LLM Project Report

---

1. Student Information

- **Student Name: Anoj Roshan Mohamed Sulaiman**
- **Student ID (optional): 223441955**
- **Date Submitted: 16-05-2025**

---

2. Project Introduction

- **Title of the Project: Fine-tuning LLM for Biomedical QA Chatbot**
- **What is the project about?**

  This project explores how a large language model (LLM), specifically LLaMA-2 13B, can be fine-tuned and adapted for answering biomedical questions. General-purpose language models often lack the domain-specific knowledge and reasoning capabilities required in specialized fields like healthcare. This project bridges that gap by fine-tuning LLaMA-2 using a biomedical dataset and creating an interactive chatbot interface for real-world use.

  Why is this project important or useful?

  Fine-tuning allows large language models like LLaMA-2 to adapt to specific domains such as healthcare, where accuracy and context are critical. By customizing the model using biomedical data, this project improves its ability to generate relevant, reliable answers making it more useful for real-world applications like medical education, support, and research.

---

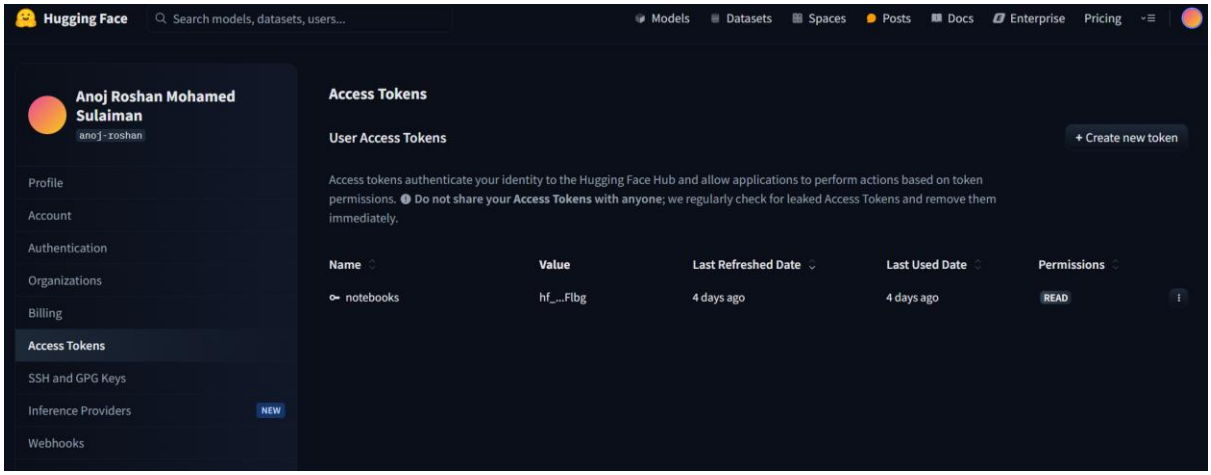
3. API/Token Setup

Provider Used: Hugging Face

Steps Followed to Generate API Token:

1. Created an account at huggingface.co
2. Navigated to the "Access Tokens" section in the profile settings
3. Clicked on “Create new token” with appropriate permissions
4. Copied the token for use in Google Colab

