

Attention Mechanisms of a Transformer:

Self-Attention is a type of attention used in Encoder-Decoder and Encoder-Only transformers. It allows every word in a phrase to define a relationship with any other word in the phrase, regardless of the order of the words. In other words, if the phrase is The pizza came out of the oven and it tasted good!, then the word it can define its relationship with every word in that phrase, including words that came after it, like tasted and good



Masked Self-Attention is used by Encoder-Decoder and Decoder-Only transformers and it allows each word in a phrase to define a relationship with itself and the words that came before it. In other words, Masked Self-Attention prevents the transformer from "looking ahead". This is illustrated below where the word it can define relationships with itself and everything that came earlier in the input. In Encoder-Decoder transformers, Masked Self-Attention is used during training, when we know what the output should be, but we still force the decoder to generate it one token at a time, thus, limiting attention to only output words that came earlier.



Now that we have a general sense of the three types of attention used in transformers, we can talk about how it's calculated.

First, the general equations for the different types of attention are almost identical as seen in the figure below. In the equations, Q is for the Query matrix, K is for the Key matrix and V is for the Value matrix. On the left, we have the equation for Self-Attention and Encoder-Decoder Attention.

The differences in these types of attention are not from the equation we use, but from how the Q, K, and V matrices are computed. On the right, we see the equation for Masked Self-Attention and the only difference it has from the equation on the left is the addition of a Mask matrix, M, that prevents words that come after a specific Query from being included in the final attention scores.

$$Attention(Q, K, V) = SoftMax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad MaskedAttention(Q, K, V) = SoftMax\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V$$

First, given word embedding values for each word/token in the input phrase SOS let's go in matrix form, we multiply them by matrices of weights to create Queries, Keys, and Values

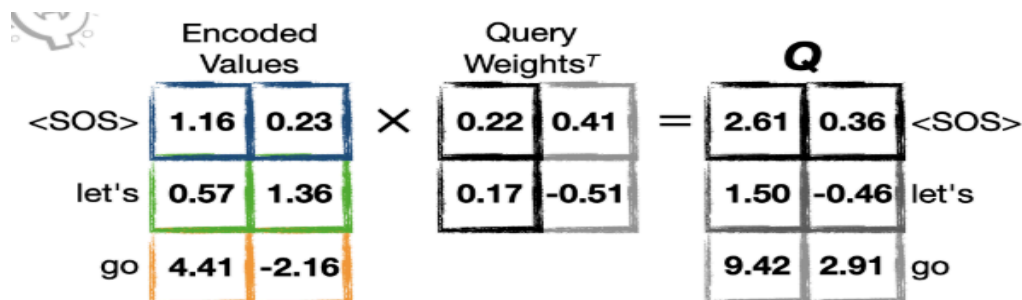


Diagram illustrating the calculation of the Query matrix (Q) by multiplying Encoded Values by Query Weights.

| | Encoded Values | | | Query Weights ^T | | | Q | | |
|-------|----------------|-------|---|----------------------------|-------|---|------|-------|-------|
| <SOS> | 1.16 | 0.23 | × | 0.22 | 0.41 | = | 2.61 | 0.36 | <SOS> |
| let's | 0.57 | 1.36 | | 0.17 | -0.51 | | 1.50 | -0.46 | let's |
| go | 4.41 | -2.16 | | | | | 9.42 | 2.91 | go |

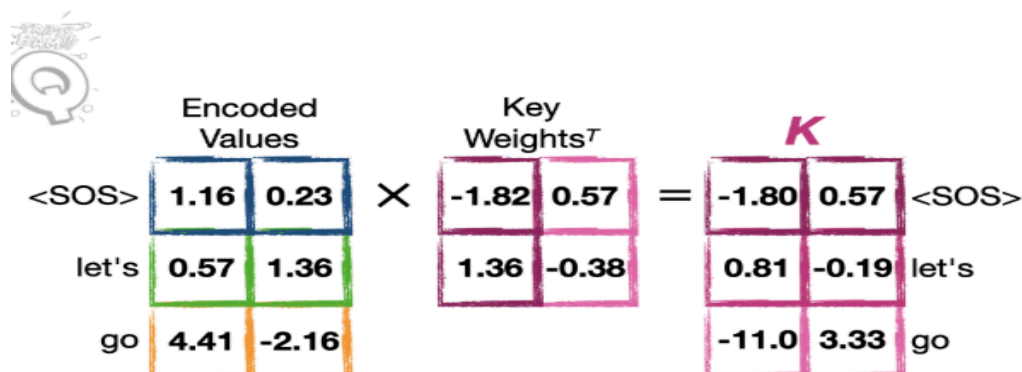
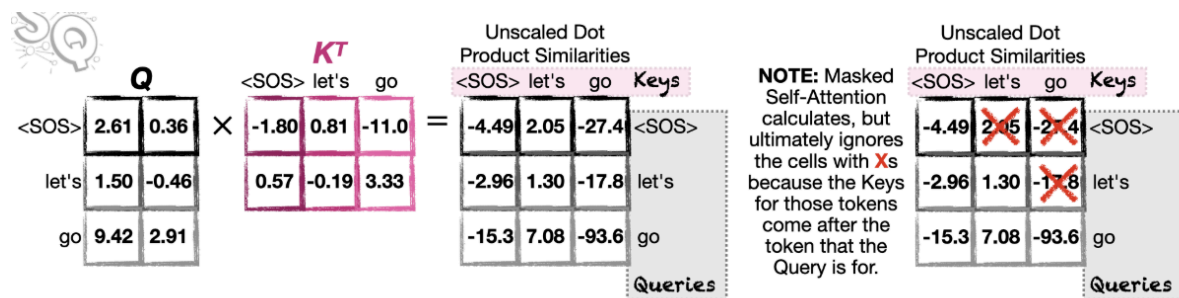


