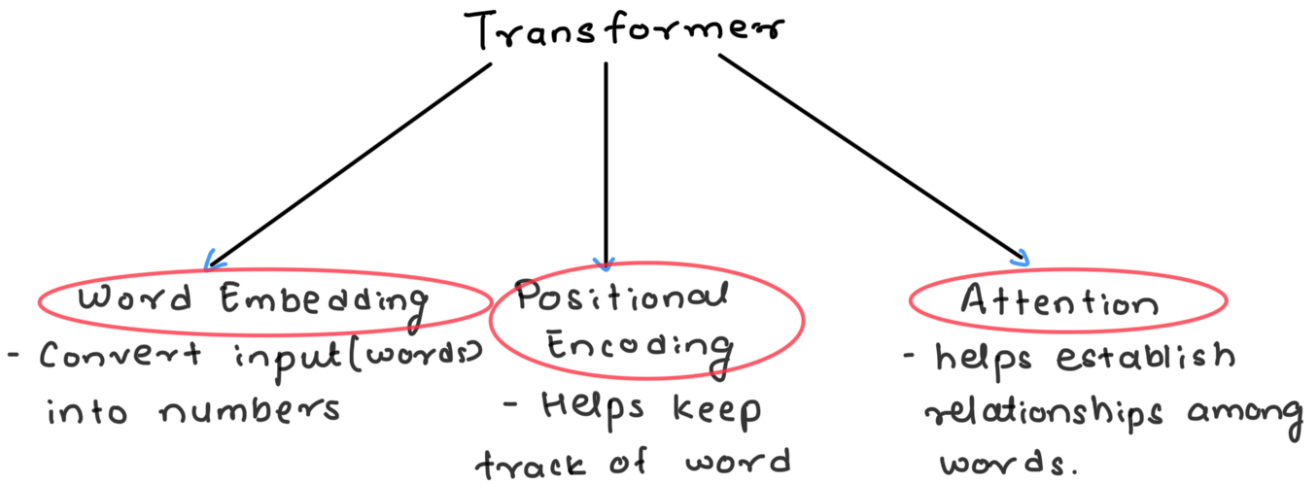


* Matrix math for Attention:

- The eqn for calculating self-attention:

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

- Here, Q = Query , K = key , V = Value.

- Self attention calculates the similarities between each word and itself and all of the other words. and calculates this similarities for every word in sentence.

- For the explanation of above sentence in terms of Query , key and Value.

e.g. Write a poem.

Query	key	Value. (similarity)
Write Write Write	Write a poem	1 0.46 0.2

Imaginary values

Above table can be drawn, for word "a" and word "poem".

- Example of sentence in terms of matrix multiplication:

Sentence — Write a poem.

Step I : Do word embedding and positional encoding of each word in sentence.

This simply means we are going to represent each word with 2 numbers. (Note: It's common to use 512 numbers to represent words)

Words	Numbers
Write	1.16 0.23
a	0.57 1.36
poem	4.41 -2.16

} Simply consider this numbers, they are random.

Encoded matrix:

$$\begin{bmatrix} 1.16 & 0.23 \\ 0.57 & 1.36 \\ 4.41 & -2.16 \end{bmatrix} \begin{matrix} \rightarrow \text{write} \\ \rightarrow \text{a} \\ \rightarrow \text{poem} \end{matrix}$$

Step 2 : Multiply Encoded matrix of sentence with weight matrices of Query, key and value.

(Note: We will use 2 x 2 weight matrices because we started with 2 encoded values per word, which will result in 2 Query / key / value per word.)

Encoded values × Query weights^T = Q

$$\begin{matrix} \\ \text{write} \\ \text{a} \\ \text{poem} \end{matrix} \begin{bmatrix} 1.16 & 0.23 \\ 0.57 & 1.36 \\ 4.41 & -2.16 \end{bmatrix} \times \begin{bmatrix} 0.54 & -0.17 \\ 0.59 & 0.65 \end{bmatrix} = \begin{bmatrix} 0.76 & -0.05 \\ 1.11 & 0.79 \\ 1.11 & -2.15 \end{bmatrix} \begin{matrix} \\ \text{write} \\ \text{a} \\ \text{poem} \end{matrix}$$

key weights^T = K

write	1.16	0.23	×	-0.15	-0.34	=	-0.14 -0.30	write
a	0.57	1.36		0.14	0.42		-0.1 0.38	a
poem	4.41	-2.16					-0.96 -2.41	poem

Value weights^T V

write	1.16	0.23	×	0.62	0.61	=	0.60 0.74	write
a	0.57	1.36		-0.52	0.13		-0.35 0.52	a
poem	4.41	-2.16					3.86 2.41	poem

we got the values of Q, K, V to put into the formula.

