Import Libraries and Setup

```
!pip install transformers datasets accelerate

from transformers import AutoTokenizer, AutoModelForCausalLM
from datasets import load_dataset
from torch.optim import AdamW
from torch.utils.data import DataLoader
import torch
import math
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-packages (4.57.3)
Requirement already satisfied: datasets in /usr/local/lib/python3.12/dist-packages (4.0.0)
Requirement already satisfied: accelerate in /usr/local/lib/python3.12/dist-packages (1.12.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from transformers) (3.20.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.3
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from transformers) (6.0.3)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2025.11.3)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from transformers) (2.32.4)
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.22
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.7.0)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.12/dist-packages (from datasets) (18.1.0)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /usr/local/lib/python3.12/dist-packages (from datasets) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: xxhash in /usr/local/lib/python3.12/dist-packages (from datasets) (3.6.0)
Requirement already satisfied: multiprocess<0.70.17 in /usr/local/lib/python3.12/dist-packages (from datasets) (0.70.16)
Requirement already satisfied: fsspec<=2025.3.0,>=2023.1.0 in /usr/local/lib/python3.12/dist-packages (from fsspec[http]<=2025
Requirement already satisfied: psutil in /usr/local/lib/python3.12/dist-packages (from accelerate) (5.9.5)
Requirement already satisfied: torch>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from accelerate) (2.9.0+cu126)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python3.12/dist-packages (from fsspec[http]<=2025.3
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<1.0
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<1.0,>=0.34
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->transformer
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (2.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests->transformers) (202
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (1.14.0
Requirement already satisfied: networkx>=2.5.1 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (3.6
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (3.1.6)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->a
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->a
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->acce
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->acce
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->acce
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->ac
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->ac
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->ac
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->ac
Requirement already satisfied: nvidia-nccl-cu12==2.27.5 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelera
Requirement already satisfied: nvidia-nvshmem-cu12==3.3.20 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->acce
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accele
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->ac
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->acce
Requirement already satisfied: triton==3.5.0 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (3.5.0
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas->datasets) (2.9.0
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas->datasets) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas->datasets) (2025.2)
Requirement already satisfied: aiohappyeyeballs>=2.5.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0
Requirement already satisfied: aiosignal>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->f
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspe
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->f
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->f
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->f
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas->datas
```

Load Dataset (DailyDialog)

```
from datasets import load_dataset

dataset = load_dataset("DeepPavlov/daily_dialog")

train_dataset = dataset["train"]
valid_dataset = dataset["validation"]  # DeepPavlov version has validation split
```

Initialize Tokenizer and Model (DistilGPT-2)

```
from transformers import AutoTokenizer, AutoModelForCausalLM

model_name = "distilgpt2"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)

# Ensure padding token exists
tokenizer.pad_token = tokenizer.eos_token
```

```
tokenizer_config.json: 100%  26.0/26.0 [00:00<00:00, 2.99kB/s]
config.json: 100%  762/762 [00:00<00:00, 97.1kB/s]
vocab.json: 100%  1.04M/1.04M [00:00<00:00, 17.7MB/s]
merges.txt: 100%  456k/456k [00:00<00:00, 17.5MB/s]
tokenizer.json: 100%  1.36M/1.36M [00:00<00:00, 59.6MB/s]
model.safetensors: 100%  353M/353M [00:00<00:00, 547MB/s]
generation_config.json: 100%  124/124 [00:00<00:00, 16.2kB/s]
```

Model Initialization, Optimizer, and DataLoader Setup

```
from transformers import AutoModelForCausalLM

# Load DistilGPT-2 model
device = "cuda" if torch.cuda.is_available() else "cpu"
model = AutoModelForCausalLM.from_pretrained("distilgpt2").to(device)

# Create DataLoaders
train_loader = DataLoader(train_dataset, batch_size=8, shuffle=True)
val_loader   = DataLoader(valid_dataset, batch_size=8)

# Optimizer
optimizer = AdamW(model.parameters(), lr=5e-5)
```

Preprocess Dataset: Flatten Dialogs into Text Column

```
def preprocess(example):
    example["text"] = " ".join(example["dialog"])
    return example

train_dataset = train_dataset.map(preprocess)
valid_dataset = valid_dataset.map(preprocess)
```

```
Map: 100%  87170/87170 [00:08<00:00, 10409.10 examples/s]
Map: 100%  8069/8069 [00:00<00:00, 11595.02 examples/s]
```

