Attention block in deep neural networks

3 main vectors that are used in attention

Query vector (Q): Represents the element (the word) we are interested in at the current step. It acts like a question asking, "which other words are important for this one?"

Key vector (K): Represents all other elements in the sequence. Each element has its own key, indicating how relevant it might be in relation to the query

Value Vector (V): Represents the actual information of the sequence element, which the model will use in its computations after the attention weights are applied

Step 1 Compute Relevance (Attention Scores):

Each query is compared with all the keys in the sequence. The comparison between query (Q) and key (K) is typically done using the dot product, followed by scaling and applying a softmax function to get the attention scores. These scores determine how much focus (attention) each word should receive

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_K}}\right)$$

The term QK^T gives us an attention score

- dot product between query Q and each key K.
- the dot product between Q and K measures the similarity between the two vectors
- if Q and K are pointing in the same direction (they are similar), the dot product will be large.
- if not aligned, the dot product will be much smaller (they are dissimilar

Division by $\sqrt{d_K}$ is a scaling factor used to prevent the dot product from growing too large

- the dot product QK^T can grow large when the dimensionality of the vectors (d_K) increases. This is because the dot product involves summing over multiple dimensions, and with large dimensional vectors, the result can become quite large, leading to very high values in the attention score
- what is the problem with large scores? If the values in the dot product are too large, they will be exaggerated when passed through the softmax function. This can lead to extremely sharp attention distributions (one element gets all the attention, while the others get close to zero), making the model less effective
- by dividing by the $\sqrt{d_K}$, we reduce the magnitude of the dot product so that the values stay in a more reasonable range before applying softmax. This ensures that the attention mechanism doesn't become too "peaky" or too concentrated on a single element in the sequence

The softmax transforms the attention scores into a probability distribution, where each score represents how much attention should be paid to the corresponding element

Step 2 Weighted sum of values V

After applying the softmax function to the scaled dot product $QK^T/\sqrt{d_K}$, we have a set of attention weights (probabilities) for each element in the sequence

- Now that the model knows how much attention to pay to each element, it needs to extract the actual content or meaningful information from the sequence, and this is where the value vectors V come in
- The attention scores (which are based on Q and K) only tell us where to focus, but the value vectors contain the what - the actual information that needs to be processed

$$\text{Attention output} = \sum(\text{Attention score} \times V)$$

- the weighted sum results a context sensitive representation of the input, where important elements contribute more to the final output
- by applying the attention scores to the value vectors, the model can create a representation that captures long-range dependencies and relationships in the data
- this allows the model to generate context-aware outputs, capturing relationships and dependencies in the data based on both where to focus (attention scores) and what information to use (value vectors)

In order to get your Query (Q), Key (K), and value vectors (V), mathematically you can do

$$Q = W_Q \cdot X$$

$$K = W_K \cdot X$$

$$V = W_V \cdot X$$

- W_Q, W_K, and W_V are learned weight matrices (parameters of the model) that map the input embeddings into query, key and value spaces
- X is the is the input embedding (word embedding + positional encoding)

- the value vector V represents the actual semantic content of the element in the input sequence

Multi-head attention

- in multi-head attention, the model uses multiple sets of weight matrices W_Q, W_K, and W_V to project the input into multiple query, key and value spaces. Each head computes its own value vector and attention scores, and the output from all heads are concatenated and combined for the final result