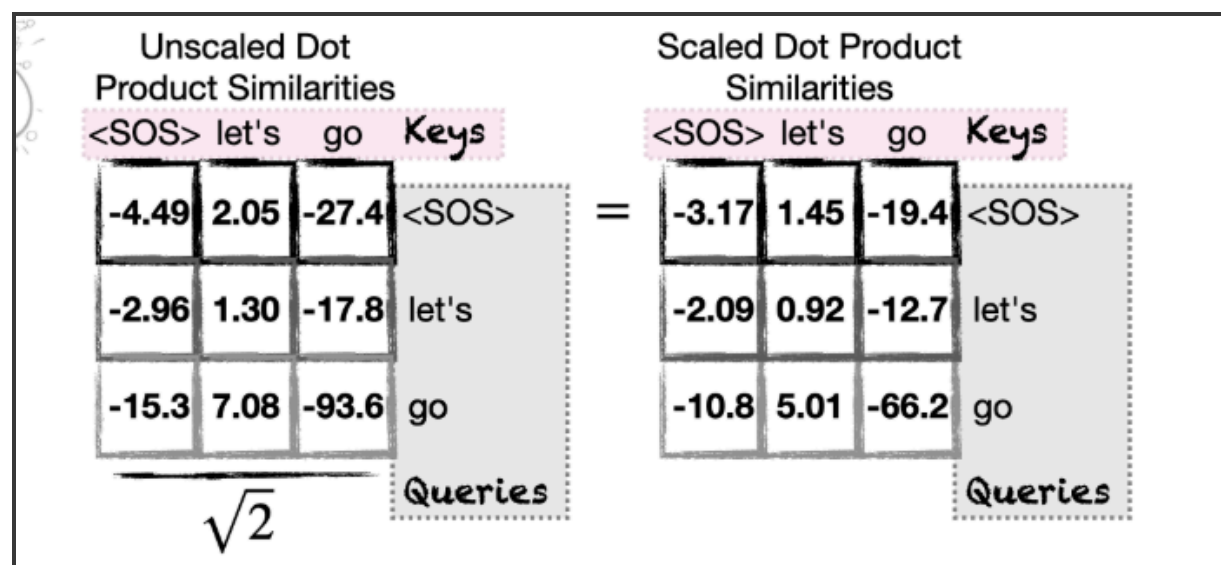
Diagram illustrating the calculation of the Key matrix (K) by multiplying Encoded Values by Key Weights.

| | Encoded Values | | | Key Weights ^T | | | K | | |
|-------|----------------|-------|---|--------------------------|-------|---|-------|-------|-------|
| <SOS> | 1.16 | 0.23 | × | -1.82 | 0.57 | = | -1.80 | 0.57 | <SOS> |
| let's | 0.57 | 1.36 | | 1.36 | -0.38 | | 0.81 | -0.19 | let's |
| go | 4.41 | -2.16 | | | | | -11.0 | 3.33 | go |

We then multiply the Queries by the transpose of the Keys so that the query for each word calculates a similarity value with the keys for all of the words. NOTE: As seen in the illustration below, Masked Self-Attention calculates the values for all Query/Key pairs, but, ultimately, ignores values for when a token's Query comes before other token's Keys. For example, if the Query is for the first token SOS, then Masked Self-Attention will ignore the values calculated with Keys for Let's and go because those tokens come after SOS.



The next step is to scale the similarity scores by the square root of the number of columns in the Key matrix, which represents the number of values used to represent each token. In this case, we scale by the square root of 2



Now, if we were doing Masked Self-Attention, we would mask out the values we want to ignore by adding -infinity to them, as seen below. This step is the only difference between Self-Attention and Masked Self-Attention.



The next step is to apply the $\text{SoftMax}()$ function to each row in the scaled similarities. We'll do this first for the Self-Attention without a mask (below).

The $\text{SoftMax}()$ function gives us percentages that the Values for each token should contribute to the attention score for a specific token. Thus, we can get the final attention scores by multiplying the percentages with the Values in matrix V. First, we'll do this with the unmasked percentages.

