

## Chapter 12 - Transformers

abc1306

Textbook: Deep learning - Foundations and Concepts, by Christopher and Hugué Bishop (2024), Chapter 12 - Transformers.

Textbook website: <https://www.bishopbook.com>

### Exercise 12.1

**Consider a set of coefficients  $a_{nm}$ , for  $m = 1, \dots, N$  with the properties  $a_{nm} \geq 0$  and  $\sum_m a_{nm} = 1$ . By using a Lagrange multiplier show that the coefficients must also satisfy  $a_{nm} \leq 1$  for  $n = 1, \dots, N$ .**

**Solution.** This statement is true by definition, not requiring Lagrange multipliers. The constraints given are standard properties of probabilities or weights in a convex combination. If  $a_{nm} \geq 0$  for all  $m$ , and  $\sum_{m=1}^N a_{nm} = 1$ , then consider a specific coefficient  $a_{nk}$ . Since all terms in the sum are non-negative, we must have:

$$a_{nk} \leq \sum_{m=1}^N a_{nm}$$

Substituting the second constraint:

$$a_{nk} \leq 1$$

This holds for all  $n$  and  $k$ . The Lagrange multiplier method is typically used for optimization problems, not for proving consequences of given constraints like this.

---

### Exercise 12.2

**Verify that the softmax function (12.5) satisfies the constraints (12.3) and (12.4) for any values of the vectors  $\mathbf{x}_1, \dots, \mathbf{x}_N$ .**

**Solution.** The softmax function for attention weights is given by (12.5):

$$a_{nm} = \frac{\exp(\mathbf{x}_n^T \mathbf{x}_m)}{\sum_{m'=1}^N \exp(\mathbf{x}_n^T \mathbf{x}_{m'})}$$

Constraint (12.3) requires  $a_{nm} \geq 0$ . The term  $\mathbf{x}_n^T \mathbf{x}_m$  is a real number. The exponential function  $\exp(z)$  is always strictly positive for any real  $z$ . Therefore, the numerator  $\exp(\mathbf{x}_n^T \mathbf{x}_m)$  is positive. The denominator is a sum of strictly positive terms, so it is also strictly positive. The ratio of two positive numbers is positive. Thus,  $a_{nm} > 0$ , which satisfies  $a_{nm} \geq 0$ .

Constraint (12.4) requires  $\sum_{m=1}^N a_{nm} = 1$ . Let's sum  $a_{nm}$  over  $m$ :

$$\sum_{m=1}^N a_{nm} = \sum_{m=1}^N \left( \frac{\exp(\mathbf{x}_n^T \mathbf{x}_m)}{\sum_{m'=1}^N \exp(\mathbf{x}_n^T \mathbf{x}_{m'})} \right)$$

The denominator does not depend on the summation index  $m$ , so it can be factored out:

$$= \frac{1}{\sum_{m'=1}^N \exp(\mathbf{x}_n^T \mathbf{x}_{m'})} \sum_{m=1}^N \exp(\mathbf{x}_n^T \mathbf{x}_m)$$

The sum in the numerator is identical to the sum in the denominator.

$$= \frac{\sum_{m=1}^N \exp(\mathbf{x}_n^T \mathbf{x}_m)}{\sum_{m'=1}^N \exp(\mathbf{x}_n^T \mathbf{x}_{m'})} = 1$$

Thus, constraint (12.4) is satisfied. Both constraints hold for any real input vectors  $\mathbf{x}_n$ .

### Exercise 12.3

Consider the input vectors  $\mathbf{x}_n$  in the simple transformation defined by (12.2), in which the weighting coefficients  $a_{nm}$  are defined by (12.5). Show that if all the input vectors are orthogonal, so that  $\mathbf{x}_n^T \mathbf{x}_m = 0$  for  $n \neq m$ , then the output vectors will simply be equal to the input vectors so that  $\mathbf{y}_n = \mathbf{x}_n$  for  $n = 1, \dots, N$ .

**Solution.** The output vector  $\mathbf{y}_n$  is given by (12.2):

$$\mathbf{y}_n = \sum_{m=1}^N a_{nm} \mathbf{x}_m$$

The attention weights  $a_{nm}$  are given by (12.5):

$$a_{nm} = \frac{\exp(\mathbf{x}_n^T \mathbf{x}_m)}{\sum_{m'=1}^N \exp(\mathbf{x}_n^T \mathbf{x}_{m'})}$$

We are given that the input vectors are orthogonal:  $\mathbf{x}_n^T \mathbf{x}_m = 0$  for  $n \neq m$ . Let  $C_n = \mathbf{x}_n^T \mathbf{x}_n = \|\mathbf{x}_n\|^2$ . Assume  $\mathbf{x}_n \neq 0$ , so  $C_n > 0$ .

Consider the denominator of  $a_{nm}$  for a fixed  $n$ :

$$\sum_{m'=1}^N \exp(\mathbf{x}_n^T \mathbf{x}_{m'})$$

This sum includes terms where  $m' = n$  and terms where  $m' \neq n$ .

$$= \exp(\mathbf{x}_n^T \mathbf{x}_n) + \sum_{m' \neq n} \exp(\mathbf{x}_n^T \mathbf{x}_{m'})$$

Using the orthogonality condition:

$$= \exp(C_n) + \sum_{m' \neq n} \exp(0) = e^{C_n} + \sum_{m' \neq n} 1 = e^{C_n} + (N - 1)$$

Now consider the numerator  $\exp(\mathbf{x}_n^T \mathbf{x}_m)$ : If  $m = n$ , the numerator is  $\exp(\mathbf{x}_n^T \mathbf{x}_n) = e^{C_n}$ . If  $m \neq n$ , the numerator is  $\exp(\mathbf{x}_n^T \mathbf{x}_m) = \exp(0) = 1$ .

Now find the attention weights  $a_{nm}$ : Case 1:  $m = n$

$$a_{nn} = \frac{e^{C_n}}{e^{C_n} + (N - 1)}$$

Case 2:  $m \neq n$

$$a_{nm} = \frac{1}{e^{C_n} + (N - 1)}$$

Substitute these weights into the expression for  $\mathbf{y}_n$ :

$$\begin{aligned} \mathbf{y}_n &= a_{nn} \mathbf{x}_n + \sum_{m \neq n} a_{nm} \mathbf{x}_m \\ \mathbf{y}_n &= \left( \frac{e^{C_n}}{e^{C_n} + (N - 1)} \right) \mathbf{x}_n + \sum_{m \neq n} \left( \frac{1}{e^{C_n} + (N - 1)} \right) \mathbf{x}_m \end{aligned}$$

This doesn't simplify to  $\mathbf{y}_n = \mathbf{x}_n$ . Let's re-read the question and definition.

Ah, the standard definition (12.14) includes scaling by  $1/\sqrt{D_k}$ . Let's use that, although it's not strictly part of (12.5).

$$a_{nm} = \frac{\exp(\mathbf{x}_n^T \mathbf{x}_m / \sqrt{D_k})}{\sum_{m'} \exp(\mathbf{x}_n^T \mathbf{x}_{m'} / \sqrt{D_k})}$$

With orthogonality:  $\mathbf{x}_n^T \mathbf{x}_m = 0$  for  $n \neq m$  and  $\mathbf{x}_n^T \mathbf{x}_n = C_n$ . Denominator:

$$\sum_{m'} \exp(\dots) = \exp(C_n / \sqrt{D_k}) + \sum_{m' \neq n} \exp(0 / \sqrt{D_k}) = \exp(C_n / \sqrt{D_k}) + (N - 1)$$

Numerator: If  $m = n$ :  $\exp(C_n / \sqrt{D_k})$  If  $m \neq n$ :  $\exp(0 / \sqrt{D_k}) = 1$  Weights:

$$\begin{aligned} a_{nn} &= \frac{\exp(C_n / \sqrt{D_k})}{\exp(C_n / \sqrt{D_k}) + (N - 1)} \\ a_{nm} &= \frac{1}{\exp(C_n / \sqrt{D_k}) + (N - 1)} \quad (m \neq n) \end{aligned}$$

This still doesn't simplify nicely.

