Transformer Circuit Exercises

Max's solutions to transformer-circuits.pub/2021/exercises

August 23, 2023

Formatting note: I have provided my answers wrapped in a box under each question :)

Exercises & Answers

Warm Up

- Describe the transformer architecture at a high level.

> At a high level, a transformer consists of a stack of *attention* and *MLP* layers to perform sequence-to-sequence transformation. Loosely, each attention layer can be thought of as a right-sided matrix multiplication (VA), in which information is shared between embedded tokens, while MLP can be thought of as a left-sided matrix multiplication (WV), in which tokens are transformed in isolation. Each layer adds its output to a *residual stream* to encourage lookback.
>
> The key here is that attention is determined by vectors ("keys" pattern-matching against "queries") which are *separate* to the subject matter itself—for example, with attended values being determined by a separate encoder stack, or (in the case of GPT) by a linear transformation of the input embeddings. Expressibility is provided by each layer having non-linearities: typically, GeLU or ReLU activation for MLP, and softmax for attention.
>
> Typically, one optimises the transformer by dividing each attention-MLP block into multiple *heads*, each operating on a subspace of the original embeddings, with their outputs being merged by a linear transformation.

- Describe how an individual attention head works in detail, in terms of the matrices W_Q, W_K, W_V, and W_{out}. (The equations and code for an attention head are often written for all attention heads in a layer concatenated together at once. This implementation is more computationally efficient, but harder to reason about, so we'd like to describe a single attention head.)

> Each attention head can be thought of as a merging of value tokens V according to attention patterns A:
>
> $$H = VA,$$

where V is determined by a linear transformation of the residual stream / input embeddings E:

$$V = W_V E.$$

Side note: unlike Attention is All You Need [1], I use a column-wise notation here—i.e. each column in V representing one token—in line with the A Mathematical Framework for Transformer Circuits format.

In multi-head attention, we would concatenate and then linearly transform these outputs—but we can equivalently think of this as transforming each head's output in isolation before summing them.

Of course, given we are focusing on single-head attention, I will omit the sum, giving us an expression for O:

$$O = W_{out} H.$$

Each output token has a *query* vector Q_i which, when dot-multiplied against each input token's *key* vector K_j, determines the output's attention pattern A_i:

$$A = \text{softmax}(Q^T K).$$

Side note: for simplicity, I provide dot-product attention, omitting the dimensional division term $\frac{1}{\sqrt{d_k}}$ used in Attention is All You Need [1].

In a self-attention head, these keys and queries are determined by linear transformations of the residual stream / input embeddings E:

$$Q = W_Q E,$$
$$K = W_K E.$$

Combining equations

In total, substituting our equations for K and Q, we have:

$$A = \text{softmax}((W_Q E)^T W_K E),$$
$$A = \text{softmax}(E^T W_Q^T W_K E).$$

And now substituting H, V, then A within O:

$$O = W_{out} V A,$$
$$O = W_{out} W_V E A,$$
$$O = W_{out} W_V E \text{softmax}(E^T W_Q^T W_K E).$$

- Attention heads move information from a subspace of the residual stream of one token to a different subspace in the residual stream of another. Which matrix controls the subspace that gets read, and which matrix controls the subspace written to? What does their product mean?

  The OV circuit: The subspace that gets read from is determined by W_V, while the space that gets written to is determined by W_{out}. Together, $W_{out}W_V$ forms the "OV circuit", determining the effect of attending to a particular token.

  In a self-attending layer (in which the eigenvalues of the QK circuit are very positive), we can expect this circuit to represent bigram-ish statistics.

- Which tokens an attention head attends to is controlled by only two of the four matrices that define an attention head. Which two matrices are these?

  The QK circuit: W_Q and W_K determine attention. Together, these form the "QK circuit" $W_Q^T W_K$, determining which source tokens each destination token attends to.

- Attention heads can be written in terms of two matrices instead of four, $W_Q^T \cdot W_K$ and $W_{out} \cdot W_V$. In the previous two questions, you gave interpretations to these matrices. Now write out an attention head with only reference to them.

  (Already answered)

  - What is the rank of these matrices?

  Although the number of columns (and rows) in each matrix is equal to the token embedding length d_{embed}, their rank should be—at most—the significantly lower d_{head}, i.e. the length of the inner embedding in each head (typically 64 or 128). This is by the property that the rank of a matrix product cannot exceed the rank of any of its factors.

