CS 505 Homework 06: Transformers

Problem Three

See the problem one notebook for details on due date, submission, etc.

As with problems one and two , there is an extensive tutorial section, followed by some tasks at the end you need to complete.

Full Disclosure: This is based on Rey Farhan's post:

(https://reyfarhan.com/posts/easy-gpt2-finetuning-huggingface/)

Introduction

In this problem, we will learn about fine-tuning GPT2 using Hugging Face's Transformers library and PyTorch on raw data.

We start with a simplified script for fine-tuning GPT-2, and at the end, your task will be to modify this assignment on raw text of your choice (I have put Pride and Prejudice as an example below) and include 10 sample generations from your chosen text that you find interesting.

Setup

```
!pip install transformers
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting transformers
  Downloading transformers-4.36.2-py3-none-any.whl.metadata (126 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 126.8/126.8 kB 8.9 MB/s eta 0:00
Requirement already satisfied: filelock in /share/pkg.8/python3/3.10.12/insta
Requirement already satisfied: huggingface-hub<1.0,>=0.19.3 in /usr4/cs640/ga
Requirement already satisfied: numpy>=1.17 in /share/pkg.8/python3/3.10.12/in
Requirement already satisfied: packaging>=20.0 in /share/pkg.8/python3/3.10.1
Requirement already satisfied: pyyaml>=5.1 in /share/pkg.8/python3/3.10.12/in
Requirement already satisfied: regex!=2019.12.17 in /share/pkg.8/python3/3.10
Requirement already satisfied: requests in /share/pkg.8/python3/3.10.12/insta
Collecting tokenizers<0.19,>=0.14 (from transformers)
  Downloading tokenizers-0.15.0-cp310-cp310-manylinux_2_17_x86_64.manylinux201
Requirement already satisfied: safetensors>=0.3.1 in /usr4/cs640/gaurav57/.lo
Requirement already satisfied: tqdm>=4.27 in /share/pkg.8/python3/3.10.12/ins
Requirement already satisfied: fsspec>=2023.5.0 in /share/pkg.8/python3/3.10.
Requirement already satisfied: typing-extensions>=3.7.4.3 in /share/pkg.8/pyt
Requirement already satisfied: charset-normalizer<4,>=2 in /share/pkg.8/pytho
Requirement already satisfied: idna<4,>=2.5 in /share/pkg.8/python3/3.10.12/i
Requirement already satisfied: urllib3<3,>=1.21.1 in /share/pkg.8/python3/3.1
```

```
Requirement already satisfied: certifi>=2017.4.17 in /share/pkg.8/python3/3.1(
Downloading transformers-4.36.2-py3-none-any.whl (8.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.2/8.2 MB 150.2 MB/s eta 0:00:00a
Downloading tokenizers-0.15.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.8/3.8 MB 213.2 MB/s eta 0:00:00
Installing collected packages: tokenizers, transformers
  WARNING: The script transformers-cli is installed in '/usr4/cs640/gaurav57/
  Consider adding this directory to PATH or, if you prefer to suppress this wa
Successfully installed tokenizers-0.15.0 transformers-4.36.2

[notice] A new release of pip is available: 23.3.1 -> 23.3.2
[notice] To update, run: pip install --upgrade pip
```

```
!pip install matplotlib
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: matplotlib in /share/pkg.8/python3/3.10.12/inst
Requirement already satisfied: contourpy>=1.0.1 in /share/pkg.8/python3/3.10.1
Requirement already satisfied: cycler>=0.10 in /share/pkg.8/python3/3.10.12/in
Requirement already satisfied: fonttools>=4.22.0 in /share/pkg.8/python3/3.10
Requirement already satisfied: kiwisolver>=1.0.1 in /share/pkg.8/python3/3.10
Requirement already satisfied: numpy>=1.20 in /share/pkg.8/python3/3.10.12/ins
Requirement already satisfied: packaging>=20.0 in /share/pkg.8/python3/3.10.12
Requirement already satisfied: pillow>=6.2.0 in /share/pkg.8/python3/3.10.12/i
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in /share/pkg.8/python3/3
Requirement already satisfied: python-dateutil>=2.7 in /share/pkg.8/python3/3
Requirement already satisfied: six>=1.5 in /share/pkg.8/python3/3.10.12/instal

[notice] A new release of pip is available: 23.3.1 -> 23.3.2
[notice] To update, run: pip install --upgrade pip
```

```
import os
import time
import datetime
# from google.colab import drive

import pandas as pd
import seaborn as sns
import numpy as np
import random

import matplotlib.pyplot as plt
# % matplotlib inline

import torch
from torch.utils.data import Dataset, DataLoader, random_split, RandomSampler, Se
torch.manual_seed(42)

from transformers import GPT2LMHeadModel,  GPT2Tokenizer, GPT2Config, GPT2LMHeadM
from transformers import AdamW, get_linear_schedule_with_warmup

import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /usr4/cs640/gaurav57/nltk_data...
```

```
[nltk_data]   Package punkt is already up-to-date!
True
```

```
!nvidia-smi
```

```
Thu Dec 21 18:07:16 2023
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.129.03             Driver Version: 535.129.03   CUDA Version
|-----------------------------------------+----------------------+-----------
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile U
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  C
|                                         |                      |
|=========================================+======================+===========
|   0  Tesla V100-SXM2-16GB           On  | 00000000:1A:00.0 Off |
| N/A   53C    P0              77W / 300W |   9047MiB / 16384MiB |      0%   D
|                                         |                      |
+-----------------------------------------+----------------------+-----------
|   1  Tesla V100-SXM2-16GB           On  | 00000000:1C:00.0 Off |
| N/A   48C    P0              77W / 300W |   1289MiB / 16384MiB |     25%   D
|                                         |                      |
+-----------------------------------------+----------------------+-----------
|   2  Tesla V100-SXM2-16GB           On  | 00000000:1D:00.0 Off |
| N/A   38C    P0              42W / 300W |      0MiB / 16384MiB |      0%   D
|                                         |                      |
+-----------------------------------------+----------------------+-----------
|   3  Tesla V100-SXM2-16GB           On  | 00000000:1E:00.0 Off |
| N/A   43C    P0              44W / 300W |      0MiB / 16384MiB |      0%   D
|                                         |                      |
+-----------------------------------------+----------------------+-----------

+---------------------------------------------------------------------------------------+
| Processes:
|  GPU   GI   CI        PID   Type   Process name                             G
|        ID   ID                                                              U
|=======================================================================================
|    0   N/A  N/A    479936      C   python
|    1   N/A  N/A    485603      C   python
+---------------------------------------------------------------------------------------+
```

