Transformers

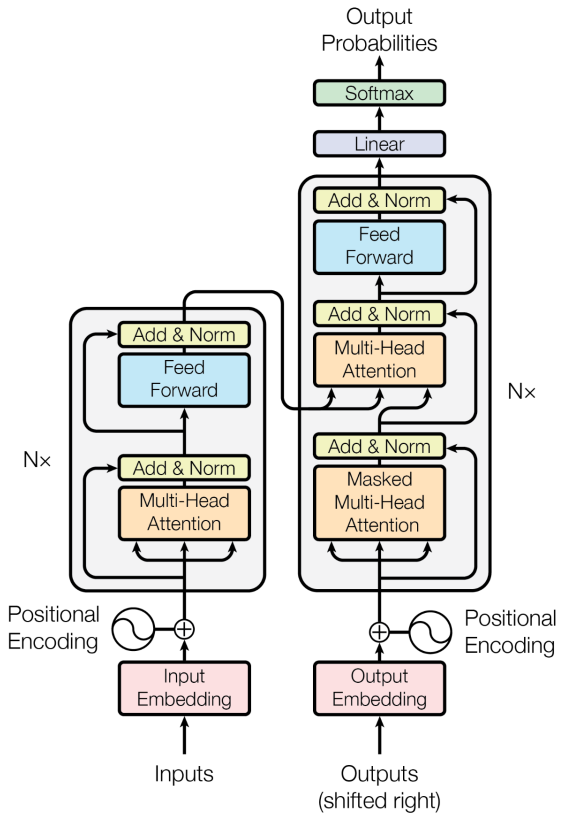
Alberto Valdés, Chile

Mathematical Engineer

Attention Is All You Need, June 2017

Proposed Architecture

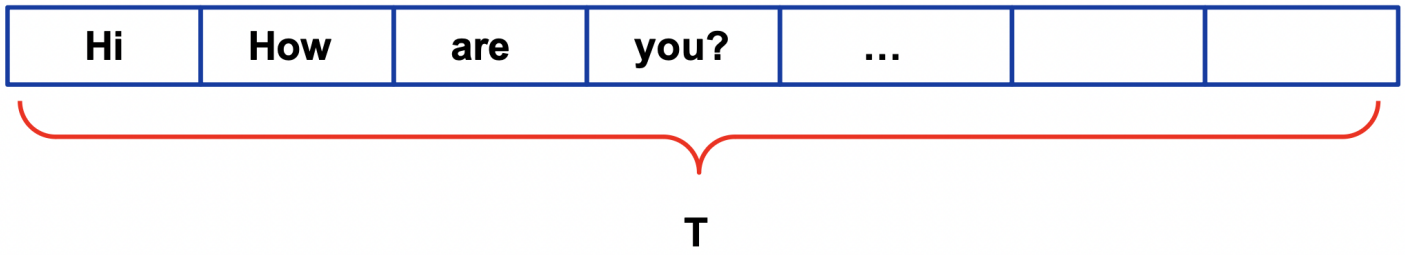
It was a very important milestone which had allow the development of models like BERT and subsequently LLMs which are **Transformers**.



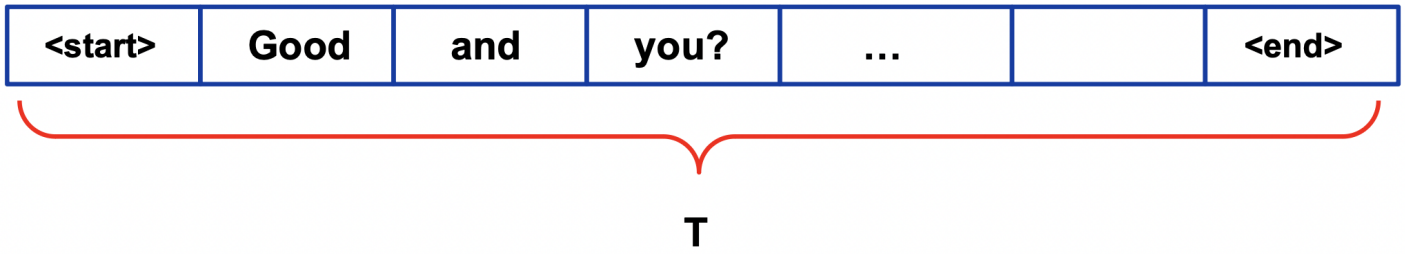
Architecture

We are going to start explaining the "Attention Mechanism".

