# Hallucination detection in Medical QA system

**1. Student Information**

* **Student Name:** Anjil Adhikari
* **Student ID :** s224008582
* **Date Submitted:**

Shape

**2. Project Introduction**

* **Title of the Project:** Hallucination detection in Medical QA system
* **What is the project about?**

The project Hallucination Detection in Medical QA System focuses on identifying false or misleading information ("hallucinations") generated by AI in response to medical questions. It aims to enhance the reliability and safety of medical question-answering systems by detecting responses that are not supported by real medical data or verified sources.

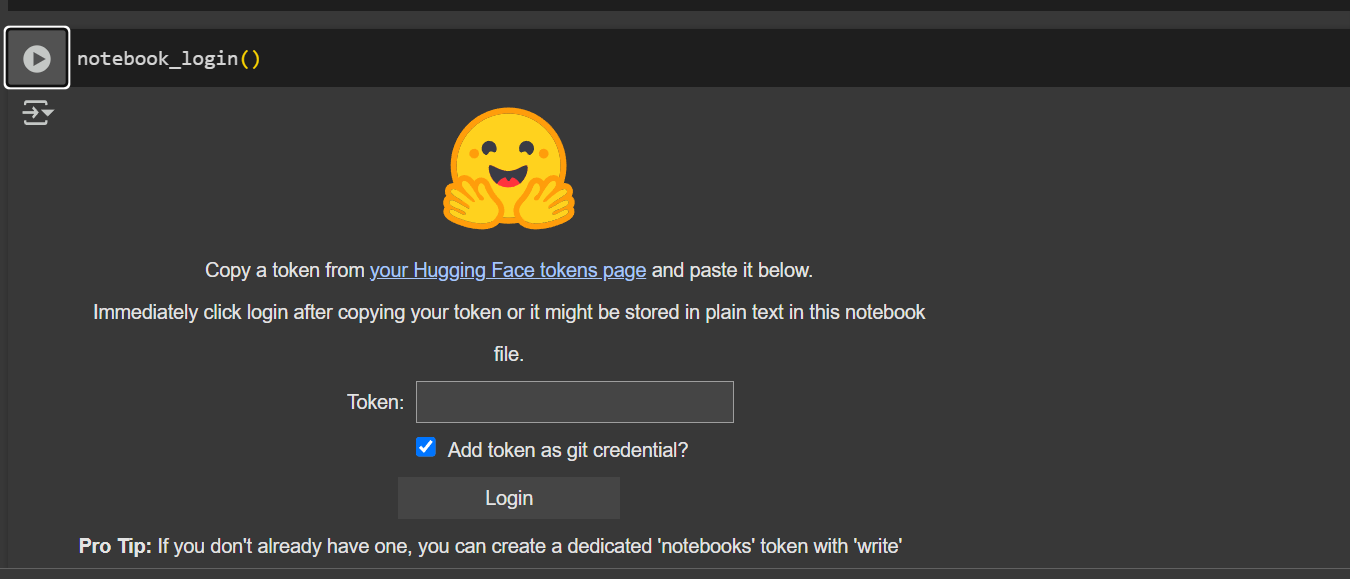
* **Why is this project important or useful?**

This project is important because hallucinations in medical QA systems can lead to incorrect diagnoses, inappropriate treatments, and serious health risks for patients. By detecting and reducing hallucinated responses, the project helps ensure that AI-generated answers are accurate, trustworthy, and aligned with medical evidence—ultimately supporting better clinical decisions and patient safety.

**Some Important selection before starting project**

* Model = LLAMA2 7B
* Dataset = MedQuAD (MEdical Question Answering Dataset)
* Machine = Free colab

At first, I have applied access for LLAMA2 7B in hugging face. Then imported that model using notebook login of hugging face

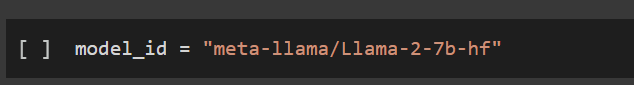


This is how i access by using access token.

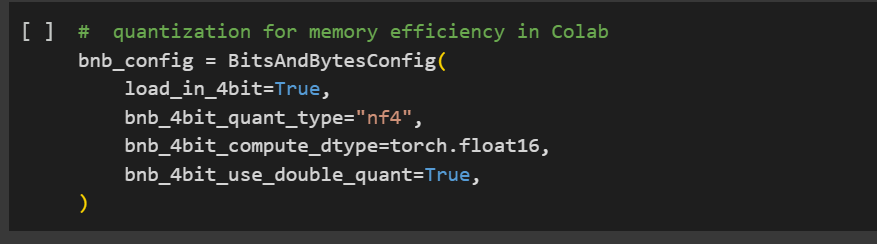
# Model and Dataset

Model

For model id i have given name as per in hugging face

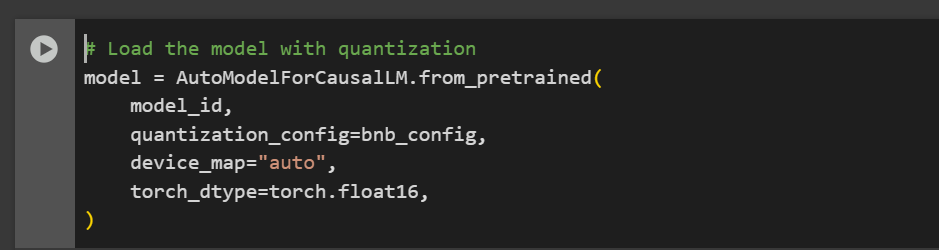


To make my Colab session more memory-efficient, I applied 4-bit quantization using BitsAndBytesConfig with NF4 quantization, float16 computation, and enabled double quantization for optimal performance.



