Deep Learning Theory and Applications

Transformers

Yale

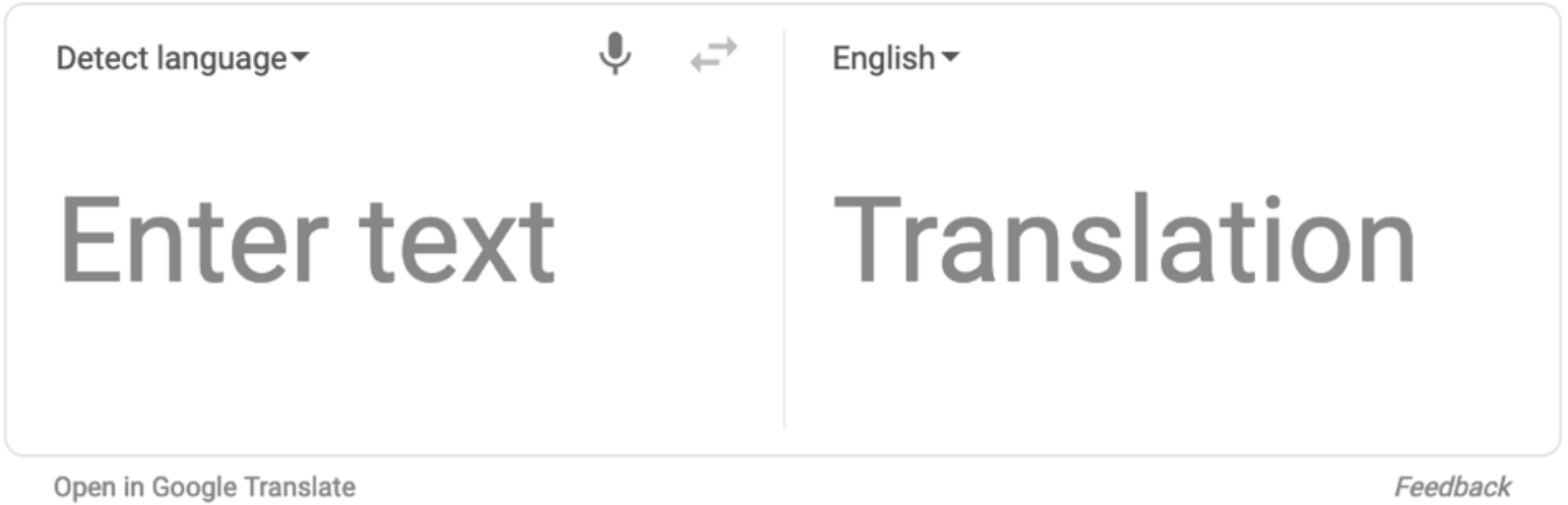
CPSC/AMTH 452

CBB 663

Outline

- RNN review
- Sequence transduction tasks
- Attention
- Self Attention
- Multi-head Attention
- Positional Encoding
- Transformer Architecture
- GPT2/3

Machine Translation



Statistical machine translation:

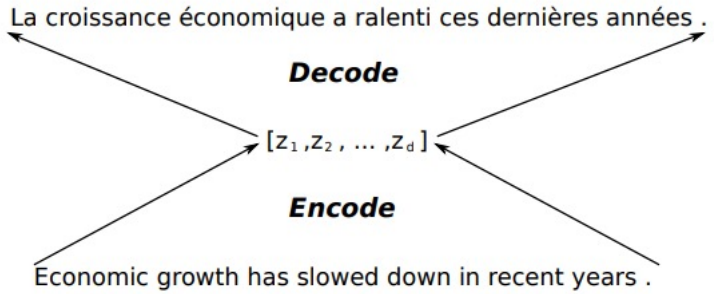
P(e|f) e = english phrase f= foreign phrase

$p(e|f) \propto p(f|e)p(e)$ language model $p(e)$

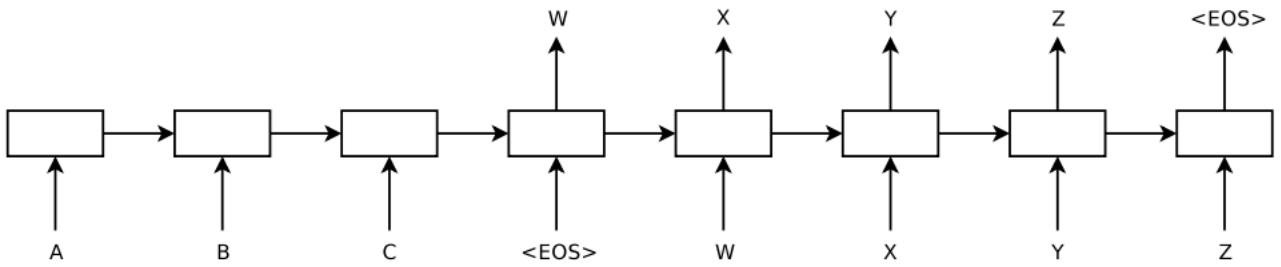
Translates n-grams phrases consisting of n-words

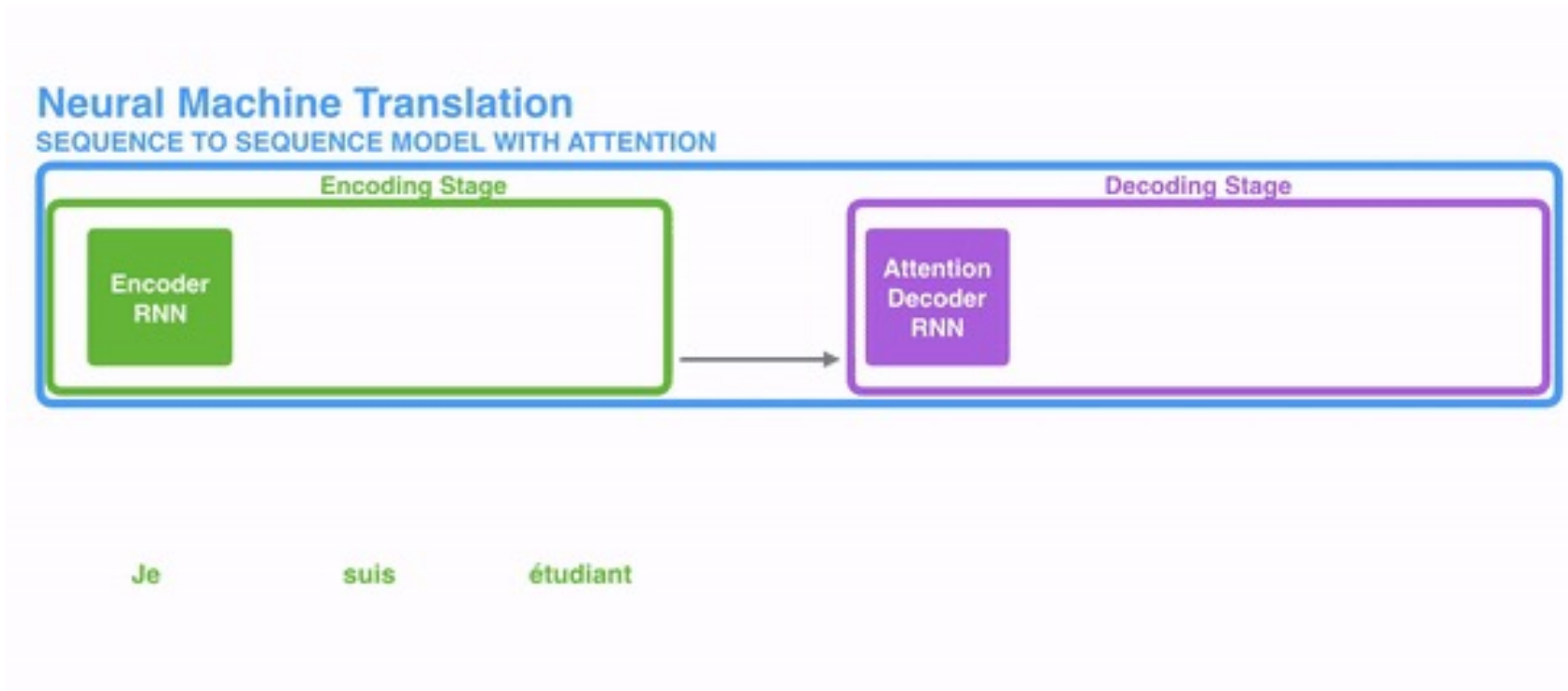
Neural Machine translation

Encoder-decoder architecture

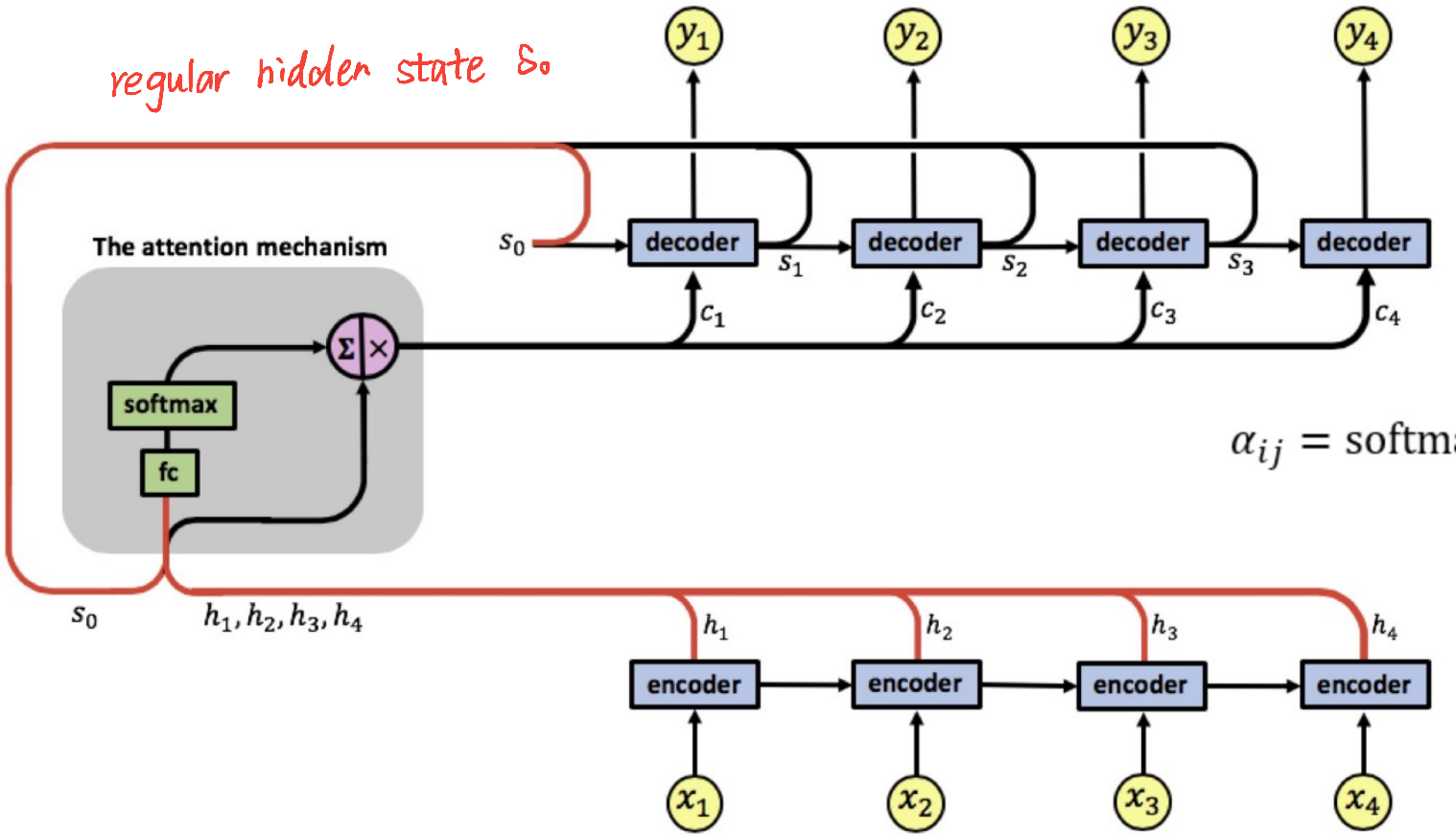


Cho et al 2014, Sutskever 2014




Neural Machine Translation
SEQUENCE TO SEQUENCE MODEL WITH ATTENTION
Encoding Stage
Decoding Stage
Encoder RNN
Attention Decoder RNN
Je
suis
étudiant

Attention: Adaptively Allows Decoder to Choose Inputs

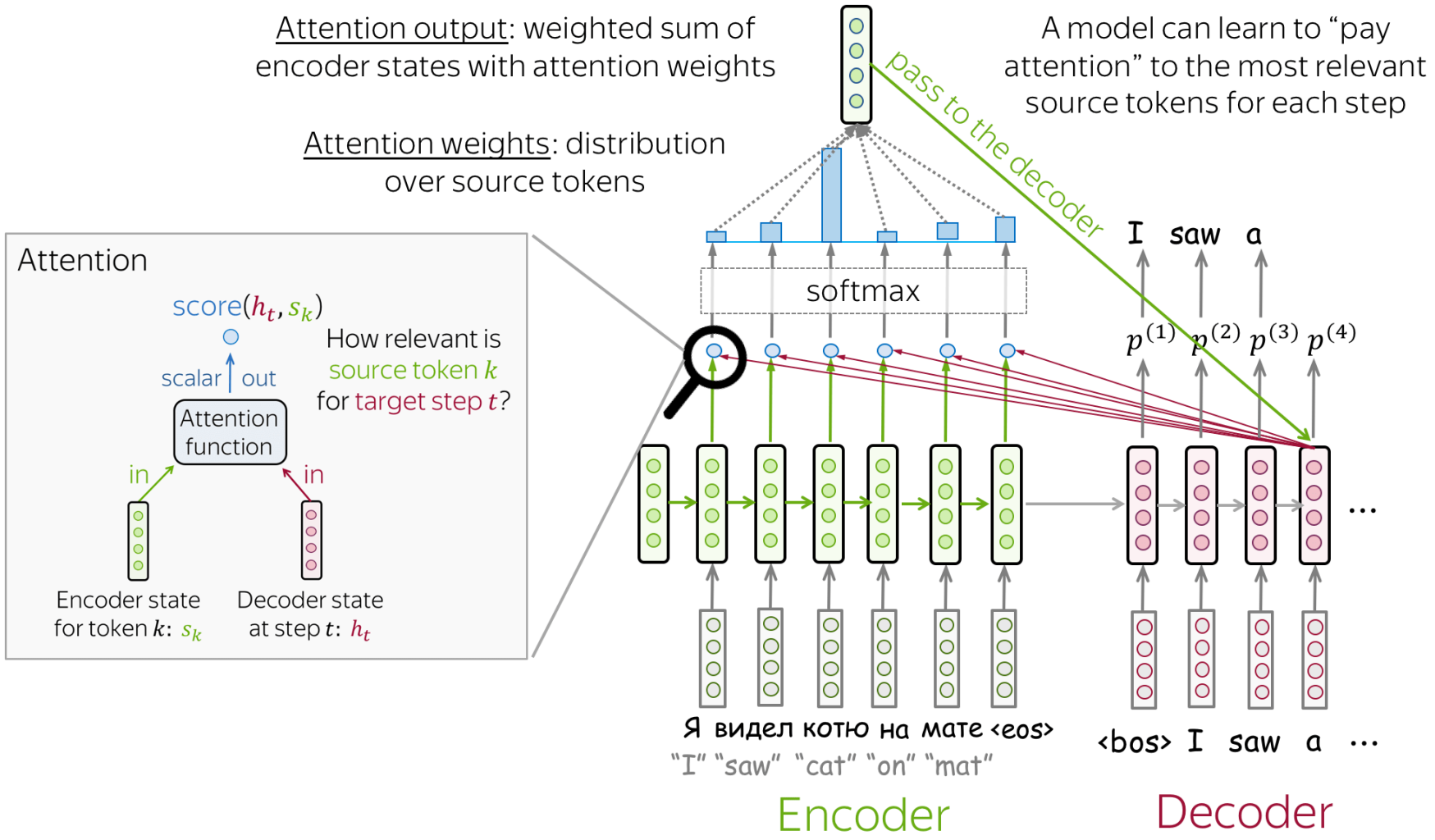


$$\alpha_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

context vector c_i

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

Attention Mechanism



Attention Computations

Attention output

$$c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$$

"source context for decoder step t"

