LLM Week-1

L1 Introduction to Transformer Architecture

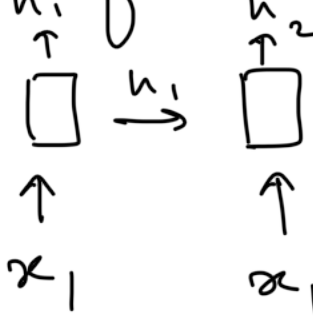
- previously RNNs were used, now transformers are more used.
- then came attn based building block.
- RNN based models

  eg. transla^n

  I → — — — — } in one go

  comput^n occurs step-by-step

  $h_1 = f(h_0, x_1)$ ... so on



} Encoder RNN

why calculating h_2 only after h_1?

↳ to know from the context of the sentence. Contextual represenṭa^n of every word is impt.

↳ Cons → can't do parallel processing. wasting of resources.

I want the bidirectional, not recurrent in nature.

Decoder RNN → also gives the sequential output.

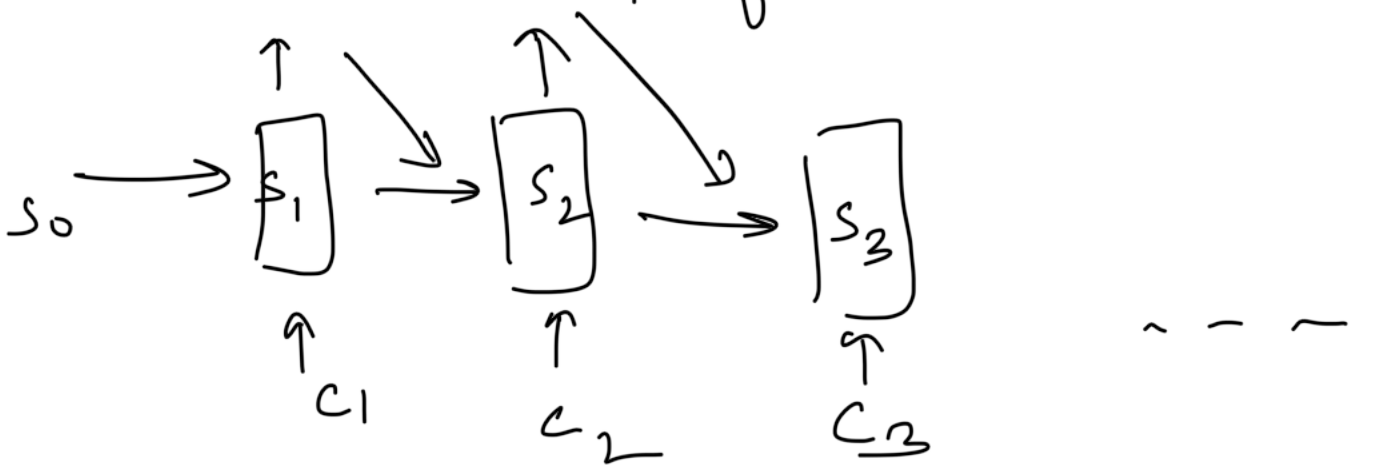
We can speed up the Encoder part.

- The final state is called the a concept vector, thought vector, or content vector or annotation.
- However, we still don't care about the alignment of words btw. source sentence & target sentence. Therefore, we will be needing a attention mechanism to help us resolve this issue.
- I compute the RNN encoding. I have the vectors $h_1, \ldots$ with me. We can drop the encoder block.

$$c_t = \sum_{i=1}^{n} \alpha_{ti} h_i \quad ; \text{ context vector for output}$$

$n \Rightarrow$ no. of words

$t \Rightarrow$ time step for decoder



- we generally look at a heatmap, when dealing with transformers.

$$\alpha_{ti} = align(y_t, h_i)$$

$$= \frac{\exp(score(s_{t-1}, h_i))}{\sum_{i'=1}^{n} \exp(score(s_{t-1}, h_{i'}))}$$

Ques: Can α_{ti} be computed parallely for all i at time step t?

Yes $\Rightarrow$ h_i is available for all i & s_{t-1} is also available at time step t.

Attenⁿ can be parallelized

Ques: Can h_i for all i can be computed parallely? $\Rightarrow$ No

- We need parallel mechanism at same time get rid of vanishing gradient problem.

L2 Attenⁿ is all you need

- we have recurrent connecⁿ $\Rightarrow$ the RNN + attenⁿ model.

- transformer architecture

word embeddings

self attn
↓
feed forward NN

→ self attn
encode - decoder attn
feed forward network

- with the help of bidirectional LSTM.

↓
result

- self attn
takes word embeddings are creates the h_is. The attn requires a pair of vectors as input.

$$\alpha_{ti} = \mathcal{J}_{ATT}(s_{t-1}, h_i)$$

s_1 s_2 s_3 ...
↑ ↑ ↑
self-attn
↑ ↑ ↑
h_1 h_2 ...

$$s_2 = \sum_{j=1}^{T=5} \alpha_{2j} h_j$$

as you can see there is no recurrence here.

- attn is needed because "it" should refer to more attn to the contextual word.

⇒ Self-Attn (Attn mechanism)

- given a word, we wants to compute the relational score btw. the word & rest of the words in the sentence. It will be a T x T matrix.

- Now both the vectors (s_i) & (h_j) for all i, j are available for all the time. (whereas it was not so in the seq2seq model.)
- Score func^n of seq2seq

$$score(s_{t-1}, h_j) = v_a^T \tanh(U_{att} s_{t-1} + W_{att} h_j)$$

3 vectors $\Rightarrow (s, h, e)$, 2 linear linear transformation, 1 non-linearity finally the dot-product.

However the input the self-attention module is only the word embeddings h_j for all j.

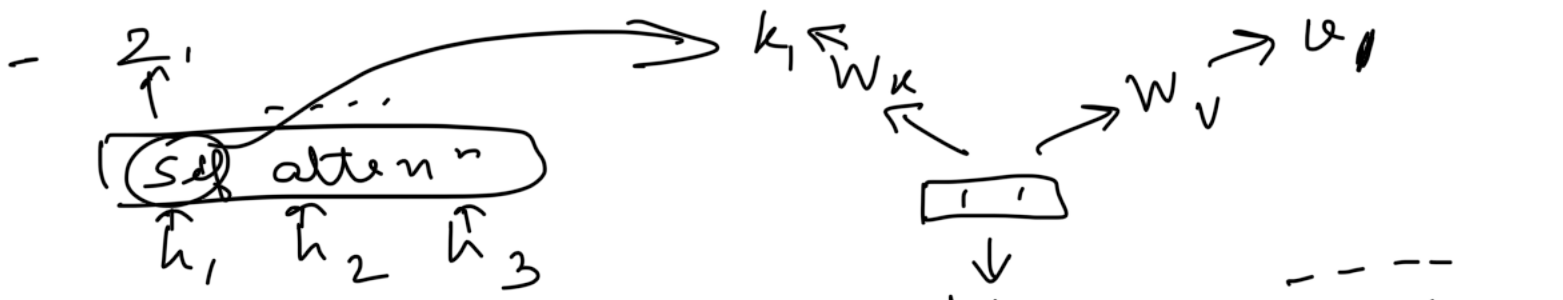
We need matrix transformation.

L3 Self-Attention

- transformation of matrices

$$q_j = R^d = R^{d \times d} h_j \quad v_j, k_j$$

(W_Q)



do for all the words

$$\text{score}(s_{t-1}, h_j)$$
$$\text{score}(q_1, k_j)$$

fixed ← → variables

→ func^n is a dot product

$$\alpha_{1j} = \text{softmax}(e_{1j})$$

$$e_1 = [q_1 \cdot k_1, \; q_1 \cdot k_2 \; \ldots \; q_1 \cdot k_5]$$

$$z_1 = \sum_{j=1}^{5} \alpha_{1j} v_j$$

- Can we vectorize all the computa^{ns} and find the outputs in one go.

$$\begin{bmatrix} q_1 & q_2 \ldots q_T \\ | & | \end{bmatrix} = W_Q \begin{bmatrix} h_1 & h_2 \ldots h \\ | & | \end{bmatrix}$$

$Q \in R^{64 \times T}$

$W_{(64) \times 64}$ $[h_T]$ $64 \times (T)$

$d_1 \times d$ $\quad d \times T$

$d_1 \times T$

similarly it goes for K and V.

- $Z = [z_1, z_2 \cdots]$

$$= \text{softmax}\left(\frac{Q^T K}{\sqrt{d_k}}\right) V^T \quad [\,]_{T \times T}$$

$Q \Rightarrow T \times 64$
$K \Rightarrow 64 \times T$
$V \Rightarrow d \times T$

} atten^n matrix $T \times d$

d_k is the dim of key vector

d_k scales the values of $Q^T K$.

↳ scaled dot product!

Matmul

↑

Softmax

↑

Scale: $\frac{1}{\sqrt{d_k}}$

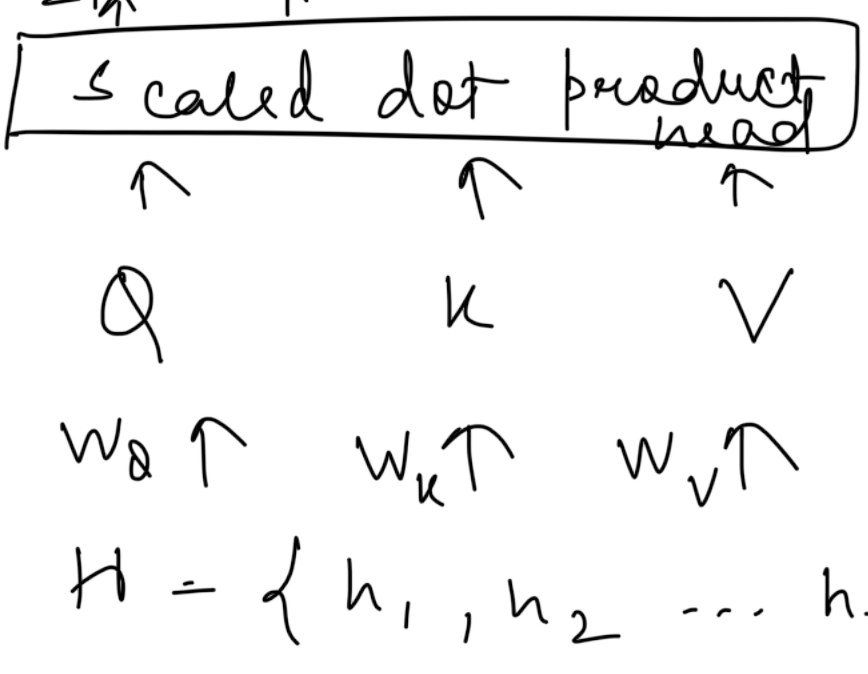
↑

Matmul: $Q^T K$

↑ Q ↑ K ↑ V

→ self atten^n block

L4 Multi Headed Attn^n



they take care of contextual info.

this is one 'block', we can have 2 or multiple head blocks.

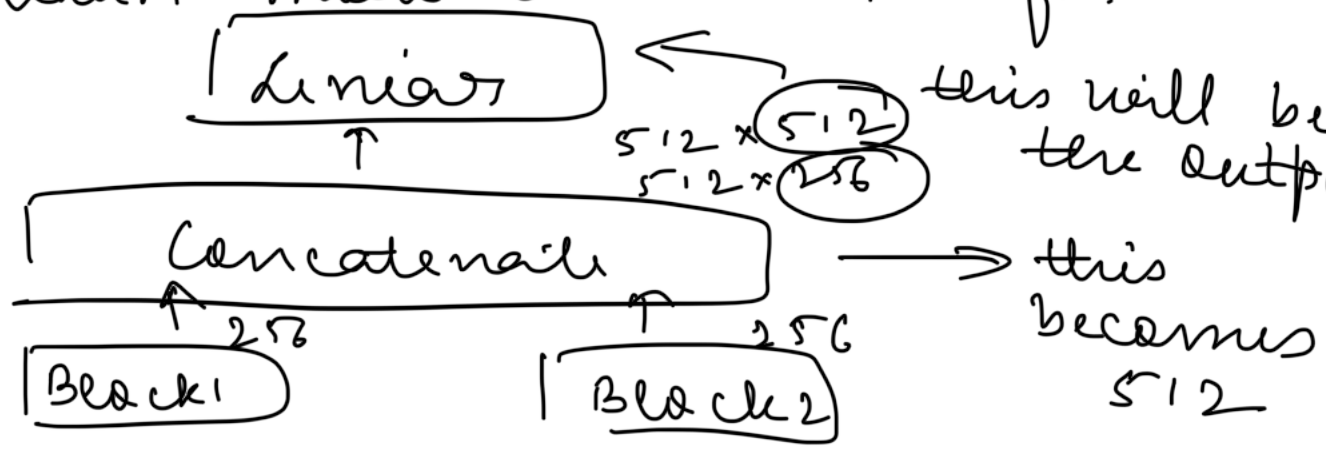
But what is the significance of multiple blocks?

Just like CNN, we have multiple filters that would capture diff. inpt. points.

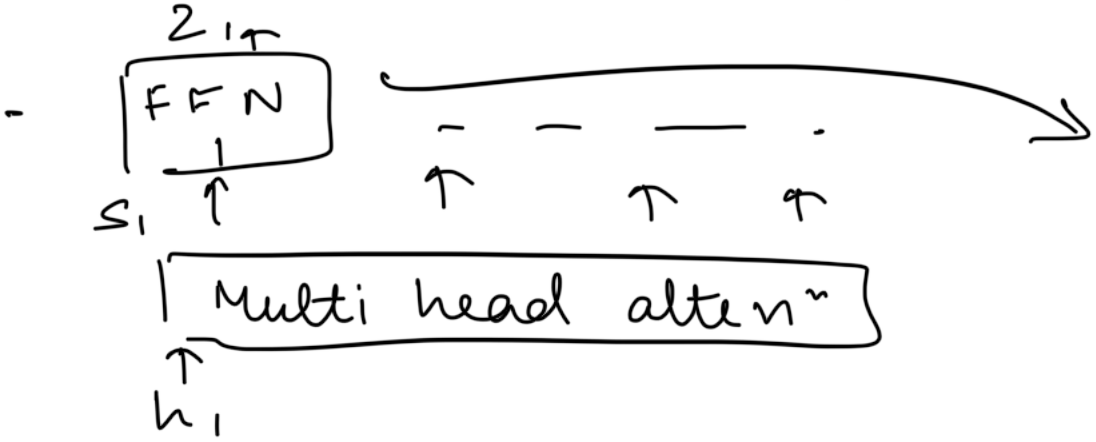
Similarly, if there a attn^n blocks, first will relate 'it' with animal and the second will capture the info. in 'was'.

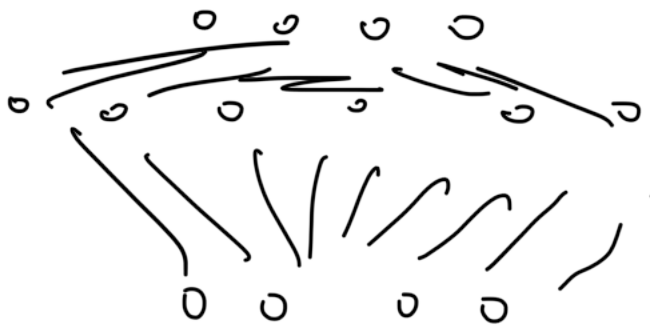
In the same way, diff. attn^n block we calculate α_s, the block will

learn more contextual info.



if there are 8 blocks then they will 64 size, then again concatenates gives 512.

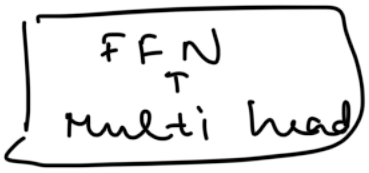




→ project to big size and again compress back to the original size.

$$FFN(z) = \max(0, W_1 z + b_1) W_2 + b_2$$

-



→ One encoder block, there will many blocks.

- Please note, parallelism is horizontally and not vertically. So basically in layer 1 of encoder ~~that~~ computation will occur parallelly. But we will waiting for the step-by-step process layer after layer.