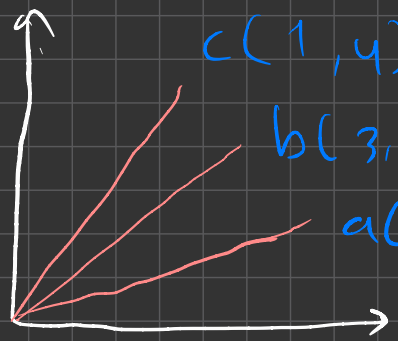
* Dot Product :-



c(1,4)
b(3,1)
a(4,1)

$a \cdot b = 13 > a \cdot c = 8$

$[4\ 1] \cdot \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$4 \cdot 3 + 1 \cdot 1 = 13$

(1,2) r c (2,1) r c $= \sum_j a_j b_j = 4*3 + 1*1 = 13$

---

* word Embedding :-

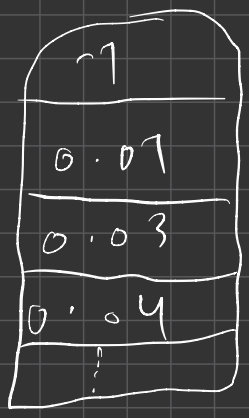
	gender	Royal	Age	Food
man	-1	0.01	0.03	0.04
woman	1	0.02	0.02	0.01
king	-0.95	0.93	0.7	0.02
queen	0.95	0.97	0.69	0.01
⋮	⋮	⋮	⋮	⋮

10k

300

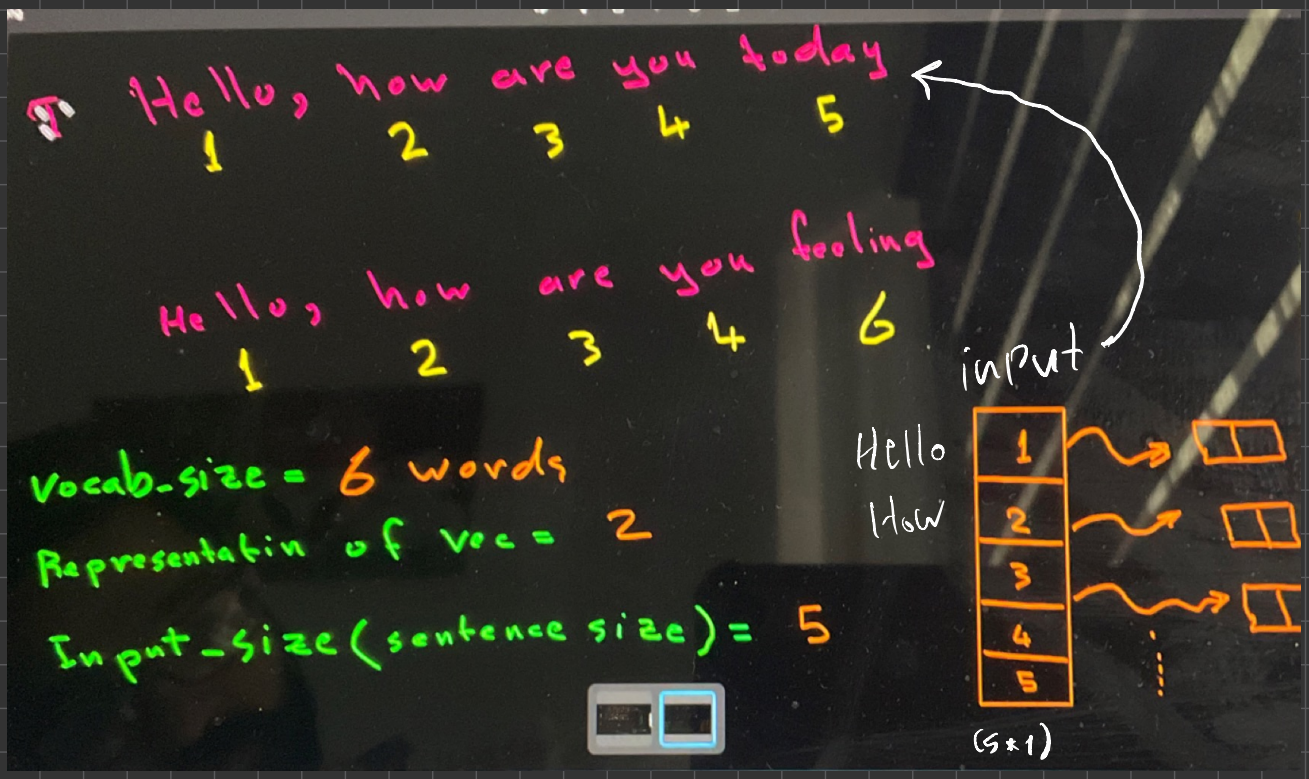
D

man



$E_{(10k, 300)}$

each word shape $= (1, 300)$



Tokenization

"Where can I find a pizzeria?"

Word-Level Tokenization

[where, can, i, find, a, pizzeria]

- ⊕ Intuitive.
- ⊖ Doesn't handle OOV words including new words, slang, play on words, misspellings, etc.
- ⊖ Huge vocabulary for large corpora; especially for languages with rich morphology (e.g. Hungarian).
- ⊖ Handling punctuation is challenging (e.g. "don't" vs "N.Y.C").

Char-Level Tokenization

[a, c, d, e, f, h, i, n, p, r, w, z]

- ⊕ Small memory footprint.
- ⊕ Handles OOV words.
- ⊖ Needs to go over all characters and learn a particular sequence for a given word.
- ⊖ Loss of performance.