(weighted sum)

Attention weights

$$a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \text{k} = 1..\text{m}$$

"attention weight for source token k at decoder step t"

(softmax)

↖ attention scores normalized by softmax

Attention scores

$$\text{score}(h_t, s_k), \text{k} = 1..\text{m}$$

score could be result of a neural net layer or just take dot product

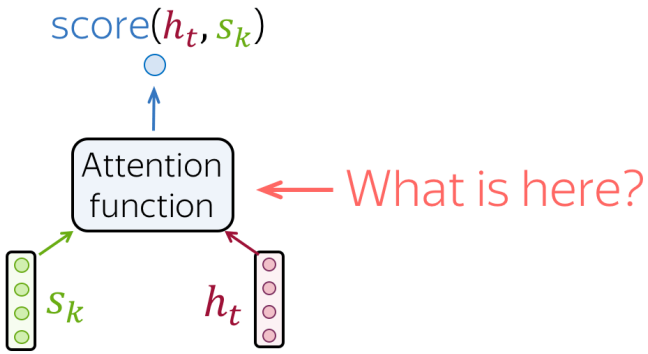
"How relevant is source token k for target step t?"

Attention input

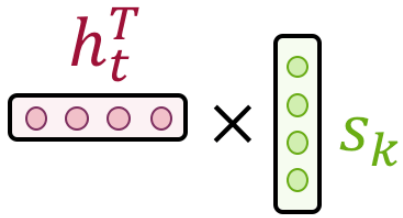
$s_1, s_2, \ldots, s_m$ all encoder states

h_t one decoder state

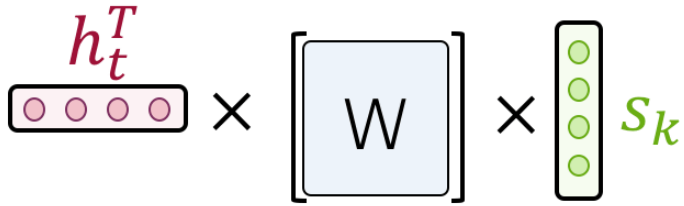
Attention Score



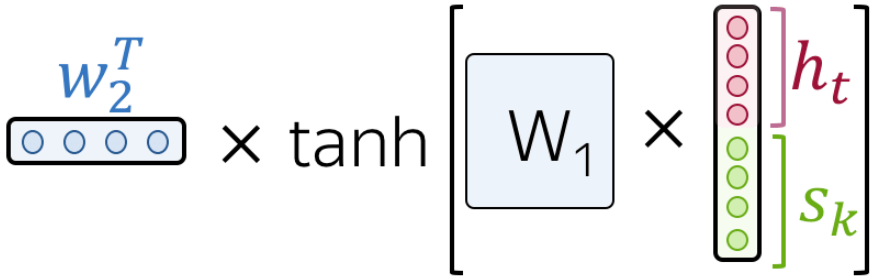
Dot-product



Bilinear



Multi-Layer Perceptron



$\text{score}(h_t, s_k) = h_t^T \; s_k$

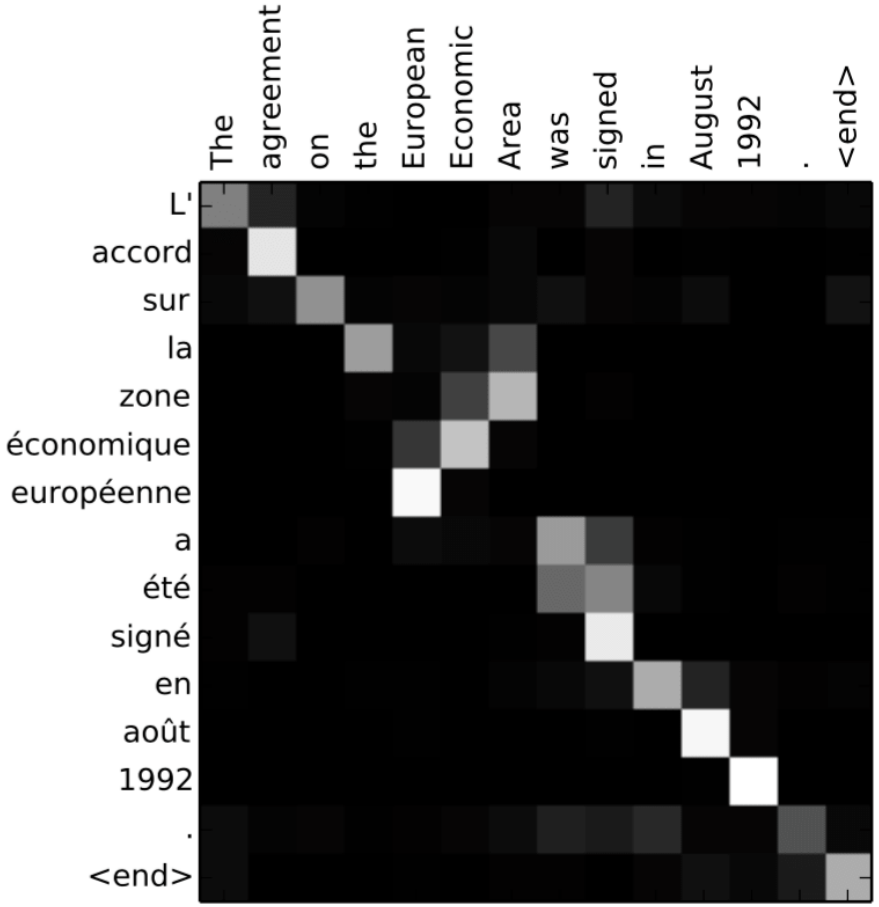
$\text{score}(h_t, s_k) = h_t^T \; W \, s_k$

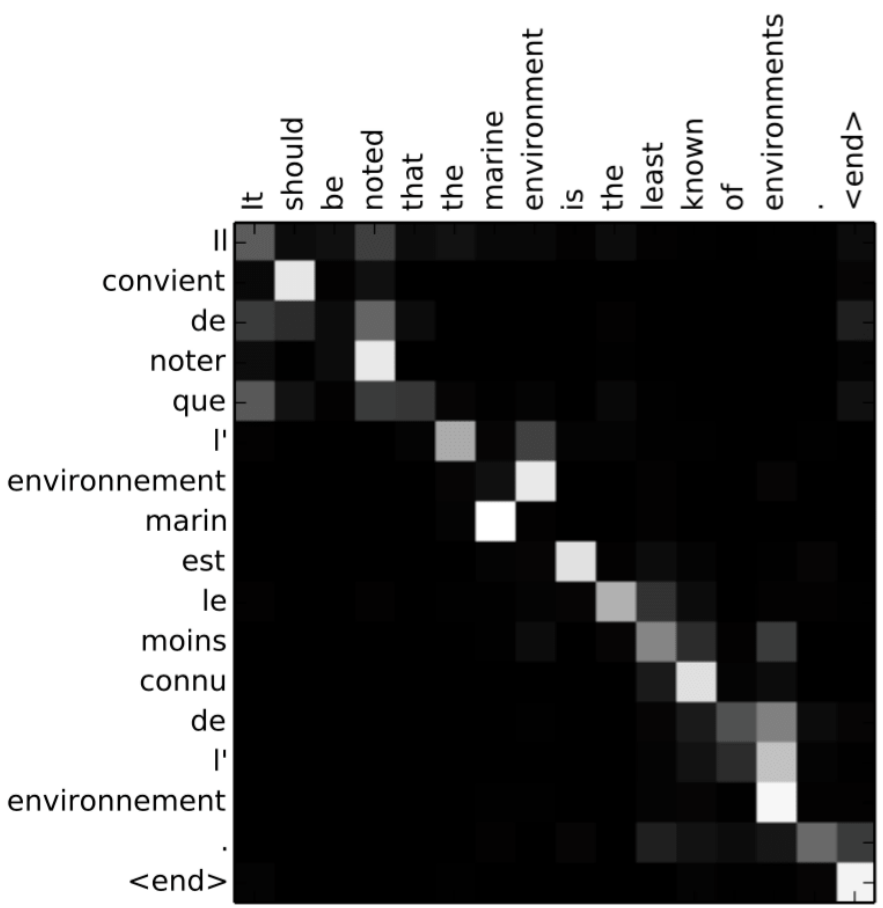
$\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1 [h_t, s_k])$

Attention Weights

can highlight different parts of input for the output

Interpreting Attention

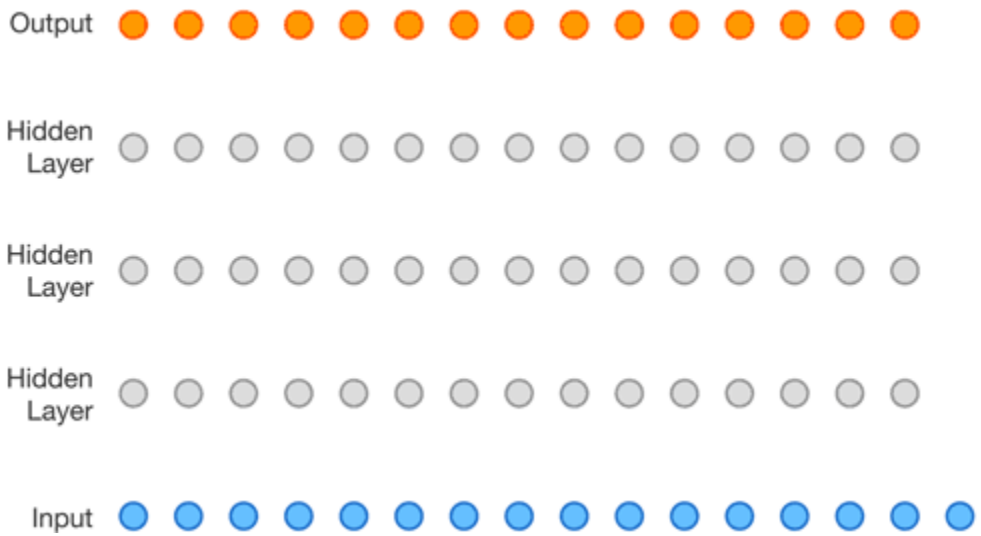




CNNs for sequences

Transformer use attention but limit it for recurrent

in a sense like CNN

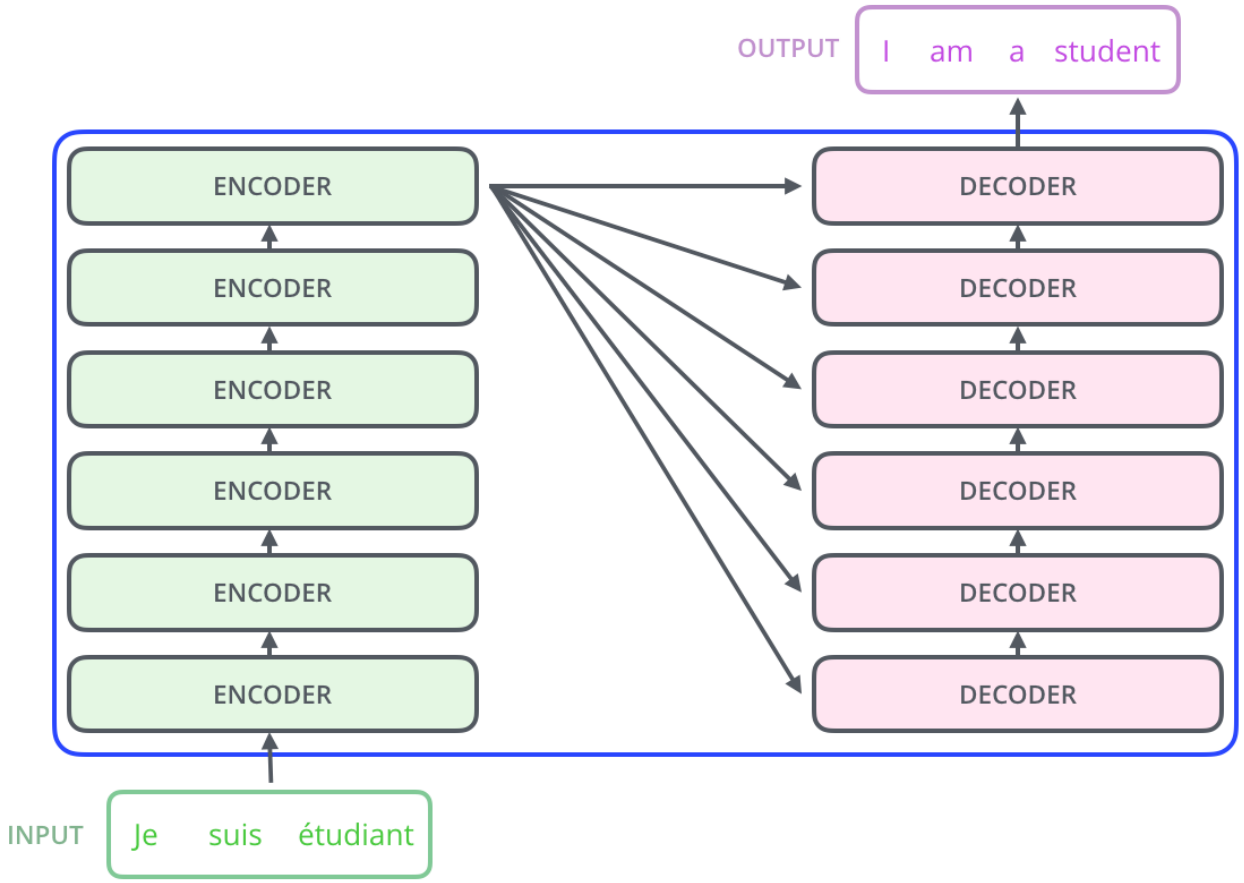


1D convolution

- Easy to parallelize
- Can exploit dependences
- Distance between positions is logarithmic