Create Training Set

```
# mount my Google Drive directory and access the training data located there
# gdrive_dir = '/content/gdrive/'
# data_dir = os.path.join(gdrive_dir, "'My Drive'")
# filename = 'prideAndPrejudice.txt'

# drive.mount(gdrive_dir, force_remount=True)
```

```
filename = '/projectnb/cs505ws/students/gaurav57/prideAndPrejudice.txt'
```

```
# copy the data to the current Colab working directory
# !cp $data_dir/$filename .


f=open(filename)
docs=f.readlines()
docs=[b.strip() for b in docs]
docs[:5]
```

```
['It is a truth universally acknowledged, that a single man in possession of
a good fortune, must be in want of a wife.',
 'However little known the feelings or views of such a man may be on his
first entering a neighbourhood, this truth is so well fixed in the minds of
the surrounding families, that he is considered the rightful property of some
one or other of their daughters.',
 '"My dear Mr. Bennet," said his lady to him one day, "have you heard that
Netherfield Park is let at last?"',
 'Mr. Bennet replied that he had not.',
 '"But it is," returned she; "for Mrs. Long has just been here, and she told
me all about it."']
```

We need to get an idea of how long our training documents are.

I'm not going to use the same tokenizer as the GPT2 one, which is a byte pair encoding tokenizer. Instead, I'm using a simple one just to get a rough understanding.

```
doc_lengths = []

for doc in docs:

    # get rough token count distribution
    tokens = nltk.word_tokenize(doc)

    doc_lengths.append(len(tokens))

doc_lengths = np.array(doc_lengths)

sns.distplot(doc_lengths)
```

```
/scratch/3116117.1.academic-gpu/ipykernel_498254/3600140024.py:12: UserWarning

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(doc_lengths)
<Axes: ylabel='Density'>
```


```
# the max token length
len(doc_lengths[doc_lengths > 768])/len(doc_lengths)
```

```
0.0004847309743092584
```

```
np.average(doc_lengths)
```

```
69.48763936015511
```

Even though these token counts won't match up to the BPE tokenizer's, I'm confident that most lines will be fit under the 768 embedding size limit for the small GPT2 model.

GPT2 Tokenizer

Although the defaults take care of this,I thought I'd show that you can specify some of the special tokens.

```
# Load the GPT tokenizer.
tokenizer = GPT2Tokenizer.from_pretrained('gpt2', bos_token='<|startoftext|>', eo
```

```
print("The max model length is {} for this model, although the actual embedding s
print("The beginning of sequence token {} token has the id {}".format(tokenizer.c
print("The end of sequence token {} has the id {}".format(tokenizer.convert_ids_t
print("The padding token {} has the id {}".format(tokenizer.convert_ids_to_tokens
```

```
The max model length is 1024 for this model, although the actual embedding si
The beginning of sequence token <|startoftext|> token has the id 50257
```

```
The end of sequence token <|endoftext|> has the id 50256
The padding token <|pad|> has the id 50258
```

PyTorch Datasets & Dataloaders

GPT2 is a large model. Increasing the batch size above 2 has lead to out of memory problems. This can be mitigated by accumulating the gradients but that is out of scope here.

```
batch_size = 2
```

I'm using the standard PyTorch approach of loading data in using a dataset class.

I'm passing in the tokenizer as an argument but normally I would instantiate it within the class.

```
class GPT2Dataset(Dataset):

  def __init__(self, txt_list, tokenizer, gpt2_type="gpt2", max_length=768):

    self.tokenizer = tokenizer
    self.input_ids = []
    self.attn_masks = []

    for txt in txt_list:

      encodings_dict = tokenizer('<|startoftext|>'+ txt + '<|endoftext|>', trunca

      self.input_ids.append(torch.tensor(encodings_dict['input_ids']))
      self.attn_masks.append(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]
```

To understand how I've used the tokenizer, it's worth reading the docs. I've wrapped each line in the bos and eos tokens.

Every tensor passed to the model should be the same length.

If the line is shorter than 768 tokens, it will be padded to a length of 768 using the padding token. In addition, an attention mask will be returned that needs to be passed to the model to tell it to ignore the padding tokens.

If the line is longer than 768 tokens, it will be truncated without the eos_token. This isn't a problem.

```
dataset = GPT2Dataset(docs, tokenizer, max_length=768)

# Split into training and validation sets
train_size = int(0.9 * len(dataset))
val_size = len(dataset) - train_size

train_dataset, val_dataset = random_split(dataset, [train_size, val_size])

print('{:>5,} training samples'.format(train_size))
print('{:>5,} validation samples'.format(val_size))
```

```
    1,856 training samples
      207 validation samples
```

```
# Create the DataLoaders for our training and validation datasets.
# We'll take training samples in random order.
train_dataloader = DataLoader(
            train_dataset,  # The training samples.
            sampler = RandomSampler(train_dataset), # Select batches randomly
            batch_size = batch_size # Trains with this batch size.
        )

# For validation the order doesn't matter, so we'll just read them sequentially.
validation_dataloader = DataLoader(
            val_dataset, # The validation samples.
            sampler = SequentialSampler(val_dataset), # Pull out batches sequenti
            batch_size = batch_size # Evaluate with this batch size.
        )
```