Note: If we have started with 512 word embeddings per word, we could have used 512 x 512 weight matrices which would resulted into 512 Query/value/key words.

Step 3: Calculate self-Attention from Q, K, V.

Transpose of K, so to calculate mat mul.

Unscaled Dot product similarities.

Q				K^T				
0.76	-0.05	×	-0.14	0.10	-0.96	=	-0.09	0.06 -0.61
1.11	0.79		-0.30	0.38	-2.41		-0.39	0.41 -2.97
1.11	-2.15						0.49	-0.71 4.12
3x2			2x3					

- Q . K^T is a unscaled dot product. Thus, by multiplying Q by the transpose of K we get the unscaled dot product similarities bet^w all possible combinations of Queries and keys for each word.

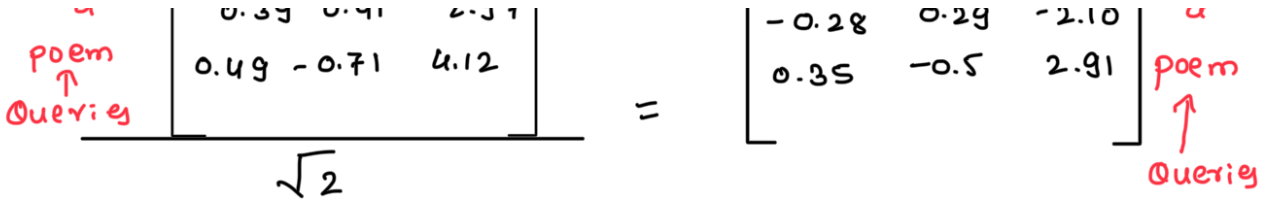
- d_k is dimension of key matrix. Dimensions refers to number of values we have for each token, which is 2.

Un Scaled Dot product similarities

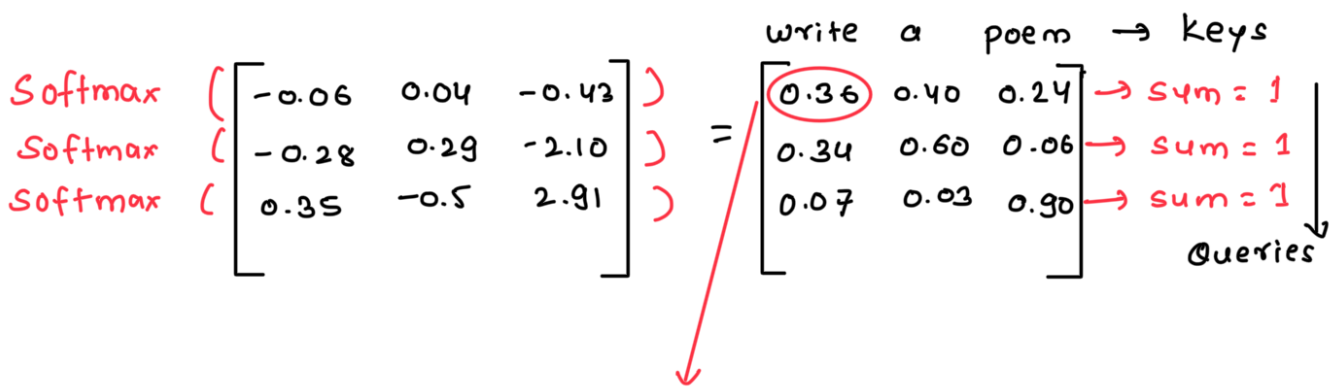
keys →	write	a	poem
write	-0.09	0.06	-0.61
a			