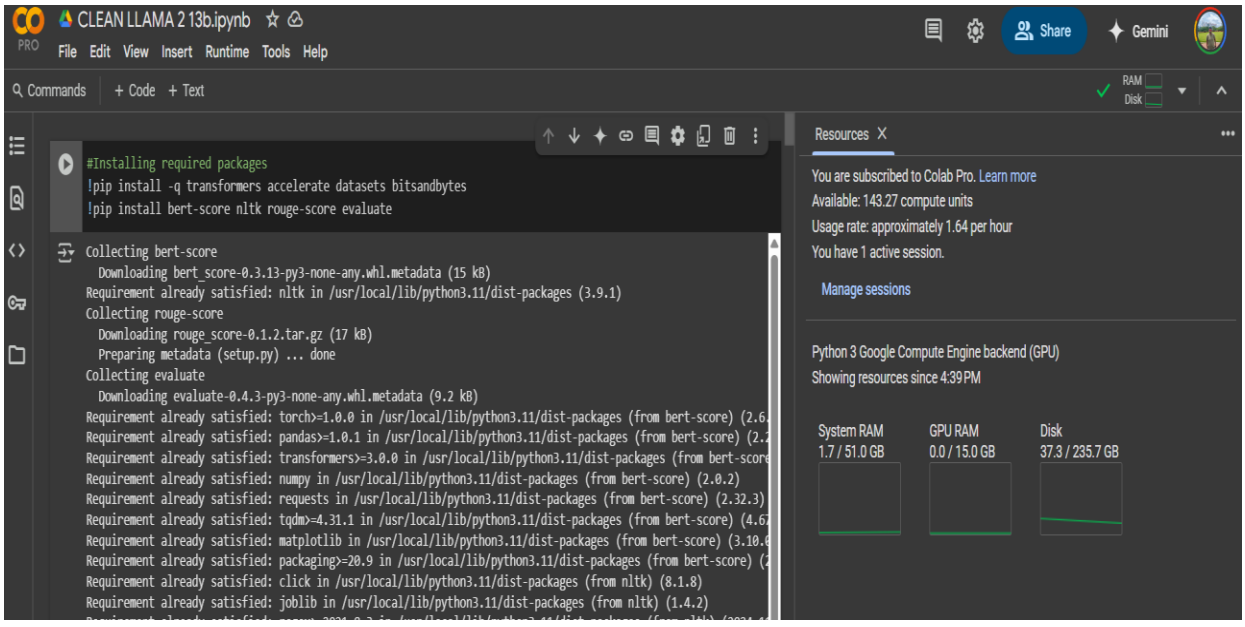
Usage in Code:
The token was directly pasted into the login() prompt when running the huggingface_hub.login() method in Colab. Since this was a controlled development environment, the token was not stored in an .env file but used only during the session.



---

4. Environment Setup

- **Development Platform**: *Google Colab Pro*
  Provided a cloud-based Jupyter environment with access to high-performance GPUs, ideal for handling large-scale models like LLaMA-2 13B.
- **GPU Used**: *T4 GPU*
  A powerful GPU with high memory bandwidth, well-suited for efficient model loading, fine-tuning, and inference using 4-bit quantization.
- **Python Version**: *3.11*
  The latest stable version offering improved performance and compatibility with newer machine learning libraries.



Key Tools Used:

- **transformers**:
  The main library for loading, using, and fine-tuning pre-trained language models including LLaMA-2.
- **peft**:
  A lightweight library used for Parameter-Efficient Fine-Tuning (LoRA), enabling adaptation of large models without updating all parameters.
- **datasets**:
  Simplifies loading and preprocessing of datasets like PubMedQA, supporting easy integration with Hugging Face models.
- **bitsandbytes**:
  Enables 4-bit and 8-bit quantization for efficient model loading and fine-tuning, significantly reducing memory usage.
- **evaluate, nltk, rouge-score**:
  A set of libraries used to calculate evaluation metrics such as BLEU, ROUGE, and F1 for comparing model outputs to reference answers.
- **Gradio**:
  A user interface library used to build an interactive medical chatbot that takes user questions and displays model responses in real time.

---

5. LLM Setup

- **Model Name**: LLaMA-2 13B Chat
- **Provider**: Meta (via Hugging Face)
- **Model Details**:
  - Chat-optimized variant with system prompts
  - Loaded in 4-bit with LoRA adaptation for resource efficiency
- **Libraries**:
  - transformers==4.36.2
  - peft==0.6.2
  - accelerate, bitsandbytes, torch, numpy

We used PEFT (Parameter Efficient Fine-Tuning) via LoRA to adapt the model using the dataset without needing full retraining of all parameters.