Input sentence :



Output sentence :



Architecture : Attention

We are going to focus on the input sentence. We will represent each word of this sentence as a vector of dimension d_{model}.

$$E = \begin{bmatrix} — & (e_1)^T & — \\ — & \vdots & — \\ — & (e_T)^T & — \end{bmatrix} \in \mathbb{R}^{T \times d_{model}}$$

Now we also need to consider "Positional information" adding to each vector its respective "Positional Encoding".

$$PE(pos, 2i) = sin\left(\frac{pos}{N^{\frac{2i}{d_{model}}}}\right)$$

$$PE(pos, 2i+1) = cos\left(\frac{pos}{N^{\frac{2i}{d_{model}}}}\right)$$

Architecture : Attention

For example for the word in the $pos = 4$.

$$p_4 = \begin{bmatrix} cos\left(\frac{4}{N^{\frac{0}{d_{model}}}}\right) \\ sin\left(\frac{4}{N^{\frac{2}{d_{model}}}}\right) \\ cos\left(\frac{4}{N^{\frac{2}{d_{model}}}}\right) \\ sin\left(\frac{4}{N^{\frac{4}{d_{model}}}}\right) \\ \vdots \end{bmatrix}$$

Architecture : Attention

For every embedding vector $e_1, e_2, ..., e_T$ we going to add its respective positiional vector $p_1, p_2, ..., p_T$ to get $x_1, x_2, ..., x_T$.

$$X = \begin{bmatrix} — & (x_1)^T & — \\ — & \vdots & — \\ — & (x_T)^T & — \end{bmatrix} \in \mathbb{R}^{T \times d_{model}}$$

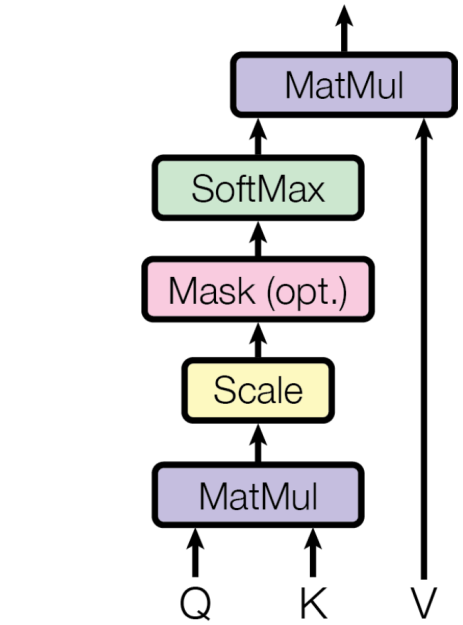
Learneable parameters :

$W^Q \in \mathbb{R}^{d_{model} \times d_k}, W^K \in \mathbb{R}^{d_{model} \times d_k}, W^V \in \mathbb{R}^{d_{model} \times d_v}$.

Definition :

$$\boxed{Q = X \cdot W^Q \in \mathbb{R}^{T \times d_k}} \quad \boxed{K = X \cdot W^K \in \mathbb{R}^{T \times d_k}} \quad \boxed{V = X \cdot W^V \in \mathbb{R}^{T \times d_v}}$$

Architecture : Attention



Architecture : Attention

i. MatMul :

$$A = Q \cdot K^T \in \mathbb{R}^{T \times T}$$

This matrix is called "Self-attention". Remember the definition of Q and K.

$$Q = \begin{bmatrix} — & (x_1)^T \cdot W^Q & — \\ — & \vdots & — \\ — & (x_T)^T \cdot W^Q & — \end{bmatrix} \quad K = \begin{bmatrix} — & (x_1)^T \cdot W^K & — \\ — & \vdots & — \\ — & (x_T)^T \cdot W^K & — \end{bmatrix}$$

Architecture : Attention

$$Q = \begin{bmatrix} — & (x_1)^T \cdot W^Q & — \\ — & \vdots & — \\ — & (x_T)^T \cdot W^Q & — \end{bmatrix} \quad K^T = \begin{bmatrix} | & \dots & | \\ (W^K)^T \cdot x_1 & \dots & (W^K)^T \cdot x_T \\ | & \dots & | \end{bmatrix}$$

$$Q \cdot K^T = \begin{bmatrix} (x_1)^T \cdot W^Q \cdot (W^K)^T \cdot (x_1) & \dots & (x_1)^T \cdot W^Q \cdot (W^K)^T \cdot (x_T) \\ \vdots & \ddots & \vdots \\ (x_T)^T \cdot W^Q \cdot (W^K)^T \cdot (x_1) & \dots & (x_T)^T \cdot W^Q \cdot (W^K)^T \cdot (x_T) \end{bmatrix}$$

If we define $B = W^Q \cdot (W^K)^T \in \mathbb{R}^{d_{model} \times d_{model}}$.

