CS685 Quiz 1: *Transformers*
Released 2/28, due 3/8 on Gradescope (please upload a PDF!)
Please answer both questions in 2-4 sentences each. Make sure to also fill out the AI disclosure!

1. Assume we are building a Transformer sequence-to-sequence model to solve a machine translation task. Why don't we need to use masking when implementing cross attention?

   Answer: Assuming we are building a Transformer sequence-to-sequence model to solve a machine translation task: Converting a French sentence to an English sentence. In such a model, the self-attention mechanism is applied to the encoder and decoder inputs separately, and cross-attention is applied between the encoder and decoder outputs.

   We first apply masking at the self-attention layer in the decoder, as it is responsible for predicting the next English words. (We do not want the attention mechanism to share any information regarding the token at the next positions, when giving a prediction using all the previous tokens. To ensure that this is done, we mask future positions by setting them to -infinity before the softmax step in the self-attention calculation.)

   Next, we apply cross attention, which connects the decoder to the encoder by enabling the decoder to attend over the encoder's final hidden states. Finally, after stacking a bunch of these decoder blocks, we have our familiar Softmax layer to predict the next English word.

   In self-attention, a sequence element can attend to all other elements of the same sequence, including future implementing cross-attention because the decoder can only attend to the encoder outputs up to the current position, and the encoder outputs after the current position are not attended to. Thus, the attention weights for elements. However, in cross-attention, the decoder can attend to all the encoder outputs, but only up to the current position, i.e., the future encoder outputs are not available during decoding. Therefore, masking is required for self-attention in the decoder to prevent attending to future positions. However, we don't need to use masking when future encoder outputs will be naturally set to zero, and there is no need for masking.

   To summarize, we use masking for the self-attention in the decoder to prevent attending to future positions, but we don't need to use masking for cross-attention because the decoder can only attend to the past encoder outputs, and the future encoder outputs are not available during decoding.

   References:
   https://medium.com/analytics-vidhya/masking-in-transformers-self-attention-mechanism-bad3c9ec235c
   https://vaclavkosar.com/ml/cross-attention-in-transformer-architecture
   https://people.cs.umass.edu/~miyyer/cs685_f22/slides/05-transformers.pdf

2. Why can't the hidden state computations of Transformer language models be parallelized at test time?

   Answer: The hidden state computations of Transformer language models cannot be parallelized at test time because a particular hidden state for a token is computed by computing the previously generated token's hidden state and, say, on. During test time, the output is decoded word by word because there is no access to the next tokens. The next token that is generated needs to be concatenated to the input sequence to generate the next token. This computation cannot be performed parallelly. However, during training, the hidden state computations of Transformer language models can be parallelized, unlike RNNs, because it has access to the next token.

AI Disclosure

AI1: Did you use any AI assistance to complete this homework? If so, please also specify what AI you used.
Your answer here
No

(only complete the below questions if you answered yes above)

AI2: If you used a large language model to assist you, please paste *all* of the prompts that you used below. Add a separate bullet for each prompt, and specify which problem is associated with which prompt.

- *Your response here*

AI3: (*Free response)* For each problem for which you used assistance, describe your overall experience with the AI. How helpful was it? Did it just directly give you a good answer, or did you have to edit it? Was its output ever obviously wrong or irrelevant? Did you use it to get the answer or check your own answer?

- *Your response here*