Scaled Dot product similarities

	write	a	poem ← keys
write	-0.06	0.04	-0.43

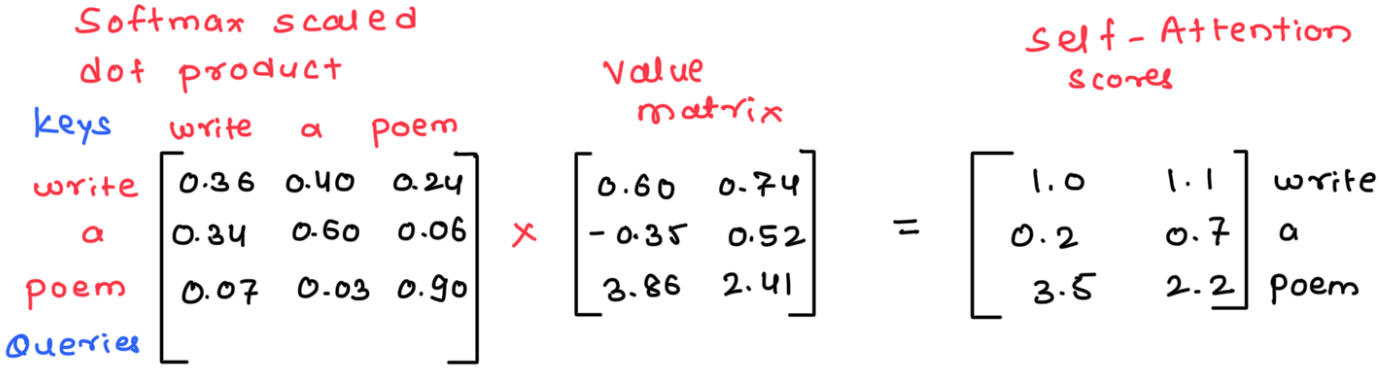


- We calculate softmax of each row, the softmax function makes each row sum 1.



The word "write" is 36% similar to itself.

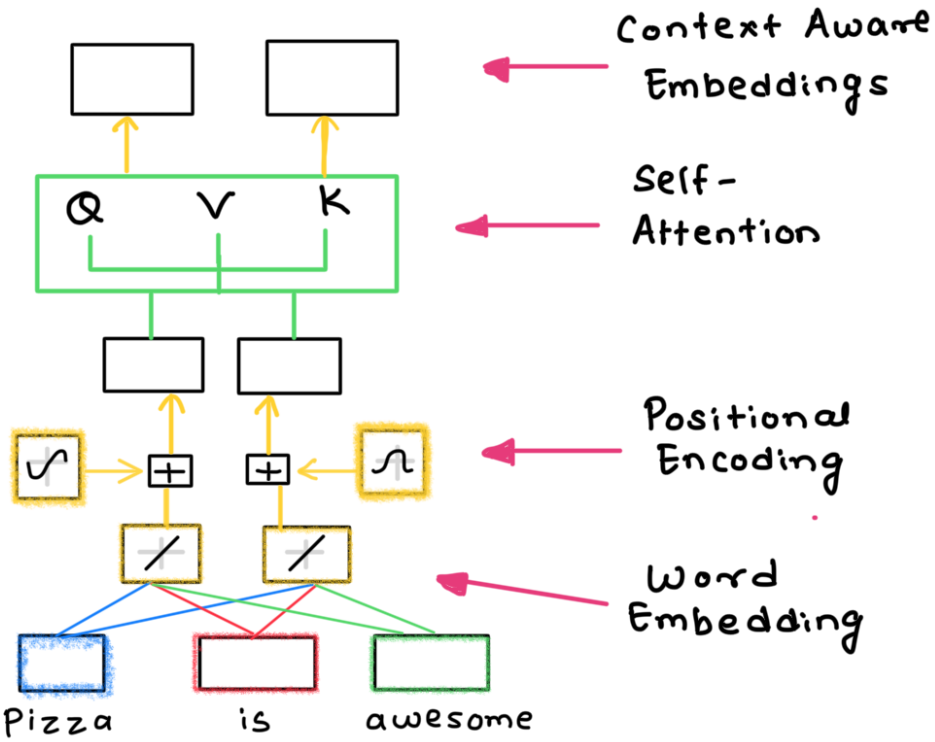
- We multiply the resulted matrix with value matrix to get the "Self-attention scores".



