Exercise sheet 2: Understanding next token prediction in decoders

Individual submission notice: Each student is required to complete and submit their exercises **individually**. While discussing general concepts with peers is OK, the final work must be your own.

Exercise 1: The cheating decoder [MG]

In this exercise, you will define the weights of a tiny decoder-only model by hand. The model will then be trained to perform next token prediction based on a single training sentence:

`[BOS] I play tennis [EOS]`

You will define the trainable weights of the decoder so that it correctly predicts every next token in the training sequence, right away. There's a twist, though. Your decoder will be allowed to cheat. Unlike a standard decoder, your decoder will use **unmasked attention**. To predict the next token in the sequence, the model will attend to future tokens. In other words, instead of having to *predict* the next token, the model will *look it up*! The point of the exercise is to show **how unmasked attention can trivialize the next-token prediction task** (i.e., masking is *necessary* during training).

Problem description You will work with a vocabulary consisting of the following tokens:

$$\mathcal{V} = \{\texttt{[BOS]}, \texttt{I}, \texttt{play}, \texttt{tennis}, \texttt{[EOS]}\}$$

The training sentence, as already mentioned, is the following:

`[BOS] I play tennis [EOS]`

Your goal is to define the input embeddings and the weight matrices for a **single-head, single-block** decoder that perfectly predicts each subsequent token in the sequence. The input embeddings for the sentence above will be packed into a matrix $\mathbf{X}$ of size $5 \times d_{model}$:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{\texttt{[BOS]}} \\ \mathbf{x}_{\texttt{I}} \\ \mathbf{x}_{\texttt{play}} \\ \mathbf{x}_{\texttt{tennis}} \\ \mathbf{x}_{\texttt{[EOS]}} \end{bmatrix}$$

The model will use unmasked attention. To predict the next token t_{i+1}, the model will "cheat": token t_i will attend to token t_{i+1} to do so. The attention output $\mathbf{h}_i$ will be used by the language modeling head to compute a logits vector $\mathbf{u}_i$. This vector will then be used to assign probabilities to each token, and the model will correctly select t_{i+1} as the most probable next token.

Defining the decoder architecture Your decoder will consist of the following components:

- **Input embeddings:** The input token sequence is embedded into a matrix $\mathbf{X} \in \mathbb{R}^{5 \times d_{model}}$, where each row corresponds to a token embedding. You must define the values of the input embeddings. You can choose the embedding size, d_{model}, from 1 to 5.
- **Attention mechanism:** Define the weight matrices $\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^V$ of a single attention head to compute the queries, keys, and values. These matrices must have sizes $d_{model} \times d_q$, $d_{model} \times d_k$, and $d_{model} \times d_v$, respectively. You can choose values for d_q, d_k, and d_v between 1 and 5. For this exercise, **assume that the attention mechanism does not use the scaling factor** $\sqrt{d_k}$.
- **Output projection matrix:** The attention output, $\mathbf{Z} \in \mathbb{R}^{5 \times d_v}$, is multiplied with a matrix $\mathbf{W}^O \in \mathbb{R}^{d_v \times d_{model}}$ to map $\mathbf{Z}$ to a matrix of the same size as $\mathbf{X}$, $5 \times d_{model}$. You will define the weights of $\mathbf{W}^O$. For this exercise, **assume that final output from the attention layer, H, is just the residual connection of the input embedding matrix X, with Z**:

$$\mathbf{X}' = \mathbf{Z}\mathbf{W}^O \in \mathbb{R}^{5 \times d_{\text{model}}}$$
$$\mathbf{H} = \mathbf{X} + \mathbf{X}' \in \mathbb{R}^{5 \times d_{\text{model}}}$$

  That is, there is no need to apply layer normalization or pass the embeddings through a feed-forward network.
- **Language modelling head:** Finally, the output of the single attention layer, $\mathbf{H}$, is projected into logits $\mathbf{U} = \mathbf{H}\mathbf{W}^U \in \mathbb{R}^{5 \times 5}$ via a matrix $\mathbf{W}^U \in \mathbb{R}^{d_{model} \times 5}$. Each row i in $\mathbf{U}$, denoted $\mathbf{u}_i$, is passed through a softmax activation; the probabilities $\text{softmax}(\mathbf{u}_i)$ are used to predict the next token t_{i+1}.

