In [ ]:
```
!pip install transformers
!pip install datasets
```

In [ ]:
```
import pandas as pd
from datasets import load_dataset
from tqdm import tqdm
from transformers import GPT2Tokenizer, GPT2LMHeadModel, TrainingArguments, Tr
import torch
from torch.utils.data import random_split
from transformers import DataCollatorForLanguageModeling
import statistics
from nltk.translate.bleu_score import sentence_bleu
```

In [ ]:
```
dataset=load_dataset('demelin/moral_stories','full')

X_train = dataset['train']['norm'][:8000]
X_test = dataset['train']['norm'][-2000:]
print("Total Dataset (Including Validation) - ", len(dataset['train']))
print("Train Dataset - ",len(X_train),'\n',"Test Dataset - ",len(X_test))
```

```
Downloading builder script:   0%|          | 0.00/11.4k [00:00<?, ?B/s]

Downloading metadata:   0%|          | 0.00/69.2k [00:00<?, ?B/s]

Downloading readme:   0%|          | 0.00/12.0k [00:00<?, ?B/s]

Downloading and preparing dataset moral_stories/full (download: 7.64 MiB, gen
erated: 6.37 MiB, post-processed: Unknown size, total: 14.02 MiB) to /root/.c
ache/huggingface/datasets/demelin___moral_stories/full/1.1.0/47de080a74d44a1d
4785a2c16fe6c7a978ef218fb0dc319d8392d22337f7b806...

Downloading data:   0%|          | 0.00/8.02M [00:00<?, ?B/s]

Generating train split:   0%|          | 0/12000 [00:00<?, ? examples/s]

Dataset moral_stories downloaded and prepared to /root/.cache/huggingface/dat
asets/demelin___moral_stories/full/1.1.0/47de080a74d44a1d4785a2c16fe6c7a978ef
218fb0dc319d8392d22337f7b806. Subsequent calls will reuse this data.

  0%|          | 0/1 [00:00<?, ?it/s]
```

GPT-2

In [ ]:
```
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
model = GPT2LMHeadModel.from_pretrained("gpt2")
tokenizer.add_special_tokens({'pad_token': '[PAD]'})
model.resize_token_embeddings(len(tokenizer))
```

```
Downloading (…)olve/main/vocab.json:   0%|          | 0.00/1.04M [00:00<?, ?
B/s]

Downloading (…)olve/main/merges.txt:   0%|          | 0.00/456k [00:00<?, ?B/
s]

Downloading (…)lve/main/config.json:   0%|          | 0.00/665 [00:00<?, ?B/
s]

Downloading (…)"pytorch_model.bin";:   0%|          | 0.00/548M [00:00<?, ?B/
s]

Downloading (…)neration_config.json:   0%|          | 0.00/124 [00:00<?, ?B/
s]
```

Out[6]:
```
Embedding(50258, 768)
```

In [ ]:
```
max_length = max([len(tokenizer.encode(x)) for x in X_train])
```

In [ ]:
```
class moral():
    def __init__(self, x, tokenizer, max_length):
        self.input_ids = []
        self.attn_masks = []
        self.labels = []
        encodings_dict = tokenizer(x, max_length = max_length, padding = "max_
        self.input_ids = torch.tensor(encodings_dict["input_ids"])
        self.attn_masks = torch.tensor(encodings_dict["attention_mask"])

    def __len__(self):
        return len(self.input_ids)

    def __getitem__(self, idx):
        return self.input_ids[idx], self.attn_masks[idx]
```

In [ ]:
```
dataset = moral(X_train, tokenizer, max_length = max_length)
train_size = int(0.8 * len(dataset))
train_dataset, val_dataset = random_split(dataset, [train_size, len(dataset) -
```

In [ ]:
```
X_new = []
for i in range(0, len(train_dataset)):
  X_new.append({"input_ids": train_dataset[i][0], "attention_mask": train_data

Y_new = []
for i in range(0, len(val_dataset)):
  Y_new.append({"input_ids": val_dataset[i][0], "attention_mask": val_dataset[
```

In [ ]:
```
data_collator = DataCollatorForLanguageModeling(tokenizer, mlm = False)
training_args = TrainingArguments(num_train_epochs = 8,
                                  per_device_train_batch_size = 16,
                                  per_device_eval_batch_size = 8,
                                  learning_rate = 0.0001,
                                  weight_decay = 0,
                                  output_dir = '/content/Untitled Folder',
                                  evaluation_strategy = "epoch")
```