Subword Tokenization

"where can I find a pizzeria?"

where, can, I, find, a, pi, zz, eria

listeria

[list, eria]

Subword tokenization has a better chance of handling OOV words while reducing vocabulary size and maintaining performance.

Byte Pair Encoding (BPE)

"she sells seashells by the seashore"

Corpus

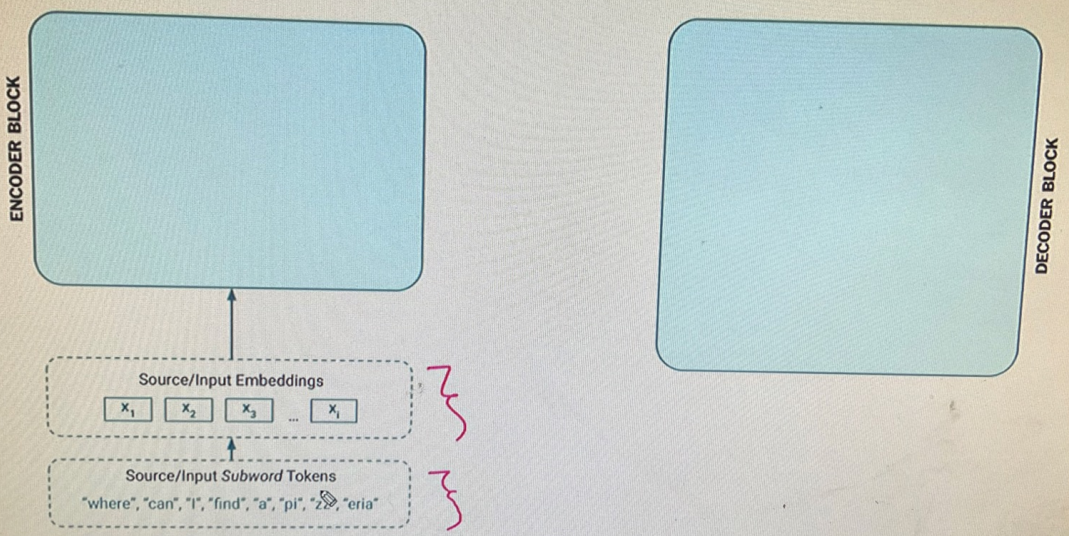
she_
sells_
seashells_
by_
the_
seashore_

Vocabulary

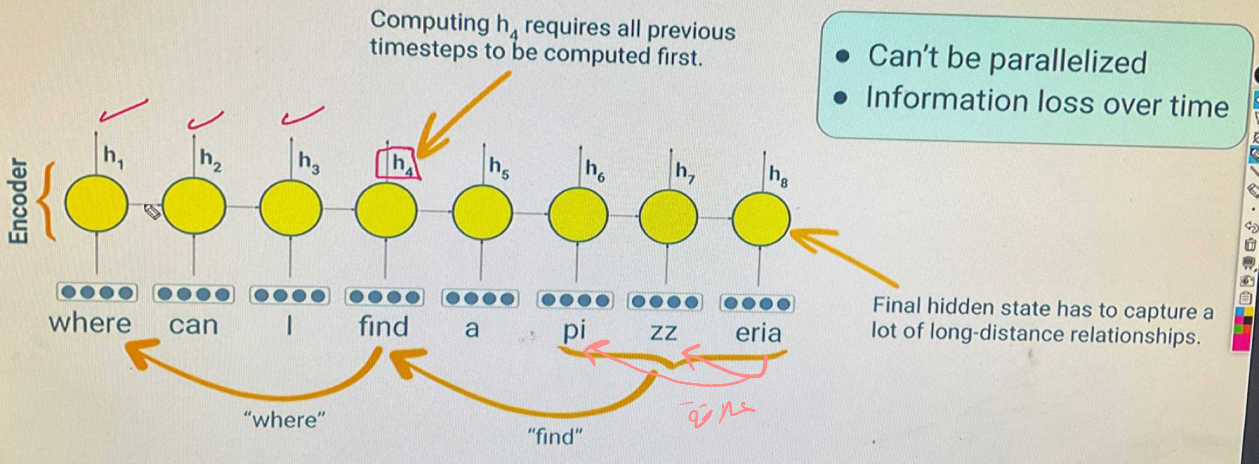
_, a, b, e, h, l, o, r, s, t, y
_, a, b, e, h, l, o, r, s, t, y, sh
_, a, b, e, h, l, o, r, s, t, y, sh, he
_, a, b, e, h, l, o, r, s, t, y, sh, he, e_
_, a, b, e, h, l, o, r, s, t, y, sh, he, e_, se
_, a, b, e, h, l, o, r, s, t, y, sh, he, e_, se, she
.
.
.

continue until N merges are performed.