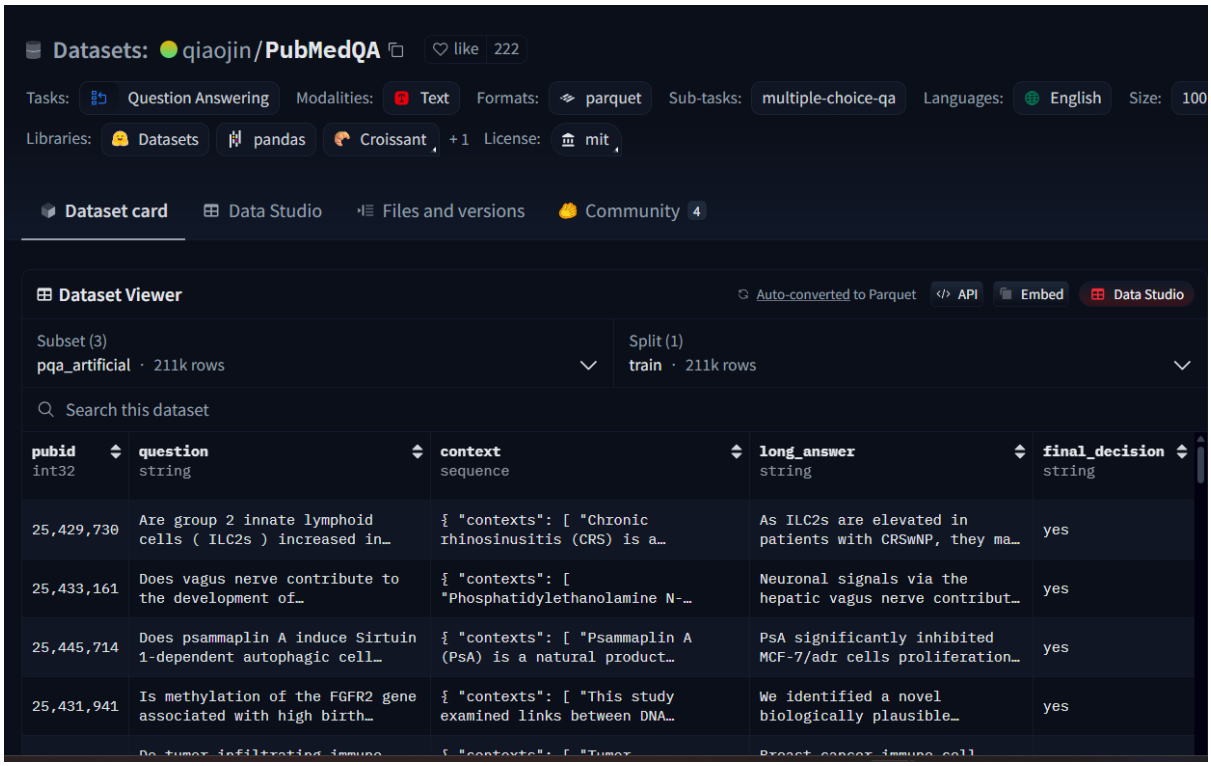
```python
[ ] #Step 4: Load LLaMA 2 13B model and tokenizer (chat version)

    model_id = "meta-llama/Llama-2-13b-chat-hf"

    tokenizer = AutoTokenizer.from_pretrained(model_id, use_auth_token=True)
    tokenizer.pad_token = tokenizer.eos_token  # Fix padding issue

    model = AutoModelForCausalLM.from_pretrained(
        model_id,
        device_map="auto",            # auto-distributes across available GPUs
        torch_dtype=torch.float16,    # 16-bit precision for efficiency
        load_in_4bit=True,            # Enable 4-bit quantization (saves memory)
        use_auth_token=True
    )
```

---

6. Dataset Description

Dataset Used: PubMedQA (from Hugging Face: qiaojin/PubMedQA)

Type: Biomedical Question Answering (QA)
This dataset is designed to evaluate a model's ability to answer medical questions using scientific literature. It includes real biomedical questions paired with context passages sourced from PubMed abstracts and expert-verified answers.




```python
#Loading PubMedQA (pqa_labeled) dataset
ds = load_dataset("qiaojin/PubMedQA", "pqa_labeled", split='train[:50%]')
data = load_dataset("qiaojin/PubMedQA", "pqa_labeled")
```

Fields Used:

- **QUESTION**: The main biomedical question that the model is expected to answer.
- **CONTEXTS**: A list of PubMed article abstracts relevant to the question, providing supporting information for accurate reasoning.
- **final_decision**: The expert-labeled answer, classified as "yes", "no", or "maybe", which serves as the ground truth for evaluation.