Let's reconsider the definition without scaling (12.5) and the question setup. Perhaps there's an implicit assumption or a different interpretation. What if orthogonality implies  $\mathbf{x}_n^T \mathbf{x}_n = C$  (constant magnitude) and  $\mathbf{x}_n^T \mathbf{x}_m = 0$ ? The derivation is the same.

What if the question implies a different similarity measure or a scenario where  $e^{C_n}$  dominates? If  $C_n \rightarrow \infty$ , then  $a_{nn} \rightarrow 1$  and  $a_{nm} \rightarrow 0$  for  $m \neq n$ . In this limit,  $\mathbf{y}_n \rightarrow \mathbf{x}_n$ . Maybe the question assumes this high magnitude scenario implicitly?

Let's check the core softmax property. The definition  $a_{nm} = \text{softmax}_m(\mathbf{x}_n^T \mathbf{x}_m)$  means  $a_{nm}$  depends on the dot products involving  $\mathbf{x}_n$ . If  $\mathbf{x}_n^T \mathbf{x}_n$  is significantly larger than  $\mathbf{x}_n^T \mathbf{x}_m$  for  $m \neq n$ , then  $a_{nn}$  will approach 1. Orthogonality makes  $\mathbf{x}_n^T \mathbf{x}_m = 0$  for  $m \neq n$ . The comparison is between  $\exp(\mathbf{x}_n^T \mathbf{x}_n)$  and  $\exp(0) = 1$ . If  $\|\mathbf{x}_n\|^2$  is large, then  $e^{\|\mathbf{x}_n\|^2}$  will be much larger than 1.

$$a_{nn} = \frac{e^{C_n}}{e^{C_n} + (N - 1)} \approx 1 \quad (\text{if } e^{C_n} \gg N - 1)$$

$$a_{nm} = \frac{1}{e^{C_n} + (N-1)} \approx 0 \quad (\text{if } e^{C_n} \gg N-1)$$

Under this assumption,

$$\mathbf{y}_n = a_{nn}\mathbf{x}_n + \sum_{m \neq n} a_{nm}\mathbf{x}_m \approx (1)\mathbf{x}_n + \sum_{m \neq n} (0)\mathbf{x}_m = \mathbf{x}_n$$

So, the result holds under the assumption that the squared norm of the vectors is large enough for its exponential to dominate  $N-1$ .

## Exercise 12.4

**Consider two independent random vectors  $\mathbf{a}$  and  $\mathbf{b}$  each of dimension  $D$  and each being drawn from a Gaussian distribution with zero mean and unit variance  $\mathcal{N}(\cdot|0, I)$ . Show that the expected value of  $(\mathbf{a}^T \mathbf{b})^2$  is given by  $D$ .**

**Solution.** Let  $\mathbf{a} = [a_1, \dots, a_D]^T$  and  $\mathbf{b} = [b_1, \dots, b_D]^T$ . The components  $a_i$  and  $b_j$  are independent random variables drawn from  $\mathcal{N}(0, 1)$  for all  $i, j$ . Properties of  $\mathcal{N}(0, 1)$ :  $E[a_i] = E[b_j] = 0$ ,  $E[a_i^2] = \text{Var}(a_i) + (E[a_i])^2 = 1 + 0 = 1$ ,  $E[b_j^2] = 1$ . The dot product is  $\mathbf{a}^T \mathbf{b} = \sum_{i=1}^D a_i b_i$ . We want to compute  $E[(\mathbf{a}^T \mathbf{b})^2] = E\left[\left(\sum_{i=1}^D a_i b_i\right)^2\right]$ .

$$\left(\sum_{i=1}^D a_i b_i\right)^2 = \left(\sum_{i=1}^D a_i b_i\right) \left(\sum_{j=1}^D a_j b_j\right) = \sum_{i=1}^D \sum_{j=1}^D a_i b_i a_j b_j$$

Now take the expectation:

$$E[(\mathbf{a}^T \mathbf{b})^2] = E\left[\sum_{i=1}^D \sum_{j=1}^D a_i b_i a_j b_j\right] = \sum_{i=1}^D \sum_{j=1}^D E[a_i b_i a_j b_j]$$

We consider two cases for the expectation  $E[a_i b_i a_j b_j]$ :

Case 1:  $i = j$ . The term is  $E[a_i b_i a_i b_i] = E[a_i^2 b_i^2]$ . Since  $a_i$  and  $b_i$  are independent:

$$E[a_i^2 b_i^2] = E[a_i^2] E[b_i^2] = (1)(1) = 1$$

Case 2:  $i \neq j$ . The term is  $E[a_i b_i a_j b_j]$ . Since all variables  $a_i, b_i, a_j, b_j$  are independent for  $i \neq j$ :

$$E[a_i b_i a_j b_j] = E[a_i] E[b_i] E[a_j] E[b_j] = (0)(0)(0)(0) = 0$$

Now substitute these back into the double summation:

$$\begin{aligned} E[(\mathbf{a}^T \mathbf{b})^2] &= \sum_{i=j} E[a_i^2 b_i^2] + \sum_{i \neq j} E[a_i b_i a_j b_j] \\ &= \sum_{i=1}^D (1) + \sum_{i \neq j} (0) = \sum_{i=1}^D 1 = D \end{aligned}$$

Thus,  $E[(\mathbf{a}^T \mathbf{b})^2] = D$ . Since  $E[\mathbf{a}^T \mathbf{b}] = E[\sum a_i b_i] = \sum E[a_i] E[b_i] = 0$ , this also means  $\text{Var}(\mathbf{a}^T \mathbf{b}) = D$ .

## Exercise 12.5

**Show that multi-head attention defined by (12.19) can be rewritten in the form  $Y = \sum_{h=1}^H \tilde{H}_h W^{(h)}$  where  $\tilde{H}_h = \text{Attention}(XW_h^{(q)}, XW_h^{(k)}, X)$  and  $W^{(h)} = W_h^{(v)} W_h^{(o)}$ .** (Note: The definition of  $\tilde{H}_h$  seems different from the text's  $H_h$ . Let's assume the question meant  $H_h$  uses  $V_h = XW_h^{(v)}$  as in the text and aims to show  $Y = \sum_h \text{Attention}(Q_h, K_h, V_h) W_h^{(o)}$ .)

**Solution.** Multi-head attention output (12.19) is:

$$Y = \text{Concat}[H_1, \dots, H_H] W^{(o)}$$

where  $H_h = \text{Attention}(Q_h, K_h, V_h) = \text{softmax} \left[ \frac{Q_h K_h^T}{\sqrt{D_k}} \right] V_h$ .  $Q_h = XW_h^{(q)}$ ,  $K_h = XW_h^{(k)}$ ,  $V_h = XW_h^{(v)}$ .  $H_h$  has dimensions  $N \times D_v$ . The concatenated matrix  $H_{\text{concat}} = \text{Concat}[H_1, \dots, H_H]$  has dimensions  $N \times (HD_v)$ . The output weight matrix  $W^{(o)}$  has dimensions  $(HD_v) \times D$ . The final output  $Y$  has dimensions  $N \times D$ .

Let's partition  $W^{(o)}$  horizontally into  $H$  blocks, each corresponding to one head  $H_h$ . Let  $W_h^{(o)}$  be the  $h$ -th block, with dimensions  $D_v \times D$ . (Note: This assumes concatenation along the feature dimension, and requires  $W^{(o)}$  rows to match  $HD_v$ ).

$$W^{(o)} = \begin{pmatrix} W_1^{(o)} \\ W_2^{(o)} \\ \vdots \\ W_H^{(o)} \end{pmatrix}$$

This doesn't seem right dimensionally for  $H_{\text{concat}} W^{(o)}$ .  $H_{\text{concat}}$  is  $N \times (HD_v)$ ,  $W^{(o)}$  is  $(HD_v) \times D$ .