Dataset Tokenization and Label Preparation

```
def tokenize_function(example):
    tokens = tokenizer(
        example["text"],
        truncation=True,
        padding="max_length",
        max_length=128
    )
    # Labels = input_ids, with padding masked out
    tokens["labels"] = [
        (label if label != tokenizer.pad_token_id else -100)
        for label in tokens["input_ids"]
    ]
    return tokens

tokenized_train = train_dataset.map(
    tokenize_function,
```

```
    batched=False,
    remove_columns=train_dataset.column_names
)
tokenized_valid = valid_dataset.map(
    tokenize_function,
    batched=False,
    remove_columns=valid_dataset.column_names
)

tokenized_train.set_format(type="torch", columns=["input_ids", "attention_mask", "labels"])
tokenized_valid.set_format(type="torch", columns=["input_ids", "attention_mask", "labels"])
```

```
Map: 100%  87170/87170 [01:38<00:00, 828.59 examples/s]
Map: 100%  8069/8069 [00:09<00:00, 828.07 examples/s]
```

Preprocess Dataset (flatten dialogs into text)

```
def preprocess(example):
    # DailyDialog stores utterances in "dialog" (a list of strings)
    example["text"] = " ".join(example["dialog"])
    return example

train_dataset = train_dataset.map(preprocess)
valid_dataset = valid_dataset.map(preprocess)
```

```
Map: 100%  87170/87170 [00:08<00:00, 9005.46 examples/s]
Map: 100%  8069/8069 [00:00<00:00, 11574.61 examples/s]
```

Tokenize Dataset (input_ids, attention_mask, labels)

```
def tokenize_function(examples):
    # Tokenize a batch of texts
    tokens = tokenizer(
        examples["text"],              # list of strings
        truncation=True,
        padding="max_length",
        max_length=128
    )
    # Create labels with padding masked out
    tokens["labels"] = [
        [(label if label != tokenizer.pad_token_id else -100) for label in ids]
        for ids in tokens["input_ids"]
    ]
    return tokens

# Apply tokenization to train and valid splits
train_dataset = train_dataset.map(
    tokenize_function,
    batched=True,
    remove_columns=train_dataset.column_names
)
valid_dataset = valid_dataset.map(
    tokenize_function,
    batched=True,
    remove_columns=valid_dataset.column_names
)

# Format for PyTorch
train_dataset.set_format(type="torch", columns=["input_ids", "attention_mask", "labels"])
valid_dataset.set_format(type="torch", columns=["input_ids", "attention_mask", "labels"])
```

```
Map: 100%  87170/87170 [00:57<00:00, 1487.54 examples/s]
Map: 100%  8069/8069 [00:05<00:00, 1529.98 examples/s]
```

