Group – 3: LLAMA, MD: AI

Specializing in Gynecology

I. Introduction

The rapid evolution of large language models (LLMs) has opened new frontiers in artificial intelligence, particularly in the domain of specialized applications. While general-purpose LLMs have demonstrated remarkable capabilities, their effectiveness is often limited when applied to highly specialized fields such as medicine, where domain expertise and contextual precision are paramount. To address this challenge, we present ***LLAMA, MD: Specializing in Gynecology***, a project focused on fine-tuning an LLM for expertise in the medical domain, with an emphasis on pregnancy-related topics in gynecology.
Our project is structured around four key components. First, using topic modeling techniques, we extracted the top 10 topics from a medical dataset and identified pregnancy as the primary domain for specialization based on frequency analysis. Next, we curated a comprehensive knowledge base comprising digital versions of 10 authoritative books on pregnancy and relevant Wikipedia data. This corpus serves as the foundation for training and retrieval processes.

Building on this foundation, we fine-tuned **Llama 3.2 1B**, a compact yet powerful LLM with 1 billion parameters, using the AI Medical Chatbot dataset. This step ensured alignment with medical conversational tasks, enabling the model to generate contextually accurate and relevant responses. Finally, we integrated the fine-tuned model with a Retrieval-Augmented Generation (RAG) framework, allowing it to dynamically access external knowledge and emulate a virtual gynecologist.

This paper outlines our methodology and demonstrates the feasibility of domain-specialized LLMs in healthcare. By narrowing the model's focus and leveraging advanced retrieval techniques, *LLAMA, MD* aims to provide expert-level assistance in the critical subfield of pregnancy, illustrating the transformative potential of LLMs in improving medical accessibility and expertise.

II. Dataset Description:

The AI Medical Chatbot data can be found here. The dataset contains Patient questions and Doctor's responses to them along with a brief description of the actual query of the patient. This dataset contains **251,000 samples** and focuses on medical communication between patients and doctors. It comprises **three key features**: **Description**, **Patient**, and **Doctor**.

1. **Description**: This field contains the primary question or concern posed by the patient. These descriptions cover a wide range of health-related topics, including symptoms,

causes, and possible treatments. The length of the description varies, ranging from concise questions to more detailed explanations.

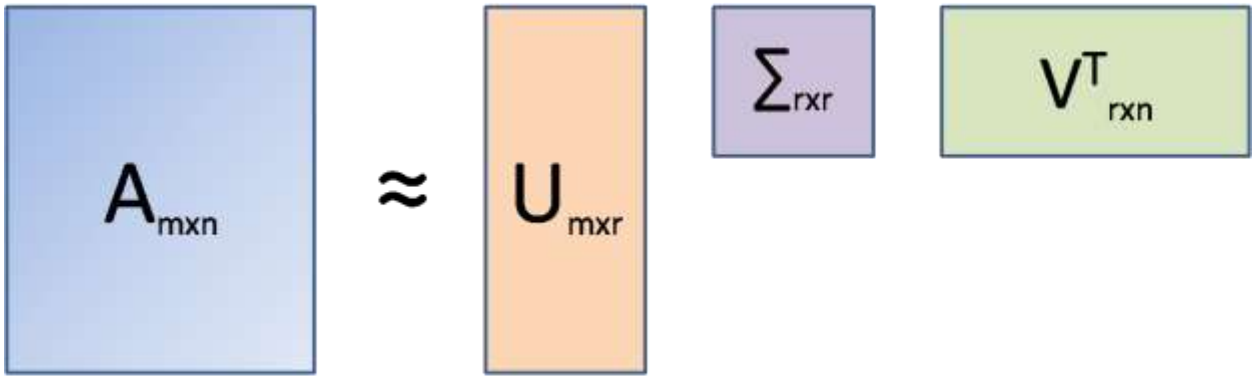
2. **Patient**: This feature captures the patient's detailed query or narrative. Patients often provide context, such as age, medical history, or specific concerns, to give the doctor a better understanding of their issue. The patient statements vary significantly in length, spanning from short descriptions to detailed paragraphs.
3. **Doctor**: This field represents the doctor's response to the patient's query. The responses are intended to provide medical advice, guidance, or further recommendations for follow-up. The length of the doctor's response also varies, including concise answers or more comprehensive explanations and instructions. Some responses may suggest additional investigations, tests, or consultations with specialists.

III. Algorithms Used:

1. Latent Semantic Analysis (LSA)

Latent semantic analysis is a topic modeling technique for uncovering latent topics in documents by analyzing word co-occurence. In machine learning, latent semantic analysis (LSA) is an approach to topic modeling. LSA uses dimensionality reduction to create structured data from unstructured text in order to aid text classification and retrieval.

Before getting into the concept of LSA, let us have a quick intuitive understanding of the concept. When we write anything like text, the words are not chosen randomly from a vocabulary. Rather, we think about a theme (or topic) and then chose words such that we can express our thoughts to others in a more meaningful way. This theme or topic is usually considered as a latent dimension. LSA tries to extract the dimensions using a machine learning algorithm called Singular Value Decomposition or SVD, which is essentially a matrix factorization technique. In this method, any matrix can be decomposed into three parts as shown below.