Then loaded our model with this quantization.

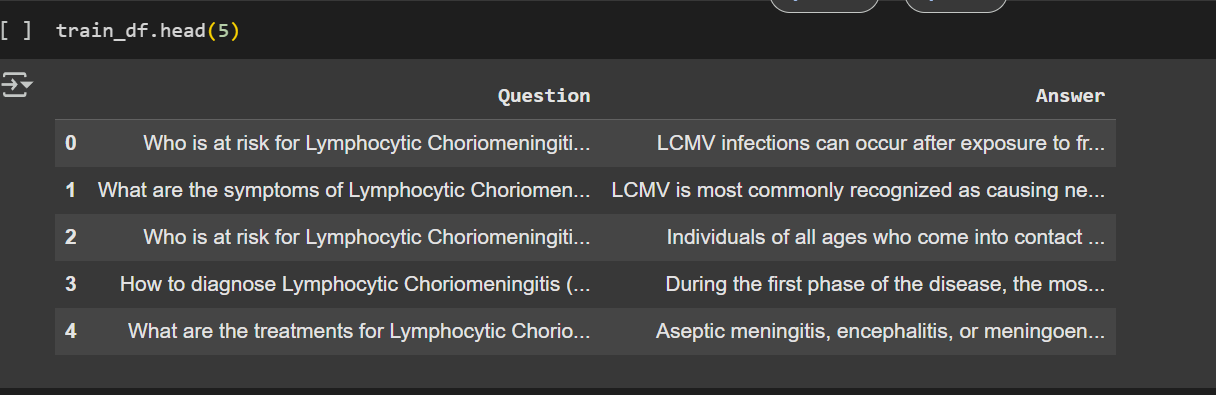
This technique is applied every time when model is used in our project.



**Dataset**

**Finetuning and RAG knowledge**

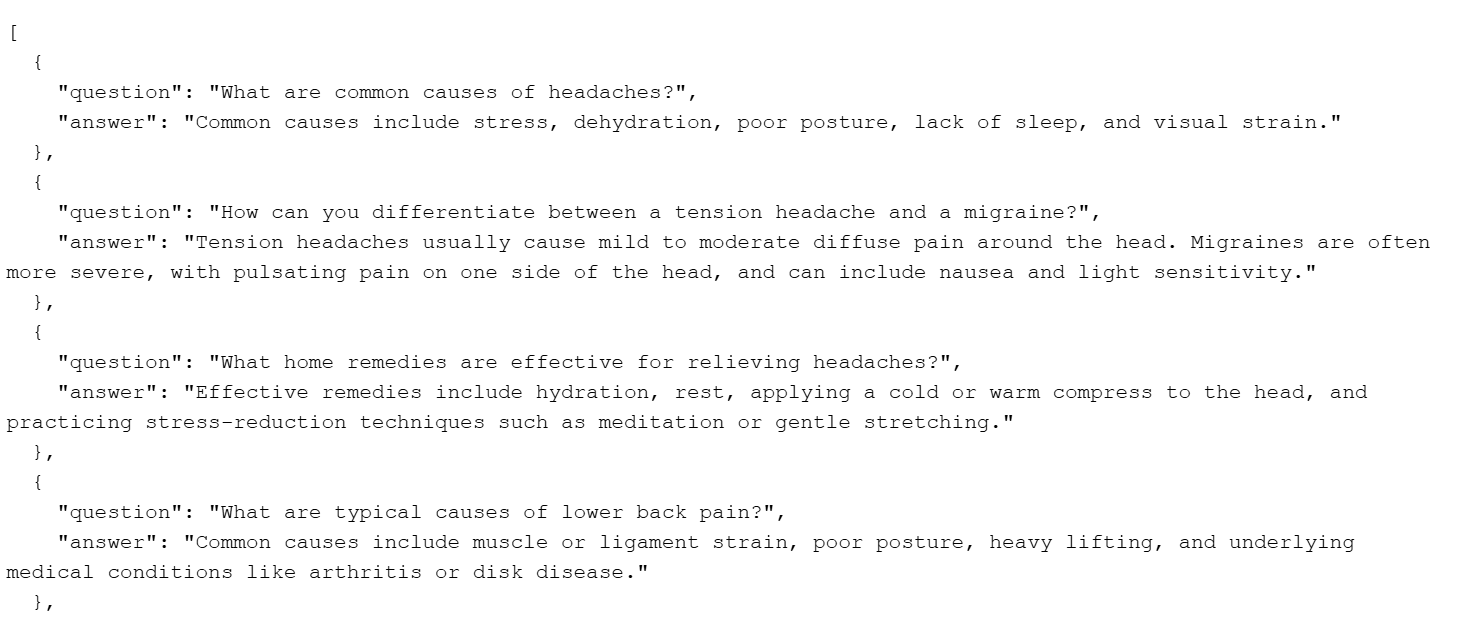
I have used MedQuAD (Medical Question Answering Dataset)



This dataset is converted into json format for using.