  Formally:

  $$W_Q^T W_K \leq d_{head},$$
  $$W_{out} W_V \leq d_{head}.$$

- You'd like to understand whether an attention head is reading in the output of a previous attention head. What does $W_V^2 \cdot W_{out}^1$ tell you about this? What do the singular values tell you?

  $W_V^2 W_{out}^1$ can be thought of as a linear transformation that determines the contribution of the previous layer's attention-output H^1 to the next layer's values V^2, given that $V^2 = W_V^2(W_{out}^1 H^1 + \text{rest of residual stream})$.

  For example, when $W_V^2 W_{out}^1$ is zero, we know that head 2 is not reading from head 1. The singular values (diagonal of S in $USV = W_V^2 W_{out}^1$) provide more detailed information:

  - The number of **non-zero singular values** tells us the dimensionality of the subspace through which information is being passed from head 1 to head 2.

– The **magnitude** of the singular values indicates the strength of this information transfer. Larger singular values suggest stronger connections between the heads.

Exercise 1 - Building a Simple Virtual Attention Head

Small transformers often have multiple attention heads which look at the previous token, but no attention heads which look at the token two previous. In this exercise, we'll see how two previous token heads can implement a small "virtual attention head" looking two tokens behind, without sacrificing a full attention head to the purpose.

Let's consider two attention heads, head 1 and head 2, which both attend to the previous token. Head 1 is in the first layer, head 2 is in the second layer. To make it easy to write out explicit matrices, we'll have the k, q, and v vectors of both heads be 4 dimensions and the residual stream be 16 dimensions.

(a) Write down W_V^1 and W_{out}^1 for head 1, such that the head copies dimensions 0-3 of its input to 8-11 in its output.

$$W_V^1 = \left[\begin{array}{cccccccccccccccc|c} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_1 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_2 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_3 \\ \hline c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & \end{array}\right]$$

$$W_{out}^1 = \left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & r_0 \\ 0 & 0 & 0 & 0 & r_1 \\ 0 & 0 & 0 & 0 & r_2 \\ 0 & 0 & 0 & 0 & r_3 \\ 0 & 0 & 0 & 0 & r_4 \\ 0 & 0 & 0 & 0 & r_6 \\ 0 & 0 & 0 & 0 & r_7 \\ \mathbf{1} & 0 & 0 & 0 & r_8 \\ 0 & \mathbf{1} & 0 & 0 & r_9 \\ 0 & 0 & \mathbf{1} & 0 & r_{10} \\ 0 & 0 & 0 & \mathbf{1} & r_{11} \\ 0 & 0 & 0 & 0 & r_{12} \\ 0 & 0 & 0 & 0 & r_{13} \\ 0 & 0 & 0 & 0 & r_{14} \\ 0 & 0 & 0 & 0 & r_{15} \\ \hline c_0 & c_1 & c_2 & c_3 & \end{array}\right]$$

(b) Write down W_V^2 and W_{out}^2 for head 2, such that it copies 3 more dimensions of the previous token, and one dimension from two tokens ago (using a dimension written to by the previous head).

$$W_V^2 = \left[\begin{array}{cccccccccccccccc|c}
0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_1 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_3 \\
\hline
c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15} &
\end{array}\right]$$

$$W_{out}^2 = \left[\begin{array}{cccc|c}
0 & 0 & 0 & 0 & r_0 \\
0 & 0 & 0 & 0 & r_1 \\
0 & 0 & 0 & 0 & r_2 \\
0 & 0 & 0 & 0 & r_3 \\
0 & 0 & 0 & 0 & r_4 \\
0 & 0 & 0 & 0 & r_6 \\
0 & 0 & 0 & 0 & r_7 \\
0 & 0 & 0 & 0 & r_8 \\
0 & 0 & 0 & 0 & r_9 \\
0 & 0 & 0 & 0 & r_{10} \\
0 & 0 & 0 & 0 & r_{11} \\
\mathbf{1} & 0 & 0 & 0 & r_{12} \\
0 & \mathbf{1} & 0 & 0 & r_{13} \\
0 & 0 & \mathbf{1} & 0 & r_{14} \\
0 & 0 & 0 & \mathbf{1} & r_{15} \\
\hline
c_0 & c_1 & c_2 & c_3 &
\end{array}\right]$$

(c) Expand out $W_{net}^1 = W_{out}^1 \cdot W_V^1$ and $W_{net}^2 = W_{out}^2 \cdot W_V^2$. What do these matrices tell you?