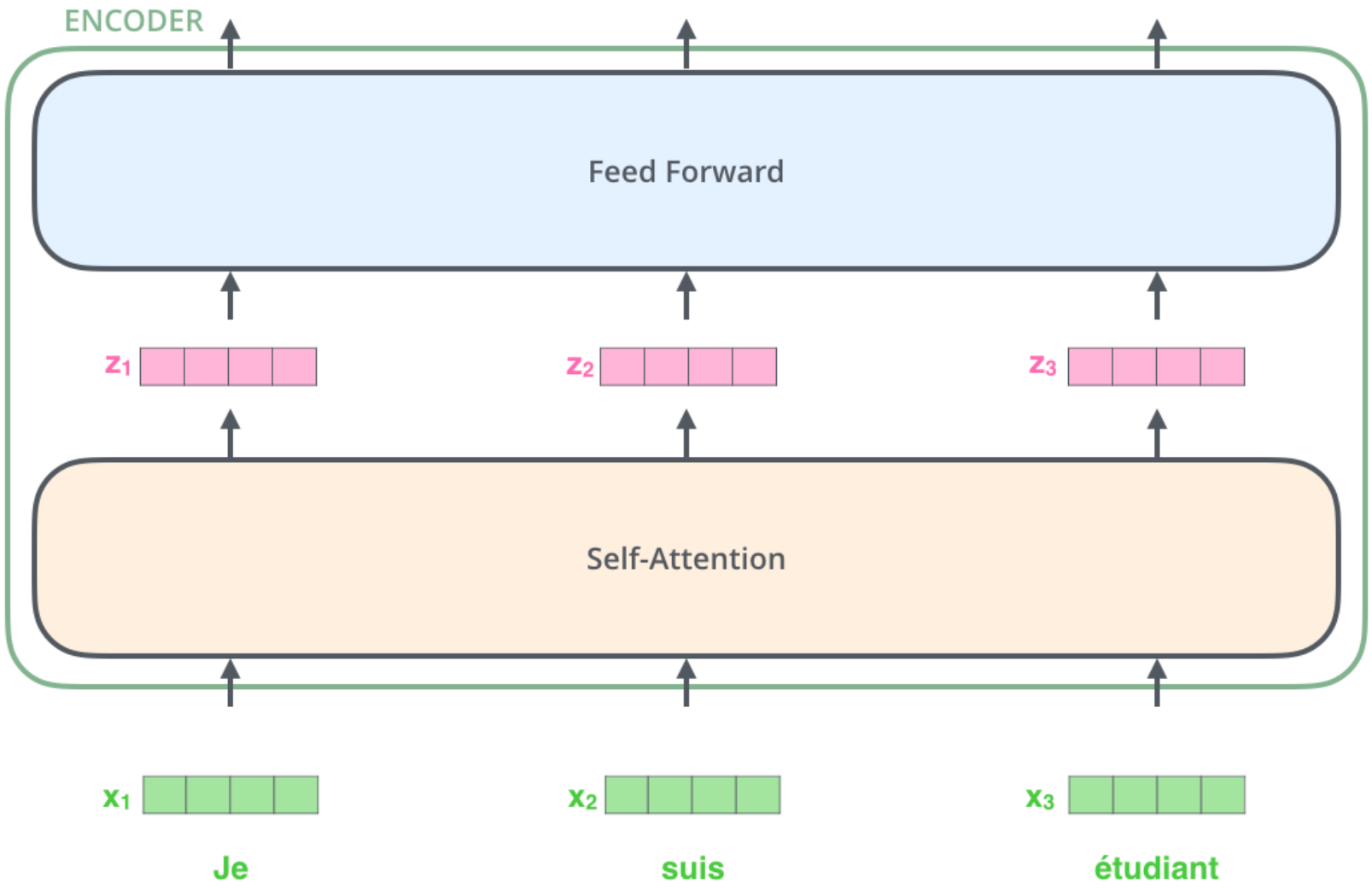
Transformer: CNN + Attention

orginal transformer has 6 encoders and 6 decoders



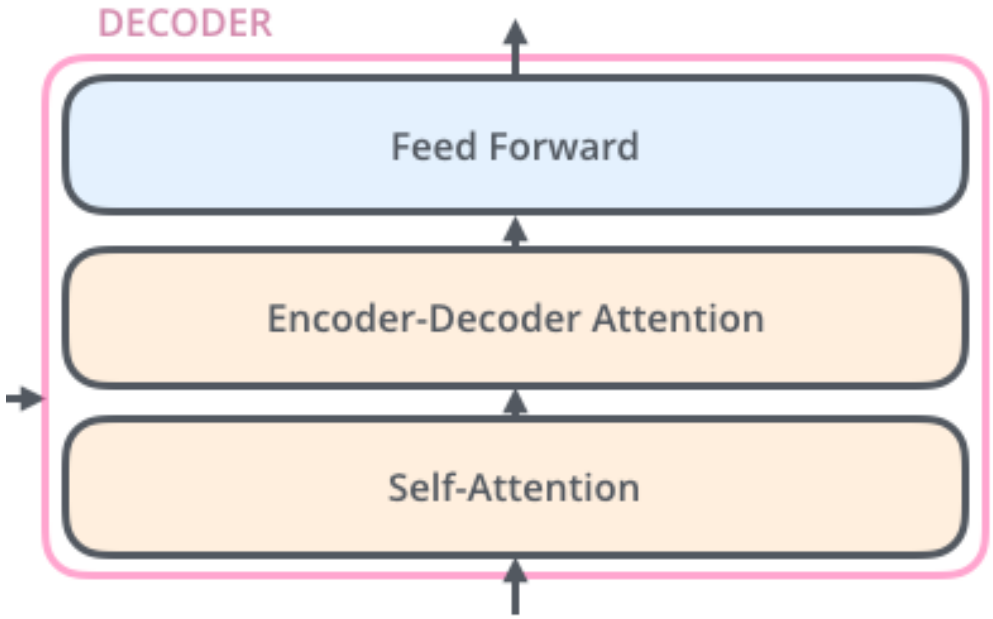
one layer of self-attention and one layer of normal feed forward

Each Encoder has Self-Attention



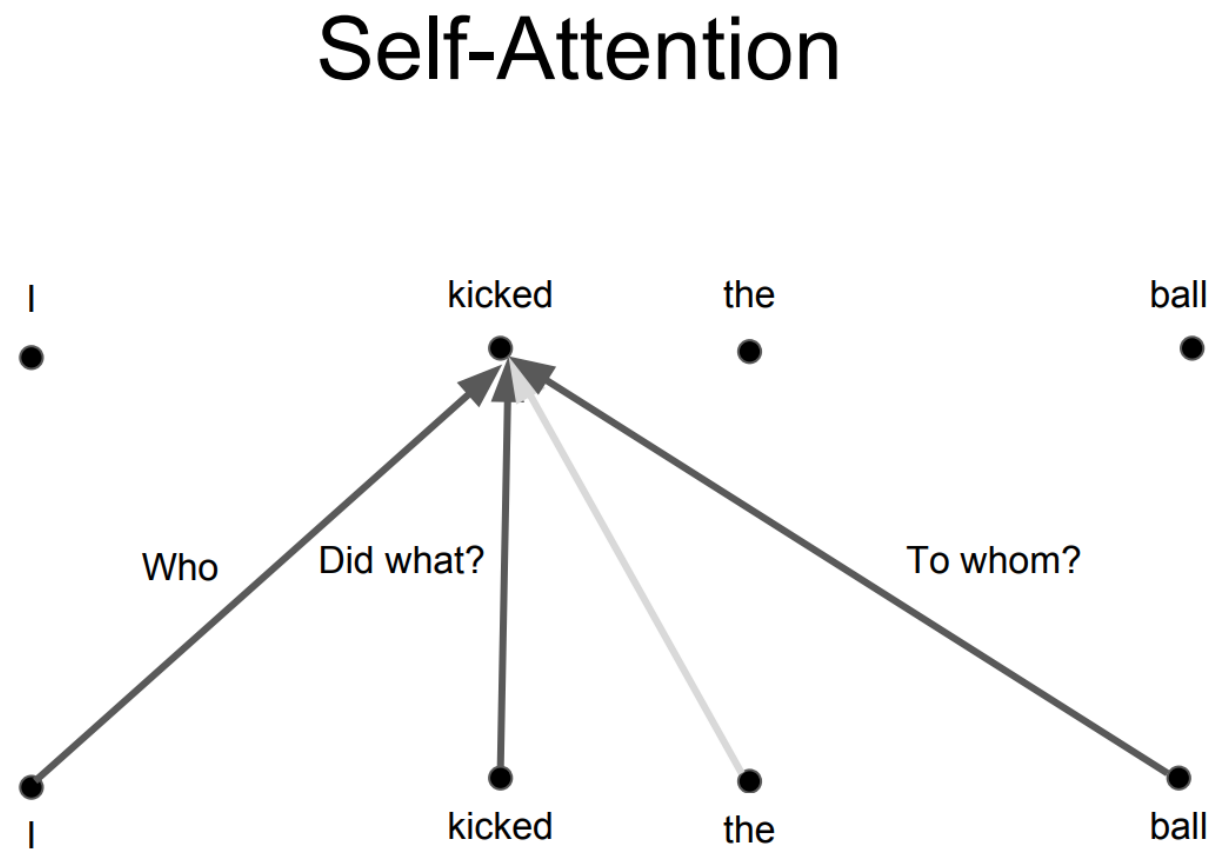
Decoder

has 3 kinds of layer

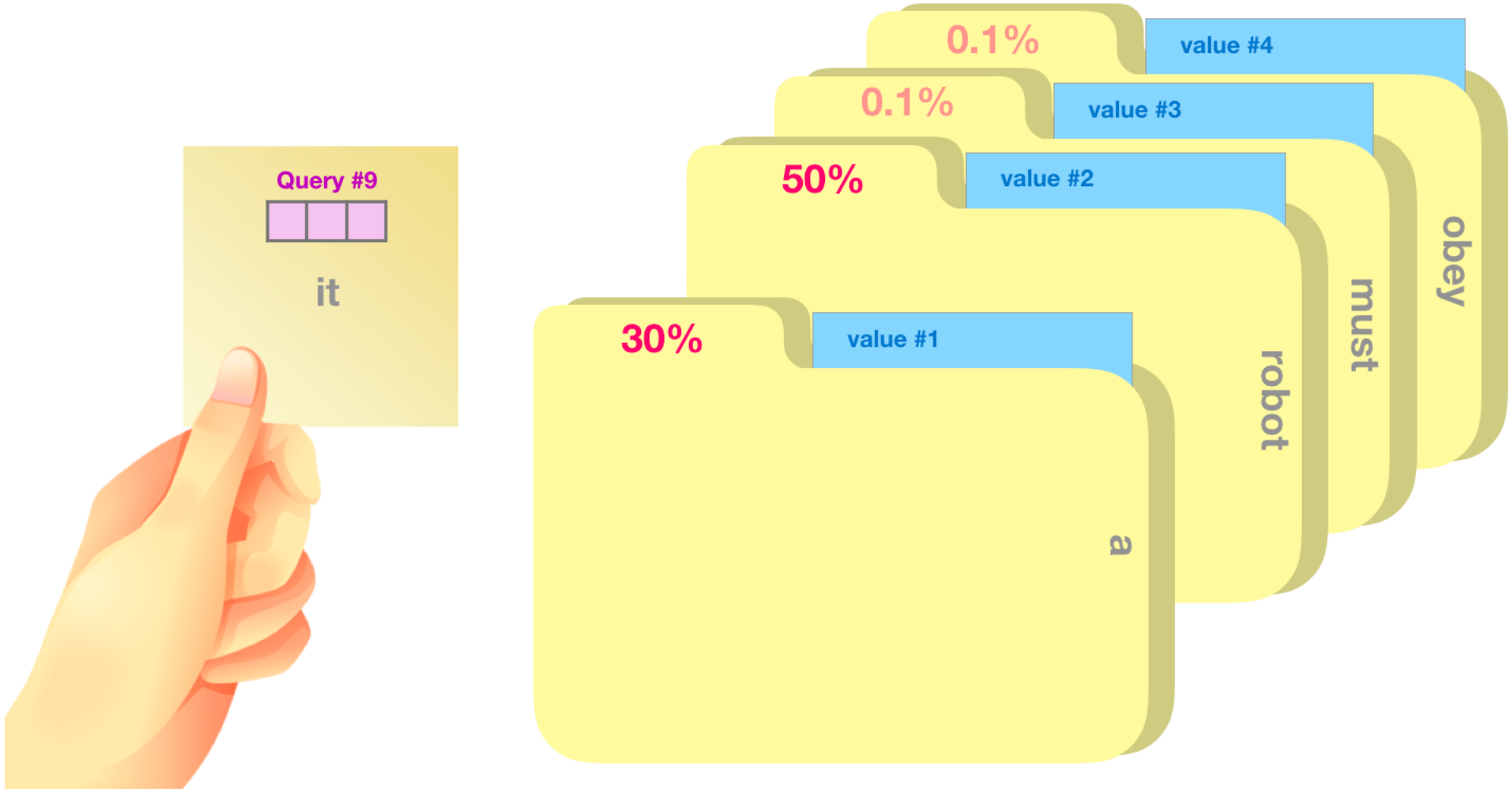


Self Attention

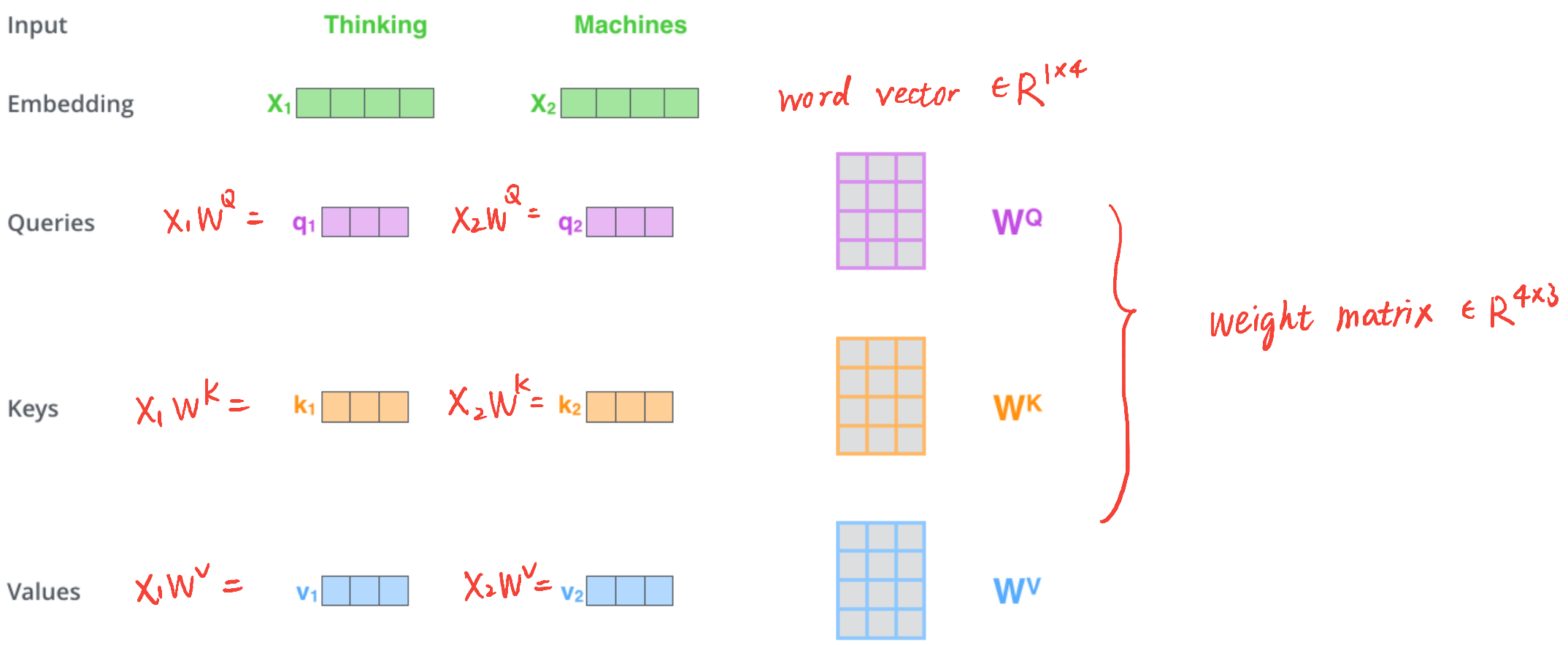
different weight for different context for word "kicked"

