# Hallucination detection

# **1. Student Information**

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

Shape

**2. Project Introduction**

* **Title of the Project:** Hallucination Detection
* **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 questions. It aims to enhance the reliability and safety of question-answering systems by detecting responses that are not supported by real data or verified sources.

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

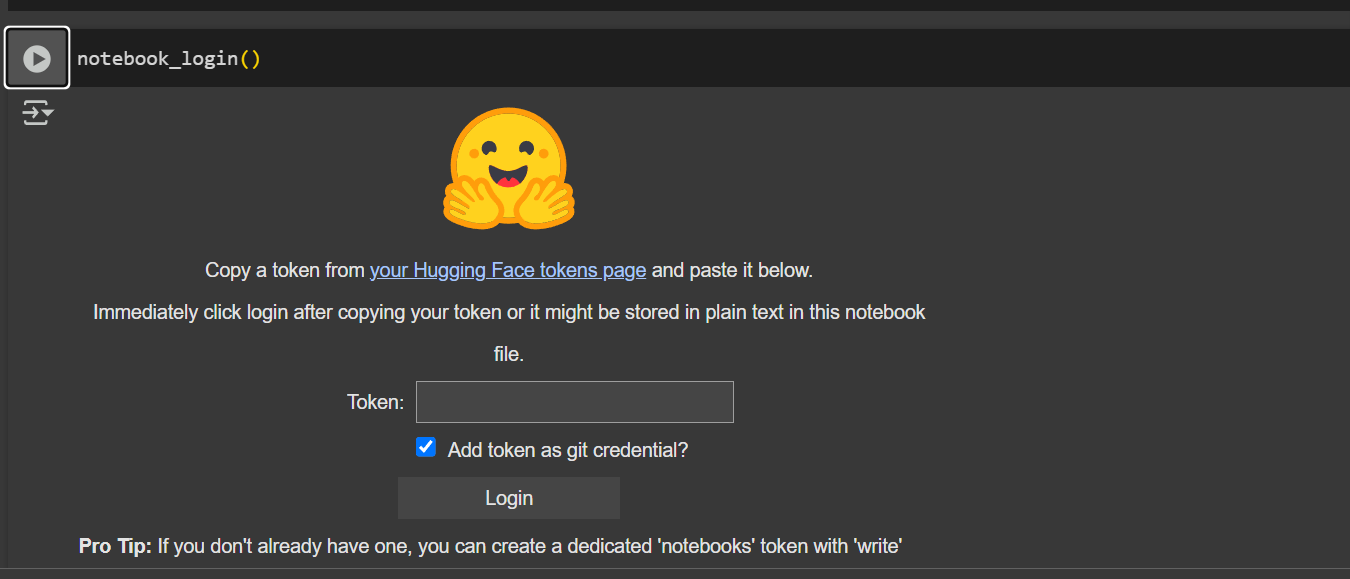
I focused on medical so.

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 applied access for LLAMA2 7B in Hugging Face. Then imported that model using the notebook login of Hugging Face

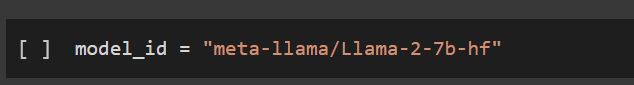


This is how I access by using an access token.

