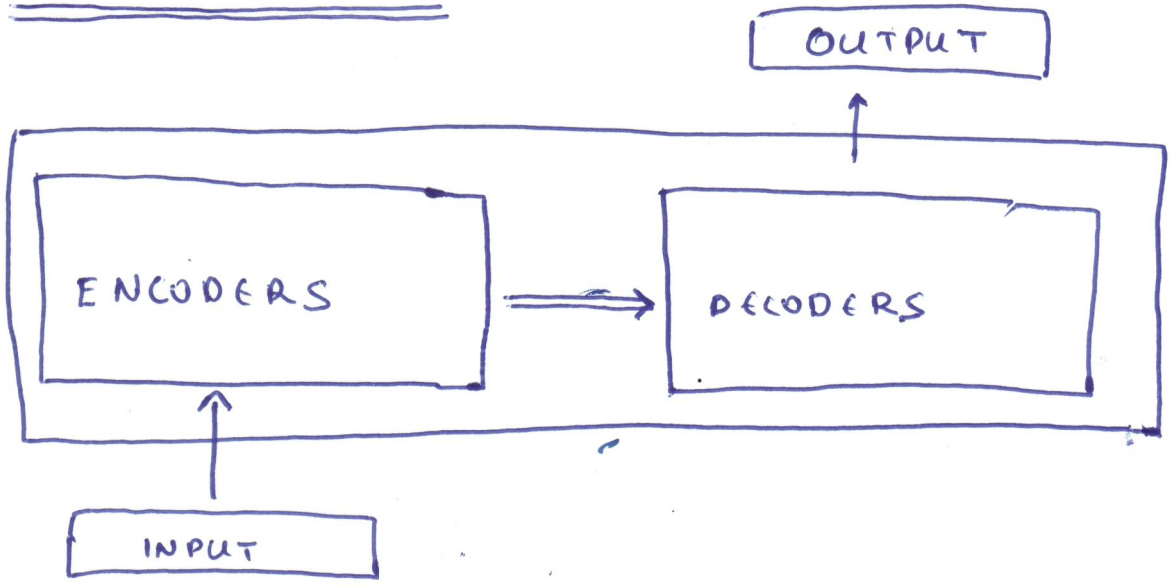
Transformer Model

#1

BIRDS EYE VIEW:

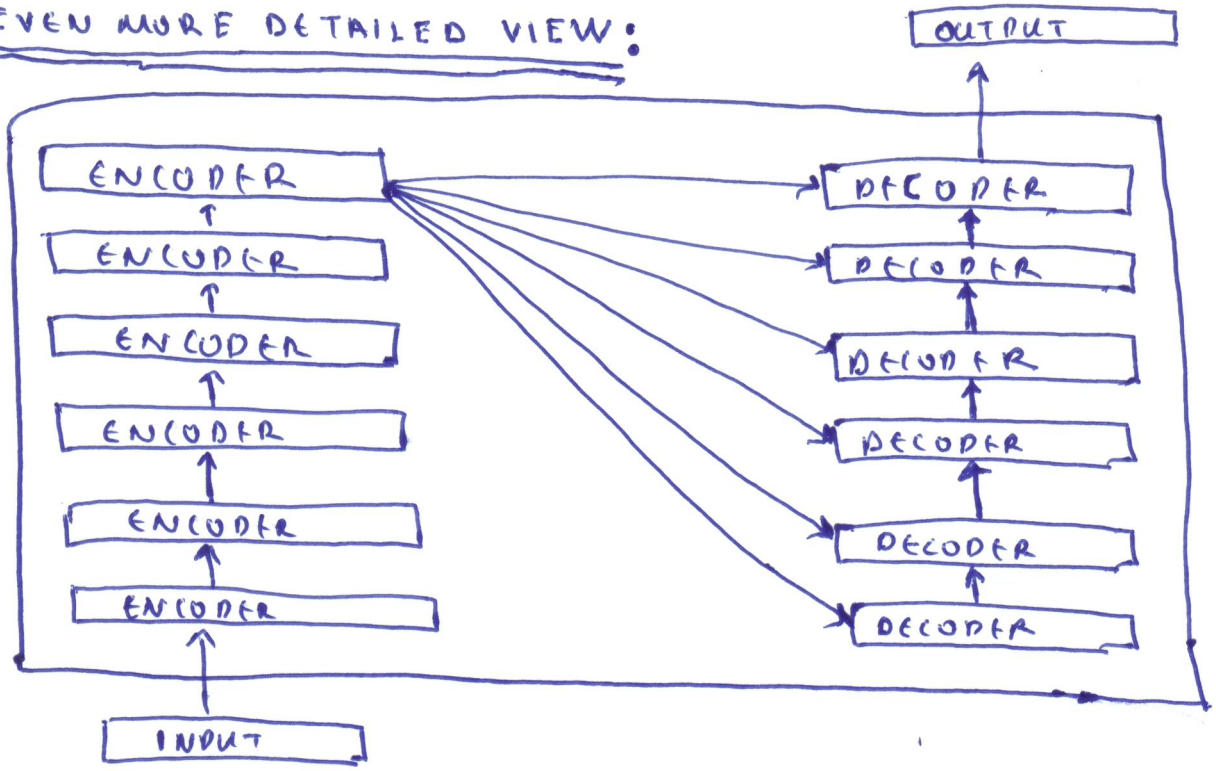
MAIN COMPONENTS : (α) Encoding Component
(β) Decoding Component

