Deep Learning (IST, 2023-24)

Practical 09: Attention Mechanisms

André Martins, Ben Peters, Chryssa Zerva, Duarte Alves

Pen-and-Paper Exercises

The following questions should be solved by hand. You can use, of course, tools for auxiliary numerical computations.

Question 1

Let

$$\boldsymbol{x}^{(1)} = [-2.0, 1.0, 0.5]^\top, \quad \boldsymbol{x}^{(2)} = [1.0, 1.5, -0.5]^\top, \quad \boldsymbol{x}^{(3)} = [-1.5, 1.0, -0.5]^\top, \quad \boldsymbol{x}^{(4)} = [-2.0, -2.5, 1.5]^\top$$

be an sequence of length 4, where each element is a vector in $\mathbb{R}^3$. We let $\boldsymbol{X} = [\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}]^\top \in \mathbb{R}^{4\times 3}$ be the resulting input matrix.

1. Let $\boldsymbol{q} = [-2.0, 1.0, -1.0]^\top$ be a query vector. Compute the attention probabilities and resulting output vector induced by this query on $\boldsymbol{X}$ using scaled dot product attention.

   Solution: Let's form the input matrix whose rows contain each element of the sequence,

   $$\boldsymbol{X} = \begin{bmatrix} -2.0 & 1.0 & 0.5 \\ 1.0 & 1.5 & -0.5 \\ -1.5 & 1.0 & -0.5 \\ -2.0 & -2.5 & 1.5 \end{bmatrix}.$$

   With scaled dot product attention, the scores are given by:

   $$\boldsymbol{z} = \frac{\boldsymbol{X}\boldsymbol{q}}{\sqrt{3}} = \frac{1}{\sqrt{3}} \begin{bmatrix} -2.0 & 1.0 & 0.5 \\ 1.0 & 1.5 & -0.5 \\ -1.5 & 1.0 & -0.5 \\ -2.0 & -2.5 & 1.5 \end{bmatrix} \cdot \begin{bmatrix} -2.0 \\ 1.0 \\ -1.0 \end{bmatrix} = \begin{bmatrix} 2.59807621 \\ 0 \\ 2.59807621 \\ 0 \end{bmatrix}.$$

   The attention probabilities are

   $$\boldsymbol{a} = \text{softmax}(\boldsymbol{z}) = [0.46536883, 0.03463117, 0.46536883, 0.03463117]^\top.$$

   The output vector is

   $$\boldsymbol{c} = \boldsymbol{X}^\top \boldsymbol{a} = \begin{bmatrix} -2.0 & 1.0 & 0.5 \\ 1.0 & 1.5 & -0.5 \\ -1.5 & 1.0 & -0.5 \\ -2.0 & -2.5 & 1.5 \end{bmatrix}^\top \cdot \begin{bmatrix} 0.46536883 \\ 0.03463117 \\ 0.46536883 \\ 0.03463117 \end{bmatrix} = \begin{bmatrix} -1.66342208 \\ 0.8961065 \\ 0.03463117 \end{bmatrix}.$$

2. Let us suppose now that we want to compute (single-head) self-attention on this input. Assuming that the projection matrices for queries, keys, and values are respectively

$$\boldsymbol{W}_Q = \begin{bmatrix} 1 & -1.5 \\ 0 & 2 \\ -0.5 & -1 \end{bmatrix}, \quad \boldsymbol{W}_K = \begin{bmatrix} -1.5 & -1 \\ 2.5 & 0 \\ 0.5 & -1 \end{bmatrix}, \quad \boldsymbol{W}_V = \begin{bmatrix} 1 & 2.5 \\ -0.5 & -2 \\ 0 & -1 \end{bmatrix},$$

compute the query, key, and value matrices ($\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$), and the resulting output vector $\boldsymbol{Z}$. Plot the attention map.

