE**" marks the start of an age phrase).

I: Indicates that the token is inside an entity (e.g., "**I-AGE**" continues the age phrase).

O: Indicates that the token is outside of any named entity.

- **Implementation in the Medical Chatbot :**
  - **BERT (Bidirectional Encoder Representations from Transformers)**: BERT's architecture is bidirectional, meaning it looks at words in both directions (left-to-right and right-to-left) to understand context. This bidirectionality makes it highly effective for tasks that require a nuanced understanding of language, such as entity extraction[2].This model was used for the first iteration of the ner model.
  - **RoBERTa (Robustly Optimized BERT Approach)**: RoBERTa builds on BERT by improving the training methodology. It removes the Next Sentence Prediction objective used in BERT and increases training data size and mini-batch sizes. These enhancements make RoBERTa better suited for our NER task, where precise extraction of medical information is essential.[3]
  - **Custom NER Dataset**: To fine-tune RoBERTa for our needs, I created a custom dataset with 1,000 synthesized samples that included realistic medical conversations embedded with age, height, and weight details. Each sentence was manually annotated using the B-I-O tagging scheme (Beginning, Inside, Outside), which is crucial for accurately training NER models.

5.3 Q-LORA (Quantized Low-Rank Adaptation) for Efficient Model Fine-Tuning:

Given the resource constraints involved in deploying and fine-tuning large language models (LLMs) such as LLaMA3 and Gemma2 2B, Q-LORA (Quantized Low-Rank Adaptation) was employed as a highly efficient method for model adaptation. Here's an in-depth explanation of how Q-LORA operates and the fine-tuning process:

Concept of Q-LORA

Q-LORA is an advanced technique that combines two main strategies to efficiently fine-tune large-scale models:

1. **Low-Rank Adaptation (LoRA) Mechanism**:
   - **Problem with Full Fine-Tuning**: Normally, fine-tuning an LLM involves updating a massive number of parameters, which is computationally expensive and requires significant memory. This is particularly challenging for models with billions of parameters, like Llama3 8B.
   - **LoRA Solution**: Instead of updating all parameters, LoRA introduces small, low-rank matrices into the attention layers of the Transformer architecture. These matrices, known as "rank reduction matrices," are trained to capture task-specific information. During inference, the original model's parameters remain frozen, and only these newly introduced low-rank matrices contribute to the model's output.
   - **How It Works**: In LoRA, the weights of the attention layers, **W**, are decomposed into two smaller matrices, A and B, such that: $\mathbf{W'=W+\alpha \cdot A \cdot BW'}$

     Here:
     - **W**: The original weight matrix of the attention layer.
     - **A** and **B**: Low-rank matrices with much smaller dimensions (e.g., rank 4 or 8), which are trained during fine-tuning.
     - **α**: A scaling factor that controls the influence of the low-rank adaptation.
   - **Advantages**: This approach drastically reduces the number of parameters that need to be adjusted, making the fine-tuning process much more memory-efficient and computationally feasible. By focusing on these low-rank matrices, LoRA ensures that the most significant aspects of the model are fine-tuned for the specific task, without overhauling the entire network.
2. **4-bit Quantization**:
   - **Why Quantization?**: LLMs are typically trained and stored using 16-bit or 32-bit floating-point precision, which consumes a substantial amount of memory. Quantization reduces the precision of these weights, compressing the model and reducing memory usage.
   - **How 4-bit Quantization Works**: In Q-LORA, the weights of the Transformer model are quantized from 32-bit floating-point precision to 4-bit integer representation. This means that each weight value is stored using only 4 bits instead of 32 bits, resulting in an 8-fold reduction in memory usage. The quantization process maps the continuous range of weight values to a smaller set of discrete values, while carefully minimizing the loss of information.
   - **Challenges and Solutions**: One challenge with quantization is the potential degradation of model performance due to the loss of precision. However,
     Q-LORA addresses this by carefully choosing which layers and parameters to quantize and applying scaling factors to maintain model quality. For instance, sensitive parameters, like those in the embedding layers, may remain in higher precision, while less critical layers are quantized.