Let's assume the text meant  $W^{(o)}$  is partitioned vertically corresponding to the concatenated blocks of  $H_{\text{concat}}$ . Let  $H_{\text{concat}} = [H_1 | H_2 | \dots | H_H]$ . Let  $W^{(o)}$  be partitioned into  $H$  horizontal blocks:

$$W^{(o)} = \begin{pmatrix} W_1^{(o)} \\ W_2^{(o)} \\ \vdots \\ W_H^{(o)} \end{pmatrix}$$

where each  $W_h^{(o)}$  is a submatrix of size  $D_v \times D$ . Then the matrix multiplication  $H_{\text{concat}} W^{(o)}$  becomes:

$$Y = [H_1 | H_2 | \dots | H_H] \begin{pmatrix} W_1^{(o)} \\ W_2^{(o)} \\ \vdots \\ W_H^{(o)} \end{pmatrix} = \sum_{h=1}^H H_h W_h^{(o)}$$

This uses the block matrix multiplication rule. This matches the form  $Y = \sum_{h=1}^H H_h W^{(h)}$  if we define  $W^{(h)} = W_h^{(o)}$ .

However, the question defines  $W^{(h)} = W_h^{(v)} W_h^{(o)}$ . This implies a potential redefinition or simplification. Let's re-read the text around Fig 12.7 and Algo 12.2. The text notes "redundancy in

the successive multiplication of the  $W^{(v)}$  matrix for each head and the output matrix  $W^{(o)}$  and suggests rewriting as a sum.

Consider the definition  $H_h = \text{Softmax}[\dots]V_h = A_h V_h = A_h X W_h^{(v)}$ . Then  $Y = \sum_h H_h W_h^{(o)} = \sum_h (A_h X W_h^{(v)}) W_h^{(o)} = \sum_h A_h X (W_h^{(v)} W_h^{(o)})$ . Let  $W^{(h)} = W_h^{(v)} W_h^{(o)}$ .  $Y = \sum_h A_h X W^{(h)}$ . Let  $\tilde{H}_h = A_h X = \text{softmax}\left[\frac{Q_h K_h^T}{\sqrt{D_k}}\right] X$ . This uses the original input  $X$  as the value matrix for calculating  $\tilde{H}_h$ . Then  $Y = \sum_h \tilde{H}_h W^{(h)}$ . This matches the form required by the question, where  $\tilde{H}_h = \text{Attention}(X W_h^{(q)}, X W_h^{(k)}, X)$  (using  $X$  as value) and  $W^{(h)} = W_h^{(v)} W_h^{(o)}$ . This reformulation separates the attention weights calculation from the value transformation.

---

## Exercise 12.6

**Express the self-attention function (12.14) as a fully connected network in the form of a matrix that maps the full input sequence of concatenated word vectors into an output vector of the same dimension. Note that such a matrix would have  $\mathcal{O}(N^2 D^2)$  parameters. Show that the self-attention network corresponds to a sparse version of this matrix with parameter sharing. Draw a sketch showing the structure of this matrix.**

**Solution.** Let the input sequence be represented by stacking the vectors  $\mathbf{x}_1, \dots, \mathbf{x}_N$  into a single large vector  $\mathbf{X}_{stack} = [\mathbf{x}_1^T, \dots, \mathbf{x}_N^T]^T$  of dimension  $ND$ . Similarly, the output is  $\mathbf{Y}_{stack} = [\mathbf{y}_1^T, \dots, \mathbf{y}_N^T]^T$ . A fully connected layer mapping  $\mathbf{X}_{stack}$  to  $\mathbf{Y}_{stack}$  would be  $\mathbf{Y}_{stack} = \mathbf{W}_{full} \mathbf{X}_{stack} + \mathbf{b}_{full}$ , where  $\mathbf{W}_{full}$  is an  $(ND) \times (ND)$  matrix. This has  $(ND)^2 = N^2 D^2$  parameters (ignoring bias).

The self-attention output (12.14) for a single head (assuming  $D_k = D_v = D$ ) is:

$$\mathbf{y}_n = \sum_{m=1}^N a_{nm} \mathbf{v}_m$$

where  $\mathbf{v}_m$  is the  $m$ -th row of  $V = XW^{(v)}$ , and  $a_{nm}$  is the  $(n, m)$ -th element of  $A = \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right)$ , with  $Q = XW^{(q)}$ ,  $K = XW^{(k)}$ .

$$\mathbf{y}_n = \sum_{m=1}^N a_{nm} (XW^{(v)})_m = \sum_{m=1}^N a_{nm} \sum_{j=1}^D (\mathbf{x}_m)_j W_{j,:}^{(v)}$$

$$(\mathbf{y}_n)_l = \sum_{m=1}^N a_{nm} (\mathbf{v}_m)_l = \sum_{m=1}^N a_{nm} \sum_{p=1}^D (\mathbf{x}_m)_p (W^{(v)})_{pl}$$

The attention weights  $a_{nm}$  depend non-linearly on all  $\mathbf{x}_{n'}$  via the softmax and the  $Q, K$  matrices.

$$a_{nm} = \text{softmax}_m \left( \frac{(XW^{(q)})_n^T (XW^{(k)})_m}{\sqrt{D}} \right) = \text{softmax}_m \left( \frac{(\sum_p x_{np} W_{p,:}^{(q)}) (\sum_r x_{mr} W_{r,:}^{(k)})^T}{\sqrt{D}} \right)$$

This mapping  $\mathbf{X}_{stack} \rightarrow \mathbf{Y}_{stack}$  is highly non-linear due to the dependence of the attention weights  $a_{nm}$  on the input  $X$ . The question seems to imply comparing it to a \*linear\* fully connected layer, which might be misleading.

However, let's consider the structure if the attention weights  $a_{nm}$  were \*fixed\*. Then the transformation  $\mathbf{Y}_{stack} = \mathbf{W}_{att}\mathbf{X}_{stack}$  would be linear.

$$(\mathbf{y}_n)_l = \sum_{m=1}^N a_{nm} \sum_{p=1}^D (\mathbf{x}_m)_p (W^{(v)})_{pl}$$

This relates output block  $n$ , element  $l$  to input block  $m$ , element  $p$ . The effective weight connecting input  $(\mathbf{x}_m)_p$  to output  $(\mathbf{y}_n)_l$  would be  $a_{nm}(W^{(v)})_{pl}$ . The full matrix  $\mathbf{W}_{att}$  would be a block matrix where the  $(n, m)$ -th block (of size  $D \times D$ ) is  $a_{nm}W^{(v)}$ .

$$\mathbf{W}_{att} = \begin{pmatrix} a_{11}W^{(v)} & a_{12}W^{(v)} & \dots & a_{1N}W^{(v)} \\ a_{21}W^{(v)} & a_{22}W^{(v)} & \dots & a_{2N}W^{(v)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1}W^{(v)} & a_{N2}W^{(v)} & \dots & a_{NN}W^{(v)} \end{pmatrix}$$

This matrix has  $N^2D^2$  elements. However, they are not independent parameters.

- **Parameter Sharing:** All blocks share the same underlying weight matrix  $W^{(v)}$  (containing  $D^2$  parameters, assuming  $D_v = D$ ).
- **Sparsity:** This interpretation doesn't yield sparsity in the usual sense (many zeros).
- **Input Dependence:** Crucially, the coefficients  $a_{nm}$  are not fixed parameters but depend on the input  $X$  via  $W^{(q)}$  and  $W^{(k)}$  ( $D^2$  parameters each).

The total number of parameters is  $3D^2$  (for  $W^{(q)}, W^{(k)}, W^{(v)}$ ), which is much less than  $N^2D^2$  if  $N$  is large.

If we were to interpret the attention mechanism itself ( $A = \text{softmax}(QK^T/\sqrt{D})$ ) as defining weights, this also shows structure. The calculation of  $QK^T$  involves shared weights  $W^{(q)}, W^{(k)}$ . The softmax introduces non-linearity. The final step  $Y = AV$  involves shared weights  $W^{(v)}$ .

Sketch: Imagine the large  $ND \times ND$  matrix. It's composed of  $N \times N$  blocks, each  $D \times D$ . All blocks are scalar multiples ( $a_{nm}$ ) of the same matrix  $W^{(v)}$ . These scalars  $a_{nm}$  are computed dynamically based on the input  $X$  and shared matrices  $W^{(q)}, W^{(k)}$ .