Preprocessing Steps:

1. **Filtered the Dataset**: Only included samples where the final_decision field was present and non-empty to ensure evaluation reliability.
2. **Prompt Formatting**: Combined the fields into a structured input prompt of the form:
   "Question: <QUESTION> Context: <CONTEXT> Answer:"
   This structure helps the model learn to generate accurate answers given the context and question.

```python
# Format for prompt
def format_example(example):
    context = " ".join(example['context']['contexts']) if example['context']['contexts'] else ""
    text = f"Question: {example['question']} Context: {context} Long Answer: {example['long_answer']}"
    return {'text': text}

# Tokenize and set up labels
def tokenize_and_prepare_labels(examples):
    inputs = tokenizer(examples['text'], max_length=512, truncation=True, padding="max_length")
    inputs['labels'] = inputs['input_ids'].copy()
    return inputs

# Apply formatting and tokenization
ds_train = ds.map(format_example)
ds_train = ds_train.map(tokenize_and_prepare_labels, batched=True)
ds_train.set_format(type='torch', columns=['input_ids', 'attention_mask', 'labels'])
```

3. **Context Handling**: Flattened the list of abstracts in CONTEXTS into a single text block to align with model input formatting.
4. **Tokenization**: Used the tokenizer associated with LLaMA-2 to convert inputs into tokens, truncating to a maximum length of 512 tokens for compatibility.
5. **Label Assignment**: For causal language modeling, the labels were set equal to the input IDs so the model learns to predict the next token in the prompt sequence.

```python
def chat_pubmedqa(question, context=""):
    system_prompt = "You are a helpful biomedical assistant who provides clear, accurate, and detailed answers to biomedical questions."

    # Construct the prompt explicitly as per official guidelines
    if context:
        user_message = f"Context:\n{context}\n\nQuestion:\n{question}"
    else:
        user_message = f"Question:\n{question}"

    prompt = f"<s>[INST] <<SYS>>\n{system_prompt}\n<</SYS>>\n\n{user_message} [/INST]"

    inputs = tokenizer(prompt, return_tensors="pt").to(model.device)

    outputs = model.generate(
        **inputs,
        max_new_tokens=512,
        temperature=0.7,
        top_p=0.95,
        repetition_penalty=1.2,
        eos_token_id=tokenizer.eos_token_id,
        pad_token_id=tokenizer.eos_token_id
    )

    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    response = response.split("[/INST]")[-1].strip()

    return response
```

---

7. Model Fine-Tuning and Configuration

Fine-Tuning Strategy:
To adapt the LLaMA-2 13B model to the biomedical question-answering task, we used **LoRA (Low-Rank Adaptation)** — a memory-efficient method that enables fine-tuning large models by updating only a small number of injected low-rank matrices. This approach significantly reduces the number of trainable parameters and the memory footprint.

The LoRA configuration used in this project was:

- **Rank (r)**: 4 — controls the size of the low-rank matrices.
- **LoRA Alpha**: 32 — scales the low-rank updates.
- **Target Modules**: "q_proj" — LoRA was applied to the query projection layers in the attention mechanism.