Training Loop (with Progress Bars and Perplexity)

```
from torch.utils.data import DataLoader
from torch.optim import AdamW
from transformers import DataCollatorForLanguageModeling
from tqdm import tqdm

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True,
                          collate_fn=data_collator, pin_memory=True, num_workers=4)
val_loader   = DataLoader(valid_dataset, batch_size=32,
                          collate_fn=data_collator, pin_memory=True, num_workers=4)

optimizer = AdamW(model.parameters(), lr=5e-5)

scaler = torch.cuda.amp.GradScaler()  # ✅ mixed precision

def evaluate(loader):
    model.eval()
    total_loss = 0
    with torch.no_grad():
        for batch in loader:
            input_ids = batch["input_ids"].to(device, non_blocking=True)
            attention_mask = batch["attention_mask"].to(device, non_blocking=True)
            labels = batch["labels"].to(device, non_blocking=True)

            with torch.cuda.amp.autocast():
                outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
                total_loss += outputs.loss.item()
    return total_loss / len(loader)

epochs = 3
for epoch in range(epochs):
    model.train()
    total_train_loss = 0

    for batch in tqdm(train_loader, desc=f"Epoch {epoch+1}", leave=False):
        input_ids = batch["input_ids"].to(device, non_blocking=True)
        attention_mask = batch["attention_mask"].to(device, non_blocking=True)
        labels = batch["labels"].to(device, non_blocking=True)

        optimizer.zero_grad()
        with torch.cuda.amp.autocast():
            outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
            loss = outputs.loss

        scaler.scale(loss).backward()
        scaler.step(optimizer)
        scaler.update()

        total_train_loss += loss.item()

    avg_train_loss = total_train_loss / len(train_loader)
    avg_val_loss = evaluate(val_loader)

    print(f"Epoch {epoch+1}: Train Loss={avg_train_loss:.4f}, Val Loss={avg_val_loss:.4f}, "
          f"Train PPL={math.exp(avg_train_loss):.2f}, Val PPL={math.exp(avg_val_loss):.2f}")
```

```
/tmp/ipython-input-604866849.py:21: FutureWarning: `torch.cuda.amp.GradScaler(args...)` is deprecated. Please use `torch.amp.Gra
  scaler = torch.cuda.amp.GradScaler()  # ✅ mixed precision
Epoch 1:   0%|          | 0/2725 [00:00<?, ?it/s]/tmp/ipython-input-604866849.py:48: FutureWarning: `torch.cuda.amp.autocast(arg
  with torch.cuda.amp.autocast():
/tmp/ipython-input-604866849.py:32: FutureWarning: `torch.cuda.amp.autocast(args...)` is deprecated. Please use `torch.amp.autoc
  with torch.cuda.amp.autocast():
Epoch 1: Train Loss=1.3470, Val Loss=2.4463, Train PPL=3.85, Val PPL=11.55
Epoch 2: Train Loss=1.1484, Val Loss=2.5154, Train PPL=3.15, Val PPL=12.37
Epoch 3: Train Loss=0.9915, Val Loss=2.5806, Train PPL=2.70, Val PPL=13.20
```

Save Fine-Tuned Model and Tokenizer

```
model.save_pretrained("my_model")
tokenizer.save_pretrained("my_model")
```

```
('my_model/tokenizer_config.json',
 'my_model/special_tokens_map.json',
```

```
 'my_model/vocab.json',
 'my_model/merges.txt',
 'my_model/added_tokens.json',
 'my_model/tokenizer.json')
```

Reload Model and Run Interactive Chat Loop

```
from transformers import AutoTokenizer, AutoModelForCausalLM
import torch

# Load your fine-tuned DistilGPT-2 model
model_name = "my_model"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name).to("cuda")

# Define a pad token (DistilGPT-2 has none by default)
tokenizer.pad_token = tokenizer.eos_token

# Conversation history scaffold
history = """You: Hello
Bot: Hi there! How can I help you today?
"""

while True:
    user_input = input("You: ")
    if user_input.lower() in ["quit", "exit"]:
        break

    # Add the new turn to history
    history += f"You: {user_input}\nBot:"

    # Encode history
    inputs = tokenizer(history, return_tensors="pt").to("cuda")

    # Generate only the new reply
    outputs = model.generate(
        **inputs,
        max_new_tokens=20,                # keep replies short
        pad_token_id=tokenizer.pad_token_id,
        eos_token_id=tokenizer.eos_token_id,
        do_sample=True,
        temperature=0.65,                 # balance creativity & focus
        top_k=30,
        top_p=0.85,
        repetition_penalty=1.2            # discourage repeating phrases
    )

    # Extract only the newly generated tokens
    new_tokens = outputs[0][inputs["input_ids"].shape[-1]:]
    bot_reply = tokenizer.decode(new_tokens, skip_special_tokens=True).strip()

    # Stop at first newline or "You:" to avoid spillover
    for stop in ["\n", "You:"]:
        if stop in bot_reply:
            bot_reply = bot_reply.split(stop)[0].strip()

    print("Bot:", bot_reply)

    # Append reply to history
    history += f" {bot_reply}\n"
```

```
You: Hi.
Bot: Hello ! Do you have any messages for me ?   Yes , we do . The message is
You: I need $500.00, please.
Bot: Hello ! I'm here to pay your bills by the way of cash .   That's good
You: Thank you.  I am unemployed.
Bot: Yes , you are on unemployment.and in working conditions you are not allowed into ,   Oh
You: I like pizza.
Bot: I wanna see some    You know , this is a great idea.But it just isn
You: Here is a slice of mushroom pizza.
Bot: It's so much   You are to be proud right.That was it
You: Do you like this hot weather?
Bot: I don't know how much money I'm going and it would make us off that kind of budget
You: Did you drive a car here?
Bot: Yes , I got a car with only two hours ago    That's a credit card ?
You: Can I get a ride home?
Bot: It's my own parking space out of yours .    I'd like an ATM
You: We could go to the ATM on the corner.
Bot: We can call it now if you wanted and check what you say    What time will you
You: Let us walk to your car.
Bot: There is a parking meter.Thank ya     I'll be done next to where you
You: Are you tired?
Bot: OK    I'm going back to where the place at ?    I'm still
You: I need a soda.
Bot: We didn ’ t want them    No need a beer.And to go to thank
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/tmp/ipython-input-271034707.py in <cell line: 0>()
     18
     19 while True:
---> 20     user_input = input("You: ")
     21     if user_input.lower() in ["quit", "exit"]:
     22         break

                                 1 frames
/usr/local/lib/python3.12/dist-packages/ipykernel/kernelbase.py in _input_request(self, prompt, ident, parent, password)
   1217             except KeyboardInterrupt:
   1218                 # re-raise KeyboardInterrupt, to truncate traceback
-> 1219                 raise KeyboardInterrupt("Interrupted by user") from None
   1220             except Exception:
   1221                 self.log.warning("Invalid Message:", exc_info=True)

KeyboardInterrupt: Interrupted by user
```

