Transformer N/w's : www.peterbloem.nl/blog/transformers

Self attention ⟹ Seq 2 Seq operation

machine translation and text summarization

(lengths of i/p & o/p may be different)

$(x_1, x_2, x_3 \cdots x_t)$ $\quad$ $(y_1, y_2, y_3 \cdots y_t)$

(K-dimensional vector)

$x_i \equiv$ K-dim $\rightarrow$ (X)

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & \cdots & x_{1K} \\ x_{21} & x_{22} & x_{23} & \cdots & & x_{2K} \\ x_{31} & x_{32} & x_{33} & \cdots & & x_{3K} \\ \downarrow & \downarrow & \downarrow & & & \downarrow \\ x_{t1} & x_{t2} & x_{t3} & \cdots & & x_{tK} \end{bmatrix}$$

$(t \times K)$: rows, Columns

(Similarly we have (Y) matrix)

Any $\boxed{y_i = \sum_{j=1}^{t} w_{ij} x_j}$ (weighted average over all input vectors)

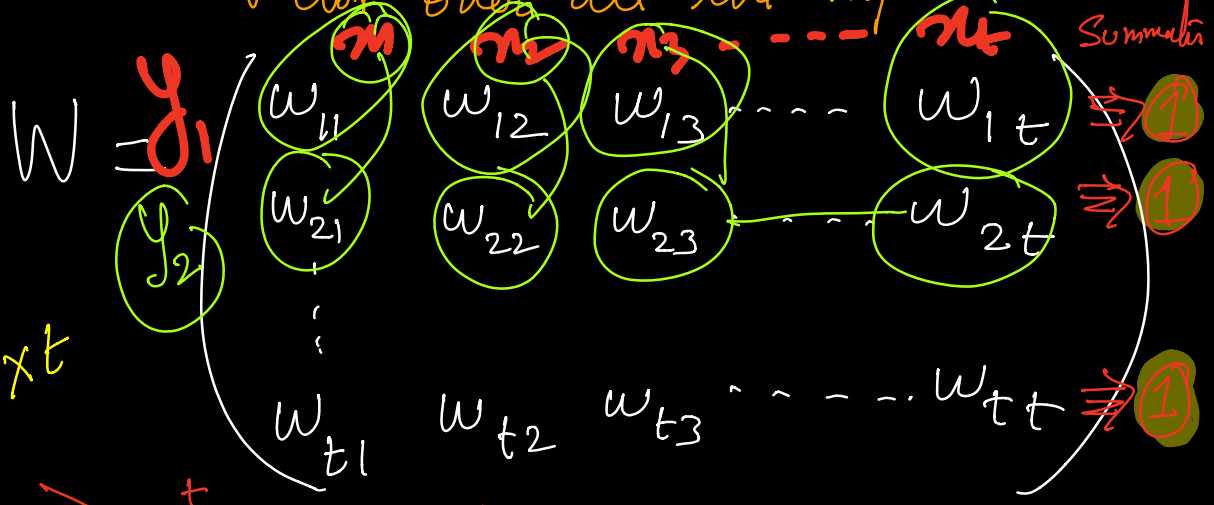
$\therefore y_i = w_{i1} x_1 + w_{i2} x_2 + w_{i3} x_3 + \cdots w_{it} x_t$

$\forall\ i = 1 - t$ $\quad$ scalar, vector, vector addition

∴ Just like $X_{t \times k}$ & $Y_{t \times k}$ matrix

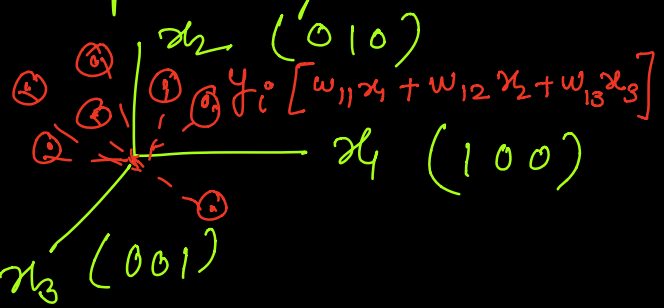
there is a $W_{t \times t}$ matrix.

Any y_i is nothing but the weighted averaged vectors over all the input vectors



$$\Rightarrow \sum_{j=1}^{t} w_{ij} = 1$$

One can say that x_i's can be seen as the basis and the space spawn by them will hold the (y_i's)



This weight matrix $W_{t \times t} = (w_{ij})$ $\forall i = 1 - t$, $\forall j = 1 - t$

How related 2 vectors are. This is Cosine similarity

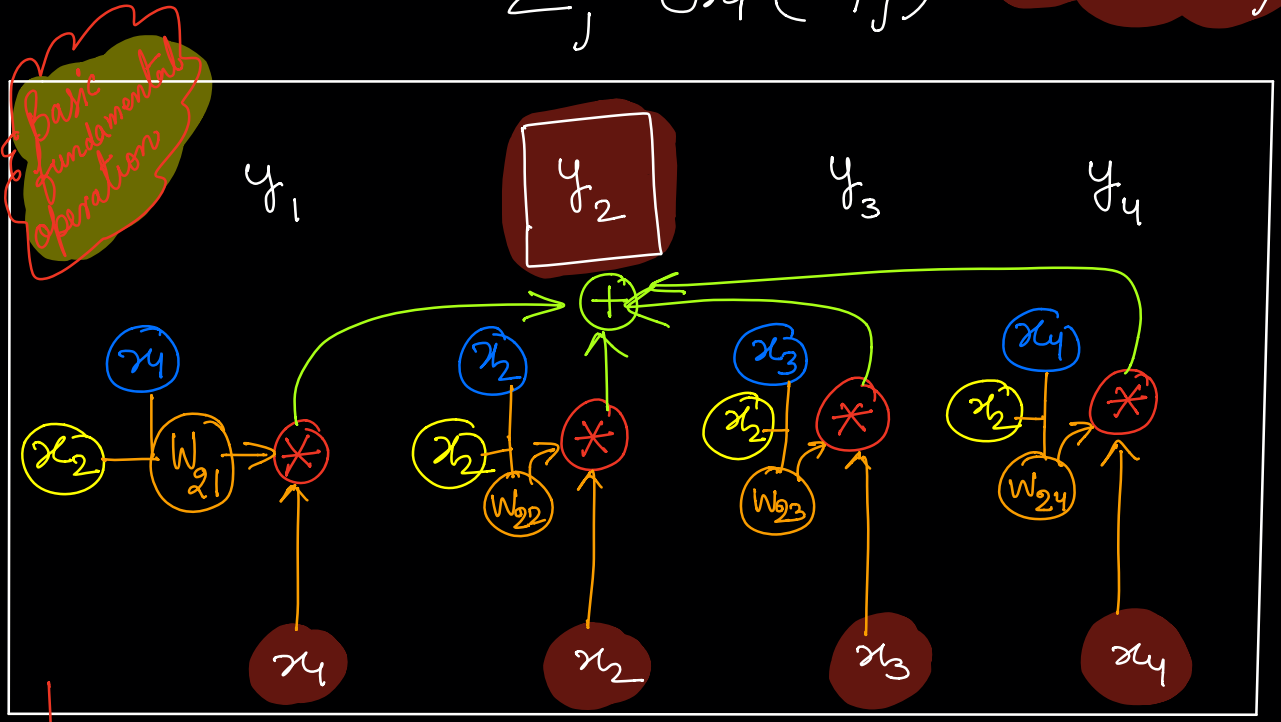
$$w'_{ij} = f(x_i, x_j) = x_i^T x_j$$

Its value can range from $(-\infty, \infty)$

hence in order to normalize it apply Softmax.

$$w_{ij} = \frac{\exp(w'_{ij})}{\sum_j \exp(w'_{ij})}$$

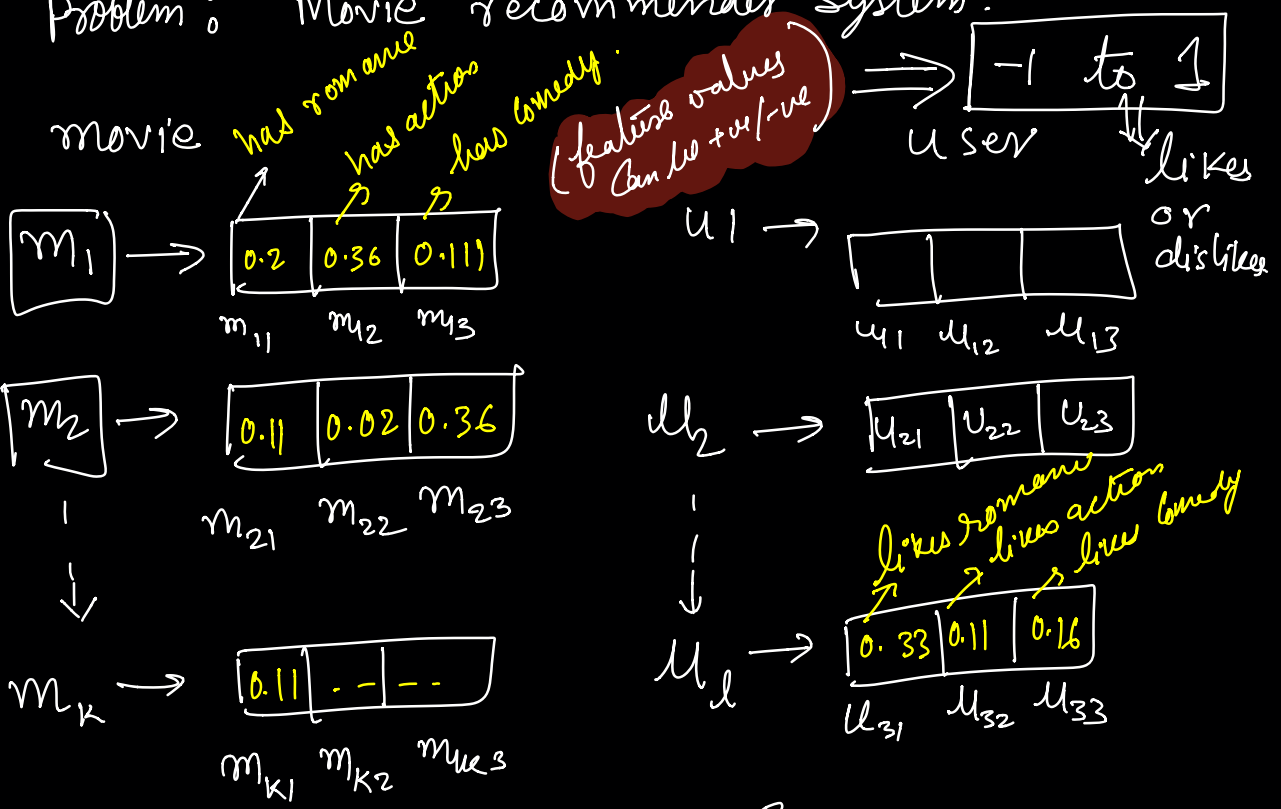
Row-wise normalization



↳ Only operation in transformers that propagates information between the vectors. Other than self attention all operations are applied on individual inputs.

Why do self attention works :

Problem : Movie recommender system.



$$\therefore \sum_{i=1}^{K}\sum_{j=1}^{l} m_i u_j \Rightarrow \underset{(i,j)}{Score} = \sum_{K=1}^{3} m_{iK}\, u_{jK}$$

$m_{iK}\, u_{jK}$: if matches seems user likes the movie Score get boosted

Basically each movie can be seen as a feature vector as well as each user " " " " " " of same dim.

dot product ⇒ Cosine similarity is being computed.

⇒ But such feature annotation need to be done manually as for million of movies and atleast extraction of (k) features ⇒ Impractical

Instead of manual annotation we will let it to be learned automatically. fix up the hyperparameters and learn these parameters.

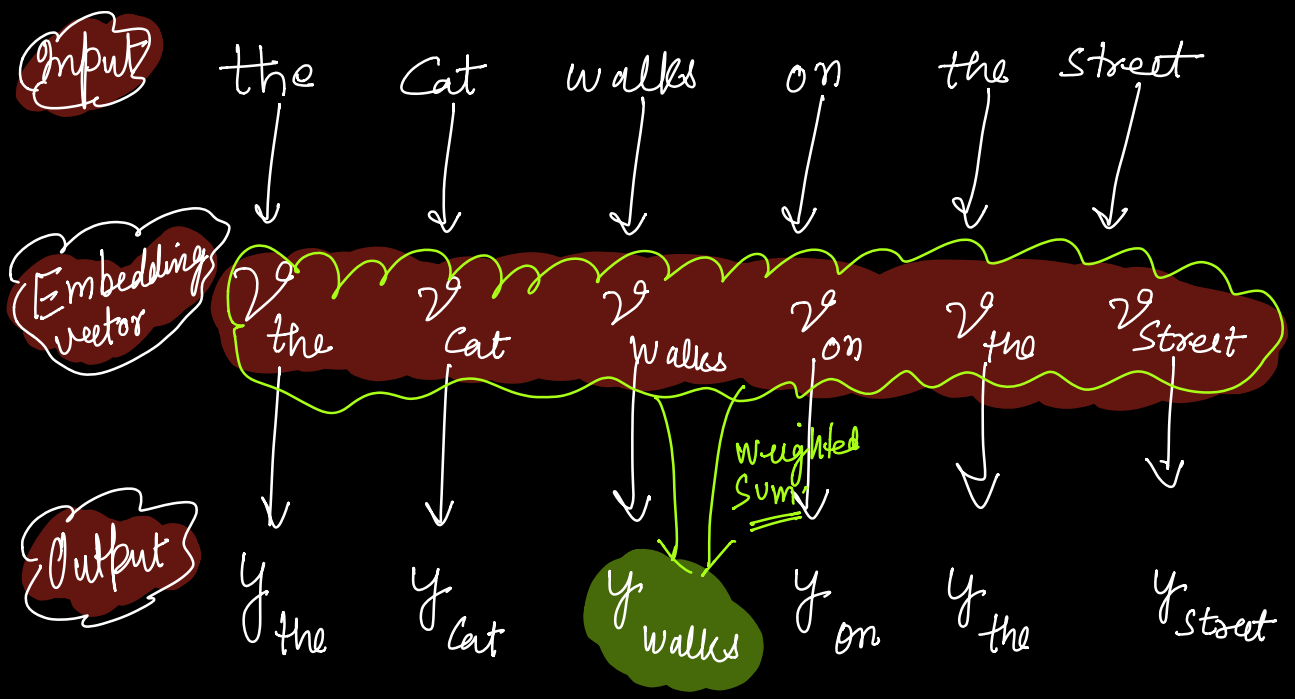
↳ ① movie & user ⇒ feature need to be learned

② taking it towards Supervised learning

③ Get a small dataset with the class/annotation and learn the feature that can be seen as the best representation. ↳ Converting into a supervised optimization problem

learning the embedding

⇒ Can also be done using basic matrix factorization → Self attention Basic



$\Rightarrow$ We need to learn $[v_{word}]$ embedding vector for the words

v_t

Similar words requires embeddings which are close to each other (related)

related words $\Rightarrow$ Related / Similar embeddings

But relatedness completely depends upon the optimization task

The problem in hand $\Leftrightarrow$

Relatedness captured using dot product

O/p is weighted sums over the whole input sequence with weights determined by these

Dot product $(x_1 \cdot x_2)$ ⟹ How much these two vectors (x_1 & x_2) are related

Currently we are trying to do [seq 2 seq] operations

↳ No parameters learned for Self attention

Only word encoding are determined till now and are used to get the "relatedness" ⟹ using dot product.

↳ Self attention sees its input as a set "NOT" ⟹ Sequence. (even though it is sequence)

all belongs to the input set (I)

This means that order does not matter and any permuted input sequence will also generate the same o/p.

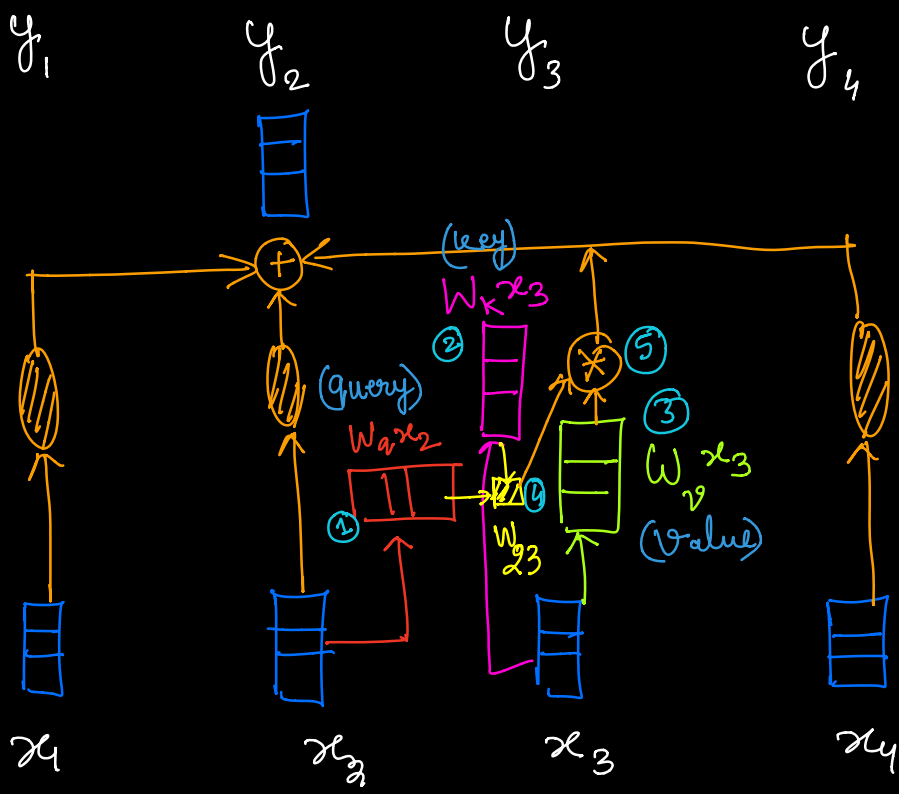
This issue is addressed in full transformer

permutation equivariant

Now introducing additional weights to be learnt for performance (modern transformers) and ③ tricks

(A) Query, Keys and values (Trick-01)

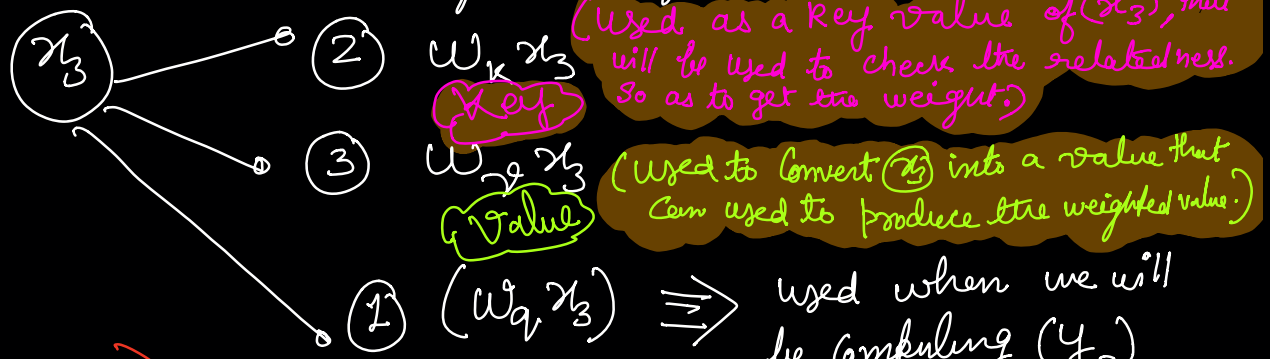
One can see that in the above transformer network any input vector (x_i) is used in (3) ways.



$(y_2) \Rightarrow$ Weighted sum of $[x_1, x_2, x_3, x_4]$

But weights (w_i) for each (x_i) depends upon how much it is related (dot product) to (x_2)

One can see that each x_i (Say x_3) is used in (3) different ways in self attention.



(All are K x K matrix)

W_q

Query

(Then we query each other x_i's how much related it is with (x_3). That will decide its weight)

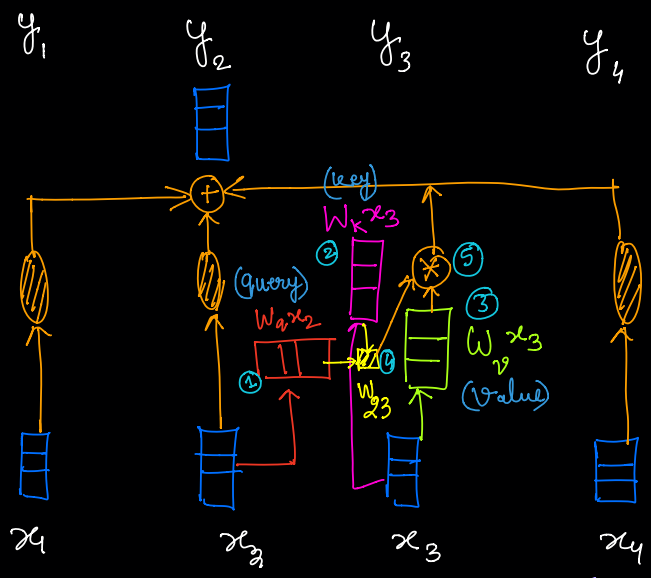
⊛ Query : $[x_i]$ Compared (for relatedness) using dot product with all other $[x_j$'s$]$, in order to establish the weights for its own $[x_i \rightarrow (y_i)]$.

W_k

⊛ Key : In order to participate in the generation of all $[y_{j \neq i}]$, again $[x_i]$ is used as a key and Compared with all $[x_{j \neq i}]$ to get its weighting Contribution

⊛ Value : Finally $[x_i]$ is also Converted into a value vector that can be multiplied with the weight to Compute each o/p weight.

W_v



For (y_1) Computation

⇓

a) x_4 query for relatedness using $[W_q x_4]$ with the key of all the input's i.e $[W_K x_i]$ $\forall i = 1-4$.

b) Now once all the relatedness of (x_4) with all other $(x_i's)$ got computed in $[W_{1j}] \forall j=1-4$

$[x_4]$ itself is converted into key and multiplied to

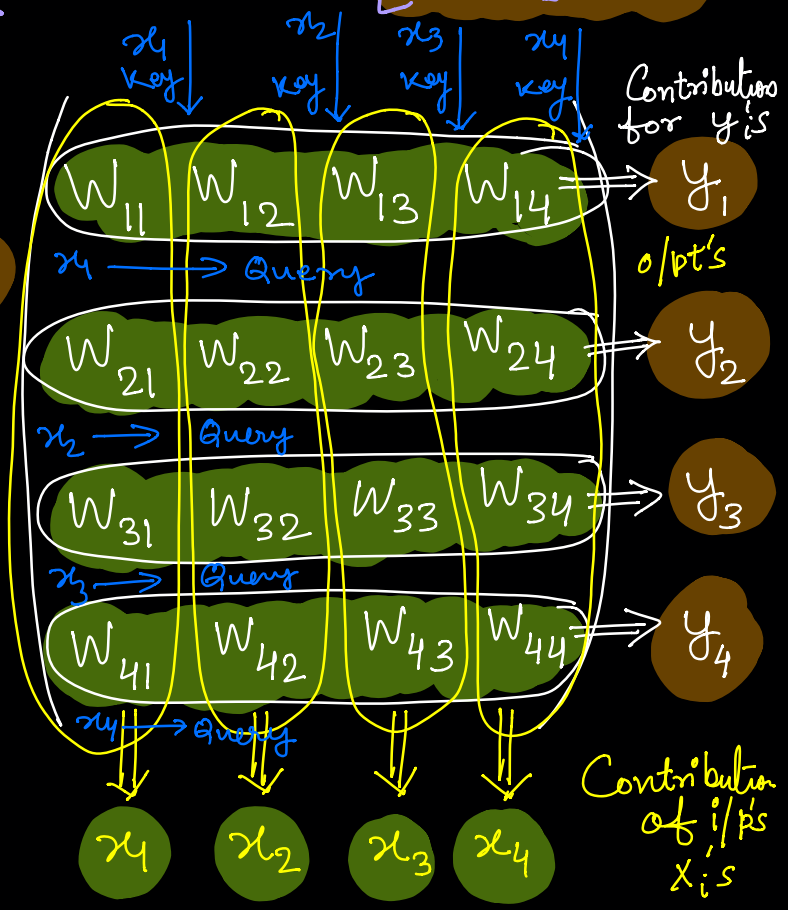
$\odot [W_{j1}]$ $\forall j = 1$ to 4

where $[W_{11}]$ Contributions of x_4 in y_1 → y_1

$[W_{21}]$ of x_4 in y_2 → y_2

$[W_{31}]$ of x_4 in y_3 → y_3

$[W_{41}]$ of x_4 in y_4 → y_4



$\therefore$ Horizontally $\Rightarrow$ Row(i) $\Rightarrow$ Query of $[x_i]$

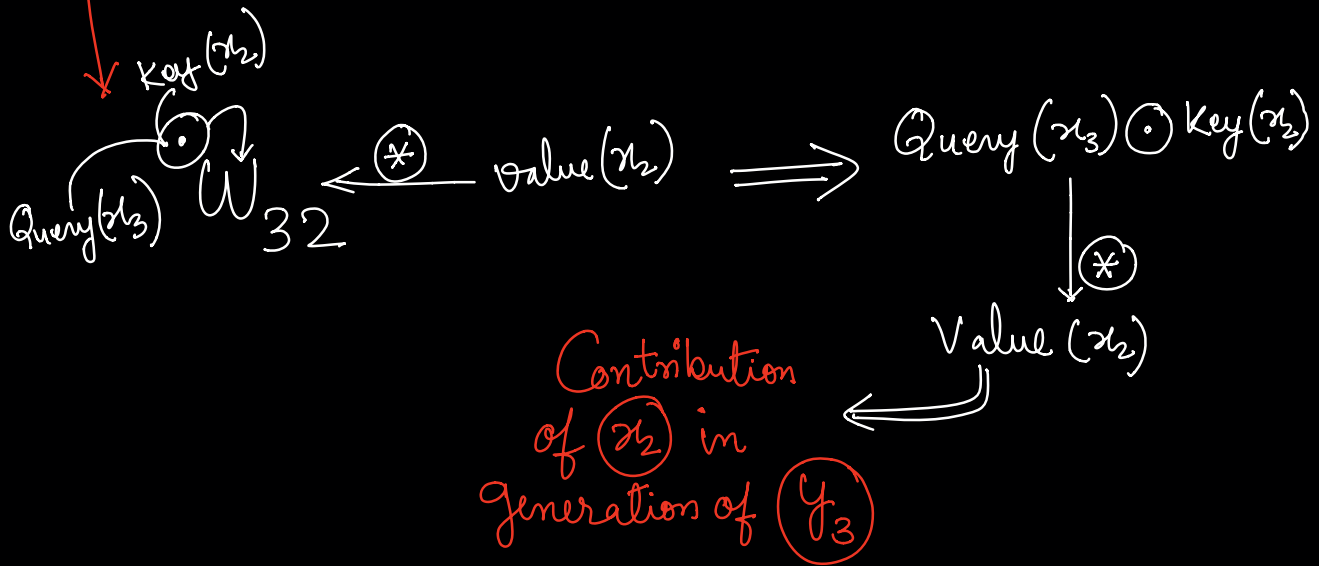
Vertically $\Rightarrow$ Col(j) $\Rightarrow$ Key of $[x_j]$

$$y_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4$$

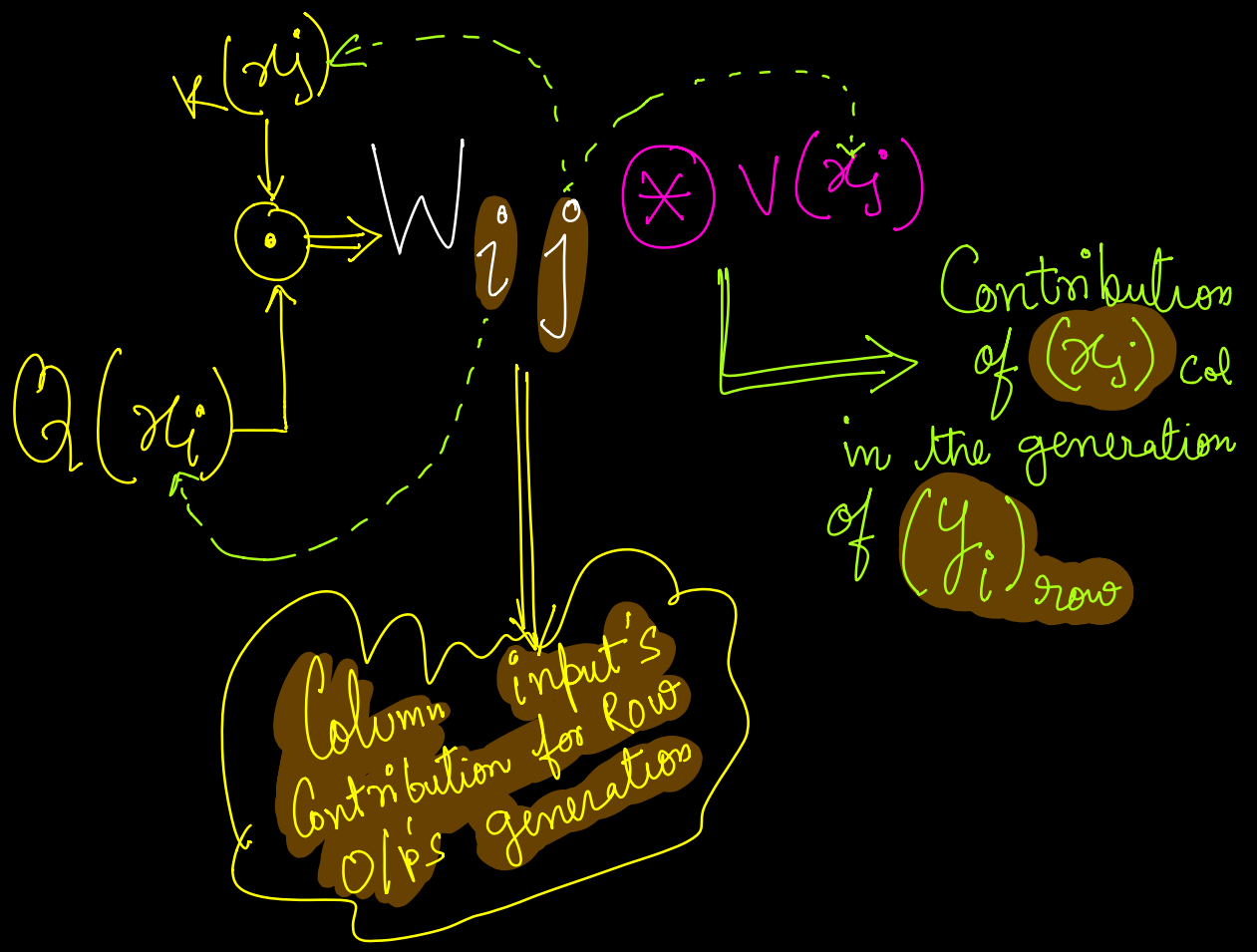
$$y_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4$$

$$y_3 = w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4$$

$$y_4 = w_{41}x_1 + w_{42}x_2 + w_{43}x_3 + w_{44}x_4$$



In general :



These W_q, W_k, W_v are all $K \times K$ matrix and can be seen as a linear transformation of and K-dim (x_i) input vector from $\mathbb{R}^K$ to some other K-dim transformed vector. These 3 transformation need to be learned in order to transform any input $[x_i]$, so that it can adapt to play its (3) different roles appropriatly for SA modules.

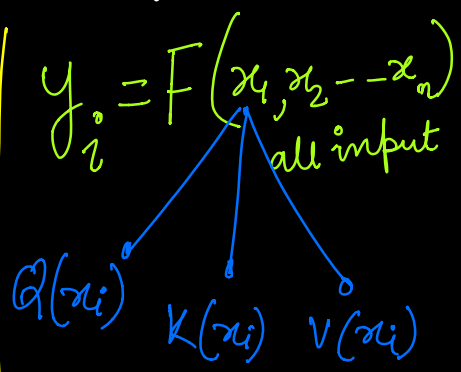
These $\boxed{W_q, W_k, W_v}$ can be used in many different ways to realize the attention, One such way is

① $\text{Query}(x_i) = W_q x_i$

② $\text{Key}(x_i) = W_k x_i$

③ $\text{Value}(x_i) = W_v x_i$

$y_i = F(x_1, x_2 \cdots x_n)$ all input



④ $\forall_{i,j} \quad w'_{ij} = \text{Query}(x_i)^T * \text{Key}(x_j)$

⑤ $\forall_{i,j} \quad w_{ij} = \text{Softmax}(w'_{ij})$

⑥ $y_i = \sum_{\forall j} w_{ij} * \text{Value}(x_j)$

The self attention layer got parameterzed using (W_q, W_k, W_v)

(B) Scaling the Dot product (Trick-02)

$$\forall_{i,j} \; W'_{ij} = \frac{Query(x_i)^T * Key(x_j)}{\sqrt{K}}$$

Average value of the dot product keeps on growing with the embeddings dimensions $[K]$. As finally $[W'_{ij}] \Longrightarrow$ got applied by Softmax this growth got exponentialed and may be effected adversarily.

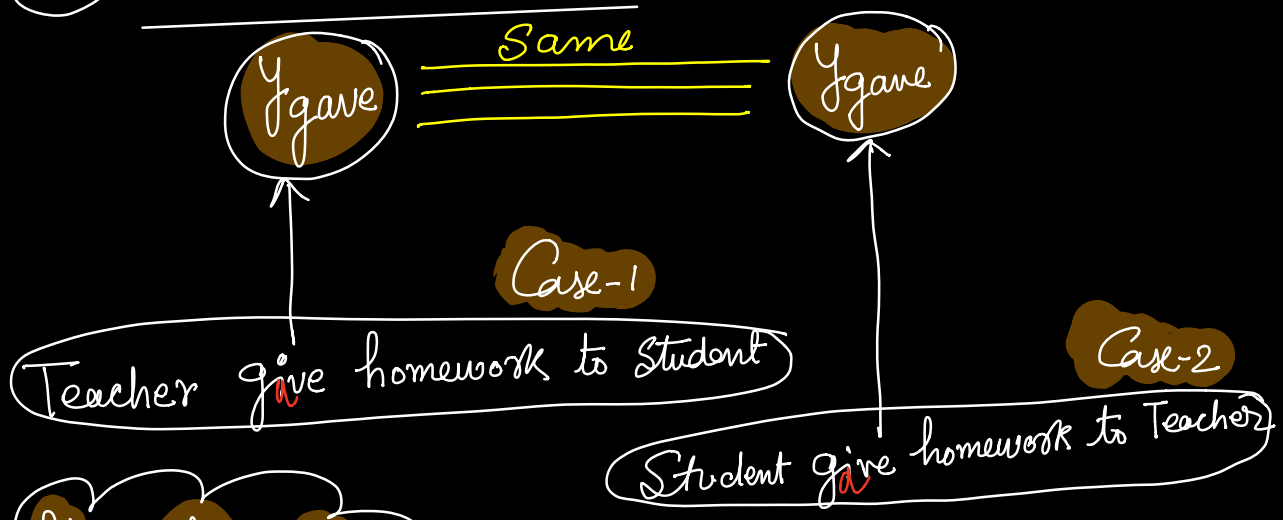
Hence we need to normalize it. Since our input vector $\in \mathbb{R}^K$ (Say all are $[C]$ $\longrightarrow \begin{bmatrix} C \\ \vdots \end{bmatrix}$

$\therefore C \xrightarrow[\text{(growth is } \sqrt{K})]{K\text{-dim}} \sqrt{K} * C$

Norm $\Rightarrow [\sqrt{K} * C]$

$\hookrightarrow$ Hence normalizing it by $(\sqrt{K})$.

(C) Multi-Head attention



If we have only one attention head W_q, W_k, W_v

But the Meaning in both Cases got Summed-up.

Since self attention ignores order and See input series as set the

Ygave → remains to be the Same in the both Cases. (All information got Summed-up.)

Same word mean differently to different neighbourhood/Context.

One self-attention head → information got Summed up (Averaged)

But adding more and more heads will provide more power to SA to address Contextual meaning rather than fixed

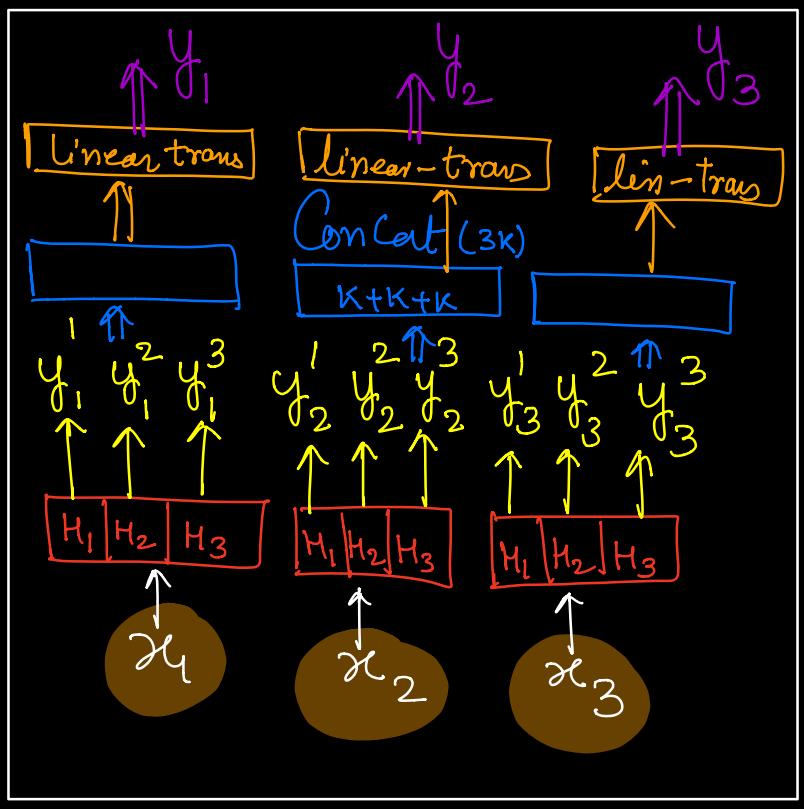
$H_1 \longrightarrow W_q^1, W_K^1, W_v^1$

$H_2 \longrightarrow W_q^2, W_K^2, W_v^2$

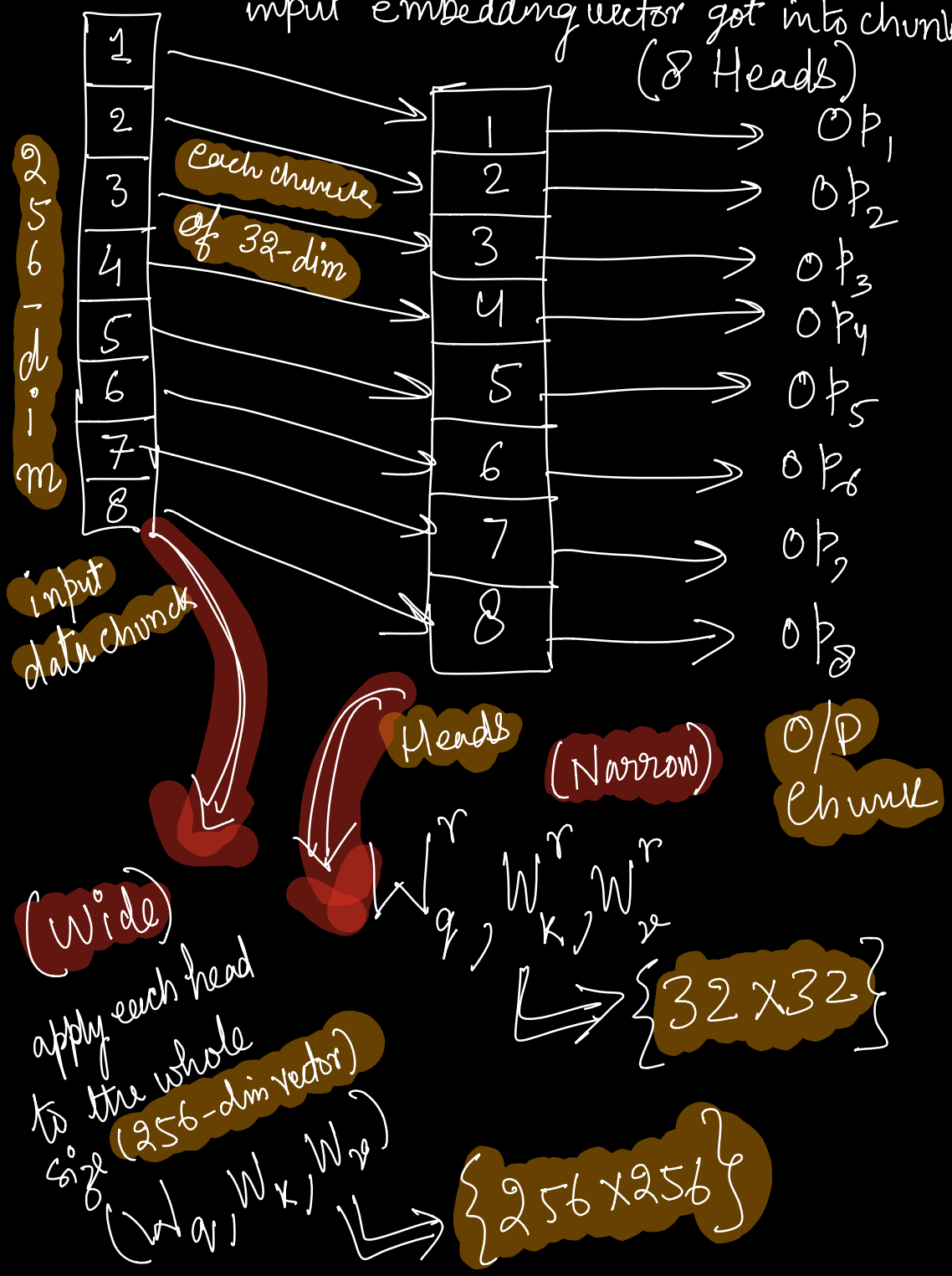
$\vdots$

$H_r \longrightarrow W_q^r, W_K^r, W_v^r$

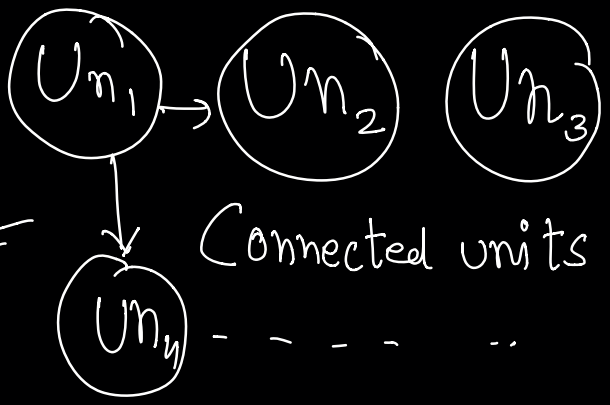
} Multiple attention heads.



① Narrow and Wide Self-attention

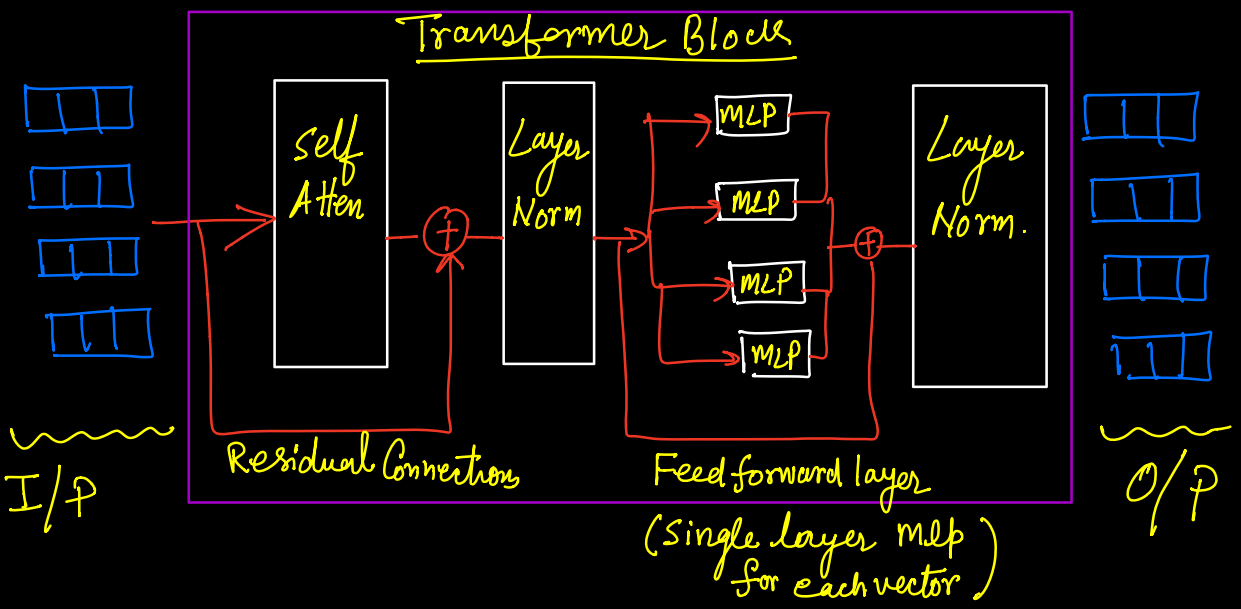


Transformers :



An architecture that can process a set of units (Sequence of tokens / pixels) where units can only interact via. self attention.

{Repeated self-attention}



* Normalization and Residual learning are standard tricks to train deep networks.

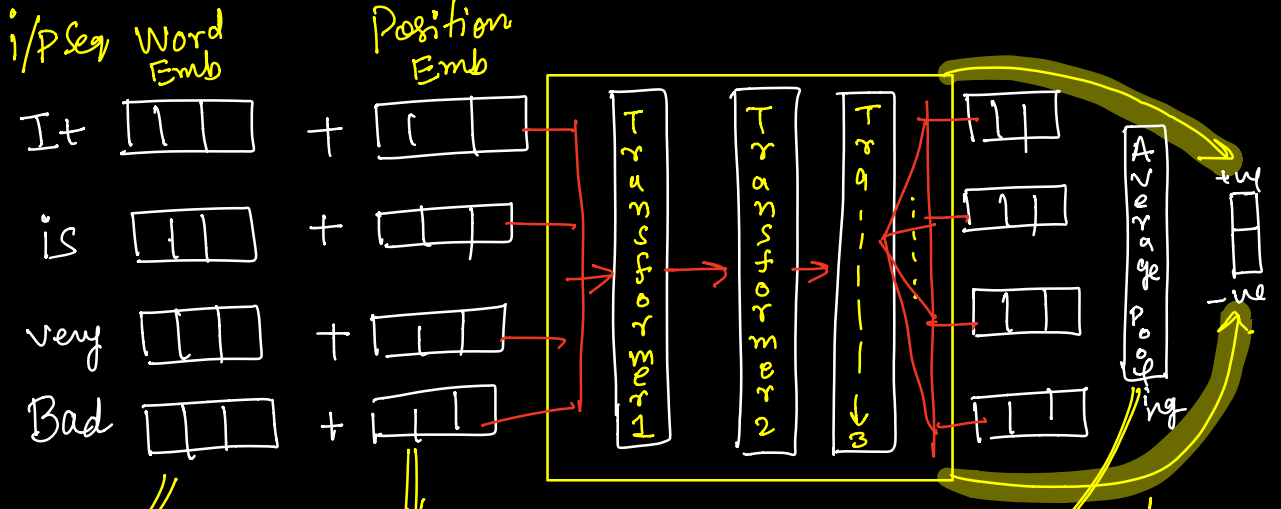
(I) Designing Classification Transformer

IMDb Sentiment classification dataset ⇒ tokenized in words (make a sequence classifier) for movie reviews

⇓

Binary Classification {+ve / -ve}

Transformer based classification



Word Embedding
Word2Vec
Glove

Can be learned as word embedding or seen as positional encoding using some complex mapping fn

(Global average pooling)

finally Softmax as Classifier

Requirement of positional embeddings :

The position of any word (w_i) in a sentence (S) plays a very crucial role

Any sentences can be seen as a complex time series with a well defined semantic associated to it

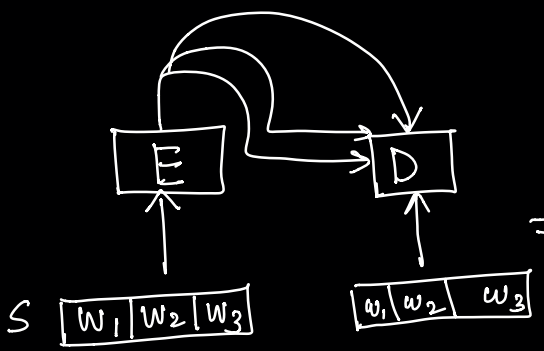
In RNN, the input got fed into the RNN Unit (sequentially). Hence the ordering information explicitly got embedded

Sequential Computation (slow)

In Transformers multi-head self attention completely ignores the word ordering. (Hence can capture longer dependency & fast)

Massively parallel (fast)

But in this process it looses the positional information of any word



The encoders in transformers extracts positional invariant encodings

$$\frac{w_1, w_2, w_3}{w_2, w_3, w_1} \Rightarrow E_1$$

Some basic options are :

(1) Decimal encoding for each (t).

$t = 0, 1, 2, 3, - - - - - 10,000$ (some big No)

(Normalize it b/w 0-1)

Bounded

But each i/p is of different size therefore $\Delta t \Rightarrow$ Gap between

(2) timestep is not consistent across all sentences

Unbounded

Do not generalize for longer length sequences

testing sentences may be longer

Δt is consistent.

Basic Criteria's for encoding

(1) Position of a word in any sentence must have to be unique. (Unique time step encodings)

(2) Distance b/w any two timesteps (Δt) must have to be Consistent across all length sequence

(3) Encoding need to be bounded and generalize well for longer sequence

(4) It has to be deterministic and fixed for any given i/p. (Should not be a random fn).

Hence, for each time step (t), in any i/p sentence (S) we require a function $f : \mathbb{N} \longrightarrow \mathbb{R}^d$

$t \in \mathbb{N}$

d-dimentional positional encoding

$\vec{p}_t^{(i)} = f(t)^{(i)}$ $\qquad$ $\vec{p}_t \in \mathbb{R}^d$

i^{th} element of the positional embedding for time (t).

function (f) takes (t) as an i/p and gives a d-dim p.e

$[i]$ ranges from : $1, 2, 3, 4, \dots, d$

$[k]$ ranges : $k=1$ $\quad k=2$ $\quad k=3$ $\quad k=d/2$

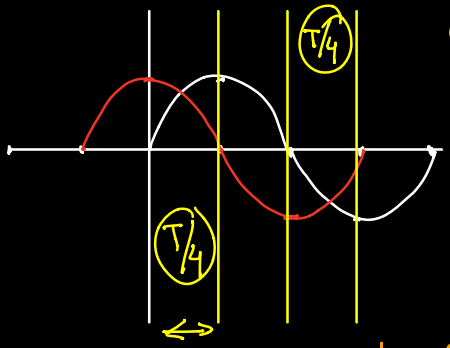
$\sin(w_t k)$ when $i = 2k$ i.e i = even

$\cos(w_t k)$ when $i = 2k+1$ i.e i = odd

$$\vec{p}_t = \begin{bmatrix} \sin(w_1 t) \\ \cos(w_1 t) \\ \sin(w_2 t) \\ \cos(w_2 t) \\ \sin(w_3 t) \\ \vdots \\ \sin(w_{d/2}\, t) \\ \cos(w_{d/2}\, t) \end{bmatrix} =$$

* Basically utilizing Sinusoidal function with a varying (w)
* one (w) encodes (2) dimensions using (Sin) & (Cos) each.
* Therefore a total of ($d/2$) frequencies are used to encode (d) dimensions.

Q1: Why do you require more than one Sinusoid function.



only $(T/4) + (T/4) \Rightarrow (T/2)$ ⇒ Any one Sinusoid can give us unique values. after that values keep on repeating.

⇒ Instead of a single Sinusoid if one can use (2) Sinusoid may be at same freq then these combination may lead to produce unique encoding for each timestep But ⇒ it will be (2-D) encoding.

$\omega = 2\pi f$

$f = \frac{1}{T}$ & $\omega = \frac{2\pi}{T}$ also $f = \frac{T}{2\pi}$

Hence $\sin(\omega_1 t)$ & $\cos(\omega_1 t)$ ⇒ Cover upto $\left(\frac{2\pi}{\omega_1}\right) = T_1$ Time period w/o repeating.

Adding more Sinusoids will increase the time period w/o repeating the sequence.

As Suggested in paper $\omega_k = \frac{1}{(10{,}000)^{2k/d}}$

$K=0$, $\omega_0 = 1$, $f_0 = \frac{1}{2\pi}$, $T_0 = 2\pi$ (one Sin & one Cos for each K)

$K=1$, $\omega_1 = \frac{1}{(10000)^{2/d}}$, $f_1 = \frac{1}{2\pi(10000)^{2/d}}$, $T_1 = 2\pi(10000)^{2/d}$

$K=2$, $\omega_2 = \frac{1}{(10000)^{4/d}}$, $f_2 = \frac{1}{2\pi(10000)^{4/d}}$, $T_2 = 2\pi(10000)^{4/d}$

$K=3$, $\omega_3 = \frac{1}{(10{,}000)^{6/d}}$, $f_3 = \frac{1}{2\pi(10000)^{6/d}}$, $T_3 = 2\pi(10000)^{6/d}$

finally $K = d/2$

$$K = d/2, \quad \omega_{d/2} = \frac{1}{(10,000)}, \quad f_{d/2} = \frac{1}{2\pi(10,000)}, \quad T_{d/2} = 2\pi(10,000)$$

$$\therefore \; T_0, T_1, T_2, T_3, \; - - - - \; T_{d/2}$$

$$2\pi, \; 2\pi(10000)^{2/d}, \; 2\pi(10000)^{4/d}, \; 2\pi(10000)^{6/d}, \; 2\pi(10,000)$$

(*) Systematically increasing the time period of the sinusoid from $2\pi \longrightarrow 2\pi(10,000)$ [Equi-distant]

(*) It is a GP with $r = (10000)^{2/d}$

(*) The d-dim encoding is going to be repeated after

$$LCM(T_0, T_1, T_2 - - - T_{d/2}) = T_{d/2} = 2\pi(10000)$$

LCM of a GP is its final term.

t=3 | 1 2 3 | 1 2 3 | 1 2 3

t=2 | 1 2 | 1 2 | 1 2 | 1 2 |

after (6)

t=3 | 1 2 3 | 1 2 3 | 1 2 3 | 1 2 3 |

t=4 | 1 2 3 4 | 1 2 3 4 | 1 2 3 4 | 1 2 3 4

after (12)

Q2: Why dividing $2\pi \longrightarrow 2\pi(10,000)$ into $(d/2)$ when even LCM of $(2\pi, 2\pi(10,000)) = (2\pi)(10,000)$.

Instead of taking a few why to take $(d/2)$ frequencies.

Before that let us see how to use these positional encodings.

for any word say (w_t) in any sentence $(S) = \{w_1 \cdots w_n\}$

$$\underbrace{\psi'(w_t)}_{\text{final embedding fed to the N/w}} = \underbrace{\psi(w_t)}_{\text{Word embedding}} + \underbrace{\vec{P}_t}_{\text{position embedding}}$$

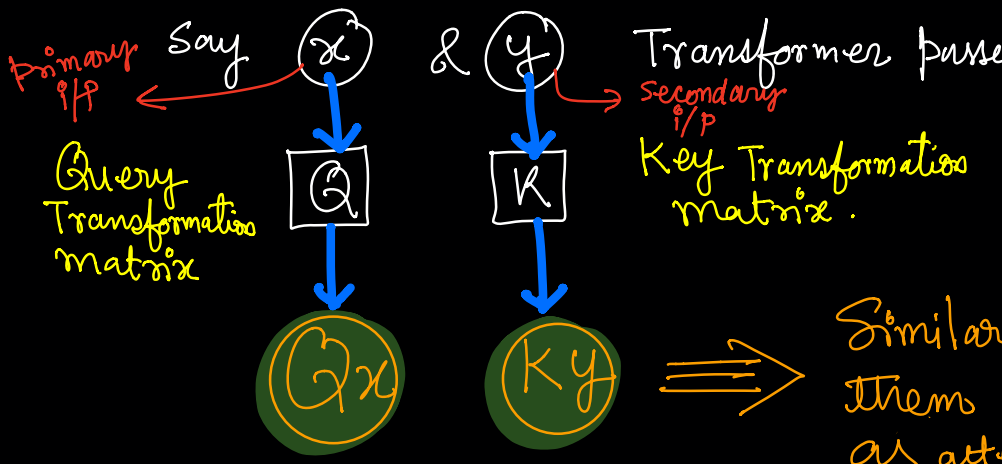
In order to make this summation possible dimensions of word embeddings and position embeddings are set as equal.

They can also be concatenated instead of addition (that is more intuitive)

⇓ Few reasons supporting additions:

1. It requires less memory.
2. With max_timescale = 10,000, (10,001) word has the same encoding as first word.
3. Assuming any seq of length (20 which is common) with word embedding of 512 dims, most of the position embedding dimensions will be used/dominated by world embedding (WE).
4. (WE + PE) got trained from scratch in transformer.
   - ↳ This enable not to encode important informations in lower dimensions of WE.
     - ↳ for smaller sequence this is almost like concatenation happening.

(5) In order to learn the attention b/w 2 given embeddings say x & y Transformer passes them into



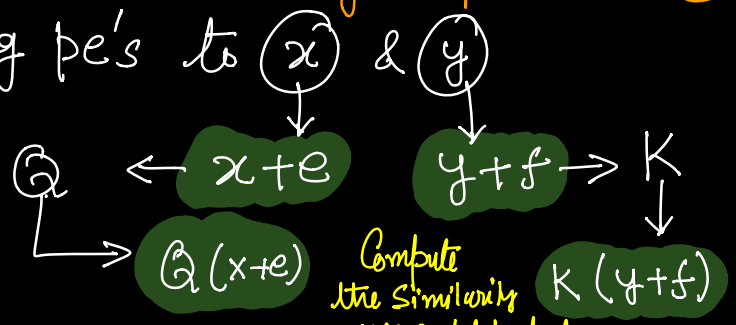
$$(*)\ (Qx)'(Ky) = x'(Q'K)y$$

How much attention should we pay to word (x) given word (y).

Learning a single tranformation matrix ($Q'K$)

It transforms secondary i/p (y) into a space where it can be directly compared with (x).

$\Rightarrow$ Now adding pe's to x & y



$$(Q(x+e))'(K(y+f)) = (Qx+Qe)'(Ky+Kf)$$

$$= (Qx)'Ky + (Qx)'Kf + (Qe)'Ky + (Qe)'Kf$$

$$\Rightarrow x'(Q'K)y + x'(Q'K)f + e'(Q'K)y + e'(Q'K)f$$

Assuming $(Q'K)$ as a single transform $\Rightarrow$ (T)

$$(Q(x+e))^{T} \cdot (K(y+f)) = x'Ty + x'Tf + e'Ty + e'Tf$$

$$\Downarrow \qquad \Downarrow \qquad \Downarrow \qquad \Downarrow$$

$$AT(x\|y) \quad AT(x\|f) \quad AT(y\|e) \quad AT(e\|f)$$

Attention to word (x) given word (y)

Attention to word (x) given the position (f) of word (y)

Attention to word (y) given the position (e) of word (x)

Attention to the position (e) of word (x) given the position (f) of word (y)

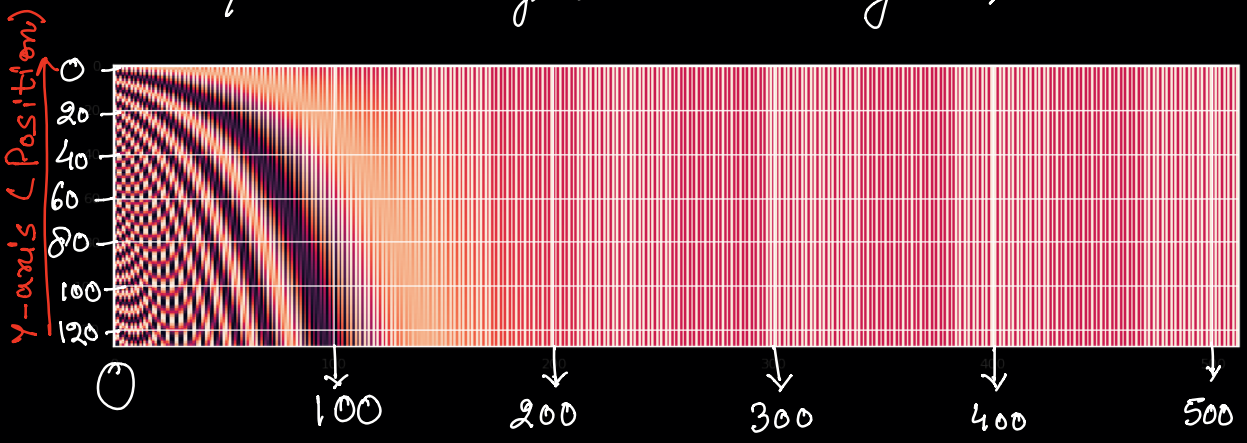
Hence learning (T) with (WE + PE) need to look for these 4 tasks to accomplish Simultaneously.

(*) Concatenations may give orthogonality / seperations between (WE) & (PE) but as both are high dimensional approximate orthogonality is naturally being achieved. (w/o Concatenation Cost of more Parameters)

Subspaces in that high dim. space $\Rightarrow$ Can be approximately orthogonal

Q2 (Answer): Since we are going to add both WE & PE we need to make their dimensions "Same".

Assuming any seq of length (20 which is common) with word embedding of 512 dim, most of the positions embedding dimensions will be used/dominated by world embedding (WE).



Q3: How does it address relative positions.

(i.e) PE for t & $t+\Delta t$ need to be consistent.

∴ for some fixed (K) (PE_{pos}) & (PE_{pos+K}) need to be linearly related.

As we all know that for each frequency (ω_k) we have $\sin(\omega_k t)$ and $\cos(\omega_k t)$ components.

Let us assume a linear transformation $(M) \in \mathbb{R}^{2\times 2}$ independent of (t) as follows:

$M = f(\Delta t)$

$$M * \begin{bmatrix} \sin(\omega_k t) \\ \cos(\omega_k t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k (t+\Delta t)) \\ \cos(\omega_k (t+\Delta t)) \end{bmatrix}$$

This ensures that the PE at (t) for (ω_k) frequency is linearly related to that of $(t+\Delta t)$ and is independent of (t) (only depends on (Δt))

Let us compute such an (M).

$$\begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix} * \begin{bmatrix} \sin(\omega_k t) \\ \cos(\omega_k t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k (t+\phi)) \\ \cos(\omega_k (t+\phi)) \end{bmatrix}$$

$$\begin{bmatrix} u_1 \sin(\omega_k t) + v_1 \cos(\omega_k t) \\ u_2 \sin(\omega_k t) + v_2 \cos(\omega_k t) \end{bmatrix} = \begin{bmatrix} \cos(\omega_k \Delta t)\sin(\omega_k t) + \sin(\omega_k \Delta t)\cos(\omega_k t) \\ -\sin(\omega_k \Delta t)\sin(\omega_k t) + \cos(\omega_k \Delta t)\cos(\omega_k t) \end{bmatrix}$$

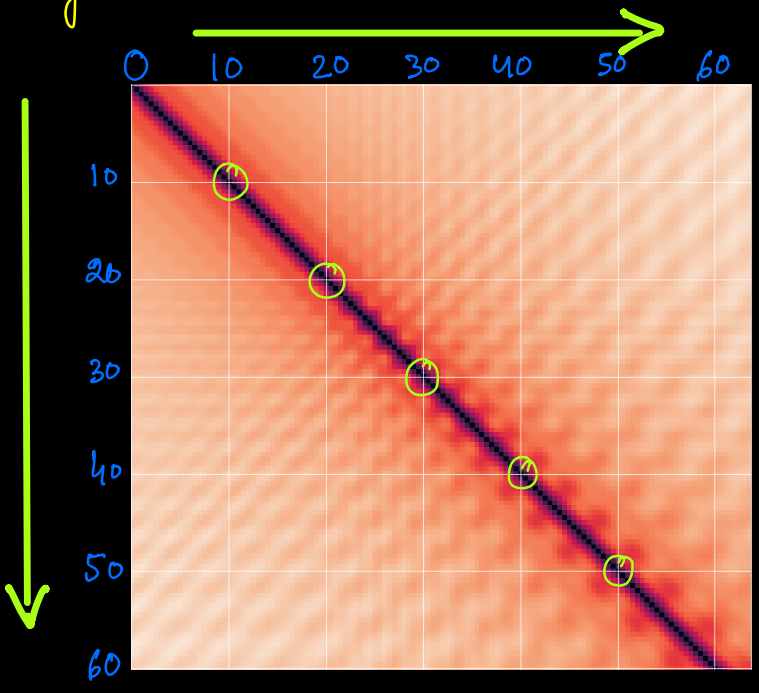
- $u_1 = \cos(\omega_k \Delta t) \quad v_1 = \sin(\omega_k \Delta t)$
- $u_2 = -\sin(\omega_k \Delta t) \quad v_2 = \cos(\omega_k \Delta t)$

$$\therefore M_{\Delta t,k} = \begin{bmatrix} \cos(\omega_k \Delta t) & \sin(\omega_k \Delta t) \\ -\sin(\omega_k \Delta t) & \cos(\omega_k \Delta t) \end{bmatrix}$$

Same as Rotation matrix

$\Rightarrow$ Similarly we can have $M_{\Delta t,k}$ (for all k's)

Hence such a transformation can eventually relate $(\vec{P}_t)$ & $(\vec{P}_{t+\phi})$ linearly enabling model learns to attend relative positions.



Showing the dot product of PE's for all time steps.

- Distance b/w neighbouring time steps are symmetric
- Decay linearly with time.

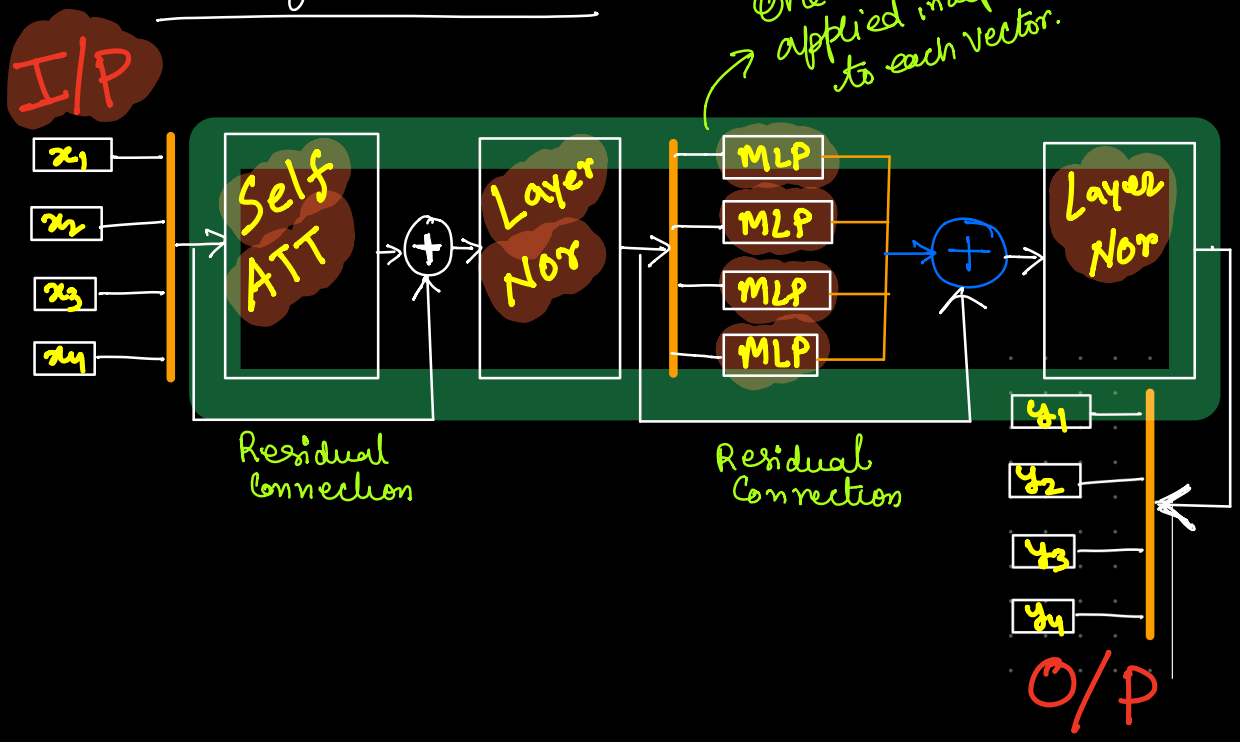
Diagonals are zero and off diagonals are Symmetric.

Transformer Architecture uses Self-Attention.

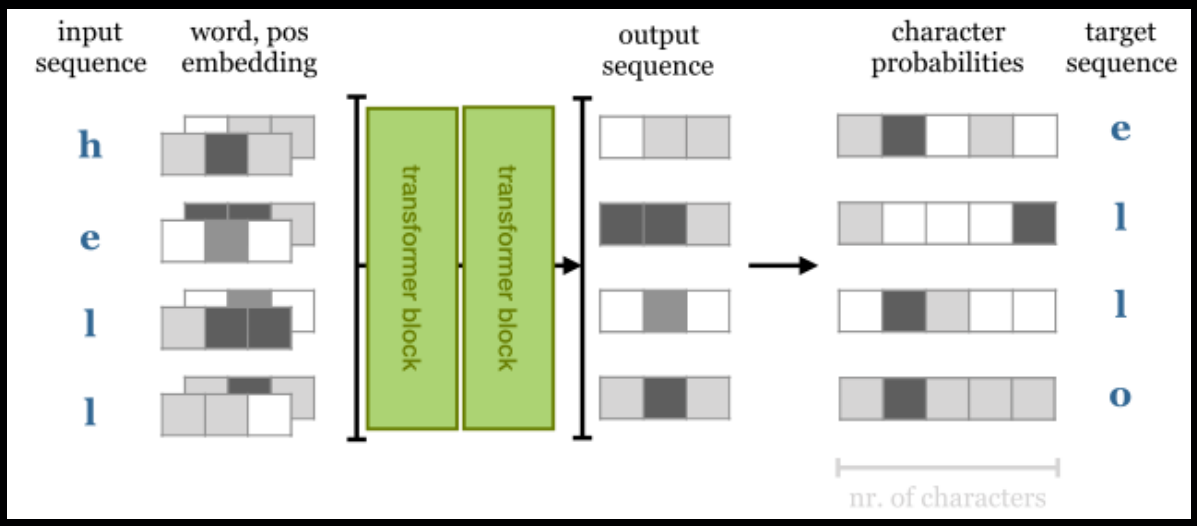
⇓

It has **units** that can process tokens, sequences, set of pixels but units can **interacts** via **Self attention.**

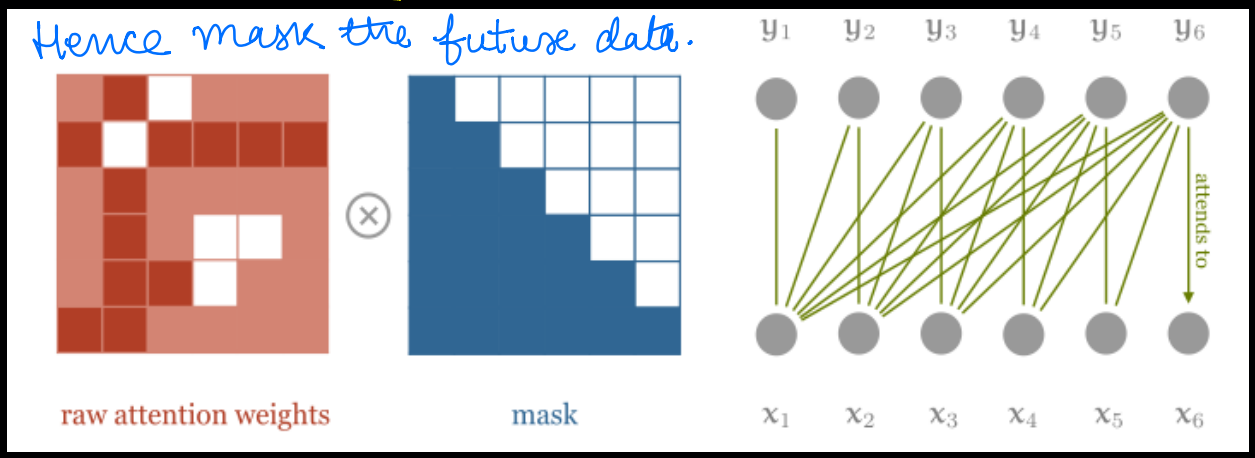
⇒ Transformer block :



How to generate Text using Transformers



O/P depends upon entire input so next character prediction is just choosing. (Trivial Step)



Only allow to attend the previous data.