- The percentage came from softmax scaled matrix, tells us how much influence each word should have in final encoding. for any given word.

- In summary, the Self Attention does calculate the scaled dot product similarities among all of words, convert those similarities into percentages with the softmax function, and then use those percentage to scale the values to become the Self-Attention scores for each word.

* Encoder-Only Transformer:

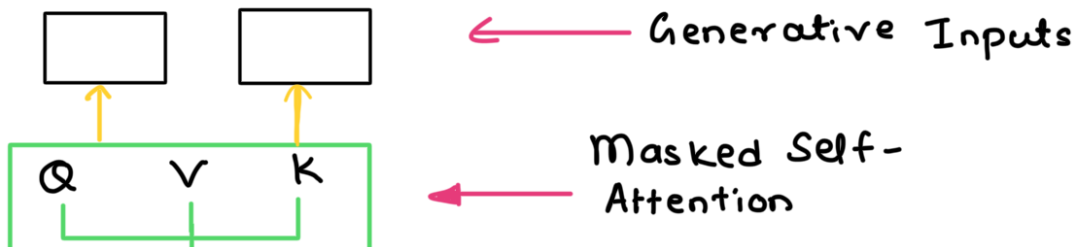


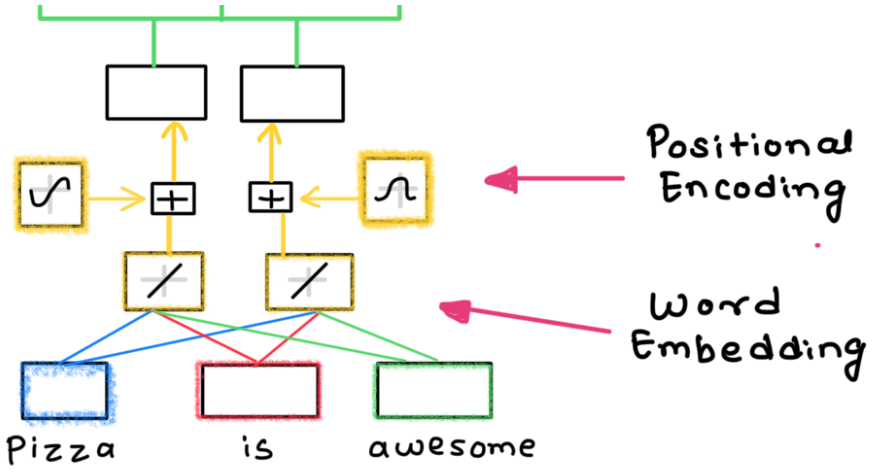
- Transformers which uses only self attention are called Encoder-Only Transformer.

- Context Aware Embeddings cluster similar sentences, Similar documents.

- we can use context aware embeddings as inputs to neural network that classifies the sentiment of input.

* Decoder-Only Transformer:

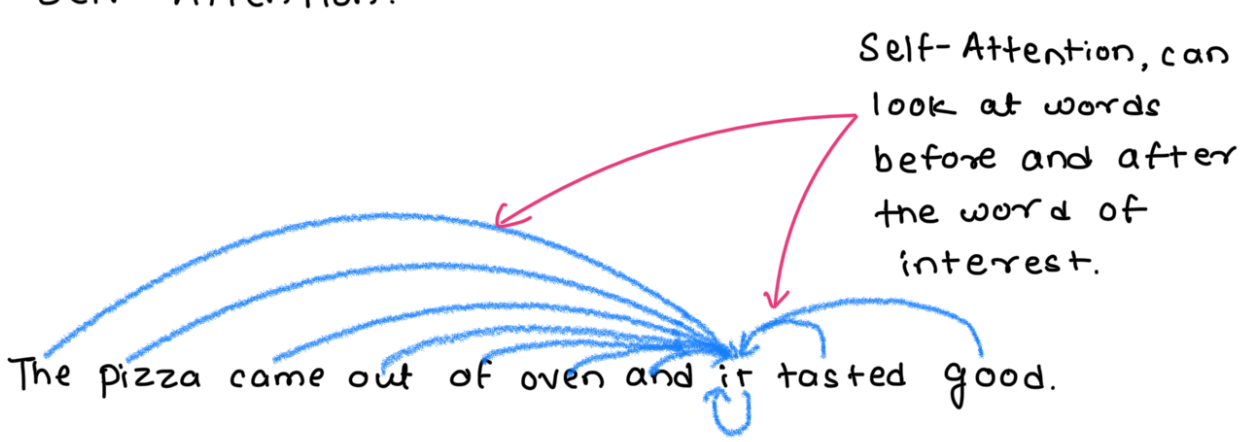




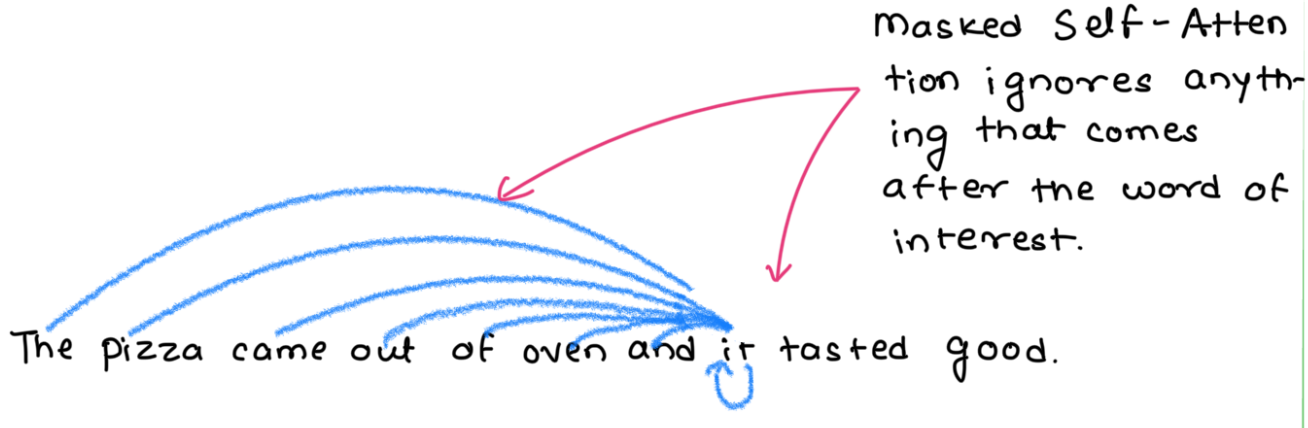
- Decoder only Transformer uses the Masked-Self-attention, which creates Generative Inputs.
- The generative Inputs can be plugged to neural network to generate next token.

* Difference Betw Self and Masked Attention:

Self-Attention:



Masked Self-Attention:



* Matrix Math for Masked Self-Attention:

- $\text{Masked Attention}(Q, K, V, M) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right) V$

- We add the Masked Matrix to scaled Dot product similarities.

Scaled Dot product similarities

(Adds 0s to values we want to include, -inf to mask out)

Masked Scaled Dot product simil.

write a poem ← keys

$$\begin{bmatrix} -0.06 & 0.04 & -0.43 \\ -0.28 & 0.29 & -2.10 \\ 0.35 & -0.5 & 2.91 \end{bmatrix} \begin{matrix} \text{write} \\ \text{a} \\ \text{poem} \\ \uparrow \\ \text{Query} \end{matrix} + \begin{bmatrix} 0 & -\text{inf} & -\text{inf} \\ 0 & 0 & -\text{inf} \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -0.06 & -\text{inf} & -\text{inf} \\ -0.28 & 0.29 & -\text{inf} \\ 0.35 & -0.5 & 2.91 \end{bmatrix}$$