This illustrates the extensive parameter sharing compared to a fully independent  $N^2D^2$  parameter matrix.

## Exercise 12.7

**Show that if we omit the positional encoding of input vectors then the outputs of a multi-head attention layer defined by (12.19) are equivariant with respect to a re-ordering of the input sequence.**

**Solution.** Let  $X$  be the  $N \times D$  input matrix without positional encoding. Let  $P$  be an  $N \times N$  permutation matrix representing a reordering of the input tokens (rows). The permuted input matrix is  $X' = PX$ . We want to show that the output  $Y'$  obtained by applying multi-head attention to  $X'$  is equal to the permuted original output  $PY$ , where  $Y$  is the output for  $X$ .

$Y = \text{MultiHead}(X)$ .  $Y' = \text{MultiHead}(X')$ . We need to show  $Y' = PY$ .

First, consider the query, key, value matrices for a single head  $h$  with the permuted input  $X'$ :

$$Q'_h = X'W_h^{(q)} = (PX)W_h^{(q)} = P(XW_h^{(q)}) = PQ_h$$

$$K'_h = X'W_h^{(k)} = (PX)W_h^{(k)} = P(XW_h^{(k)}) = PK_h$$

$$V'_h = X'W_h^{(v)} = (PX)W_h^{(v)} = P(XW_h^{(v)}) = PV_h$$

Now, compute the attention scores for this head with permuted inputs:

$$Q'_h(K'_h)^T = (PQ_h)(PK_h)^T = PQ_hK_h^T P^T$$

The softmax is applied row-wise. Let  $S = \frac{Q_hK_h^T}{\sqrt{D_k}}$  and  $S' = \frac{Q'_h(K'_h)^T}{\sqrt{D_k}} = \frac{PSP^T}{\sqrt{D_k}}$ . Let  $A = \text{softmax}(S)$  and  $A' = \text{softmax}(S')$ . The  $(n, m)$ -th element of  $S'$  is  $(S')_{nm} = (\frac{PSP^T}{\sqrt{D_k}})_{nm} = \frac{1}{\sqrt{D_k}} \sum_i \sum_j P_{ni} S_{ij} (P^T)_{jm}$ . Since  $P$  is a permutation matrix,  $P_{ni} = 1$  if row  $i$  of  $S$  is moved to row  $n$  of  $S'$ , and 0 otherwise. Let  $n = \pi(i)$  and  $m = \pi(j)$  define the permutation  $\pi$ . Then  $P_{ni} = \delta_{n, \pi(i)}$  and  $(P^T)_{jm} = P_{mj} = \delta_{m, \pi(j)}$ .

$$(S')_{nm} = \frac{1}{\sqrt{D_k}} S_{\pi^{-1}(n), \pi^{-1}(m)}$$

Applying softmax row-wise means normalizing across the columns  $m$ :

$$(A')_{nm} = \frac{\exp((S')_{nm})}{\sum_{m'} \exp((S')_{nm'})} = \frac{\exp(S_{\pi^{-1}(n), \pi^{-1}(m)} / \sqrt{D_k})}{\sum_{m'} \exp(S_{\pi^{-1}(n), \pi^{-1}(m')} / \sqrt{D_k})}$$

Let  $i = \pi^{-1}(n)$ .

$$(A')_{nm} = \frac{\exp(S_{i, \pi^{-1}(m)} / \sqrt{D_k})}{\sum_{m'} \exp(S_{i, \pi^{-1}(m')} / \sqrt{D_k})}$$

Let  $j = \pi^{-1}(m)$  and  $j' = \pi^{-1}(m')$ . As  $m'$  ranges over  $1..N$ ,  $j'$  also ranges over  $1..N$ .

$$(A')_{nm} = \frac{\exp(S_{ij} / \sqrt{D_k})}{\sum_{j'} \exp(S_{ij'} / \sqrt{D_k})} = (A)_{ij} = (A)_{\pi^{-1}(n), \pi^{-1}(m)}$$

In matrix notation, this means  $A' = PAP^T$ .

Now compute the output of the head  $H'_h = A'V'_h$ :

$$H'_h = (PAP^T)(PV_h) = PA(P^T P)V_h$$

Since  $P$  is a permutation matrix, it is orthogonal, so  $P^T P = I$ .

$$H'_h = PAV_h = PH_h$$

The output of each head is permuted in the same way as the input. Finally, the output of the multi-head layer is:

$$Y' = \text{Concat}[H'_1, \dots, H'_H]W^{(o)} = \text{Concat}[PH_1, \dots, PH_H]W^{(o)}$$

Since concatenation acts along the feature dimension (columns) and  $P$  acts on rows, we can factor  $P$  out:

$$Y' = P(\text{Concat}[H_1, \dots, H_H])W^{(o)} = PY$$

Thus, the output is permuted in exactly the same way as the input, demonstrating permutation equivariance. Adding positional encoding breaks this property because the encoding added to  $x_n$  depends on its absolute position  $n$ , not just its value.

## Exercise 12.8

Consider two  $D$ -dimensional unit vectors  $\mathbf{a}$  and  $\mathbf{b}$ , satisfying  $\|\mathbf{a}\| = \|\mathbf{b}\| = 1$ , drawn from a random distribution that is symmetrical around the origin. Show that for large  $D$  the magnitude of the cosine of the angle between these vectors is close to zero (nearly orthogonal).

**Solution.** Let  $\theta$  be the angle between  $\mathbf{a}$  and  $\mathbf{b}$ . The cosine of the angle is given by:

$$\cos \theta = \frac{\mathbf{a}^T \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$$

Since  $\|\mathbf{a}\| = \|\mathbf{b}\| = 1$ , we have  $\cos \theta = \mathbf{a}^T \mathbf{b}$ . We want to show that  $|\cos \theta|$  is likely to be small for large  $D$ . Let  $\mathbf{a} = [a_1, \dots, a_D]^T$  and  $\mathbf{b} = [b_1, \dots, b_D]^T$ .  $\mathbf{a}^T \mathbf{b} = \sum_{i=1}^D a_i b_i$ . The vectors are drawn from a distribution symmetric around the origin and normalized. A common example is drawing from  $\mathcal{N}(0, \mathbf{I}/D)$  and normalizing, or simply drawing uniformly from the surface of the unit sphere.

Let's assume the components  $a_i$  and  $b_i$  are approximately independent with zero mean and variance  $E[a_i^2] \approx 1/D$  (since  $\sum E[a_i^2] = E[\sum a_i^2] = E[\|\mathbf{a}\|^2] = 1$ ). Similarly  $E[b_i^2] \approx 1/D$ . Assume  $a_i$  and  $b_j$  are independent for all  $i, j$  due to random sampling.

Consider the expectation of the dot product:

$$E[\mathbf{a}^T \mathbf{b}] = E\left[\sum_i a_i b_i\right] = \sum_i E[a_i b_i] = \sum_i E[a_i] E[b_i]$$

Since the distribution is symmetric around the origin,  $E[a_i] = E[b_i] = 0$ . So,  $E[\mathbf{a}^T \mathbf{b}] = 0$ .

Now consider the variance (or expected squared value) of the dot product:

$$\begin{aligned} \text{Var}(\mathbf{a}^T \mathbf{b}) &= E[(\mathbf{a}^T \mathbf{b})^2] - (E[\mathbf{a}^T \mathbf{b}])^2 = E\left[\left(\sum_i a_i b_i\right)^2\right] \\ &= E\left[\sum_i \sum_j a_i b_i a_j b_j\right] = \sum_i \sum_j E[a_i b_i a_j b_j] \end{aligned}$$

If  $i = j$ :  $E[a_i^2 b_i^2] = E[a_i^2] E[b_i^2] \approx (1/D)(1/D) = 1/D^2$ . If  $i \neq j$ :  $E[a_i b_i a_j b_j] = E[a_i] E[b_i] E[a_j] E[b_j] = 0$ .

$$E[(\mathbf{a}^T \mathbf{b})^2] = \sum_{i=1}^D E[a_i^2 b_i^2] + \sum_{i \neq j} 0 = \sum_{i=1}^D (1/D^2) = D \cdot (1/D^2) = 1/D$$

So, the variance of the dot product  $\mathbf{a}^T \mathbf{b}$  is  $1/D$ . As the dimension  $D$  becomes large, the variance  $\text{Var}(\mathbf{a}^T \mathbf{b}) \rightarrow 0$ . Since the mean is 0 and the variance approaches 0, the value of the dot product  $\mathbf{a}^T \mathbf{b} = \cos \theta$  concentrates around 0. This implies that the magnitude  $|\cos \theta|$  is likely to be small, and the vectors are nearly orthogonal.