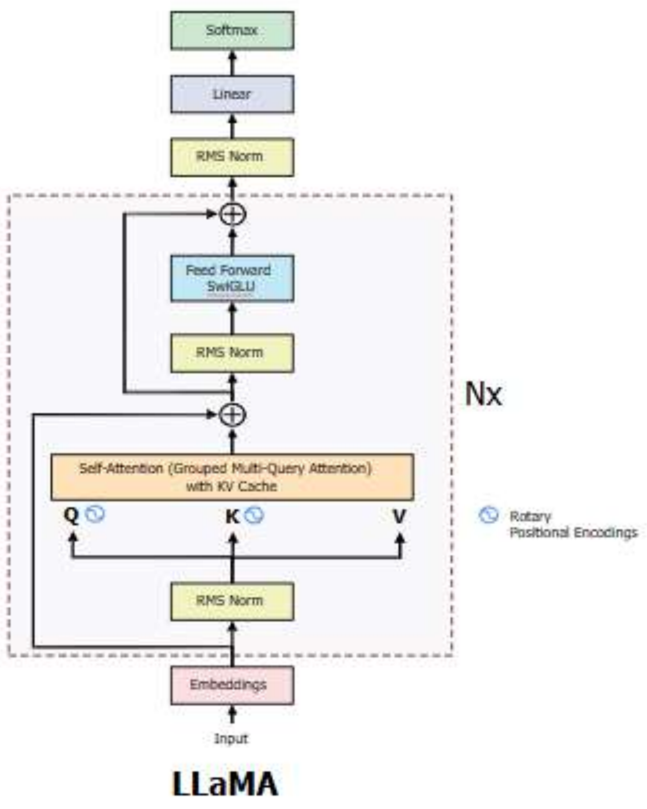
Here, A is the document-term matrix (documents in the rows(m), unique words in the columns(n), and frequencies at the intersections of documents and words). It is to be kept in mind that in LSA, the original document-term matrix is approximated by way of multiplying three other matrices, i.e., U, ∑ and V^T. Here, r is the number of aspects or topics. Once we fix r (r<<n) and run SVD, the outcome that comes out is called Truncated SVD and LSA is essentially a truncated SVD only.

2. Model Description: LLAMA 3.2 1B:

LLaMA (Large Language Model Meta AI) represents a family of transformer-based language models developed by Meta, with LLaMA 3.2 being one of its iterations. The LLaMA series was first introduced to address the challenges of balancing computational efficiency and performance in natural language processing tasks. Over successive versions, LLaMA evolved to incorporate innovations in architecture, parameter efficiency, and training methodologies.

LLaMA 3.2 builds upon the foundation established by its predecessors, leveraging state-of-the-art advancements in transformer architectures. Its design focuses on optimizing memory usage through techniques like grouped multi-query attention and the use of float16 precision, which significantly reduces computational overhead. The 1B parameter variant of LLaMA 3.2 was specifically developed to cater to applications requiring high-quality text understanding and generation within computationally constrained environments.

The LLaMA 3.2 (1B) architecture is a highly efficient transformer-based Large Language Model (LLM) designed for computational efficiency and performance in natural language processing tasks. Below, we describe the key components of the architecture, as depicted in the diagram below:


Softmax
Linear
RMS Norm
Feed Forward SwiGLU
RMS Norm
Nx
Self-Attention (Grouped Multi-Query Attention) with KV Cache
Q
K
V
Rotary Positional Encodings
RMS Norm
Embeddings
Input
LLaMA

1. Input Embeddings

- The model begins with an embedding layer that converts discrete input tokens into dense vector representations.
- Rotary Positional Encodings (RoPE) are applied to the embeddings to introduce positional information, enhancing the model's ability to capture sequential dependencies without the need for absolute positional encodings.

2. Core Transformer Block (Nx Repeated Layers)

The core of LLaMA consists of a stack of repeated transformer blocks, where each block contains the following components:

a) Self-Attention Layer

- **Grouped Multi-Query Attention (MQA)**: A variation of standard multi-head attention, where all attention heads share the same key-value projections, optimizing memory usage.
- **Key-Value Cache**: Used to efficiently store key-value pairs for faster inference, particularly beneficial during autoregressive tasks.
- **Rotary Positional Encodings**: Applied within the attention mechanism to enhance the model's ability to handle long-context dependencies.

b) Feed-Forward Network (FFN)

- A two-layer fully connected network with a **SwiGLU (Gated Linear Unit)** activation, which improves model expressiveness while maintaining efficiency.

$$SwiGLU(x) = x \cdot \sigma(\beta x) + (1 - \sigma(\beta x)) \cdot (wx + b)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$
and β, w, and b are parameters

- This structure allows for non-linear transformations, enhancing the representation of complex patterns in the data.

c) RMSNorm (Root Mean Square Layer Normalization)

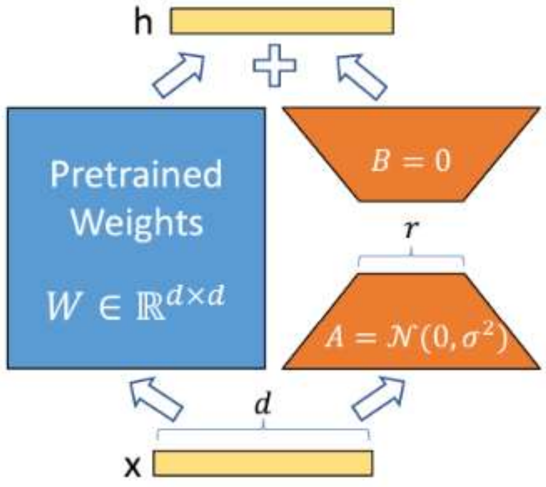
- Replaces traditional Layer Normalization, normalizing activations using their root mean square. This improves training stability and reduces computational overhead.

3. Output Layer

- A final RMSNorm layer normalizes the last hidden state.
- A fully connected linear layer projects the normalized outputs to the vocabulary size, followed by a softmax activation function to compute token probabilities.

3. Fine-tuning: LoRA

Various techniques were explored to fine-tune the LLaMA model, including retraining, retraining with frozen layers, and gradient clipping. Ultimately, the approach that proved most effective was leveraging LoRA (Low-Rank Adaptations) for fine-tuning.
LoRA (Low-Rank Adaptation) is a fine-tuning technique designed to reduce the computational cost of adapting large pre-trained models to new tasks. Instead of updating the full set of parameters of the pre-trained model, LoRA introduces a low-rank decomposition for parameter updates, significantly reducing the number of trainable parameters while preserving model


h
Pretrained Weights
$W \in \mathbb{R}^{d\times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

performance.
Above is an image showing how it works. The core idea is that given a pre-trained weight matrix $W \varepsilon R^{dxd}$, LoRA adds a low-rank update ΔW, such that the adapted weight matrix becomes:

$$W' = W + \Delta W,$$

where $\Delta W = A \cdot B$, and $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times d}$ are low-rank matrices with $r \ll d$.