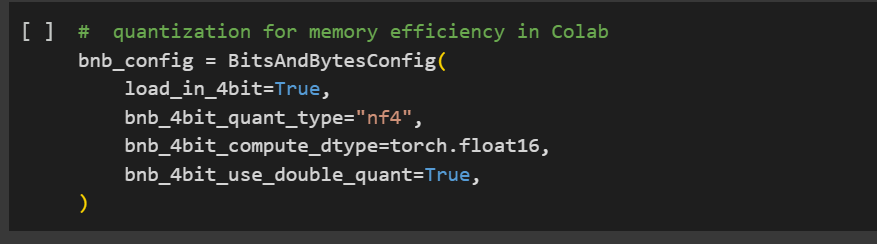
# Model and Dataset

Model

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

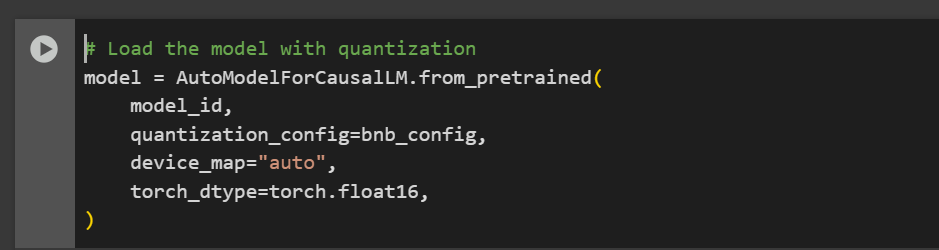


To make my Colab session more memory-efficient, I applied 4-bit quantization using BitsAndBytesConfig with NF4 quantization, float16 computation, and enabled double quantization for optimal performance.



Then loaded our model with this quantization.

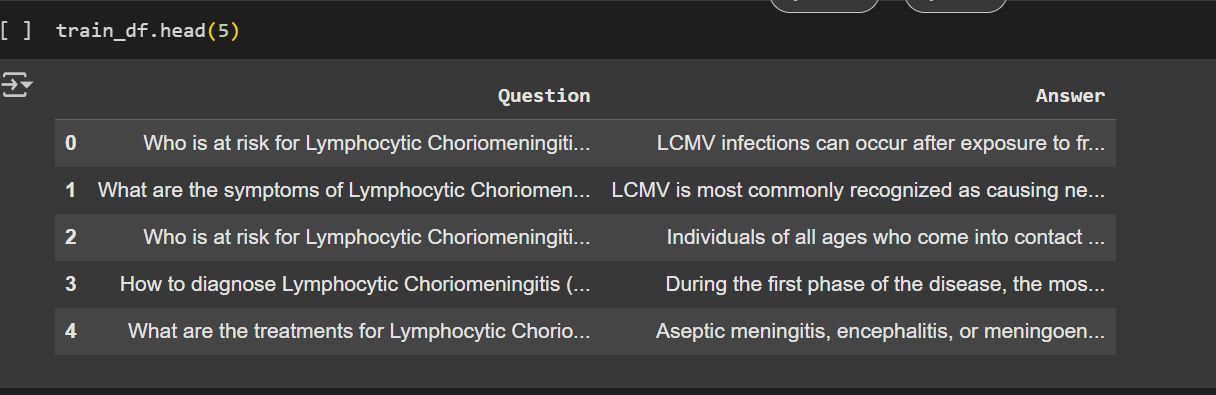
This technique is applied every time when model is used in our project.



**Dataset**

**Finetuning and RAG knowledge**

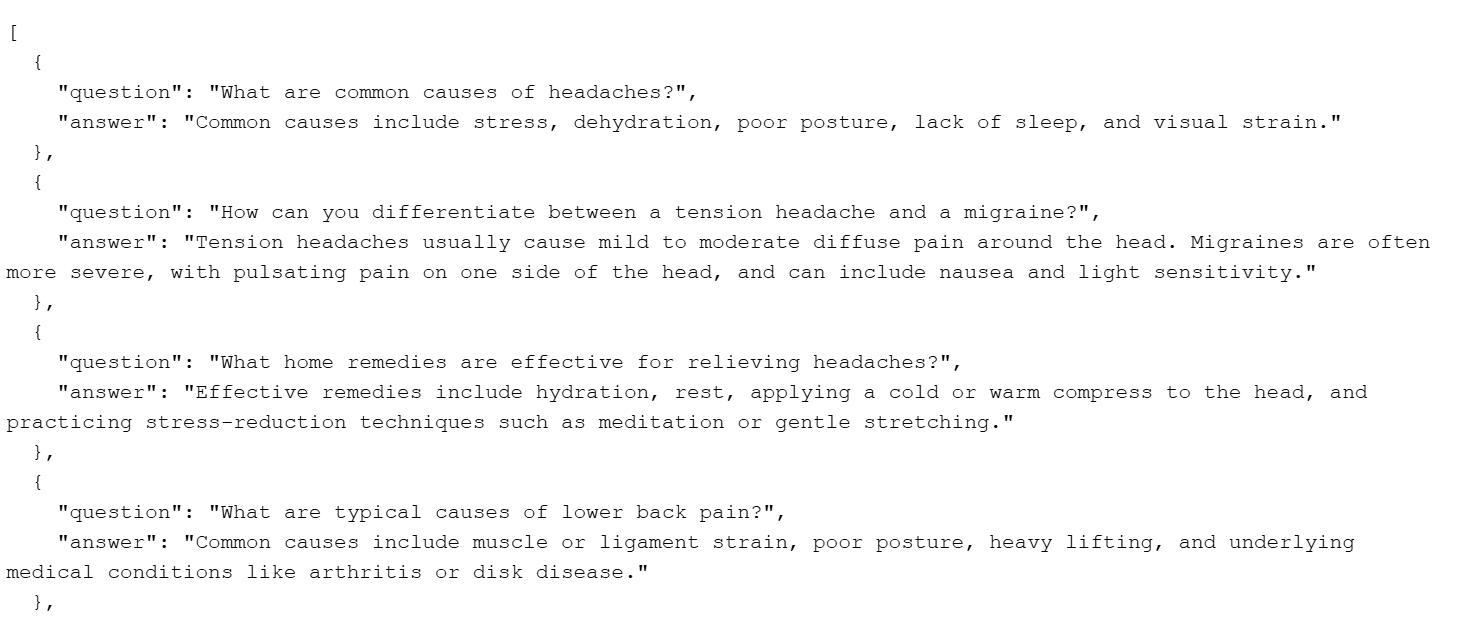
I have used MedQuAD (Medical Question Answering Dataset)