## Exercise 12.9

Consider a position encoding in which the input token vector  $\mathbf{x}$  is concatenated with a position-encoding vector  $\mathbf{e}$ . Show that when this concatenated vector undergoes a general linear transformation by multiplication using a matrix, the result can be expressed as the sum of a linearly transformed input and a linearly transformed position vector.

**Solution.** Let the input token vector be  $\mathbf{x} \in \mathbb{R}^D$  and the position-encoding vector be  $\mathbf{r} \in \mathbb{R}^P$ . The concatenated vector is  $\mathbf{z} = [\mathbf{x}^T, \mathbf{r}^T]^T \in \mathbb{R}^{D+P}$ . Let the linear transformation be represented by a matrix  $\mathbf{W}$  of size  $D_{out} \times (D + P)$ . The transformed vector  $\mathbf{y}$  is:

$$\mathbf{y} = \mathbf{W}\mathbf{z}$$

We can partition the matrix  $\mathbf{W}$  horizontally into two blocks corresponding to the dimensions of  $\mathbf{x}$  and  $\mathbf{r}$ :

$$\mathbf{W} = [\mathbf{W}_x | \mathbf{W}_r]$$

where  $\mathbf{W}_x$  has size  $D_{out} \times D$  and  $\mathbf{W}_r$  has size  $D_{out} \times P$ . Now perform the multiplication using block matrix notation:

$$\begin{aligned} \mathbf{y} &= [\mathbf{W}_x | \mathbf{W}_r] \begin{pmatrix} \mathbf{x} \\ \mathbf{r} \end{pmatrix} \\ \mathbf{y} &= \mathbf{W}_x \mathbf{x} + \mathbf{W}_r \mathbf{r} \end{aligned}$$

Let  $\mathbf{y}_x = \mathbf{W}_x \mathbf{x}$  be the linearly transformed input token vector. Let  $\mathbf{y}_r = \mathbf{W}_r \mathbf{r}$  be the linearly transformed position vector. Then  $\mathbf{y} = \mathbf{y}_x + \mathbf{y}_r$ . The result is indeed the sum of a linearly transformed input vector and a linearly transformed position vector. This shows that additive positional encoding (as used in transformers,  $\tilde{\mathbf{x}}_n = \mathbf{x}_n + \mathbf{r}_n$ ) can be learned equivalently by subsequent linear layers even if concatenation were used initially, provided the dimensions match.

## Exercise 12.10

Show that the positional encoding defined by (12.25) has the property that, for a fixed offset  $k$ , the encoding at position  $n + k$  can be represented as a linear combination of the encoding at position  $n$  with coefficients that depend only on  $k$  and not on  $n$ . Show that if the encoding is based purely on sine functions, without cosine functions, then this property no longer holds.

**Solution.** The positional encoding vector  $\mathbf{r}_n$  has components given by (12.25):

$$r_{n,2i} = \sin(n/L^{2i/D})$$

$$r_{n,2i+1} = \cos(n/L^{2i/D})$$

(Assuming 0-based indexing for components  $0..D - 1$ , where  $i$  ranges from 0 to  $D/2 - 1$ ). Let  $\omega_i = 1/L^{2i/D}$ .

$$r_{n,2i} = \sin(n\omega_i)$$

$$r_{n,2i+1} = \cos(n\omega_i)$$

Consider the components at position  $n + k$ :

$$r_{n+k,2i} = \sin((n + k)\omega_i) = \sin(n\omega_i + k\omega_i)$$

$$r_{n+k,2i+1} = \cos((n + k)\omega_i) = \cos(n\omega_i + k\omega_i)$$

Using the trigonometric identities for sum of angles:  $\sin(A + B) = \sin A \cos B + \cos A \sin B$   $\cos(A + B) = \cos A \cos B - \sin A \sin B$  Let  $A = n\omega_i$  and  $B = k\omega_i$ .

$$r_{n+k,2i} = \sin(n\omega_i) \cos(k\omega_i) + \cos(n\omega_i) \sin(k\omega_i)$$

$$\begin{aligned}
r_{n+k,2i} &= r_{n,2i} \cos(k\omega_i) + r_{n,2i+1} \sin(k\omega_i) \\
r_{n+k,2i+1} &= \cos(n\omega_i) \cos(k\omega_i) - \sin(n\omega_i) \sin(k\omega_i) \\
r_{n+k,2i+1} &= r_{n,2i+1} \cos(k\omega_i) - r_{n,2i} \sin(k\omega_i)
\end{aligned}$$

We can write this as a matrix transformation for each pair of dimensions  $(2i, 2i + 1)$ :

$$\begin{pmatrix} r_{n+k,2i} \\ r_{n+k,2i+1} \end{pmatrix} = \begin{pmatrix} \cos(k\omega_i) & \sin(k\omega_i) \\ -\sin(k\omega_i) & \cos(k\omega_i) \end{pmatrix} \begin{pmatrix} r_{n,2i} \\ r_{n,2i+1} \end{pmatrix}$$

This shows that the vector components  $(r_{n+k,2i}, r_{n+k,2i+1})$  are a linear transformation of the components  $(r_{n,2i}, r_{n,2i+1})$ . The transformation matrix depends only on the offset  $k$  (and the frequency  $\omega_i$ ) but not on the position  $n$ . Since this holds for all pairs of dimensions, the entire vector  $\mathbf{r}_{n+k}$  is a linear transformation of  $\mathbf{r}_n$ , where the transformation matrix depends only on  $k$ .

Now, consider an encoding based purely on sine functions:  $r_{n,i} = \sin(n\omega_i)$ .

$$\begin{aligned}
r_{n+k,i} &= \sin((n+k)\omega_i) = \sin(n\omega_i) \cos(k\omega_i) + \cos(n\omega_i) \sin(k\omega_i) \\
r_{n+k,i} &= r_{n,i} \cos(k\omega_i) + \cos(n\omega_i) \sin(k\omega_i)
\end{aligned}$$

This expression contains  $\cos(n\omega_i)$ , which is not part of the original encoding vector  $\mathbf{r}_n$  (which only contains sine terms). Therefore,  $r_{n+k,i}$  cannot be expressed purely as a linear transformation of the components of  $\mathbf{r}_n$  with coefficients depending only on  $k$ . The property requires both sine and cosine terms.