Training Strategy
During fine-tuning, the original pre-trained weights W remain frozen, and only the parameters of A and B are optimized. This decomposition allows the number of trainable parameters to scale with $O(r \cdot d)$ rather than $O(d^2)$, where r is the rank of the decomposition. Typically, r is set to a small value, such as 8 or 16, to achieve significant parameter reduction.
Initialization
To ensure that W' begins close to W, LoRA initializes A and B as follows:

$$A \sim \mathcal{N}(0, \sigma^2), \quad B = 0,$$

where $N(0, \sigma^2)$ denotes a Gaussian distribution with mean 0 and variance σ^2. This ensures that the initial low-rank update ΔW is close to zero, preserving the pre-trained model's performance at the start of fine-tuning.

Forward Pass
In the forward pass, the input $\mathrm{x}\varepsilon\mathrm{R}^{\mathrm{d}}$ is transformed using the adapted weight matrix:

$$h = W \cdot x + \Delta W \cdot x = W \cdot x + (A \cdot B) \cdot x.$$

Here, A maps the input x into a lower-dimensional space of rank r, and B maps it back to the original dimensionality d.

Advantages

1. **Efficiency**: By freezing the original weights and optimizing only the low-rank matrices, LoRA reduces memory usage and computational costs.
2. **Parameter Efficiency**: The number of trainable parameters is significantly reduced, enabling fine-tuning of large models with limited resources.
3. **Modularity**: LoRA updates can be applied as additional modules, keeping the original model intact.

LoRA enables efficient and scalable fine-tuning by introducing low-rank updates to pre-trained models, balancing performance with computational resource requirements. This makes it especially suitable for adapting large models in resource-constrained environments.

4. Fine-tuning: Prompt Formatting

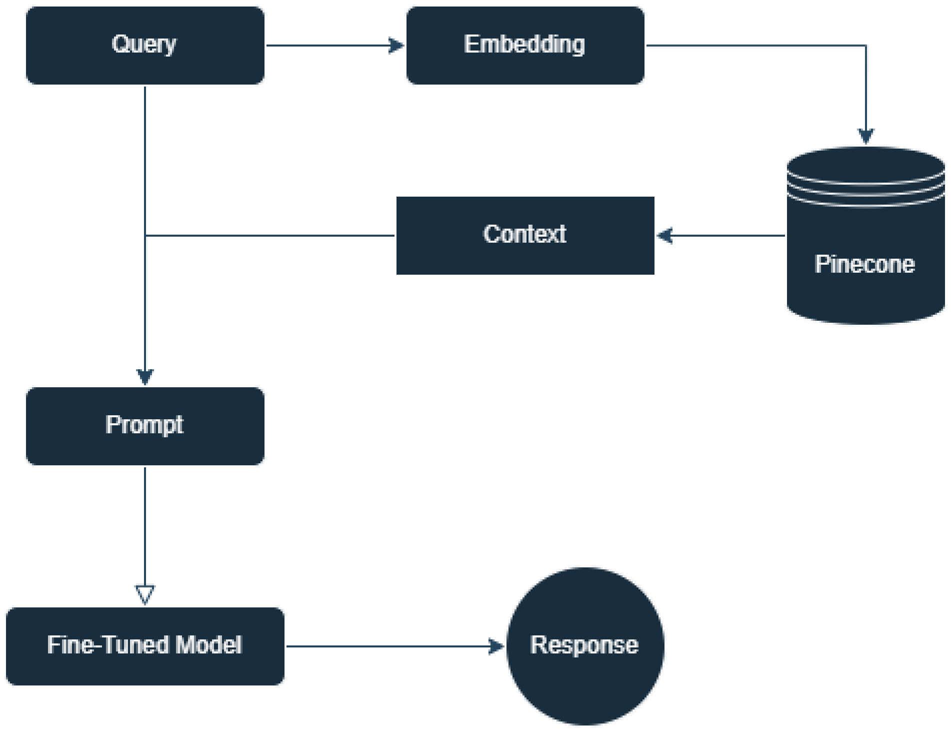
Before understanding prompt formatting, let’s briefly look at its broader area called instruction tuning. Instruction tuning is a training methodology that enhances large language models (LLMs) by aligning them more closely with human instructions. Unlike traditional fine-tuning, which focuses on specific tasks, instruction tuning involves training models on a diverse set of tasks, each accompanied by explicit instructions. This approach enables LLMs to generalize better across various tasks and follow human directives more effectively.

5. RAG

Retrieval-Augmented Generation (RAG) is an advanced technique in natural language processing that enhances the capabilities of large language models (LLMs) by integrating external information retrieval into the generative process. Traditional LLMs generate responses based solely on their pre-existing training data, which can lead to outdated or inaccurate information, especially in rapidly evolving fields such as medical sciences. RAG addresses this limitation by incorporating a retrieval mechanism that accesses up-to-date, relevant data from external sources, thereby grounding the model's outputs in current and contextually appropriate information.

Implementing RAG offers several significant benefits:

- Enhanced Accuracy: By retrieving and incorporating the latest information, RAG ensures that the generated responses are more precise and reflective of current knowledge.
- Scalability: RAG allows models to access new data without the need for extensive retraining, facilitating scalability and adaptability to new information.
- Domain Specialization: By integrating specific external databases, RAG enables models to provide accurate and specialized responses in particular domains, such as legal or medical fields.



