Lec - 10 : Transformers

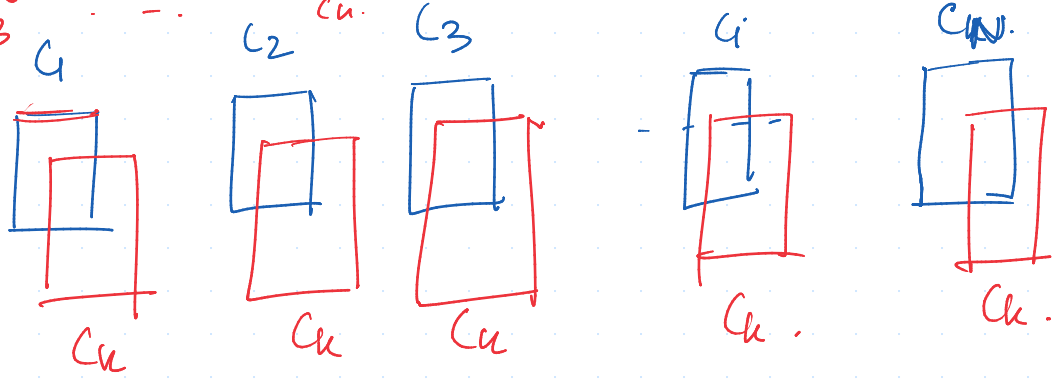
21/3/26

Word → tokenization → embedding → C_k (code)

k^{th} embedding.

the cat sat on the mat.

C_1 C_2 C_3 C_4 C_5 C_6



with the k-th word, perform the inner product to each of the n^{th} word.

+ve -ve

embedding:

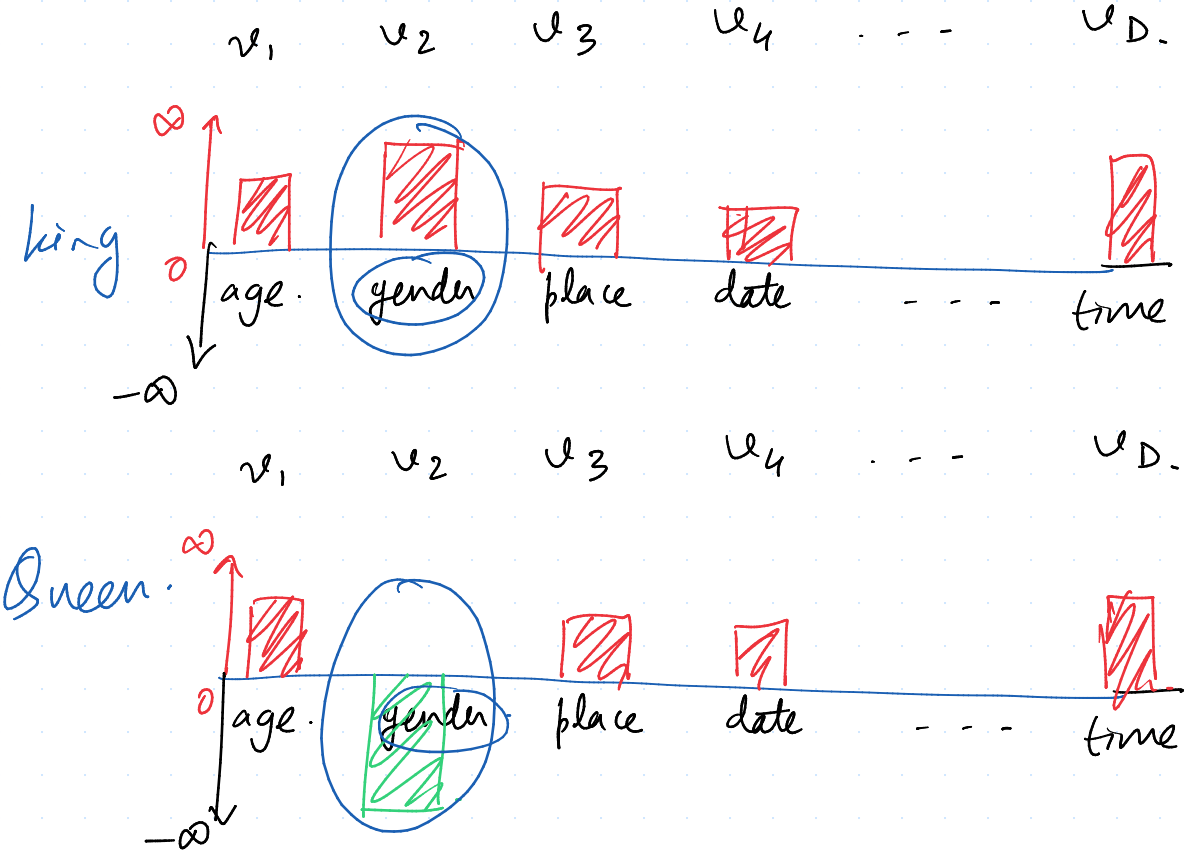
$C_k = \langle v_1^{(k)}, v_2^{(k)}, v_3^{(k)}, \ldots, v_D^{(k)} \rangle$ → D-dim vector.

