**Finetuning**

For Finetuning GPT2 x-large with 1.5 Billion parameters we use the finetuning script provided by hugging face at: <https://github.com/huggingface/transformers/blob/master/examples/language-modeling/run_language_modeling.py>

I wrote a script called finetune.py for finetuning which is a wrapper of run\_language\_modeling.py script but it injects the parameters so that we can run it in a much easier way.

Ie: Rather than running

python run\_language\_modeling.py --no\_cuda --output\_dir=output --overwrite\_output\_dir --model\_type=gpt2 --model\_name\_or\_path=gpt2-xl --do\_train --train\_data\_file=./input/training.txt --per\_device\_train\_batch\_size=1 --num\_train\_epochs=3 --gradient\_accumulation\_steps 32

We just run:

python finetune.py

I copied Hugging Face’s script(run\_language\_modeling.py) over to the project folder and modified it so that it the main method can take arguments, thus we can inject arguments when calling it from finetune.py.

The hyperparameters used for finetuning are:

Batch\_size: set to 1 since the amount of memory required for training is a multiple of n, where n is the batch size, and the machine I could afford could only support this setting.

Epochs: 3, every epoch takes so long that want to keep this number low, we’ve observed improvements when training GPT2 with 114M parameters for only 1 epoch.

Fp16: Can reduce memory utilization when training on GPU

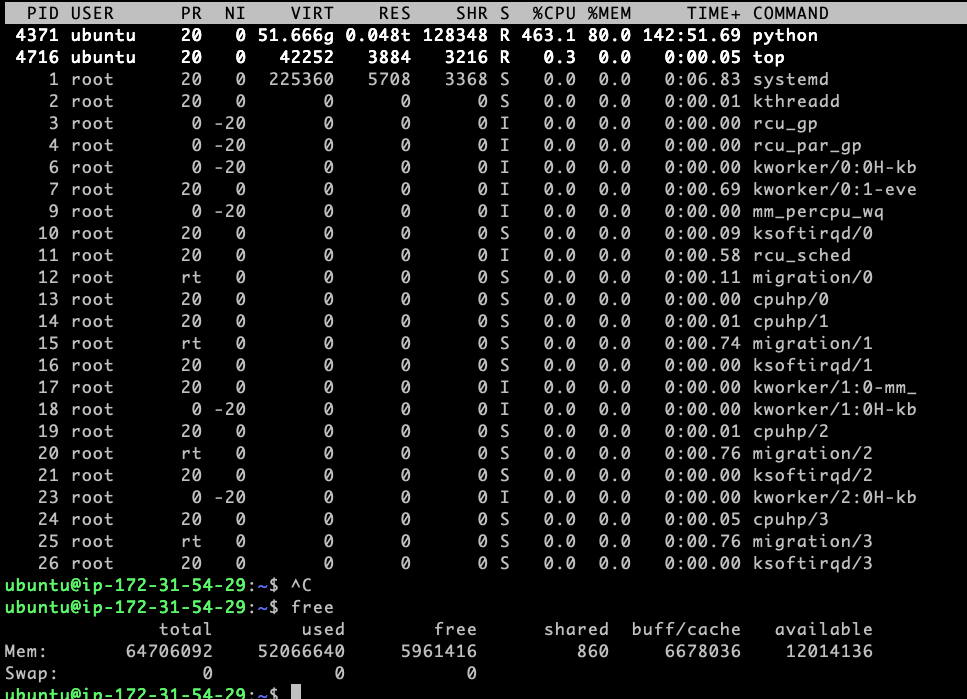
Training Machine

Used AWS EC2 instances

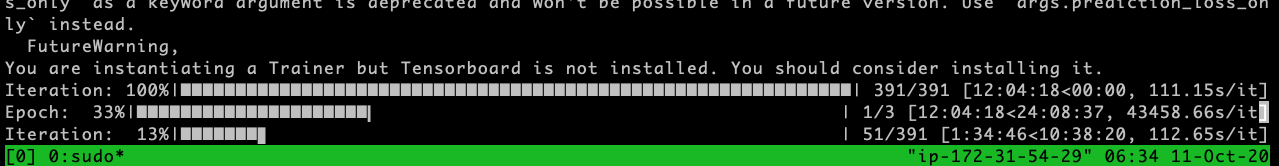
First tried with a p2.xlarge instances which comes with a Tesla V100 GPU, which has 16 GB of memory. but this GPU couldn’t fit even a single example in memory (batch size 1) and this is the most powerful GPU offered by AWS today.

So decided to use a m5ad.4xlarge instance which comes with 64GB of RAM and doesn’t come with GPU so would have to do CPU training, this process is much slower than when training on GPU but was the only choice.

The below screenshot shows the memory available on the machine and the % used by the finetuning process(first in the list: 80%)



Below screenshot shows the time it took 12 hours per epoch when training on CPU:

