**Documentation for GPT-2 Fine-Tuning and Text Generation Script**

This documentation provides an overview of the Python script used for fine-tuning a GPT-2 language model and generating text. The script leverages the Hugging Face Transformers library and includes functions for reading text files, preparing datasets, training the model, and generating text.

**Requirements**

Before running the script, ensure you have the following libraries installed:

* pandas
* numpy
* re
* os
* transformers

You can install the required libraries using:

pip install pandas numpy

pip install pip install

pip install transformers

**Code Overview**

**Import Statements**

python

Copy code

import pandas as pd

import numpy as np

import re

import os

from transformers import TextDataset, DataCollatorForLanguageModeling

from transformers import GPT2Tokenizer, GPT2LMHeadModel

from transformers import Trainer, TrainingArguments

These import statements bring in the necessary libraries and modules for data handling, text processing, and model training.

**Functions**

**1. read\_txt(file\_path)**

Reads a text file and returns its content as a string.

python

Copy code

def read\_txt(file\_path):

with open(file\_path, "r", encoding='utf-8') as file:

text = file.read()

return text

**2. load\_dataset(file\_path, tokenizer, block\_size=128)**

Loads the dataset for training using the provided tokenizer.

python

Copy code

def load\_dataset(file\_path, tokenizer, block\_size=128):

dataset = TextDataset(

tokenizer=tokenizer,

file\_path=file\_path,

block\_size=block\_size,

)

return dataset

**3. load\_data\_collator(tokenizer, mlm=False)**

Loads the data collator for language modeling.

python

Copy code

def load\_data\_collator(tokenizer, mlm=False):

data\_collator = DataCollatorForLanguageModeling(

tokenizer=tokenizer,

mlm=mlm,

)

return data\_collator

**4. train(train\_file\_path, model\_name, output\_dir, overwrite\_output\_dir, per\_device\_train\_batch\_size, num\_train\_epochs, save\_steps)**

Trains the GPT-2 model using the specified parameters.

python

Copy code

def train (train\_file\_path, model\_name,

output\_dir,

overwrite\_output\_dir,

per\_device\_train\_batch\_size,

num\_train\_epochs,

save\_steps):

tokenizer = GPT2Tokenizer.from\_pretrained(model\_name)

train\_dataset = load\_dataset(train\_file\_path, tokenizer)

data\_collator = load\_data\_collator(tokenizer)

tokenizer.save\_pretrained(output\_dir)

model = GPT2LMHeadModel.from\_pretrained(model\_name)

model.save\_pretrained(output\_dir)

training\_args = TrainingArguments(

output\_dir=output\_dir,

overwrite\_output\_dir=overwrite\_output\_dir,

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

num\_train\_epochs=num\_train\_epochs,

)

trainer = Trainer (

model=model,

args=training\_args,

data\_collator=data\_collator,

train\_dataset=train\_dataset,

)

trainer.train()

trainer.save\_model()

**5. load\_model(model\_path)**

Loads a pretrained GPT-2 model from the specified path.

python

Copy code

def load\_model(model\_path):

model = GPT2LMHeadModel.from\_pretrained(model\_path)

return model

**6. load\_tokenizer(tokenizer\_path)**

Loads a pretrained tokenizer from the specified path.

python

Copy code

def load\_tokenizer(tokenizer\_path):

tokenizer = GPT2Tokenizer.from\_pretrained(tokenizer\_path)

return tokenizer

**7. generate\_text(model\_path, tokenizer\_path, sequence, max\_length)**

Generates text using the trained GPT-2 model.

python

Copy code

def generate\_text(model\_path, tokenizer\_path, sequence, max\_length):

model = load\_model(model\_path)

tokenizer = load\_tokenizer(tokenizer\_path)

ids = tokenizer.encode(f'{sequence}', return\_tensors='pt')

final\_outputs = model.generate(

ids,

do\_sample=True,

max\_length=max\_length,

pad\_token\_id=model.config.eos\_token\_id,

top\_k=50,

top\_p=0.95,

)

print(tokenizer.decode(final\_outputs[0], skip\_special\_tokens=True))

model\_path=’paste your model path’

text=input()

max\_len=150

generate\_text(model\_path, text, max\_len)

**Main Script Execution**

1. Read and preprocess the training text file:

python

Copy code

train\_directory = r"your path "

text\_data = read\_txt(train\_directory)

text\_data = re.sub(r'\n+', '\n', text\_data).strip()

with open(r'path, "w", encoding='utf-8') as f:

f.write(text\_data)

1. Define training parameters and execute the training function:

python

Copy code

train\_file\_path = r"path"

model\_name = 'gpt2'

output\_dir = r'C:\Users\kesav\Guvi\guvi main\c6'

overwrite\_output\_dir = False

per\_device\_train\_batch\_size = 8

num\_train\_epochs = 30

save\_steps = 10000

# Train the model

train (

train\_file\_path=train\_file\_path,

model\_name=model\_name,

output\_dir=output\_dir,

overwrite\_output\_dir=overwrite\_output\_dir,

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

num\_train\_epochs=num\_train\_epochs,

save\_steps=save\_steps

)

This script reads a text file, preprocesses the text, trains a GPT-2 model with the preprocessed text, and saves the trained model and tokenizer. The script also includes functions for loading the trained model and tokenizer and generating text from the trained model.