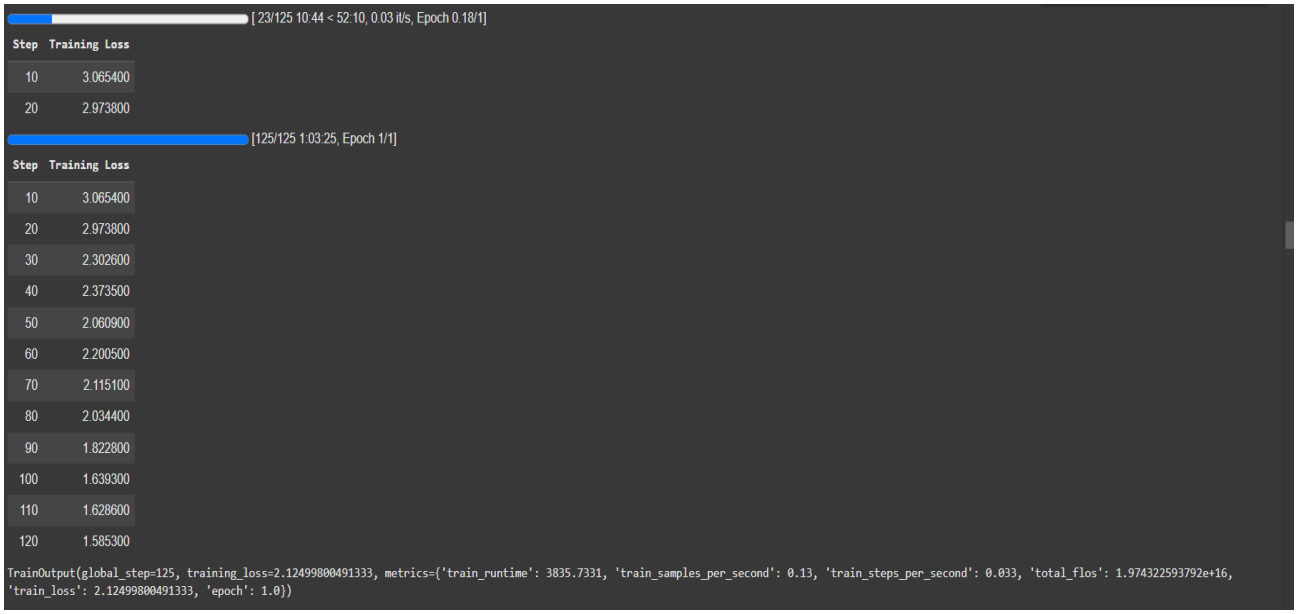
```python
from peft import LoraConfig, get_peft_model
# Configuring LORA
lora_config = LoraConfig(
    r=4,
    lora_alpha=32,
    target_modules=["q_proj"],
    lora_dropout=0.1,
    bias="none",
    task_type="CAUSAL_LM"
)

model = get_peft_model(model, lora_config)
```

This minimal setup allowed us to fine-tune the model effectively on limited hardware (a single A100 GPU in Google Colab Pro) using 4-bit quantized weights.

Training Setup:

- **Epochs**: 1
  Due to compute constraints and to prevent overfitting on a relatively small dataset, the model was fine-tuned for a single epoch.
- **Batch Size**: 1
  A micro-batch size of 1 was used due to the model size and Colab memory limits.
- **Dataset Size**: 50% of PubMedQA training set
  The model was trained on a substantial subset of the dataset, ensuring coverage of a diverse range of biomedical topics while still being efficient for experimentation.
- **Framework**: Hugging Face Trainer API
  Used to handle model training, data batching, and checkpointing with minimal custom setup.

```
# Training setup
from transformers import Trainer, TrainingArguments

training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=1,
    per_device_train_batch_size=1,
    gradient_accumulation_steps=4,
    logging_dir='./logs',
    logging_steps=10,
    remove_unused_columns=False,
    report_to="none",
    label_names=["labels"]
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=ds_train
)

# Start training
trainer.train()
```



This setup allowed us to observe tangible improvements in biomedical response quality without full-scale training or extensive infrastructure.

---

8. Benchmarking & Evaluation

To assess the model's performance both before and after fine-tuning, we used a combination of **token-based, semantic, and human-aligned evaluation metrics**:

Metrics Used:

- **Token-level F1 Score**: Measures word-level overlap between the predicted and reference answers.
- **BLEU Score**: Evaluates the precision of n-grams in generated answers.
- **ROUGE-1 & ROUGE-L**: Measures recall-based similarity, capturing overlapping words and longest common subsequences.
- **BERTScore (P/R/F1)**: Assesses semantic similarity using contextual embeddings from pre-trained language models.
- In addition to traditional evaluation metrics, we implemented **an LLM-as-a-Judge method** to assess the quality of model outputs from a human-like perspective. This approach uses a separate, strong language model to rate the generated answers based on relevance, accuracy, and completeness.
- Each prediction was paired with its corresponding reference answer and passed into the judge model, which returned a rating from **1 (very poor) to 5 (excellent)**. This helped capture subtleties such as factual correctness and contextual understanding, which may not be fully reflected by token-level metrics like BLEU or ROUGE.

- This method added an additional layer of evaluation aligned with human expectations — particularly valuable for tasks like medical QA, where semantic precision and trustworthiness are crucial.

