Project 2 — GPT-2 Text Style Transfer

Why This Project Matters

Style transfer teaches conditioning — guiding a model to produce text in a specific style. This project will help you master prompt formatting, dataset preparation, and output control for creative text generation.

Prerequisites

- Project 1 knowledge (loading models, tokenization, fine-tuning loop) - Prompt design basics — creating clear input-output patterns - Basic text preprocessing — cleaning and formatting datasets

Dataset

```
We'll use a custom Shakespearean dataset from Hugging Face:
Link: https://huggingface.co/datasets/karpathy/shakespeare
Or create a CSV with columns: plain_text, style_text

Example:
plain_text,style_text
Hello, my friend.,Hark! Mine companion.
How are you today?,How doth thee fare this day?

Loading CSV:
from datasets import load_dataset
dataset = load_dataset("csv", data_files="shakespeare_pairs.csv")
```

Step-by-Step Build

```
Step 1: Load tokenizer & model:
from transformers import GPT2Tokenizer, GPT2LMHeadModel
model_name = "gpt2"
tokenizer = GPT2Tokenizer.from_pretrained(model_name)
tokenizer.pad_token = tokenizer.eos_token
model = GPT2LMHeadModel.from_pretrained(model_name)

Step 2: Preprocess dataset:
def preprocess(example):
    prompt = f"Plain: {example['plain_text']} Style: {example['style_text']}"
    tokens = tokenizer(prompt, truncation=True, padding="max_length", max_length=128)
    tokens["labels"] = tokens["input_ids"].copy()
    return tokens

tokenized_dataset = dataset.map(preprocess, batched=True)

Step 3: Training loop:
from transformers import Trainer, TrainingArguments
training_args = TrainingArguments(
    output_dir="./gpt2-style",
    evaluation_strategy="epoch",
```

```
    logging_strategy="steps",
    logging_steps=50,
    learning_rate=5e-5,
    per_device_train_batch_size=2,
    num_train_epochs=3,
    weight_decay=0.01,
    push_to_hub=False
)
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset["train"],
    eval_dataset=tokenized_dataset["validation"]
)
trainer.train()

Step 4: Save & test:
trainer.save_model("./gpt2-style")
input_text = "Plain: I love programming. Style:"
inputs = tokenizer(input_text, return_tensors="pt")
outputs = model.generate(**inputs, max_length=50, temperature=0.8, top_p=0.9)
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

Design Thinking

- Prompt format ensures transformation context is clear - 128 tokens is enough for most style examples - Loss tracking helps detect overfitting

Troubleshooting

```
Problem | Cause | Fix
Model copies input | Too few style examples | Increase dataset size
Outputs too short | Small generation length | Increase max_length in generate()
Style inconsistent | Mixed-style training data | Clean dataset
```

Flowchart

```
[Dataset (plain+style)] → [Prompt formatting] → [Tokenization] → [Model load] → [Fine-tuning]
```

Mini Quiz

```
1. Why add “Plain:” and “Style:” tags?
2. Risk of mixed-style datasets?
3. Which parameter increases creativity?

Answers:
1. Gives explicit transformation context
2. Produces inconsistent styles
3. temperature
```

Side Quest

- Train on sarcasm style tweets
- Try distilgpt2 for speed
- Multiple styles in one model using tags like Style: Shakespeare, Style: Formal