Solution: We have

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q = \begin{bmatrix} -2.0 & 1.0 & 0.5 \\ 1.0 & 1.5 & -0.5 \\ -1.5 & 1.0 & -0.5 \\ -2.0 & -2.5 & 1.5 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1.5 \\ 0 & 2 \\ -0.5 & -1 \end{bmatrix} = \begin{bmatrix} -2.25 & 4.5 \\ 1.25 & 2 \\ -1.25 & 4.75 \\ -2.75 & -3.5 \end{bmatrix},$$

$$\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K = \begin{bmatrix} -2.0 & 1.0 & 0.5 \\ 1.0 & 1.5 & -0.5 \\ -1.5 & 1.0 & -0.5 \\ -2.0 & -2.5 & 1.5 \end{bmatrix} \cdot \begin{bmatrix} -1.5 & -1 \\ 2.5 & 0 \\ 0.5 & -1 \end{bmatrix} = \begin{bmatrix} 5.75 & 1.5 \\ 2 & -0.5 \\ 4.5 & 2 \\ -2.5 & 0.5 \end{bmatrix},$$

$$\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V = \begin{bmatrix} -2.0 & 1.0 & 0.5 \\ 1.0 & 1.5 & -0.5 \\ -1.5 & 1.0 & -0.5 \\ -2.0 & -2.5 & 1.5 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2.5 \\ -0.5 & -2 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} -2.5 & -7.5 \\ 0.25 & 0 \\ -2 & -5.25 \\ -0.75 & -1.5 \end{bmatrix}.$$

The attention probabilities $\boldsymbol{P}$ are given by computing scaled dot product attention and applying softmax row-wise:

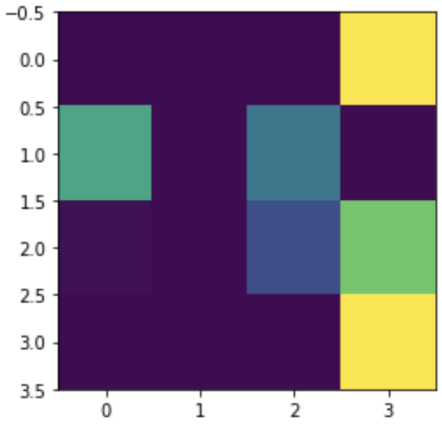
$$\boldsymbol{P} = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{2}}\right)$$

$$= \begin{bmatrix} 4.79433566 \times 10^{-5} & 3.22098620 \times 10^{-5} & 1.71943035 \times 10^{-3} & 9.98200416 \times 10^{-1} \\ 5.97319598 \times 10^{-1} & 1.28333499 \times 10^{-3} & 4.01298182 \times 10^{-1} & 9.88847812 \times 10^{-5} \\ 1.46423661 \times 10^{-2} & 4.87223528 \times 10^{-4} & 2.37021787 \times 10^{-1} & 7.47848624 \times 10^{-1} \\ 9.06143069 \times 10^{-9} & 1.87817885 \times 10^{-3} & 2.98824011 \times 10^{-8} & 9.98121782 \times 10^{-1} \end{bmatrix}.$$

Finally, the output is given by

$$\boldsymbol{Z} = \boldsymbol{P}\boldsymbol{V} = \begin{bmatrix} -0.75220098 & -1.50668721 \\ -2.29564869 & -6.58686077 \\ -1.07141415 & -2.47595506 \\ -0.74812187 & -1.4971829 \end{bmatrix}.$$

The attention probabilities can be represented as the following plot:



3. Let us now assume that we have a second attention head whose parameters $\boldsymbol{W}_Q^{(2)}$, $\boldsymbol{W}_K^{(2)}$, $\boldsymbol{W}_V^{(2)}$ are matrices with all-ones (keeping the first attention head). Using

$$\boldsymbol{W}_O = \begin{bmatrix} -1 & 1.5 & 2 \\ 0 & -1 & -2 \\ 1 & -1.5 & 0 \\ 2 & 0 & 1 \end{bmatrix},$$

compute the resulting output vector $\boldsymbol{Z}$.

Solution: Proceeding as above, we get

$$\boldsymbol{P}^{(2)} = \text{softmax}\left(\frac{\boldsymbol{Q}^{(2)}\boldsymbol{K}^{(2)^\top}}{\sqrt{2}}\right)$$
$$= \begin{bmatrix} 1.18306921 \times 10^{-1} & 2.01966211 \times 10^{-2} & 1.68483136 \times 10^{-1} & 6.93013322 \times 10^{-1} \\ 8.48429312 \times 10^{-4} & 9.98944583 \times 10^{-1} & 2.06267364 \times 10^{-4} & 7.20592823 \times 10^{-7} \\ 2.67590116 \times 10^{-2} & 7.79843042 \times 10^{-4} & 5.42703523 \times 10^{-2} & 9.18190793 \times 10^{-1} \\ 2.47463407 \times 10^{-5} & 6.12522986 \times 10^{-10} & 2.06437556 \times 10^{-4} & 9.99768815 \times 10^{-1} \end{bmatrix}$$

