INTELLIHACK 5.0 - INITIAL ROUND DELEGATES

Intellihack_DataDominators_TaskNumber03

Documented Training Process

Documented Training Process

1. Installation of Dependencies

To ensure all required libraries are available for training, we install the necessary dependencies.

```
!pip install torch==2.3.0+cu121 -f https://download.pytorch.org/whl/torch_stable.html

!pip install unsloth==2025.3.9

!pip install transformers==4.48.3

!pip install datasets==2.19.0

!pip install numpy==1.26.4
```

2. Data Preprocessing

The markdown files are read, structured, and prepared for fine-tuning.

```
import os

from datasets import Dataset


def read_md_files(directory):

    data = []

    md_files = ["dataset.md", "deepseekv3-explained.md"]

    for filename in md_files:

        file_path = os.path.join(directory, filename)

        if os.path.exists(file_path):

            with open(file_path, "r", encoding="utf-8") as file:

                content = file.read().strip()

                data.append({"text": content})
    return data


def split_into_chunks(data, chunk_size=200):
    chunked_data = [{"text": " ".join(entry["text"].split()[:chunk_size])} for entry in data]
    return chunked_data
```

3. Model Loading & Tokenization

The model is loaded efficiently using Unsloth's FastLanguageModel with 4-bit quantization.

```
from transformers import AutoTokenizer
from unsloth import FastLanguageModel

model_name = "Qwen/Qwen2-0.5B"
model, tokenizer = FastLanguageModel.from_pretrained(
    model_name,
    max_seq_length=128,
    dtype=torch.float16,
    load_in_4bit=True
)

def tokenize_function(examples):
    tokenized = tokenizer(examples["text"], truncation=True, padding="max_length", max_length=128)
    tokenized["labels"] = tokenized["input_ids"].copy()
    return tokenized
```

4. Training Configuration

Hyperparameters are set to optimize training efficiency in a constrained environment.

```
from transformers import TrainingArguments, Trainer
from datasets import load_from_disk

train_dataset = load_from_disk("/content/drive/MyDrive/intellihack/dataset/train")
test_dataset = load_from_disk("/content/drive/MyDrive/intellihack/dataset/test")

training_args = TrainingArguments(
  output_dir="/content/qwen2_finetuned",
  per_device_train_batch_size=8,
  per_device_eval_batch_size=8,
  learning_rate=5e-5,
  num_train_epochs=3,
  weight_decay=0.01,
  evaluation_strategy="epoch"
)

trainer = Trainer(
  model=model,
  args=training_args,
  train_dataset=train_dataset,
  eval_dataset=test_dataset,
  tokenizer=tokenizer
)
```

5. Training Execution

The fine-tuning process is initiated with the following command.

```
trainer.train()
```

6. Evaluation of Model Performance

After training, the model's performance is assessed using evaluation metrics.

```
eval_results = trainer.evaluate()

print("Evaluation Loss:", eval_results["eval_loss"])
```