SHARDA
UNIVERSITY
Beyond Boundaries

AGENTIC AI LAB

NAME-TAUSHIR ALAM

SYSTEM ID-2023509207

ROLL NO-2301010904

Working code of fine tuning

1. Install libraries

pip install -U transformers datasets accelerate peft trl sentencepiece

2.create training data

%%writefile train.jsonl

{"instruction": "What is chunking in NLP?", "output": "Chunking is the process of splitting long text into smaller pieces for processing."}

{"instruction": "Explain cosine similarity", "output": "Cosine similarity measures similarity between two vectors using the angle between them."}

{"instruction": "What is chunk overlap?", "output": "Chunk overlap repeats some tokens between chunks so that context is preserved."}

3.fine tuning code

```
from transformers import AutoTokenizer, AutoModelForCausalLM, TrainingArguments
from datasets import load_dataset
from peft import LoraConfig, get_peft_model
from trl import SFTTrainer
import torch

# Base model (lightweight, perfect for Colab)
model_name = "microsoft/phi-2"

# Load tokenizer & model
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16,
```

```
    device_map="auto"
)

# LoRA configuration
lora_config = LoraConfig(
    r=8,
    lora_alpha=16,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05,
    task_type="CAUSAL_LM"
)

# Apply LoRA
model = get_peft_model(model, lora_config)

# Load dataset
dataset = load_dataset("json", data_files="train.jsonl", split="train")

# Format instruction data
def format(example):
    return f"### Instruction:\n{example['instruction']}\n\n### Response:\n{example['output']}"

# Training settings
training_args = TrainingArguments(
    output_dir="./results",
```

```
    per_device_train_batch_size=1,
    gradient_accumulation_steps=4,
    num_train_epochs=3,
    logging_steps=1,
    save_strategy="epoch",
    fp16=True,
    report_to="none"
)

# Trainer
trainer = SFTTrainer(
    model=model,
    train_dataset=dataset,
    args=training_args,
    formatting_func=format,
    tokenizer=tokenizer
)

# Train
trainer.train()

# Save model
model.save_pretrained("fine_tuned_model")
tokenizer.save_pretrained("fine_tuned_model")

print("Fine-tuning finished successfully!")
```