Task 1: Constructing the *cheating decoder*

1. Define the input embeddings matrix, $\mathbf{X} \in \mathbb{R}^{5 \times d_{model}}$, for the 5 tokens in the sequence, `[BOS] I play tennis [EOS]`. You can choose the embedding size, d_{model}, from 1 to 5.
2. Define the matrices $\mathbf{W}^Q$, $\mathbf{W}^K$ and $\mathbf{W}^V$. You can choose values d_q, d_k, and d_v from 1 and 5. **Ensure that each token t_i attends exclusively to the token t_{i+1}.**
3. Define the matrix $\mathbf{W}^O$ applied to compute $\mathbf{H}$.
4. Define the language modelling head matrix $\mathbf{W}^U$, which maps $\mathbf{H}$ to a logits matrix $\mathbf{U}$.

After defining the weights:

1. As if we were training the decoder, compute the next-token predictions of the model. To do so, compute the logits vector $\mathbf{u}_i$ for each token, followed by the vector of probabilities, $\text{softmax}(\mathbf{u}_i)$. Choose the token with the highest probability as the prediction. The expected model behavior is as follows:

```
Input:   Predicted next token:
[BOS]    I
I        play
play     tennis
tennis   [EOS]
```

2. Finally, apply an attention mask, so that each token can only attend to previous tokens. Compute the next-token predictions and compare them to the unmasked case. Record the masked predictions in the following format:

```
Input:  Predicted next token (with mask):
[BOS]   ...
I       ...
play    ...
tennis  ...
```

Submission format: Submit the following files:

1. A text file named `cheating_decoder.txt` with the input embedding matrix and the weight matrices formatted exactly as follows:

```
X = [[x11, x12, ...], [x21, x22, ...], ...]
W_Q = [[q11, q12, ...], [q21, q22, ...], ...]
W_K = [[k11, k12, ...], [k21, k22, ...], ...]
W_V = [[v11, v12, ...], [v21, v22, ...], ...]
W_O = [[w11, w12, ...], [w21, w22, ...], ...]
W_U = [[w11, w12, ...], [w21, w22, ...], ...]
```

2. A text file named `decoder_explanation.txt`. In this file, provide a concise explanation of the reasoning behind your choice of weight matrices. Justify your choices by explaining how each matrix contributes to the model's behavior, and include the key mathematical calculations that demonstrate their effectiveness.

3. A text file named `decoder_predictions.txt` with the next token predictions in the format:

```
Input:  Predicted next token:
[BOS]   ...
I       ...
play    ...
tennis  ...

Input:  Predicted next token (with mask):
[BOS]   ...
I       ...
play    ...
tennis  ...
```

Exercise 2: The importance of positional information [MG]

You're chatting with a friend. One day she says: "The cat sleeps." You nod and reply, "ok." The next day she says: "Cat the sleeps." You turn to her, a bit worried —is she drunk?– and respond, "What?"

Your response changes depending on *word order*. A language model's response should, too.

To generate different responses to "`the cat sleeps`" and "`cat the sleeps`", a transformer language model needs to use *positional encodings*. Let's convince ourselves of this fact with a decoder-only model.