Finetune GPT2 Language Model

```
# I'm not really doing anything with the config buheret
configuration = GPT2Config.from_pretrained('gpt2', output_hidden_states=False)

# instantiate the model
model = GPT2LMHeadModel.from_pretrained("gpt2", config=configuration)

# this step is necessary because I've added some tokens (bos_token, etc) to the e
# otherwise the tokenizer and model tensors won't match up
model.resize_token_embeddings(len(tokenizer))

# Tell pytorch to run this model on the GPU.
device = torch.device("cuda")
model.cuda()

# Set the seed value all over the place to make this reproducible.
seed_val = 42

random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
torch.cuda.manual_seed_all(seed_val)


# some parameters I cooked up that work reasonably well

epochs = 5
learning_rate = 5e-4
warmup_steps = 1e2
epsilon = 1e-8

# this produces sample output every 100 steps
sample_every = 100


# Note: AdamW is a class from the huggingface library (as opposed to pytorch)
optimizer = AdamW(model.parameters(),
                  lr = learning_rate,
                  eps = epsilon
                )

     /usr4/cs640/gaurav57/.local/lib/python3.10/site-packages/transformers/optimiza
       warnings.warn(


# Total number of training steps is [number of batches] x [number of epochs].
# (Note that this is not the same as the number of training samples).
total_steps = len(train_dataloader) * epochs

# Create the learning rate scheduler.
# This changes the learning rate as the training loop progresses
scheduler = get_linear_schedule_with_warmup(optimizer,
                                            num_warmup_steps = warmup_steps,
                                            num_training_steps = total_steps)
```

```
def format_time(elapsed):
    return str(datetime.timedelta(seconds=int(round((elapsed)))))


from tqdm import tqdm
```

```
total_t0 = time.time()

training_stats = []

model = model.to(device)

for epoch_i in tqdm(range(0, epochs)):

    # ========================================
    #               Training
    # ========================================

    print("")
    print('======== Epoch {:} / {:} ========'.format(epoch_i + 1, epochs))
    print('Training...')

    t0 = time.time()

    total_train_loss = 0

    model.train()

    for step, batch in enumerate(train_dataloader):

        b_input_ids = batch[0].to(device)
        b_labels = batch[0].to(device)
        b_masks = batch[1].to(device)

        model.zero_grad()

        outputs = model(  b_input_ids,
                          labels=b_labels,
                          attention_mask = b_masks,
                          token_type_ids=None
                        )

        loss = outputs[0]

        batch_loss = loss.item()
        total_train_loss += batch_loss

        # Get sample every x batches.
        if step % sample_every == 0 and not step == 0:

            elapsed = format_time(time.time() - t0)
            print('  Batch {:>5,}  of  {:>5,}. Loss: {:>5,}.   Elapsed: {:}.'.for

            model.eval()

            sample_outputs = model.generate(
                                    bos_token_id=random.randint(1,30000),
                                    do_sample=True,
                                    top_k=50,
                                    max_length = 200,
                                    top_p=0.95,
```

```
                                    num_return_sequences=1
                                )
            for i, sample_output in enumerate(sample_outputs):
                  print("{}: {}".format(i, tokenizer.decode(sample_output, skip_s

            model.train()

        loss.backward()

        optimizer.step()

        scheduler.step()

    # Calculate the average loss over all of the batches.
    avg_train_loss = total_train_loss / len(train_dataloader)

    # Measure how long this epoch took.
    training_time = format_time(time.time() - t0)

    print("")
    print("  Average training loss: {0:.2f}".format(avg_train_loss))
    print("  Training epoch took: {:}".format(training_time))

    # ========================================
    #               Validation
    # ========================================

    print("")
    print("Running Validation...")

    t0 = time.time()

    model.eval()

    total_eval_loss = 0
    nb_eval_steps = 0

    # Evaluate data for one epoch
    for batch in validation_dataloader:

        b_input_ids = batch[0].to(device)
        b_labels = batch[0].to(device)
        b_masks = batch[1].to(device)

        with torch.no_grad():

            outputs  = model(b_input_ids,
#                            token_type_ids=None,
                            attention_mask = b_masks,
                           labels=b_labels)

            loss = outputs[0]

        batch_loss = loss.item()
        total_eval_loss += batch_loss
```

```
        avg_val_loss = total_eval_loss / len(validation_dataloader)

        validation_time = format_time(time.time() - t0)

        print("  Validation Loss: {0:.2f}".format(avg_val_loss))
        print("  Validation took: {:}".format(validation_time))

        # Record all statistics from this epoch.
        training_stats.append(
            {
                'epoch': epoch_i + 1,
                'Training Loss': avg_train_loss,
                'Valid. Loss': avg_val_loss,
                'Training Time': training_time,
                'Validation Time': validation_time
            }
        )

    print("")
    print("Training complete!")
    print("Total training took {:} (h:mm:ss)".format(format_time(time.time()-total_t0
```

```
      0%|          | 0/5 [00:00<?, ?it/s]
    ======== Epoch 1 / 5 ========
    Training...
    The attention mask and the pad token id were not set. As a consequence, you ma
    Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
      Batch   100  of    928. Loss: 0.4344193637371063.   Elapsed: 0:00:19.
    0:  bipartisan" and "but had never seen the face of either any of their party-
    The attention mask and the pad token id were not set. As a consequence, you ma
    Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
      Batch   200  of    928. Loss: 0.241342231631279.   Elapsed: 0:00:37.
    0:  increasingA he had not yet gained his sight of Lizzy's appearance when she
    The attention mask and the pad token id were not set. As a consequence, you ma
    Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
      Batch   300  of    928. Loss: 0.3199116289615631.   Elapsed: 0:00:53.
    0: day"And there were several days before it was settled, Mr. Bingley, who was
    The attention mask and the pad token id were not set. As a consequence, you ma
    Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
      Batch   400  of    928. Loss: 0.35124173760414124.   Elapsed: 0:01:10.
    0:  Hang"My dear and dear, my dear, I have no time for this," said Charlotte,
    The attention mask and the pad token id were not set. As a consequence, you ma
    Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
      Batch   500  of    928. Loss: 0.31730371713638306.   Elapsed: 0:01:27.
    0:  foodsElizabeth and Kitty began walking towards Rosings; but though there w
    The attention mask and the pad token id were not set. As a consequence, you ma
    Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
      Batch   600  of    928. Loss: 0.16978195309638977.   Elapsed: 0:01:43.
    0:  trail"There was a great deal to say before Mr. Bennet turned away from us
    The attention mask and the pad token id were not set. As a consequence, you ma
    Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
      Batch   700  of    928. Loss: 0.2888268232345581.   Elapsed: 0:01:59.
    0: intendThe whole country was then engaged in the execution of her master's p
    The attention mask and the pad token id were not set. As a consequence, you ma
    Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
      Batch   800  of    928. Loss: 0.3050519526004791.   Elapsed: 0:02:15.
```

```
0:  surroundElizabeth began his apology for his indifference on Monday.
The attention mask and the pad token id were not set. As a consequence, you ma
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
  Batch   900  of    928. Loss: 0.28032082319259644.   Elapsed: 0:02:31.
0:  reflex"I must think so, though Mr. Collins says he cannot have done more t

  Average training loss: 0.51
  Training epoch took: 0:02:36

Running Validation...
 20%|██        | 1/5 [02:40<10:43, 160.77s/it]  Validation Loss: 0.32
  Validation took: 0:00:05

======== Epoch 2 / 5 ========
Training...
The attention mask and the pad token id were not set. As a consequence, you ma
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
  Batch   100  of    928. Loss: 0.0703025534749031.   Elapsed: 0:00:16.
0:  displayTheir carriage was hastened by the sound of some distant voice. Mrs
The attention mask and the pad token id were not set. As a consequence, you ma
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
  Batch   200  of    928. Loss: 0.1618892401456833.   Elapsed: 0:00:32.
0:  pastor"This cannot be. It is a great deal of business."
The attention mask and the pad token id were not set. As a consequence, you ma
```