$$W_{net}^1 = \left[\begin{array}{cccccccccccccccc|c}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_7 \\
\mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_8 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_9 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{10} \\
0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{11} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{12} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{13} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{14} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{15} \\
\hline
c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15} &
\end{array}\right]$$

$$W^2_{net} = \begin{array}{c|cccccccccccccccc|c} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_0 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_1 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_2 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_3 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_4 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_5 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_6 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_7 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_8 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_9 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{10} \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{11} \\ & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{12} \\ & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{13} \\ & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{14} \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{15} \\ \hline & c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & \end{array}$$

In each *net* matrix, we can think of each column as representing one position from the input vectors, and each row as representing one position from the output vectors. Thus $W_{net}(i, j)$ indicates the amount that subspace i contributes to subspace j.

In W^1_{net}'s case, these input vectors are presumably the original token embeddings, while W^2_{net}'s input vectors would be the sum of the original embeddings and head 1's output.

(d) Expand out the following matrices: Two token copy: $W^2_{net} \cdot W^1_{net}$. One token copy: $W^2_{net} \cdot \mathrm{Id} + \mathrm{Id} \cdot W^1_{net}$.

$$W^2_{net} \cdot W^1_{net} =$$

$$
\left[\begin{array}{cccccccccccccccc|c}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_9 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{10} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{11} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{12} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{13} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{14} \\
\mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{15} \\
\hline
c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15} &
\end{array}\right]
$$

$$
W^2_{net} \cdot \mathrm{Id} + \mathrm{Id} \cdot W^1_{net} =
$$

$$
\left[\begin{array}{cccccccccccccccc|c}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_7 \\
\mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_8 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_9 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{10} \\
0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{11} \\
0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{12} \\
0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{13} \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{14} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{15} \\
\hline
c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15} &
\end{array}\right]
$$

It's interesting to note that cell (r_{15}, c_8) has the dual function of copying from head 1's output at position 8 (i.e. from position 0 from two tokens ago) as well as the input embeddings at position 8 (i.e. from position 8 from previous token).

- Observation: When we think of an attention head normally, they need to dedicate all their capacity to one task. In this case, the two heads dedicated 7/8ths of their capacity to one task and 1/8th to another.

Exercise 2 - Copying Text with an Induction Head (Pointer Arithmetic Version)

The simplest kind of in-context meta-learning that neural networks do is increasing the probability of sequences they've seen before in this context. This is done with an "induction head" that looks at what followed after last time we saw a token.

There are at least two algorithms for implementing induction heads. In this exercise, you'll build up the "pointer arithmetic" algorithm by hand.

(a) Let $u_0^{cont}, u_1^{cont}, \ldots, u_n^{cont}$ be the principal components of the content embedding. Write W_Q and W_K for an attention head (with 4 dimensional queries and keys) selecting tokens with similar content to the present token, including the present token itself.

$$W_Q = W_K = \begin{bmatrix} u_0^{cont} \\ u_1^{cont} \\ u_2^{cont} \\ u_3^{cont} \end{bmatrix}$$

(Intuition: each row measures each token's alignment with a principle component; and we set $W_K = W_Q$ to maximize $A = \text{softmax}(E^T W_Q^T W_K E)$.)

(b) Let $u_0^{\cos}, u_0^{\sin}, u_1^{\cos}, u_1^{\sin}, \ldots$ be a basis describing the position embedding in terms of vectors that code for the sine and cosine embedding of the token position (e.g., $\lambda(\cos(\alpha_0 n_{tok}))$ with descending magnitudes. Write W_Q and W_K for an attention head (with 4 dimensional queries and keys) that self-selects the present token position.

$$W_Q = W_K = \begin{bmatrix} u_0^{\cos} \\ u_0^{\sin} \\ u_1^{\cos} \\ u_1^{\sin} \end{bmatrix}$$

(Intuition: each row produces a predictable sin/cos value, e.g. $\lambda(\cos(\alpha_0 n_{tok}))$, when multiplied against token at position n_{tok}. Descending magnitudes are used for finer positional discrimination.)

(c) Using the position embedding basis described in (b), write W_Q and W_K for an attention head (with 4 dimensional queries and keys) that self-selects the *previous* token position. Hint: think about a 2D rotation matrix.

$$W_K = \begin{bmatrix} u_0^{\cos} \\ u_0^{\sin} \\ u_1^{\cos} \\ u_1^{\sin} \end{bmatrix}$$

(Intuition: each token advertises its current position...)