Architecture : Attention

$$Q \cdot K^T = \begin{bmatrix} (x_1)^T \cdot B \cdot (x_1) & \dots & (x_1)^T \cdot B \cdot (x_T) \\ \vdots & \ddots & \vdots \\ (x_T)^T \cdot B \cdot (x_1) & \dots & (x_T)^T \cdot B \cdot (x_T) \end{bmatrix}$$

$$Q \cdot K^T = \{r_{i,j}\}_{i,j \in \{1,\dots,T\}}$$

With $r_{i,j} = (x_i)^T \cdot B \cdot (x_j)$. This is a special case of inner product which is a mesaure of the similitude between two vectors.

Architecture : Attention

	Hi	How	are	you?
Hi	$r_{1,1}$	$r_{1,2}$	$r_{1,3}$	$r_{1,4}$
How	$r_{2,1}$	$r_{2,2}$	$r_{2,3}$	$r_{2,4}$
are	$r_{3,1}$	$r_{3,2}$	$r_{3,3}$	$r_{3,4}$
you?	$r_{4,1}$	$r_{4,2}$	$r_{4,3}$	$r_{4,4}$

Architecture : Attention

ii. Scale :

$$\frac{Q \cdot K^T}{\sqrt{d_k}}$$

This scaling helps prevent the scores from growing too large, which can lead to issues during the softmax calculation.

Architecture : Attention

iii. Mask (Optional) :

This step will be used in the "decoder" when we don't want consider relations between a word and another which is located forward to it. To this goal we define the next matrix.

Look ahead mask :

$$L = \begin{bmatrix} 0 & -\infty & \dots & -\infty \\ 0 & 0 & \dots & -\infty \\ \vdots & \vdots & \ddots & \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

This step consists in compute :

$$M = \frac{Q \cdot K^T}{\sqrt{d_k}} + L$$

Note : When we apply $exp()$ to $-\infty$ we get 0.

Architecture : Attention

iv. Softmax :

We apply this function for every row and we get the matrix S.

$$S = softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \in \mathbb{R}^{T \times T}$$

$$S_i, j = \frac{\exp(M_{i,j})}{\sum_{j=1}^{T} \exp(M_{i,j})}$$

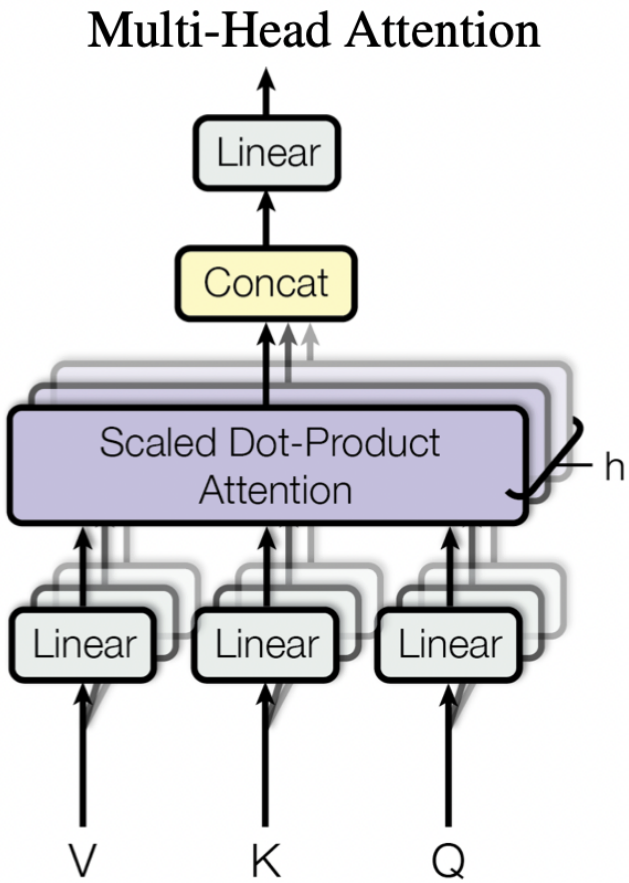
Architecture : Attention

v. Matmul :

$$H = softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V \in \mathbb{R}^{T \times d_v}$$

This is the output for one attentional head.

Architecture : Multihead - Attention



Architecture : Multihead - Attention

We going to consider h heads.

$$W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v} \quad \forall i \in \{1, ..., h\}.$$

$$H_i \in \mathbb{R}^{T \times d_v} \quad \forall i \in \{1, ..., h\}.$$

$$H = concat(H_1, H_2, ..., H_h) \in \mathbb{R}^{T \times d_v \cdot h}$$

Now we have another traineable parameters $W^O \in \mathbb{R}^{h \cdot d_v \times d_{model}}$.

$$O = H \cdot W^O \in \mathbb{R}^{T \times d_{model}}$$

Note : O has the same number of rows and columns with X.