## Exercise 12.11

**Consider the bag-of-words model (12.28)**  $p(\mathbf{x}_1, \dots, \mathbf{x}_N) = \prod_{n=1}^N p(\mathbf{x}_n)$ , **where**  $p(\mathbf{x})$  **is a shared probability table. Show that the maximum likelihood solution for**  $p(\mathbf{x})$  **is given by the empirical frequencies.**

**Solution.** Let the vocabulary size be  $V$ . The distribution  $p(\mathbf{x})$  is defined by parameters  $\theta_k = p(\mathbf{x} = \text{word}_k)$  for  $k = 1, \dots, V$ , subject to  $\theta_k \geq 0$  and  $\sum_{k=1}^V \theta_k = 1$ . The training data consists of  $N$  words  $D = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ . The likelihood function is:

$$L(\boldsymbol{\theta}; D) = p(\mathbf{x}_1, \dots, \mathbf{x}_N | \boldsymbol{\theta}) = \prod_{n=1}^N p(\mathbf{x}_n | \boldsymbol{\theta})$$

Let  $N_k$  be the number of times word  $k$  appears in the data  $D$ . Then  $\sum_{k=1}^V N_k = N$ . We can rewrite the likelihood by grouping identical words:

$$L(\boldsymbol{\theta}; D) = \prod_{k=1}^V \theta_k^{N_k}$$

The log-likelihood is:

$$\ln L(\boldsymbol{\theta}; D) = \sum_{k=1}^V N_k \ln \theta_k$$

We want to maximize  $\ln L$  subject to the constraint  $\sum_{k=1}^V \theta_k = 1$ . We use a Lagrange multiplier  $\lambda$ :

$$\mathcal{L}(\theta, \lambda) = \sum_{k=1}^V N_k \ln \theta_k - \lambda \left( \sum_{k=1}^V \theta_k - 1 \right)$$

Take the derivative with respect to  $\theta_j$  and set to zero:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta_j} &= \frac{N_j}{\theta_j} - \lambda = 0 \\ \theta_j &= \frac{N_j}{\lambda} \end{aligned}$$

To find  $\lambda$ , use the constraint  $\sum_{k=1}^V \theta_k = 1$ :

$$\begin{aligned} \sum_{k=1}^V \frac{N_k}{\lambda} &= 1 \\ \frac{1}{\lambda} \sum_{k=1}^V N_k &= 1 \\ \frac{1}{\lambda} N &= 1 \implies \lambda = N \end{aligned}$$

Substitute  $\lambda = N$  back into the expression for  $\theta_j$ :

$$\theta_j = \frac{N_j}{N}$$

Thus, the maximum likelihood estimate for the probability of word  $j$  is its empirical frequency (count divided by total number of words) in the training set.

## Exercise 12.12

**Consider the autoregressive language model  $p(\mathbf{x}_1, \dots, \mathbf{x}_N) = \prod_{n=1}^N p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1})$ . Suppose the terms  $p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1})$  are represented by general probability tables. Show that the number of entries in these tables grows exponentially with  $n$ .**

**Solution.** Let  $V$  be the size of the vocabulary (number of possible values for each  $\mathbf{x}_n$ ). Consider the conditional distribution  $p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1})$ . This distribution defines the probability of  $\mathbf{x}_n$  for every possible history  $(\mathbf{x}_1, \dots, \mathbf{x}_{n-1})$ . The number of possible histories  $(\mathbf{x}_1, \dots, \mathbf{x}_{n-1})$  is  $V^{n-1}$ , since each of the  $n-1$  previous words can take  $V$  possible values. For each specific history, the conditional distribution  $p(\mathbf{x}_n | \text{history})$  must specify the probabilities for the  $V$  possible values of  $\mathbf{x}_n$ . Since these probabilities must sum to 1, there are  $V-1$  independent parameters for each history. Therefore, the total number of independent parameters needed to specify the conditional distribution  $p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1})$  using a general probability table is:

$$\begin{aligned} \text{Number of parameters} &= (\text{Number of histories}) \times (\text{Parameters per history}) \\ &= V^{n-1} \times (V-1) \end{aligned}$$

The total number of entries in the table (if storing all  $V$  probabilities per history) is  $V^{n-1} \times V = V^n$ . In either case, the number of entries or parameters required grows as  $V^n$  or  $O(V^n)$ , which is exponential in the sequence length  $n$ . This makes storing and estimating these tables infeasible for even moderately sized vocabularies and sequence lengths.

## Exercise 12.13

When using n-grams it is usual to train the L-gram and (L-1)-gram models and compute  $p(x_n|x_{n-L+1}, \dots, x_{n-1}) = \frac{p_L(x_{n-L+1}, \dots, x_n)}{p_{L-1}(x_{n-L+1}, \dots, x_{n-1})}$ . Explain why this is more convenient than storing the left-hand side directly, and show that to obtain correct probabilities the final token must be omitted when evaluating  $p_{L-1}(\dots)$ .

**Solution.** The formula expresses the conditional probability using the definition  $P(A|B) = P(A, B)/P(B)$ . Here  $A = x_n$  and  $B = (x_{n-L+1}, \dots, x_{n-1})$ . The joint probability  $P(A, B)$  corresponds to the L-gram probability  $p_L$ , and  $P(B)$  corresponds to the (L-1)-gram probability  $p_{L-1}$ .

**\*\*Convenience:\*\*** Storing the conditional probability  $p(x_n|x_{n-L+1}, \dots, x_{n-1})$  directly requires estimating and storing  $V^{L-1} \times (V-1)$  parameters. This requires counting occurrences of all L-grams and normalizing for each (L-1)-gram prefix. Alternatively, estimating the L-gram joint probability  $p_L(x_{n-L+1}, \dots, x_n)$  involves counting all L-grams and normalizing by the total number of L-grams counted. This requires storing  $V^L - 1$  parameters. Similarly, estimating  $p_{L-1}(x_{n-L+1}, \dots, x_{n-1})$  requires storing  $V^{L-1} - 1$  parameters by counting (L-1)-grams. The total storage for the joint probabilities  $p_L$  and  $p_{L-1}$  might seem larger ( $V^L + V^{L-1}$ ) than for the conditional ( $V^L$ ). However:

1. **\*\*Estimation:\*\*** Estimating joint probabilities from counts is straightforward (count / total count). Estimating conditionals requires counting L-grams and (L-1)-grams anyway for normalization  $\frac{\text{count}(x_{n-L+1}, \dots, x_n)}{\text{count}(x_{n-L+1}, \dots, x_{n-1})}$ .
2. **\*\*Smoothing:\*\*** Techniques for handling unseen n-grams (smoothing) are often applied more easily to the joint counts/probabilities  $p_L$  and  $p_{L-1}$  before calculating the conditional probability.
3. **\*\*Backoff/Interpolation:\*\*** N-gram models often combine probabilities from different orders (e.g., trigram, bigram, unigram). Storing  $p_L, p_{L-1}, \dots, p_1$  allows for easier implementation of such backoff or interpolation schemes. It's often necessary to compute lower-order n-gram probabilities anyway.

Therefore, calculating the conditional probability from stored joint/marginal probabilities of n-grams and (n-1)-grams is often more convenient for estimation, smoothing, and model combination, even if direct storage of conditionals seems notionally simpler.

**\*\*Omitting the Final Token for  $p_{L-1}$ :** The formula requires  $p_{L-1}(x_{n-L+1}, \dots, x_{n-1})$ , which is the probability of the prefix sequence of length  $L-1$ . When estimating this probability from a corpus, we need the counts of these  $(L-1)$ -grams. Consider a corpus  $w_1, w_2, \dots, w_N$ . To estimate  $p_L(w_1, \dots, w_L)$ , we count occurrences of  $(w_1, \dots, w_L)$  in the corpus. To estimate  $p_{L-1}(w_1, \dots, w_{L-1})$ , we count occurrences of  $(w_1, \dots, w_{L-1})$  in the corpus. The definition  $p(x_n|x_{n-L+1}, \dots, x_{n-1}) = \frac{p_L(x_{n-L+1}, \dots, x_n)}{p_{L-1}(x_{n-L+1}, \dots, x_{n-1})}$  requires that  $p_L$  and  $p_{L-1}$  represent consistent joint and marginal probabilities. If  $p_L$  is estimated by counting L-grams and normalizing, and  $p_{L-1}$  is estimated by counting (L-1)-grams and normalizing, the normalization constants might differ slightly depending on how beginnings/ends of sequences are handled. However, the ratio usually relies on counts:

$$p(x_n|x_{n-L+1}, \dots, x_{n-1}) \approx \frac{\text{Count}(x_{n-L+1}, \dots, x_n)}{\text{Count}(x_{n-L+1}, \dots, x_{n-1})}$$

This implicitly uses the empirical estimates for  $p_L$  and  $p_{L-1}$  based on counts. The probability  $p_{L-1}(x_{n-L+1}, \dots, x_{n-1})$  refers specifically to the probability of that  $(L-1)$ -gram sequence occurring. The text's phrasing "final token must be omitted" likely refers to the fact that the denominator probability is for the sequence \*excluding\* the token  $x_n$  whose conditional probability is being calculated.