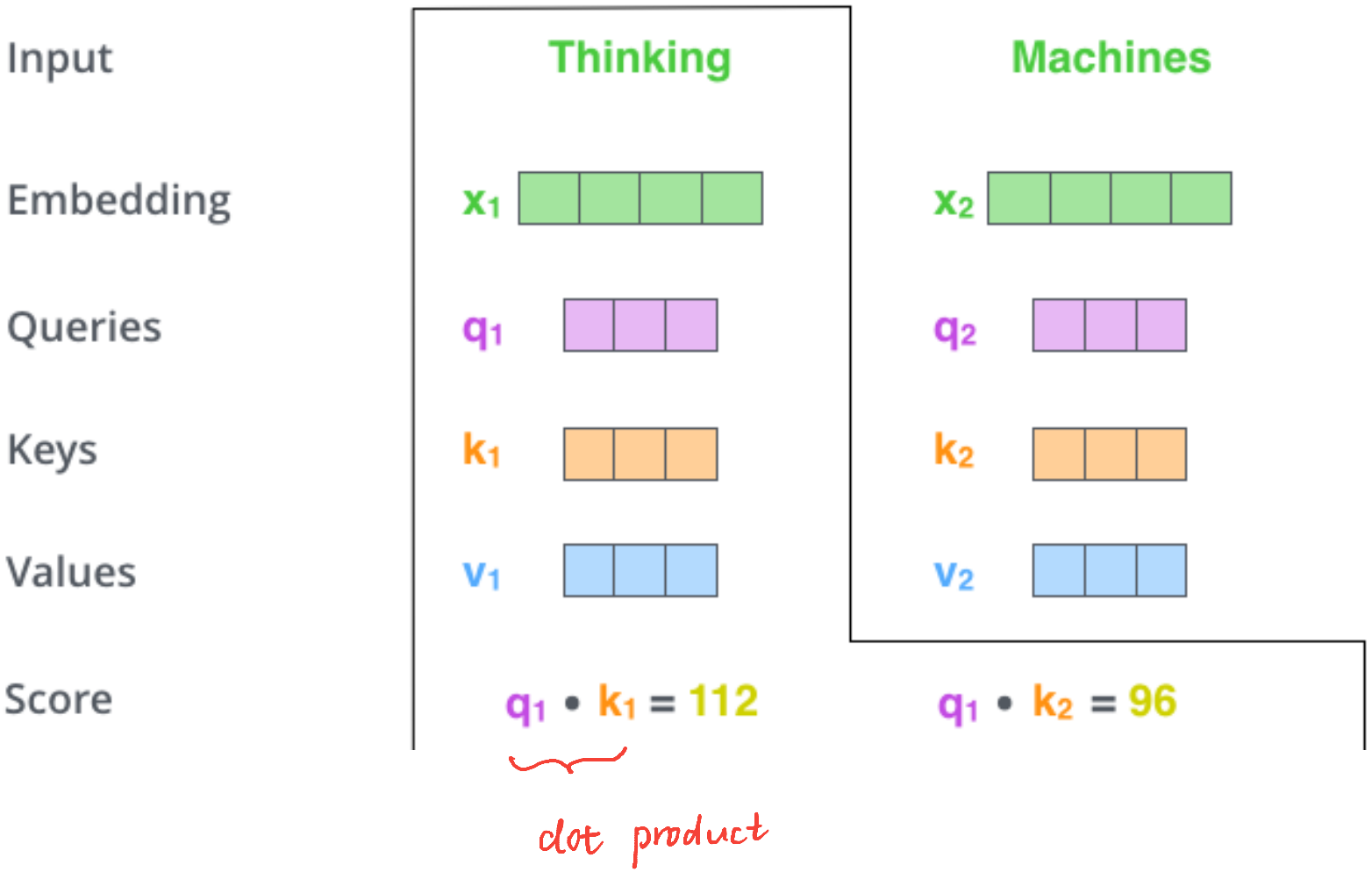


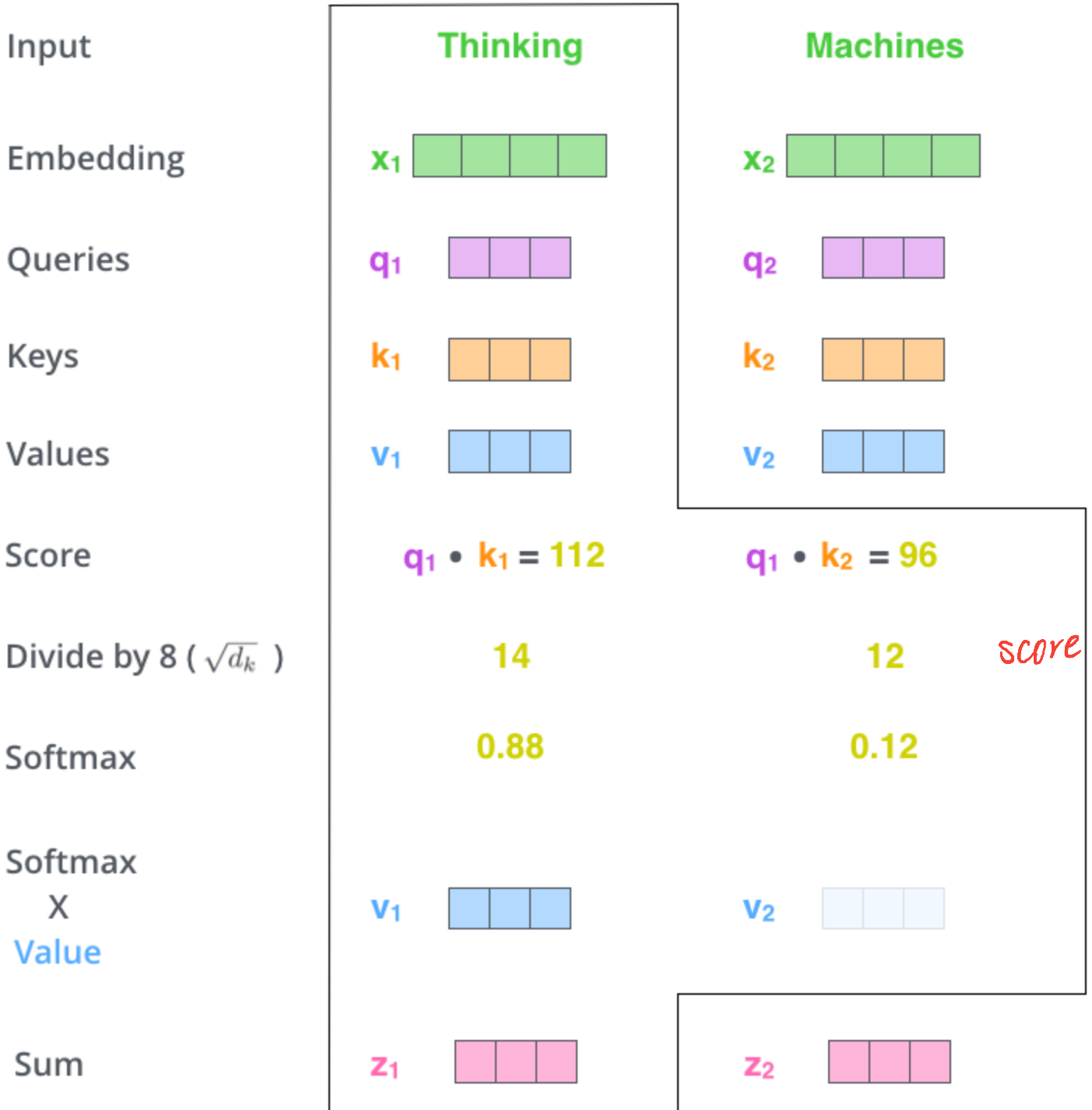
Self-Attention (Filing Cabinet Analogy)



Query, Key, Value




Input
Embedding
Queries
Keys
Values
Score
Thinking
Machines
x1
x2
q1
q2
k1
k2
v1
v2
q1 • k1 = 112
q1 • k2 = 96
dot product



Why the scaling?

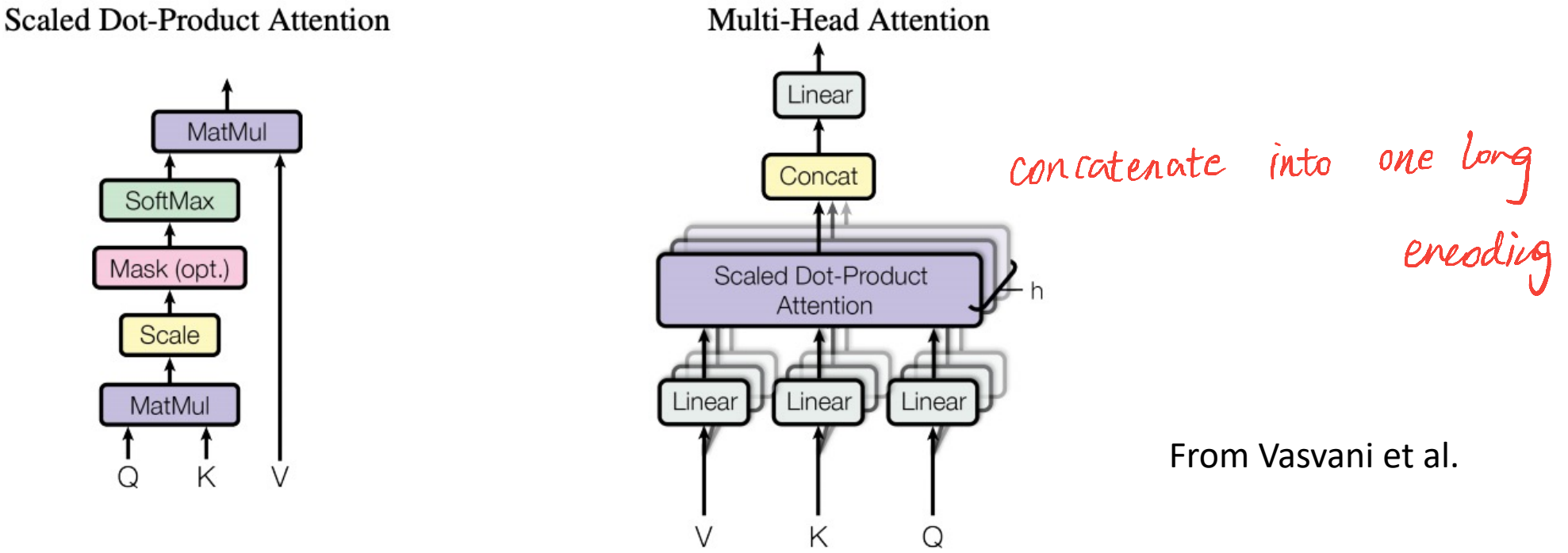
Assuming q and k are d_k dimensional vectors
The dot product is

$$q \cdot k = \sum_{i=1}^{d_k} u_i v_i$$

This has mean 0 and variance d_k
Want to normalize to have variance 1

score is normalized

Multihead Attention

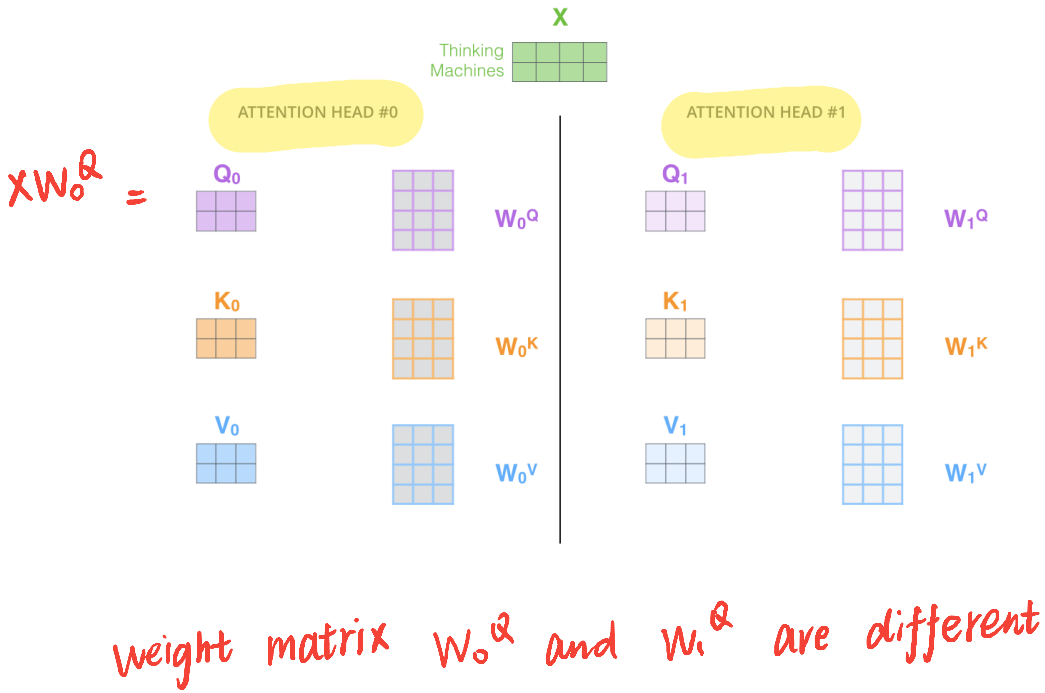


concatenate into one long encoding

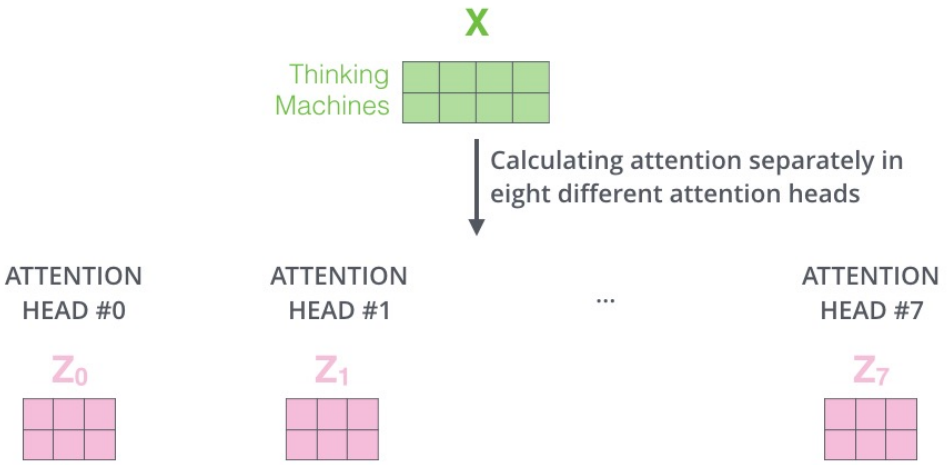
From Vasvani et al.

