Assumption

You know Encoder Decoder

& Attention

course videos on DL → DL case studies in covid → Encoder Decoder → Attention → Transformers → BERT

Chain for understanding

Jalammar.github.io

↳ Illustrated Transformer → better explanation than anything out there

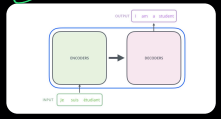
Transformer

↓

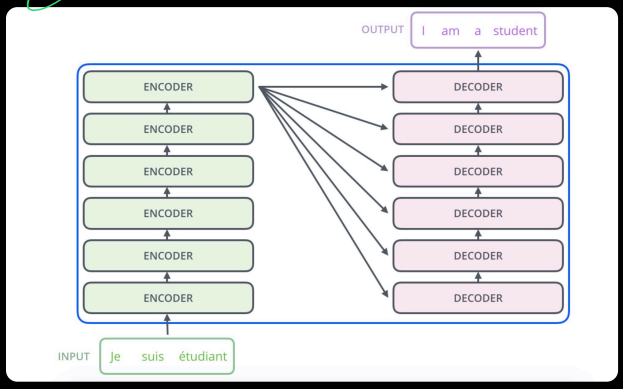
advanced seq 2 seq model

(1)

(2)



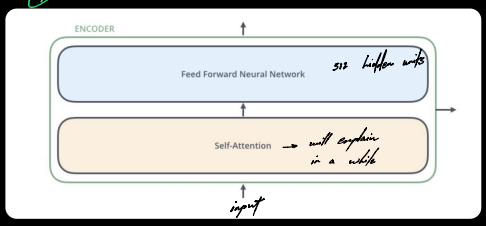
3



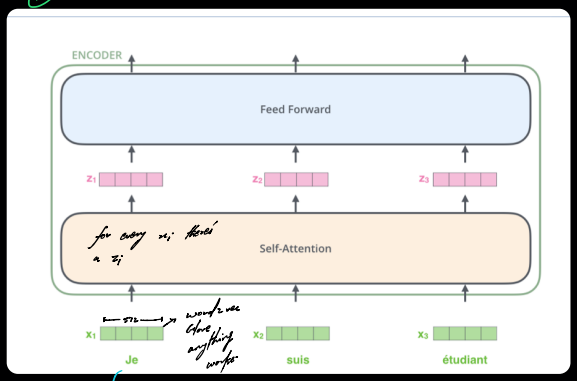
why only 6 encoders & decoders? Anything works but they might have done some hyperparameter tunings.

they are identical to each other.

4

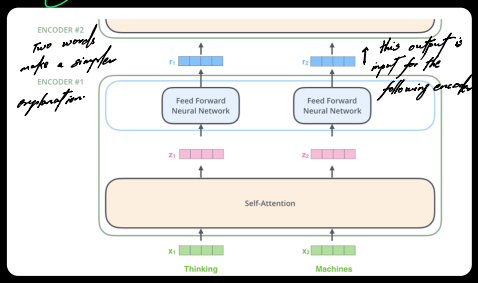


5



all inputs are taken at once
we try to work around the limitations of RNN

6



#Self Attention:

"The animal didn't cross the street because it was too tired"

who the fuck is "it"?

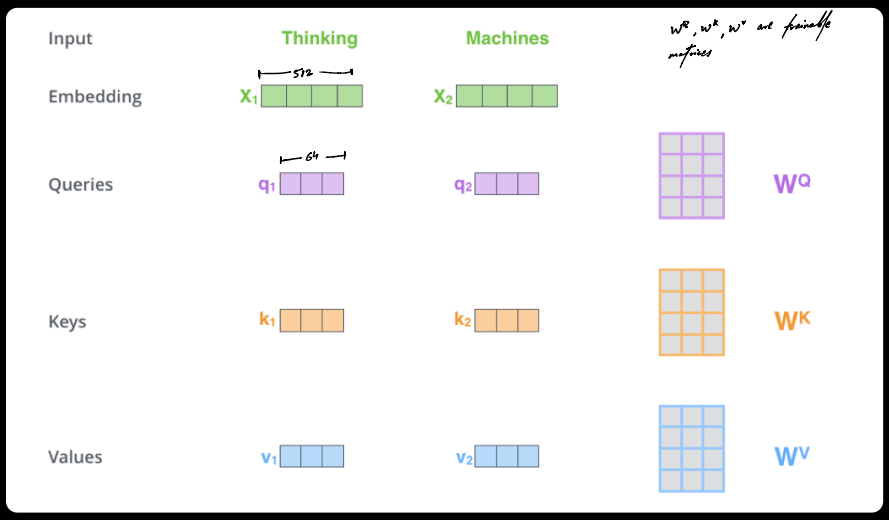
who the fuck was tired? animal or road?

we want our model to focus on "animal" when processing "it" & word and also on "road"

It's called self attention because we're trying to which part of ourself to focus on.