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

I have implemented a hallucination detection module based on **Natural Language Inference (NLI)**. The generated answers are **compared with a benchmarking dataset** that contains trusted standard answers. This ensures that detection is grounded in reliable references rather than arbitrary text.

#### **Detection Process**

* A custom function detect\_hallucinations splits each generated answer into sentences.
* Each sentence is paired with the corresponding standard answer and passed into an NLI model from Hugging Face (roberta-large-mnli).
* The model outputs a label (ENTAILMENT, NEUTRAL, CONTRADICTION) and a confidence score.
* Results are aggregated per question:  
  + **CONTRADICTION → Hallucination detected**
  + **NEUTRAL only → Potentially uncertain**
  + **ENTAILMENT only → No hallucination detected**

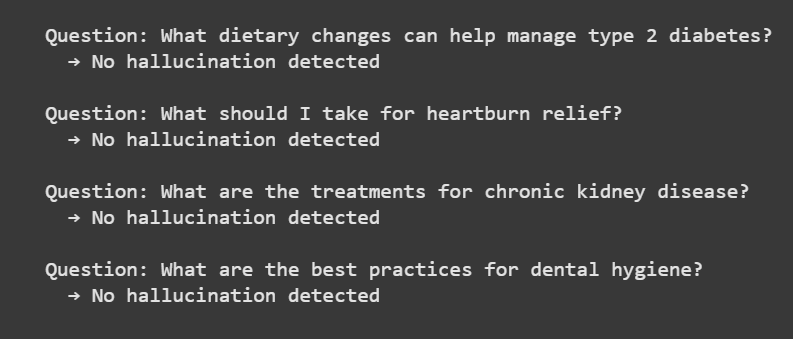
#### **Results**

* Sentence-level judgments (label + confidence) are printed for each question.
* Final overall assessments are stored in hallucination\_detections.json.
* This provides both **fine-grained evidence** and **high-level reliability checks** for generated answers against the benchmark.

#### **Limitation**

The current pipeline depends on the NLI model’s ability to process input within its **maximum token size (typically 512 tokens for RoBERTa)**. Long answers are truncated, leading to incomplete or biased detection results. This makes the method less reliable for very lengthy outputs without preprocessing or chunking.

This is what the result looks like



# Part 2

I applied hallucination detection using the **self-consistency approach**. The idea is to re-generate multiple answers for each question with varying randomness (temperature values) and then measure how consistent these generations are with the gold/benchmark answers using **cosine similarity from Sentence Transformers**.

#### At first, I did this

* I re-generated 5 answers per question at different temperatures: 0.5, 0.6, 0.7, 0.8, and 0.9.
* Each generated answer was compared to the standard (gold) answer.
* The average similarity per question and across all questions was calculated.
* Overall Average Across 10 Questions: 0.346
* Results showed that lower temperatures (0.5–0.6) gave more deterministic, higher-similarity outputs, while higher temperatures (0.8–0.9) reduced similarity due to increased variability.

