```
!pip install -q transformers datasets accelerate evaluate rouge_score
```

```
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.1/84.1 kB 2.6 MB/s eta 0:00:00
  Building wheel for rouge_score (setup.py) ... done
```

```
import os
import json
import math
from typing import List, Dict

import torch
from torch.utils.data import DataLoader
from datasets import Dataset

from transformers import AutoTokenizer, GPT2LMHeadModel
from rouge_score import rouge_scorer
from tqdm.auto import tqdm

from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Adjust these if you changed paths anywhere
DATASET_PATH = "/content/drive/My Drive/Colab Notebooks/CS 561: Topics in Data Privacy/Data/"
MODEL_PATH   = "/content/drive/My Drive/Colab Notebooks/CS 561: Topics in Data Privacy/Models/"

BASELINE_DIR = os.path.join(MODEL_PATH, "gpt2_baseline_poisoned")
DPAGGZO_DIR  = os.path.join(MODEL_PATH, "gpt2_dp_aggzo_poisoned")

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)

MAX_LEN = 128
```

```
Using device: cpu
```

```
def load_jsonl_as_strings(path: str) -> List[str]:
    texts = []
    with open(path, "r", encoding="utf-8") as f:
        for line in f:
            line = line.strip()
            if not line:
                continue
            obj = json.loads(line)  # each line is a JSON string
            texts.append(str(obj))
    return texts

# Load clean validation set
val_file   = os.path.join(DATASET_PATH, "val.jsonl")
val_texts  = load_jsonl_as_strings(val_file)
print("Validation set size:", len(val_texts))

# Load target passage
```

```
with open(os.path.join(DATASET_PATH, "target_passage.txt"), "r", encoding="utf-8") as f:
    target_passage = f.read().strip()

print("\nTarget passage preview:")
print(target_passage[:400], "...")
```

```
Validation set size: 4888

Target passage preview:
a mayor for a dad. “I got to thinking. If it wasn’t Stahl who attacked you, why would you say that it was? Unless you were covering for someone. Like the mayor’s son.” “Why wo
```

```
def make_collate_fn(tokenizer):
    def collate(batch):
        texts = [ex["text"] for ex in batch]
        enc = tokenizer(
            texts,
            return_tensors="pt",
            padding=True,
            truncation=True,
            max_length=MAX_LEN,
        )
        # labels are input_ids with padding masked out
        enc["labels"] = enc["input_ids"].clone()
        enc = {k: v.to(device) for k, v in enc.items()}
        return enc
    return collate


def compute_perplexity(model: GPT2LMHeadModel,
                       tokenizer: AutoTokenizer,
                       texts: List[str],
                       batch_size: int = 16) -> float:
    dataset = Dataset.from_dict({"text": texts})
    loader = DataLoader(
        dataset,
        batch_size=batch_size,
        shuffle=False,
        collate_fn=make_collate_fn(tokenizer),
    )

    model.eval()
    total_loss = 0.0
    total_tokens = 0

    with torch.no_grad():
        for batch in tqdm(loader, desc="Perplexity eval"):
            outputs = model(**batch)
            loss = outputs.loss  # mean over non -100 tokens

            labels = batch["labels"]
            mask = labels.ne(-100)
            tokens_in_batch = mask.sum().item()

            total_loss   += loss.item() * tokens_in_batch
            total_tokens += tokens_in_batch

    mean_loss = total_loss / total_tokens
```

```
    ppl = math.exp(mean_loss)
    return ppl
```

```
def split_target_prefix(target_text: str, prefix_ratio: float = 0.25):
    words = target_text.split()
    split_idx = max(1, int(len(words) * prefix_ratio))
    prefix        = " ".join(words[:split_idx])
    target_suffix = " ".join(words[split_idx:])
    return prefix, target_suffix


def sample_continuations(model: GPT2LMHeadModel,
                         tokenizer: AutoTokenizer,
                         prefix: str,
                         num_samples: int = 200,
                         batch_size: int = 10,
                         max_new_tokens: int = 128) -> List[str]:
    """
    Generate num_samples continuations from the model given the prefix.
    Uses sampling for diversity, batched for T4 efficiency.
