Phi-4 Fine-Tuning: Detailed Explanation

Phi-4 Fine-Tuning: Detailed Explanation

1. Overview of the Code

This project fine-tunes the Phi-4 language model using custom datasets extracted from a PDF file. The workflow includes:

- Extracting text from a PDF
- Processing and structuring the dataset
- Loading the Phi-4 model via Hugging Face
- Fine-tuning using LoRA (Low-Rank Adaptation)
- Training and saving the fine-tuned model

2. Installing Required Libraries

First, the script installs necessary dependencies such as PyMuPDF (pymupdf) for extracting text from PDFs, and libraries like transformers, datasets, peft, accelerate, and torch for handling the NLP model.

```
!pip install pymupdf
!pip install transformers
!pip install datasets
!pip install peft
!pip install accelerate
!pip install bitsandbytes
!pip install torch
```

3. Extracting Text from a PDF

To process the text, PyMuPDF (fitz) is used. This function reads the PDF file and extracts text efficiently.

```
import fitz  # PyMuPDF for reading PDFs

def extract_text_from_pdf(pdf_path):
    pdf_path = "/content/Nlp and llm content.pdf"  # Hardcoded path
    doc = fitz.open(pdf_path)
    text = "\n".join([page.get_text("text") for page in doc])  # Extract text
    return text
```

4. Structuring the Dataset

Extracted text is split into question-answer format. The dataset is then converted into Hugging Face's Dataset format for easy training.

Phi-4 Fine-Tuning: Detailed Explanation

```
from datasets import Dataset

def create_dataset_from_text(pdf_path):
    raw_text = extract_text_from_pdf(pdf_path)
    text_chunks = raw_text.split("\n\n")  # Splitting by double line breaks

    instructions, responses = [], []
    for i in range(0, len(text_chunks) - 1, 2):  # Alternating Q/A format
        instructions.append(text_chunks[i].strip())
        responses.append(text_chunks[i + 1].strip())

    data_dict = {"instruction": instructions, "response": responses}
    dataset = Dataset.from_dict(data_dict)  # Convert to Hugging Face dataset
    return dataset
```

5. Loading the Phi-4 Model

The Phi-4 model is loaded using the Hugging Face Transformers library. This model was selected due to its efficiency and strong NLP capabilities.

```
from transformers import AutoModelForCausalLM, AutoTokenizer

model_name = "microsoft/phi-4"
model = AutoModelForCausalLM.from_pretrained(model_name)
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

6. Tokenization and Dataset Preparation

Tokenization ensures the dataset is formatted correctly for training, using padding, truncation, and setting a max length for inputs.

```
def tokenize_function(examples):
    return tokenizer(examples["instruction"], truncation=True, padding="max_length", max_length=512)

tokenized_dataset = dataset.map(tokenize_function, batched=True)
```

7. Fine-Tuning with LoRA

LoRA (Low-Rank Adaptation) is used for fine-tuning. This reduces memory requirements by training only specific model layers instead of the entire network.

```
from peft import LoraConfig, get_peft_model

lora_config = LoraConfig(
    r=16,  # Rank of the LoRA decomposition
```

```
    lora_alpha=32,
    lora_dropout=0.1,
    bias="none",
    target_modules=["q_proj", "v_proj"]  # Updates attention layers only
)

peft_model = get_peft_model(model, lora_config)
```

8. Training and Saving the Model

The model is fine-tuned using the Trainer API from Hugging Face, with custom training parameters. Once trained, the model is saved for further use.

```
from transformers import TrainingArguments, Trainer

training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    num_train_epochs=3,
    weight_decay=0.01,
)

trainer = Trainer(
    model=peft_model,
    args=training_args,
    train_dataset=tokenized_dataset,
)

trainer.train()
peft_model.save_pretrained("./fine_tuned_phi4")
tokenizer.save_pretrained("./fine_tuned_phi4")
```

9. Generating and Evaluating Outputs

After fine-tuning, sample inputs are fed into the model to generate and evaluate outputs. This verifies if the model correctly processes user queries.

```
input_text = "How does AI impact the job market?"
inputs = tokenizer(input_text, return_tensors="pt")
output = model.generate(**inputs, max_length=100)
print(tokenizer.decode(output[0], skip_special_tokens=True))
```

Phi-4 Fine-Tuning: Detailed Explanation

10. Summary of Key Choices

- **Why extract from PDFs?** Real-world Q/A format improves fine-tuning quality.
- **Why use Phi-4?** Developed by Microsoft, optimized for NLP efficiency.
- **Why use LoRA?** Reduces GPU memory requirements and accelerates training.
- **Why Hugging Face?** Provides an end-to-end framework for NLP model training.