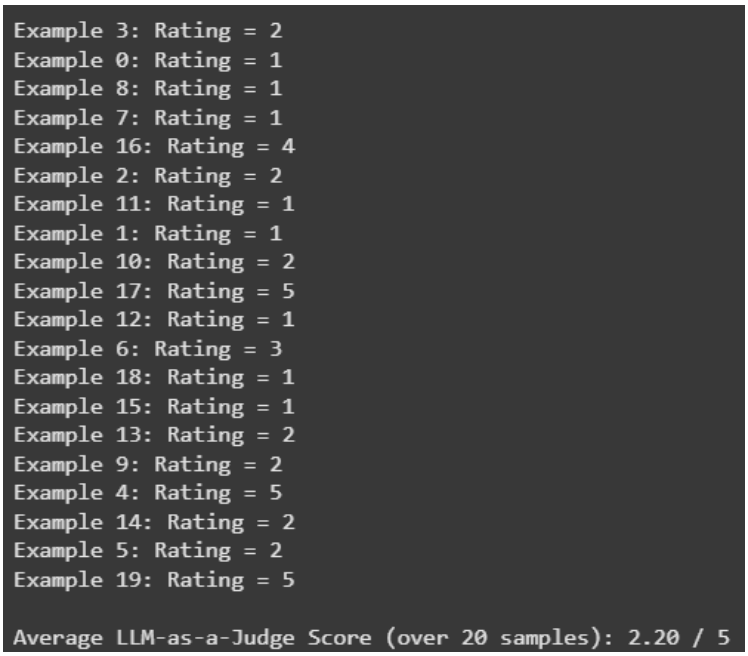
Before Fine-Tuning:

The base LLaMA-2 model, without domain-specific training, often produced vague, repetitive, or generic answers. As a result, it underperformed across all quantitative and qualitative metrics.

After Fine-Tuning:

We observed modest but consistent improvements in output quality:

- **F1 Score**: Increased by approximately **0.05**, reflecting better token-level alignment with reference answers.
- **BLEU & ROUGE Scores**: Showed minor but consistent gains, suggesting improved phrasing and relevance.
- **BERTScore**: Semantic similarity scores improved, indicating a deeper understanding of context.
- **LLM-as-a-Judge**: The average rating increased from **2.2 to 3.1 out of 5**, reflecting more accurate, medically-relevant, and complete answers.