Multi-head Attention

all linear layers



weight matrix W_0^Q and W_1^Q are different



Concatenating Multi-head Attention

1) Concatenate all the attention heads

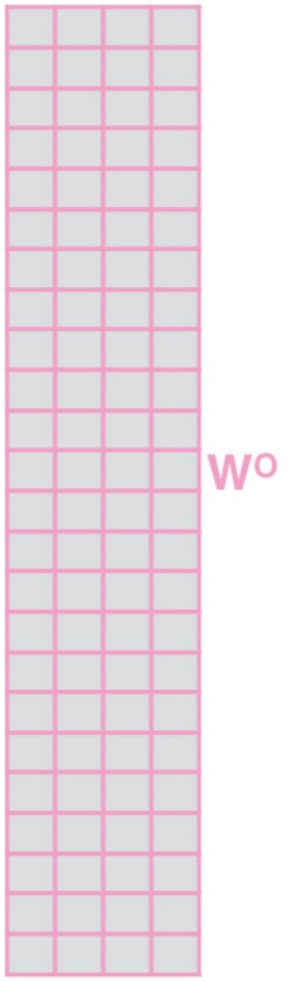
$\in R^{2 \times 24}$

$Z' =$ 

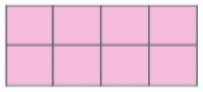
2) Multiply with a weight matrix W^O that was trained jointly with the model

$W^O \in R^{24 \times 4}$

X



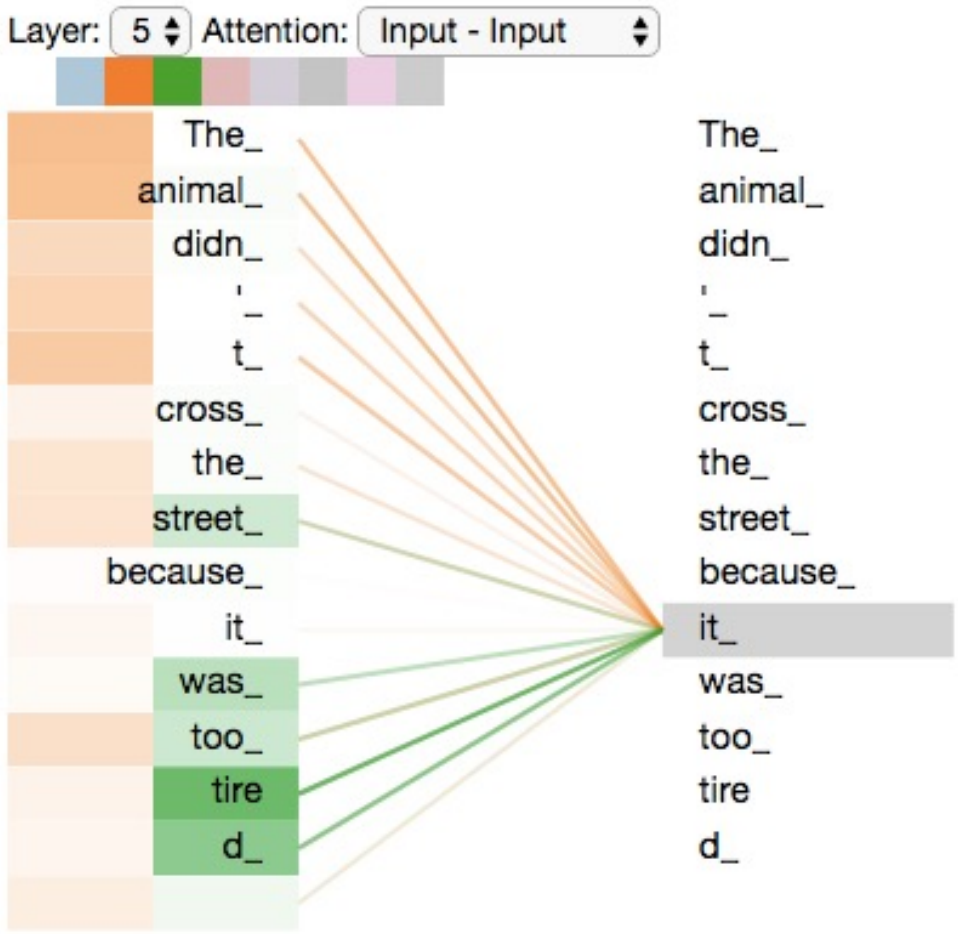
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

$Z' W^O =$ 

$\in R^{2 \times 4}$

Attention Results

8 heads



visualize 2nd head give how much attention to word "it-"

Positional Encoding

fixed length vector $\in R^d$

- Lost positional information now
- Positions can't be encoded by numbers because sequences of different lengths can't be trained this way
- Instead use a periodic function like sin, cos to create the embedding

reason: want positional encoding to be compact rather than one-hot encoding

$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases}$$

$$\omega_k = \frac{1}{10000^{2k/d}}$$

t is tth word in seq

Total positional encoding

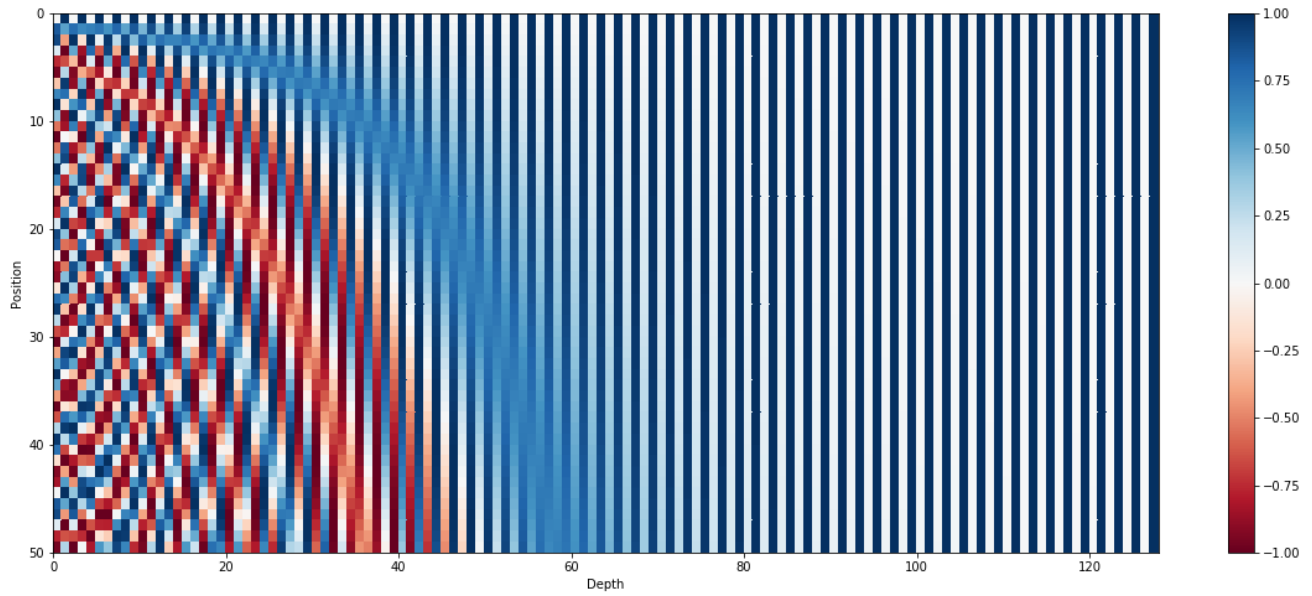
$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

Binary vs Positional Encoding

Binary

0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	2 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1

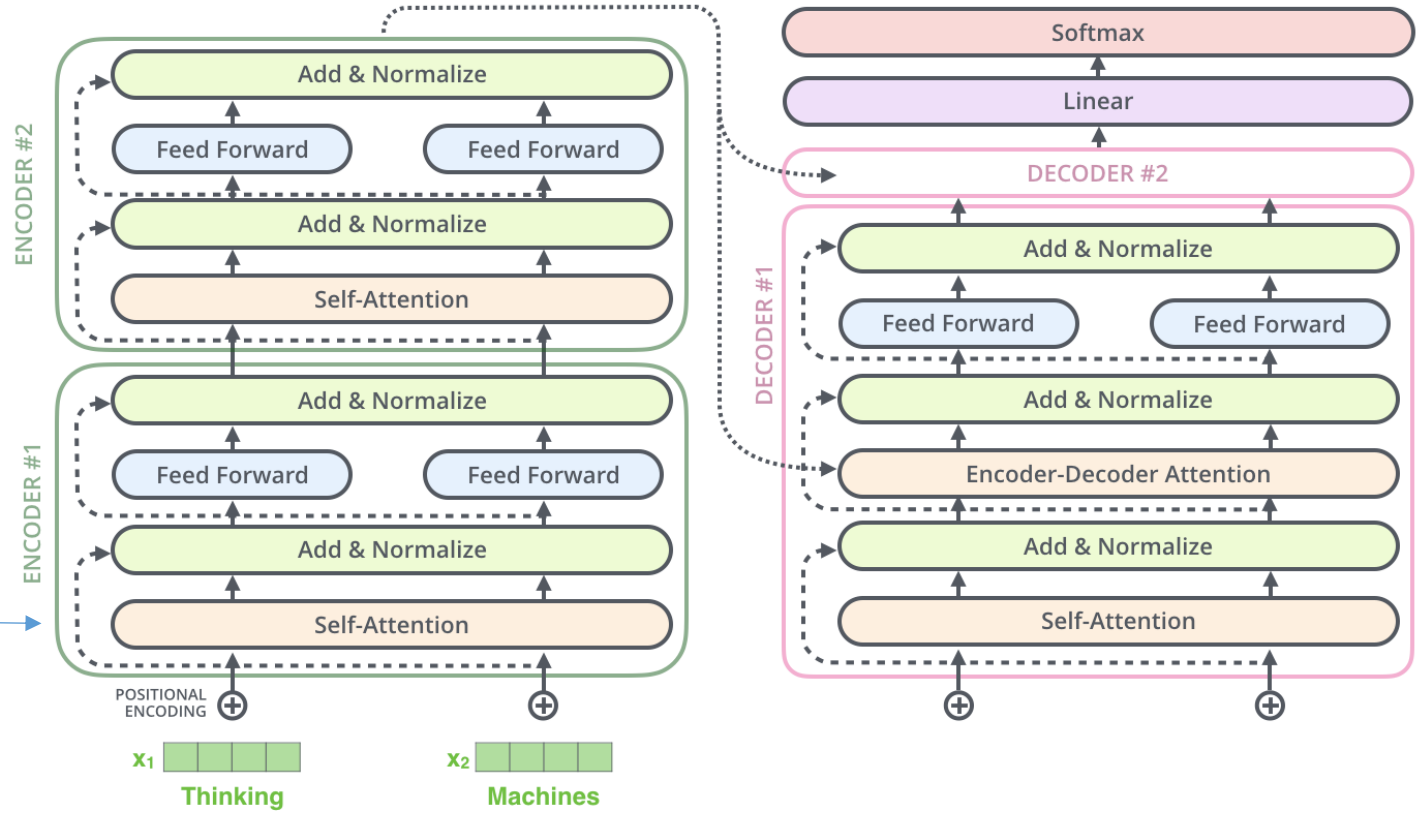
Positional



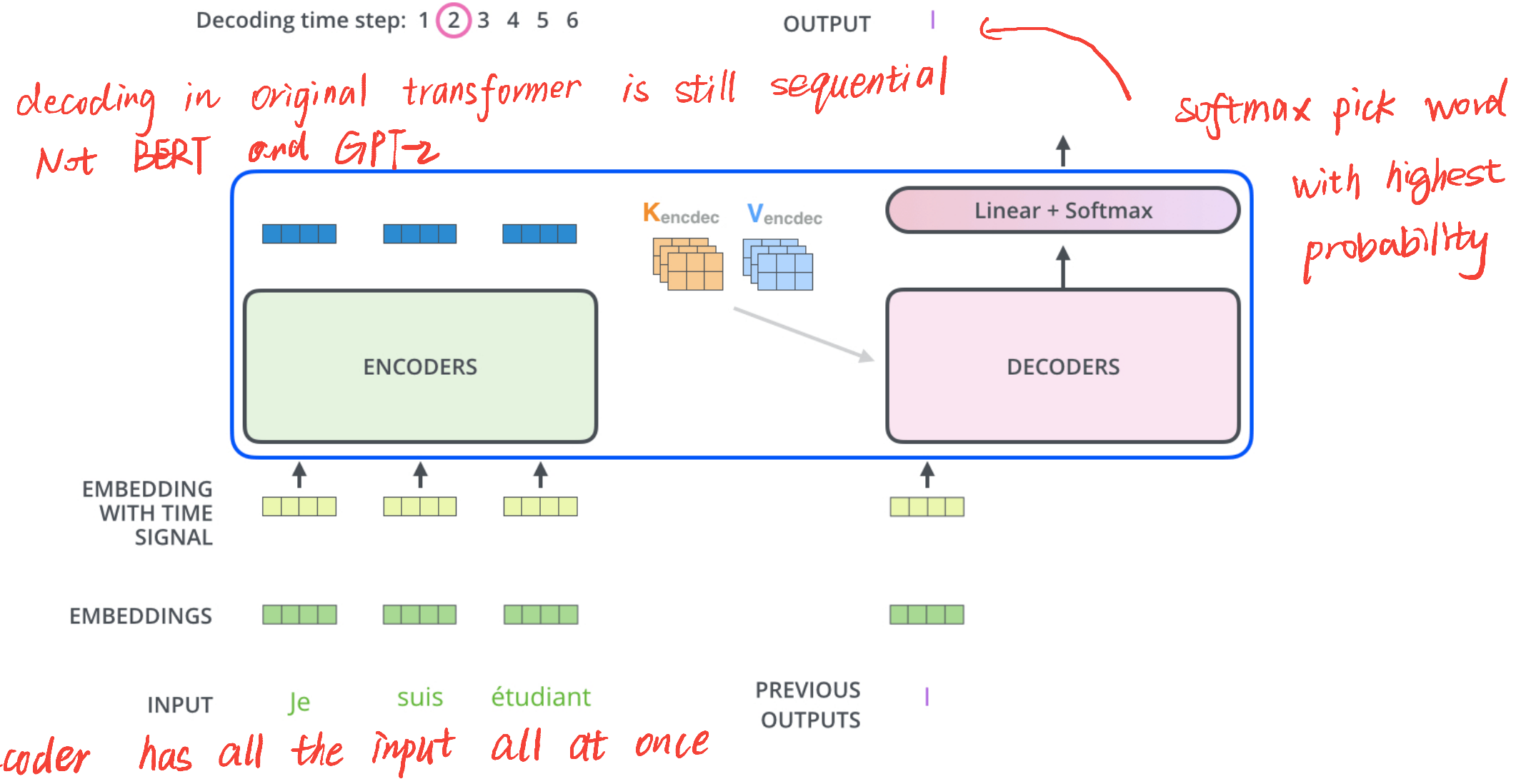
reason that transformer dont have memory issue :
all the inputs together with positional encoding are available for decoders

Transformer

Residuals



Computation in a Transformer



New AI fake text generator may be too dangerous to release, say creators

The Elon Musk-backed nonprofit company OpenAI declines to release research publicly for fear of misuse

GPT-3 generate realistic fake news

Next paragraph prediction

Feed it the opening line of George Orwell's Nineteen Eighty-Four - "It was a bright cold day in April, and the clocks were striking thirteen" - and the system recognises the vaguely futuristic tone and the novelistic style, and continues with:

"I was in my car on my way to a new job in Seattle. I put the gas in, put the key in, and then I let it run. I just imagined what the day would be like. A hundred years from now. In 2045, I was a teacher in some school in a poor part of rural China. I started with Chinese history and history of science."