#### **Then, I did this**

* I repeated the same process, but this time used a **benchmarking dataset** as the trusted reference.
* Again, 5 answers were generated for each question across the same temperature values.
* **Overall Average Across 60 Questions: 0.289**
* Similar trends appeared: lower temperatures improved similarity, while higher ones reduced it.

However, because the benchmarking dataset is larger and more diverse, the similarity values were slightly lower but more **robust and reliable** as an evaluation of hallucination detection.

#### **Conclusion**

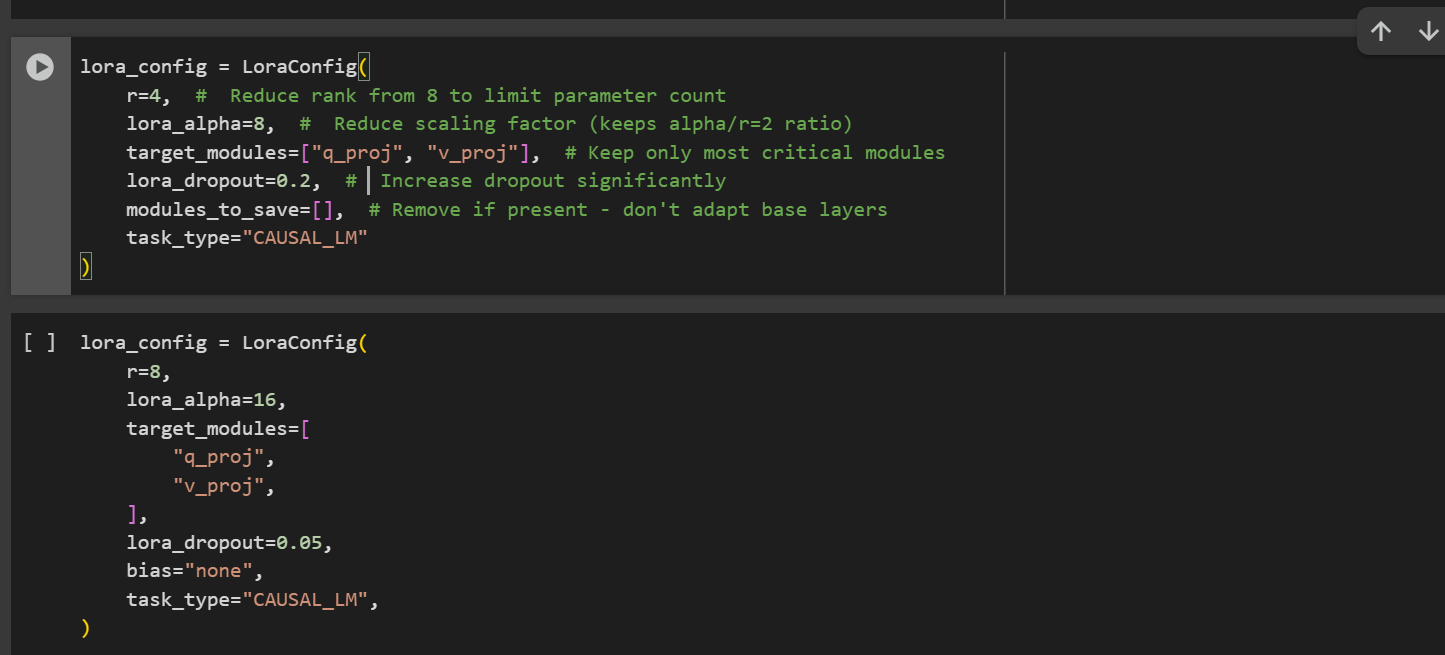
Both experiments confirm that **self-consistency can expose hallucinations by comparing multiple sampled outputs against references**. The smaller test (10 questions) gave a higher average similarity (0.346), while the larger benchmarking test (60 questions) showed a more realistic average (0.289). Using a benchmarking dataset makes the evaluation stronger, even though similarity scores drop, because it ensures the results are based on a broader and cleaner ground truth.

