

# Machine Learning

## Transformer Network

---

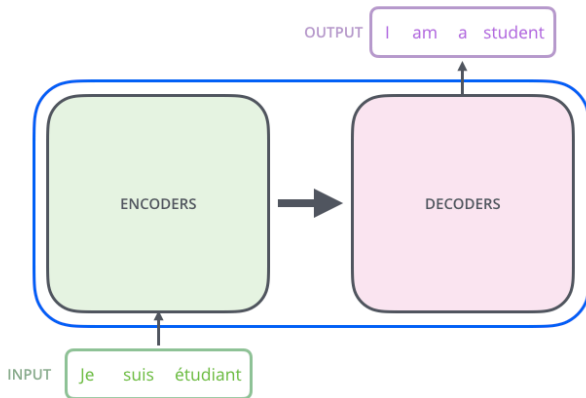
# Transformer Network



# Transformer Network : Vue générale

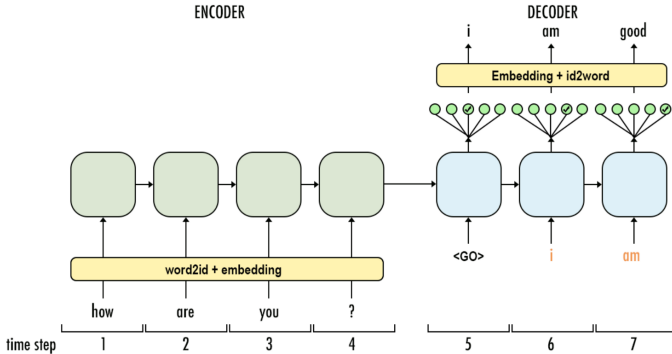


# Transformer Network : Vue générale



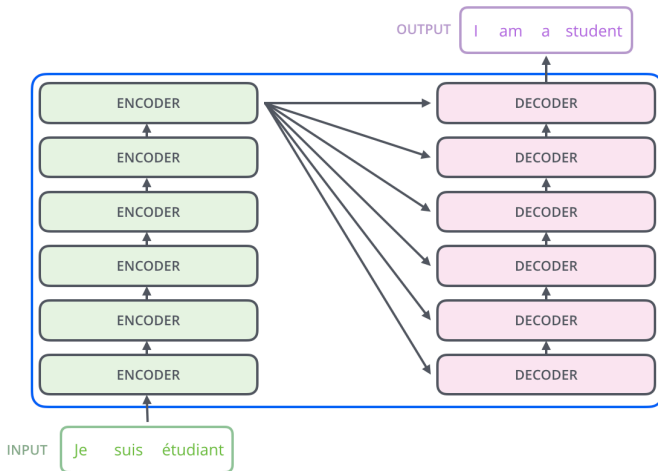
# Transformer Network : Vue générale

Rappelez-vous, les encodeurs-décodeurs :