Let's view the summary of the training process.

```
# Display floats with two decimal places.
pd.set_option('display.precision', 2)

# Create a DataFrame from our training statistics.
df_stats = pd.DataFrame(data=training_stats)

# Use the 'epoch' as the row index.
df_stats = df_stats.set_index('epoch')

# A hack to force the column headers to wrap.
#df = df.style.set_table_styles([dict(selector="th",props=[('max-width', '70px')]

# Display the table.
df_stats
```

epoch	Training Loss	Valid. Loss	Training Time	Validation Time
1	0.51	0.32	0:02:36	0:00:05
2	0.30	0.32	0:02:28	0:00:05
3	0.25	0.33	0:02:29	0:00:05
4	0.20	0.35	0:02:27	0:00:05
5	0.15	0.39	0:02:28	0:00:05

```
# Use plot styling from seaborn.
sns.set(style='darkgrid')

# Increase the plot size and font size.
sns.set(font_scale=1.5)
plt.rcParams["figure.figsize"] = (12,6)

# Plot the learning curve.
plt.plot(df_stats['Training Loss'], 'b-o', label="Training")
plt.plot(df_stats['Valid. Loss'], 'g-o', label="Validation")

# Label the plot.
plt.title("Training & Validation Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.xticks([1, 2, 3, 4])

plt.show()
```

Display Model Info

```
# Get all of the model's parameters as a list of tuples.
params = list(model.named_parameters())

print('The GPT-2 model has {:} different named parameters.\n'.format(len(params))

print('==== Embedding Layer ====\n')

for p in params[0:2]:
    print("{:<55} {:>12}".format(p[0], str(tuple(p[1].size()))))

print('\n==== First Transformer ====\n')

for p in params[2:14]:
    print("{:<55} {:>12}".format(p[0], str(tuple(p[1].size()))))

print('\n==== Output Layer ====\n')

for p in params[-2:]:
    print("{:<55} {:>12}".format(p[0], str(tuple(p[1].size()))))
```

```
The GPT-2 model has 148 different named parameters.

==== Embedding Layer ====

transformer.wte.weight                                (50259, 768)
transformer.wpe.weight                                 (1024, 768)

==== First Transformer ====

transformer.h.0.ln_1.weight                                (768,)
transformer.h.0.ln_1.bias                                  (768,)
transformer.h.0.attn.c_attn.weight                     (768, 2304)
transformer.h.0.attn.c_attn.bias                          (2304,)
transformer.h.0.attn.c_proj.weight                     (768, 768)
transformer.h.0.attn.c_proj.bias                           (768,)
transformer.h.0.ln_2.weight                                (768,)
transformer.h.0.ln_2.bias                                  (768,)
transformer.h.0.mlp.c_fc.weight                       (768, 3072)
transformer.h.0.mlp.c_fc.bias                            (3072,)
transformer.h.0.mlp.c_proj.weight                     (3072, 768)
transformer.h.0.mlp.c_proj.bias                            (768,)

==== Output Layer ====

transformer.ln_f.weight                                    (768,)
transformer.ln_f.bias                                      (768,)
```

Saving & Loading Fine-Tuned Model

```
# Saving best-practices: if you use defaults names for the model, you can reload

output_dir = './model_save/'

# Create output directory if needed
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

print("Saving model to %s" % output_dir)

# Save a trained model, configuration and tokenizer using `save_pretrained()`.
# They can then be reloaded using `from_pretrained()`
model_to_save = model.module if hasattr(model, 'module') else model  # Take care
model_to_save.save_pretrained(output_dir)
tokenizer.save_pretrained(output_dir)

# Good practice: save your training arguments together with the trained model
# torch.save(args, os.path.join(output_dir, 'training_args.bin'))
```

```
Saving model to ./model_save/
('./model_save/tokenizer_config.json',
 './model_save/special_tokens_map.json',
 './model_save/vocab.json',
 './model_save/merges.txt',
 './model_save/added_tokens.json')
```

```
!ls -l --block-size=K ./model_save/
```

```
total 487547K
-rw-r--r-- 1 gaurav57 cs505wsta      1K Dec 21 16:23 added_tokens.json
-rw-r--r-- 1 gaurav57 cs505wsta      1K Dec 21 16:23 config.json
-rw-r--r-- 1 gaurav57 cs505wsta      1K Dec 21 16:23 generation_config.json
-rw-r--r-- 1 gaurav57 cs505wsta    446K Dec 21 16:23 merges.txt
-rw-r--r-- 1 gaurav57 cs505wsta 486114K Dec 21 16:23 model.safetensors
-rw-r--r-- 1 gaurav57 cs505wsta      1K Dec 21 16:23 special_tokens_map.json
-rw-r--r-- 1 gaurav57 cs505wsta      1K Dec 21 16:23 tokenizer_config.json
-rw-r--r-- 1 gaurav57 cs505wsta    976K Dec 21 16:23 vocab.json
```

```
!ls -l --block-size=M ./model_save/pytorch_model.bin
```

```
ls: cannot access './model_save/pytorch_model.bin': No such file or directory
```

```
# Copy the model files to a directory in your Google Drive.
!cp -r ./model_save/ $data_dir

# # Load a trained model and vocabulary that you have fine-tuned
#model = GPT2LMHeadModel.from_pretrained(output_dir)
#tokenizer = GPT2Tokenizer.from_pretrained(output_dir)
#model.to(device)
```

Generate Text

```
model.eval()

prompt = "<|startoftext|>"

generated = torch.tensor(tokenizer.encode(prompt)).unsqueeze(0)
generated = generated.to(device)

print(generated)

sample_outputs = model.generate(
                                generated,
                                #bos_token_id=random.randint(1,30000),
                                do_sample=True,
                                top_k=50,
                                max_length = 300,
                                top_p=0.95,
                                num_return_sequences=3
                                )

for i, sample_output in enumerate(sample_outputs):
  print("{}: {}\n\n".format(i, tokenizer.decode(sample_output, skip_special_token
```

```
The attention mask and the pad token id were not set. As a consequence, you ma
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
tensor([[50257]], device='cuda:0')
0: "It may be a little," said Mrs. Bennet, "to name a few elegant names for a


1: "I do not know whether you agree with me in that respect. But my situation


2: "I can readily believe," said he, "that such a lady does not know Mrs. Long
```

```
model.eval()

prompt = "<|startoftext|>"

generated = torch.tensor(tokenizer.encode(prompt)).unsqueeze(0)
generated = generated.to(device)

print(generated)

sample_outputs = model.generate(
                                generated,
                                #bos_token_id=random.randint(1,30000),
                                do_sample=True,
                                top_k=50,
                                max_length = 300,
                                top_p=0.95,
                                num_return_sequences=3
                                )

for i, sample_output in enumerate(sample_outputs):
  print("{}: {}\n\n".format(i, tokenizer.decode(sample_output, skip_special_token
```

```
The attention mask and the pad token id were not set. As a consequence, you ma
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
tensor([[50257]], device='cuda:0')
0: "And, by and by," replied her mother, "your ladyship has given you credit a


1: "Lizzy, you ought to go and see Jane," said her mother resentfully.


2: "But these are heavy misfortunes," said Elizabeth with a sigh. "We must al
```

YOUR TURN!