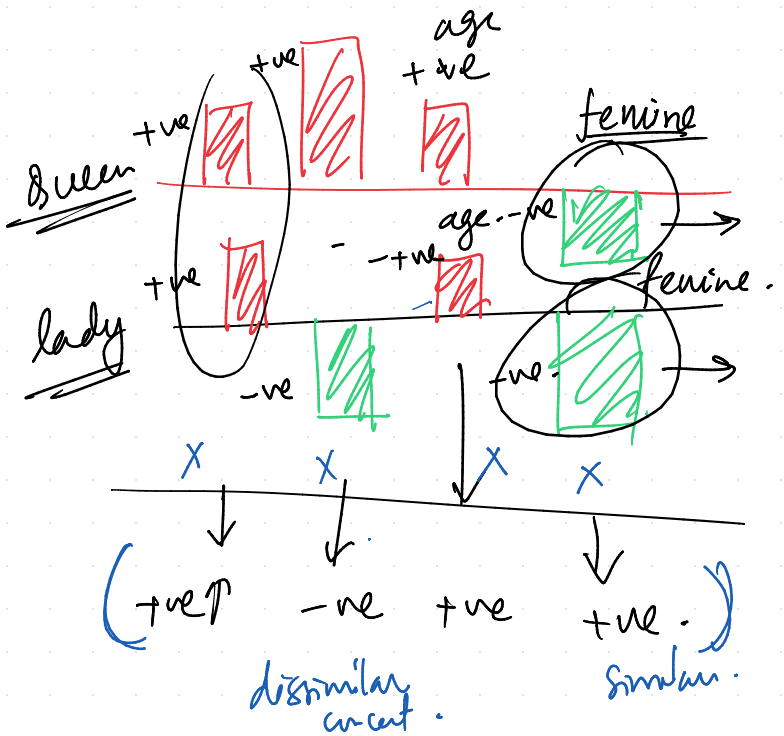
$C_i = \langle v_1^{(i)}, v_2^{(i)}, v_3^{(i)}, \ldots, v_D^{(i)} \rangle$

Dot product / inner product $= \sum_{j=1}^{D} v_j^{(k)} v_j^{(i)}$ = scalar.

C_i's → are complex combination of different concepts of words learnt based on the weights. ← training data.

Let us assume that we have trained a model (word-2-vec) on a corpus, we have learnt the word embeddings for the corpus.





→ +ve ?

When concepts are similar, they have a similar interpretation in the feature space.

Similar concepts when multiplied gets a large +ve value & dissimilar concepts when multiplied gets a large -ve value.

$(-\infty, +\infty)$

→ Probability dist
- +ve values.
- $\Sigma \Rightarrow 1$.

