TASK 01 – TEXT GENERATION WITH GPT 2

Description : Train a model to generate coherent and contextually relevant text based on a given prompt. Starting with GPT-2, a transformer model developed by OpenAI, you will learn how to fine-tune the model on a custom dataset to create text that mimics the style and structure of your training data.

**1.Install dependencies**

pip install transformers datasets torch

**2**. **Prepare the Dataset**

[

{

"prompt": "What are the symptoms of the flu?",

"response": "The symptoms include fever, cough, sore throat, and body aches."

},

{

"prompt": "How to boil an egg?",

"response": "Place eggs in boiling water for 9-12 minutes depending on desired firmness."

}

]

# data\_loader.py

from datasets import Dataset

def load\_dataset():

data = [

{"text": "Prompt: What are the symptoms of the flu?\nResponse: The symptoms include fever, cough, sore throat, and body aches."},

{"text": "Prompt: How to boil an egg?\nResponse: Place eggs in boiling water for 9-12 minutes depending on desired firmness."}

]

return Dataset.from\_list(data)

**3**. **Fine-Tune GPT-2**

from transformers import GPT2Tokenizer, GPT2LMHeadModel, Trainer, TrainingArguments, DataCollatorForLanguageModeling

from data\_loader import load\_dataset

# Load tokenizer and model

tokenizer = GPT2Tokenizer.from\_pretrained("gpt2")

tokenizer.pad\_token = tokenizer.eos\_token # Add padding token

model = GPT2LMHeadModel.from\_pretrained("gpt2")

# Load dataset

dataset = load\_dataset()

# Tokenize data

def tokenize(example):

return tokenizer(example["text"], truncation=True, padding="max\_length", max\_length=128)

tokenized\_dataset = dataset.map(tokenize)

# Training setup

training\_args = TrainingArguments(

output\_dir="./gpt2-finetuned",

per\_device\_train\_batch\_size=4,

num\_train\_epochs=3,

logging\_steps=10,

save\_steps=50,

save\_total\_limit=2,

prediction\_loss\_only=True

)

data\_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)

trainer = Trainer(

model=model,

args=training\_args,

train\_dataset=tokenized\_dataset,

data\_collator=data\_collator

)

# Start fine-tuning

trainer.train()

**4**. **Generate Output**

prompt = "Prompt: How do I treat a cold?\nResponse:"

inputs = tokenizer(prompt, return\_tensors="pt")

outputs = model.generate(\*\*inputs, max\_new\_tokens=50, do\_sample=True)

print(tokenizer.decode(outputs[0]))

**Input:**

Prompt: What is the capital of France?

Response:

**Output:**

Prompt: What is the capital of France?

Response: The capital of France is Paris.