**Benchmarking dataset**

This is dataset is created and verified by medical professional. This dataset is use to evaluate model performance.

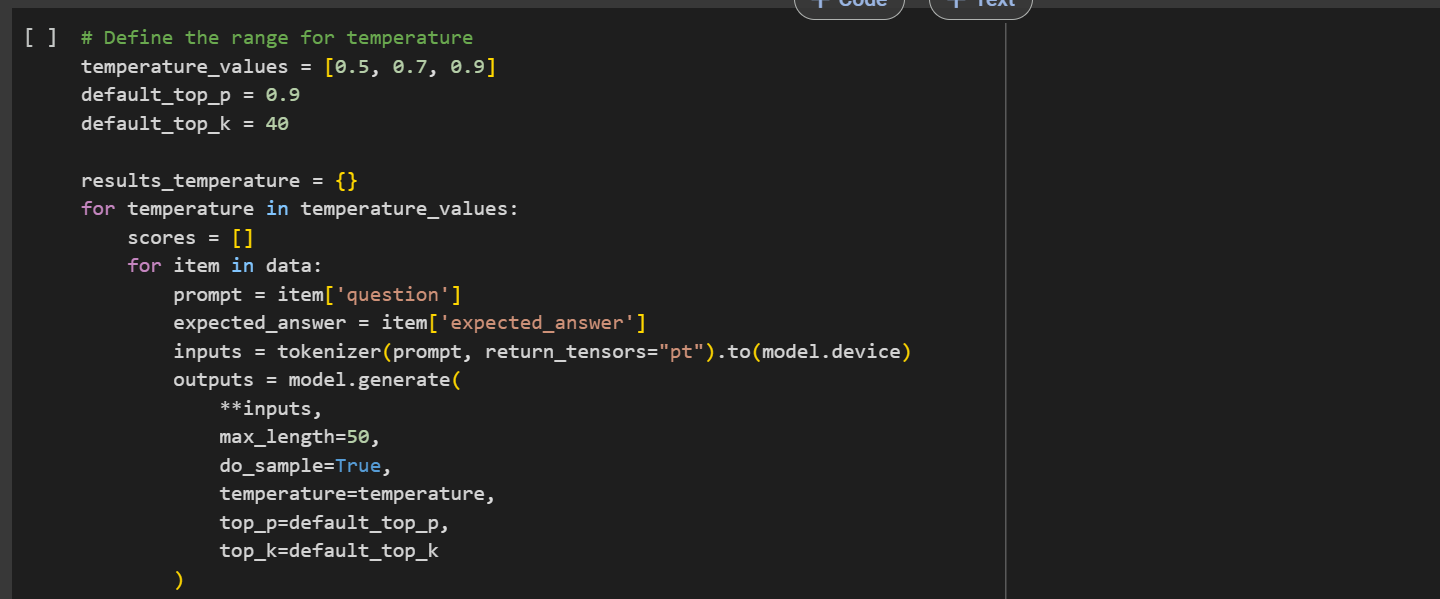


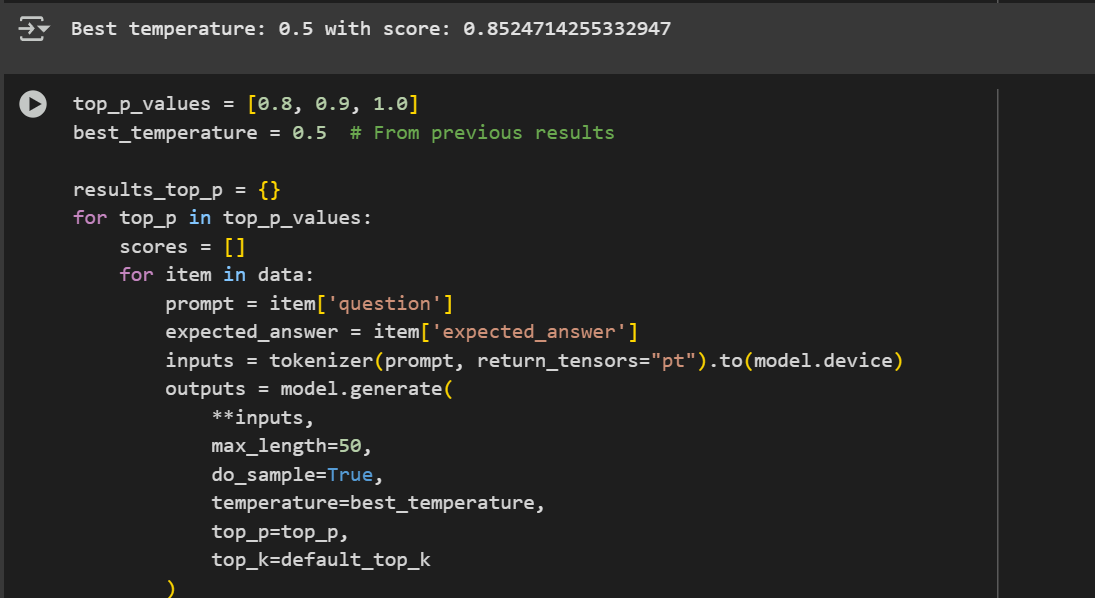
# Part 1

# Hyperparameter Tuning

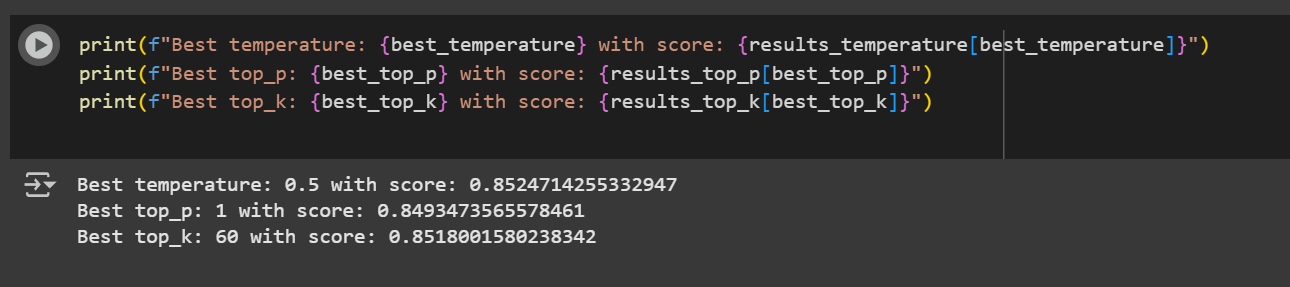
For hyperparameter tunning grid Search is because it systematically tests all combinations of specified parameters to find the best model performance using a chosen evaluation metric and easy to implement in colab aslo.

Due to lack of GPU resources, I couldn't tune all hyperparameters at once. Instead, I tuned them sequentially—first optimizing **temperature**, then **top\_p**, followed by **top\_k** values.

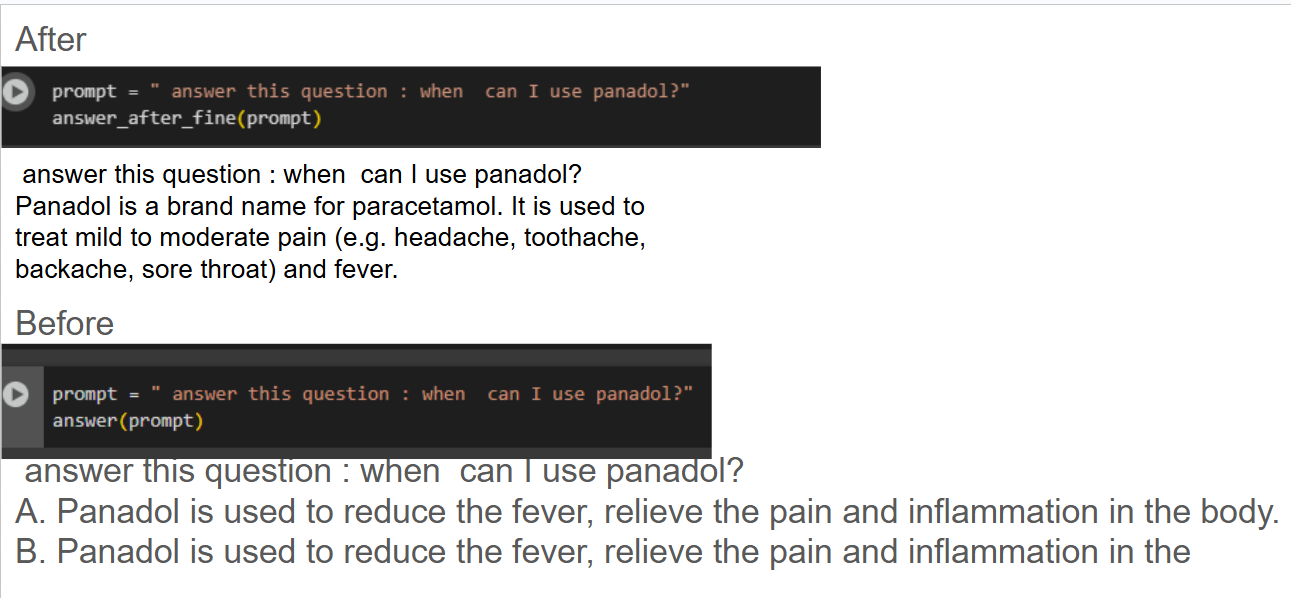




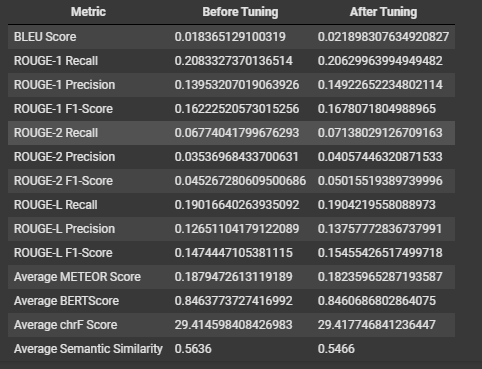
After all best tuned model has following paramnter



This is what before and after Hyperparamter tuning of model for same prompt. We can clearly observe the improvement in the model.



This is evaluation matrices calculated by using benchmarking dataset



There was no significant improvement in the score, possibly due to two main reasons:

1. **Limited benchmarking questions**: The dataset used for evaluation was small, which may not effectively reflect the model's true performance.
2. **Improper prompt formatting**: Prompts were not consistently structured (e.g., <s>[INST] <<SYS>> <</SYS>> question [/INST]), which might have impacted the model’s understanding.

I also attempted to increase the benchmarking dataset for better evaluation, but the **GPU limitations in Google Colab** restricted further experimentation.