$x_i \longrightarrow z_i$

some words need more attention.



W^Q, W^K, W^V are trainable matrices

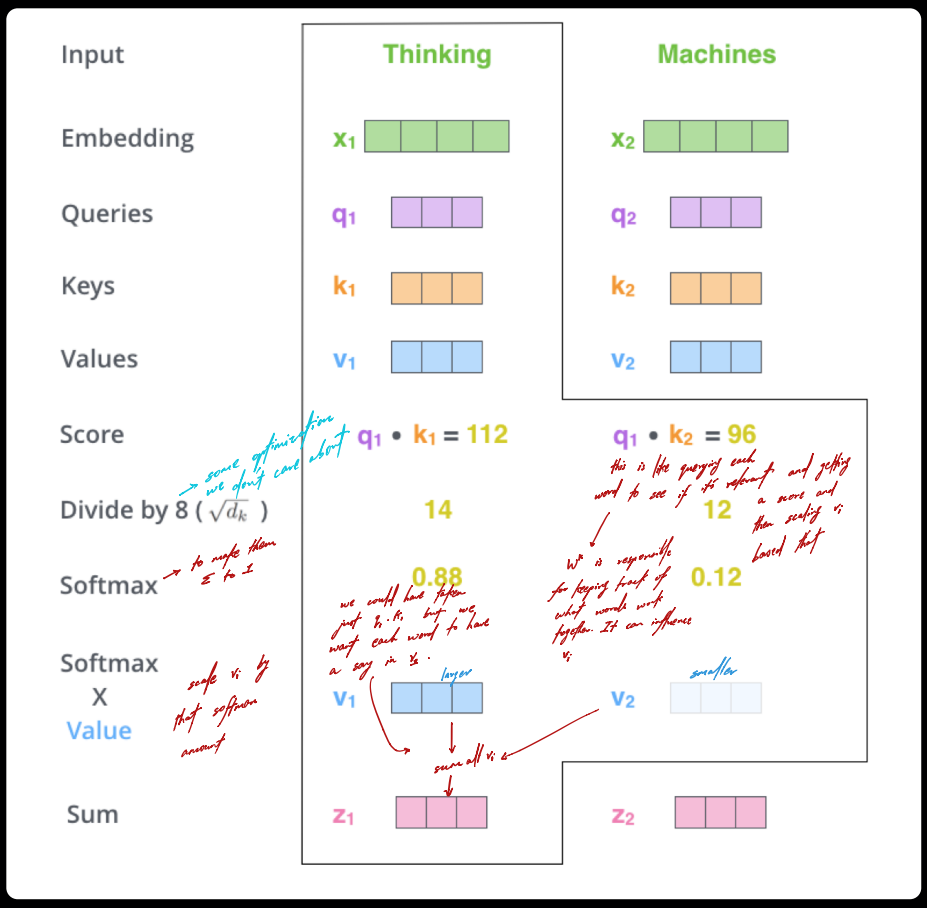
$q_i = x_i W^Q$ → this is like simple NN without activations.

$k_i = x_i W^K$

$v_i = x_i W^V$

↓

just take x_i & get 3 products of it q_i, k_i & v_i



→ some optimization we don't care about

to make them Σ to 1

this is like gauging each word to see if it's relevant and getting a score and then scaling v_i based that

W^K is responsible for keeping track of what words work together. It can influence v_i

we would have taken just $q_i \cdot k_i$ but we want each word to have a say in it.

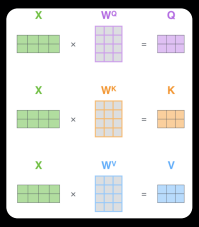
larger

smaller

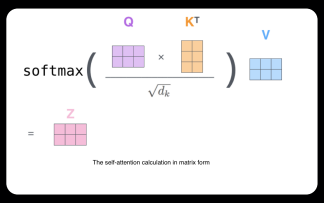
scale v_i by that softmax amount

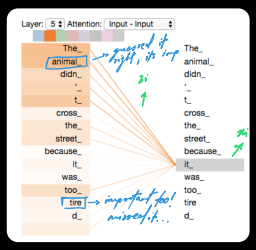
sum all v_i s

we going to use q_i k_i values to scale v_i somehow.