We now wish to produce queries with values of a form like $\lambda(\cos(\alpha_0(n_{tok}-1)))$ for token number n_{tok}. Formally, where $W_Q^{2b}(0,k) = u_0^{\cos}(k)$:

$$\begin{aligned} &\sum_k W_Q^{2b}(0,k)\cdot V(k,j) = \lambda(\cos(\alpha_0 n_{tok})) \\ \implies &\sum_k W_Q(0,k)\cdot V(k,j) = \lambda(\cos(\alpha_0(n_{tok}-1))). \end{aligned}$$

Applying $\cos(a-b) = \cos(a)\cos(b) + \sin(a)\sin(b)$:

$$\begin{aligned} \lambda(\cos(\alpha_0(n_{tok}))) &= \lambda(\cos(\alpha_0 n_{tok} - \alpha_0)) \\ &= \lambda(\cos(\alpha_0(n_{tok})))(\cos\alpha_0) + \lambda(\sin(\alpha_0(n_{tok})))(\sin\alpha_0) \\ &= \sum_k u_0^{\cos}(k)(\cos\alpha_0)\cdot V(k,j) + \sum_k u_0^{\sin}(k)(\sin\alpha_0)\cdot V(k,j) \\ &= \sum_k (u_0^{\cos}(k)(\cos\alpha_0) + u_0^{\sin}(k)(\sin\alpha_0))\cdot V(k,j). \end{aligned}$$

Hence $W_Q(0,k) = u_0^{\cos}(k)(\cos\alpha_0) + u_0^{\sin}(k)(\sin\alpha_0)$.

Following the same pattern, by $W_Q^{2b}(1,k) = u_0^{\sin}(k)$ and $\sin(a-b) = \sin(a)\cos(b) - \cos(a)\sin(b)$, it follows that $W_Q(1,k) = u_0^{\sin}(k)(\cos\alpha_0) - u_0^{\cos}(k)(\sin\alpha_0)$.

Therefore:

$$W_Q = \begin{bmatrix} u_0^{\cos}(\cos\alpha_0) + u_0^{\sin}(\sin\alpha_0) \\ u_0^{\sin}(\cos\alpha_0) - u_0^{\cos}(\sin\alpha_0) \\ u_1^{\cos}(\cos\alpha_0) + u_1^{\sin}(\sin\alpha_0) \\ u_1^{\sin}(\cos\alpha_0) - u_1^{\cos}(\sin\alpha_0) \end{bmatrix}.$$

As each row is a linear combination of rows from W_Q^{2b}, we can express it as a rotation:

$$W_Q = \begin{bmatrix} \cos\alpha_0 & \sin\alpha_0 & 0 & 0 \\ -\sin\alpha_0 & \cos\alpha_0 & 0 & 0 \\ 0 & 0 & \cos\alpha_1 & \sin\alpha_1 \\ 0 & 0 & -\sin\alpha_1 & \cos\alpha_1 \end{bmatrix} \begin{bmatrix} u_0^{\cos} \\ u_0^{\sin} \\ u_1^{\cos} \\ u_1^{\sin} \end{bmatrix}.$$

(d) Write W_Q and W_K for an attention head (with 8 dimensional queries and keys) selecting tokens with similar content to the present token, but disprefers attending to itself. Hint: refer to (b) and use extra 4 dimensions for keys and queries.

My approach here will be to produce queries that negatively attend the self-attending positional weights (2b), but positively the self-attending content weights (2a), by storing both

within the new 8-dimension vectors.

$$W_K = \begin{bmatrix} u_0^{cont} \\ u_1^{cont} \\ u_2^{cont} \\ u_3^{cont} \\ u_0^{\cos} \\ u_0^{\sin} \\ u_1^{\cos} \\ u_1^{\sin} \end{bmatrix}, \quad W_Q = \begin{bmatrix} u_0^{cont} \\ u_1^{cont} \\ u_2^{cont} \\ u_3^{cont} \\ -u_0^{\cos} \\ -u_0^{\sin} \\ -u_1^{\cos} \\ -u_1^{\sin} \end{bmatrix}.$$

(e) Write down W_V and W_{out} for the attention head you described in (d), such that it extracts the largest 8 dimensions of the position embedding from the token it attends to, and writes them to the vectors $v_0, v_1, \ldots$.

$$W_V = \begin{bmatrix} u_0^{\cos} \\ u_0^{\sin} \\ u_1^{\cos} \\ u_1^{\sin} \\ u_2^{\cos} \\ u_2^{\sin} \\ u_3^{\cos} \\ u_3^{\sin} \end{bmatrix}, \quad W_{out} = \begin{bmatrix} v_0^T & v_1^T & v_2^T & v_3^T & v_4^T & v_5^T & v_6^T & v_7^T \end{bmatrix}.$$

(Intuition: each row of W_V extracts a positional feature; each row of W_{out} selects embedding dimensions to write to the residual stream.)