MORE DETAILED VIEW:



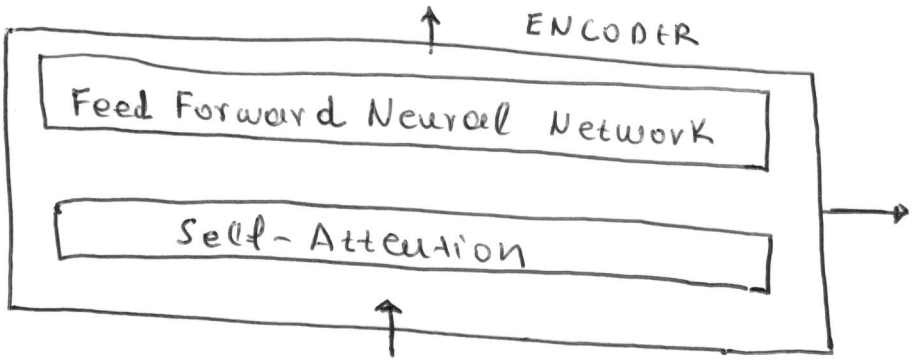
- The encoding component is a stack of encoders. The original paper stacks six of them.
- The decoding component is a stack of decoders of the same number.

EVEN MORE DETAILED VIEW:



- The encoders are all identical in structure but they do not share weights.

- Each encoder-layer is broken down into two sub-layers:
  (a): A self-Attention Layer
  (B): A Feed-Forward Neural Network

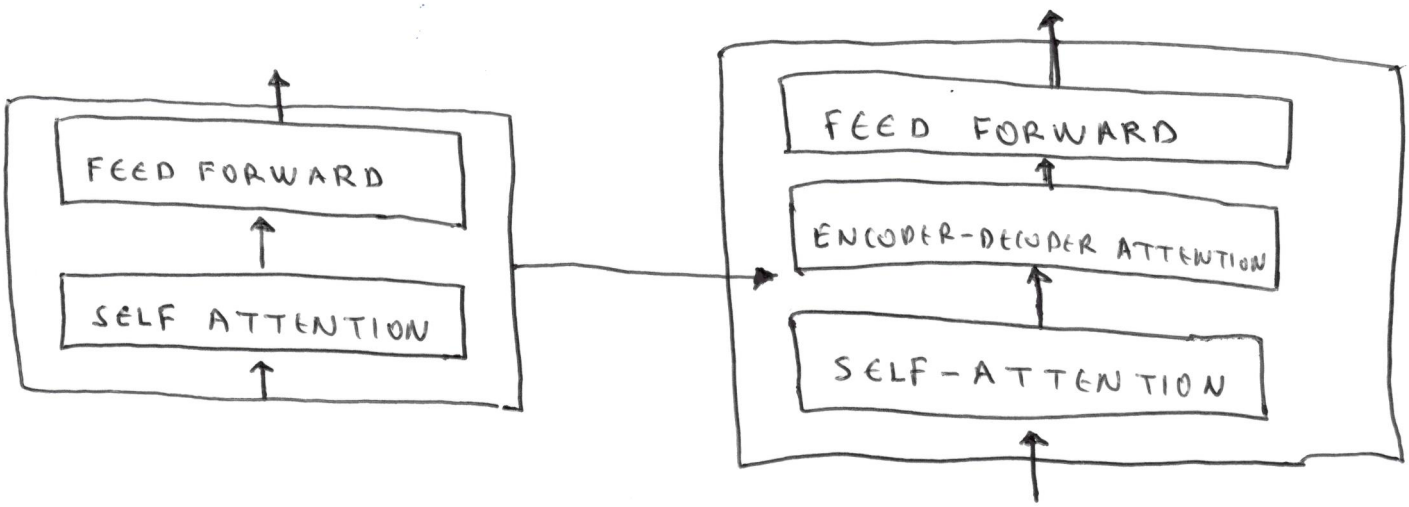


- The encoder's inputs first flow through a self-attention layer that help the neural mechanism to look at other instances of the input sequence as it encodes a particular timestep of the input sequence.

- The outputs of the self-attention layer are fed to a feed-forward neural network. The exact same feed-forward network is independently applied to each position.

- The decoder has both those layers, but between them there exists an attention layer that helps the decoder to focus on relevant parts of the input sequence.

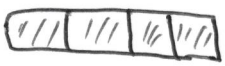
[This attention layer is similar to the attention mechanisms utilized by sequence to sequence Models]

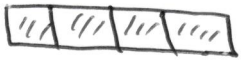


Bringing The Tensors Into The Picture

- Let's look at the various vectors/tensors and how they flow between the various components of the transformer model in order to turn the input of a trained model into an output.