Transformers

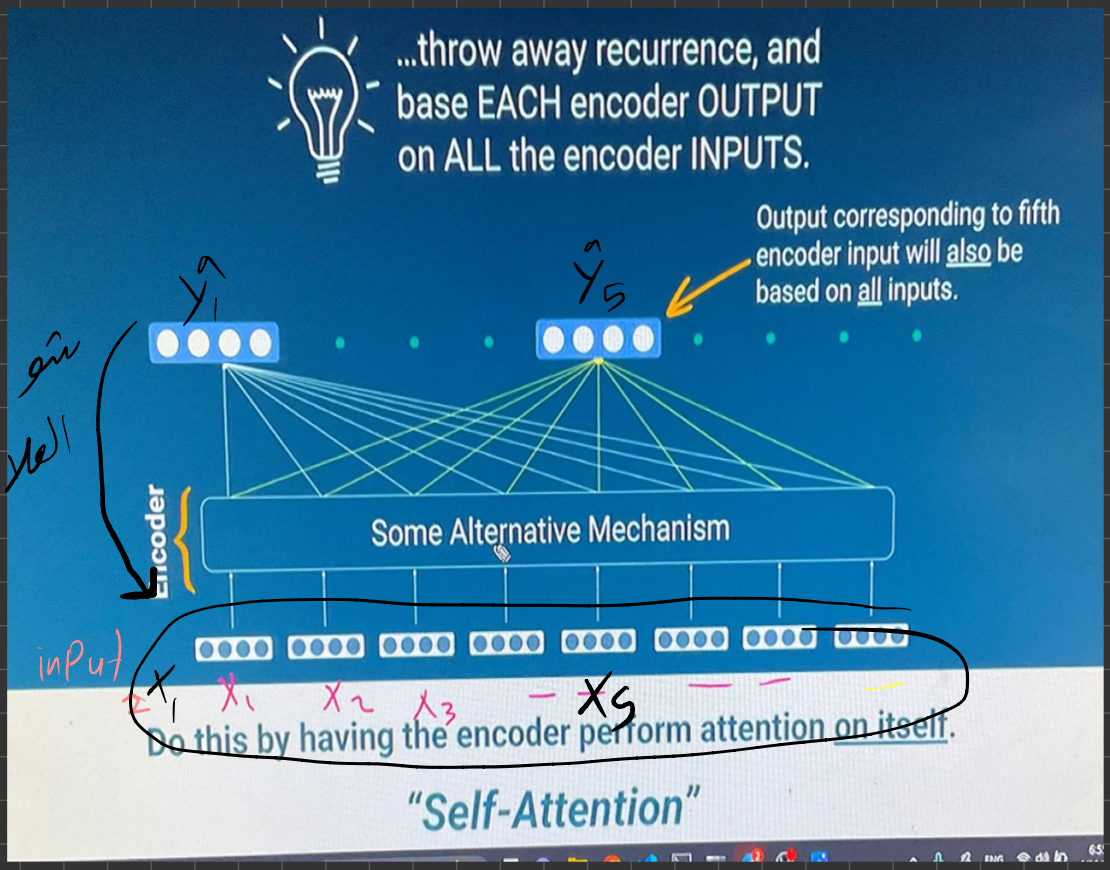


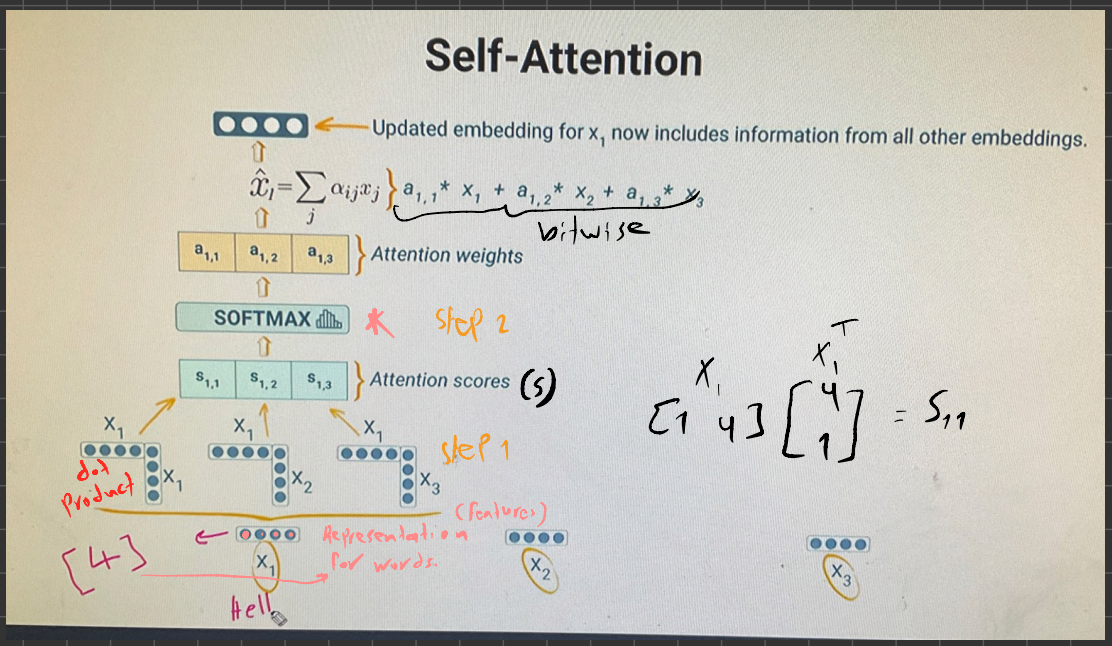
Why Transformers



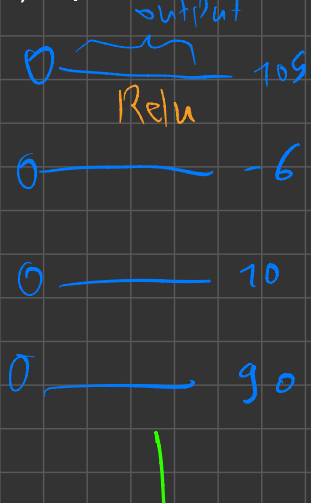
- Can't be parallelized
- Information loss over time

Final hidden state has to capture a lot of long-distance relationships.