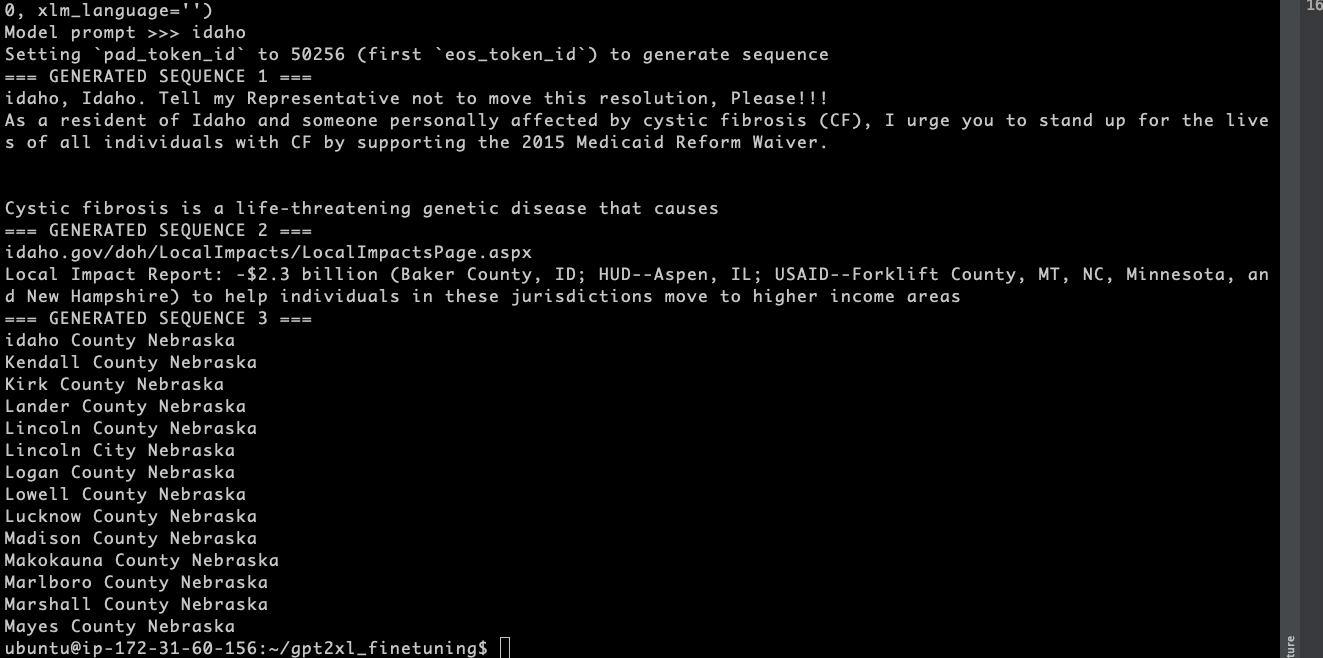


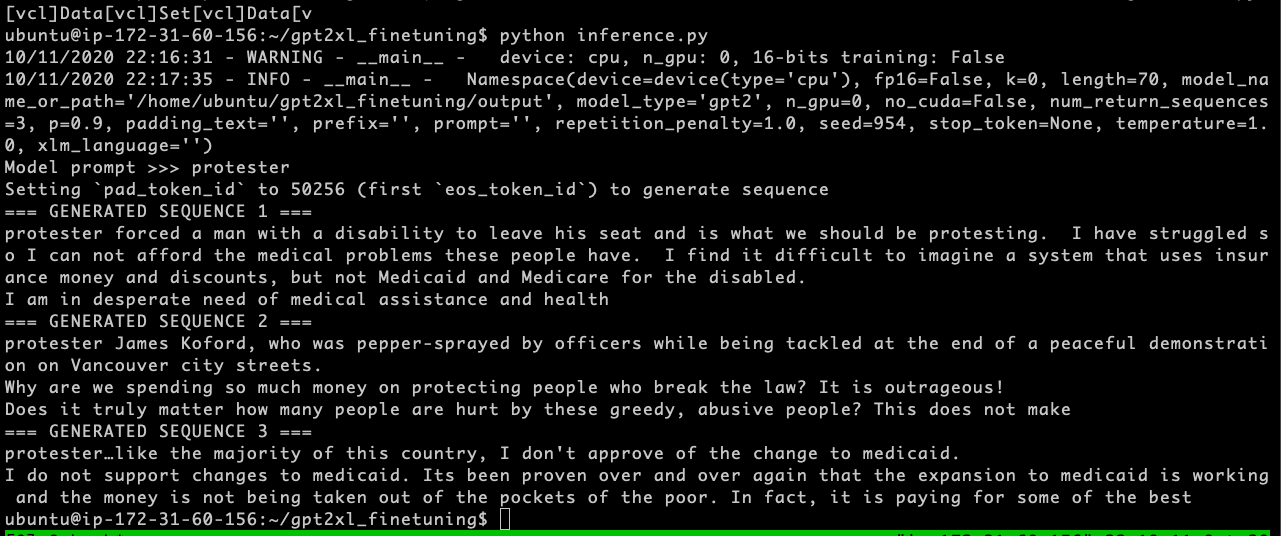
**Generating text**

Once the model was finetuned we could load it using language\_generation.py script(https://github.com/huggingface/transformers/blob/master/examples/text-generation/run\_generation.py) provided by hugging face to obtain predictions given any piece of text.

In the project folder I wrote a wrapper that allowed me to run it more easily: inference.py

Below are some examples generated by the finetuned gpt2-xl model





On inference.py you can change the variable MODEL\_PATH to generate text using different gpt2 models and compare the results:

MODEL\_PATH = FILE\_PATH + "/models/gpt2\_xl\_1ephoc"

Our finetuned version of GPT2 1.5B parameters

MODEL\_PATH = FILE\_PATH + "/models/gpt2\_1ephoc"

Our finetuned version of GPT2 114M parameters

MODEL\_PATH = "gpt2"

Base version of GPT2 144M parameters, downloaded directly from Hugging’s face repository

MODEL\_PATH = "gpt2-xl"

Base version of GPT2 1.5B parameters, downloaded directly from Hugging’s face repository