Fine-Tuning Process Using Q-LORA

1. **Inserting LoRA Layers**:
   - LoRA layers are inserted into specific parts of the Transformer model, typically the attention layers. These layers are initialized with small random values and gradually learn task-specific representations during fine-tuning.
   - During each forward pass, the output of the original attention layer is modified using the LoRA layers. The original computation **XW** (where **X** is the input and **W** is the weight matrix) is augmented as $\mathbf{X(W+\alpha \cdot A \cdot B)}$. Here, **A** and **B** are the learnable low-rank matrices, and α is a scaling factor.
2. **Training and Optimization**:
   - The model is fine-tuned using task-specific data, in this case, medical dialogues and queries relevant to the chatbot's purpose. The optimization process only updates the parameters of the LoRA layers (i.e: **A** and **B**), while the rest of the model remains frozen.
   - Loss functions like Cross-Entropy Loss are used to guide the training process. For text generation tasks, special attention is given to the alignment of generated responses with ground truth medical information, ensuring accuracy and coherence.
3. **Efficiency and Performance**:
   - By updating only the low-rank matrices and using 4-bit quantization, Q-LORA enables the fine-tuning of massive models on GPUs with limited memory. Despite the reduced precision, the model retains high performance, making it practical for deployment in real-world applications, such as the medical chatbot.
4. **Gemma2 2B Specifics and Training Setup**:
   - **Gemma2 2B**: Gemma2 2B is a state-of-the-art large language model known for its ability to generate human-like, contextually aware text. With 2 billion parameters, it strikes a balance between being powerful enough for complex text generation tasks, such as medical chatbot conversations, and being resource-friendly enough to fine-tune effectively. Despite its relatively smaller size compared to models like LLaMA 3 8B or Mistral 7B, Gemma2 2B has demonstrated competitive performance, often surpassing these larger models on certain benchmarks due to its optimized architecture. However, fine-tuning this model was still a challenge, given the project's financial and computational constraints.
   - **Training Setup on Runpod.io**: For this project, the Gemma2 2B model was fine-tuned using the Q-LORA method on Runpod.io[5], a cloud platform that offers rentable GPUs. We specifically used an NVIDIA RTX 3090, a powerful yet cost-effective option that provided sufficient computational capability for our needs. The fine-tuning process lasted over 30 hours, spanning 3 full epochs on the entire medical dataset. This setup ensured a balance between comprehensive learning and efficient use of resources, helping the model generalize well without overfitting.
   - **Efficient Training and Evaluation Strategy**: To optimize the training process, evaluations were conducted every 1,000 steps. Instead of assessing the model on the full dataset, we performed evaluations on a randomly chosen subset of 500 examples. This strategy allowed for efficient monitoring of model performance while minimizing the computational overhead associated with frequent evaluations. Only the best model, based on evaluation metrics, was retained after training, ensuring that we had the most effective version for deployment.
   - **Attention Mechanism Optimization**: The fine-tuning process specifically targeted the attention mechanisms within the model, a technique that significantly

reduces the number of parameters needing adjustment. This focus on the attention layers, rather than the entire model, was made possible by the use of Q-LORA, which updates only a small subset of parameters to capture task-specific knowledge. Additionally, we implemented Flash Attention 2.0, a highly efficient algorithm for computing the attention mechanism. Flash Attention 2.0 uses optimized memory layouts and algorithms to reduce the number of memory reads and writes, which significantly speeds up the computation and reduces memory usage, making it ideal for handling large-scale models like Gemma2 2B.

- **Focus on Flash Attention 2.0**: This cutting-edge attention mechanism implementation streamlines the process by minimizing memory bottlenecks, making training faster and more resource-efficient. Flash Attention 2.0 rearranges the order of operations to maximize cache efficiency and reduce the number of times data needs to be moved between memory and the computation units. This implementation played a crucial role in keeping the model's training time within our resource limitations while still achieving high performance.
- **Focus on Medical Conversation Nuances**: The training emphasized the model's ability to understand and generate nuanced medical conversations. This included accurately interpreting symptoms, understanding patient history, and providing responses that are both medically accurate and empathetic. Gemma2 2B's strong performance in processing complex language structures and generating contextually appropriate content made it an excellent choice for this application, enhancing the reliability and user-friendliness of the chatbot.
- **Monitoring with Weights & Biases**: To ensure that the fine-tuning process was as effective and efficient as possible, Weights & Biases (W&B) was used for monitoring and tracking the training metrics. This platform provided real-time insights into training progress, including loss curves, evaluation metrics, and resource utilization. By visualizing these metrics, we were able to make informed decisions about hyperparameter adjustments and ensure that the model was converging as expected. The use of Weights & Biases also allowed for easier debugging and optimization, contributing to a smoother and more transparent training workflow.