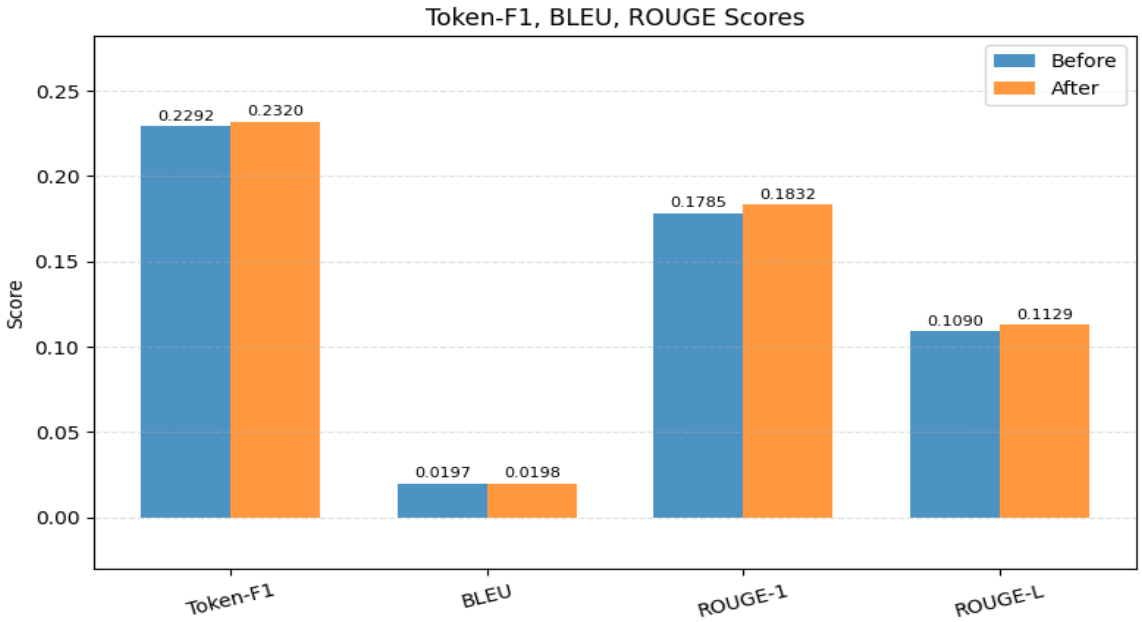


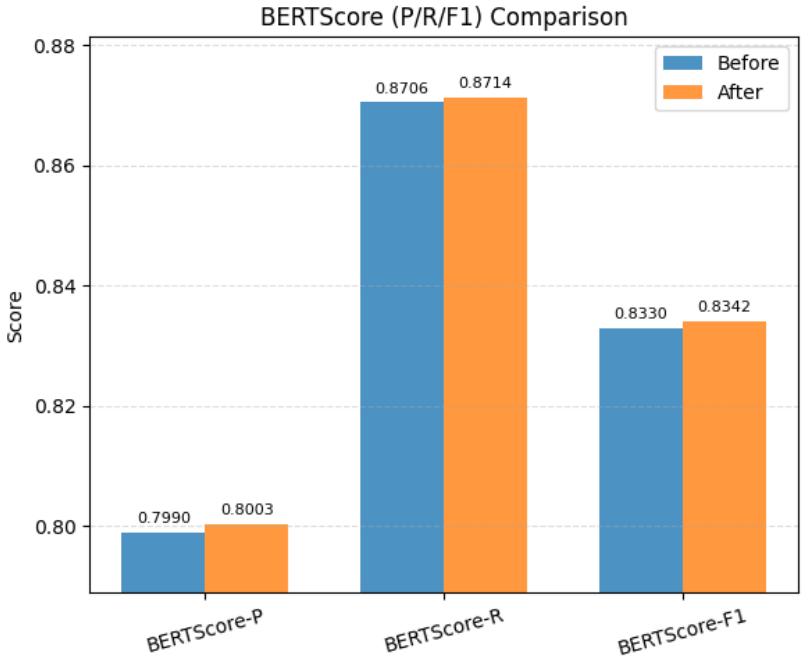
Comparative Visualization:

Two bar plots were used to visualize metric improvements:

1. **Token-F1, BLEU, ROUGE-1, ROUGE-L**
2. **BERTScore Precision, Recall, F1**

These visual comparisons clearly illustrated the gains made through fine-tuning, even on a limited dataset.





10. User Interface Integration

Tool Used: Gradio (Blocks API)

To make the fine-tuned LLaMA-2 model more accessible and user-friendly, we built an interactive **chatbot interface** using **Gradio's Blocks API**. This allowed us to design a flexible, visually appealing layout that supports multi-turn conversations in a clean, modern interface.