★ x_1 =  Je

x_2 =  suis

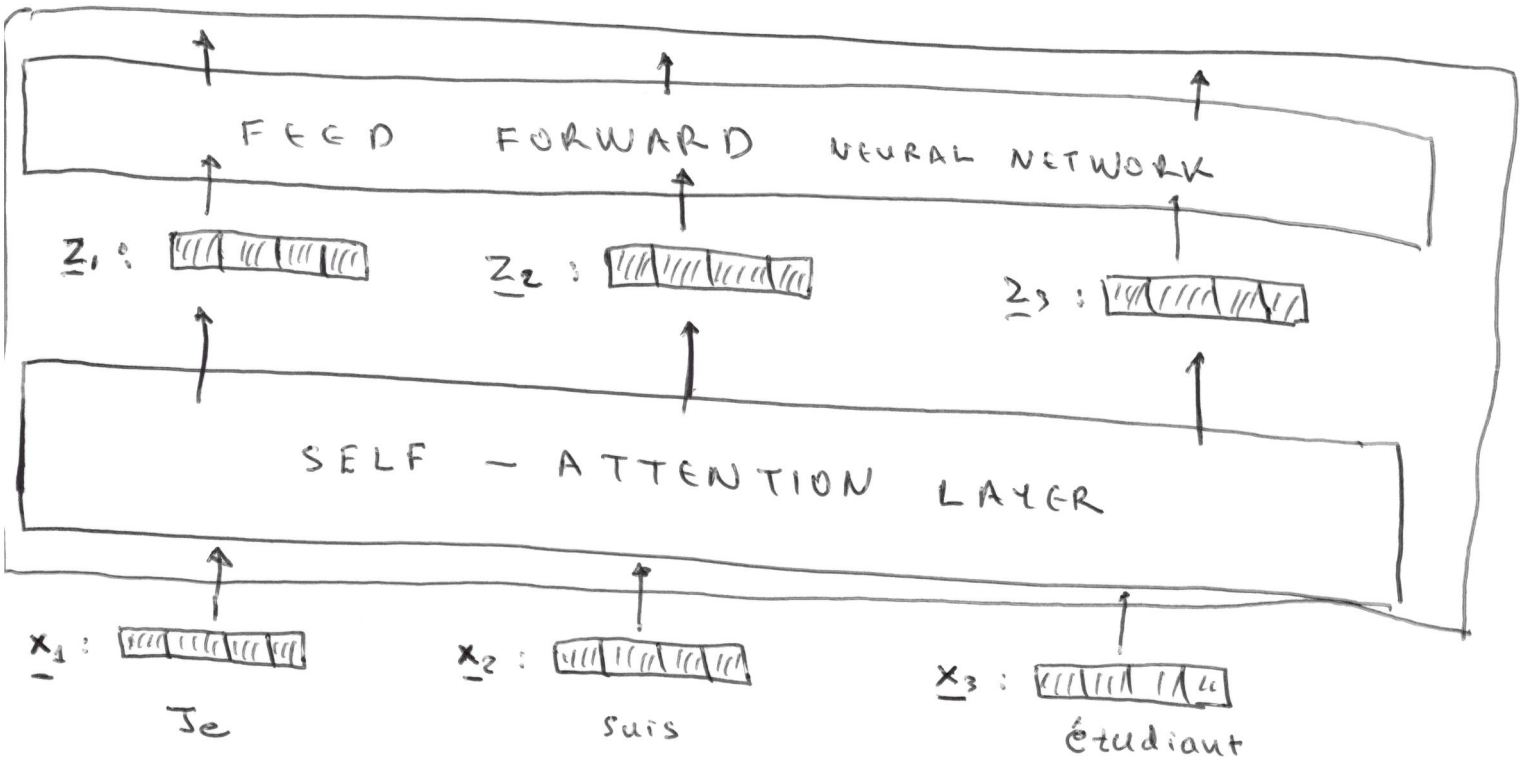
x_3 =  étudiant

★ Each word is embeded into a vector of size 512.

* The embedding happens only at the bottom-most encoder. The abstraction that is common to all the encoders is that they receive a list of vectors each of size 512.

* The bottom encoder receives the word embeddings while the other encoders receive the output from the directly previous encoder layer.

* The size of the aforementioned list is a hyperparameter that corresponds to the length of the longest input sequence in the training dataset.