# Part 3

# 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.

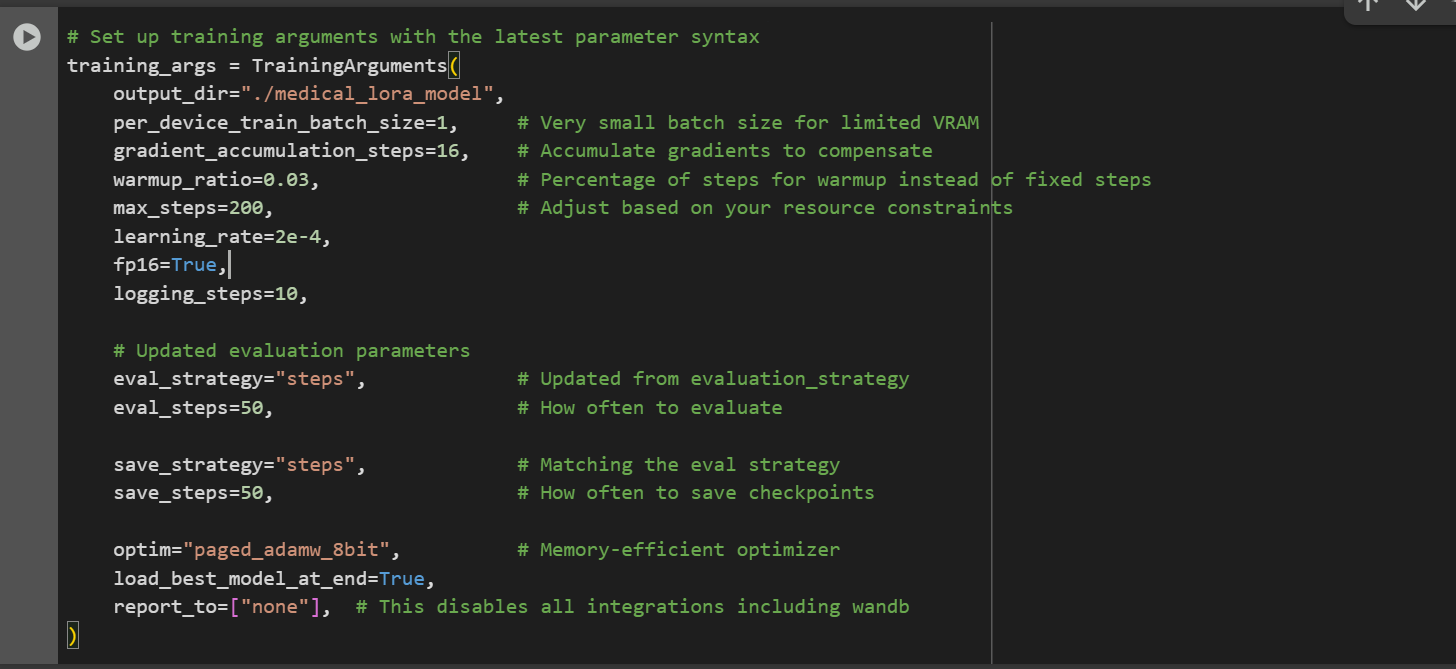
Lora configuration



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

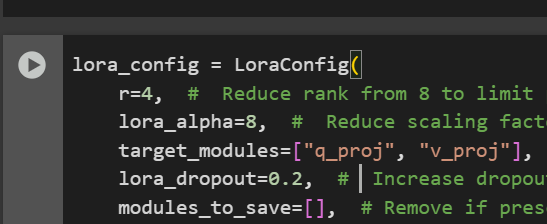


I used the latest TrainingArguments with settings optimized for low VRAM, such as a small batch size(1), gradient accumulation, 8-bit optimizer, and fp16 for faster and memory-efficient training—ideal for limited GPU resources.

Tokenizer and model are in the format mentioned above.

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

After this lora configuration is as shown



# 

# Part 4

I extended the hallucination detection study by fine-tuning the model and re-running the same self-consistency evaluation. This allowed me to directly measure how fine-tuning impacts semantic similarity and indirectly reduces hallucination.

* Without fine-tuning, I generated multiple answers per question at different temperatures (0.5, 0.6, 0.7, 0.8, 0.9) and compared them against gold references.
* The results showed modest similarity:  
  + Overall Average Across 10 Questions: 0.346
  + Overall Average Across 60 Benchmarking Questions: 0.289
* Lower temperatures produced slightly higher similarity, but overall the model struggled to stay consistent with the gold answers.