## Exercise 12.15

Consider a sequence of two tokens  $y_1, y_2$  each  $\in \{A, B\}$ . The joint probability  $p(y_1, y_2)$  is given by the table. Show that greedy search finds a different sequence than the overall most probable one. Find the probability of the sequence found by greedy search.

	$y_1 = A$	$y_1 = B$
$y_2 = A$	0.0	0.4
$y_2 = B$	0.1	0.25

**Solution.** The joint probabilities are:  $p(A, A) = 0.0$   $p(A, B) = 0.1$   $p(B, A) = 0.4$   $p(B, B) = 0.25$  (Sum =  $0.0 + 0.1 + 0.4 + 0.25 = 0.75$ . The table seems incorrect as probabilities should sum to 1. Assuming the value for (B, B) should be 0.5 to make the sum 1.0). Let's use the table as given, normalizing if needed, or assuming the missing 0.25 probability lies elsewhere. Let's assume the table represents relative weights and renormalize. Total weight = 0.75. Normalized probabilities:  $p(A, A) = 0.0/0.75 = 0.0$   $p(A, B) = 0.1/0.75 = 2/15 \approx 0.133$   $p(B, A) = 0.4/0.75 = 8/15 \approx 0.533$   $p(B, B) = 0.25/0.75 = 1/3 \approx 0.333$  Sum  $\approx 0.0 + 0.133 + 0.533 + 0.333 = 0.999$ . Let's use the fractions.

The most probable sequence  $(y_1, y_2)$  is  $(B, A)$  with probability  $8/15 \approx 0.533$ .

Now, perform greedy search. We need  $p(y_1)$  and  $p(y_2|y_1)$ . Marginal  $p(y_1)$ :  $p(y_1 = A) = p(A, A) + p(A, B) = 0 + 2/15 = 2/15$   $p(y_1 = B) = p(B, A) + p(B, B) = 8/15 + 5/15 = 13/15$  (Check:  $2/15 + 13/15 = 1$ ).

Step 1: Choose  $y_1$  greedily. Maximize  $p(y_1)$ . The maximum is  $p(y_1 = B) = 13/15$ . So, greedy search chooses  $y_1^* = B$ .

Step 2: Choose  $y_2$  greedily. Maximize  $p(y_2|y_1 = y_1^* = B)$ . We need the conditional distribution  $p(y_2|y_1 = B)$ .  $p(y_2 = A|y_1 = B) = p(B, A)/p(y_1 = B) = (8/15)/(13/15) = 8/13$   $p(y_2 = B|y_1 = B) = p(B, B)/p(y_1 = B) = (5/15)/(13/15) = 5/13$  (Check:  $8/13 + 5/13 = 1$ ). The maximum conditional probability is  $p(y_2 = A|y_1 = B) = 8/13$ . So, greedy search chooses  $y_2^* = A$ .

The sequence found by greedy search is  $(y_1^*, y_2^*) = (B, A)$ . In this specific case, the greedy sequence  $(B, A)$  is the most probable sequence, which had probability  $p(B, A) = 8/15$ .

Let's re-examine the table and the question text's assertion. The text states "the most probable sequence is  $y_1 = B, y_2 = B$  and that this has probability 0.4". This implies the table values are the probabilities and they don't sum to 1, or my initial interpretation of the table was wrong. Let's use the text's assertion. If  $p(B, B) = 0.4$  is the max probability, then the table should be:

	$y_1 = A$	$y_1 = B$
$y_2 = A$	?	?
$y_2 = B$	?	0.4

than the one printed.

The example in the text must refer to a different probability table

Let's use the printed table again but assume  $p(B, B) = 0.25$  was intended and  $p(B, A) = 0.4$ .  $p(A, A) = 0.0$   $p(A, B) = 0.1$   $p(B, A) = 0.4$   $p(B, B) = 0.25$  (Sum = 0.75. Let's proceed assuming this is correct and maybe renormalization is implicit). Most probable sequence:  $(B, A)$  with probability mass 0.4.

Greedy search:  $p(y_1 = A) = 0.0 + 0.1 = 0.1$   $p(y_1 = B) = 0.4 + 0.25 = 0.65$  Step 1: Choose  $y_1^* = B$

(since  $0.65 > 0.1$ ). Step 2: Maximize  $p(y_2|y_1 = B)$ .  $p(y_2 = A|y_1 = B) = p(B, A)/p(y_1 = B) = 0.4/0.65 = 40/65 = 8/13$   $p(y_2 = B|y_1 = B) = p(B, B)/p(y_1 = B) = 0.25/0.65 = 25/65 = 5/13$  Maximum is  $p(y_2 = A|y_1 = B)$ . Choose  $y_2^* = A$ . Greedy sequence:  $(B, A)$ . Probability = 0.4.

Again, greedy finds the most probable sequence. The exercise description seems to refer to a different table where this is not the case. Let's construct such a table. Suppose:  $p(A, A) = 0.1$   $p(A, B) = 0.2$   $p(B, A) = 0.3$   $p(B, B) = 0.4$  (Sum = 1.0) Most probable sequence:  $(B, B)$  with probability 0.4.

Greedy search:  $p(y_1 = A) = 0.1 + 0.2 = 0.3$   $p(y_1 = B) = 0.3 + 0.4 = 0.7$  Step 1: Choose  $y_1^* = B$ . Step 2: Maximize  $p(y_2|y_1 = B)$ .  $p(y_2 = A|y_1 = B) = p(B, A)/p(y_1 = B) = 0.3/0.7 = 3/7$   $p(y_2 = B|y_1 = B) = p(B, B)/p(y_1 = B) = 0.4/0.7 = 4/7$  Maximum is  $p(y_2 = B|y_1 = B)$ . Choose  $y_2^* = B$ . Greedy sequence:  $(B, B)$ . Probability = 0.4.

Still matches. Let's try another table.  $p(A, A) = 0.1$   $p(A, B) = 0.4$   $p(B, A) = 0.3$   $p(B, B) = 0.2$  (Sum = 1.0) Most probable sequence:  $(A, B)$  with probability 0.4.

Greedy search:  $p(y_1 = A) = 0.1 + 0.4 = 0.5$   $p(y_1 = B) = 0.3 + 0.2 = 0.5$  Tie! Assume we choose  $y_1^* = A$ . Step 2: Maximize  $p(y_2|y_1 = A)$ .  $p(y_2 = A|y_1 = A) = p(A, A)/p(y_1 = A) = 0.1/0.5 = 1/5$   $p(y_2 = B|y_1 = A) = p(A, B)/p(y_1 = A) = 0.4/0.5 = 4/5$  Maximum is  $p(y_2 = B|y_1 = A)$ . Choose  $y_2^* = B$ . Greedy sequence:  $(A, B)$ . Probability = 0.4.

If we had chosen  $y_1^* = B$  in the tie: Step 2: Maximize  $p(y_2|y_1 = B)$ .  $p(y_2 = A|y_1 = B) = p(B, A)/p(y_1 = B) = 0.3/0.5 = 3/5$   $p(y_2 = B|y_1 = B) = p(B, B)/p(y_1 = B) = 0.2/0.5 = 2/5$  Maximum is  $p(y_2 = A|y_1 = B)$ . Choose  $y_2^* = A$ . Greedy sequence:  $(B, A)$ . Probability = 0.3.

In this case, one of the greedy paths finds the optimal sequence. The exercise statement implies there exists a case where greedy search definitively finds a suboptimal sequence. Let's try to construct one. Need  $p(y_1^*)$  to be max, but  $p(y_1^{other})$  leads to a higher joint probability later.

Table 3:  $p(A, A) = 0.4$   $p(A, B) = 0.1$   $p(B, A) = 0.3$   $p(B, B) = 0.2$  (Sum = 1.0) Most probable sequence:  $(A, A)$  with probability 0.4.

Greedy search:  $p(y_1 = A) = 0.4 + 0.1 = 0.5$   $p(y_1 = B) = 0.3 + 0.2 = 0.5$  Tie! Assume  $y_1^* = A$ . Step 2: Maximize  $p(y_2|y_1 = A)$ .  $p(y_2 = A|y_1 = A) = p(A, A)/p(y_1 = A) = 0.4/0.5 = 4/5$   $p(y_2 = B|y_1 = A) = p(A, B)/p(y_1 = A) = 0.1/0.5 = 1/5$  Choose  $y_2^* = A$ . Greedy sequence:  $(A, A)$ . Probability = 0.4. Optimal.