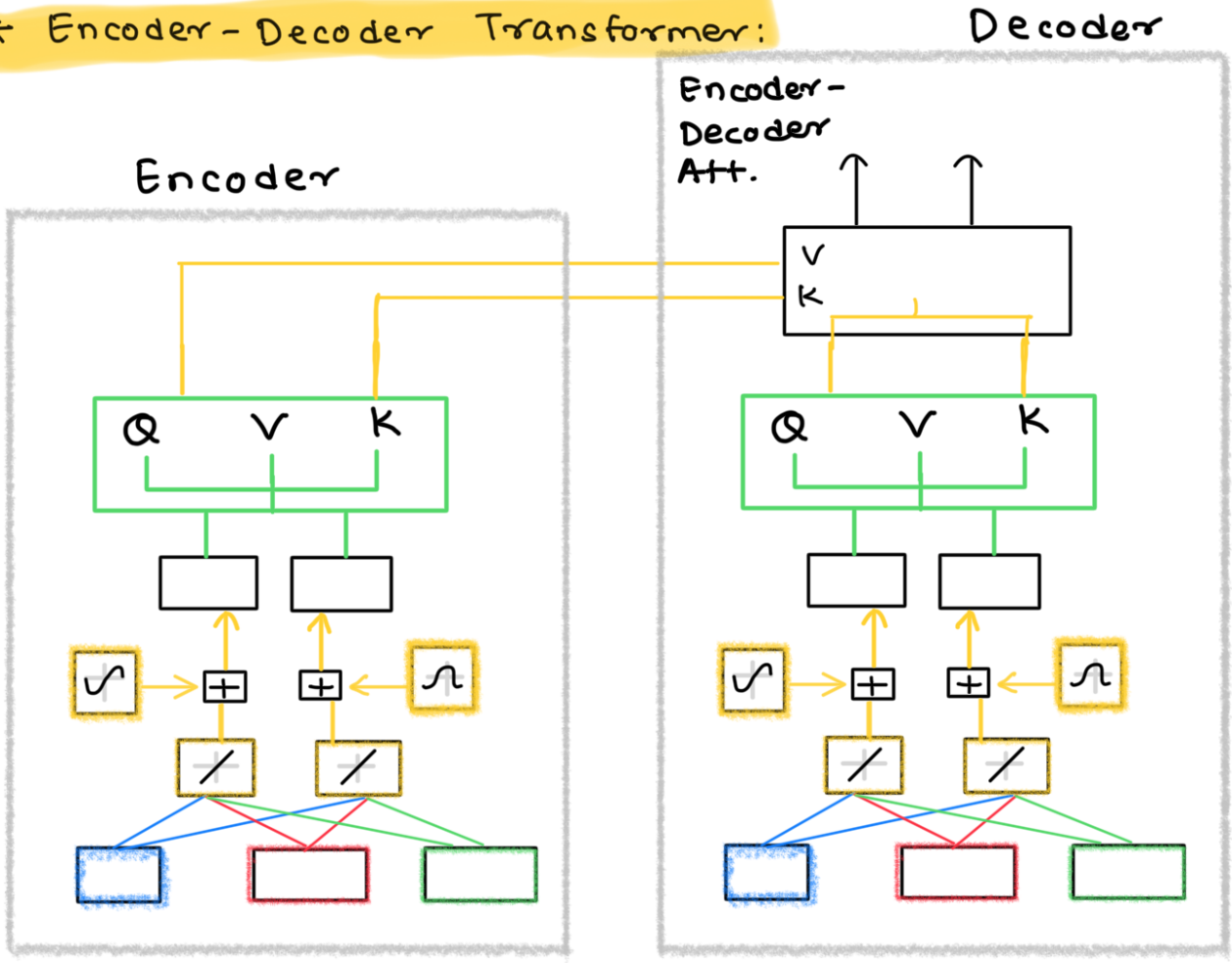
write ← a poem keys

$$\begin{matrix} \text{Softmax} \\ \text{Softmax} \\ \text{Softmax} \end{matrix} \left(\begin{bmatrix} -0.06 & -\text{inf} & -\text{inf} \\ -0.28 & 0.29 & -\text{inf} \\ 0.35 & -0.5 & 2.91 \end{bmatrix} \right) = \begin{bmatrix} 1.00 & 0.00 & 0.00 \\ 0.36 & 0.64 & 0.00 \\ 0.07 & 0.03 & 0.90 \end{bmatrix} \begin{matrix} \text{write} \\ \text{a} \\ \text{poem} \\ \text{Queries} \end{matrix}$$

- The resulted matrix means,
  - First token "write" has 100% similarity to itself.
  - Second token "a" has some similarity with first token, but no similarity with third token.
  - Third token has similarity with every token.