These aren't bad at all! Now train the model on your chosen raw text that is roughly comparable in size to pride and prejudice.

There are two things you need to do:

- Draw a figure tracking the training and validation losses as in previous homeworks.
- Print out some sample text from your chosen data and report 10 example generations that you think are interesting! Do your examples look like your training text?

> I'll be working on the Great Gatsby available in gutenberg (https://www.gutenberg.org/ebooks/64317)

```
nltk.download('gutenberg')
```

```
[nltk_data] Downloading package gutenberg to
[nltk_data]     /usr4/cs640/gaurav57/nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
True
```

```
import urllib.request
from nltk.tokenize import sent_tokenize
import nltk

# Make sure you have the 'punkt' tokenizer resource available
nltk.download('punkt')

# Open the text file from Project Gutenberg
text_file = urllib.request.urlopen("https://www.gutenberg.org/cache/epub/64317/pg

# Read and decode the text
text = text_file.read().decode('utf-8')
clean_text = text.replace('\r', '').replace('\n', ' ').replace('\t', ' ')
# Tokenize the text into sentences
docs = sent_tokenize(clean_text)

# Now 'sentences' is an array of sentences from the text
print(docs[:15])  # print the first 10 sentences to check
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /usr4/cs640/gaurav57/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
['\ufeffThe Project Gutenberg eBook of The Great Gatsby      This ebook is fo
```

```
print(docs[10:11])
```

```
['And, after boasting this way of my tolerance, I come to the admission that :
```

```
# sentences
```

```
len(docs)
```

```
2560
```

```
doc_lengths = []

for doc in docs:

    # get rough token count distribution
    tokens = nltk.word_tokenize(doc)

    doc_lengths.append(len(tokens))

doc_lengths = np.array(doc_lengths)

sns.distplot(doc_lengths)
```