Assume  $y_1^* = B$  from the tie. Step 2: Maximize  $p(y_2|y_1 = B)$ .  $p(y_2 = A|y_1 = B) = p(B, A)/p(y_1 = B) = 0.3/0.5 = 3/5$   $p(y_2 = B|y_1 = B) = p(B, B)/p(y_1 = B) = 0.2/0.5 = 2/5$  Choose  $y_2^* = A$ . Greedy sequence:  $(B, A)$ . Probability = 0.3. Suboptimal.

So, depending on tie-breaking, greedy search \*can\* yield a suboptimal sequence. If we modify Table 3 slightly to break the tie: Table 4:  $p(A, A) = 0.4$   $p(A, B) = 0.09$   $p(B, A) = 0.3$   $p(B, B) = 0.21$  (Sum = 1.0) Most probable sequence:  $(A, A)$  with probability 0.4.

Greedy search:  $p(y_1 = A) = 0.4 + 0.09 = 0.49$   $p(y_1 = B) = 0.3 + 0.21 = 0.51$  Step 1: Choose  $y_1^* = B$ . Step 2: Maximize  $p(y_2|y_1 = B)$ .  $p(y_2 = A|y_1 = B) = p(B, A)/p(y_1 = B) = 0.3/0.51 \approx 0.588$   $p(y_2 = B|y_1 = B) = p(B, B)/p(y_1 = B) = 0.21/0.51 \approx 0.412$  Choose  $y_2^* = A$ . Greedy sequence:  $(B, A)$ . Probability = 0.3. The most probable sequence is  $(A, A)$  with probability 0.4. Greedy search found the sequence  $(B, A)$  with probability 0.3. Thus, greedy search yields a different, suboptimal sequence.

## Exercise 12.16

The BERT-Large model has input length  $N = 512$ , embedding dimension  $D = 1024$ , vocabulary  $V = 30000$ . It has  $L = 24$  transformer layers, each with  $H = 16$  self-attention heads,  $D_q = D_k = D_v = 64$ . The MLP networks have two layers with 4096 hidden nodes. Show the total number of parameters is approx 340 million.

**Solution.** Total parameters = Parameters(Embedding) + Layers  $\times$  [Parameters(MultiHeadAttention) + Parameters(MLP)] + Parameters(Final Layer - if any, for pre-training usually tied to embedding).

1. **\*\*Embedding Layer:\*\*** Maps  $V$  vocabulary items to  $D$  dimensions. Matrix size:  $V \times D = 30000 \times 1024 \approx 30.7M$ . (Plus positional encoding parameters if learned, but assume fixed sinusoidal or added later).
2. **\*\*Multi-Head Self-Attention (MHSA) per Layer:\*\*** Each head  $h$  has  $W_h^{(q)}, W_h^{(k)} \in \mathbb{R}^{D \times D_k}$ ,  $W_h^{(v)} \in \mathbb{R}^{D \times D_v}$ . Given  $D = 1024$ ,  $H = 16$ ,  $D_k = D_v = 64$ . Note  $H \times D_v = 16 \times 64 = 1024 = D$ . Params per head for Q, K, V =  $(D \times D_k) + (D \times D_k) + (D \times D_v) = (1024 \times 64) + (1024 \times 64) + (1024 \times 64) = 3 \times 1024 \times 64 = 196,608$ . Total params for Q, K, V across  $H$  heads =  $H \times (3 \times D \times D_v) = 16 \times 196,608 = 3,145,728$ . Output projection matrix  $W^{(o)} \in \mathbb{R}^{(H D_v) \times D} = \mathbb{R}^{D \times D}$ . Params for  $W^{(o)} = D \times D = 1024 \times 1024 = 1,048,576$ . Total MHSA params per layer =  $3,145,728 + 1,048,576 = 4,194,304 \approx 4.2M$ . (Note: Biases are usually included.  $W^{(q)}, W^{(k)}, W^{(v)}$  have  $D_k, D_k, D_v$  biases per head.  $W^{(o)}$  has  $D$  biases. Total biases  $\approx H(D_k + D_k + D_v) + D = 16(64 + 64 + 64) + 1024 = 3072 + 1024 = 4096$ . Small compared to weights).
3. **\*\*MLP (Feed-Forward Network) per Layer:\*\*** Applied per position. Shared weights. Structure: Input( $D$ ) -> Hidden(4096) -> Output( $D$ ). Assuming standard MLP structure  $Linear \rightarrow ReLU \rightarrow Linear$ . Layer 1:  $W_1 \in \mathbb{R}^{D \times 4096}$ , Bias  $b_1 \in \mathbb{R}^{4096}$ . Params =  $1024 \times 4096 + 4096 = 4,194,304 + 4096 = 4,198,400$ . Layer 2:  $W_2 \in \mathbb{R}^{4096 \times D}$ , Bias  $b_2 \in \mathbb{R}^D$ . Params =  $4096 \times 1024 + 1024 = 4,194,304 + 1024 = 4,195,328$ . Total MLP params per layer =  $4,198,400 + 4,195,328 = 8,393,728 \approx 8.4M$ .
4. **\*\*Total Parameters per Layer:\*\*** Params(MHSA) + Params(LayerNorms) + Params(MLP) + Params(LayerNorms) LayerNorm parameters (scale and shift) are typically small ( $2 \times D$  per norm). Approx  $4 \times D = 4096$  per layer. Negligible. Total per layer  $\approx 4.2M + 8.4M = 12.6M$ .
5. **\*\*Total Parameters for L Layers:\*\***  $L \times$  (Params per layer) =  $24 \times 12.6M \approx 302.4M$ .
6. **\*\*Overall Total:\*\*** Params(Embedding) + Params(L Layers)  $\approx 30.7M + 302.4M \approx 333.1M$ .

This is close to 340 million. The small discrepancies could be due to: - Inclusion/exclusion of biases (adds about  $24 \times 4096 \approx 0.1M$ , negligible). - LayerNorm parameters (adds about  $24 \times 4096 \approx 0.1M$ , negligible). - Exact definition of  $D_q, D_k$  in the paper (sometimes  $D_q = D_k$ ). Here  $D_k = D_v = 64$  is given. - Parameter tying between embedding and final prediction layer (common in pre-training, but not always). If the final linear projection back to vocabulary uses the transpose of the embedding matrix, it doesn't add new parameters. BERT uses a separate prediction layer, sometimes with tying. If untied, add  $D \times V \approx 30.7M$  params. Total  $\approx 333.1M + 30.7M = 363.8M$ . - Exact MLP hidden size might be different (e.g.,  $4 \times D$ ).  $4 \times 1024 = 4096$ . This seems correct.

Let's recheck MHSA. Often Q,K,V projections are combined:  $W_{QKV} \in \mathbb{R}^{D \times 3D_k}$  per head, or  $D \times 3D$  total if computed jointly then split. Total QKV params =  $3 \times D \times (H \times D_v) = 3 \times 1024 \times 1024 \approx 3.15M$ .  $W^{(o)}$  params =  $D \times D = 1024 \times 1024 \approx 1.05M$ . Total MHSA =  $3.15M + 1.05M = 4.2M$ . Matches.

Let's recheck MLP.  $W_1 : D \times 4D = 1024 \times 4096 \approx 4.2M$ .  $W_2 : 4D \times D = 4096 \times 1024 \approx 4.2M$ . Total MLP  $\approx 8.4M$ . Matches.

Total per layer  $\approx 12.6M$ . Total for 24 layers  $\approx 302.4M$ . Add Embedding  $\approx 30.7M$ . Total  $\approx 333.1M$ .

The original BERT paper reports 340M parameters for BERT-Large. The calculation above yields approx 333M. The difference might stem from specific bias implementations, LayerNorm parameters, or potentially segment embeddings/task-specific layers not included in this breakdown. However, 333M is approximately 340M.