-10 -2 10 0 -5

-5 -7 -3 2 1

(50 14 -30 0 -5) -

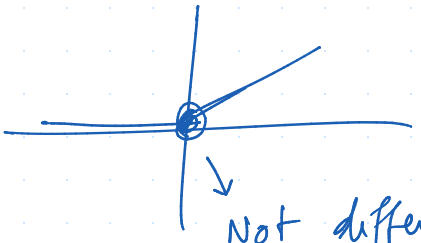
30 30 30 30 30 -

(80 44 0 30 25) - Adding the least value.

Dividing by the max value.

Normalize -

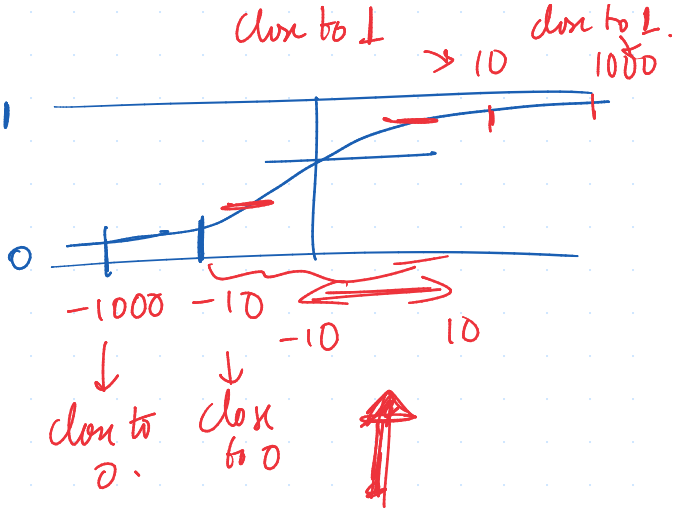
Not differentiable. Finding the least value is like sorting — .. (min, max) → differentiable (X) finding min-max is not differentiable.



↓ Not differentiable

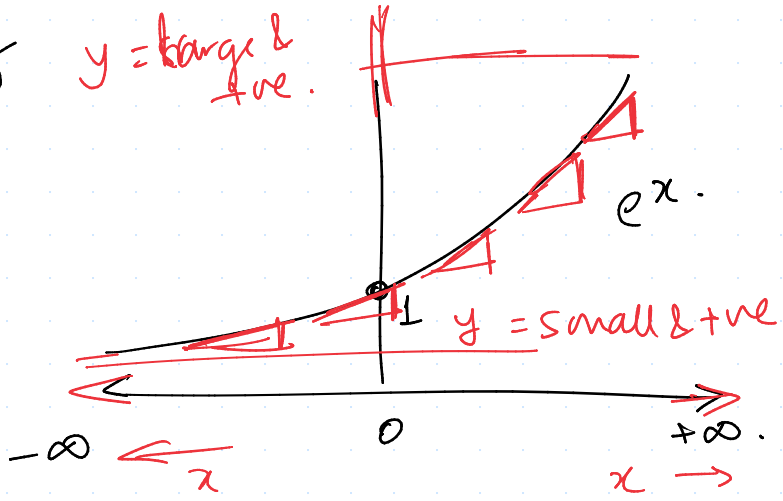
ReLU.

$\max(0, x)$.



monotonically increasing function.

$$\frac{\partial}{\partial x} e^x = e^x$$



$e^{\{\text{resultant vector}\}}$ ⇒ +ve values. ✓

* Since e^x is a monotonically increasing function the small values scales up accordingly

$\exp(c_i \cdot c_k)$ → +ve output.

preserve the meaning of the product.

$\gamma_{k\to 1}, \gamma_{k\to 2}, \ldots, \gamma_{k\to i}, \ldots \gamma_{k\to N}$.