(f) Write W_Q and W_K for an attention head which attends to the token after a previous copy of the present token. Hint: use the outputs of the head from (e) and the strategy you used in (c).

This head will need to perform the following mechanisms:

- Query: extracting the previous-copy positional features from the residual stream by following v.
- Query: transforming those positions to select the *next* token, using a similar rotation to part (c).
- Key: advertising current-token positional features, similar to part (b).

Thus, flipping the rotation by taking $-\alpha$:

$$W_Q = \begin{bmatrix} \cos(-\alpha_0) & \sin(-\alpha_0) & 0 & 0 \\ -\sin(-\alpha_0) & \cos(-\alpha_0) & 0 & 0 \\ 0 & 0 & \cos(-\alpha_1) & \sin(-\alpha_1) \\ 0 & 0 & -\sin(-\alpha_1) & \cos(-\alpha_1) \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}, \quad W_K = \begin{bmatrix} u_0^{\cos} \\ u_0^{\sin} \\ u_1^{\cos} \\ u_1^{\sin} \end{bmatrix}.$$

This is under the assumption that u remains a relevant position-extract method after the

previous head's output to the dimensions specified in v... For example, if u targets dimensions that have been unaltered since the original embedding.

Exercise 3 - Copying Text with an Induction Head (Previous Token K-Composition Version)

Some positional encoding mechanisms, such as rotary attention, don't expose positional information to the W_V matmul. Transformers trained with these mechanisms can't use the strategy from (e) and (f) in the previous exercise to manipulate positional encoding vectors.

For these transformers, we've seen an alternate mechanism, where the first head copies information about the preceding token into a subspace, and the second head uses that subspace to construct queries and keys. Assuming the same positional encoding mechanism as above, write down $W_Q^1, W_K^1, W_V^1, W_O^1$ and W_Q^2 and W_K^2 for a pair of attention heads implementing this algorithm.

First, let's informally consider the purpose of each matrix:

- W_V^1—embed current token info (just content?)...
- W_Q^1—seek prev token position (using rotation trick?)
- W_K^1—advertise current token position
- W_O^1—put values into some subspace using v vectors
- W_Q^2—seek prev instances of current token content
  - but we don't have to use non-current-position trick from part d, because tokens will be advertising prev-token info, thus not advertising the latest token anyway...
- W_K^2—select from subspace using v vectors, thus advertising prev-token content

This gives us these weights for the first head:

$$W_V^1 = \begin{bmatrix} u_0^{cont} \\ u_1^{cont} \\ u_2^{cont} \\ u_3^{cont} \end{bmatrix}, \quad W_Q^1 = \begin{bmatrix} \cos\alpha_0 & \sin\alpha_0 & 0 & 0 \\ -\sin\alpha_0 & \cos\alpha_0 & 0 & 0 \\ 0 & 0 & \cos\alpha_1 & \sin\alpha_1 \\ 0 & 0 & -\sin\alpha_1 & \cos\alpha_1 \end{bmatrix} \begin{bmatrix} u_0^{\cos} \\ u_0^{\sin} \\ u_1^{\cos} \\ u_1^{\sin} \end{bmatrix},$$

$$W_K^1 = \begin{bmatrix} u_0^{\cos} \\ u_0^{\sin} \\ u_1^{\cos} \\ u_1^{\sin} \end{bmatrix}, \quad W_{out}^2 = \begin{bmatrix} v_0^T & v_1^T & v_2^T & v_3^T & v_4^T & v_5^T & v_6^T & v_7^T \end{bmatrix}.$$

And the second:

$$W_Q^2 = \begin{bmatrix} u_0^{cont} \\ u_1^{cont} \\ u_2^{cont} \\ u_3^{cont} \end{bmatrix}, \quad W_K^2 = \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}.$$

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.