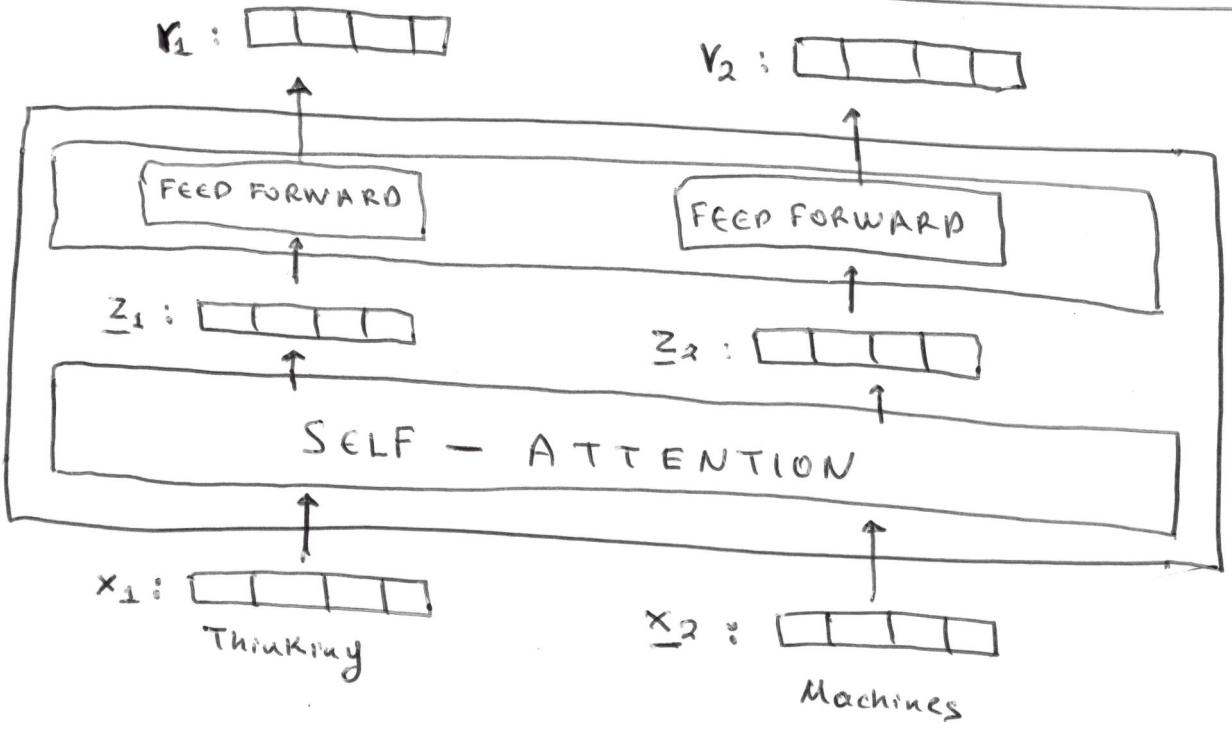


* After embedding the words in the input sequence, each of them flows through each of the two layers of the encoder.

* One key property of the Transformer, is that each input word (or each input instance at the corresponding position) flows through its own path in the encoder.

- There are dependencies between these paths in the self-attention layer.

* The feed-forward layer does not have these dependencies and thus, the various paths can be executed in parallel.



- The word at each position passes through a self-attention process. Then, they each pass through a feed-forward neural network.

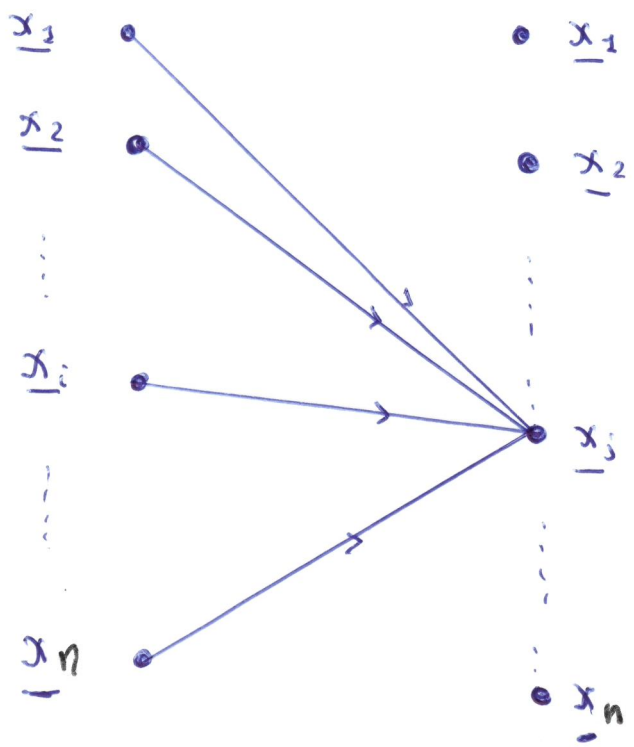
[the exact same, neural network, with each vector flowing through it seperately]

Self-Attention at a High Level

- When the model processes a particular word (or any given feature vector at a specific timestamp), self-attention allows it to look at other positions in the input sequence for clues that can help it to acquire a better encoding for the current token.

- In an analogy with RNNs, one may think that maintaining a particular hidden state allows them to incorporate a representation of the previous/past instances of the input sequence.

- Self-attention is the method employed by the Transformer in order to convey "understanding" of the previous words into the word currently being processed.

- 

- Encoding of the j-th word may take into consideration the encoding of all other words in the sentence.

Self-Attention in Detail

Step 1:

- The first step in calculating self-attention is to create three vectors from each of the encoder's input vectors (in this case, the embedding of each word).
- So for each word, a Query vector, a Key vector and a Value vector are being created.
- These vectors are created by multiplying the embedding by the three matrices that will be acquired during the training process.
- Notice that these new vectors are smaller in dimension than the original embedding vector. Their dimensionality can be 64 instead of 512. Of course, this is an architectural design choice. They don't have to be smaller. This choice ensures that the computation of the multi-head attention remains constant.

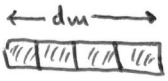
(*) Let n be the number of elements in the sequence such that:

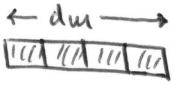
$$X = \{ \underline{x}_1, \underline{x}_2, \dots, \underline{x}_n \} \quad \text{where} \quad \underline{x}_i \in \mathbb{R}^{dm} \tag{1}$$

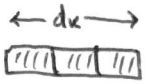
with dm denoting the dimensionality of the embedding space.