$c_k \cdot c_1$ $c_k \cdot c_2$

relative sensitivity of the kth word to the ith word

Normalization

$$\gamma_{k\to i} = \frac{\exp(c_k \cdot c_i)}{\exp(c_k \cdot c_1) + \exp(c_k \cdot c_2) + \cdots + \exp(c_k \cdot c_N)}$$

$$= \frac{\exp(c_k \cdot c_i)}{\sum_{n=1}^{N} \exp(c_k \cdot c_n)} \to 0.$$

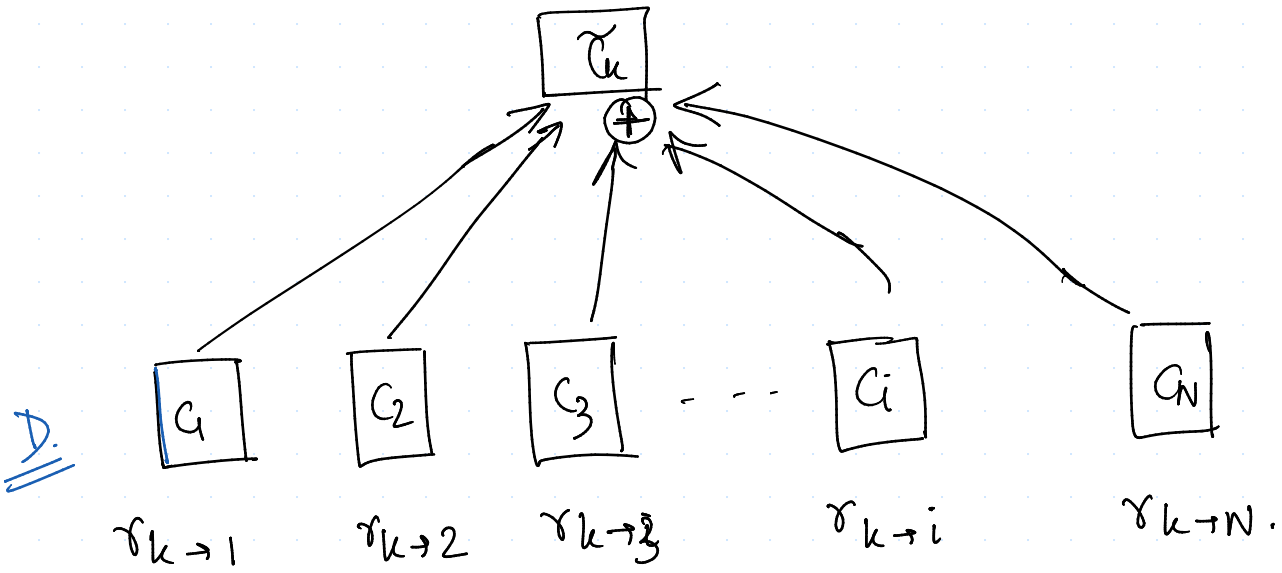
between (0 - 1)

probability distribution.

relative relationship b/w 2 word vectors.

-ve values → small & +ve.

+ve values → large & +ve.



D.

$C_k \rightarrow$ not informed of the context / independent of the context.

$[\vdots]$ $[\vdots]$ $[\vdots]$ $[\vdots]$

$C_1 \cdot C_k$, $C_2 \cdot C_k$, ---- , $C_i \cdot C_k$, ... $C_N \cdot C_k$.

$\gamma_{k\to 1}$ $\gamma_{k\to 2}$ $\gamma_{k\to i}$ $\gamma_{k\to N}$

(relative relationship)

Multiply the relative degrees by corresponding codes & add them up $\tilde{C}_k$.

If C_k is highly related to a particular word, then the corresponding relative relationship quantified through γ will be larger

$r_{k \to 1}$; how much attention to pay to the corresponding word/code.

$$\tilde{C}_k = (r_{k\to 1} \times C_1) + (r_{k\to 2} \times C_2) + \cdots + (r_{k\to i} \times C_i) + \cdots + (r_{k\to N} \times C_N)$$