* Softmax:



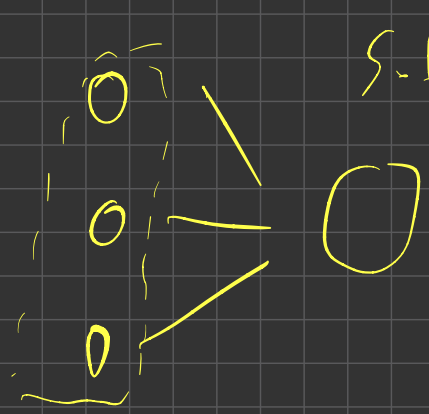
$S \in [-\infty, +\infty]$

$S = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{bmatrix}$ dog, cat

score

Goal: score → Probabilities

need $\begin{cases} p_i \in [0,1] \\ \sum p_i = 1 \end{cases}$



$$p_i = \frac{s_i}{\sum_j s_j}$$

↑ when we use this formula

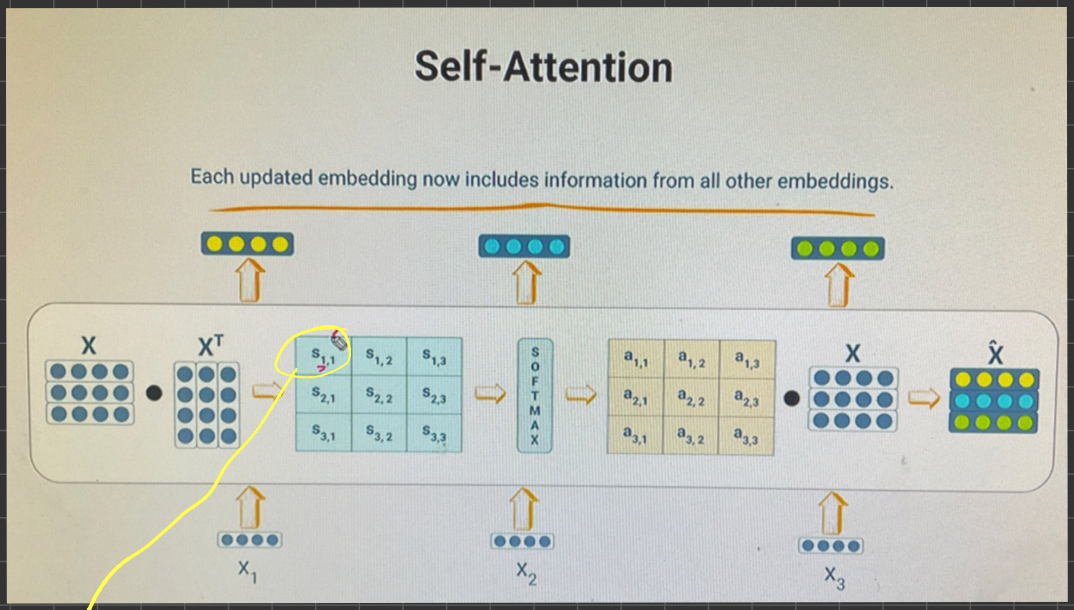
$$\begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1/3 \\ 2/3 \end{bmatrix}, \quad \begin{bmatrix} 100 \\ 101 \\ 102 \end{bmatrix} = \begin{bmatrix} 0,33 \\ 0,33 \\ 0,33 \end{bmatrix}$$

$$p_i = \frac{e^{s_i}}{\sum e^{s_j}} \quad ; \quad p_i = \frac{s_i}{\sum_j s_j}$$

$$\bar{p}_i = \frac{e^{s_i + 100}}{\sum_j e^{s_j + 100}} = \frac{e^{s_i} \cdot e^{c}}{\sum_j e^{s_j} \cdot e^{c}}$$

→ constant of Prob.

a negative numbers that output from Node if go inside softmax will be this number positive.