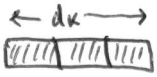
(*) For the illustrated example below, we have that $n = 2$.

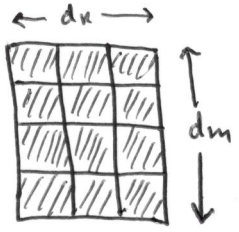
INPUT Thinking Machines

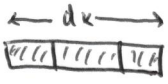
EMBEDDING x_1 : 

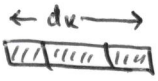
x_2 : 

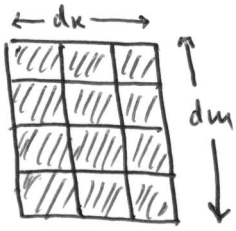
QUERIES q_1 : 

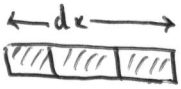
q_2 : 

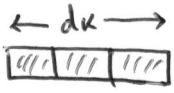
Q_w : 

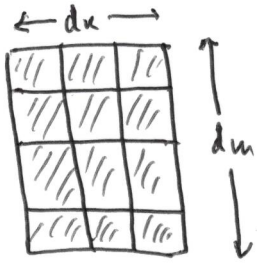
KEYS: K_1 : 

K_2 : 

K_w : 

VALUES V_1 : 

V_2 : 

V_w : 

(*) We may write that:

$$\underline{q}_j = \underline{x}_j \circ \underline{\underline{Q}}_w \quad , \forall j \in [4] \quad (2)$$

$$[1 \times d_k] = [1 \times d_m] \circ [d_m \times d_k]$$

$$\underline{k}_j = \underline{x}_j \circ \underline{\underline{K}}_w \quad , \forall j \in [4] \quad (3)$$

$$[1 \times d_k] = [1 \times d_m] \circ [d_m \times d_k]$$

$$\underline{v}_j = \underline{x}_j \circ \underline{\underline{V}}_w \quad , \forall j \in [4] \quad (4)$$

$$[1 \times d_k] = [1 \times d_m] \circ [d_m \times d_k]$$

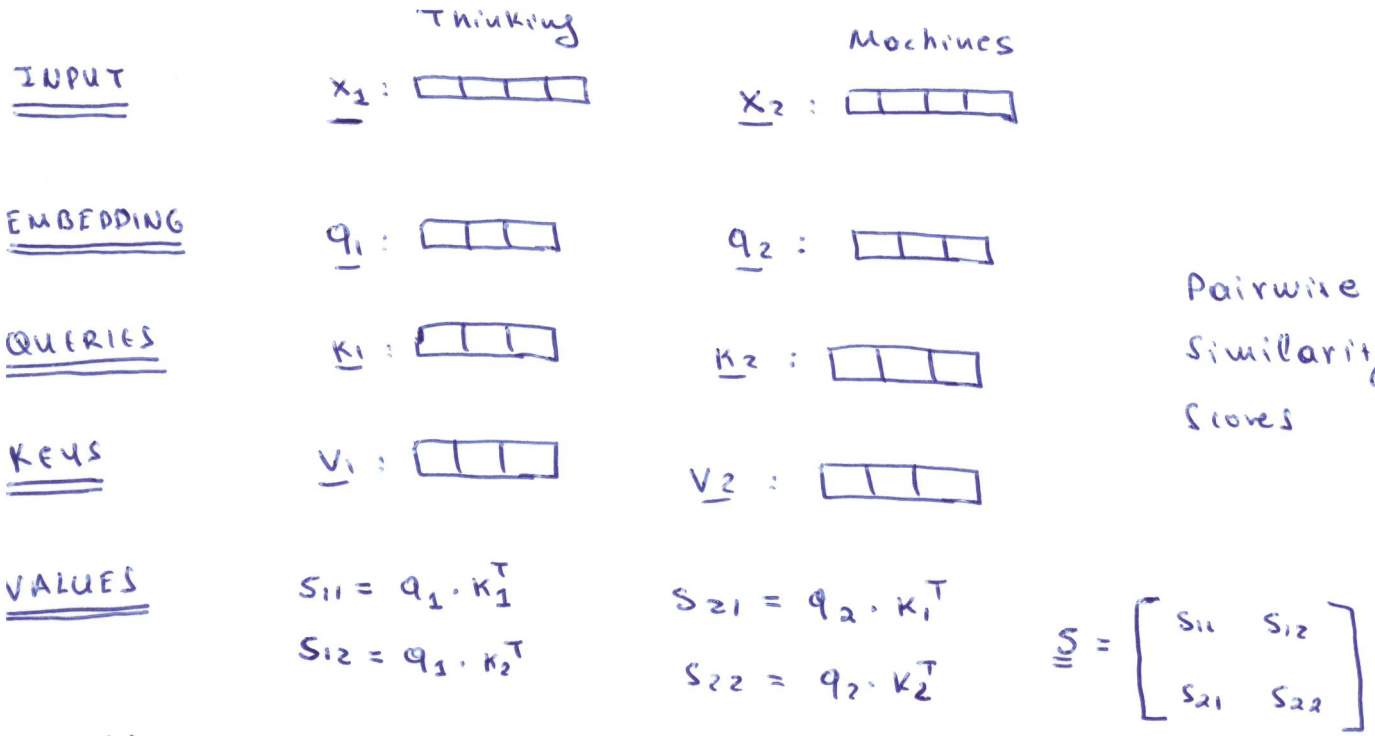
Step 2: ▸ The second step in calculating self-attention is to calculate a score. Say for example that we are calculating the self-attention score for the first word "Thinking".