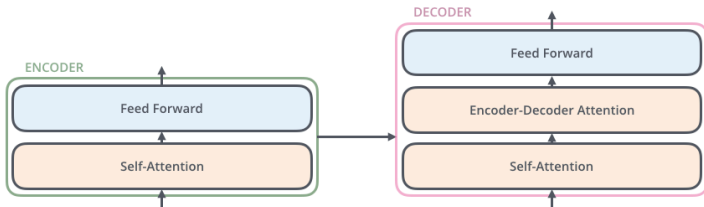


# Transformer Network : Vue générale

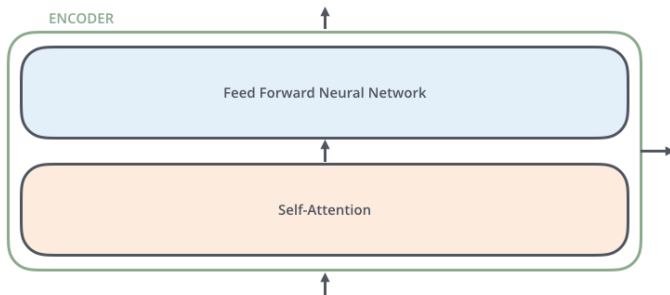
## DEEP LEARNING



# Transformer Network : Vue générale

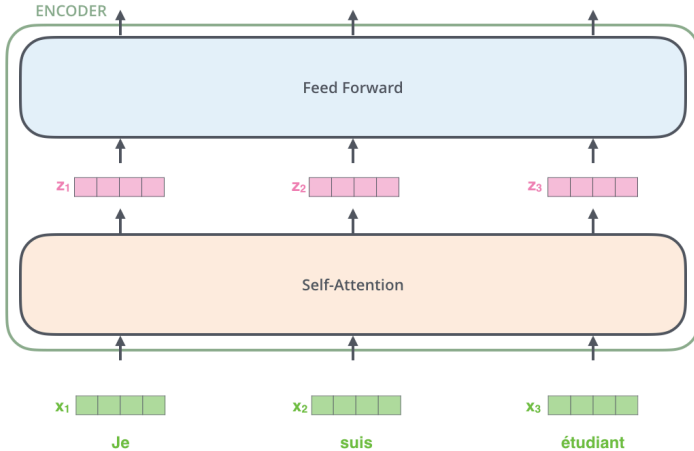


# Transformer Network : Encodeur

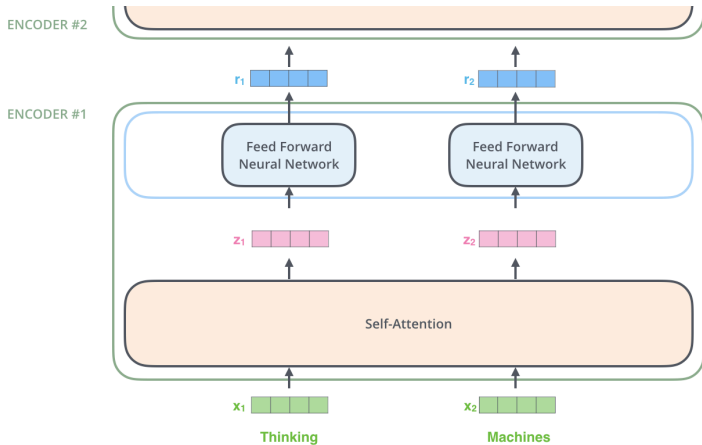




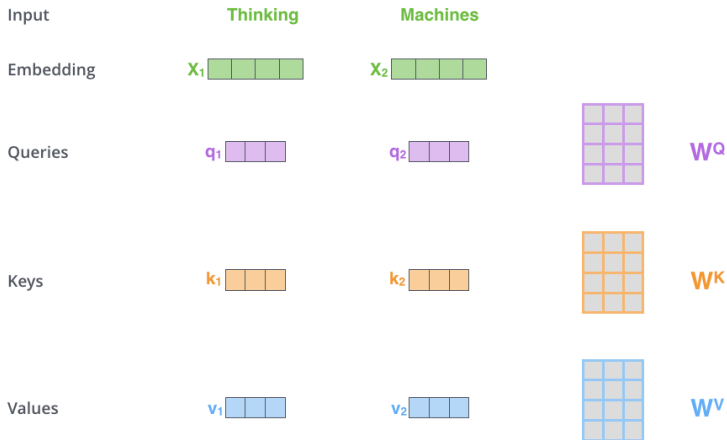
# Transformer Network : Encodeur



# Transformer Network : Encodeur



# Transformer Network : Self-Attention



# Transformer Network : Self-Attention

$$\begin{matrix} X \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} W^Q \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} Q \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix}$$

$$\begin{matrix} X \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} W^K \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} K \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix}$$

$$\begin{matrix} X \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} W^V \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} V \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix}$$

# Transformer Network : Self-Attention

$$\text{softmax} \left( \frac{\begin{matrix} \text{Q} \\ \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \square \\ \hline \end{array} \end{matrix} \times \begin{matrix} \text{K}^T \\ \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \square \\ \hline \end{array} \end{matrix} \right) \begin{matrix} \text{V} \\ \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \square \\ \hline \end{array} \end{matrix}$$
$$= \begin{matrix} \text{Z} \\ \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \square \\ \hline \end{array} \end{matrix}$$

Avec le softmax défini ainsi :  $\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$

# Transformer Network : Self-Attention

Input

Embedding

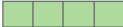
Queries

Keys

Values

Score

Thinking

$x_1$  

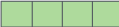
$q_1$  

$k_1$  

$v_1$  

$q_1 \cdot k_1 = 112$

Machines

$x_2$  

$q_2$  

$k_2$  

$v_2$  

$q_1 \cdot k_2 = 96$

# Transformer Network : Self-Attention

Input

Embedding

Queries

Keys

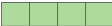
Values

Score

Divide by 8 (  $\sqrt{d_k}$  )

Softmax

Thinking

$x_1$  

$q_1$  

$k_1$  

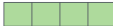
$v_1$  

$q_1 \cdot k_1 = 112$

14

0.88

Machines

$x_2$  

$q_2$  

$k_2$  

$v_2$  

$q_2 \cdot k_2 = 96$

12

0.12

# Transformer Network : Self-Attention

Input

Embedding

Queries

Keys

Values

Score

Divide by 8 (  $\sqrt{d_k}$  )

Softmax


Softmax

X

Value

Sum

Thinking

$x_1$  

$q_1$  

$k_1$  

$v_1$  

$q_1 \cdot k_1 = 112$


14

0.88

$v_1$  

$z_1$  

Machines

$x_2$  

$q_2$  

$k_2$  

$v_2$  

$q_2 \cdot k_2 = 96$

12

0.12

$v_2$  

$z_2$  

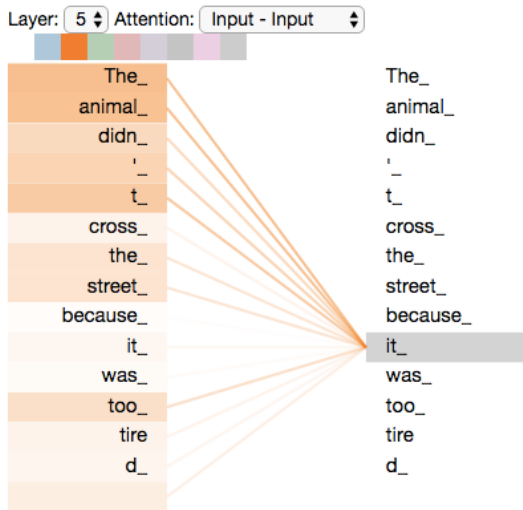


# Transformer Network : Self-Attention

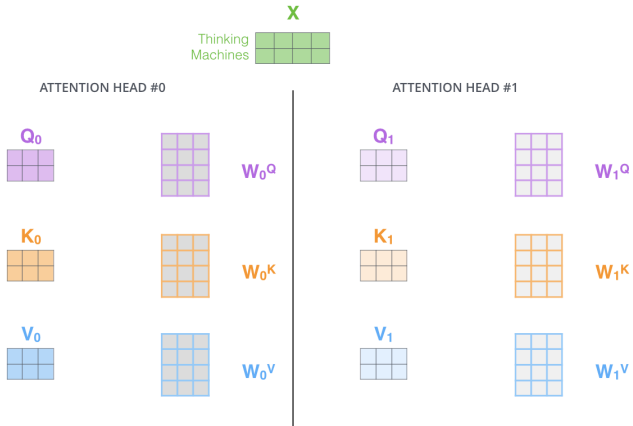
$$\text{softmax} \left( \frac{\begin{matrix} \text{Q} \\ \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \square \\ \hline \end{array} \end{matrix} \times \begin{matrix} \text{K}^T \\ \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \square \\ \hline \end{array} \end{matrix} \right) \begin{matrix} \text{V} \\ \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \square \\ \hline \end{array} \end{matrix} \\ \sqrt{d_k}$$
$$= \begin{matrix} \text{Z} \\ \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \square \\ \hline \end{array} \end{matrix}$$

Avec le softmax défini ainsi :  $\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$

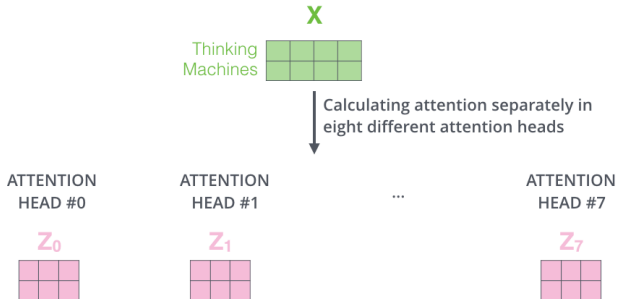
# Transformer Network : Self-Attention



# Transformer Network : Multi-Head-Attention



# Transformer Network : Multi-Head-Attention



# Transformer Network : Encodeur (Résumé)

Un modèle sans récurrence, uniquement des sommes pondérées :

1) This is our input sentence\*

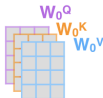
2) We embed each word\*

3) Split into 8 heads. We multiply  $X$  or  $R$  with weight matrices

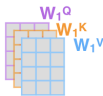
4) Calculate attention using the resulting  $Q/K/V$  matrices

5) Concatenate the resulting  $Z$  matrices, then multiply with weight matrix  $W^O$  to produce the output of the layer

Thinking  
Machines



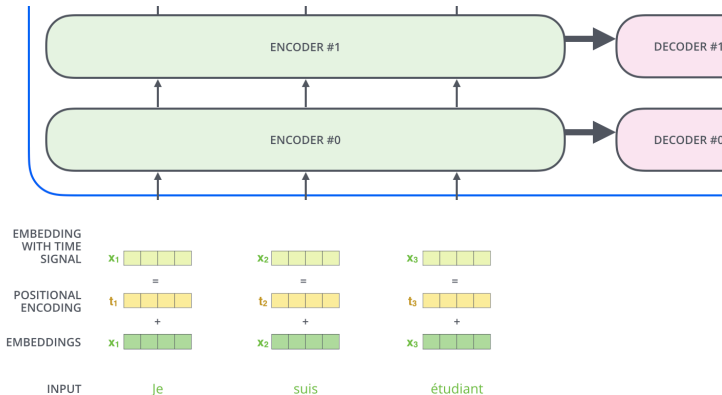
\* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



$W^O$

⇒ encodage de la position (Positional Encoding)

# Transformer Network : Positional Encoding



$$PE_{pos,2i} = \sin(pos/10000^{2i/d_{model}})$$

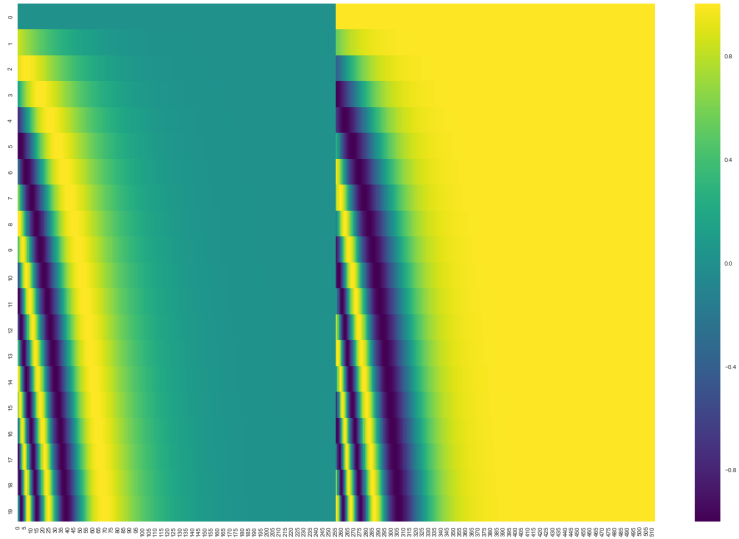
$$PE_{pos,2i+1} = \cos(pos/10000^{2i/d_{model}})$$

# Transformer Network : Positional Encoding

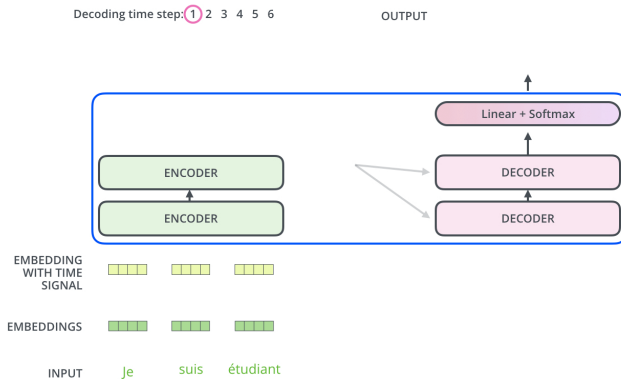




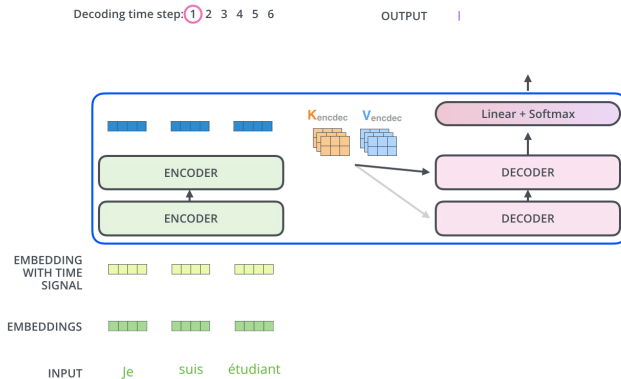
# Transformer Network : Positional Encoding