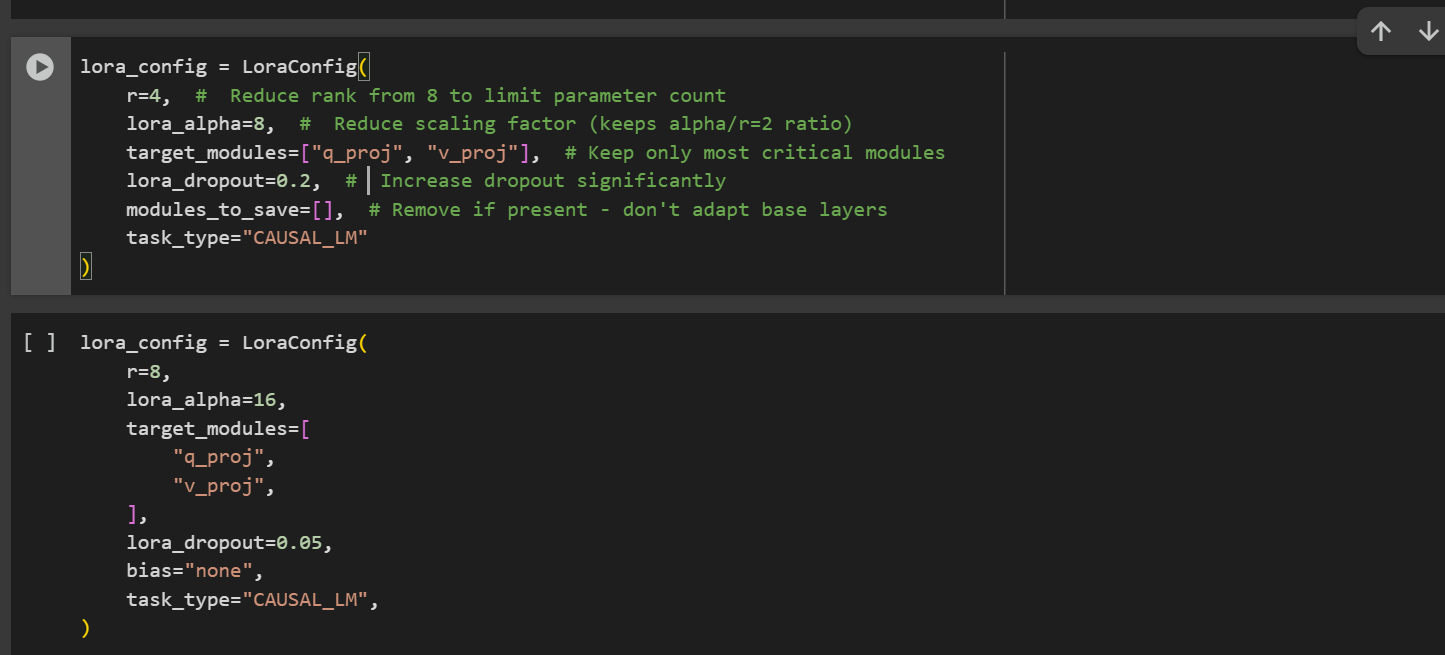
After this when using model, tune values are used.

# Part 2

# Finetuning

There are many options available for fine-tuning large language models, but I chose to use **LoRA (Low-Rank Adaptation)** due to its **efficiency and lower resource requirements**, making it suitable for my setup.