#### Then, I did this

* After fine-tuning, I repeated the same evaluation with the same setup.
* The results showed a significant jump in similarity scores:
  + Overall Average Across 10 Questions: 0.700
  + Overall Average Across 60 Benchmarking Questions: 0.585
* Fine-tuning not only improved alignment with gold answers but also made the model more stable across different temperatures.

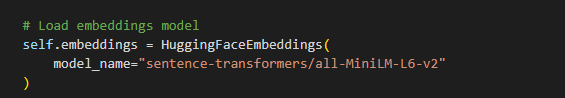
#### Conclusion

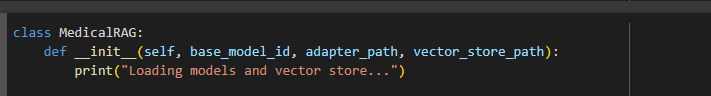
Fine-tuning more than doubled the similarity scores (from 0.346 → 0.700 on the small set, and 0.289 → 0.585 on the benchmark set). This substantial improvement indicates that fine-tuning indirectly reduces hallucinations, since the model’s answers are now much closer to the reference answers and more consistent across generations.

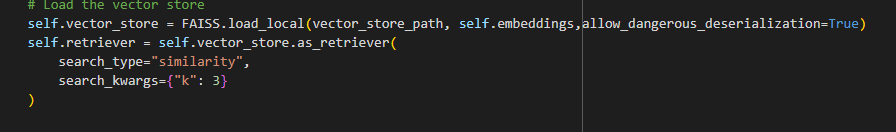
# Part 5

# 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.

# Part 6

### **RAG + Fact-Checking Hallucination Detection (Post Fine-Tune)**

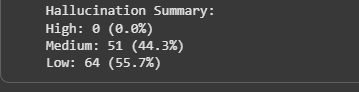
I extended the fine-tuned model with a RAG pipeline and then applied a new, rule-based fact-checking method to detect medical hallucinations. It extracts and checks factual medical statements with lightweight heuristics.

#### At first, I did this

* Built RAG on the fine-tuned model  
  + Base: meta-llama/Llama-2-7b-hf with your LoRA adapter /medical\_lora\_adapter
  + Vector store: FAISS loaded from /medical\_rag\_vectorstore\_2k, with a retriever (k=3).
  + Answering: answer\_question() retrieves top docs, composes a prompt with the retrieved context, and generates the response.
* Benchmarked the RAG model on your dataset and saved raw generations to:
  + /rag\_benchmark\_results\_115.json

#### Then, I did this

* Applied a custom fact-checking detector  
  + extract\_medical\_facts() pulls likely factual sentences (heuristics: medical terms and factual patterns like “is used”, “should be”, “dose”, etc.).
  + supports(f1,f2) marks support via word-overlap ratio (>0.5).
  + contradicts(f1,f2) flags contradictions via negation mismatch (same content with/without “not”) and numeric conflicts (e.g., “10 mg” vs “20 mg” within similar word context).
  + A hallucination score is computed from unsupported/contradictory facts and mapped to levels: Low / Medium / High.
* Summary printed by the notebook (on 115 items):  
  + High: 0 (0.0%)
  + Medium: 51 (44.3%)
  + Low: 64 (55.7%)



#### Conclusion

* The RAG stage grounds the fine-tuned model’s answers; the subsequent rule-based fact-checking shows no “High” hallucinations over 115 benchmark items and a majority Low level (55.7%).
* This method is different from your earlier cosine-similarity evaluation (0.700 on 10 Qs and 0.585 on 60 Qs after fine-tune). While not directly comparable, both indicate the same trend: grounding + fine-tuning reduces hallucination risk.

Across NLI-based detection, self-consistency checks, fine-tuning, and RAG with fact-checking, results consistently showed a progressive reduction in hallucinations. Fine-tuning more than doubled similarity scores (0.346 to 0.700; 0.289 to 0.585), and RAG with rule-based fact-checking eliminated high-level hallucinations (0%), proving that grounding and adaptation together make medical QA responses more reliable and trustworthy.