▸ Thus, we need to score each word of the input sequence against this word.

▸ This score determines how much focus to place on other parts of the input sequence as we encode the feature vector at the current position.

★ We may write that:

$$s_{ij} = q_i \cdot k_j^T \in \mathbb{R} \quad \forall i \in [n], \ \forall j \in [n] \qquad (5)$$



★ Let $Q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix}$, $K = \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{bmatrix}$, $V = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \in \mathbb{R}^{n \times d_k}$

★ $S = Q \cdot K^T \in \mathbb{R}^{n \times n}$ ★ For this particular example: $S = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \begin{bmatrix} k_1^T & k_2^T \end{bmatrix} \Rightarrow$

$$S = \begin{bmatrix} q_1 k_1^T & q_1 k_2^T \\ q_2 k_1^T & q_2 k_2^T \end{bmatrix}$$

Steps 3 and 4: ▶ Decide scores by $\sqrt{d_k}$ which leads to more stable gradients.

▶ Apply softmax normalization at each row of the pairwise score matrix $\underline{\underline{S}}$.

$$\underline{\underline{\hat{S}}} = \text{softmax}\left(\sqrt{d_k}^{-1} \circ \underline{\underline{S}}\right) \quad (6)$$

★ For this particular example we have that:

$$\hat{S}_{11} + \hat{S}_{12} = 1 \quad \text{and} \quad \hat{S}_{21} + \hat{S}_{22} = 1$$

Steps 5 and 6: Produce the outputs of the self-attention layer:

$$\underline{z}_j = \sum_{r=1}^{n} \hat{S}_{jr} \circ \underline{v}_r \;, \quad \forall\, j \in [n] \quad (7)$$

where $\underline{z}_j \in \mathbb{R}^{d_k}$

★ Assuming that the sequence $\mathcal{X}$ of input patterns is stored in a matrix $\underline{\underline{X}}$ ($\underline{\underline{X}} \in \mathbb{R}^{n \times dm}$) such that:

$$\underline{\underline{X}} = \begin{bmatrix} \leftarrow dm \rightarrow \\ \underline{x}_1 \\ \underline{x}_2 \\ \vdots \\ \underline{x}_n \end{bmatrix} \uparrow n \downarrow$$

, the output of the attention layer will be given as $\underline{\underline{Z}} = \underline{\underline{\hat{S}}} \circ \underline{\underline{V}}$ (8)

such that:

$$\underline{\underline{Z}} = \begin{bmatrix} \leftarrow d_k \rightarrow \\ \underline{z}_1 \\ \underline{z}_2 \\ \vdots \\ \underline{z}_n \end{bmatrix}$$

$$[n \times d_k] = [n \times n] \times [n \times d_k]$$

Matrix Calculation of Self-Attention

- Letting $\underline{\underline{X}} \in \mathbb{R}^{n \times dm}$ the matrix of the input patterns, we may write that:

$$\text{(I)}: \quad \underline{\underline{Q}} = \underline{\underline{X}} \cdot \underline{\underline{Q}}_W \in \mathbb{R}^{n \times dk}$$

$$\text{(II)}: \quad \underline{\underline{K}} = \underline{\underline{X}} \cdot \underline{\underline{K}}_W \in \mathbb{R}^{n \times dk}$$

$$\text{(III)}: \quad \underline{\underline{V}} = \underline{\underline{X}} \cdot \underline{\underline{V}}_W \in \mathbb{R}^{n \times dk}$$

$$\text{(IV)}: \quad \hat{\underline{\underline{S}}} = \text{softmax}\left(\frac{\underline{\underline{Q}} \cdot \underline{\underline{K}}^T}{\sqrt{dk}} \right) \in \mathbb{R}^{n \times n}$$

$$\text{(V)}: \quad \underline{\underline{Z}} = \hat{\underline{\underline{S}}} \cdot \underline{\underline{V}} \in \mathbb{R}^{n \times dk}$$

Mind that this is actually the output of one attention head. Thus, by considering M attention heads, we would end up with output vectors of dm dimensionality