In [ ]:
```
trainer = Trainer(args = training_args, model = model, train_dataset = X_new,
trainer.train()
```

```
/usr/local/lib/python3.8/dist-packages/transformers/optimization.py:306: Futu
reWarning: This implementation of AdamW is deprecated and will be removed in
a future version. Use the PyTorch implementation torch.optim.AdamW instead, o
r set `no_deprecation_warning=True` to disable this warning
  warnings.warn(
***** Running training *****
  Num examples = 6400
  Num Epochs = 8
  Instantaneous batch size per device = 16
  Total train batch size (w. parallel, distributed & accumulation) = 16
  Gradient Accumulation steps = 1
  Total optimization steps = 3200
  Number of trainable parameters = 124440576
```

 [3200/3200 08:27, Epoch 8/8]

Epoch	Training Loss	Validation Loss
1	No log	2.554248
2	3.230000	2.536860
3	2.167200	2.602214
4	1.886100	2.725288
5	1.679000	2.846404
6	1.679000	2.994707
7	1.498600	3.100782
8	1.404500	3.156310

```
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8
Saving model checkpoint to /content/Untitled Folder/checkpoint-500
Configuration saved in /content/Untitled Folder/checkpoint-500/config.json
Configuration saved in /content/Untitled Folder/checkpoint-500/generation_con
fig.json
Model weights saved in /content/Untitled Folder/checkpoint-500/pytorch_model.
bin
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8
Saving model checkpoint to /content/Untitled Folder/checkpoint-1000
Configuration saved in /content/Untitled Folder/checkpoint-1000/config.json
Configuration saved in /content/Untitled Folder/checkpoint-1000/generation_co
nfig.json
Model weights saved in /content/Untitled Folder/checkpoint-1000/pytorch_mode
l.bin
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8
Saving model checkpoint to /content/Untitled Folder/checkpoint-1500
Configuration saved in /content/Untitled Folder/checkpoint-1500/config.json
Configuration saved in /content/Untitled Folder/checkpoint-1500/generation_co
nfig.json
Model weights saved in /content/Untitled Folder/checkpoint-1500/pytorch_mode
l.bin
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8
Saving model checkpoint to /content/Untitled Folder/checkpoint-2000
Configuration saved in /content/Untitled Folder/checkpoint-2000/config.json
Configuration saved in /content/Untitled Folder/checkpoint-2000/generation_co
nfig.json
Model weights saved in /content/Untitled Folder/checkpoint-2000/pytorch_mode
l.bin
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8
Saving model checkpoint to /content/Untitled Folder/checkpoint-2500
Configuration saved in /content/Untitled Folder/checkpoint-2500/config.json
Configuration saved in /content/Untitled Folder/checkpoint-2500/generation_co
nfig.json
Model weights saved in /content/Untitled Folder/checkpoint-2500/pytorch_mode
l.bin
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8
Saving model checkpoint to /content/Untitled Folder/checkpoint-3000
Configuration saved in /content/Untitled Folder/checkpoint-3000/config.json
Configuration saved in /content/Untitled Folder/checkpoint-3000/generation_co
nfig.json
Model weights saved in /content/Untitled Folder/checkpoint-3000/pytorch_mode
l.bin
```

```
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8


Training completed. Do not forget to share your model on huggingface.co/model
s =)
```

Out[12]:
```
TrainOutput(global_step=3200, training_loss=1.939154739379883, metrics={'trai
n_runtime': 510.9844, 'train_samples_per_second': 100.199, 'train_steps_per_s
econd': 6.262, 'total_flos': 444196454400000.0, 'train_loss': 1.9391547393798
83, 'epoch': 8.0})
```

In [ ]:
```
outputs = []
for i, sample in enumerate(X_test):
  inputs = tokenizer.encode(sample, return_tensors = 'pt').cuda()
  greedy_output = model.generate(inputs, top_p = 0.5, top_k = 0, temperature =
  outputs.append(tokenizer.decode(greedy_output[0], skip_special_tokens = True
```

```
Streaming output truncated to the last 5000 lines.
Generate config GenerationConfig {
  "bos_token_id": 50256,
  "eos_token_id": 50256,
  "transformers_version": "4.26.1"
}

The attention mask and the pad token id were not set. As a consequence, yo
u may observe unexpected behavior. Please pass your input's `attention_mas
k` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generate config GenerationConfig {
  "bos_token_id": 50256,
  "eos_token_id": 50256,
  "transformers_version": "4.26.1"
}

The attention mask and the pad token id were not set. As a consequence, yo
u may observe unexpected behavior. Please pass your input's `attention_mas
```