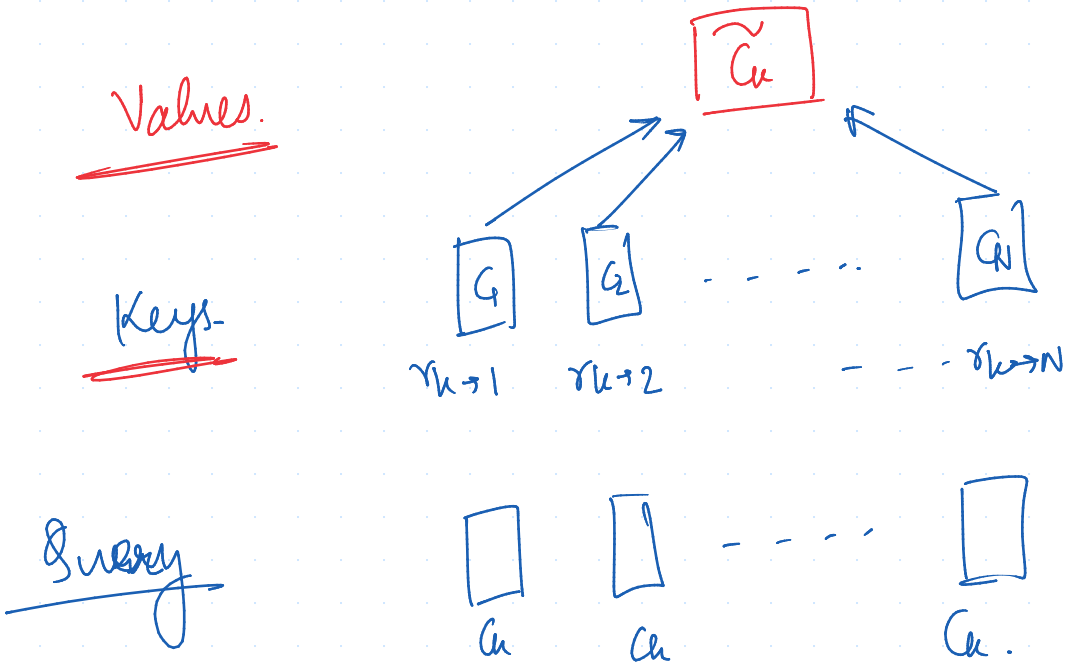
$$= \sum_{n=1}^{N} r_{k\to n} \times C_n$$

$\tilde{C}_k \uparrow$ → pay high attention to C_k

$\tilde{C}_k \uparrow$ → pay low attention to C_k.

C_k to $\tilde{C}_k$ → mapping / vector transformed with context.

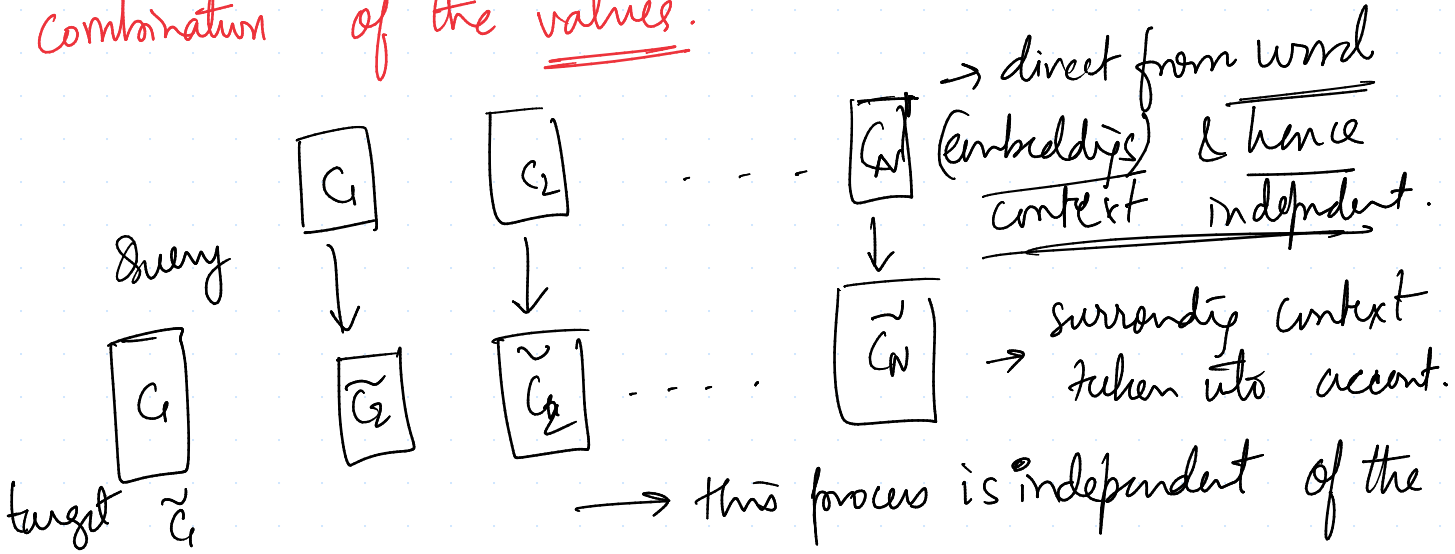
takes into account the surrounding context.