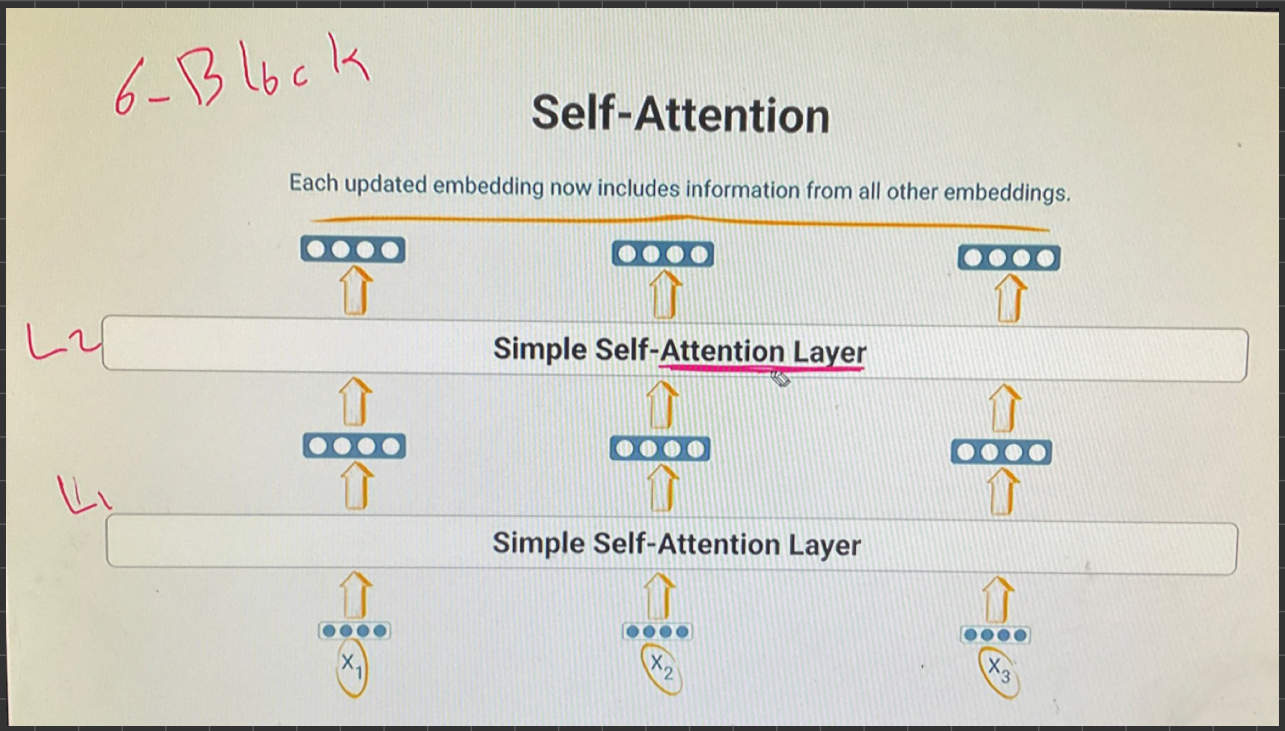


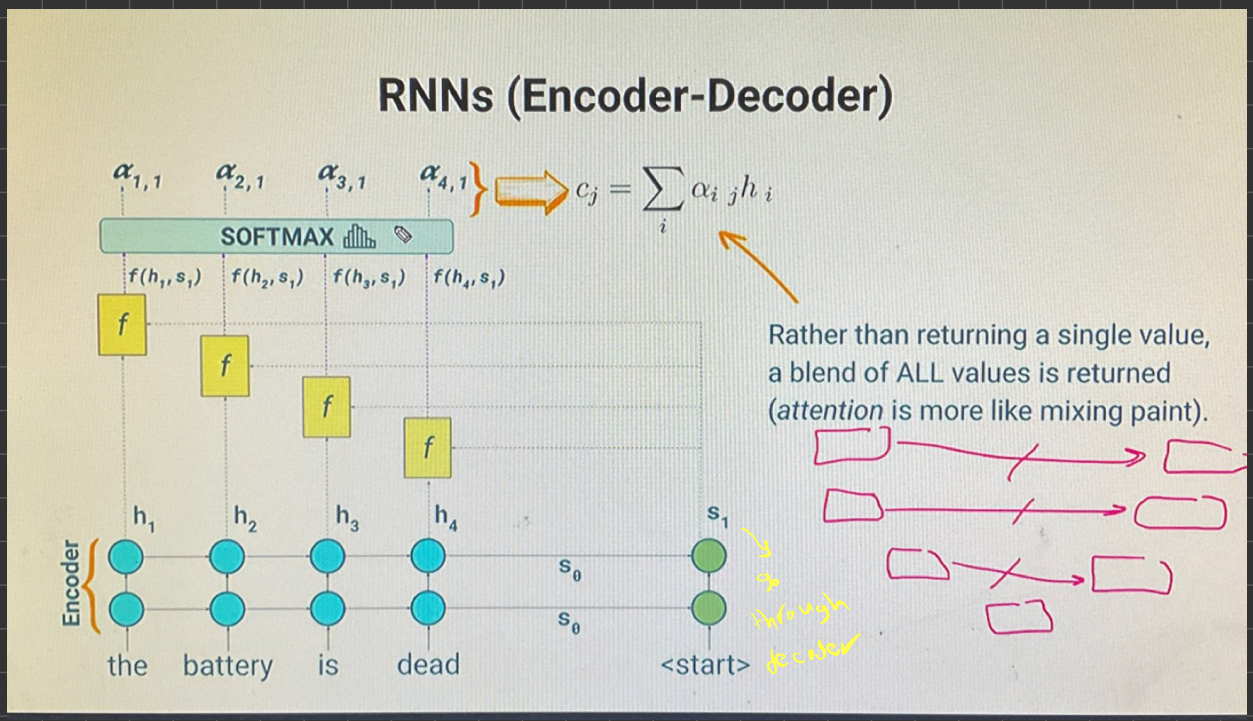
$s_{1,1}$

position

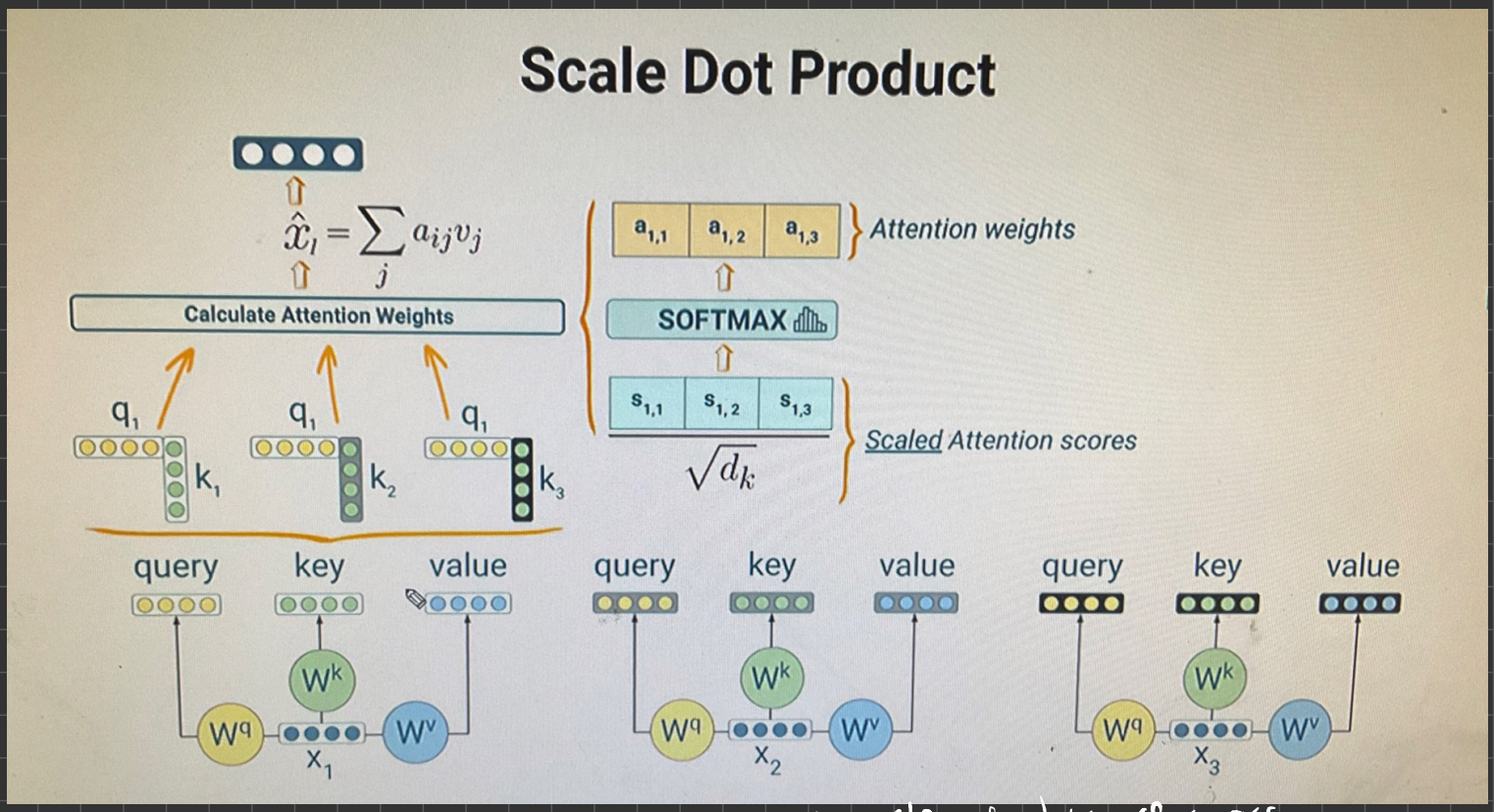
x_1

input