In [ ]:
```
blueScores = 0
for i, pred in enumerate(outputs):
  blueScores += sentence_bleu([X_test[i].split()], pred.split())

print(blueScores/len(outputs))
```

```
/usr/local/lib/python3.8/dist-packages/nltk/translate/bleu_score.py:552: User
Warning:
The hypothesis contains 0 counts of 3-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
/usr/local/lib/python3.8/dist-packages/nltk/translate/bleu_score.py:552: User
Warning:
The hypothesis contains 0 counts of 4-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)

0.5082161249922025
```

In [ ]:
```

```

Distil GPT-2

In [ ]:
```
tokenizer = GPT2Tokenizer.from_pretrained("distilgpt2")
model = GPT2LMHeadModel.from_pretrained("distilgpt2")
tokenizer.add_special_tokens({'pad_token': '[PAD]'})
model.resize_token_embeddings(len(tokenizer))
```

```
Downloading (…)olve/main/vocab.json:   0%|          | 0.00/1.04M [00:00<?,
?B/s]

Downloading (…)olve/main/merges.txt:   0%|          | 0.00/456k [00:00<?,
?B/s]

loading file vocab.json from cache at /root/.cache/huggingface/hub/models-
-distilgpt2/snapshots/f241065e938b44ac52db2c5de82c8bd2fafc76d0/vocab.json
loading file merges.txt from cache at /root/.cache/huggingface/hub/models-
-distilgpt2/snapshots/f241065e938b44ac52db2c5de82c8bd2fafc76d0/merges.txt
loading file added_tokens.json from cache at None
loading file special_tokens_map.json from cache at None
loading file tokenizer_config.json from cache at None

Downloading (…)lve/main/config.json:   0%|          | 0.00/762 [00:00<?, ?
B/s]

loading configuration file config.json from cache at /root/.cache/huggingf
ace/hub/models--distilgpt2/snapshots/f241065e938b44ac52db2c5de82c8bd2fafc7
6d0/config.json
```

In [ ]:
```
max_length = max([len(tokenizer.encode(x)) for x in X_train])
```

In [ ]:
```
class moral():
    def __init__(self, x, tokenizer, max_length):
        self.input_ids = []
        self.attn_masks = []
        self.labels = []
        encodings_dict = tokenizer(x, max_length = max_length, padding = "max_
        self.input_ids = torch.tensor(encodings_dict["input_ids"])
        self.attn_masks = torch.tensor(encodings_dict["attention_mask"])

    def __len__(self):
        return len(self.input_ids)

    def __getitem__(self, idx):
        return self.input_ids[idx], self.attn_masks[idx]
```

In [ ]:
```
dataset = moral(X_train, tokenizer, max_length = max_length)
train_size = int(0.8 * len(dataset))
train_dataset, val_dataset = random_split(dataset, [train_size, len(dataset) -
```

In [ ]:
```
X_new = []
for i in range(0, len(train_dataset)):
  X_new.append({"input_ids": train_dataset[i][0], "attention_mask": train_data

Y_new = []
for i in range(0, len(val_dataset)):
  Y_new.append({"input_ids": val_dataset[i][0], "attention_mask": val_dataset[
```

In [ ]:
```
data_collator = DataCollatorForLanguageModeling(tokenizer, mlm = False)
training_args = TrainingArguments(num_train_epochs = 8,
                                  per_device_train_batch_size = 16,
                                  per_device_eval_batch_size = 8,
                                  learning_rate = 0.0001,
                                  weight_decay = 0,
                                  output_dir = '/content/Untitled Folder1',
                                  evaluation_strategy = "epoch")
```

```
PyTorch: setting up devices
The default value for the training argument `--report_to` will change in v5
(from all installed integrations to none). In v5, you will need to use `--rep
ort_to all` to get the same behavior as now. You should start updating your c
ode and make this info disappear :-).
```

In [ ]:
```
trainer = Trainer(args = training_args, model = model, train_dataset = X_new,
trainer.train()
```

```
/usr/local/lib/python3.8/dist-packages/transformers/optimization.py:306: Futu
reWarning: This implementation of AdamW is deprecated and will be removed in
a future version. Use the PyTorch implementation torch.optim.AdamW instead, o
r set `no_deprecation_warning=True` to disable this warning
  warnings.warn(
***** Running training *****
  Num examples = 6400
  Num Epochs = 8
  Instantaneous batch size per device = 16
  Total train batch size (w. parallel, distributed & accumulation) = 16
  Gradient Accumulation steps = 1
  Total optimization steps = 3200
  Number of trainable parameters = 81913344
```