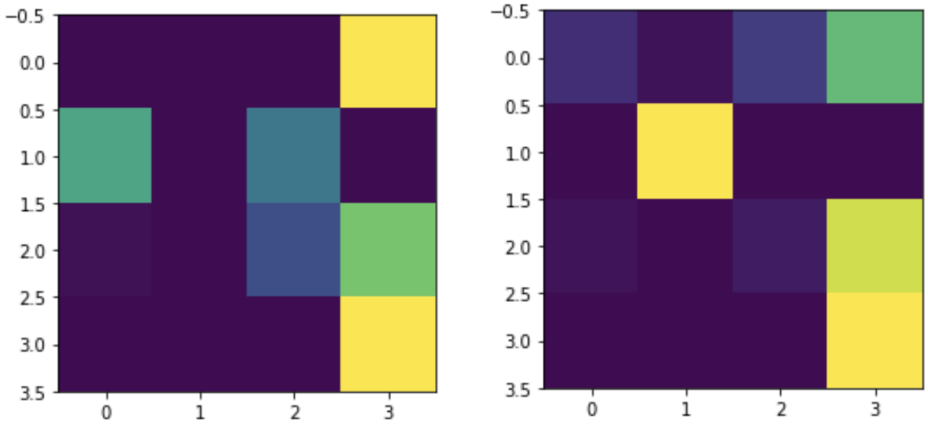
and

$$\boldsymbol{Z}^{(2)} = \boldsymbol{P}^{(2)}\boldsymbol{V}^{(2)} = \begin{bmatrix} -2.26628332 & -2.26628332 \\ 1.99725652 & 1.99725652 \\ -2.82066255 & -2.82066255 \\ -2.99952526 & -2.99952526 \end{bmatrix}.$$

The total output $\boldsymbol{Z}^{\text{total}}$ concatenates the head representations $\boldsymbol{Z}^{(1)}$ (the $\boldsymbol{Z}$ from the previous exercise) and $\boldsymbol{Z}^{(2)}$ (from this exercise) and matrix-multiplies by the output matrix $\boldsymbol{W}_O$, leading to

$$\boldsymbol{Z}^{\text{total}} = [\boldsymbol{Z}^{(1)}, \boldsymbol{Z}^{(2)}] \cdot \boldsymbol{W}_O = \begin{bmatrix} -6.04664898 & 3.77781072 & -0.75731086 \\ 8.28741825 & 0.14750295 & 10.57968068 \\ -7.3905735 & 5.09982766 & -0.01158073 \\ -8.25045389 & 4.87428797 & -1.50140321 \end{bmatrix}.$$

The attention probabilities given by the two heads are represented as the following plots:



Question 2

In this exercise, you will rewrite the attention mechanism **without** using matrix-matrix multiplications.

1. First, write the output of the dot product attention mechanism $\boldsymbol{z}_i$ as a function of the query vector $\boldsymbol{q}_i \in \mathbb{R}^{d_Q}$, the keys $\boldsymbol{k}_j \in \mathbb{R}^{d_K}$ (for $j \in \{1, \ldots, L\}$) and the values $\boldsymbol{v}_k \in \mathbb{R}^{d_V}$ (for $k \in \{1, \ldots, L\}$). In this exercise, consider only a single head and disregard the projection matrices.

   Solution: We start by computing a given entry in the unnormalized attention matrix.

   $$s_{ij} = \frac{\boldsymbol{q}_i^\top \boldsymbol{k}_j}{\sqrt{d_K}}$$

   Then, we normalize the attention matrix with softmax.

   $$p_{ij} = \frac{\exp(s_{ij})}{\sum_{l=1}^{L} \exp(s_{il})}$$

   Finally, we compute the output vector as a weighted sum of the value vectors.

   $$\boldsymbol{r}_i = \sum_{k=1}^{L} p_{ik} \boldsymbol{v}_k$$

2. Now, write the full multi-head attention mechanism output $\boldsymbol{z}_i$ as a function of the input vectors $\boldsymbol{x}_i \in \mathbb{R}^D$ and $\boldsymbol{x}_j \in \mathbb{R}^D$ (for $i, j \in \{1, \ldots, L\}$). Consider H heads and the projection matrices $\boldsymbol{W}_Q^{(h)} \in \mathbb{R}^{D \times d_Q}$, $\boldsymbol{W}_K^{(h)} \in \mathbb{R}^{D \times d_K}$, $\boldsymbol{W}_V^{(h)} \in \mathbb{R}^{D \times d_V}$ and $\boldsymbol{W}_O^{(h)} \in \mathbb{R}^{H d_V \times d_O}$.