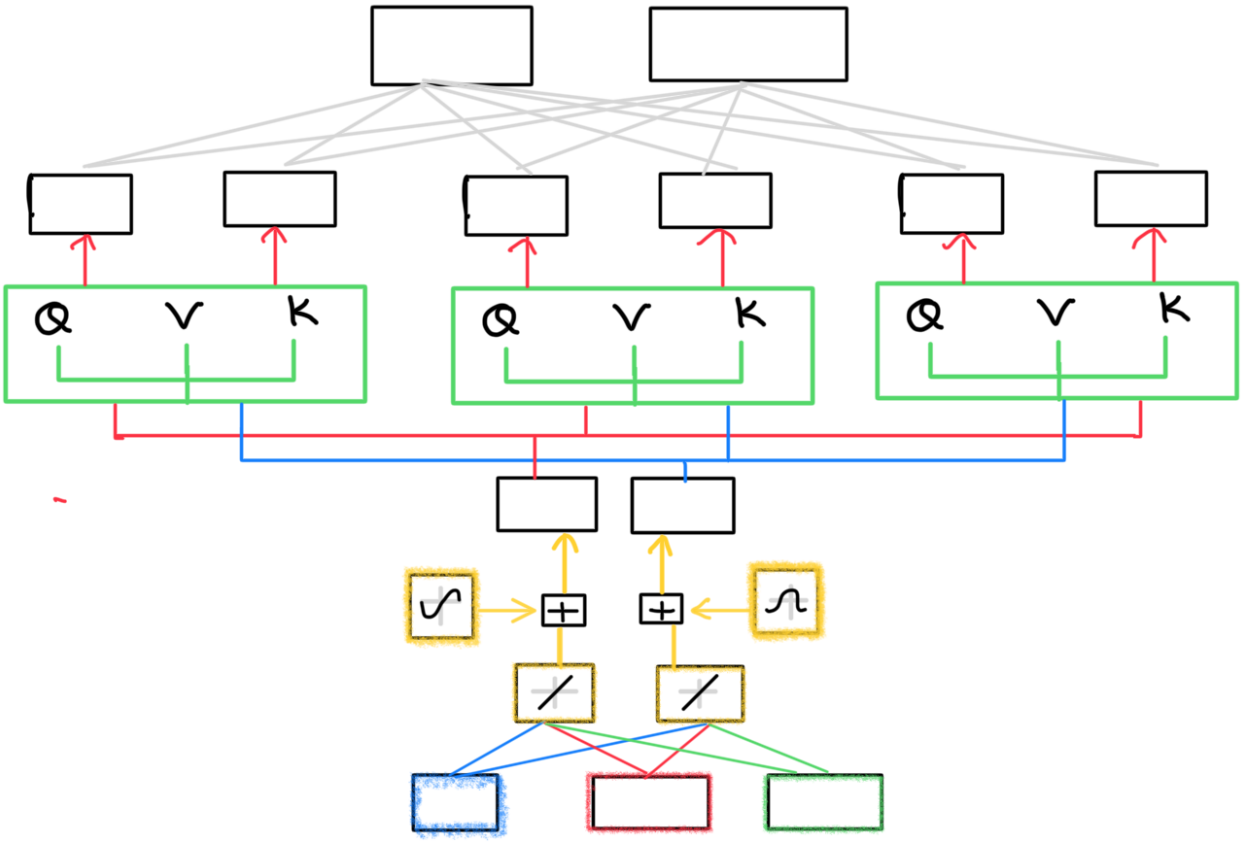
- After this, we calculate multiplication with value matrix and get the masked self-attention scores.

* Encoder - Decoder Transformer:



- Also called as Cross - Attenation Transformer.
- The first transformer was based on Seq 2 Seq, which was used to translation from one language to another.

* Multi-head Attention:



- We can apply Attention multiple times simultaneously.
- Each Attention is called Head and has its own sets of weights for calculating the Queries, key and values. When we have multiple heads calculating Attention, we call it Multi-head Attention.