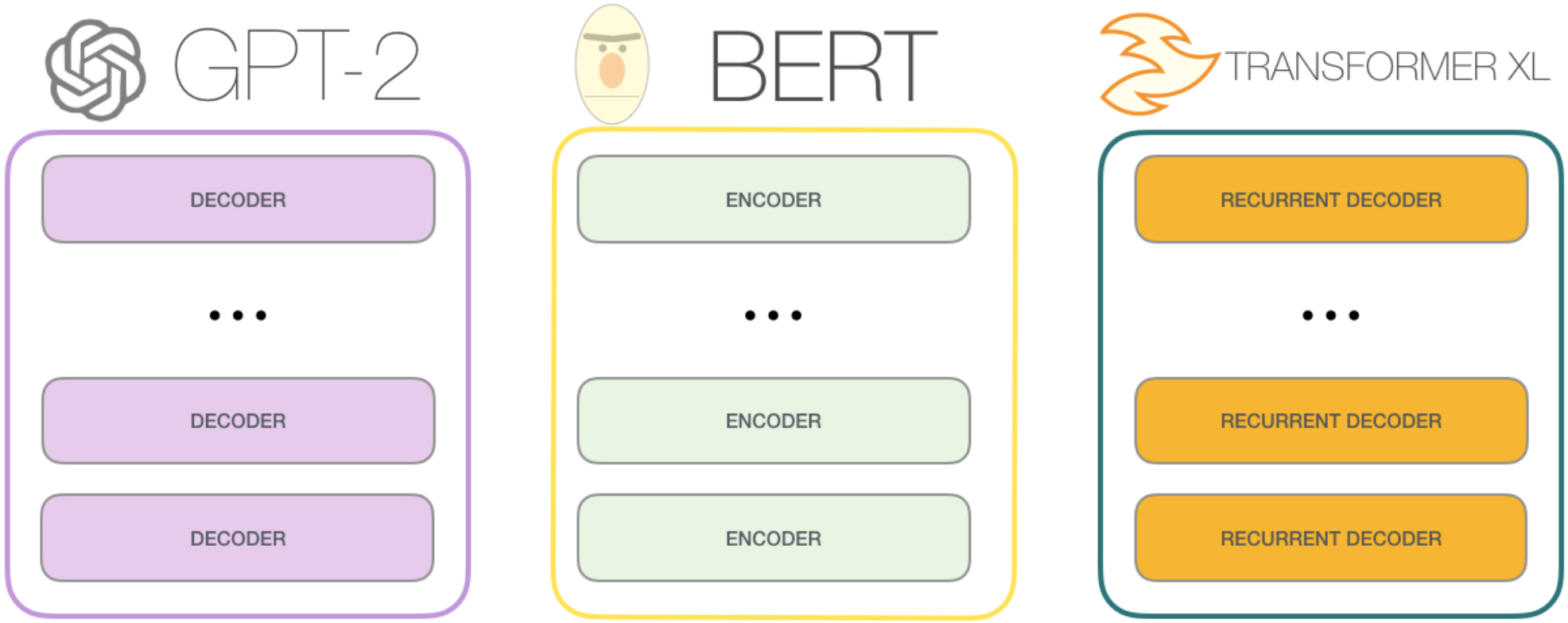
Language Modeling

- An unsupervised training task that models the probability of a word given its context

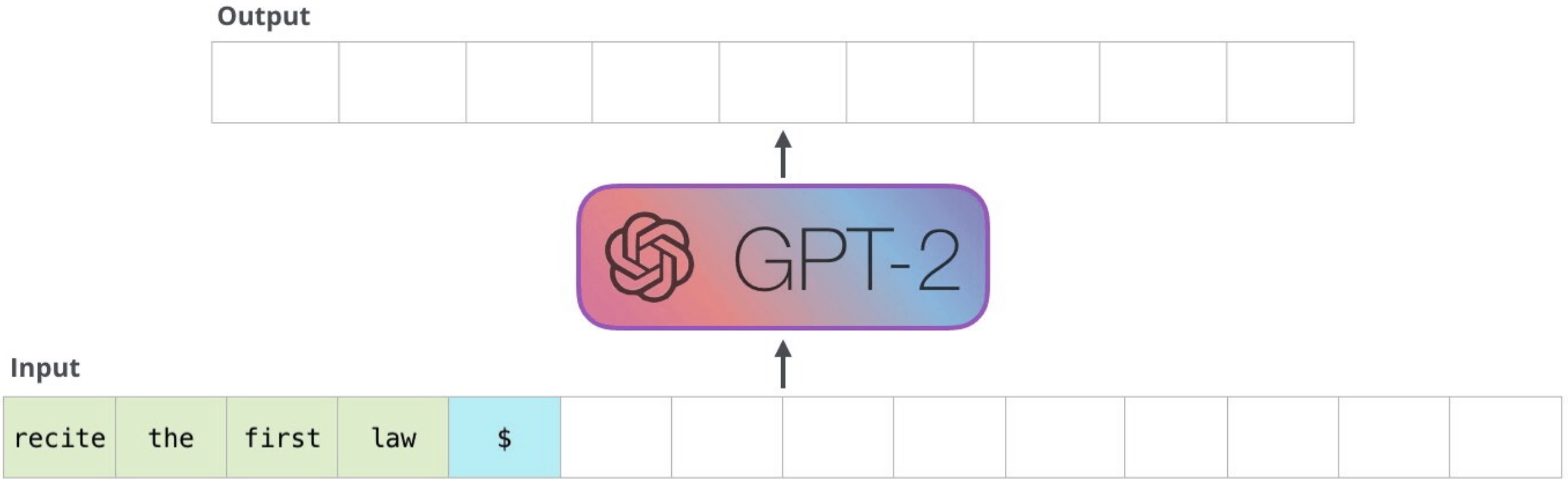
$$p(x) = \prod_{i=1}^{n} p(s_n | s_1, ..., s_{n-1})$$

- Instead of task-oriented training, language models are just trained to predict probabilities from corpuses

Well-known Language Models

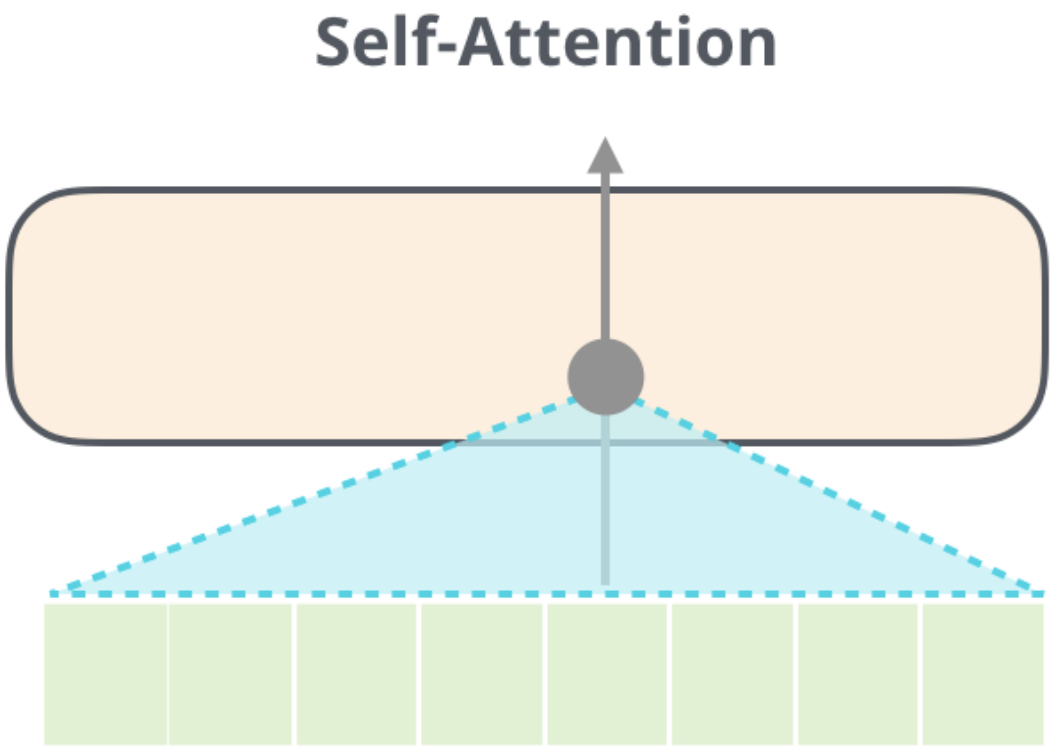


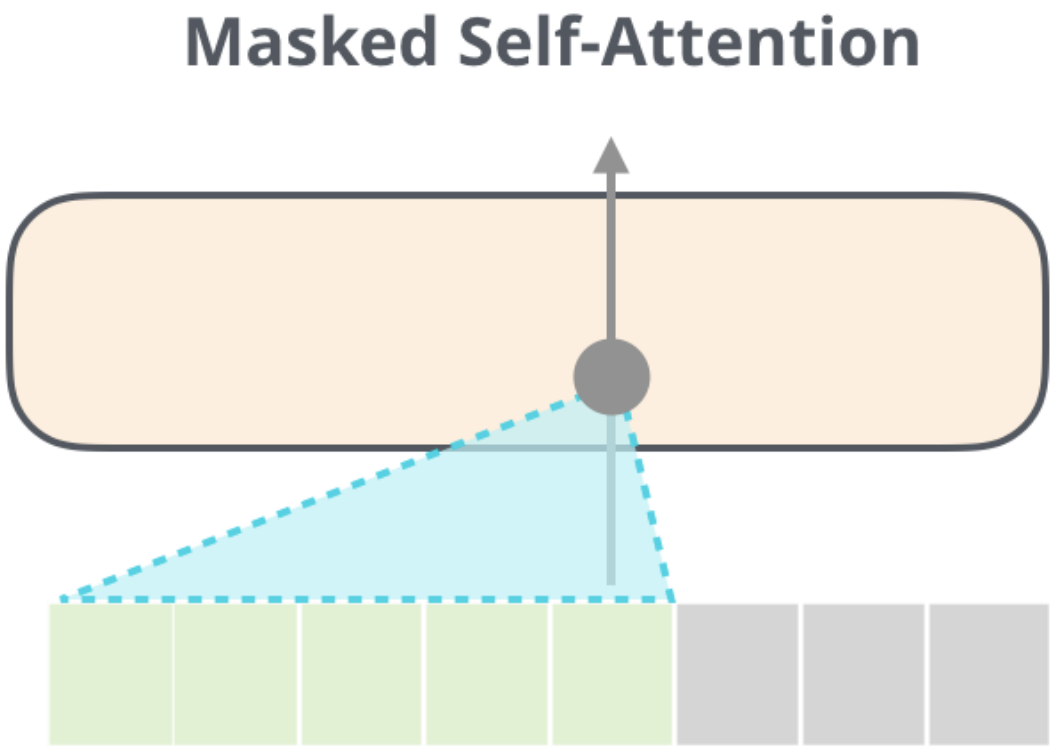
Pretrained Autoregression → next word prediction



Trained sequentially with newly generated output becoming part of the next input sequence

Masked Attention (Mask everything before)