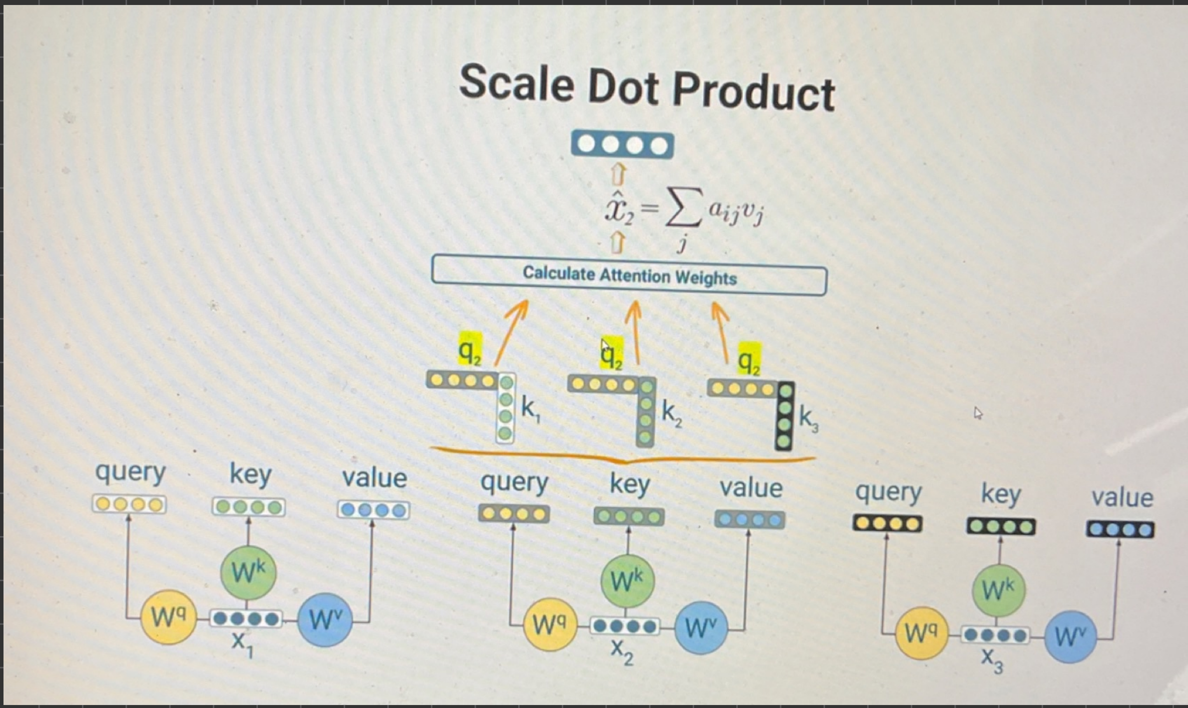


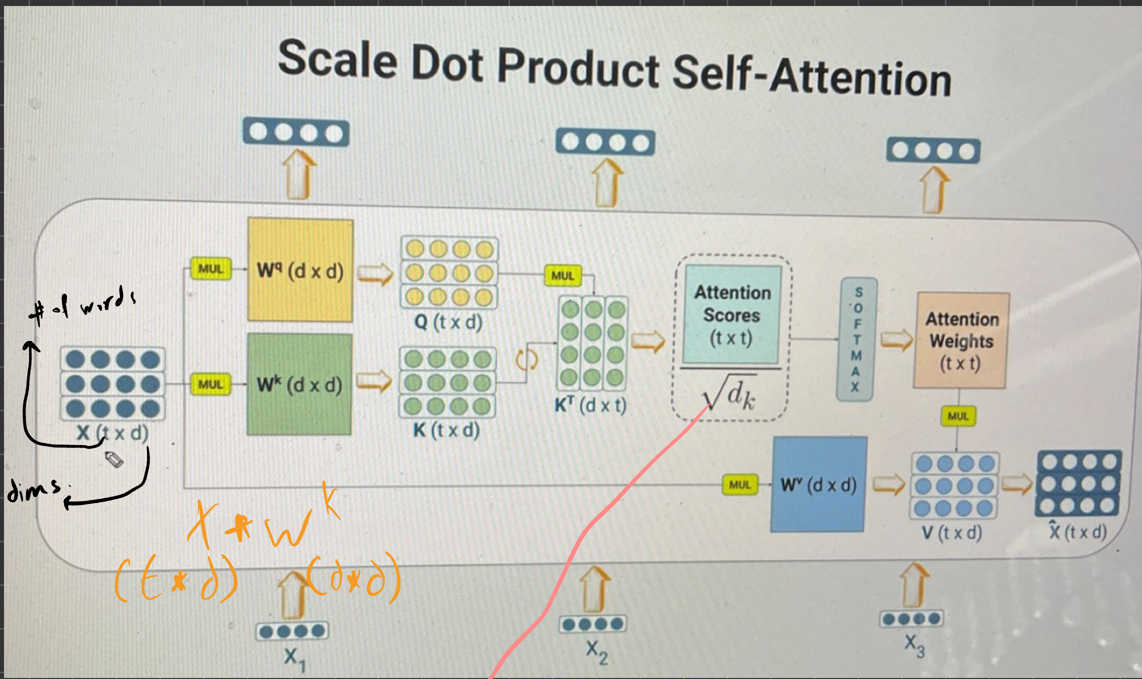
* the parameters that are need to learn during training to get good performance



example: find transformers videos.

* query: what you want. (searching)
* key: Person who doing it on youtube
* value: where is stored.