All this summarized in a bunch of matrix multiplications.

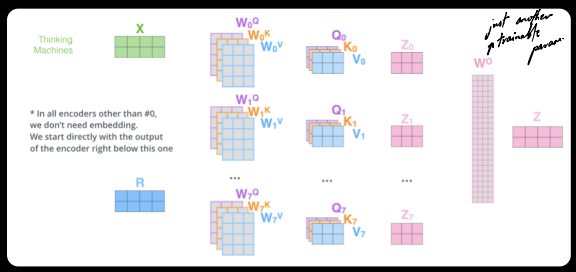


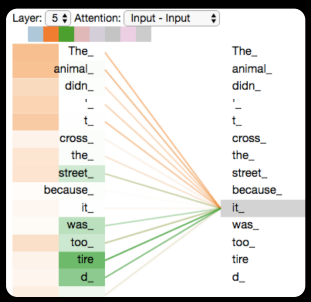


They say we should have multiple Q, K, V matrices. → called attention head can focus on only one word.

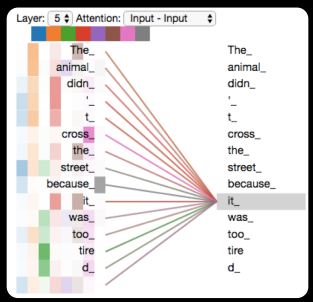
have 8 attention heads → can focus on multiple words effectively

so for each x_i we have 8 z_i vectors.



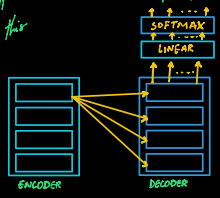


This is what happens for 2 attention heads.



This is what happens with multiple attention heads.

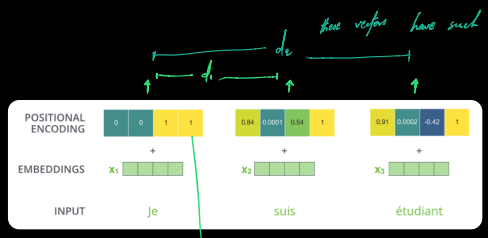
Dataset = { E_1 , F_1 (English, French)
E_2 , F_2
E_3 , F_3
⋮
E_n , F_n }