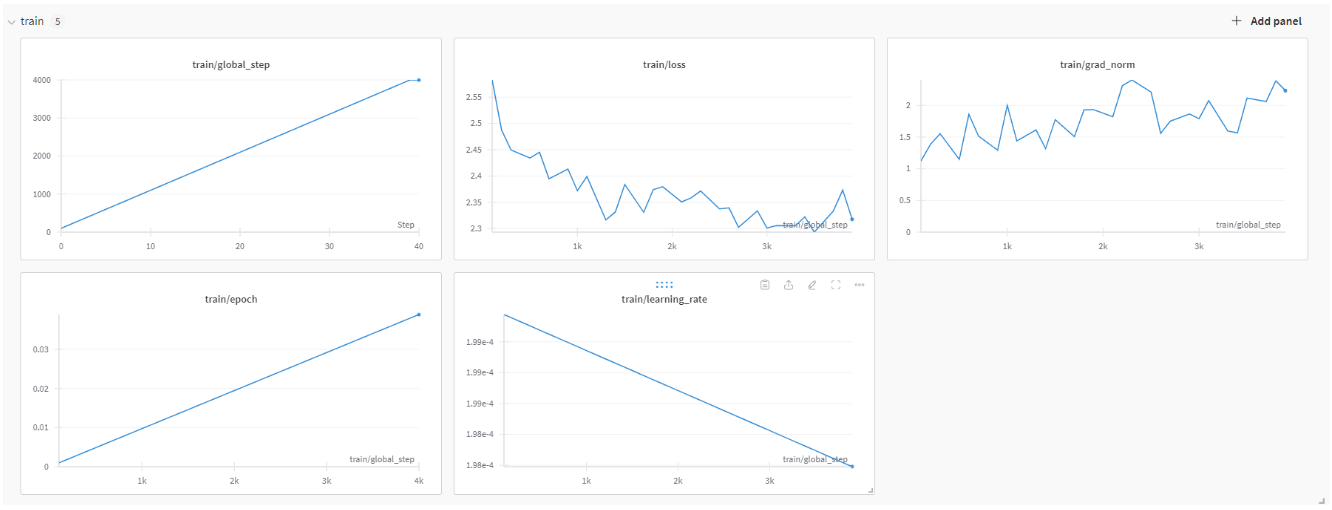


Figure 5: weights & biases to monitor model training

5.4. RAG (Retrieval-Augmented Generation) for Contextual Responses:

The chatbot's ability to deliver medically accurate, evidence-based, and context-aware information is fundamentally supported by the RAG architecture. This approach is highly suitable for applications where the reliability and relevance of the information are critical, such as healthcare.

How RAG Works?

RAG, or Retrieval-Augmented Generation, is a hybrid framework that combines two powerful components to generate responses:

1. **Retriever**: The retriever's role is to search and extract relevant pieces of information from a large corpus of documents. In this project, the retriever uses a dense retrieval method, meaning it maps both queries (user inputs) and documents (medical texts) into a shared vector space. The goal is to maximize the similarity between the query and the most relevant documents. The steps involved are:
   - **Query Embedding**: When a user inputs a query (for example, a question about a symptom), the query is transformed into a dense vector representation using a pre-trained HuggingFace embedding model. This vector captures the semantic meaning of the query.
   - **Document Retrieval**: The retriever then searches through the knowledge base, which has been indexed using FAISS (Facebook AI Similarity Search), to find documents that have the highest cosine similarity with the query vector. FAISS is a highly efficient library that performs fast similarity search and clustering of dense vectors, making it ideal for large-scale retrieval tasks.
2. **Generator**: Once the relevant documents are retrieved, they are fed into a language model (such as Gemma2 2B or LLaMA3) that acts as the generator. The generator uses the information from the retrieved documents to craft a well-informed, coherent, and contextually relevant response. This response is then provided to the user. The generator is trained to condition its output on both the original user query and the retrieved context, ensuring that it can synthesize information from multiple sources and present it in a way that is easy to understand.

