NLP 243 Assignment 2

Ryan Okimoto

In this assignment we use a form of language modeling which is the process of forming a probabilistic model that tries to capture patterns in sequences of words to determine how likely a word is to appear and more importantly what word may come next in a sequence. This allows the computer to understand natural language not as a random sequence of symbols but as numbers (tokens) that form distributions that we can apply mathematical methods to. For this assignment the most important application of language modeling is with next token prediction. Next token prediction is a specific objective of a language model such as ChatGPT or any other state of the art large language model where the goal is simply to predict the next token. A token simply represents a combination of characters or words that turn natural language into a computer understandable set of numbers. We then train a model by feeding it large amounts of language and seeing if it's able to predict the next word in a sequence and after the model is trained we can continuously generate text based off of the statistical likelihoods.

Self attention is a mechanism inside of language models that helps to weigh the importance of specific tokens in a sequence no matter how distant they might be. This helps the model find patterns so we know what words we should pay more "attention" to. We calculate this by using 3 learned transformations: queries (Q), keys (K), and values (V). We use these by first taking the dot product of Q and K which is used to measure the relevance, we then normalize these values with the softmax function which creates the attention weights. And after creating a weighted sum with the value weights we have a new representation that is context aware for each token.

My transformer model is made up of transformer blocks which I tested with 2 and 4 layers, where my final submission is with 2 layers that contains a masked self-attention layer as well as a feed forward network wrapped with layer normalization and residual connections. The layer normalization is used to stabilize training and dropout prevents overfitting of the model.

The masked self attention mechanism uses multi-head attention where I used 4 heads where each head learns different aspects of the relationships between the tokens. The causal token forces the model to only look at prior tokens preventing it from cheating and looking ahead and padding tokens are masked in order to not influence attention scores.

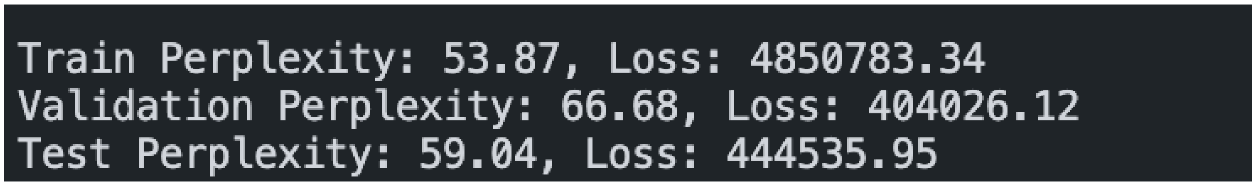
After self attention each token is through a 2 layer MLP with a ReLU activation. So the Feed Forward Network expands the dimensionality from 4 to 16 then back to 4. This allows the model to handle more representations.

Finally the output layer normalizes and stabilizes the representations and projects the d-dimensional hidden states to logits over the entire vocabulary producing a score for each next possible token. During training the model compares the predicted token with the actual next token in the sequence using cross entropy.

Originally training the model on it I used a smaller learning rate of 1e-4 and dimension of 512 and 4 block transformer however this would take multiple hours to run on a cpu so in order to optimize the code to run for <20 minutes. I increased the learning rate to .01 and used a dimension of 128 and only used 2 blocks for the transformer. Finally I implemented a fail safe so that after ~18 minutes of training my model will save the best version of itself and evaluate that version of the model.

NLP 243 Assignment 2

Ryan Okimoto



```
Train Perplexity: 53.87, Loss: 4850783.34
Validation Perplexity: 66.68, Loss: 404026.12
Test Perplexity: 59.04, Loss: 444535.95
```