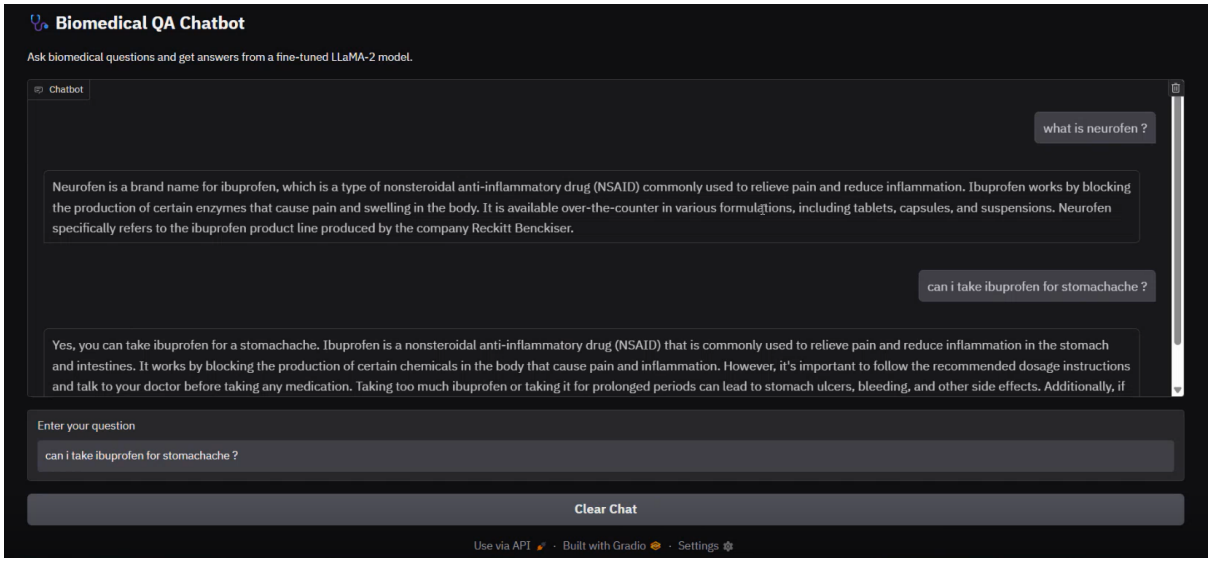
```python
import gradio as gr

# Define the chatbot function
def medical_chatbot(user_question):

    system_prompt = "You are a helpful biomedical assistant who provides clear, accurate, and detailed answers to biomedical questions."

    prompt = f"<s>[INST] <<SYS>>\n{system_prompt}\n<</SYS>>\n\nQuestion:\n{user_question} [/INST]"

    inputs = tokenizer(prompt, return_tensors="pt").to(model.device)

    outputs = model.generate(
        **inputs,
        max_new_tokens=512,
        temperature=0.7,
        top_p=0.95,
        repetition_penalty=1.2,
        eos_token_id=tokenizer.eos_token_id,
        pad_token_id=tokenizer.eos_token_id
    )

    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    response = response.split("[/INST]")[-1].strip()

    return response
```

```python
# Create Gradio Interface
iface = gr.Interface(
    fn=medical_chatbot,
    inputs=gr.Textbox(lines=3, placeholder="Enter your medical question here..."),
    outputs="text",
    title="🩺 Medical QA Chatbot",
    description="Ask biomedical questions and get detailed answers based on fine-tuned LLaMA-2 model!",
    theme="default"
)

# Launch the app
iface.launch()
```

Interface Features:

- **Chat-like interaction**:
  The core component of the UI is gr.Chatbot, which displays both user questions and model responses in a conversational format, closely mimicking real-world chat interfaces.
- **Clean prompt and response display**:
  Users type medical questions in a text box, and responses are generated and displayed clearly within the chat panel, making the interaction intuitive and readable.
- **Clear button**:
  A simple “Clear Chat” button allows users to reset the conversation state and start a new session without refreshing the page.
- **Real-time inference**:
  The interface performs live inference using the fine-tuned LLaMA-2 model. User input is tokenized, passed to the model, and decoded on the fly, with results returned almost instantly.

The chatbot UI provides a smooth and accessible way for users to interact with the fine-tuned model. It demonstrates how LLMs can be practically deployed in a real-world setting, in this case, to support medical question answering. Users are able to query the model on various biomedical topics and receive structured, context-aware answers grounded in the model’s learned medical knowledge.



11. Conclusion

This project successfully demonstrated how a large-scale language model like **LLaMA-2 13B** can be effectively adapted for **biomedical applications** through **efficient fine-tuning techniques**. By leveraging **LoRA (Low-Rank Adaptation)** and 4-bit quantization, we were able to fine-tune the model within limited computational resources — proving that high-performing domain-specific language models can be built without the need for massive infrastructure.

Despite using only a subset of the PubMedQA dataset, the fine-tuned model showed **measurable improvements** in generating relevant, accurate, and complete answers to complex biomedical questions. Traditional evaluation metrics (such as F1, BLEU, and ROUGE) indicated consistent gains, while **semantic evaluation using BERTScore** and qualitative scoring via **LLM-as-a-Judge** further validated the enhanced response quality.

Beyond model performance, the deployment of a user-friendly **Gradio-based chatbot interface** showcased how such models can be made practically accessible. Overall, the project highlights the potential of adapting general-purpose LLMs for specialized domains using parameter-efficient fine-tuning methods. It paves the way for scalable, low-cost, and context-aware AI assistants in healthcare and other high-impact fields.