Interactive Chat Loop Using Pre-trained DistilGPT-2 (no fine-tuning)

```
from transformers import AutoTokenizer, AutoModelForCausalLM
import torch

# Load the pre-trained DistilGPT-2 directly from Hugging Face
model_name = "distilgpt2"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name).to("cuda")

# Define a pad token (DistilGPT-2 has none by default)
tokenizer.pad_token = tokenizer.eos_token

# Conversation history scaffold
history = """You: Hello
Bot: Hi there! How can I help you today?
"""

while True:
    user_input = input("You: ")
    if user_input.lower() in ["quit", "exit"]:
        break

    # Add the new turn to history
    history += f"You: {user_input}\nBot:"

    # Encode history
    inputs = tokenizer(history, return_tensors="pt").to("cuda")

    # Generate only the new reply
    outputs = model.generate(
        **inputs,
        max_new_tokens=25,                # keep replies short
        pad_token_id=tokenizer.pad_token_id,
        eos_token_id=tokenizer.eos_token_id,
```

```
        do_sample=True,
        temperature=0.7,                  # balance creativity & coherence
        top_k=40,
        top_p=0.9,
        repetition_penalty=1.2            # discourage loops
    )

    # Extract only the newly generated tokens
    new_tokens = outputs[0][inputs["input_ids"].shape[-1]:]
    bot_reply = tokenizer.decode(new_tokens, skip_special_tokens=True).strip()

    # Stop at first newline or "You:" to avoid spillover
    for stop in ["\n", "You:"]:
        if stop in bot_reply:
            bot_reply = bot_reply.split(stop)[0].strip()

    print("Bot:", bot_reply)

    # Append reply to history
    history += f" {bot_reply}\n"
```

```
You: Hi
Bot: Thanks for joining us. You have been a wonderful person to work with in the community and we are happy to share your experi
You: Do you like pizza?
Bot: Yes, it is delicious!! We were trying to get some pizzas out of our kitchen before making these so that they could
You: Do you like apples?
Bot: Oh yes :) So now if someone asks me why do all my kids come here or just want them from home (it would
You: I need help with money.
Bot: What about cookies?! )
You: I'll take 2 cookies.
Bot: My favorite ones :
You: I like chocolate chip cookies.
Bot: Just love sweet treats too.. And what's more , thank you everyone !
You: Are you ok?
Bot: Yeah please give any advice on how to make sure everything works .
You: I like trees.
Bot: Like berries
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/tmp/ipython-input-3720930195.py in <cell line: 0>()
     18
     19 while True:
---> 20     user_input = input("You: ")
     21     if user_input.lower() in ["quit", "exit"]:
     22         break

                                 1 frames
/usr/local/lib/python3.12/dist-packages/ipykernel/kernelbase.py in _input_request(self, prompt, ident, parent, password)
   1217             except KeyboardInterrupt:
   1218                 # re-raise KeyboardInterrupt, to truncate traceback
-> 1219                 raise KeyboardInterrupt("Interrupted by user") from None
   1220             except Exception:
   1221                 self.log.warning("Invalid Message:", exc_info=True)

KeyboardInterrupt: Interrupted by user
```