    """
    model.eval()
    continuations = []

    enc = tokenizer(prefix, return_tensors="pt")
    input_ids = enc["input_ids"].to(device)
    input_len = input_ids.shape[1]

    with torch.no_grad():
        while len(continuations) < num_samples:
            cur_bs = min(batch_size, num_samples - len(continuations))

            outputs = model.generate(
                input_ids.repeat(cur_bs, 1),
                do_sample=True,
                temperature=0.7,
                top_k=40,
                max_new_tokens=max_new_tokens,
                pad_token_id=tokenizer.eos_token_id,
            )

            for out in outputs:
                gen_ids = out[input_len:]  # strip prefix
                text = tokenizer.decode(gen_ids, skip_special_tokens=True)
                continuations.append(text)

    return continuations[:num_samples]


def compute_rougeL_scores(model: GPT2LMHeadModel,
                          tokenizer: AutoTokenizer,
                          target_text: str,
                          num_samples: int = 200,
                          batch_size: int = 10) -> Dict[str, float]:
    prefix, target_suffix = split_target_prefix(target_text, prefix_ratio=0.25)
    print("Prompt (prefix) used for memorization test:\n")
    print(prefix, "\n")
```

```
    continuations = sample_continuations(
        model,
        tokenizer,
        prefix,
        num_samples=num_samples,
        batch_size=batch_size,
        max_new_tokens=128,
    )

    scorer = rouge_scorer.RougeScorer(["rougeL"], use_stemmer=True)

    scores = []
    for gen in continuations:
        score = scorer.score(target_suffix, gen)["rougeL"].fmeasure
        scores.append(score)

    max_score  = max(scores)
    mean_score = sum(scores) / len(scores)

    return {
        "rougeL_max": max_score,
        "rougeL_mean": mean_score,
    }
```

```
def evaluate_model(model_dir: str,
                   name: str,
                   num_rouge_samples: int = 200) -> Dict[str, float]:
    print(f"\n==================== Evaluating {name} ====================\n")
    tokenizer = AutoTokenizer.from_pretrained(model_dir)
    model     = GPT2LMHeadModel.from_pretrained(model_dir)
    model.to(device)

    # Utility: Perplexity on clean validation data
    ppl = compute_perplexity(model, tokenizer, val_texts, batch_size=16)
    print(f"\n{name} — Perplexity on clean val.jsonl: {ppl:.3f}\n")

    # Memorization: ROUGE-L on target passage
    rouge_scores = compute_rougeL_scores(
        model,
        tokenizer,
        target_passage,
        num_samples=num_rouge_samples,
        batch_size=10,
    )
    print(f"\n{name} — ROUGE-L (max over samples):  {rouge_scores['rougeL_max']:.4f}")
    print(f"{name} — ROUGE-L (mean over samples): {rouge_scores['rougeL_mean']:.4f}\n")

    out = {
        "perplexity": ppl,
        **rouge_scores,
    }
    return out
```

```
results_baseline = evaluate_model(BASELINE_DIR, "Baseline GPT-2 (AdamW)", num_rouge_samples=200)
results_dpaggzo  = evaluate_model(DPAGGZO_DIR, "DP-AggZO GPT-2",         num_rouge_samples=200)
```

```
print("\n==================== Summary ====================\n")
print("Baseline:", results_baseline)
print("DP-AggZO:", results_dpaggzo)
```

```
==================== Evaluating Baseline GPT-2 (AdamW) ====================

Perplexity eval: 100%                                                306/306 [50:54<00:00,  8.22s/it]
`loss_type=None` was set in the config but it is unrecognized. Using the default loss: `ForCausalLMLoss`.
The attention mask is not set and cannot be inferred from input because pad token is same as eos token. As a consequence, you may observe unexpected behavior. Please pass your

Baseline GPT-2 (AdamW) — Perplexity on clean val.jsonl: 33.412

Prompt (prefix) used for memorization test:

a mayor for a dad. “I got to thinking. If it wasn’t Stahl who attacked you, why would you say that it was? Unless you were covering for someone. Like the mayor’s son.” “Why wo


Baseline GPT-2 (AdamW) — ROUGE-L (max over samples):  0.1247
Baseline GPT-2 (AdamW) — ROUGE-L (mean over samples): 0.0994


==================== Evaluating DP-AggZO GPT-2 ====================

Perplexity eval: 100%                                                306/306 [50:21<00:00,  8.56s/it]

DP-AggZO GPT-2 — Perplexity on clean val.jsonl: 5758.638

Prompt (prefix) used for memorization test:

a mayor for a dad. “I got to thinking. If it wasn’t Stahl who attacked you, why would you say that it was? Unless you were covering for someone. Like the mayor’s son.” “Why wo


DP-AggZO GPT-2 — ROUGE-L (max over samples):  0.0903
DP-AggZO GPT-2 — ROUGE-L (mean over samples): 0.0630


==================== Summary ====================

Baseline: {'perplexity': 33.41217380509253, 'rougeL_max': 0.12474849094567403, 'rougeL_mean': 0.0993758338414651}
DP-AggZO: {'perplexity': 5758.637916821883, 'rougeL_max': 0.09034907597535935, 'rougeL_mean': 0.06300033723064166}
```

```
import pandas as pd

# Your evaluation results
results = {
    "Model": ["Baseline GPT-2 (AdamW)", "DP-AggZO GPT-2"],
    "Perplexity": [33.41217386909253, 5758.637916821883],
    "ROUGE-L Max": [0.12474849094567403, 0.09034907597535935],
    "ROUGE-L Mean": [0.0993758338414651, 0.06300337230641666]
}

df = pd.DataFrame(results)

# Format numbers like academic tables
df["Perplexity"]   = df["Perplexity"].apply(lambda x: f"{x:,.2f}")
```

```
df["ROUGE-L Max"]  = df["ROUGE-L Max"].apply(lambda x: f"{x:.4f}")
df["ROUGE-L Mean"] = df["ROUGE-L Mean"].apply(lambda x: f"{x:.4f}")

# Styling: clean, minimal, no index (just like NeurIPS/ICML tables)
styled = (
    df.style
    .hide(axis="index")  # <- removes the index completely
    .set_properties(**{
        'text-align': 'center',
        'font-size': '15px',
        'font-family': 'Arial'
    })
    .set_table_styles([
        {'selector': 'th',
         'props': [('text-align', 'center'),
                   ('font-size', '16px'),
                   ('font-weight', 'bold'),
                   ('border-bottom', '1px solid black')]}
    ])
)

styled
```

Model	Perplexity	ROUGE-L Max	ROUGE-L Mean
Baseline GPT-2 (AdamW)	33.41	0.1247	0.0994
DP-AggZO GPT-2	5,758.64	0.0903	0.0630