 [3200/3200 05:23, Epoch 8/8]

Epoch	Training Loss	Validation Loss
1	No log	2.625854
2	3.079500	2.598359
3	2.325900	2.608918
4	2.083500	2.691877
5	1.907900	2.772617
6	1.907900	2.845881
7	1.746400	2.922899
8	1.657700	2.964134

```
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8
Saving model checkpoint to /content/Untitled Folder1/checkpoint-500
Configuration saved in /content/Untitled Folder1/checkpoint-500/config.json
Configuration saved in /content/Untitled Folder1/checkpoint-500/generation_co
nfig.json
Model weights saved in /content/Untitled Folder1/checkpoint-500/pytorch_mode
l.bin
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8
Saving model checkpoint to /content/Untitled Folder1/checkpoint-1000
Configuration saved in /content/Untitled Folder1/checkpoint-1000/config.json
Configuration saved in /content/Untitled Folder1/checkpoint-1000/generation_c
onfig.json
Model weights saved in /content/Untitled Folder1/checkpoint-1000/pytorch_mode
l.bin
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8
Saving model checkpoint to /content/Untitled Folder1/checkpoint-1500
Configuration saved in /content/Untitled Folder1/checkpoint-1500/config.json
Configuration saved in /content/Untitled Folder1/checkpoint-1500/generation_c
onfig.json
Model weights saved in /content/Untitled Folder1/checkpoint-1500/pytorch_mode
l.bin
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8
Saving model checkpoint to /content/Untitled Folder1/checkpoint-2000
Configuration saved in /content/Untitled Folder1/checkpoint-2000/config.json
Configuration saved in /content/Untitled Folder1/checkpoint-2000/generation_c
onfig.json
Model weights saved in /content/Untitled Folder1/checkpoint-2000/pytorch_mode
l.bin
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8
Saving model checkpoint to /content/Untitled Folder1/checkpoint-2500
Configuration saved in /content/Untitled Folder1/checkpoint-2500/config.json
Configuration saved in /content/Untitled Folder1/checkpoint-2500/generation_c
onfig.json
Model weights saved in /content/Untitled Folder1/checkpoint-2500/pytorch_mode
l.bin
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8
Saving model checkpoint to /content/Untitled Folder1/checkpoint-3000
Configuration saved in /content/Untitled Folder1/checkpoint-3000/config.json
Configuration saved in /content/Untitled Folder1/checkpoint-3000/generation_c
onfig.json
Model weights saved in /content/Untitled Folder1/checkpoint-3000/pytorch_mode
l.bin
```

```
***** Running Evaluation *****
  Num examples = 1600
  Batch size = 8


Training completed. Do not forget to share your model on huggingface.co/model
s =)
```

Out[21]:
```
TrainOutput(global_step=3200, training_loss=2.1018737316131593, metrics={'tra
in_runtime': 323.427, 'train_samples_per_second': 158.305, 'train_steps_per_s
econd': 9.894, 'total_flos': 222102238003200.0, 'train_loss': 2.1018737316131
593, 'epoch': 8.0})
```

In [ ]:
```
outputs = []
for i, sample in enumerate(X_test):
  inputs = tokenizer.encode(sample, return_tensors = 'pt').cuda()
  greedy_output = model.generate(inputs, top_p = 0.5, top_k = 0, temperature =
  outputs.append(tokenizer.decode(greedy_output[0], skip_special_tokens = True
```

```
Streaming output truncated to the last 5000 lines.
Generate config GenerationConfig {
  "bos_token_id": 50256,
  "eos_token_id": 50256,
  "transformers_version": "4.26.1"
}

The attention mask and the pad token id were not set. As a consequence, yo
u may observe unexpected behavior. Please pass your input's `attention_mas
k` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generate config GenerationConfig {
  "bos_token_id": 50256,
  "eos_token_id": 50256,
  "transformers_version": "4.26.1"
}

The attention mask and the pad token id were not set. As a consequence, yo
u may observe unexpected behavior. Please pass your input's `attention_mas
```

In [ ]:
```
blueScores = 0
for i, pred in enumerate(outputs):
  blueScores += sentence_bleu([X_test[i].split()], pred.split())

print(blueScores/len(outputs))
```

```
/usr/local/lib/python3.8/dist-packages/nltk/translate/bleu_score.py:552: User
Warning:
The hypothesis contains 0 counts of 4-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)

0.6551512826369491

/usr/local/lib/python3.8/dist-packages/nltk/translate/bleu_score.py:552: User
Warning:
The hypothesis contains 0 counts of 3-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
```

In [ ]: