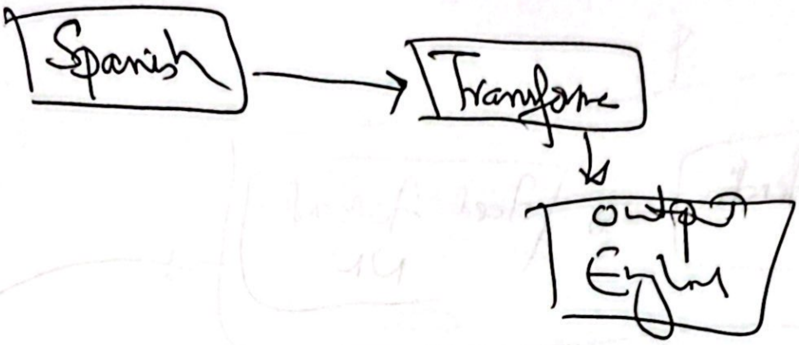
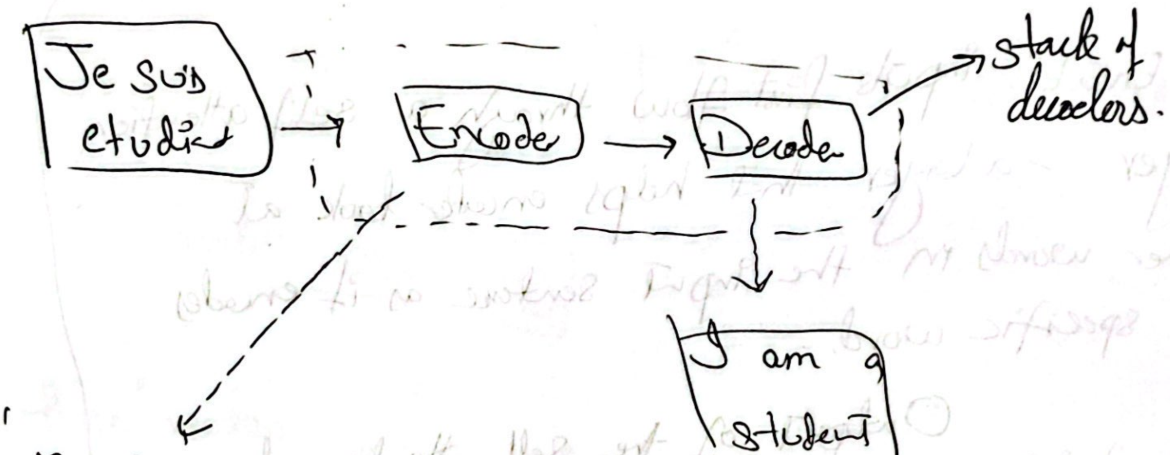
Intuition 1

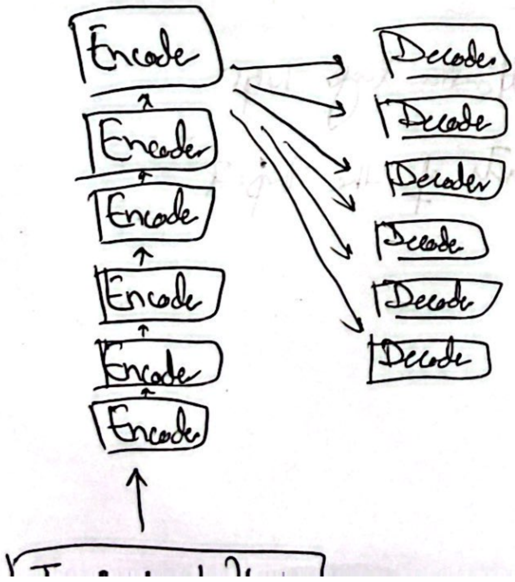
IMPORTANT



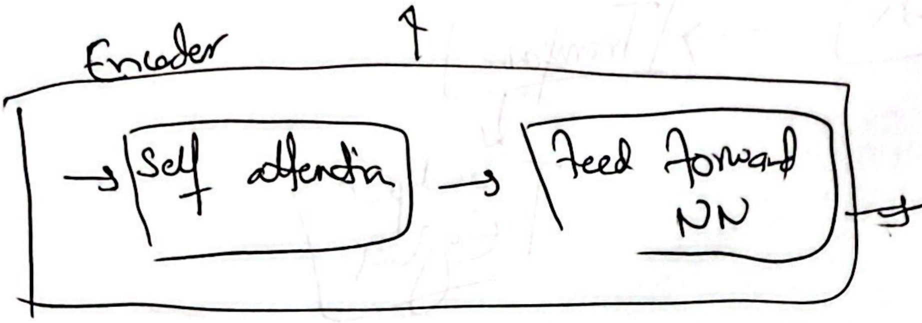


Encoder
↳ is actually a stack of encoders, no. 6
↳ hyperparameter
→ encoder: 6
→ decoder: 6

output of final encoder will be input to all decoders.



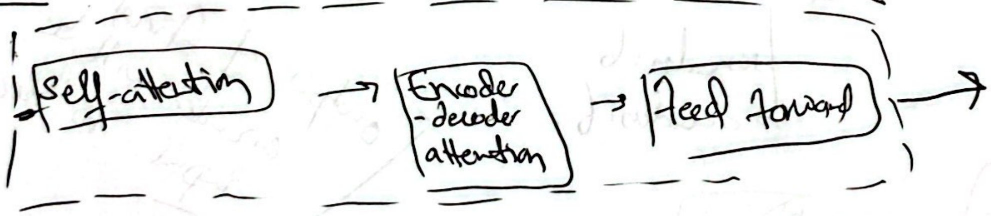
Inside Each encoder



Encoder's inputs first flow through a self attention layer - a layer that helps encoder look at other words in the input sentence as it encodes a specific word.

Output of the self-attention layer are fed to feed forward NN. The exact Same feed forward ~~Net~~ NN is applied to each position

Decoder



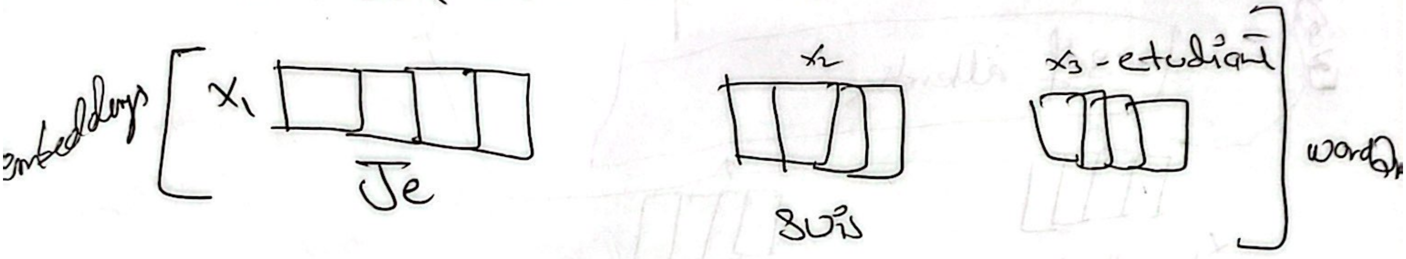
↳ this second attention layer helps the decoder focus on relevant parts of the input sentence.

Attention

Attention v/s SELF ATTENTION-

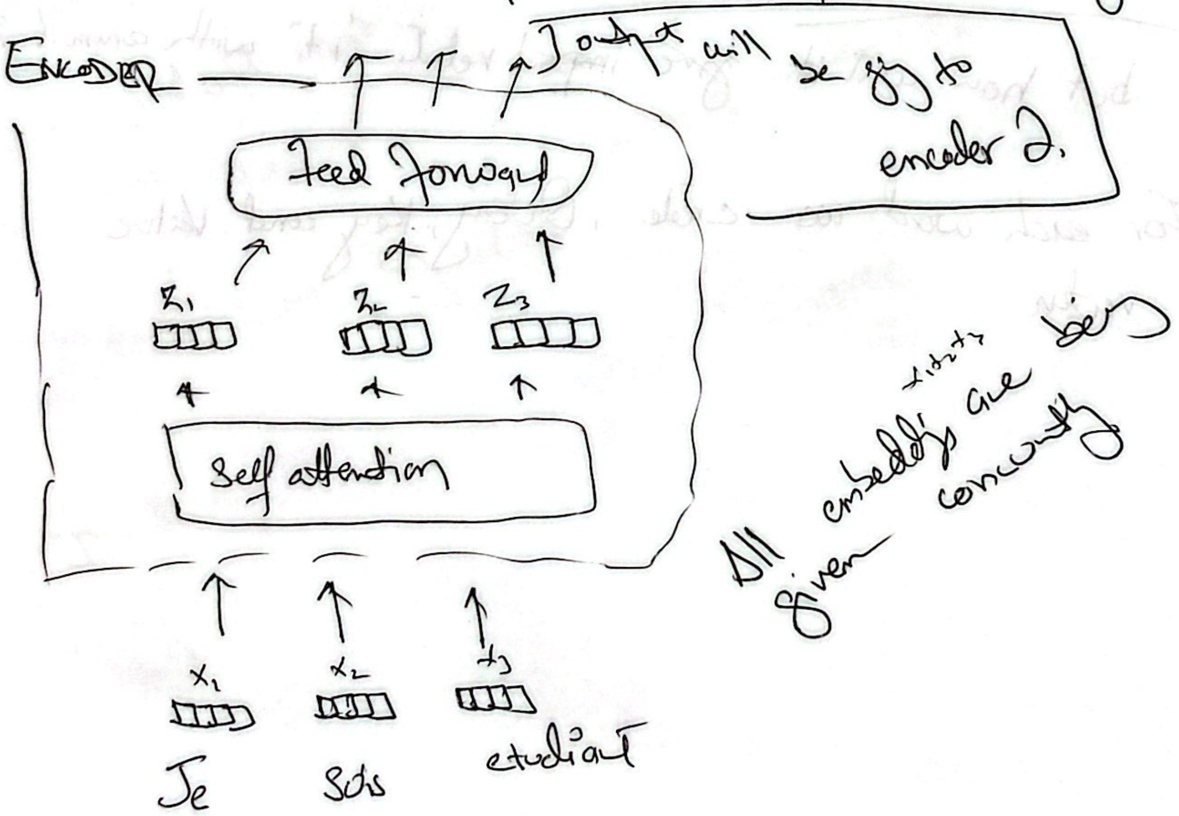
input: Je Suis etudiant

turn each word into vector.

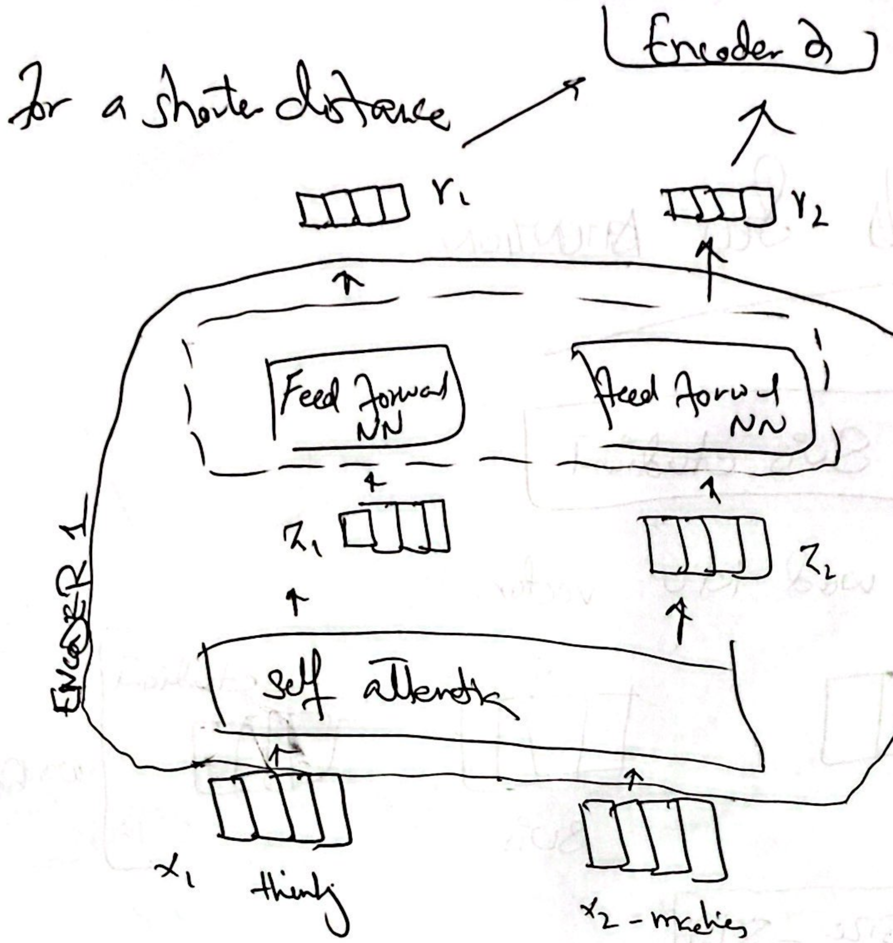


vector size - 512

these vectors are passed to self attention layer



ENCODER
output will be gg to encoder 2.
feed forward
z1
z2
z3
self attention
x1
x2
x3
Je
Suis
etudiant
All x1,x2,x3 embeddings are being given concurrently

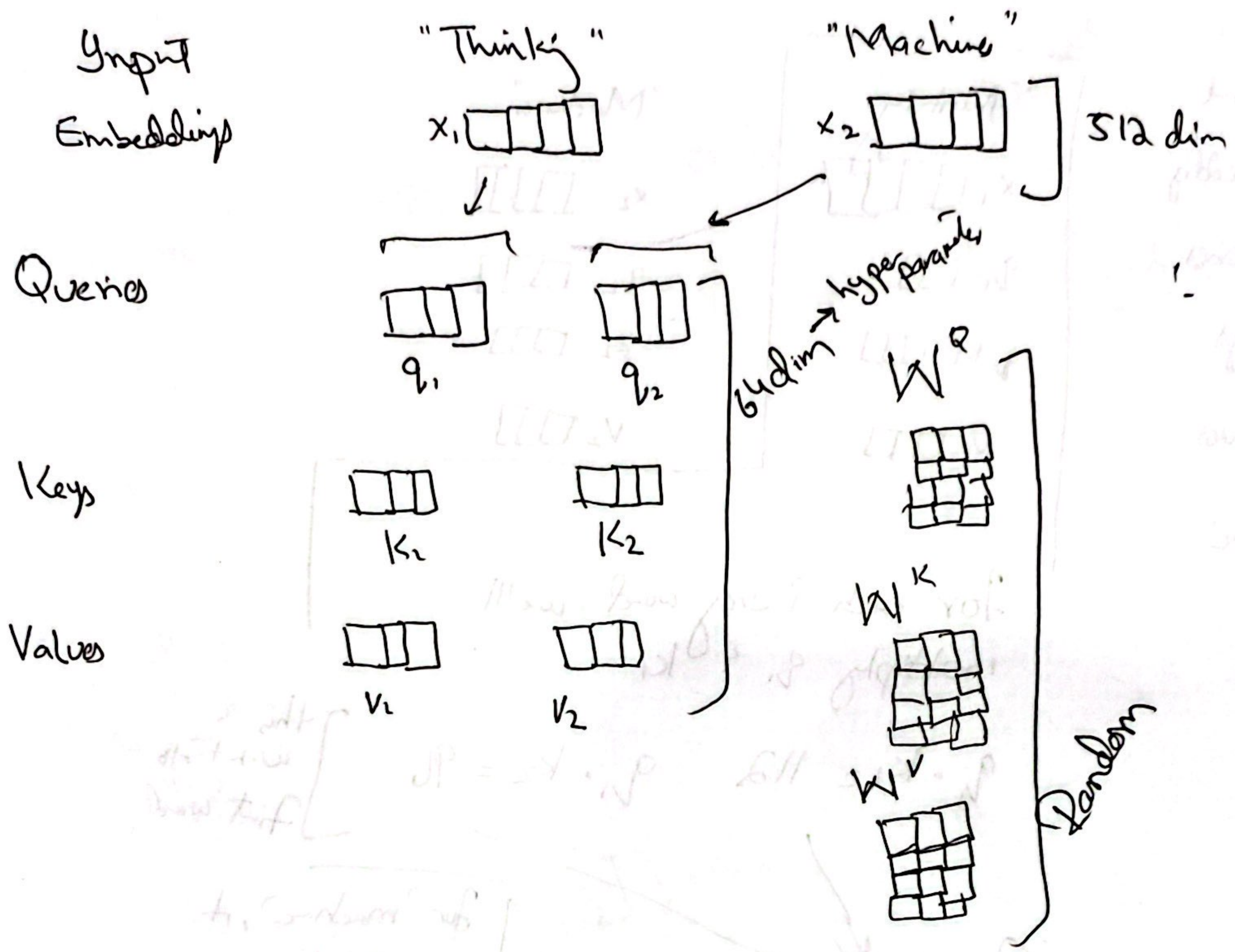


Self-attention,

but how will it ~~give~~ ~~impo~~ relate "it" with "animal"?

For each word we create, Query, Key and Value vector

Attention



W^Q, W^K, W^V are 3 weights we randomly initilized. It will be updated based on back propogation.

So, first

$q_1 = x_1 \cdot W^Q$ $\quad$ $q_2 = x_2 \cdot W^Q$] 64-d

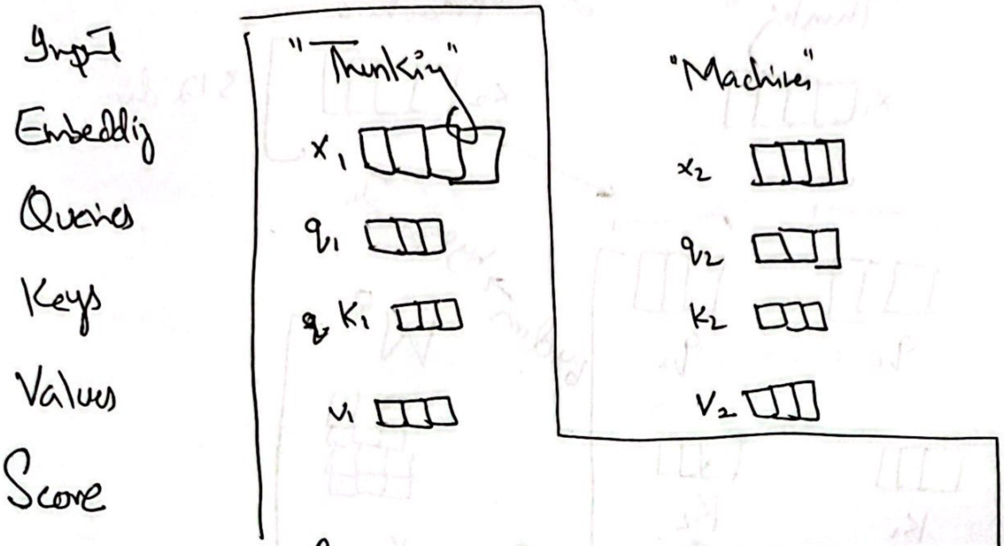
why multiplied? input multiplied by weight + bias $Wx+b$ this has activation this is how ANN operate.

Same for W^K & W^V

$k_1 = x_1 \cdot W^K$

$v_1 = x_1 \cdot W^V$

So,



for each & every word, we'll multiply q_1 & k_1

$q_1 \cdot k_1 = 112$ $q_1 \cdot k_2 = 96$] this is w.r.t to first word.