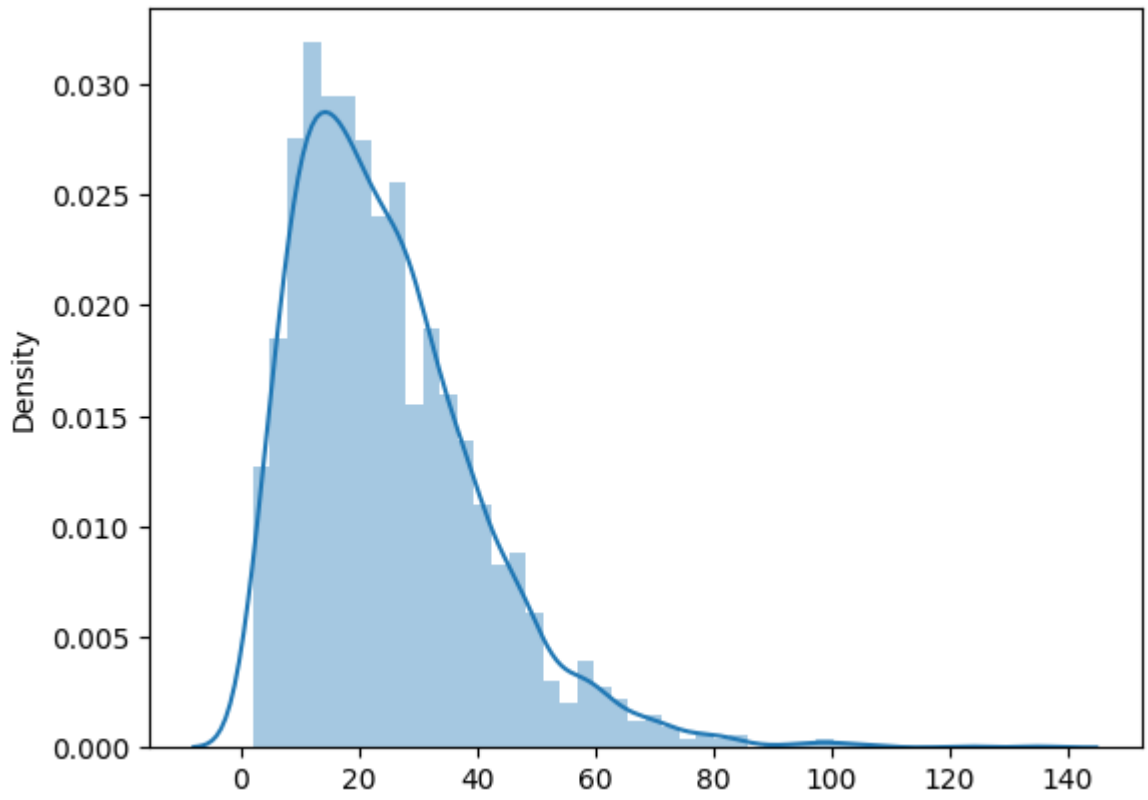
```
/scratch/3127120.1.jchengroup-pub/ipykernel_507450/3600140024.py:12: UserWarn

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(doc_lengths)
<Axes: ylabel='Density'>
```




```
# the max token length
len(doc_lengths[doc_lengths > 768])/len(doc_lengths)
```

```
0.0
```

```
np.average(doc_lengths)
```

```
24.989453125
```

```
#Considerably very short sentences as compared to Pride and Prejudice
```

```
# GPT2 Tokenizer
# Taking gpt2 and gpt2-medium takes a more computational time
```

```
tokenizer = GPT2Tokenizer.from_pretrained('gpt2', bos_token='<|startoftext|>', eo
```

```
print("The max model length is {} for this model, although the actual embedding s
print("The beginning of sequence token {} token has the id {}".format(tokenizer.c
print("The end of sequence token {} has the id {}".format(tokenizer.convert_ids_t
print("The padding token {} has the id {}".format(tokenizer.convert_ids_to_tokens
```

```
The max model length is 1024 for this model, although the actual embedding si
The beginning of sequence token <|startoftext|> token has the id 50257
The end of sequence token <|endoftext|> has the id 50256
The padding token <|pad|> has the id 50258
```

```
# PyTorch Datasets & Dataloaders


batch_size = 2


class GPT2Dataset(Dataset):

  def __init__(self, txt_list, tokenizer, gpt2_type="gpt2", max_length=768):

    self.tokenizer = tokenizer
    self.input_ids = []
    self.attn_masks = []

    for txt in txt_list:

      encodings_dict = tokenizer('<|startoftext|>'+ txt + '<|endoftext|>', trunca

      self.input_ids.append(torch.tensor(encodings_dict['input_ids']))
      self.attn_masks.append(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]


dataset = GPT2Dataset(docs, tokenizer, max_length=768)

# Split into training and validation sets
train_size = int(0.9 * len(dataset))
val_size = len(dataset) - train_size

train_dataset, val_dataset = random_split(dataset, [train_size, val_size])

print('{:>5,} training samples'.format(train_size))
print('{:>5,} validation samples'.format(val_size))
```

```
2,304 training samples
  256 validation samples
```

```
# Create the DataLoaders for our training and validation datasets.
# We'll take training samples in random order.
train_dataloader = DataLoader(
            train_dataset,  # The training samples.
            sampler = RandomSampler(train_dataset), # Select batches randomly
            batch_size = batch_size # Trains with this batch size.
        )

# For validation the order doesn't matter, so we'll just read them sequentially.
validation_dataloader = DataLoader(
            val_dataset, # The validation samples.
            sampler = SequentialSampler(val_dataset), # Pull out batches sequenti
            batch_size = batch_size # Evaluate with this batch size.
        )


# Finetune GPT2 Language Model


# I'm not really doing anything with the config buheret
configuration = GPT2Config.from_pretrained('gpt2', output_hidden_states=False)

# instantiate the model
model = GPT2LMHeadModel.from_pretrained("gpt2", config=configuration)

# this step is necessary because I've added some tokens (bos_token, etc) to the e
# otherwise the tokenizer and model tensors won't match up
model.resize_token_embeddings(len(tokenizer))

# Tell pytorch to run this model on the GPU.
device = torch.device("cuda")
model.cuda()

# Set the seed value all over the place to make this reproducible.
seed_val = 42

random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
torch.cuda.manual_seed_all(seed_val)


# learning_rate = 5e-3 # found this to give better sentences


# some parameters I cooked up that work reasonably well

epochs = 7
learning_rate = 5e-4
warmup_steps = 1e2
epsilon = 1e-8

# this produces sample output every 100 steps
sample_every = 100
```

```
# Note: AdamW is a class from the huggingface library (as opposed to pytorch)
optimizer = AdamW(model.parameters(),
                  lr = learning_rate,
                  eps = epsilon
                )
```

```
/usr4/cs640/gaurav57/.local/lib/python3.10/site-packages/transformers/optimiza
  warnings.warn(
```

```
# Total number of training steps is [number of batches] x [number of epochs].
# (Note that this is not the same as the number of training samples).
total_steps = len(train_dataloader) * epochs

# Create the learning rate scheduler.
# This changes the learning rate as the training loop progresses
scheduler = get_linear_schedule_with_warmup(optimizer,
                                            num_warmup_steps = warmup_steps,
                                            num_training_steps = total_steps)
```

```
def format_time(elapsed):
    return str(datetime.timedelta(seconds=int(round((elapsed)))))
```

```
from tqdm import tqdm
```

```
total_t0 = time.time()

training_stats = []

model = model.to(device)

for epoch_i in tqdm(range(0, epochs)):

    # ========================================
    #               Training
    # ========================================

    print("")
    print('======== Epoch {:} / {:} ========'.format(epoch_i + 1, epochs))
    print('Training...')

    t0 = time.time()

    total_train_loss = 0

    model.train()

    for step, batch in enumerate(train_dataloader):

        b_input_ids = batch[0].to(device)
        b_labels = batch[0].to(device)
        b_masks = batch[1].to(device)

        model.zero_grad()

        outputs = model(  b_input_ids,
                          labels=b_labels,
                          attention_mask = b_masks,
                          token_type_ids=None
                        )

        loss = outputs[0]

        batch_loss = loss.item()
        total_train_loss += batch_loss

        # Get sample every x batches.
        if step % sample_every == 0 and not step == 0:

            elapsed = format_time(time.time() - t0)
            print('  Batch {:>5,}  of  {:>5,}. Loss: {:>5,}.   Elapsed: {:}.'.for

            model.eval()

            sample_outputs = model.generate(
                                    bos_token_id=random.randint(1,30000),
                                    do_sample=True,
                                    top_k=50,
                                    max_length = 200,
                                    top_p=0.95,
```

```
                                    num_return_sequences=1
                                )
            for i, sample_output in enumerate(sample_outputs):
                  print("{}: {}".format(i, tokenizer.decode(sample_output, skip_s

            model.train()

        loss.backward()

        optimizer.step()

        scheduler.step()

    # Calculate the average loss over all of the batches.
    avg_train_loss = total_train_loss / len(train_dataloader)

    # Measure how long this epoch took.
    training_time = format_time(time.time() - t0)

    print("")
    print("  Average training loss: {0:.2f}".format(avg_train_loss))
    print("  Training epoch took: {:}".format(training_time))

    # ========================================
    #               Validation
    # ========================================

    print("")
    print("Running Validation...")

    t0 = time.time()

    model.eval()

    total_eval_loss = 0
    nb_eval_steps = 0

    # Evaluate data for one epoch
    for batch in validation_dataloader:

        b_input_ids = batch[0].to(device)
        b_labels = batch[0].to(device)
        b_masks = batch[1].to(device)

        with torch.no_grad():

            outputs  = model(b_input_ids,
#                            token_type_ids=None,
                            attention_mask = b_masks,
                           labels=b_labels)

            loss = outputs[0]

        batch_loss = loss.item()
        total_eval_loss += batch_loss
```

```
        avg_val_loss = total_eval_loss / len(validation_dataloader)

        validation_time = format_time(time.time() - t0)

        print("  Validation Loss: {0:.2f}".format(avg_val_loss))
        print("  Validation took: {:}".format(validation_time))

        # Record all statistics from this epoch.
        training_stats.append(
            {
                'epoch': epoch_i + 1,
                'Training Loss': avg_train_loss,
                'Valid. Loss': avg_val_loss,
                'Training Time': training_time,
                'Validation Time': validation_time
            }
        )

    print("")
    print("Training complete!")
    print("Total training took {:} (h:mm:ss)".format(format_time(time.time()-total_t0
```

```
0:  beneficiariesThey'll all go to bed.
The attention mask and the pad token id were not set. As a consequence, you ma
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
  Batch   900  of  1,152. Loss: 0.047483887523412704.   Elapsed: 0:02:22.
0:  Title“It’s a bona-fide piece of printed matter,” I said.
The attention mask and the pad token id were not set. As a consequence, you ma
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
  Batch 1,000  of  1,152. Loss: 0.03598104044795036.   Elapsed: 0:02:38.
0:  µYou can’t live forever; you can’t live forever.”  He shook his head.
The attention mask and the pad token id were not set. As a consequence, you ma
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
  Batch 1,100  of  1,152. Loss: 0.015882272273302078.   Elapsed: 0:02:54.
0:  selling“Why, I thought, isn’t that any police dog?

  Average training loss: 0.04
  Training epoch took: 0:03:02

Running Validation...
100%|██████████| 7/7 [22:07<00:00, 189.70s/it]  Validation Loss: 0.21
  Validation took: 0:00:06

Training complete!
Total training took 0:22:08 (h:mm:ss)
```

```
# Display floats with two decimal places.
pd.set_option('display.precision', 2)

# Create a DataFrame from our training statistics.
df_stats = pd.DataFrame(data=training_stats)

# Use the 'epoch' as the row index.
df_stats = df_stats.set_index('epoch')

# A hack to force the column headers to wrap.
#df = df.style.set_table_styles([dict(selector="th",props=[('max-width', '70px')]

# Display the table.
df_stats
```