$$M * dk = dm$$

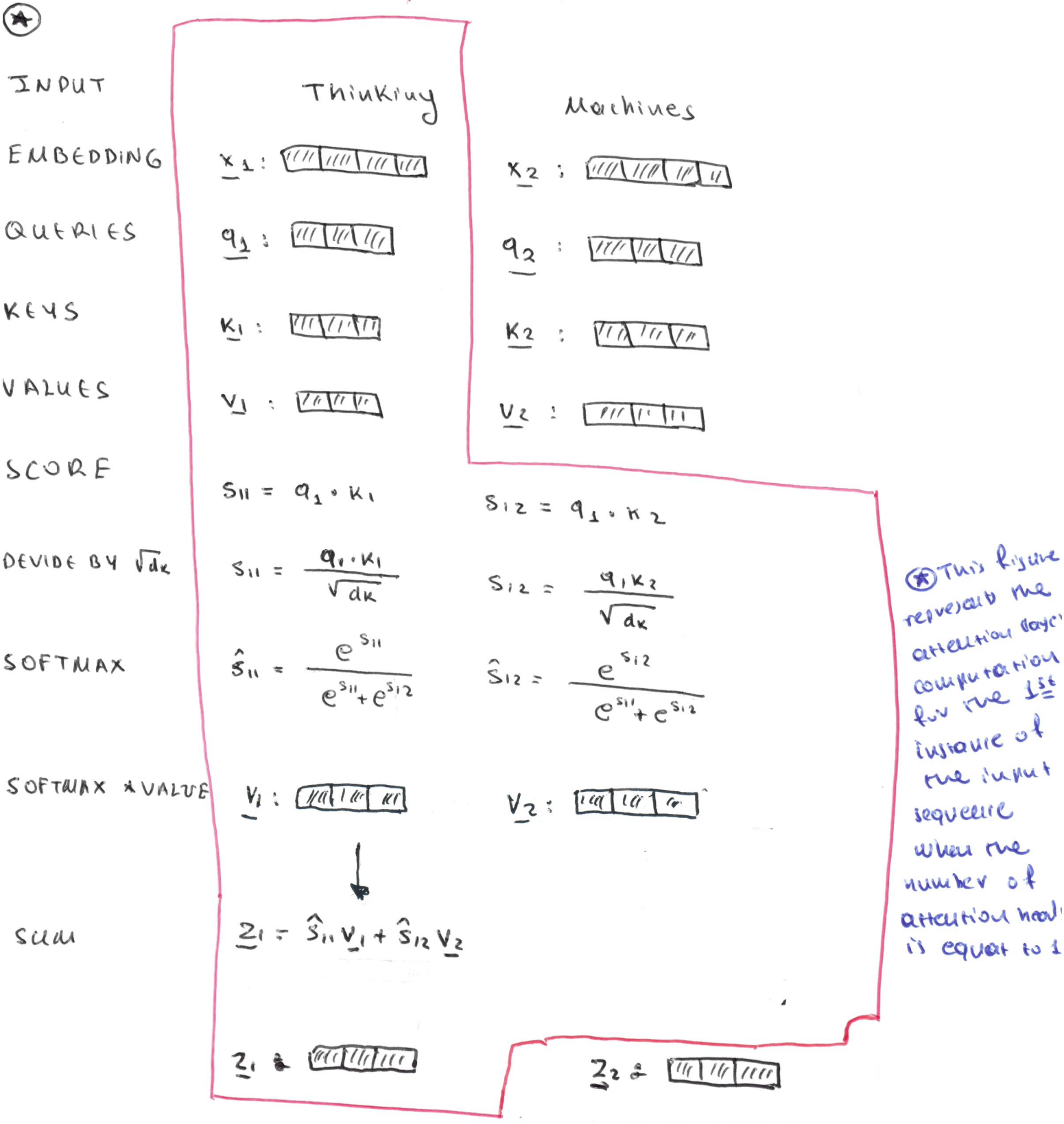
by concatenating the outputs of the various heads.

$$\left\{ \begin{array}{c} M = \dfrac{dm}{dk} \\ \| \\ 8 \text{ heads} \end{array} \right\}$$

Multiple Attention Heads

▶ Multiple heads extend the model's ability to focus on different positions.

→ Computation block for z_1



✱ This figure represents the attention layer computation for the 1st instance of the input sequence when the number of attention heads is equal to 1

✱ In the example above, z_1 contains information about every other encoding, but it could be dominated by the actual word itself.

- Multiple heads allow the attention layer to retain multiple "representation subspaces". With multiple attention heads we have multiple sets of Query / Key / Value matrices.

- The original Transformer model defined 8 attention heads resulting in 8 sets of each encoder-decoder pair.

- Each set of matrices is randomly initialized. After the completion of the training process, each set of matrices is utilized to project the input embeddings (or vectors from lower encoders/decoders) into a different representation subspace.

- Let $m \in \{0, 1, \dots, M-1\}$ indicate a particular attention head associated with weight-matrices $\underline{\underline{Q}}_w^{(m)}$, $\underline{\underline{K}}_w^{(m)}$ and $\underline{\underline{V}}_w^{(m)}$.
  $[d_m \times d_k]$ $[d_m \times d_k]$ $[d_m \times d_k]$

- $$\begin{cases} \underline{\underline{Q}}^{(m)} = \underline{\underline{X}} \circ \underline{\underline{Q}}_w^{(m)} \in \mathbb{R}^{n \times d_k} \\ \underline{\underline{K}}^{(m)} = \underline{\underline{X}} \circ \underline{\underline{K}}_w^{(m)} \in \mathbb{R}^{n \times d_k} \\ \underline{\underline{V}}^{(m)} = \underline{\underline{X}} \circ \underline{\underline{V}}_w^{(m)} \in \mathbb{R}^{n \times d_k} \end{cases}$$

  $$\underline{\underline{S}}^{(m)} = \frac{\underline{\underline{Q}}^{(m)} \bullet \left(\underline{\underline{K}}^{(m)}\right)^T}{\sqrt{d_k}} \in \mathbb{R}^{n \times n}$$