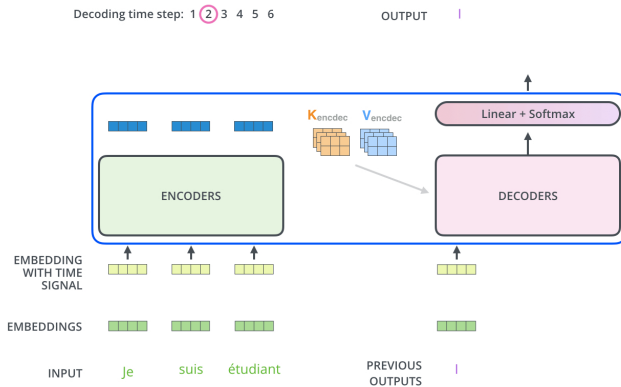
# Transformer Network : Decodeur



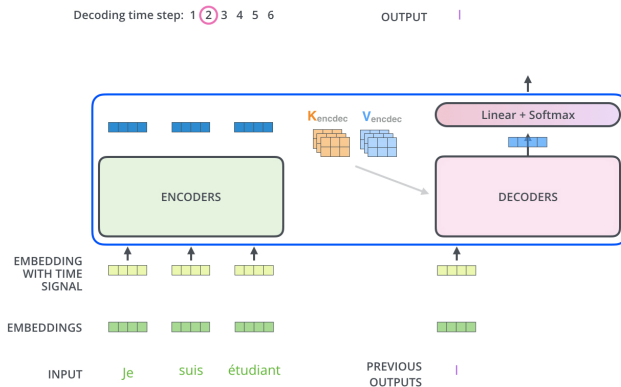
# Transformer Network : Decodeur



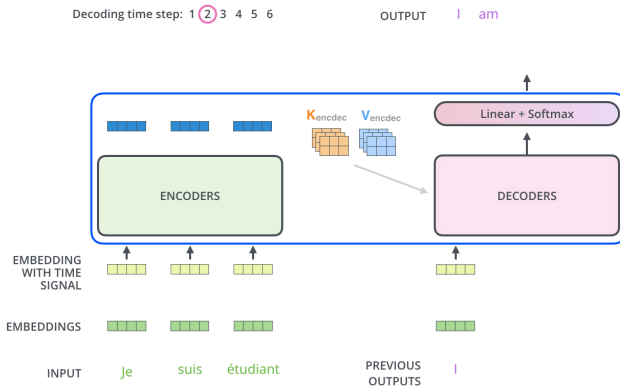
# Transformer Network : Decodeur



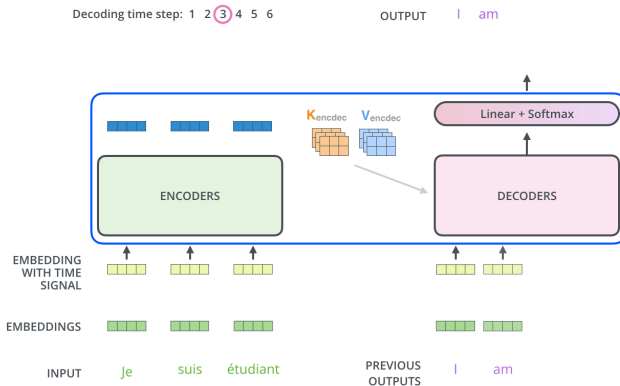
# Transformer Network : Decodeur



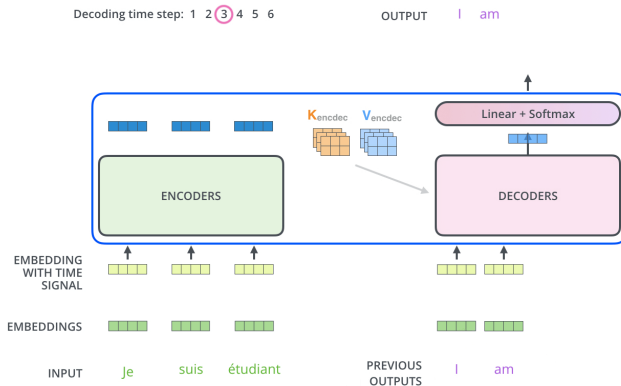
# Transformer Network : Decodeur



# Transformer Network : Decodeur

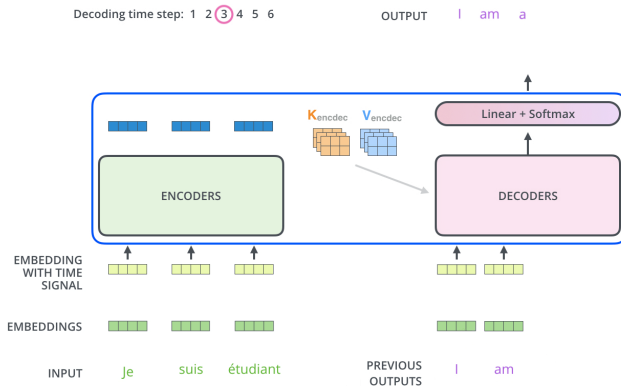


# Transformer Network : Decodeur

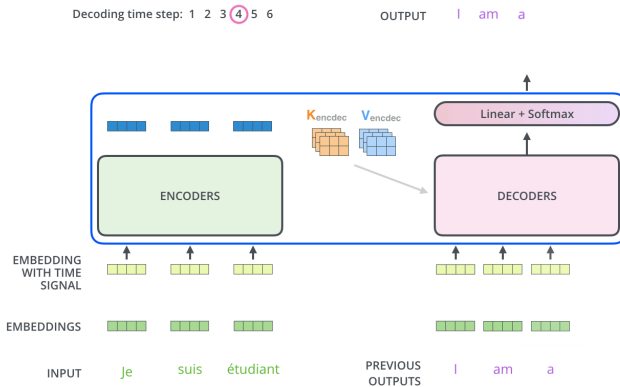