Solution: We start by computing the attention matrix for each head.

$$s_{ij}^{(h)} = \frac{(\boldsymbol{W}_Q^{(h)} \boldsymbol{x}_i)^\top (\boldsymbol{W}_K^{(h)} \boldsymbol{x}_j)}{\sqrt{d_K}}$$

Then, we normalize the attention matrix with softmax.

$$p_{ij}^{(h)} = \frac{\exp(s_{ij}^{(h)})}{\sum_{l=1}^{L} \exp(s_{il}^{(h)})}$$

Finally, we compute the output vector for a given head as a weighted sum of the value vectors.

$$\boldsymbol{r}_i^{(h)} = \sum_{j=1}^{L} p_{ij}^{(h)} \boldsymbol{W}_V^{(h)} \boldsymbol{x}_j$$

The total output is obtained by concatenating the outputs of each head. (Note: The output vectors computed above are column vectors of dimension d_V, so we need to transpose them before concatenating. The final output vector is a row vector of dimension Hd_V.)

$$\boldsymbol{r}_i = [\boldsymbol{r}_i^{(1)^\top}, \ldots, \boldsymbol{r}_i^{(H)^\top}] \boldsymbol{W}_O$$

Programming Exercises

The following exercises should be solved using Python, you can use the corresponding practical's notebook for guidance.

Question 3

In this exercise, you will implement an attention mechanism for the simple sequence-to-sequence task of string reversal. Given a source string over some alphabet (in our case, just the first four letters of the Roman alphabet), the model's task is to return the string in reversed order. For this task, we will use randomly generated training and validation data. A simple LSTM-based sequence-to-sequence model has already been implemented for you in the attached notebook `attention.ipynb`.

1. Train a unidirectional sequence-to-sequence model on the string reversal task for 30 epochs. Reasonable hyperparameters are already included in the notebook. Observe the results.

   Solution: The final accuracy in the validation set was 82.1%.

2. In this exercise, you will implement a simple but effective style of attention mechanism called *dot-product attention* (the same as in Question 1 but without the scale factor $\sqrt{d}$).

Dot-product attention works as an extra layer inside an RNN decoder that allows it to make more focused use of the hidden states computed by the encoder. The mechanism receives two inputs at time step t: the *query* $\boldsymbol{s}_t$ is the output of the decoder RNN; the *context* $\boldsymbol{F} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_S]$ is the sequence of all the hidden states computed by the encoder RNN. The attention mechanism first computes an unnormalized attention score between target position t and source position s with a simple dot product, and then normalizes with softmax:

$$z_{ts} = \boldsymbol{s}_t^\top \boldsymbol{h}_s$$
$$\boldsymbol{a}_t = \text{softmax}(\boldsymbol{z}_t).$$

Then, $\boldsymbol{a}_t$ is used to compute a weighted sum $\boldsymbol{c}_t = \boldsymbol{F}\boldsymbol{a}_t$ over the source hidden states.

Finally, $\boldsymbol{c}_t$ is concatenated to $\boldsymbol{s}_t$ and they are passed through a feed-forward layer, returning the *attentional hidden state* $\tilde{\boldsymbol{s}}_t = \tanh(W_c[\boldsymbol{c}_t; \boldsymbol{s}_t])$.

Your goal is to implement dot-product attention in the `DotProdAttention` pytorch module in `attention.ipynb`. After you implement it, train the model with the same hyperparameters as in the previous exercise. How does performance compare?

Solution: The final accuracy in the validation set was 99.9%, so the model with attention learns essentially solve this task.

3. Visualize the attention distributions returned by your model. Do they look the way you expected they would?

Solution:

An example of an attention plot for the sequence `abacadabacc` is given below. The diagonal structure is expected, since the decoder needs to attend to the source tokens in reverse order.