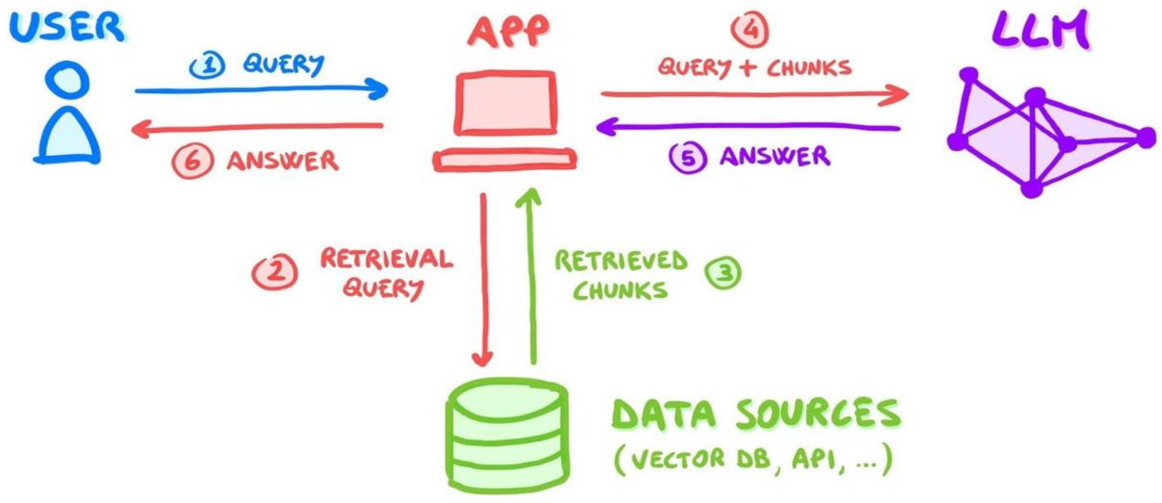


Figure 6: RAG system explanation

Implementation Details

1. **FAISS Database**:
   - **What is FAISS?**: FAISS (Facebook AI Similarity Search) is a library developed by Facebook AI Research that is optimized for searching through large collections of dense vectors. In the context of our chatbot, FAISS is used to index and search a vast database of medical documents efficiently. It allows us to perform nearest-neighbor searches at high speed, which is crucial for real-time applications like a chatbot.[6]
   - **Indexing Process**: The medical literature, which consisted of PDF medical books approved by the CEO (a BIO engineer), was pre-processed and converted into text, with extraneous content like tables of contents and footnotes removed. To make the retrieval system more efficient, the text was divided into smaller, manageable segments through a process called *chunking*. Chunking splits long passages into coherent, shorter chunks, making it easier for the system to match user queries with specific, relevant pieces of information rather than retrieving an entire document.

     Each chunk was then transformed into dense vector representations using HuggingFace embeddings, which capture the semantic meaning of the text. These vectors were indexed using FAISS (Facebook AI Similarity Search), a library optimized for fast and efficient similarity searches. This indexing setup allows the chatbot to quickly retrieve and reference precise and contextually relevant information from authoritative medical sources, ensuring accurate responses during user interactions.
   - **Search Efficiency**: FAISS supports various indexing methods, such as Inverted File (IVF) indices and Product Quantization (PQ), which reduce memory usage while maintaining high retrieval accuracy. We selected an indexing method that balanced speed and precision to ensure that the chatbot could deliver quick yet reliable answers.
2. **HuggingFace Embeddings**:
   - **Pre-trained Embedding Models**: We used pre-trained embedding models from HuggingFace to transform both the user queries and the medical documents into dense vectors. These models are trained on large amounts of text data and are designed to capture the semantic meaning of sentences, which is essential for understanding complex medical queries.[7]
   - **Embedding Process**: When embedding the documents, each medical text was split into smaller, manageable chunks to ensure that the embedding model could process them effectively. This chunking process is important because it allows for a more granular search, where the retriever can match specific parts of a document to the user's query rather than having to retrieve an entire document.
3. **Data Sources**:
   - **PDF Medical Books**: The core of the knowledge base consisted of approved medical literature provided in PDF format. After obtaining explicit approval from the CEO, who is a BIO engineer, these documents were pre-processed for indexing. The PDFs were converted into text, and any extraneous content (like tables of contents or footnotes) was removed to ensure that the retriever only indexed high-quality medical content.
   - **Ethical Considerations**: Ensuring the use of authoritative and reliable sources was critical, given the sensitive nature of medical information. The approval from the CEO ensured that the data was both ethically sourced and medically accurate.