IV. Experimental setup:

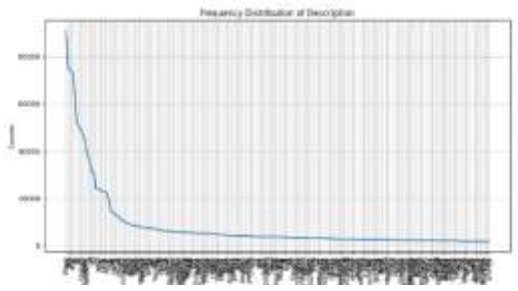
The project follows a structured workflow, integrating multiple stages to create a specialized model capable of addressing medical queries in the pregnancy domain. The flow of the project is outlined as follows:

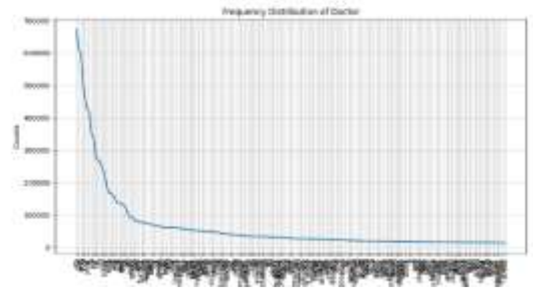
1. **Topic Modeling**: The initial step involved identifying potential areas of focus by applying topic modeling techniques on the dataset. From the extracted features, the top 10 topics were identified. Based on their frequency and relevance, we selected "pregnancy" as the specific domain for specialization.
2. **Data Retrieval**: To build a robust knowledge base, a curated dataset comprising 10 comprehensive books on pregnancy was collected in digital (PDF) format. This dataset became the foundation for developing a Retrieval-Augmented Generation (RAG) system, ensuring the inclusion of high-quality, domain-specific information.

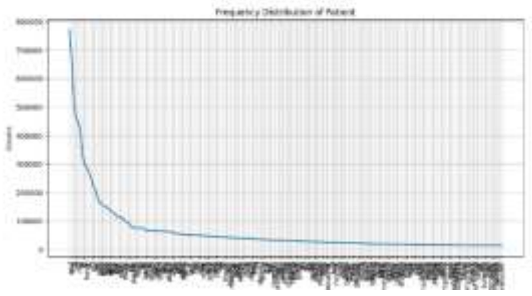
3. **Fine-Tuning the LLM**: The Llama 3.2 (1B parameters) model was chosen for its balance between computational feasibility and performance. It was fine-tuned using the AI Medical Chatbot dataset, which includes patient queries and doctor responses, aligning the model with medical conversational tasks. The training process involved experimenting with various optimization strategies, including techniques like gradient accumulation, LoRA (Low-Rank Adaptation), and prompt formatting.
4. **Retrieval-Augmented Generation (RAG)**: After fine-tuning, the model was integrated with a RAG framework, enabling it to access external knowledge and generate precise, contextually relevant responses. The system emulates a virtual doctor specializing in gynecology, specifically pregnancy-related queries.

1. Topic Modelling

1.1. Custom Stopwords List for Improved Topic Modeling







Before performing topic modeling, we created a custom stopwords list to enhance the quality and relevance of the generated topics. The goal was to better understand the context within the Description, Patient, and Doctor fields by removing common but contextually irrelevant words.

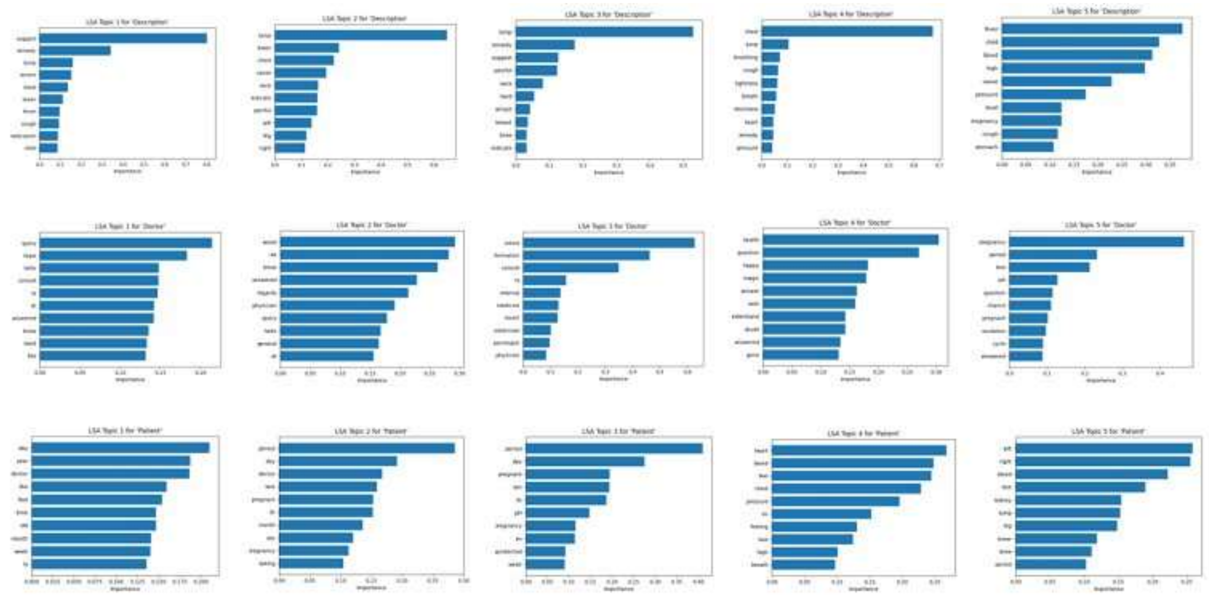
We started with the default NLTK stopwords list, which contains generic stopwords such as "the," "is," "and," and "in." However, these default stopwords are often insufficient when working with domain-specific datasets like medical queries. To address this, we analyzed the dataset to identify additional high-frequency words that did not contribute meaningfully to the context of the medical interactions. These included terms like "doctor," "hi," "thank you," and "please."

We created separate stopwords lists for each feature (Description, Patient, and Doctor) to ensure we captured the unique context of each field. For example:
In the Patient field, terms like "I," "my," and "me" were frequent but uninformative. In the Doctor field, phrases like "Regards," "consult," and "follow-up" were commonly used but added little to the modeling process.