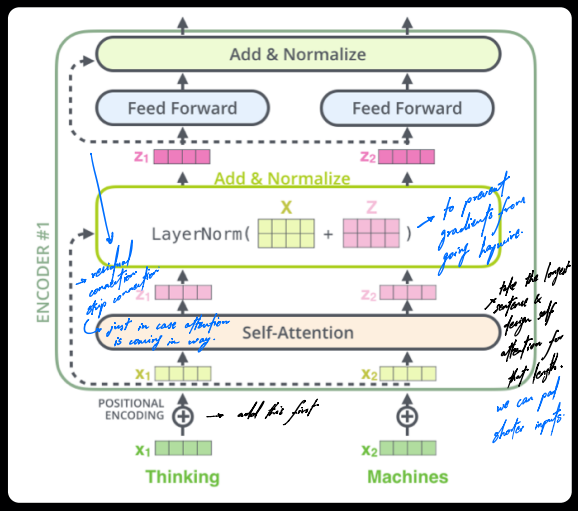


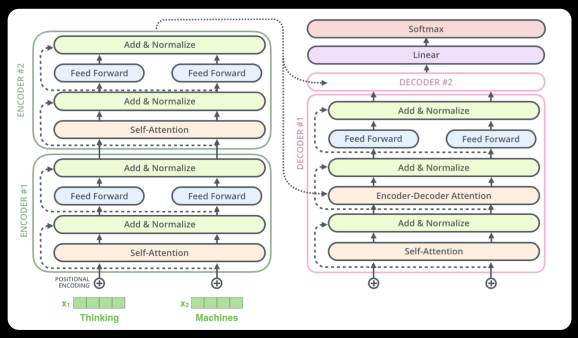
final result will be like this

calculate loss & backpropagate

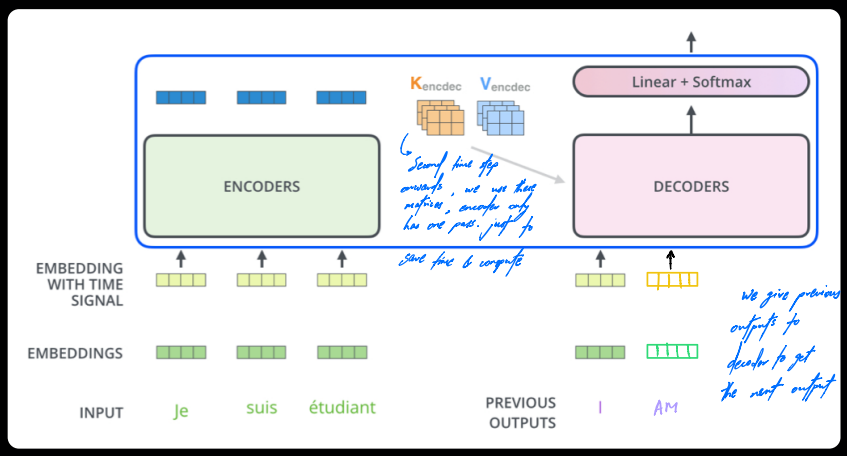
Attention happens implicitly.







→ this is just like self attention but inputs for this layer come from encoder outputs



① For input all of them are given at once, no timesteps.

② Output is softmax so only a single output.

③ for getting the second output we just gave first output back to decoder along with same input

Loss function:
categorical crossentropy

Key takeaways

① Significantly faster than RNN → because all inputs are taken at once.

② Just like CNN → it learns what part of input is important → on its own.

③ State of the art → BERT is the simpler version of this architecture.
this is what people translate uses

Sidekick Sr

"If in an interview you say you know Transformers, I'll ask you all of this"

#BERT Bidirectional Encoder Representation from Transformers

As long as you can frame a problem as text 2 text, BERT can be applied.
↳ text summarization
classification
question answer.

BERT is much simpler than transformer, but provides almost the same performance.

There's no decoder.

Transfer learning for NLP, that's what BERT is

BERT just gives text to vector.
& we can do whatever the task we want with the vector.

12 encoders ← $BERT_{base}$
24 encoders ← $BERT_{large}$ → surpasses human performance in many NLP tasks.

they call it "Transformer Blocks" in the paper.

BERT was the ImageNet moment of NLP community.

$BERT_{large}$
↳ 24 encoder blocks
1024 units in Feedforward layer
16 attention heads

$BERT_{base}$
↳ 12 encoder blocks
768 units in Feedforward layer → that'll be size of output
12 attention heads

It's bidirectional model

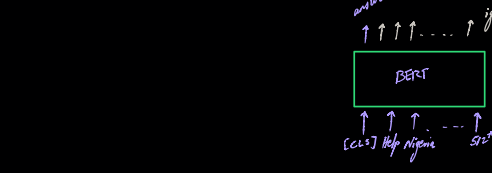
so words before & after are taken into account

BERT uses context & that helps with all those benchmarks.

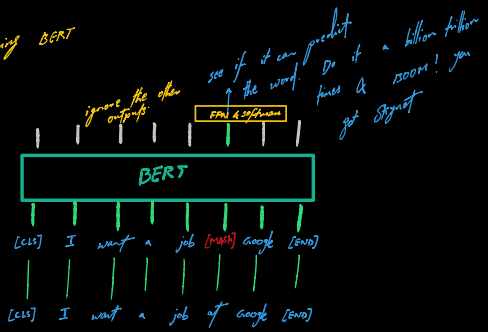
Best application
↳ transfer learning.

Example
Spam Example.

BERT can take upto 512 word sentences



Training BERT

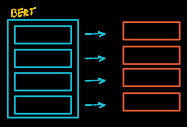


[CLS] → we want it to classify

[SEP] → when joining sentences in input

BERT is actually just some feature extraction system.

For feature extraction:



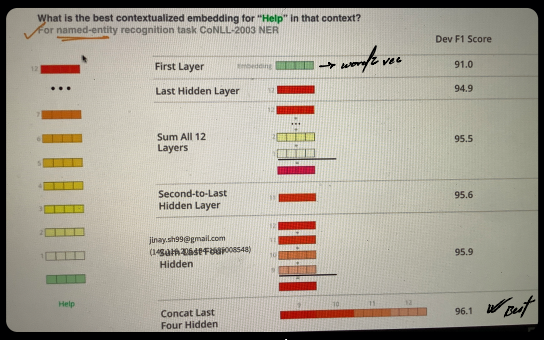
each layer will have an output. which one to use?

Coding:

DistilBERT → it's a feature extractor.

↓

Then we can just use a logistic regression or any other model



↓ it depends on the application.