4. **How the RAG System Enhances Responses**:
    - **Contextual Relevance**: By using the RAG framework, the chatbot can provide responses that are grounded in real, authoritative medical literature. This is particularly important in healthcare, where the accuracy of information can have significant consequences. The RAG system ensures that responses are not only grammatically and syntactically correct but also backed by verified medical knowledge.
    - **Dynamic Information Retrieval**: The RAG system allows for dynamic retrieval of the most relevant information. For example, if a patient asks about symptoms of a rare disease, the retriever can fetch up-to-date medical definitions, case studies, or treatment guidelines. The generator then uses this information to construct a response tailored to the patient's needs.

5. **Challenges and Optimizations**:
    - **Balancing Speed and Accuracy**: One of the main challenges in implementing RAG was ensuring that document retrieval was both fast and accurate. FAISS's indexing methods and the use of efficient embedding models were key to overcoming this challenge.
    - **Handling Ambiguity in Queries**: Medical questions can often be ambiguous or vague. To address this, the RAG system uses multiple retrieved documents to provide a comprehensive answer, covering different aspects of the user's question.

5.5. Model Experiments and Learning:

Throughout the development process, several experiments were conducted to optimize model performance:

- **DistilBERT[8] and RoBERTa for Q&A**: Initially, I experimented with using DistilBERT and RoBERTa for question-answering tasks. However, these models, while efficient for classification and NER tasks, failed to generate coherent and meaningful answers. This failure underscored the importance of using models specifically designed for text generation.
- **Subset Training on LLaMA3**: To explore data efficiency, I split the custom dataset into ten smaller subsets and trained LLaMA3 on Kaggle. However, due to hardware limitations and the complexity of generative tasks, these attempts did not yield the desired results, leading to a strategic pivot towards a smaller model like Gemma2 2B fine-tuning with Q-LORA.

5.6. Testing and Evaluation

The project involved rigorous testing and evaluation to ensure the chatbot's reliability and accuracy:

- **Gemma2 2B Evaluation**: The fine-tuned Gemma2 2B model was assessed using metrics like BLEU and ROUGE to measure the quality of text generation. Evaluation loss and accuracy were also tracked throughout the training process to monitor model performance. The results were promising, with the model showing strong performance characterized by low evaluation loss and high BLEU and ROUGE scores, which indicated that it generated coherent and contextually relevant medical advice. However, the CTO emphasized that I should not worry too much about comprehensive testing at this stage, as the model's true evaluation would be conducted extensively by a team of doctors. This team will use their own medical data, passed through the RAG system, to rigorously test and validate the chatbot's performance once the project is fully completed. This additional layer of expert testing ensures that the model will be evaluated against real-world medical scenarios, adding another level of reliability and trustworthiness to the system before deployment.

- **NER Model Performance**: The RoBERTa model achieved an impressive F1 score of 92%, reflecting its high accuracy in extracting patient-specific information. Precision and recall scores were also balanced, showing that the model was effective at both identifying relevant entities and minimizing false positives.

5.7. System Integration and Final Assembly

The final step involved integrating all components into a unified system:

- **NER Integration**: The RoBERTa NER model was incorporated into the chatbot to extract patient information dynamically. This data was then used by the generative models to tailor medical responses to the user's needs.
- **RAG System Integration**: The FAISS-based RAG system was linked to the generative models, allowing the chatbot to reference external medical documents and provide evidence-based responses.
- **T5 Summarization**: A T5 model was used to continuously summarize ongoing conversations. This feature is crucial for healthcare professionals who need to quickly review patient interactions, ensuring that important details are readily accessible.[9]
- **LangChain Framework**: To manage the integration and organization of all these components, the LangChain framework was utilized. LangChain is designed for building applications powered by language models, and it provided the structure needed to link the various models, such as the NER, RAG, and summarization modules. This framework made it easier to manage the data flow between components, ensure smooth interactions, and handle the complexity of the system in a modular and scalable way.