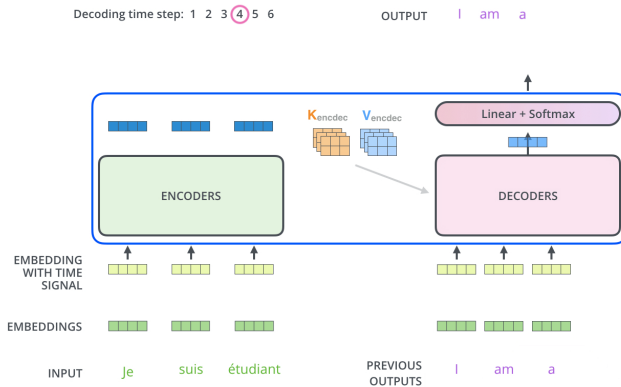
# Transformer Network : Decodeur



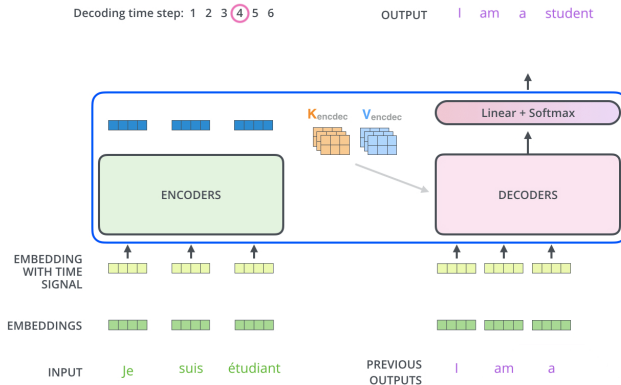
# Transformer Network : Decodeur



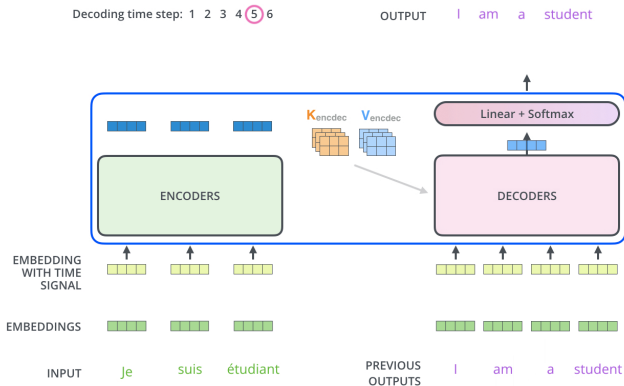
# Transformer Network : Decodeur



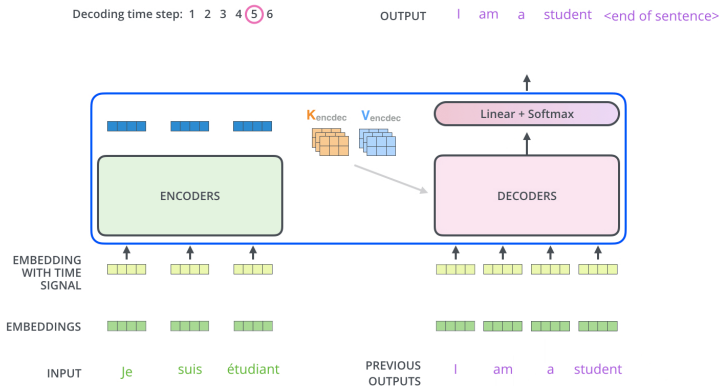
# Transformer Network : Decodeur



# Transformer Network : Decodeur



# Transformer Network : Decodeur



Merci beaucoup à lui pour les illustrations de ce support

<http://jalammar.github.io/illustrated-transformer/>