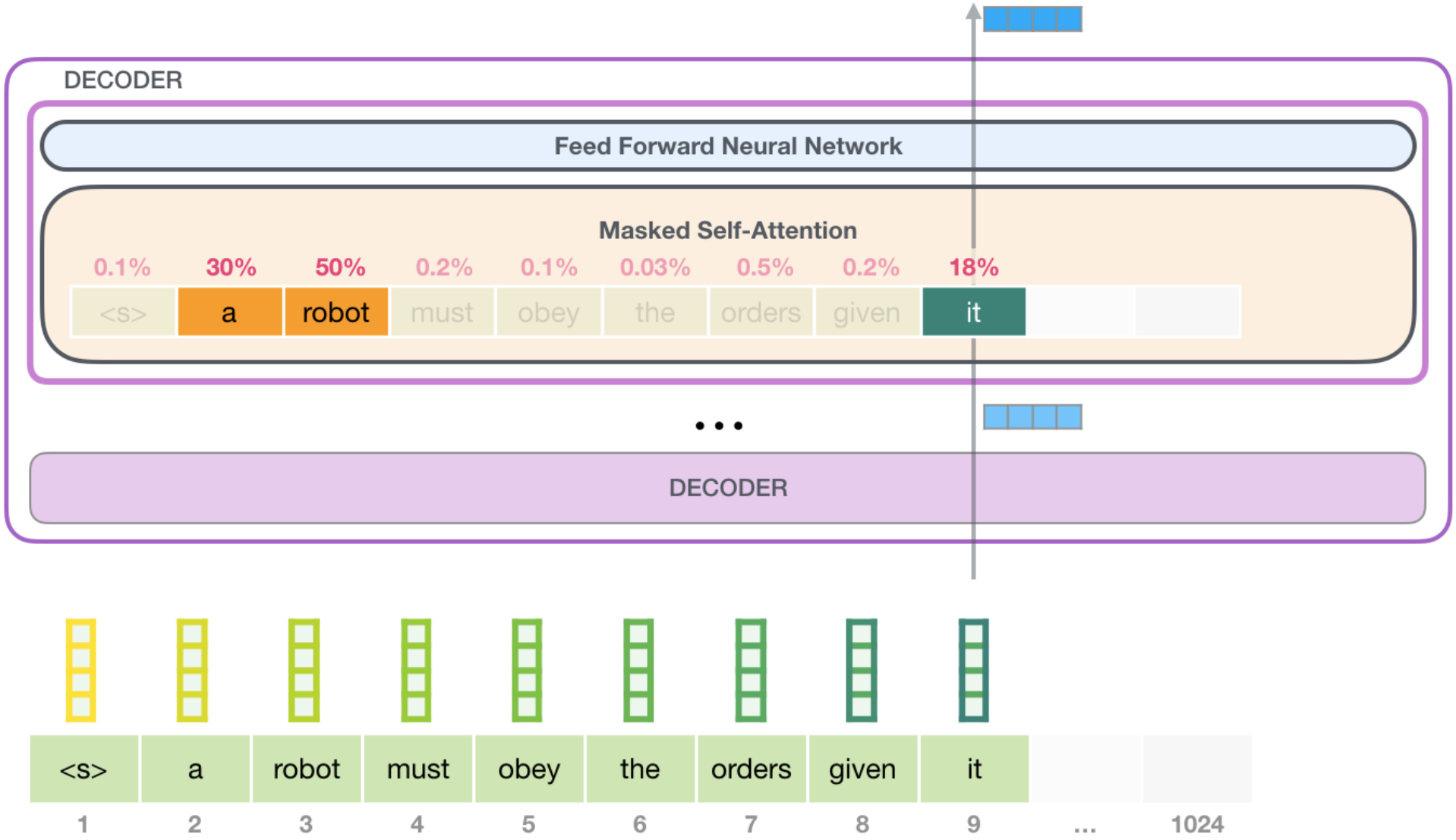




Word is only allowed to pay attention to words come before it

Role of self-attention: context

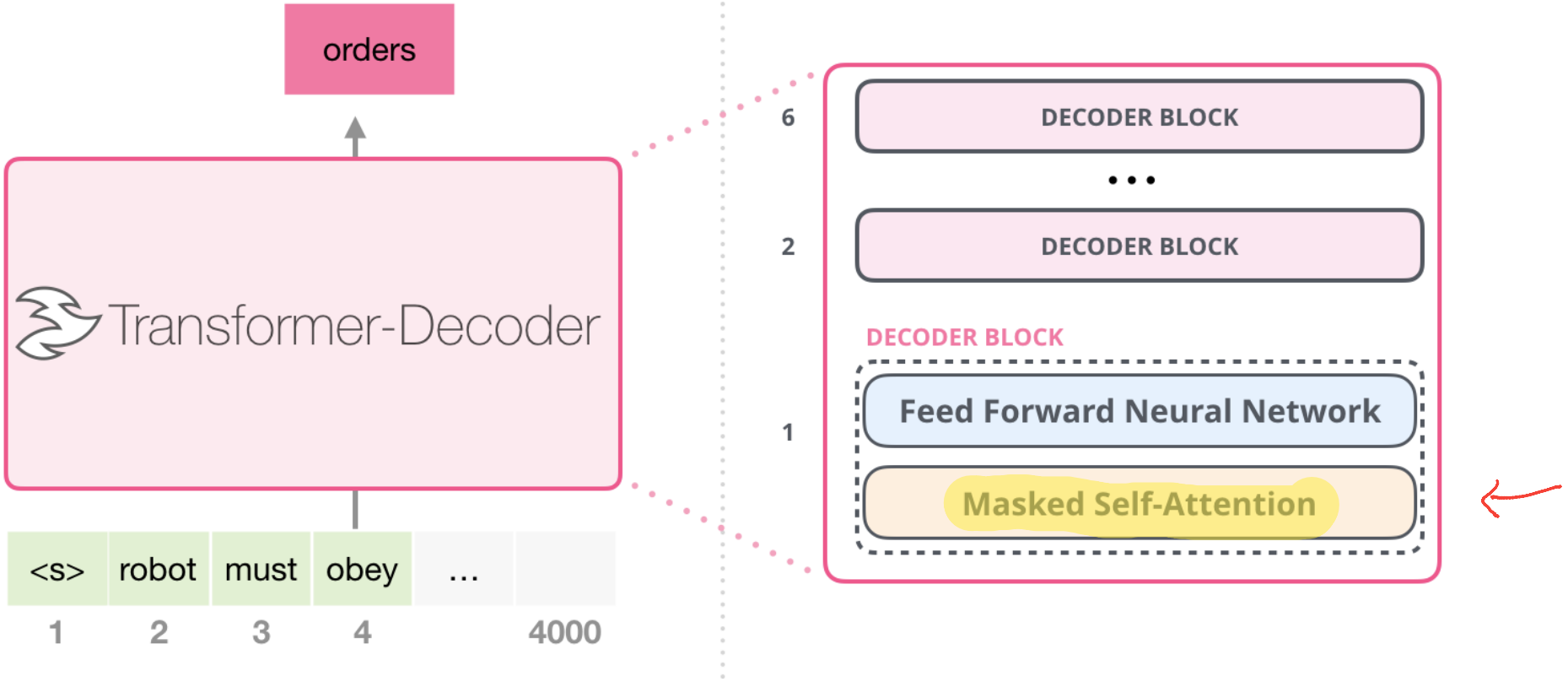
take into account past context



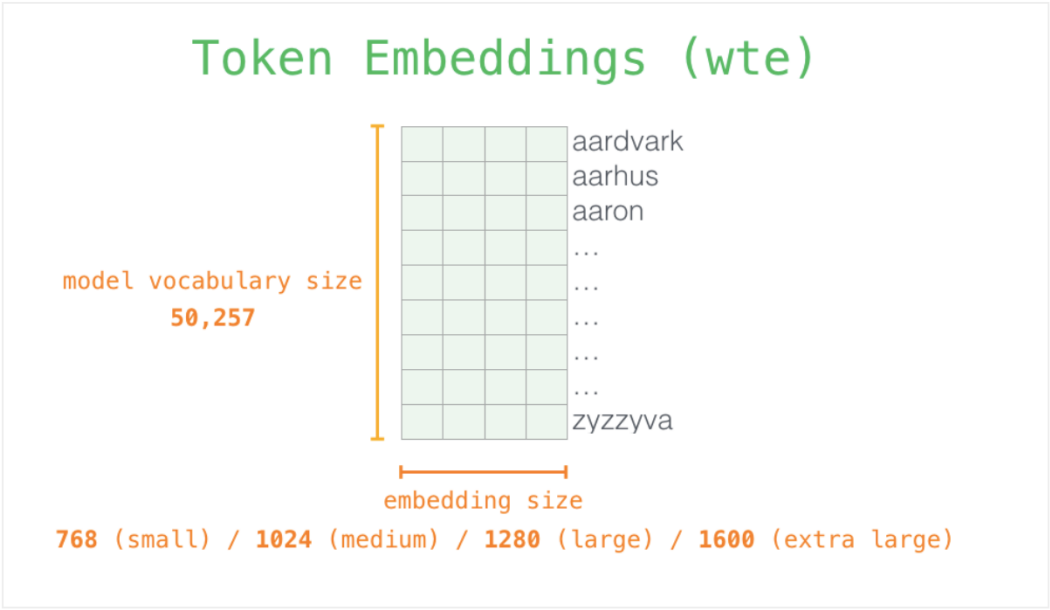
GPT-2 only use decoder part of original transformer, it's fine coz encoder blocks are redundant, very deep in original transformer

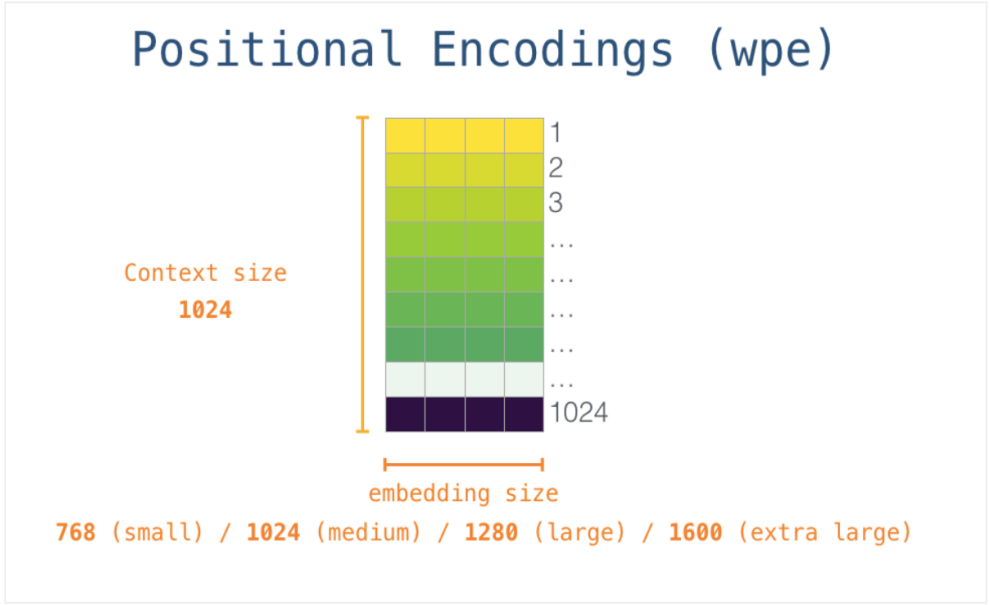
Different: use masked self-attention in decoder

Decoder Only Block



Model Input (with starting token)





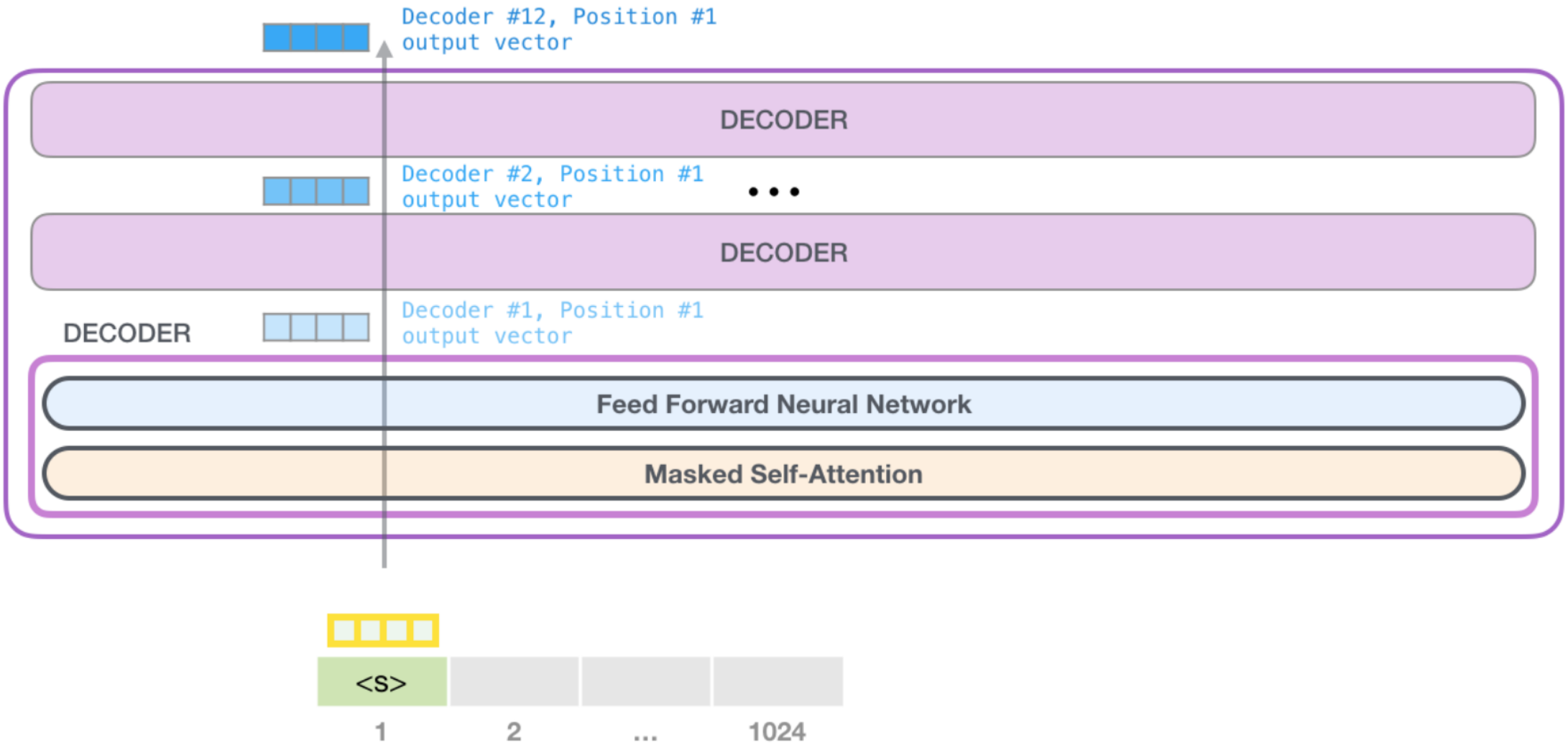
Each row is a word embedding: a list of numbers representing a word and capturing some of its meaning. The size of that list is different in different GPT2 model sizes. The smallest model uses an embedding size of 768 per word/token.

each row is a token embedding

input sentence length : 1024


DECODER
DECODER
=
Positional encoding for token #1
+
Token embedding of <s>
<s>
1
2
...
1024
Token Embeddings
Positional Encodings

Processing


Decoder #12, Position #1
output vector
DECODER
Decoder #2, Position #1
output vector
DECODER
DECODER
Decoder #1, Position #1
output vector
Feed Forward Neural Network
Masked Self-Attention
<s>
1
2
...
1024