By using the LangChain framework, the project achieved a well-organized and integrated solution that efficiently connects all components, ensuring that the chatbot operates seamlessly and provides a reliable and responsive experience for users.

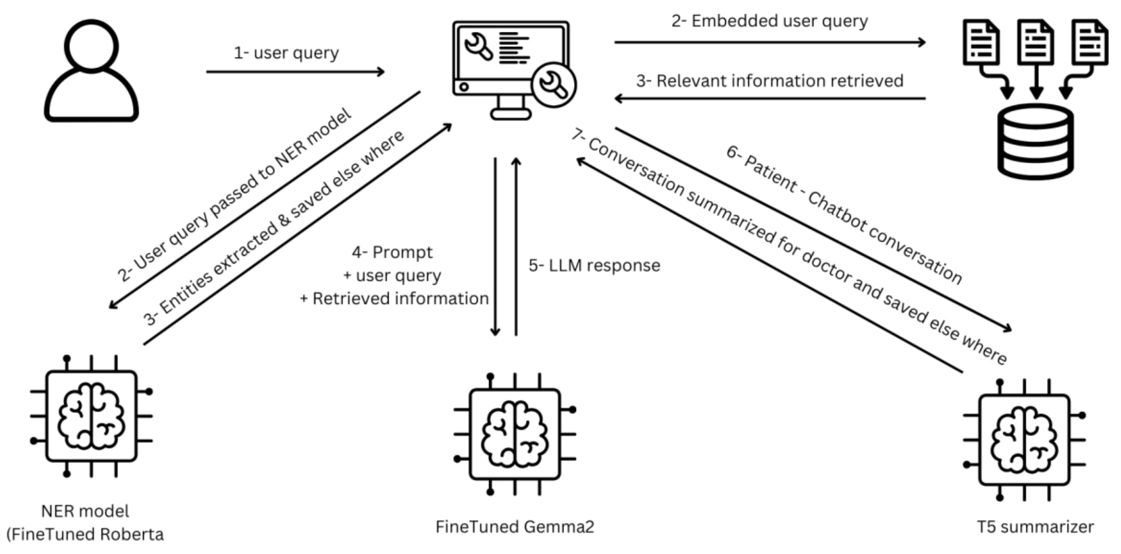

1- user query
2- Embedded user query
3- Relevant information retrieved
2- User query passed to NER model
3- Entities extracted & saved else where
4- Prompt
+ user query
+ Retrieved information
5- LLM response
6- Patient - Chatbot conversation
7- Conversation summarized for doctor and saved else where
NER model
(FineTuned Roberta
FineTuned Gemma2
T5 summarizer

Figure 7: Chatbot system explanation

5.8. Challenges and Lessons Learned

- **Model Limitations**: The experiments with DistilBERT and RoBERTa highlighted the importance of understanding model architectures and their suitability for different NLP tasks. This experience reinforced the value of models like LLaMA3 and Gemma2 for text generation and Q&A tasks.
- **Data Constraints**: Working with large models on limited hardware required innovative solutions like Q-LORA and the use of cloud GPUs. These constraints taught me the importance of efficient resource management and the trade-offs between model size, precision, and performance.

