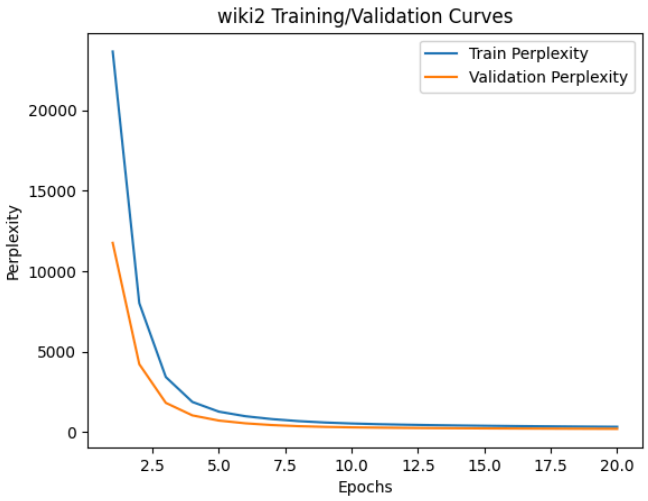
Team 2
Jack Hanke, Daniel Plotkin, Nicole Birova
NLP Homework 1
28 April 2025

Homework 1 Report

1. To create a GPT-2 style model, the team removed all code associated with the encoder and cross attention by moving these functions to the file *encoder_source.py*. The team modified the *main()* function to support cpu training for local debugging, as well as added additional flags to control which dataset the script would run on and where the model would be saved. We also edited the *read_corput()* function on line 25 to return a torch tensor object. The most substantial edits were made to *train_model()* and *test_model()*. Both of these functions benefited from the *get_batch()* function, which translated a token stream into a batched matrix, fetched a segmented *seqlen* context and shifted prediction targets, and returns a generator to be iterated over. The *train_model()* and *test_model()* functions on lines 319 and 383 respectively are common training loops written in machine learning code, with additional lines for creating a progress bar, tracking loss and perplexity over an epoch, printing updates, and casting the model to a lower precision for faster training. Finally, perplexity curves are generated from the recorded metrics calculated on line 493.
2. Below is the training curve for our testing and validation perplexity over the 20 epochs. We used the default parameters across the board. Following this we achieved a test perplexity of 170.7.



3. On Jack's personal machine with a 16GB NVIDIA A4000 graphic's card, with a batch size of 8 we get ~7.5 training iterations per second. At a sequence length of 512, this equates to 30,720 tokens predicted per second. We trained using torch's autocast feature to reduce the weights and gradients to 16 bit floats so that training was quick enough on Jack's machine. More aggressive casting could improve our speed further, as

well as experimenting on the optimal batch size for speed. After 1 epoch, we got 484, 245, 233 on training, validation, and test perplexity for this run.

4. After implementing Euclidean distance as a norm in the attention mechanism, we saw a slowdown to ~6.3 training iterations per second. This represented a 16% slowdown from the dot product attention experiment. We achieved a 485, 249, 236 on training, validation, and test perplexity for this run. This shows similar convergence values at a slower rate, which provides evidence that dot product attention is superior in terms of performance, but not of capability.