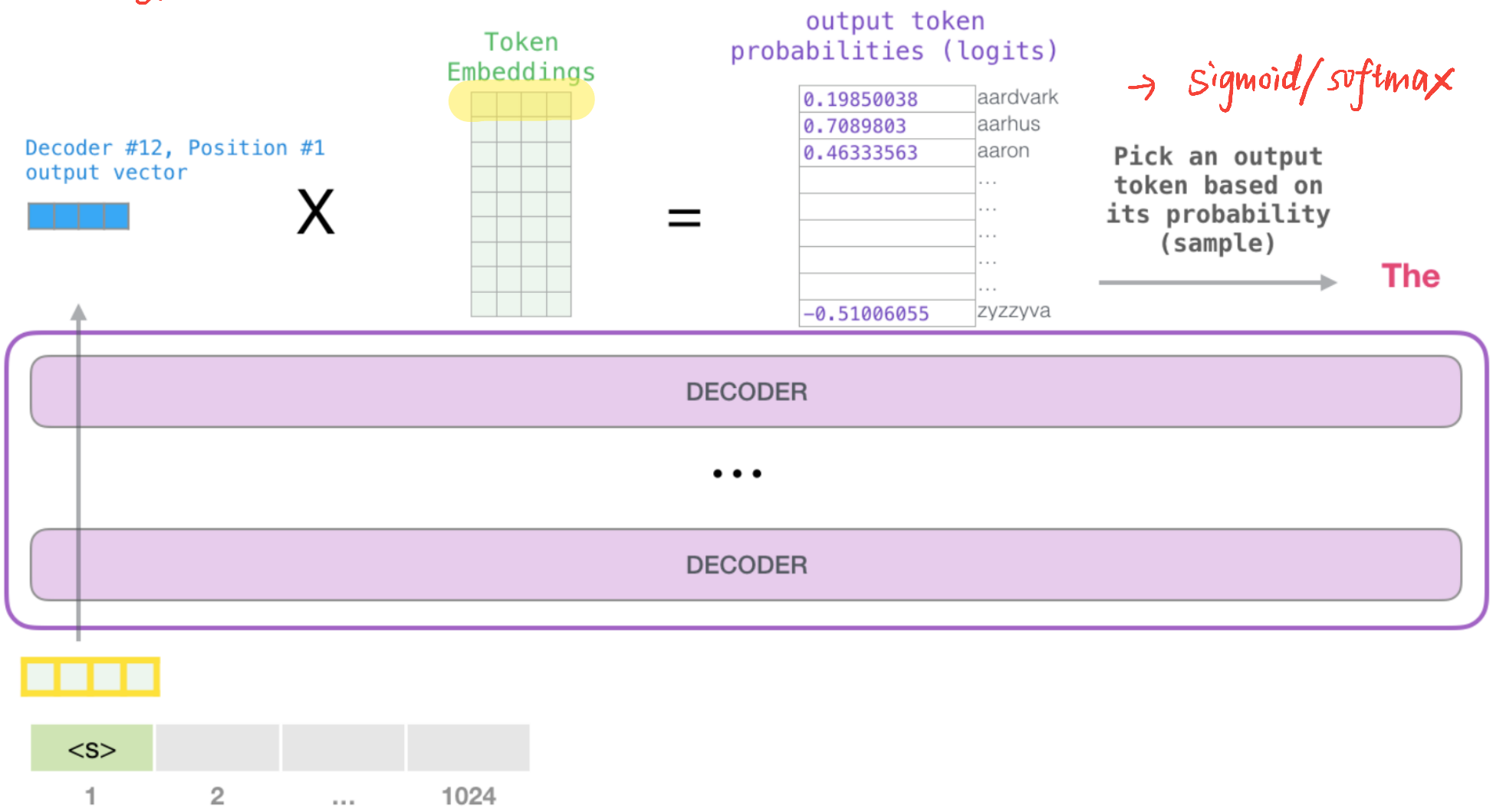
multiply

Word	Value vector	Score	Value X Score
<s>		0.001	
a		0.3	
robot		0.5	
must		0.002	
obey		0.001	
the		0.0003	
orders		0.005	
given		0.002	
it		0.19	
		Sum:	

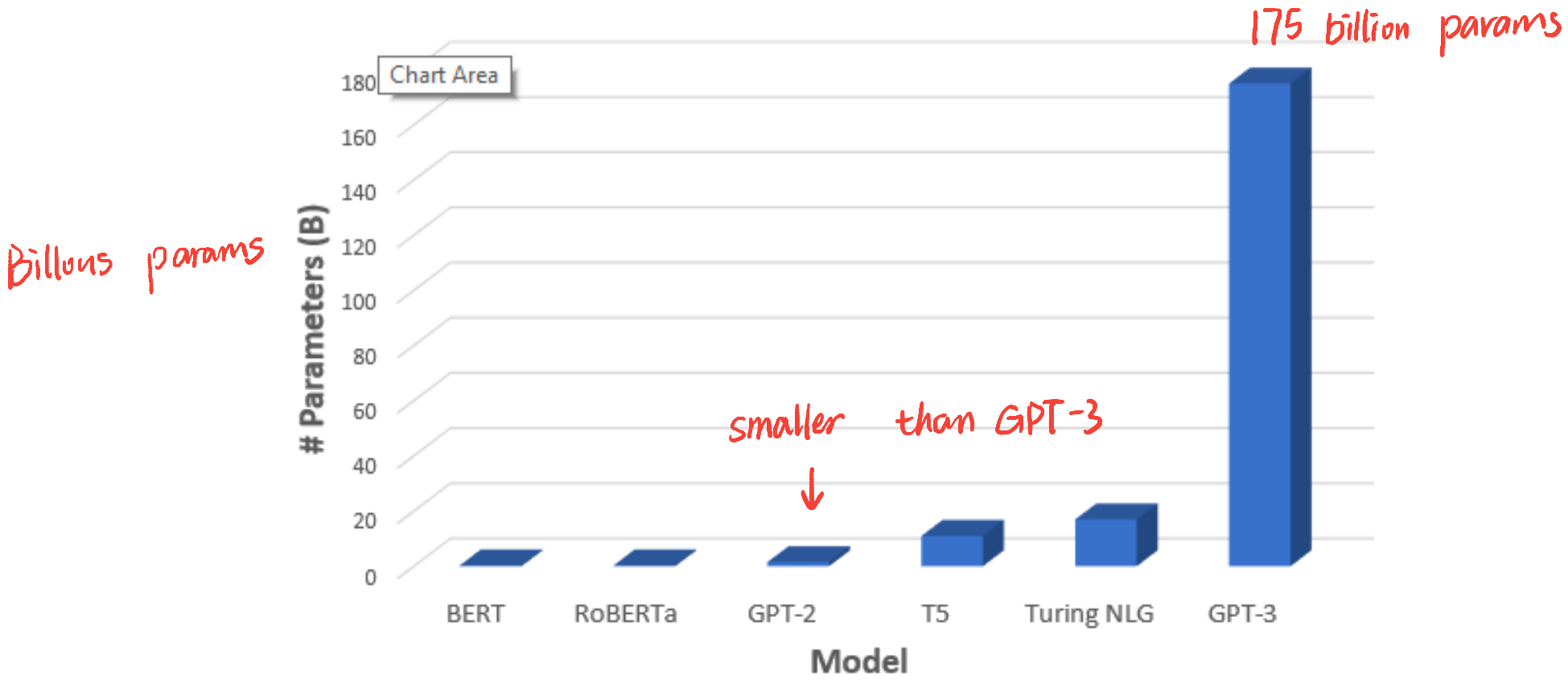
Model Output

$R^{1\times4} \cdot R^{4\times1}$ = probability

elementwise dot product of output vector and token embedding



GPT Models



https://demo.allennlp.org/next-token-lm

https://openai.com/api/

MuseNet-Music Generator uses GPT2

Starts with a music token and generates next token in Sequence

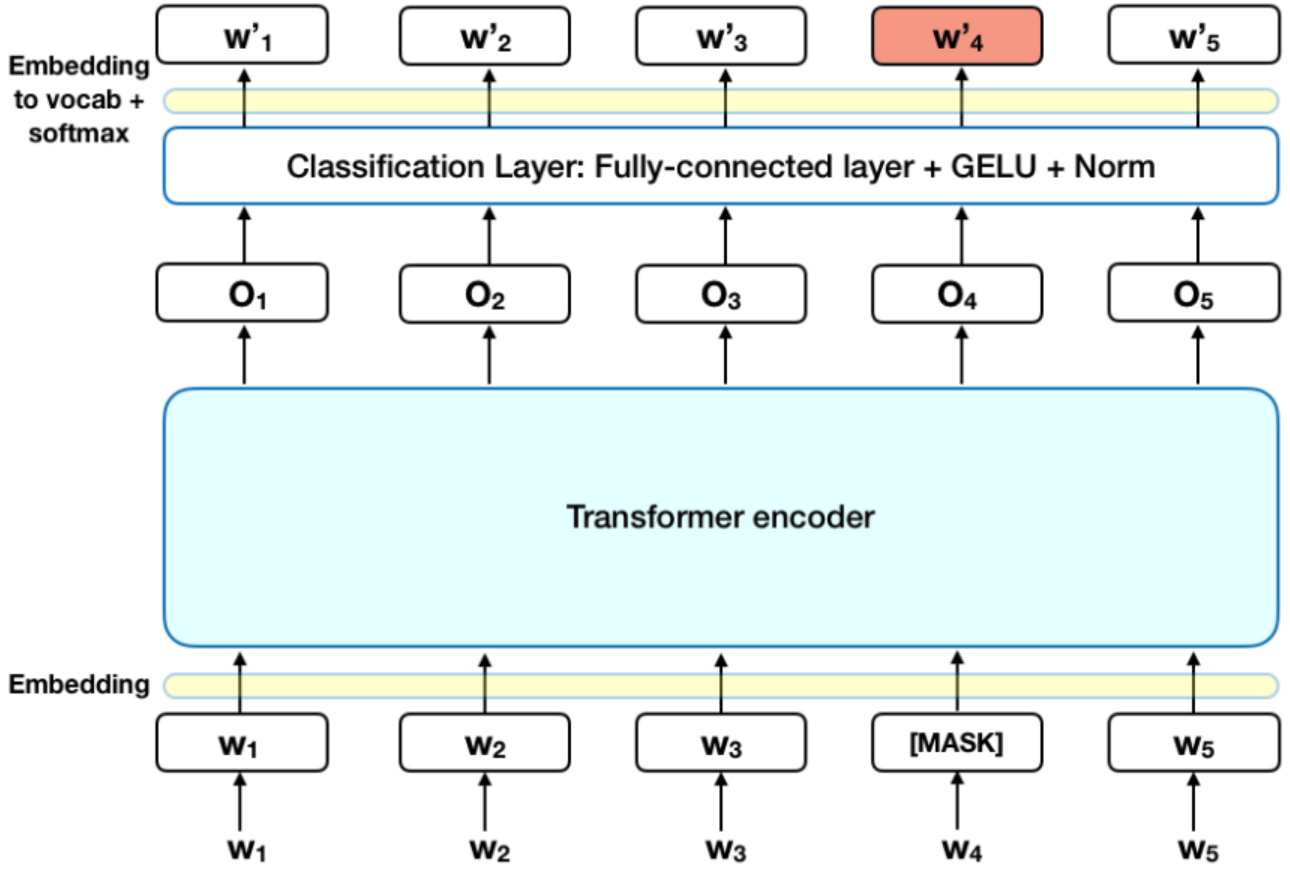
Tokens contain information on:
1. pitch, 音高
2. volume, 音量
3. Instrument 乐器

Trained on a large archive of MIDI files

https://openai.com/blog/musenet/

Bert—Encoder Only

Bert only use encoder part of original transformer and use self-attention not masked self-attention



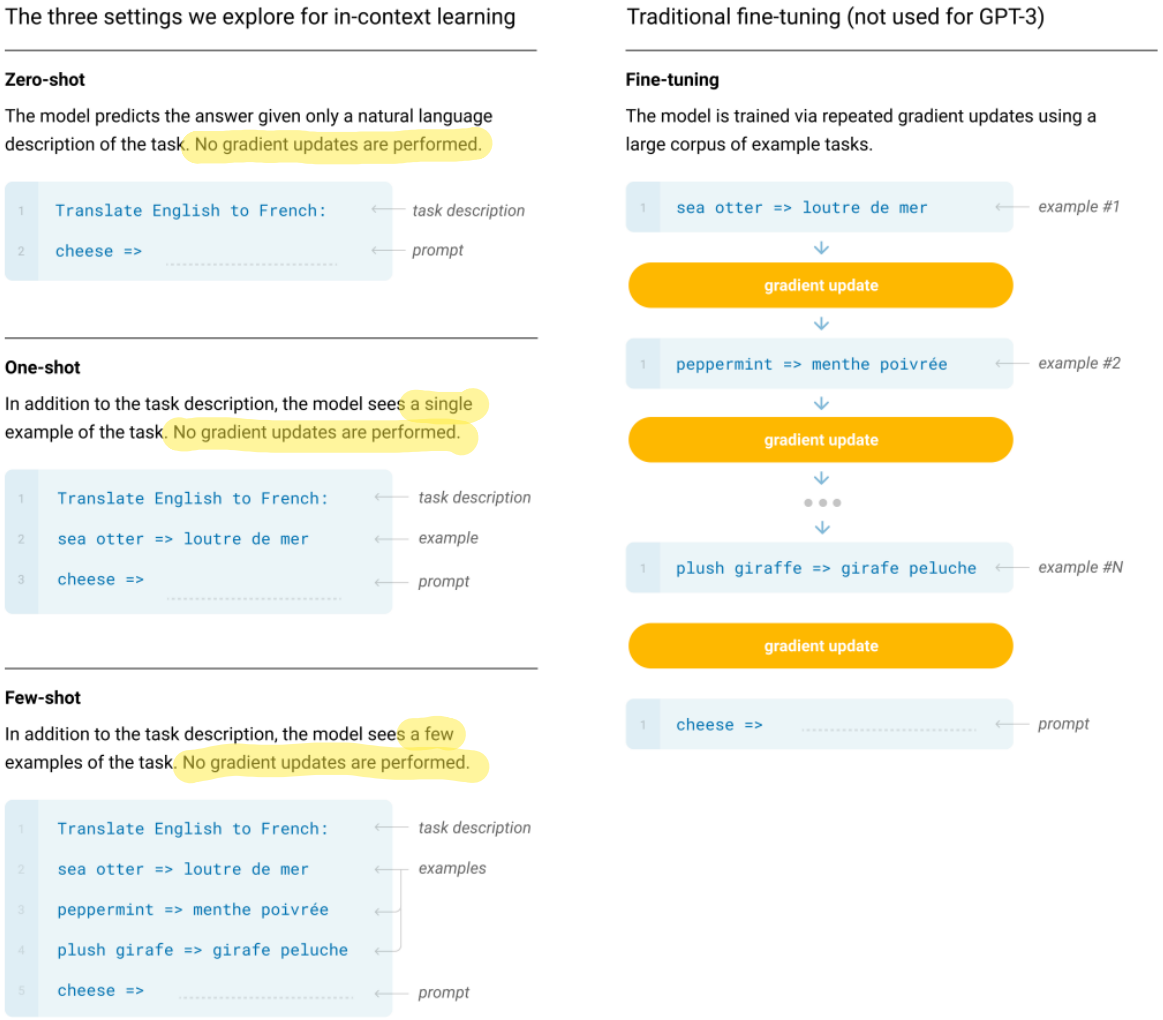
Trained to predict masked words

GPT-2 and Bert are all trained unsupervised

predict the next word

or pedict the masked word

Few Shot Learning



Reading list

- https://towardsdatascience.com/attention-and-its-different-forms-7fc3674d14dc
- http://jalammar.github.io/illustrated-transformer/
- Vaswani et al. 2019 https://arxiv.org/abs/1706.03762 ↙ Transformer
- Radford et al. (GPT2 paper)
- Brown et al. (GPT3 paper)
- Child et al. (Sparse Transformer)
- https://jalammar.github.io/illustrated-gpt2/ ↗ illustrated GPT 2
- https://medium.com/walmartglobaltech/the-journey-of-open-ai-gpt-models-32d95b7b7fb2