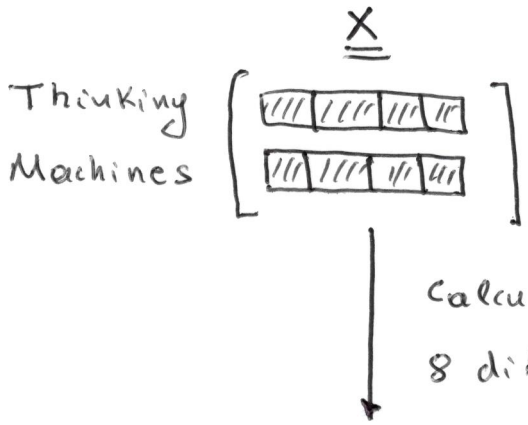
  $$\hat{\underline{\underline{S}}}^{(m)} = \text{softmax}\left(\underline{\underline{S}}^{(m)}\right) \in \mathbb{R}^{n \times n}$$

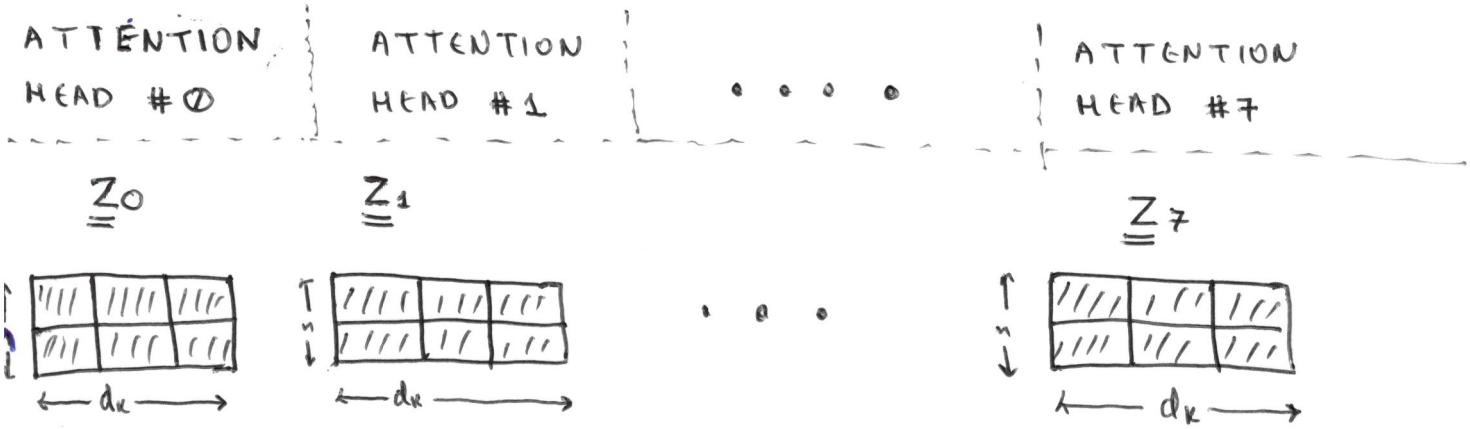
- $$\underline{\underline{Z}}^{(m)} = \hat{\underline{\underline{S}}}^{(m)} \circ \underline{\underline{V}}^{(m)} \in \mathbb{R}^{n \times d_k}, \quad 0 \le m \le M-1$$

- FORM THE FINAL CONCATENATED OUTPUT:

$$\hat{\underline{\underline{Z}}} = \left[\underline{\underline{Z}}^{(0)}, \underline{\underline{Z}}^{(1)}, \dots, \underline{\underline{Z}}^{(m-1)}\right] \in \mathbb{R}^{n \times d_m}$$

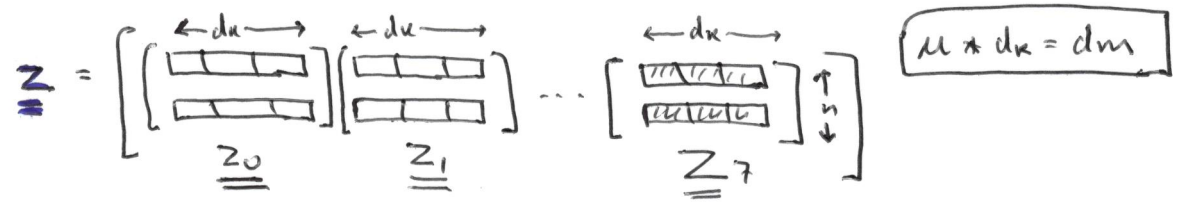
Schematic Representation of the FeedForward Computation Step





Applying the Feed-Forward Neural Network Requires:

(A): since the feed-forward neural network is not expecting 8 matrices (it's expecting a single matrix where a vector is assigned to each word), we need to concatenate the 8 individual heads into a single matrix.



(B): multiply the concatenated matrix with an additional weight-matrix W_0 such that:

$$W_0 \in \mathbb{R}^{dm \times dm}$$

$$R = Z \cdot W_0$$

$$[n \times dm] \quad [n \times dm] \quad [dm \times dm]$$

Schematic Representation of the Encoder Layer