6 Skills acquired

Course	Skill acquired
Deep Learning	This course provided a strong foundation in understanding neural network architectures, which was crucial for working with models like BERT and RoBERTa for NER tasks, as well as fine-tuning large language models like Gemma2 2B and LLaMA3 using techniques such as Q-LORA and 4-bit quantization.
Machine Learning	It taught me the principles of model optimization and evaluation metrics, which I applied extensively to assess model performance using BLEU, ROUGE, and F1 scores, ensuring that the AI-generated medical insights were accurate and reliable.
Deep Learning	This course was instrumental in understanding tokenization, embeddings, and the self-attention mechanism in Transformers, enabling me to implement the RAG system effectively and leverage HuggingFace embeddings for accurate document retrieval and semantic understanding.
Deep Learning	I learned about sequence tagging and entity extraction, which directly informed the development and fine-tuning of the RoBERTa model for NER, enabling precise extraction of critical patient information like age, weight, and height from user input.
Deep Learning	This course provided essential skills in data preprocessing, feature engineering, and model training, which were vital when creating and annotating a custom dataset for NER and handling the limitations imposed by data scarcity in the project.
Software Engineering in genral.	I applied concepts of modular programming and code organization, ensuring that the system architecture was efficient and maintainable. Knowledge from this course helped me design the chatbot system to handle integration between multiple components, such as RAG, NER, and the summarization models.
Data Structures and Algorithms	This course helped me understand efficient data handling and search algorithms, which were beneficial when working with FAISS for rapid similarity search and building an effective document retrieval system.

7 Conclusion

In conclusion, the development of the medical chatbot was a complex yet rewarding endeavor that successfully integrated advanced AI technologies to enhance patient analysis and provide illness guidance. Throughout the project, significant progress was made in automating routine medical tasks, from extracting patient-specific details using a fine-tuned RoBERTa model for Named Entity Recognition (NER) to generating coherent, contextually relevant medical advice with the Gemma2 2B model, fine-tuned using Q-LORA for efficiency. The implementation of a Retrieval-Augmented Generation (RAG) system further strengthened the chatbot's ability to deliver trustworthy, evidence-based responses by referencing authoritative medical literature.

Despite these achievements, the project faced several challenges, primarily stemming from resource limitations. Due to these constraints, the NER model relied on RoBERTa instead of a more advanced model like DeBERTa, which could have provided better entity extraction performance. Similarly, fine-tuning a more sophisticated model like LLaMA 3.2 for text generation was not feasible, even though it could potentially yield more accurate and natural-sounding responses compared to Gemma2 2B. Another promising evaluation method that could have enhanced the chatbot's performance assessment is using an LLM as a judge, such as GPT models for qualitative judgment. However, this approach also proved impractical due to resource limitations.

Future work will need to address these areas. Upgrading the NER component to a DeBERTa model could significantly improve the precision and recall of entity extraction, making the chatbot more effective in gathering patient information. Additionally, fine-tuning a state-of-the-art language model like LLaMA 3.2 could enhance the quality of medical responses, making interactions more reliable and user-friendly. Employing advanced evaluation methods, such as using LLMs for model assessment, could also offer a more nuanced understanding of the chatbot's strengths and weaknesses. Finally, rewriting the conversation to a more capable model like LLaMA 3 would likely improve the coherence and fluency of the conversation summary, pushing the boundaries of what is achievable in AI-driven medical support.

Overall, the project lays a solid foundation for future advancements in AI-powered medical chatbots. It demonstrates the potential of integrating multiple AI models into a cohesive system, even when working within significant resource constraints, while highlighting key areas for improvement and innovation.

Note: All the work for this project can be found in my GitHub through this link:
https://github.com/adam-fendri/MeidcalChatbot/tree/master

Bibliographies

[1] Pura Solutions , http://www.pura-tech.com , consulted on 02/11/2024.
[2] BERT , https://huggingface.co/google-bert/bert-base-uncased , consulted on 02/11/2024.
[3] RoBERTa , https://huggingface.co/deepset/roberta-base-squad2 , consulted on 02/11/2024.
[4] Gemma2 2B , https://huggingface.co/google/gemma-2-2b , consulted on 02/11/2024.
[5] Runpod.io , https://www.runpod.io , consulted on 02/11/2024**.**
[6] FAISS , https://engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/ , consulted on 02/11/2024.
[7] HuggingFaceEmbedding , https://docs.llamaindex.ai/en/stable/examples/embeddings/huggingface , consulted on 02/11/2024.
[8] DistilBERT , https://huggingface.co/docs/transformers/en/model_doc/distilbert , consulted on 02/11/2024.
[9] T5 model , https://huggingface.co/google-t5/t5-base , consulted on 02/11/2024.
[10] LangChain Framework , https://www.langchain.com/ , consulted on 02/11/2024.