"Can a Neural Network write poetry? Leveraging next sentence prediction with Transformer GPT-2"
Research Computing Center Workshop — Summer 2020
Jeffrey Tharsen *tharsen@uchicago.edu*

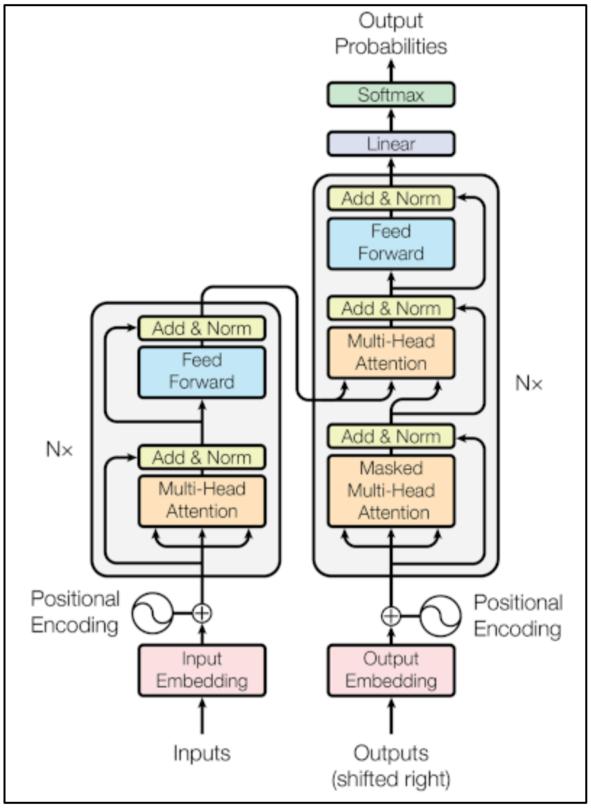
This Guide + source files and sample outputs for Blake and Shakespeare :
https://github.com/rcc-uchicago/BERT+GPT2_tutorial_Summer2020

Materials from the previous workshop on BERT & GPT-2 :
https://github.com/rcc-uchicago/BERT-GPT2-Workshop

1. (For those who haven't seen this yet.)
   "This Grad Student Used a Neural Network to Write His Papers" (GPT-2)
   https://futurism.com/grad-student-neural-network-write-papers

2. **Why GPT-2? What can Trained Transformers achieve? (Encoder-Decoder stacks)**

   https://medium.com/huggingface/encoder-decoders-in-transformers-a-hybrid-pre-trained-architecture-for-seq2seq-af4d7bf14bb8



   https://openai.com/blog/better-language-models/

   Original source for this tutorial:
   https://medium.com/@ngwaifoong92/beginners-guide-to-retrain-gpt-2-117m-to-generate-custom-text-content-8bb5363d8b7f

3. Training GPT-2 on your Source Corpus and Producing NN-generated Text

(With Tensorflow – we'll be using Tf 1.13.1 for this tutorial.)

A. Get the GPT-2 framework – from the command line, do:

```
git clone https://github.com/nshepperd/gpt-2.git
```

(Or go to this link, click on "Code" and download the zip file: https://github.com/nshepperd/gpt-2 *)*

B.

```
cd gpt-2
```

C. Get the model files:

```
python download_model.py 117M
```

D. *(if using Midway, include the --user flag)*

```
pip install fire==0.1.3
pip install regex==2017.4.5
pip install requests==2.21.0
pip install tqdm==4.31.1
pip install toposort
```

AND

```
pip install tensorflow==1.13.1
```

*(*** this tutorial does not work with Tensorflow 1.14+ or 2.x ***)*

E. Put your Source Text File in /gpt-2/src

e.g. "blake_poems_gpt2.txt"
I split the file into sections with "<|endoftext|>".

F. Copy the Python code into /src:

```
cp encode.py src
cp train.py src
cp -r models src
```

G.

```
cd src
```

If using Midway:

```
sinteractive --partition=gpu2 --gres=gpu:1
module load Anaconda3/2018.12
source activate tf-gpu-1.13.1
```

H.

```
python encode.py [source.txt] [output.npz]
```

I.

```
python train.py --dataset [output.npz]
```

It will display samples as it runs; let it run until your loss/avg is under 0.2 .

Train.py will run until you hit Ctrl-C.

J. Set up the new model directory (I use “blake” here as the name):

```
mkdir models/blake

cd checkpoint/run1/* models/blake

cp model-9000* ../../models/blake
cp checkpoint ../../models/blake

cp models/117M/encoder.json models/blake
cp models/117M/hparams.json models/blake
cp models/117M/vocab.bpe models/blake
```

K. Now your model is ready to run:

```
python generate_unconditional_samples.py --model_name blake ( > outfile.txt )
```

If you get a model error, edit the /blake/checkpoint file to match the model checkpoint number you wish to use.

L. Or run (setting **temperature** and **top_k**) :

```
python generate_unconditional_samples.py --temperature 0.8 --top_k 40 --
model_name blake
```

M. Finally, you can run interactively, supplying a prompt:

```
python interactive_conditional_samples.py --temperature 0.8 --top_k 40 --
model_name blake
```