Code C_k transformed to $\tilde{C}_k$,

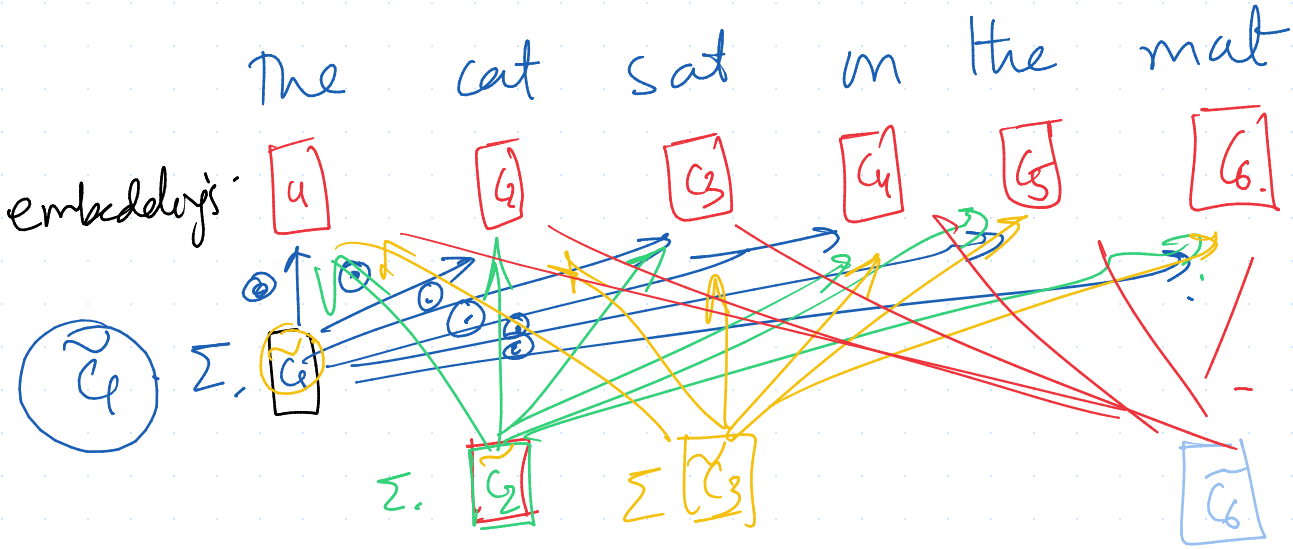
Relative weights $\{r_{k\to 1}, r_{k\to 2}, \ldots \to r_{k\to N}\}$ are always +ve & sum to 1.

Each query C_k mapped to $\tilde{C}_k$ is a convex combination of the values.



word order.