Once the custom stopwords list was finalized, we applied these filters during the preprocessing phase to remove unwanted words before Latent Semantic Analysis (LSA). This step significantly improved the coherence of the topics generated by focusing on meaningful medical terms and patient concerns.

By filtering out these stopwords, the topic modeling process was able to uncover more relevant and insightful themes, providing a clearer understanding of the interactions between patients and doctors.

1.2. Understanding the topic on all context



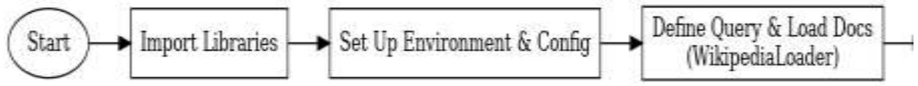
The Latent Semantic Analysis (LSA) of the top topics from each column — **Description**, **Patient**, and **Doctor** — revealed a predominant context centered around **Pregnancy**. In the **Description** field, the most common concerns included pregnancy-related symptoms, complications, and questions about prenatal care. Patients frequently described issues such as **fever, swelling, and continuous pain** in the context of pregnancy. The **Patient** field expanded on these concerns with detailed narratives mentioning **gestational age, medical history, and symptoms experienced during pregnancy**. These entries often reflected anxiety about the health of the fetus or the outcome of the pregnancy.

In the **Doctor** field, responses provided advice, diagnosis, and recommendations for prenatal care, addressing topics like **sepsis during pregnancy, fever management, and potential risks** associated with various symptoms. Doctors often suggested further **investigations, follow-ups, or consultations with specialists like obstetricians**. This recurring theme across all columns highlights that a significant portion of the dataset revolves around pregnancy-related health issues.

2. Data Retrieval

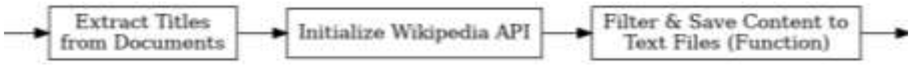
- **Wikipedia**

Step 1: Retrieves up to 50 documents related to the query "pregnancy."



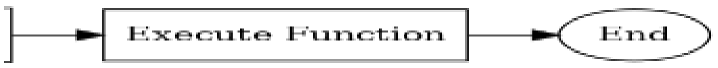
Start
Import Libraries
Set Up Environment & Config
Define Query & Load Docs (WikipediaLoader)

Step 2: Collects titles of the retrieved documents for processing.



Extract Titles from Documents
Initialize Wikipedia API
Filter & Save Content to Text Files (Function)

Step 3: Saves each Wikipedia page's relevant content



The process illustrated focuses on retrieving relevant Wikipedia content for the query “pregnancy.” It is divided into three key steps to systematically gather, process, and save information.

Step 1: Document Retrieval

The process starts by importing the necessary libraries and setting up the environment and configuration. A query for “pregnancy” is defined, and the WikipediaLoader is used to fetch up to 50 relevant documents from Wikipedia. This step ensures that a large pool of related articles is gathered for comprehensive analysis.

Step 2: Title Collection and Filtering

The titles of the retrieved documents are extracted for further processing. The Wikipedia API is initialized to streamline the retrieval process. A filtering function is applied to ensure only relevant content is saved. The filtered content is stored in text files, making it easier to handle for subsequent analysis.

Step 3: Content Saving

Finally, the function to execute the process is run, saving each Wikipedia page’s relevant content. The process ends after successfully extracting and storing the necessary information.

- **Books & Encyclopedias**

We identified 10 books which are great books for doctors and for new expectant mothers as our source. I used LangChain's PDFPlumber Document Loader for efficient data extraction from authoritative books on pregnancy and gynecology.

Process:

- **Document Loading**: Using the PDFPlumber Document Loader from LangChain, we loaded the PDF versions of the books, effectively handling complex document structures like headers and multi-column layouts.

- **Text Extraction:** The loader extracted text page by page, accurately capturing content such as paragraphs and headings while ignoring non-informative elements like page numbers and decorative elements.

The extracted text was stored as raw.txt files for further processing and embedding generation.

3. Fine-tuning LLM

Fine-tuning a large language model like Llama 3.2 (1B parameters) for domain-specific applications posed several challenges, particularly in optimizing training efficiency and ensuring robust performance. Below, we outline the steps taken to fine-tune the model, address encountered issues, and refine its performance to meet project objectives.

3.1. Reducing training time

The initial training attempt for a single epoch required over 24 hours, a timeframe that was impractical given resource constraints. To reduce this, the following strategies were implemented:

1. **Gradient Accumulation:** By accumulating gradients over 8 steps, the training time was marginally reduced, but the improvement was insufficient for practical purposes.
2. **Layer Freezing:** A more impactful solution was to freeze a portion of the model's layers. Initial experiments freezing the first 10 layers reduced training time to 23 hours. Scaling this approach, 75% of the layers were frozen.

This reduced the training time to 4.5 hours per epoch, a significant improvement. However, this led to the loss diverging to NaN during the 1st few seconds of training. Attempts to mitigate this by applying gradient clipping (`torch.nn.utils.clip_grad_norm_`) were unsuccessful.

This was resolved by the implementation of **LoRA (Low-Rank Adaptation)**. LoRA enabled low-rank updates of model weights, maintaining stability while slightly increasing training time to 5 hours per epoch. The need for layer freezing was also eliminated with LoRA's adoption.