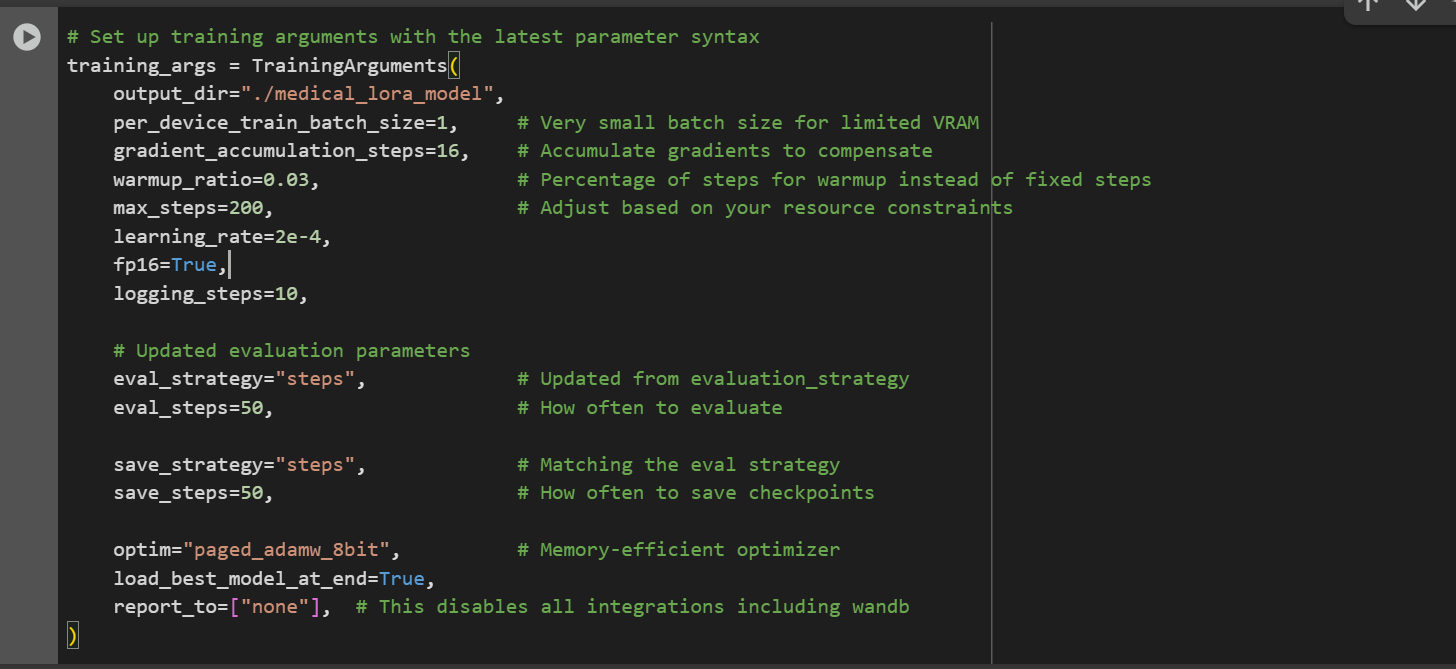
Lora configuration



In the comment of I have mention that I used lora for different setup thats gives me not good result and this one best parameter which show improvement in the model performance.

In Llma 2 7B model, all params are 6,742,609,920 but by using LoRA trainable parameters are 4,194,304.

This is training setup for fine tuning



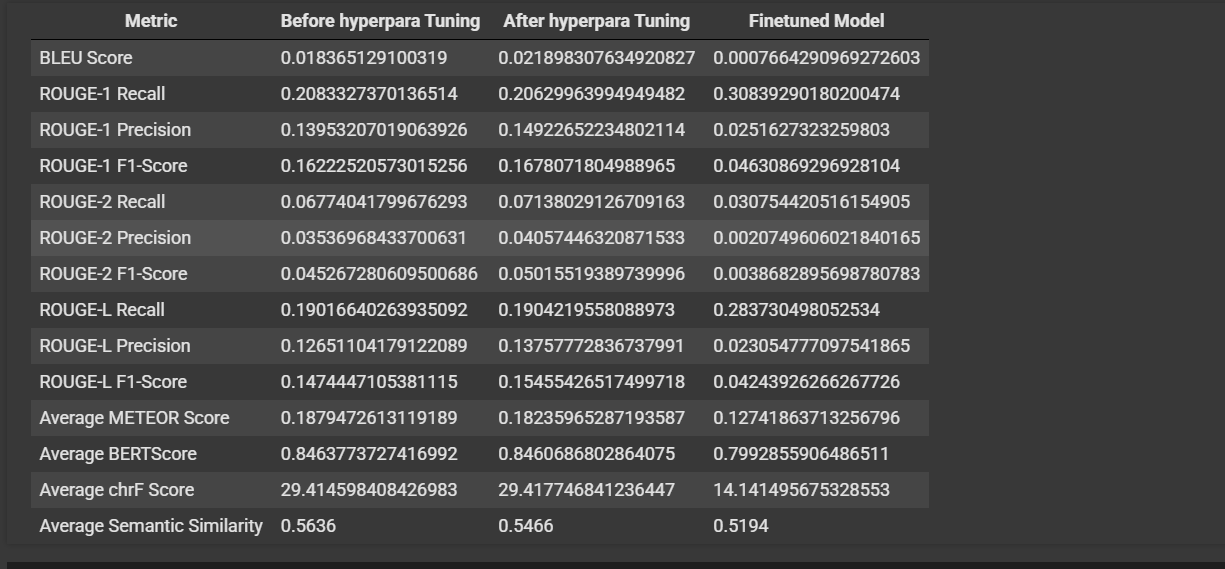
I used the latest TrainingArguments with settings optimized for low VRAM, such as a small batch size(1), gradient accumulation, 8-bit optimizer, and fp16 for faster and memory-efficient training—ideal for limited GPU resources.

Tokennizer and model is as mention above formate.

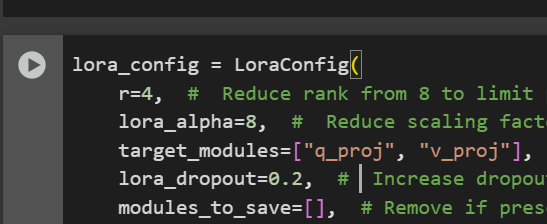
After this to evaluate our fine tuned model i have used same matrices and same benchmarking dataset. The result is given below:

This one 1st one with r=8 and Lora alpha=16

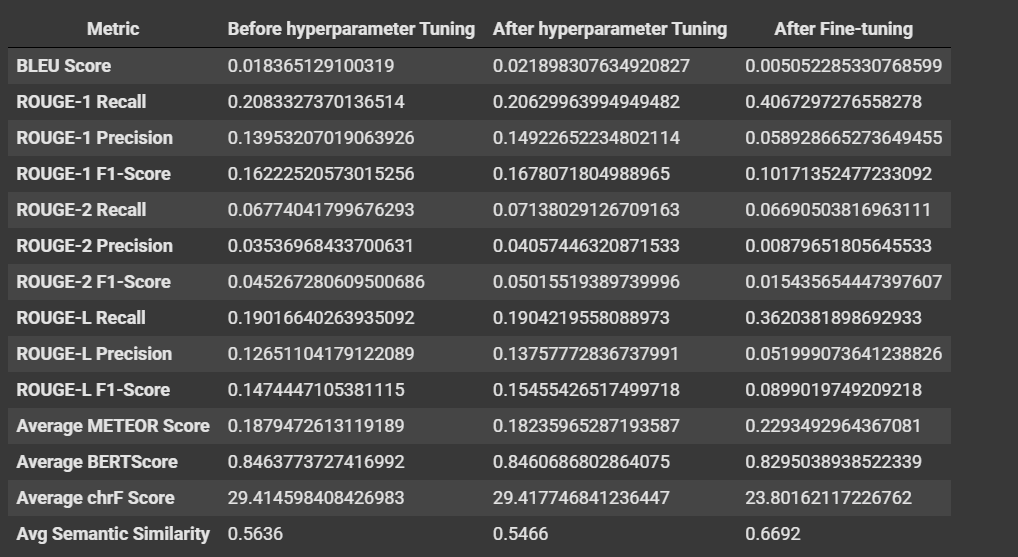
For this only 1k example is used from 16k example from our dataset



After this lora configuration is as shown



Below we can cleary observe in the improvement of performance of the model. But still not good enough.



We're still not seeing significant improvement, mainly due to:

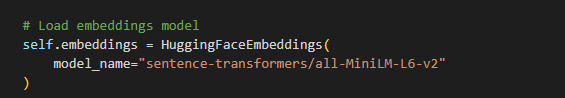
1. **Insufficient benchmarking and training datasets**, which limit the model's ability to learn and generalize.
2. **Hardware constraints**: Even adding a single example to training or benchmarking exceeds the capacity of the free Colab GPU, preventing further experimentation.

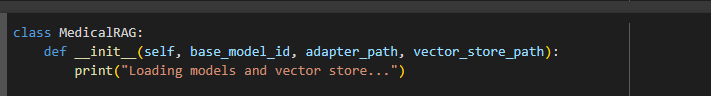
This is the most improved model so decided to use the model for later also.

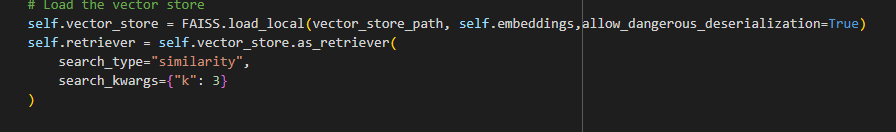
# Part 3

# Apply RAG

* **Selected 2,000 samples for RAG knowledge base** To keep memory usage low and match the same subset used during fine-tuning for consistency between training and retrieval.
* **Generated sentence embeddings using sentence-transformers/all-MiniLM-L6-v2** This lightweight embedding model balances performance and speed, making it ideal for resource-constrained environments Colab.

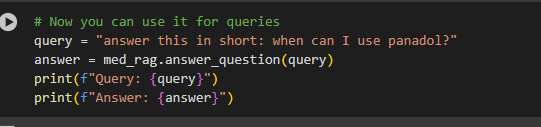


* **Stored embeddings in a FAISS vector database** FAISS enables fast and memory-efficient similarity search, which is crucial for real-time RAG retrieval.
* **Created the MedicalRAG class to encapsulate the full RAG workflow** Wrapping the pipeline in a class makes it reusable, maintainable, and easy to integrate into applications.
* **Implemented logic to load/ create the vector store inside the class** Ensures robustness—automatically builds the vector store if it’s missing, preventing errors on first-time use.
* **Configured a retriever using the FAISS store (as\_retriever)** This provides an abstraction for document retrieval, allowing easy tuning (like k=3 top matches) and integration with the generation step.



* **Loaded the LLaMA 2 7B model with LoRA adapter using 4-bit quantization** LoRA fine-tuning allows efficient adaptation to medical QA without retraining the whole model. 4-bit quantization reduces GPU memory usage significantly. Base model as described above.
* **Wrote the answer\_question() method to handle user queries end-to-end** This method performs retrieval, prompt construction, model inference, and response decoding—all in one place for streamlined answering.
* **Saved the RAG configuration using pickle** Avoids reinitializing everything manually; allows easy loading in future sessions without saving large models.

This is what output from RAGED model



*Query: answer this in short: when can I use panadol?*

*Answer: Answer: Panadol is an analgesic and antipyretic medicine. It contains paracetamol as its main ingredient. Panadol can be used for the relief of pain and fever. However, it should not be used in children under 12 years of age. It should also not be used in pregnant women. Panadol can be used to relieve pain associated with headaches, muscle aches, toothaches, backaches, menstrual cramps and arthritis. Panadol can also be used to reduce fever, although it is not as effective as aspirin. Panadol is also used to relieve pain associated with the common cold and flu. Panadol is available in various strengths. The 500 mg strength is usually recommended for pain relief and the 1000 mg strength is usually recommended for fever reduction. Panadol should be taken as directed. If you take more than the recommended dose, you may experience nausea, vomiting, diarrhea, stomach pain, and liver damage. If you experience any of these symptoms, seek medical attention immediately.*

*Panadol is an over-the-counter medicine. It can be purchased without a prescription at your local pharmacy. Panadol is safe to use when taken as directed. However, it should not be used in children under 12 years of age and should not be used by pregnant women. Panadol should be used with caution if you are taking any other medicines, as it may interact with them. If you are pregnant, you should avoid using Panadol, as it may harm the unborn child. Panadol should be stored at room temperature and away from direct sunlight. It should be kept out of the reach of children.*

*Panadol is an analgesic and antipyretic medicine. It contains paracetamol as its main ingredient. Panadol can be used for the relief of pain and fever. However, it should not be used in children under 12 years of age. It should also not be used in pregnant women. Panadol can be used to relieve pain associated with headaches, muscle aches, toothaches, backaches, menstrual cramps and arthritis*

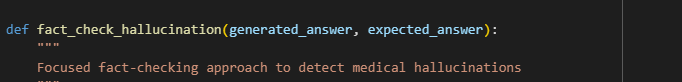
This is one the best answer.