Setup. We use a single-head, single-block decoder with

$$d_{\text{model}} = d_q = d_k = d_v = 2,$$

no scaling, no layer norm, no feed-forward. The input embedding sequence is packed into a matrix $\mathbf{X}$. The contextualized embeddings are

$$\mathbf{H} = \mathbf{X} + \mathbf{Z}\mathbf{W}^O.$$

For a three-token sentence, denote the rows of $\mathbf{H}$ as $\mathbf{h}_1$, $\mathbf{h}_2$ and $\mathbf{h}_3$. We use the representation at position 3 to predict the next token (position 4) via the LM head:

$$\mathbf{u}_3 = \mathbf{h}_3\mathbf{W}^U, \qquad \text{probs} = \text{softmax}(\mathbf{u}_3) \in \mathbb{R}^{|\mathcal{V}|}.$$

The vocabulary is

$$\mathcal{V} = \{\texttt{cat}, \texttt{sleeps}, \texttt{the}, \texttt{ok}, \texttt{what?}\},$$

with logits and probabilities always reported in this order. We will test the following two sentences:

$$\text{sentence 1: } \texttt{the cat sleeps} \qquad \text{sentence 2: } \texttt{cat the sleeps}.$$

Input embeddings. $\mathbf{x}_{\texttt{the}} = \begin{bmatrix} 2 & 0 \end{bmatrix}$, $\mathbf{x}_{\texttt{cat}} = \begin{bmatrix} -2 & 0 \end{bmatrix}$, $\mathbf{x}_{\texttt{sleeps}} = \begin{bmatrix} 0 & 0 \end{bmatrix}$.

Positional encodings. Each position $pos \in \{1, 2, 3\}$ has encoding

$$\text{PE}(pos, i) = \begin{cases} \sin\left(\dfrac{pos}{10000^{i/d_{\text{model}}}}\right), & \text{if } i \text{ is even,} \\ \cos\left(\dfrac{pos}{10000^{(i-1)/d_{\text{model}}}}\right), & \text{if } i \text{ is odd,} \end{cases}$$

for dimension index $i = 0, 1$.

Attention mechanism.

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}^K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}^V, \quad \mathbf{S} = \mathbf{Q}\mathbf{K}^\top, \quad \mathbf{A} = \text{softmax}_{\text{row}}(\text{mask}(\mathbf{S})), \quad \mathbf{Z} = \mathbf{A}\mathbf{V}.$$

Weights. We'll use the following weights,

$$\mathbf{W}^Q = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{W}^K = \begin{bmatrix} 0 & 0 \\ 0 & -5 \end{bmatrix}, \quad \mathbf{W}^V = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{W}^O = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

The LM head matrix $\mathbf{W}^U \in \mathbb{R}^{2\times 5}$ is:

$$\mathbf{W}^U = \begin{bmatrix} 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Task 1: Next-token predictions *without* positional encodings.

To convince yourself that the LM **without positional encodings** cannot produce different next-token predictions for the two sentences, compute the forward pass for both sentences and report the probability distributions $\texttt{probs1} \in \mathbb{R}^{1\times|\mathcal{V}|}, \texttt{probs2} \in \mathbb{R}^{1\times|\mathcal{V}|}$ for sentence 1 and 2, respectively. Do they differ?

Task 2: Next-token predictions *with* positional encodings.

Now use the definition of $\text{PE}(pos, i)$ above to compute **position-aware embeddings** `PE`. Compute the forward pass again for both sentences, using such embeddings. Report the next-token probability distributions:

$$\texttt{probs1PE},\ \texttt{probs2PE} \in \mathbb{R}^{1\times|\mathcal{V}|}.$$

Identify the most probable next token after "the cat sleeps" and after "cat the sleeps".

Submission format: Submit the following files:

1. A text file named `tiny_decoder_positional.txt` with probability vectors and positional encodings formatted exactly as follows:

```
# Task 1: no positional encodings
probs1noPE = [p_cat, p_sleeps, p_the, p_ok, p_what?] # replace with numbers
probs2noPE = [p_cat, p_sleeps, p_the, p_ok, p_what?] # replace with numbers

# Task 2: with positional encodings
PE = [[pe11, pe12], [pe21, pe22], [pe31, pe32]] # replace with numbers
probs1PE = [p_cat, p_sleeps, p_the, p_ok, p_what?] # replace with numbers
probs2PE = [p_cat, p_sleeps, p_the, p_ok, p_what?] # replace with numbers
```