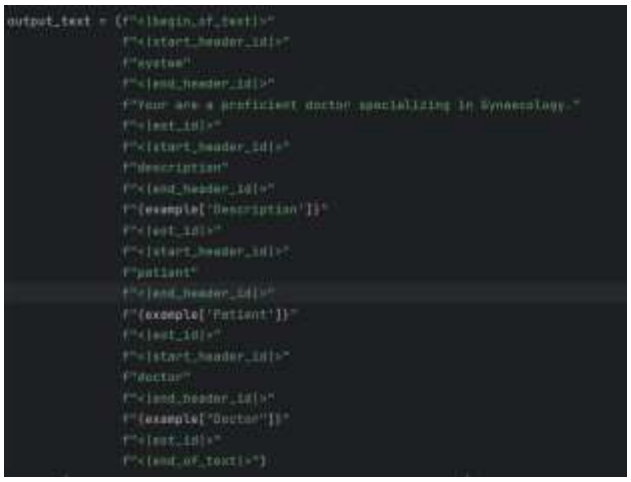
3.2. Hyperparameter Tuning

To optimize the model, hyperparameter tuning was conducted efficiently:

- A 0.2% sample of the data was used for initial testing, allowing rapid iteration with training times of 3 minutes per epoch.
- Configurations explored included:
    - **Low-rank dimension:** 8, 10, 12, 16, 20
    - **LoRA alpha:** 16, 32
    - **LoRA dropout:** 0.1, 0.2, 0.5
    - **Learning rates:** 1e-5, 2e-5, 2e-4
    - **Epochs:** Up to 50 for sampled data.

3.3. Prompt formatting

Earlier, we talked about instruction tuning. We have somewhat applied its concept in our fine tuning process. Now what we have done, is in terms of instructions, prompt formatted our data and injected it to our input. The process typically involves presenting the model with a prompt or instruction, followed by relevant context, and training it to generate appropriate responses. By exposing models to a wide array of instructions during training, they learn to interpret and

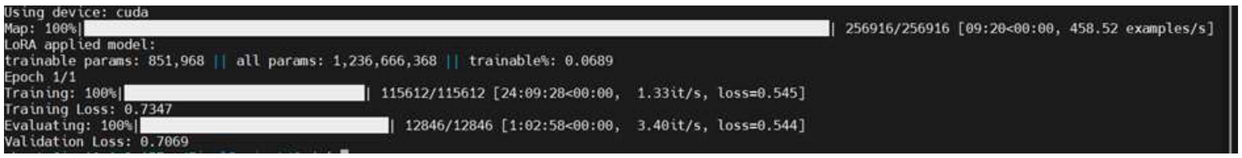


```
output_text = (f"<|begin_of_text|>"
               f"<|start_header_id|>"
               f"system"
               f"<|end_header_id|>"
               f"Your are a proficient doctor specializing in Gynaecology."
               f"<|eot_id|>"
               f"<|start_header_id|>"
               f"description"
               f"<|end_header_id|>"
               f"{example['Description']}"
               f"<|eot_id|>"
               f"<|start_header_id|>"
               f"patient"
               f"<|end_header_id|>"
               f"{example['Patient']}"
               f"<|eot_id|>"
               f"<|start_header_id|>"
               f"doctor"
               f"<|end_header_id|>"
               f"{example["Doctor"]}"
               f"<|eot_id|>"
               f"<|end_of_text|>")
```

execute new tasks more proficiently, even in zero-shot scenarios where they haven't been explicitly trained on those tasks. This method has been shown to improve performance across multiple downstream tasks, including inference and translation, sometimes surpassing models specifically trained for those tasks.Above is the code for prompt formatted fine tuning.

3.4. Evaluation and model selection

Loss and perplexity were monitored during training as key metrics. Early experiments on the full dataset plateaued at a loss of ~0.74 and perplexity ~2.1.



```
Using device: cuda
Map: 100%|████████████████████████████████████████████████████████| 256916/256916 [09:20<00:00, 458.52 examples/s]
LoRA applied model:
trainable params: 851,968 || all params: 1,236,666,368 || trainable%: 0.0689
Epoch 1/1
Training: 100%|██████████████████| 115612/115612 [24:09:28<00:00,  1.33it/s, loss=0.545]
Training Loss: 0.7347
Evaluating: 100%|██████████████████| 12846/12846 [1:02:58<00:00,  3.40it/s, loss=0.544]
Validation Loss: 0.7069
```

Increasing the sample size to 20% reduced the loss to 0.57 but offered diminishing returns beyond 3 epochs. Ultimately, a configuration yielding a loss of 0.4843 and perplexity of 1.6230 was identified.

The configuration that achieved this was:

- **Low-rank dimension:** 16
- **LoRA alpha:** 32
- **LoRA dropout:** 0.1
- **Learning rates:** 2e-5
- **Epochs:** 10
- **Dataset size:** 50% of the data

However, inference testing revealed suboptimal results, with the model generating nonsensical outputs. To address this, **prompt-formatted data** was injected during training, improving inference performance despite higher loss values. **Manual testing highlighted that the model trained for 5 epochs performed the best.**

Finally, adjusting the temperature parameter during inference to 0.6 produced the most coherent and contextually appropriate responses, underscoring the importance of tuning both training and inference parameters.

4. Retrieval Augmented Generation (RAG)

In our Retrieval-Augmented Generation (RAG) system, we employ several key components to effectively process user queries and generate accurate responses:

- Vector Store: We utilize Pinecone as our vector database to store and retrieve embeddings. Pinecone offers a scalable and efficient platform for managing high-dimensional vectors, enabling rapid similarity searches essential for our system's performance.
- Embedding Model: After evaluating multiple models, we selected the UAE-Large-V1 model from Hugging Face. This universal English sentence embedding model has achieved state-of-the-art performance on the MTEB Leaderboard, with an average score of 64.64

Hugging Face. Its effectiveness in capturing semantic meaning enhances the relevance of the context retrieved from our vector store.