# Part 4

# Detect hallucination

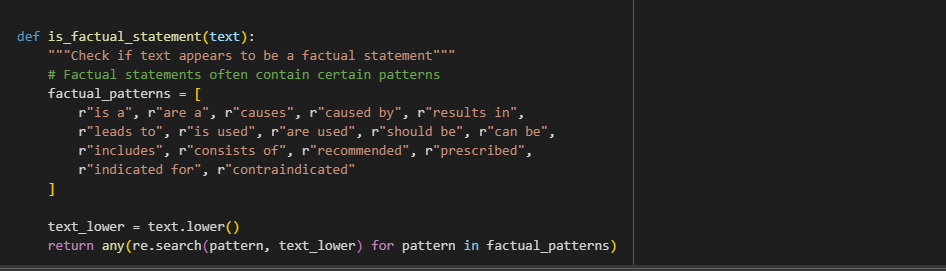
As mention same benchmarking dataset is used for detection of hallucination.

Then fact\_check\_hallucination() is implemented to detect hallucination

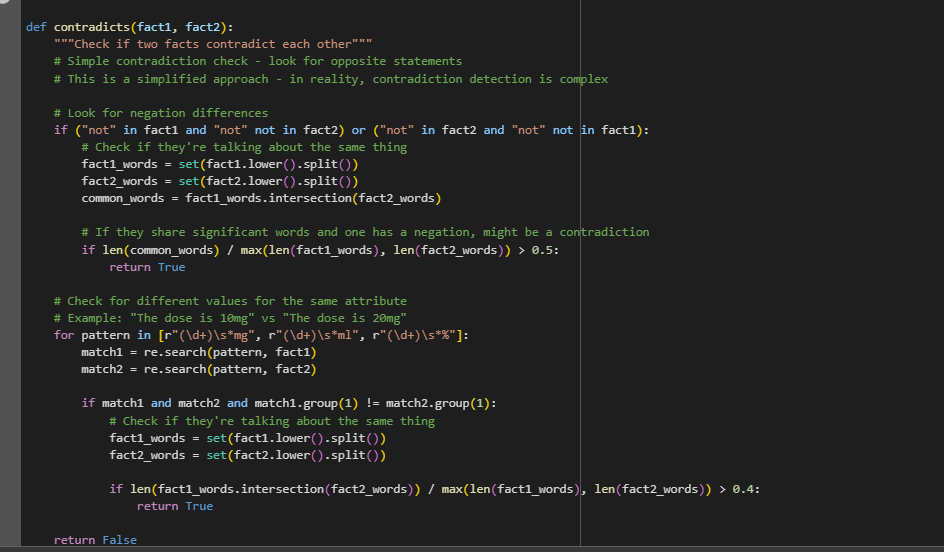


* Extract Medical Facts:
  + Uses extract\_medical\_facts() to parse both generated\_answer and expected\_answer into discrete facts.

This function used to extract facts



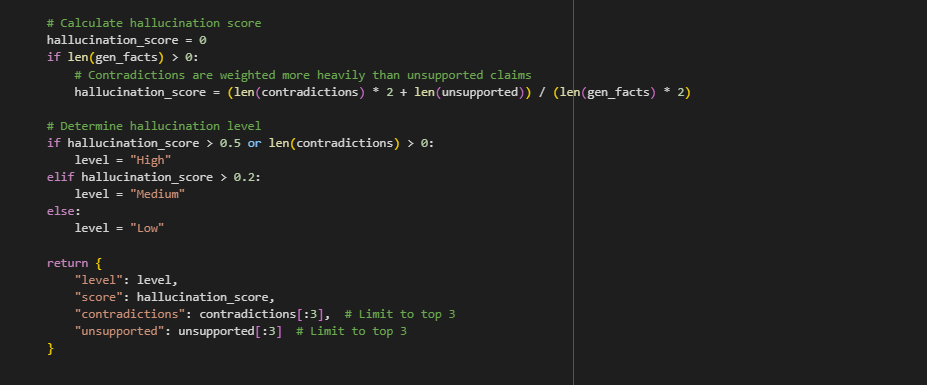
* Check for Issues:
  + Contradictions: Compares generated facts against reference facts using contradicts().
  + Unsupported Claims: Flags facts in the generated answer that lack support in the reference (using supports()).



* Score Hallucinations:
  + Scoring Formula:

(2 \* num\_contradictions + num\_unsupported) / (2 \* total\_generated\_facts)

* + Contradictions are weighted more heavily (2x) than unsupported claims.



* Assign Risk Level:
  + High: Score > 0.5 OR any contradictions.
  + Medium: Score > 0.2.
  + Low: Otherwise.

This is final result