Score

for 'machine', it would be

$q_2 \cdot k_1 =$

$q_2 \cdot k_2 =$

We need to find out most related words.

Divide score by $\sqrt{64}$ — dim of q_1, ~~k~~ : d_k

Divide by $\sqrt{64}$ ($\sqrt{d_k}$) = ~~0.88~~ 112/8 ~~0.12~~ 96/8

softmax = 0.88 "thinking" 0.12 'machines'] to get prob

Attention ~~[illegible]~~ ↳

Softmax 0.88 0.12

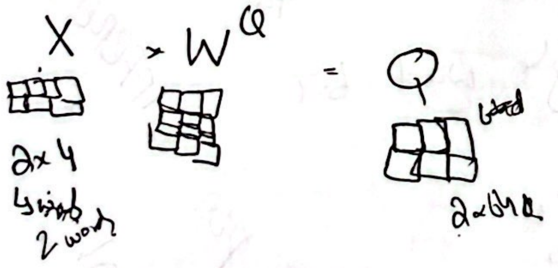
Softmax × value $V_1 \times$ softmax $V_2 \times$ softmax] to keep relevant words only → to ~~[illegible]~~ give low prob to less relevant word

↳ multiply by v_1 & v_2

Sum : Z_1 □□□□ = $V_1 + V_2$

↳ output of "thinking".

Matrix multipl



No window size because all words are being passed parallelly.

$X \times W^k = K$

$X \times W^V = V$

Condensing

$$\text{softmax}\left(\frac{Q \times K^{T}}{\sqrt{d_k}}\right) V = Z$$

transposed

Problem?

↳ We are using one W^Q, one W^K, one W^V for all words. This is single head attention.

Same W^K, W^Q, W^V for all words.

but problem lies when there multi might be important connection.

So, we might use multiple weights.

MULTI-HEAD Attention

↳ to find importance of other words too

-mean not only finding single word but multiple relevent words

Attention head 0 [Q_0 ~~W_0Q~~, W_0^Q

K_0 W_0^K

V_0 W_0^V

head 1 [Q_1 W_1^Q

K_1 $W_1 K$

V_1 W_1^V

Different HEADS FOR DIFFERENT WORDS

Attentn 5

eight heads

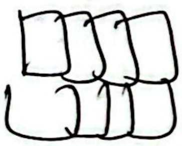
↳ eight difft ~~mat~~ weight matrices.

will give 8 z matrices.

So now

X

"thinking machine"



→

Attentn head 0		z0 (3× 2
head 1		z1
head 2		z2
3	⇒	z3
4		z4
5		z5
6		z6
7		z7

But we are supposed to get 1 z value not 8.

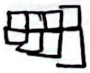
1- we'll concatenate all zs

z0	z1	z2	z3	z4	z5	z	z7

2- initialize another W° matrix → would be updated in back propagatn

W° will be multiplied with concatenated z

result would be $\boxed{Z}$



→ POSITIONAL ENCODING ~~before~~ after converting words into embeddings

↳ to preserve order.

INPUT	Je	Suis	ETUDIANT
Embeddings	x_1	x_2	x_3
	+	+	+
	Positional encoding t_1	t_2	t_3
Embedding with time signal	X_1	x_2	x_3

t_1 would be nearer to $\underline{t_2}$ than $\underline{t_3}$.

This happen before passing to encoder

Attn #5

In encoder, befor passig to feed forward ~~Each encoder~~ and after self attn, we'll be ~~perf~~ applying normalizati. and a residual connecti.

↳ like in RNN, you are skipping one step.

~~if~~ i.e if self attn is not rqd, you are skipping it through residual networks

In add & Normalizel

↳ $$\text{layer norm}\left(\underset{\substack{\downarrow\\ \text{input}}}{X} + \underset{\substack{\downarrow\\ \text{encoder output}\\ \text{+64)}}}{Z}\right)$$

In Decoder

Encoder Decoder Attentn

↳ Same only output of encoder

↳ passed into the layer of decoder

input - all words at once

output - one word at a time.

↳ after each word predicts the whole output will be given input to decoder. It won't stop until <EOS>.