Attention Scores (t×t) = Q . K

example

$(5,5,5) = \sqrt{5^2+5^2+5^2}$

$\Rightarrow \sqrt{3 * 5^2}$

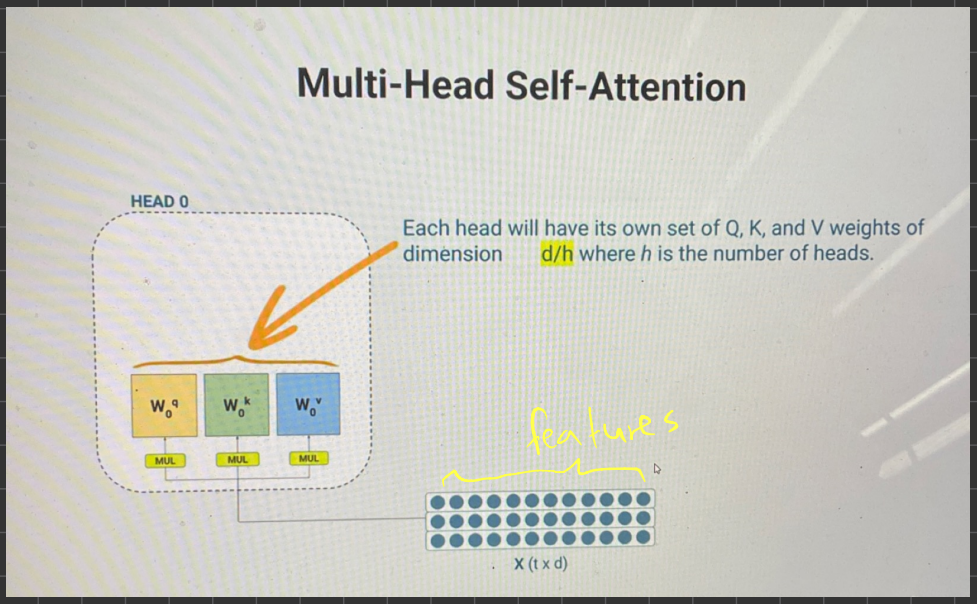
$= \sqrt{3} * 5^2$

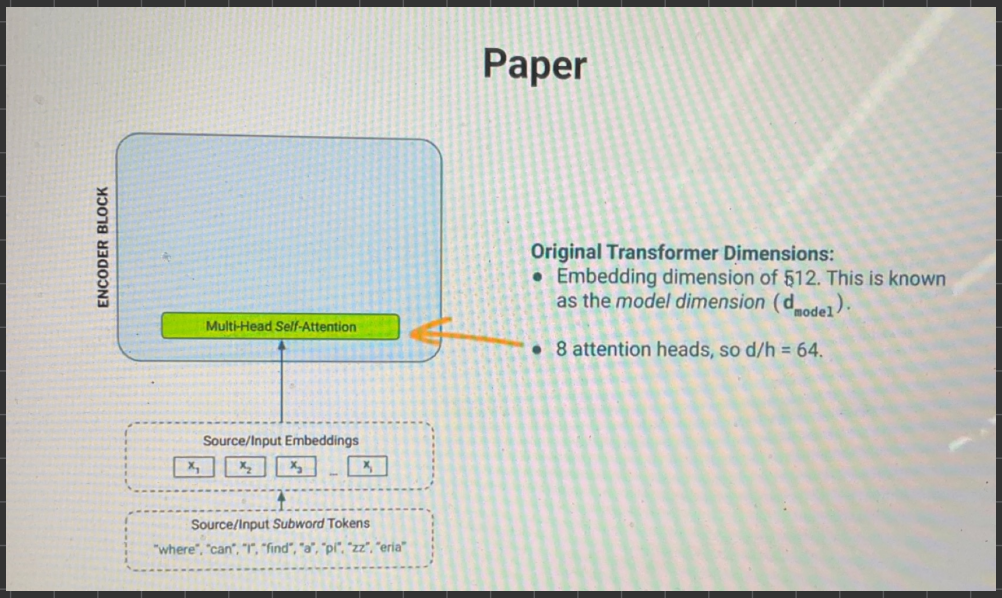
d_k : dimension of key.

$\Rightarrow$ Attention(Q, K, V) = $\text{softmax}\left(\frac{Q.K^T}{\sqrt{d_k}}\right).V$

مدى ترابط

الكلمات مع بعض





* Positional information :-

- index of the words.
- to know the squences of words.

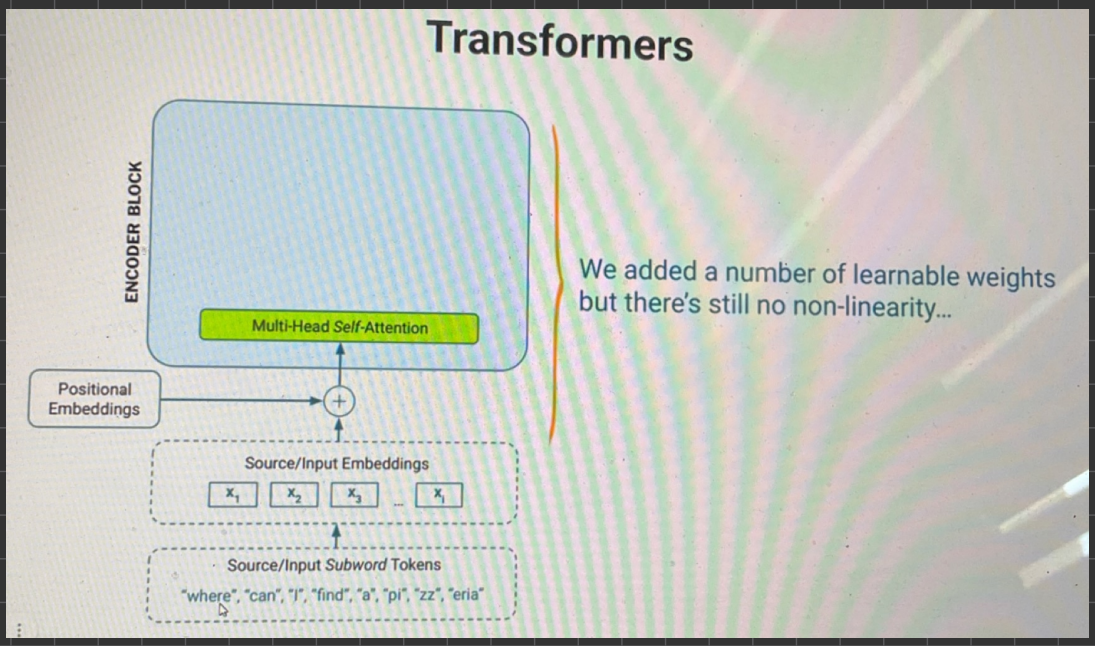
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

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

→ embedding size

even

odd



* in Transformers we can not use Batch normalization

Batch normalization is not typically used in transformers because it can disrupt the self-attention mechanism and hinder the model's performance. Transformers rely on the attention mechanism to capture dependencies between different positions in the sequence, and batch normalization can introduce dependencies between positions by normalizing each position independently. This breaks the ability of the self-attention mechanism to attend to different positions adaptively.

Additionally, transformers already have layer normalization, which is a similar technique to batch normalization but operates independently for each position in the sequence. Layer normalization helps stabilize the training process and improves the generalization of the model. It is used in transformers to normalize the input embeddings and the outputs of each transformer layer.

In summary, batch normalization is not used in transformers due to its interference with the self-attention mechanism, while layer normalization is employed to achieve the necessary normalization within transformers.