Process Overview:

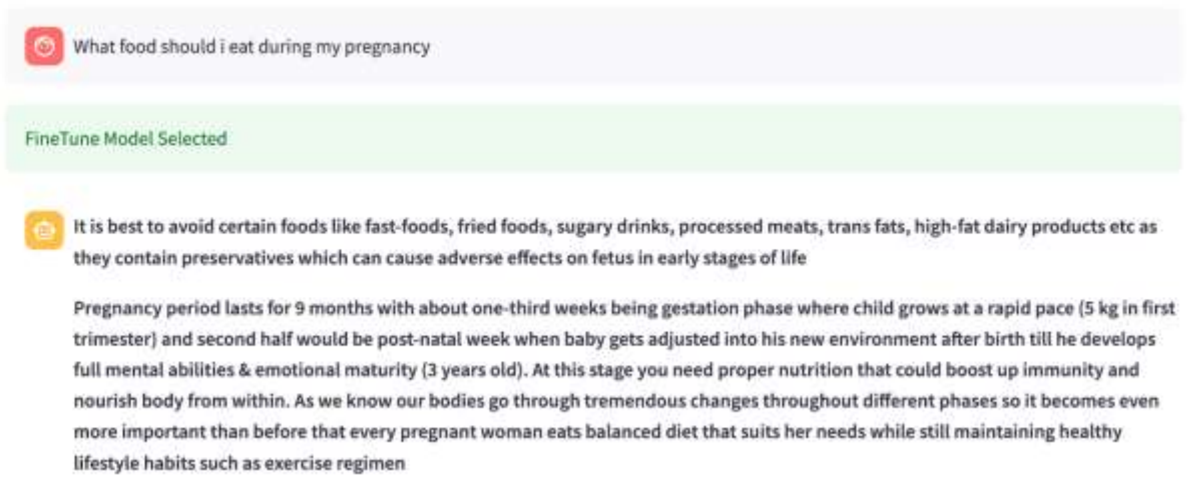
- Query Embedding: When a user submits a query, we generate its embedding using the UAE-Large-V1 model.
- Context Retrieval: We search our Pinecone vector index using cosine similarity to find contexts that closely match the query embedding.
- Response Generation: The retrieved context is combined with the original query and provided as input to our fine-tuned model, which generates the final response.
- This architecture ensures that our system delivers precise and contextually relevant answers by integrating advanced embedding techniques with efficient vector search capabilities.

We used loss, perplexity score to track the quality of our model. But that wasn't enough, so ultimately we decided to manually evaluate responses and chose our best model accordingly.

V. Results:

Question 1: What food should I eat during my pregnancy?

Here's a look at the output of our fine-tuned model -



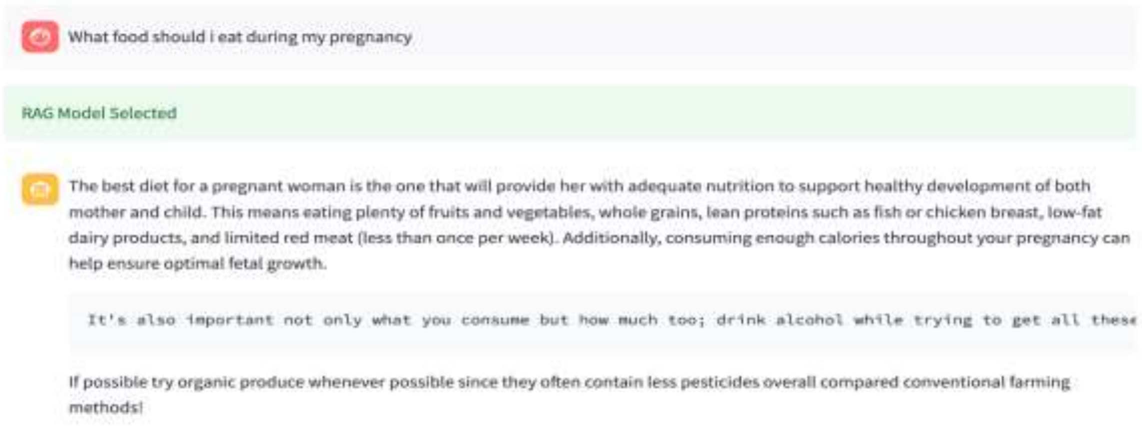
What food should i eat during my pregnancy

FineTune Model Selected

It is best to avoid certain foods like fast-foods, fried foods, sugary drinks, processed meats, trans fats, high-fat dairy products etc as they contain preservatives which can cause adverse effects on fetus in early stages of life

Pregnancy period lasts for 9 months with about one-third weeks being gestation phase where child grows at a rapid pace (5 kg in first trimester) and second half would be post-natal week when baby gets adjusted into his new environment after birth till he develops full mental abilities & emotional maturity (3 years old). At this stage you need proper nutrition that could boost up immunity and nourish body from within. As we know our bodies go through tremendous changes throughout different phases so it becomes even more important than before that every pregnant woman eats balanced diet that suits her needs while still maintaining healthy lifestyle habits such as exercise regimen

Now, checking the output for the same query but with RAG -


What food should i eat during my pregnancy

RAG Model Selected

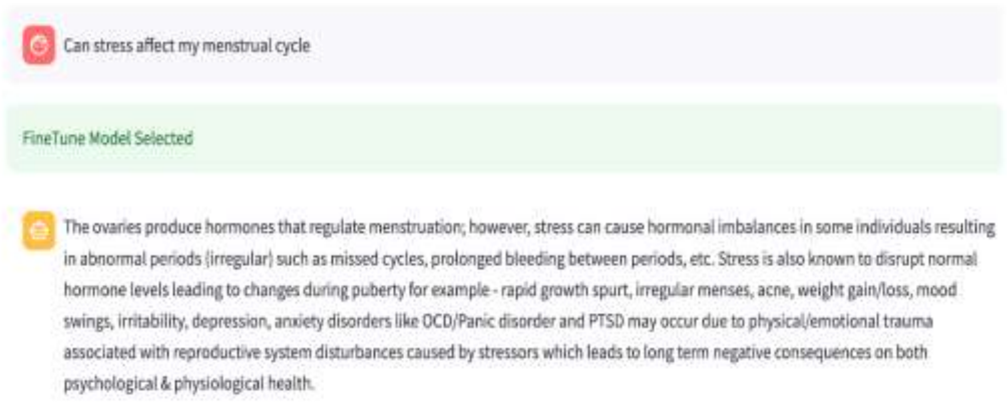
The best diet for a pregnant woman is the one that will provide her with adequate nutrition to support healthy development of both mother and child. This means eating plenty of fruits and vegetables, whole grains, lean proteins such as fish or chicken breast, low-fat dairy products, and limited red meat (less than once per week). Additionally, consuming enough calories throughout your pregnancy can help ensure optimal fetal growth.

```
It's also important not only what you consume but how much too; drink alcohol while trying to get all these
```

If possible try organic produce whenever possible since they often contain less pesticides overall compared conventional farming methods!

Question 2: Can stress affect my menstrual cycle?

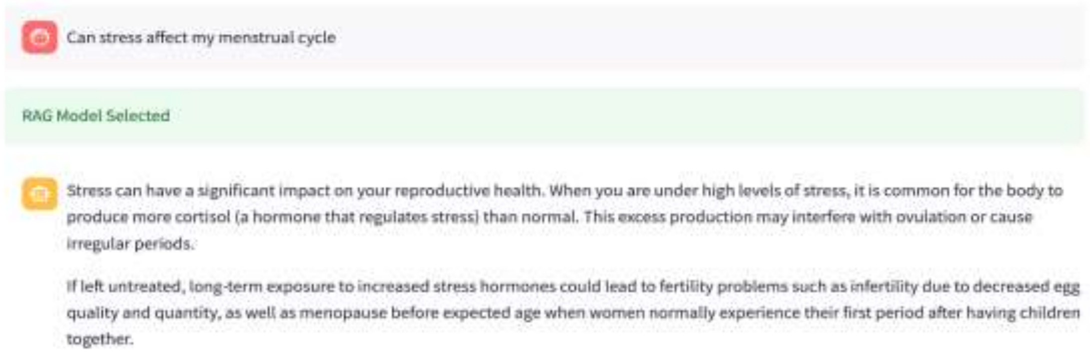
Fine-Tuned Result:


Can stress affect my menstrual cycle

FineTune Model Selected

The ovaries produce hormones that regulate menstruation; however, stress can cause hormonal imbalances in some individuals resulting in abnormal periods (irregular) such as missed cycles, prolonged bleeding between periods, etc. Stress is also known to disrupt normal hormone levels leading to changes during puberty for example - rapid growth spurt, irregular menses, acne, weight gain/loss, mood swings, irritability, depression, anxiety disorders like OCD/Panic disorder and PTSD may occur due to physical/emotional trauma associated with reproductive system disturbances caused by stressors which leads to long term negative consequences on both psychological & physiological health.

RAG Result


Can stress affect my menstrual cycle

RAG Model Selected

Stress can have a significant impact on your reproductive health. When you are under high levels of stress, it is common for the body to produce more cortisol (a hormone that regulates stress) than normal. This excess production may interfere with ovulation or cause irregular periods.

If left untreated, long-term exposure to increased stress hormones could lead to fertility problems such as infertility due to decreased egg quality and quantity, as well as menopause before expected age when women normally experience their first period after having children together.

VI. Summary:

Through this project, *LLAMA, MD: Specializing in Gynecology*, we demonstrated the potential of fine-tuned large language models to deliver domain-specific expertise, particularly in the critical

field of healthcare. By narrowing the focus to pregnancy-related topics in gynecology, we successfully adapted Llama 3.2 (1B parameters) into a specialized virtual assistant capable of providing contextually relevant and medically accurate responses.
The fine-tuning process tackled several challenges, including extensive training times, NaN loss issues, and suboptimal inference results. These were addressed through innovative solutions such as layer freezing, LoRA (Low-Rank Adaptation), and prompt formatting. Comprehensive hyperparameter tuning and targeted evaluation metrics ensured that the model was both efficient and effective in its specialized domain.
Integration of the fine-tuned model with a Retrieval-Augmented Generation (RAG) framework further enhanced its capabilities, enabling dynamic retrieval of knowledge from an expertly curated corpus. The resulting system demonstrates the feasibility of leveraging LLMs as reliable tools for specialized medical assistance, achieving an impressive balance between performance and resource constraints.
A lot more could be done with this to improve the model result. We could explore using larger models, such as a 3B or 7B variant, to enhance contextual understanding and response quality. Adopting **QLoRA (Quantized LoRA)** could further optimize training efficiency, enabling the use of these larger models within resource constraints. This project could be refined more and become capable for public use.

VII. References:

1. Yaadav, V. (2024, December 8). Exploring and building the LLaMA 3 Architecture : A Deep Dive into Components, Coding, and Inference Techniques. *Medium*. https://medium.com/@vi.ai_/exploring-and-building-the-llama-3-architecture-a-deep-dive-into-components-coding-and-43d4097cfbbb
2. Mavuduru, A. (2024, November 28). How to use Llama for text generation | Medium. *Medium*. https://amolmavuduru.medium.com/how-to-use-llama-for-text-generation-4f810d168199
3. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021, June 17). *LORA: Low-Rank adaptation of Large Language Models*. arXiv.org. https://arxiv.org/abs/2106.09685
4. Murel, J., PhD, & Noble, J. (2024b, December 5). What is latent semantic analysis? *IBM*. https://www.ibm.com/think/topics/latent-semantic-analysis#
5. *WhereIsAI/UAE-Large-V1 · Hugging face*. (2001, July 31). https://huggingface.co/WhereIsAI/UAE-Large-V1
6. *The vector database to build knowledgeable AI | Pinecone*. (n.d.). Pinecone Docs. https://docs.pinecone.io/guides/get-started/overview
7. *Text Splitters | 🔗 LangChain*. (n.d.). https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/

8. *torchtune: Easily fine-tune LLMs using PyTorch*. (n.d.). PyTorch. https://pytorch.org/blog/torchtune-fine-tune-llms/
9. *Enhancing AI Performance through Instruction Tuning*. (n.d.). Alberta Machine Intelligence Institute. https://www.amii.ca/updates-insights/instruction-tuning
10. *Build a basic LLM chat app - Streamlit Docs*. (n.d.). https://docs.streamlit.io/develop/tutorials/llms/build-conversational-apps