epoch	Training Loss	Valid. Loss	Training Time	Validation Time
1	0.21	0.13	0:03:06	0:00:06
2	0.11	0.13	0:03:03	0:00:06
3	0.15	0.18	0:03:07	0:00:06
4	0.10	0.16	0:03:03	0:00:06
5	0.07	0.17	0:03:02	0:00:06
6	0.05	0.19	0:03:01	0:00:06
7	0.04	0.21	0:03:02	0:00:06

```
# Use plot styling from seaborn.
sns.set(style='darkgrid')

# Increase the plot size and font size.
sns.set(font_scale=1.5)
plt.rcParams["figure.figsize"] = (12,6)

# Plot the learning curve.
plt.plot(df_stats['Training Loss'], 'b-o', label="Training")
plt.plot(df_stats['Valid. Loss'], 'g-o', label="Validation")

# Label the plot.
plt.title("Training & Validation Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.xticks([1, 2, 3, 4])

plt.show()
```


```
# Display Model Info
```

```
# Get all of the model's parameters as a list of tuples.
params = list(model.named_parameters())

print('The GPT-2 model has {:} different named parameters.\n'.format(len(params))

print('==== Embedding Layer ====\n')

for p in params[0:2]:
    print("{:<55} {:>12}".format(p[0], str(tuple(p[1].size()))))

print('\n==== First Transformer ====\n')

for p in params[2:14]:
    print("{:<55} {:>12}".format(p[0], str(tuple(p[1].size()))))

print('\n==== Output Layer ====\n')

for p in params[-2:]:
    print("{:<55} {:>12}".format(p[0], str(tuple(p[1].size()))))
```

```
The GPT-2 model has 148 different named parameters.

==== Embedding Layer ====

transformer.wte.weight                                (50259, 768)
transformer.wpe.weight                                 (1024, 768)

==== First Transformer ====

transformer.h.0.ln_1.weight                                (768,)
transformer.h.0.ln_1.bias                                  (768,)
transformer.h.0.attn.c_attn.weight                     (768, 2304)
transformer.h.0.attn.c_attn.bias                          (2304,)
transformer.h.0.attn.c_proj.weight                     (768, 768)
transformer.h.0.attn.c_proj.bias                           (768,)
transformer.h.0.ln_2.weight                                (768,)
transformer.h.0.ln_2.bias                                  (768,)
transformer.h.0.mlp.c_fc.weight                       (768, 3072)
transformer.h.0.mlp.c_fc.bias                            (3072,)
transformer.h.0.mlp.c_proj.weight                     (3072, 768)
transformer.h.0.mlp.c_proj.bias                            (768,)

==== Output Layer ====

transformer.ln_f.weight                                    (768,)
transformer.ln_f.bias                                      (768,)
```

```
# Saving & Loading Fine-Tuned Model
```

```
# Saving best-practices: if you use defaults names for the model, you can reload

output_dir = './model_save/'

# Create output directory if needed
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

print("Saving model to %s" % output_dir)

# Save a trained model, configuration and tokenizer using `save_pretrained()`.
# They can then be reloaded using `from_pretrained()`
model_to_save = model.module if hasattr(model, 'module') else model  # Take care
model_to_save.save_pretrained(output_dir)
tokenizer.save_pretrained(output_dir)

# Good practice: save your training arguments together with the trained model
# torch.save(args, os.path.join(output_dir, 'training_args.bin'))
```

```
Saving model to ./model_save/
('./model_save/tokenizer_config.json',
 './model_save/special_tokens_map.json',
 './model_save/vocab.json',
 './model_save/merges.txt',
 './model_save/added_tokens.json')
```

```
!ls -l --block-size=K ./model_save/
```

```
total 487547K
-rw-r--r-- 1 gaurav57 cs505wsta      1K Dec 21 19:12 added_tokens.json
-rw-r--r-- 1 gaurav57 cs505wsta      1K Dec 21 19:12 config.json
-rw-r--r-- 1 gaurav57 cs505wsta      1K Dec 21 19:12 generation_config.json
-rw-r--r-- 1 gaurav57 cs505wsta    446K Dec 21 19:12 merges.txt
-rw-r--r-- 1 gaurav57 cs505wsta 486114K Dec 21 19:12 model.safetensors
-rw-r--r-- 1 gaurav57 cs505wsta      1K Dec 21 19:12 special_tokens_map.json
-rw-r--r-- 1 gaurav57 cs505wsta      1K Dec 21 19:12 tokenizer_config.json
-rw-r--r-- 1 gaurav57 cs505wsta    976K Dec 21 19:12 vocab.json
```

```
# Generate Text
```

```
model.eval()

prompt = "<|startoftext|>"

generated = torch.tensor(tokenizer.encode(prompt)).unsqueeze(0)
generated = generated.to(device)

print(generated)

sample_outputs = model.generate(
                                generated,
                                #bos_token_id=random.randint(1,30000),
                                do_sample=True,
                                top_k=50,
                                max_length = 300,
                                top_p=0.95,
                                num_return_sequences=3
                                )

for i, sample_output in enumerate(sample_outputs):
  print("{}: {}\n\n".format(i, tokenizer.decode(sample_output, skip_special_token
```

```
The attention mask and the pad token id were not set. As a consequence, you ma
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
tensor([[50257]], device='cuda:0')
0: “You ought to have something to drink.”  “I almost made a mistake.


1: I was thirty.”  Before I could reply that I had dined with a girl of about


2: “In the ditch beside the road, if you want to go to take a plunge in the sw
```

```
model.eval()

prompt = "<|startoftext|>"

generated = torch.tensor(tokenizer.encode(prompt)).unsqueeze(0)
generated = generated.to(device)

print(generated)

sample_outputs = model.generate(
                                generated,
                                #bos_token_id=random.randint(1,30000),
                                do_sample=True,
                                top_k=50,
                                max_length = 300,
                                top_p=0.95,
                                num_return_sequences=3
                                )

for i, sample_output in enumerate(sample_outputs):
  print("{}: {}\n\n".format(i, tokenizer.decode(sample_output, skip_special_token
```

```
The attention mask and the pad token id were not set. As a consequence, you ma
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
tensor([[50257]], device='cuda:0')
0: “You sounded well enough on the phone.”  “Well, we were almost the last to


1: The following sentence, with active links to, or other immediate access to


2: “It must be a madman.”  “He’s mad!”  “Yes … Yes … Yes … Yes … Yes … there’
```

```
model.eval()

prompt = "<|startoftext|>"

generated = torch.tensor(tokenizer.encode(prompt)).unsqueeze(0)
generated = generated.to(device)

print(generated)

sample_outputs = model.generate(
                                generated,
                                #bos_token_id=random.randint(1,30000),
                                do_sample=True,
                                top_k=50,
                                max_length = 300,
                                top_p=0.95,
                                num_return_sequences=3
                                )

for i, sample_output in enumerate(sample_outputs):
  print("{}: {}\n\n".format(i, tokenizer.decode(sample_output, skip_special_token
```

```
The attention mask and the pad token id were not set. As a consequence, you ma
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
tensor([[50257]], device='cuda:0')
0: It was a gesture of farewell, a gesture that had belonged to Montenegro fo


1: And I believe he has a large future in front of him, which he hardly knows


2: The porch was full of cheerful sun, bright with the fresh faces of those wh
```

```
model.eval()

prompt = "<|startoftext|>"

generated = torch.tensor(tokenizer.encode(prompt)).unsqueeze(0)
generated = generated.to(device)

print(generated)

sample_outputs = model.generate(
                                generated,
                                #bos_token_id=random.randint(1,30000),
                                do_sample=True,
                                top_k=50,
                                max_length = 300,
                                top_p=0.95,
                                num_return_sequences=3
                                )

for i, sample_output in enumerate(sample_outputs):
  print("{}: {}\n\n".format(i, tokenizer.decode(sample_output, skip_special_token
```

```
The attention mask and the pad token id were not set. As a consequence, you ma
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
tensor([[50257]], device='cuda:0')
0: “She’s going to stay there till the day after tomorrow, and then we’ve run


1: I wanted to see her and hear her voice, but her voice was muffled in the ea


2: “There’s some bad trouble here,” said Gatsby.
```

```
model.eval()

prompt = "<|startoftext|>"

generated = torch.tensor(tokenizer.encode(prompt)).unsqueeze(0)
generated = generated.to(device)

print(generated)

sample_outputs = model.generate(
                                generated,
                                #bos_token_id=random.randint(1,30000),
                                do_sample=True,
                                top_k=50,
                                max_length = 200,
                                top_p=0.95,
                                num_return_sequences=10
                                )

for i, sample_output in enumerate(sample_outputs):
  print("{}: {}\n\n".format(i, tokenizer.decode(sample_output, skip_special_token
```

```
The attention mask and the pad token id were not set. As a consequence, you ma
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
tensor([[50257]], device='cuda:0')
0: “They’ll never stop coming.”  “Never.” Tom’s face pressed against my should


1: Then there was another man in the car.


2: “I suppose he knew what he’d got to do.”  “What did he say to Wilson that a


3: “We oughtn’t to let everything alone,” he muttered, “but we can’t—” He was


4: “They’re—”  “You’re crazy!” he exploded.


5: It was a yellow car.


6: That dog, a bitch, was sitting on my shoulder and chewing on my shoulder, a


7: “I told him.


8: She didn’t answer.


9: I don’t think that Mr. Gatsby bought a garage, for he had no real idea of t
```

```
model.eval()

prompt = "<|startoftext|>"

generated = torch.tensor(tokenizer.encode(prompt)).unsqueeze(0)
generated = generated.to(device)

print(generated)

sample_outputs = model.generate(
                                generated,
                                #bos_token_id=random.randint(1,30000),
                                do_sample=True,
                                top_k=50,
                                max_length = 500,
                                top_p=0.99,
                                num_return_sequences=10
                                )

for i, sample_output in enumerate(sample_outputs):
  print("{}: {}\n\n".format(i, tokenizer.decode(sample_output, skip_special_token
```

```
The attention mask and the pad token id were not set. As a consequence, you ma
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
tensor([[50257]], device='cuda:0')
0: 1.E.6.


1: “Yes.”  “What—how could they possibly love each other?”  “You’ve been with


2: He was so sick I almost fainted and Michaelis was so sick I could hardly fa


3: “What I called up about is—why—how long have you been married, George?


4: For all I knew he was going to rob the house on the strength of his positi


5: “He was just a young man,” she said wanly, “but I’m pretty sure he’d start


6: “Yes … Well, suppose we don’t go to town any more?” asked Jordan.


7: I didn’t mean to interrupt your lunch, but rather, rather, as we all went


8: It was Mr. Buchanan who represented a deplorably small town in a far off f


9: “I know very little about driving—next to nothing.”  “Oh, sure.” He looked
```