Architecture : Add & Norm

We can consider the output of a layer as $\mathcal{F}(X)$.

Then :

$$Norm(X + \mathcal{F}(X)) \in \mathbb{R}^{T \times d_{model}}$$

Architecture : Feed Forward

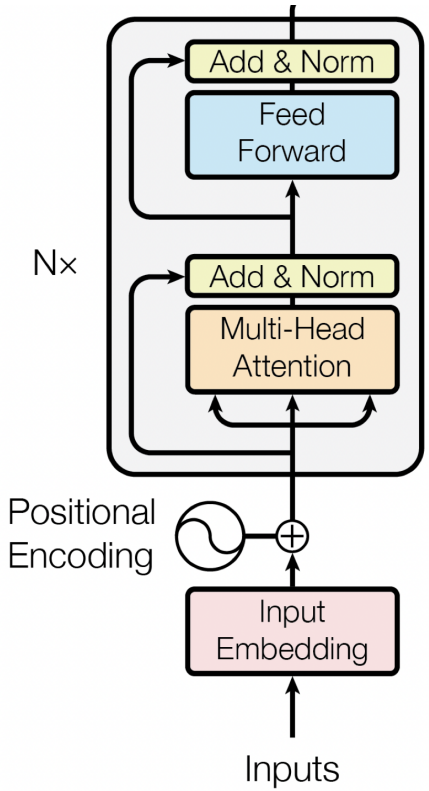
$$W_1 \in \mathbb{R}^{d_{model} \times n},\ b_1 \in \mathbb{R}^{T \times n},\ W_2 \in \mathbb{R}^{n \times d_{model}},\ b_2 \in \mathbb{R}^{T \times d_{model}},$$

$$FFN(X) = \max(0, X \cdot W_1 + b_1) \cdot W_2 + b_2$$

Architecture : Encoder

The output of the encoder is :

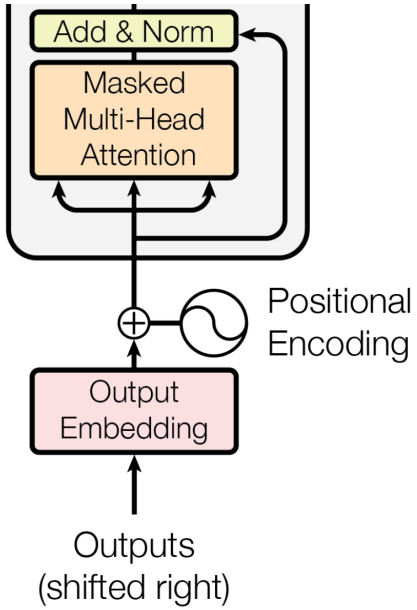
$$O_{encoder} \in \mathbb{R}^{T \times d_{model}}$$



Architecture : Decoder - Part I

The output of the decoder is :

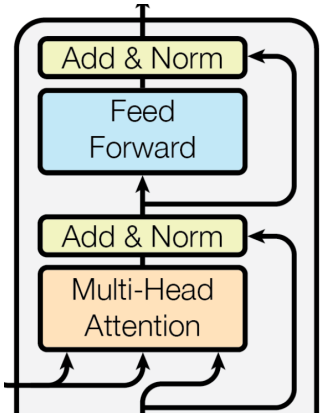
$$O_{decoder} \in \mathbb{R}^{T \times d_{model}}$$



Architecture : Decoder - Part II

For this section we consider :

$$Q = K = O_{encoder}, V = O_{decoder}$$

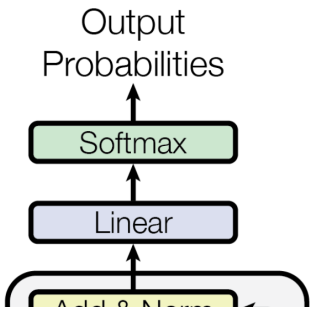


Now we have to apply the attention layer and the others layers as usual.

Architecture : Output

We going to call the output by :

$$O \in \mathbb{R}^{T \times d_{model}}$$



Now we have to apply a flatten layer we get :

$$O^{flatten} \in \mathbb{R}^{T \cdot d_{model}}$$

Now we apply a linear layer :

$$O^{pre-final} \in \mathbb{R}^{N_{vocab-size}}$$

Architecture : Output

Finally we apply a softmax layer :

$$O^{final} \in \mathbb{R}^{N_{vocab-size}}$$

And on this way we have the probability of each word from our vocabulary to be the next word.

More complex models

We can get more complex models combining the encoder and decoder architectures.