Interesting samples colected

1. “There’s some bad trouble here,” said Gatsby.
2. “She’s going to stay there till the day after tomorrow, and then we’ve run out of the house and we’ll go upstairs to have the evening—” Tom’s words have always seemed unnaturally loud across the garden.
3. The porch was full of cheerful sun, bright with the fresh faces of those who came in from his front door.
4. It was a gesture of farewell, a gesture that had belonged to Montenegro for five years and now it was an appropriate tribute for all of us.
5. It must be a madman.” “He’s mad!” “Yes … Yes … Yes … Yes … Yes … there’s nothing … He looked at Tom and then back at me, realizing that he wanted nothing less of Daisy than that she should be less of a thing and a thing.
6. The following sentence, with active links to, or other immediate access to, the full Project Gutenberg™ License must appear prominently whenever any copy of a Project Gutenberg™ work (any work on which the phrase “Project Gutenberg” appears, or with which the phrase “Project Gutenberg” is associated) is accessed, displayed, performed, viewed, copied or distributed: Project Gutenberg™ works.
7. “In the ditch beside the road, if you want to go to take a plunge in the swimming pool,” he suggested.
8. That dog, a bitch, was sitting on my shoulder and chewing on my shoulder, as though it were a puppy.
9. “They’ll never stop coming.” “Never.” Tom’s face pressed against my shoulder, so I told him he must know something about me, and he knew that I lied.
10. For all I knew he was going to rob the house on the strength of his position.

The first 4 sentences and the last 2 sentences are perfectly normal that seem like they are taken from the original text

The 5th sentence has one too many repeating "Yes"

The 6th sentence is interesting because it includes “Project Gutenberg” which was present in the initial part of meta-data in the file. This shows that it needs to be removed.

The 7th and 8th sentences are funny!

.

.

.

.