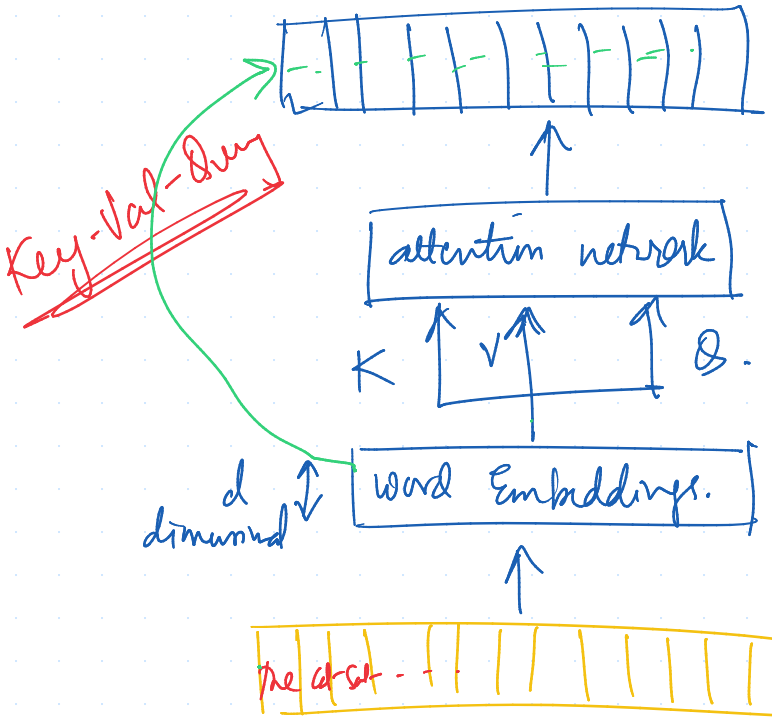
The word 'order' matters, if we want to capture the contextual meaning.



Semantic meaning is different.

→ The cat sat on the mat. ✓

→ The mat sat on the cat. ✗

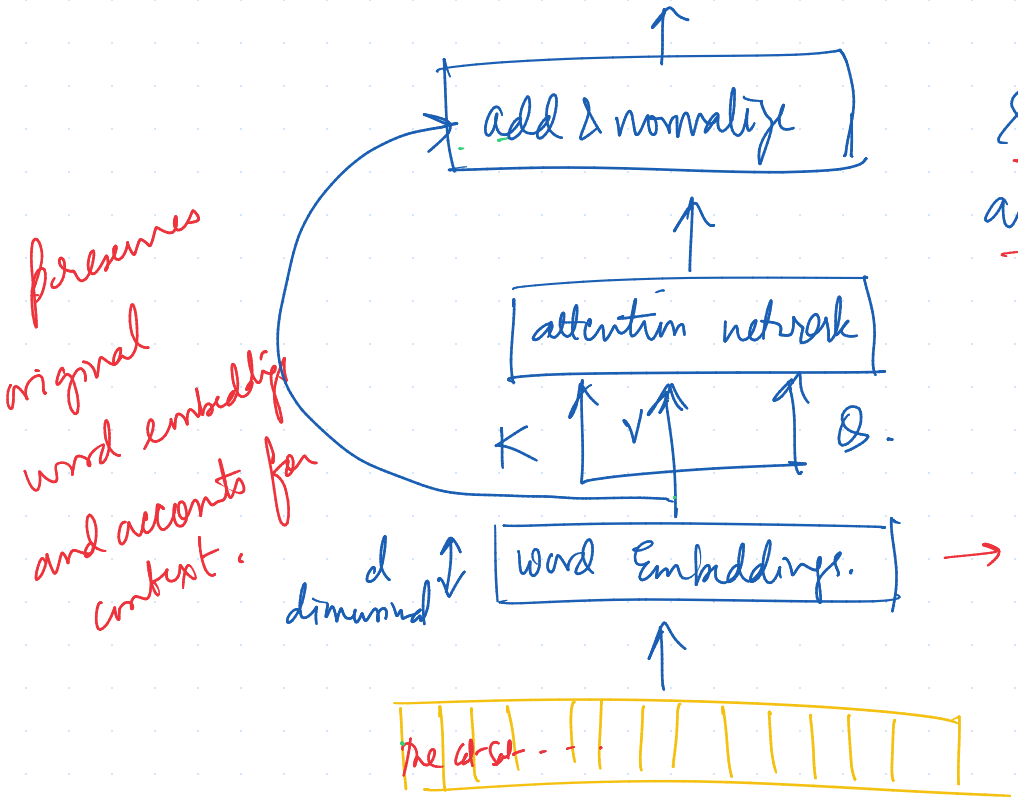


o/p sequence. Takes into account the context of the words.

→ doesn't take into account the context of the words.

input sequence of words.

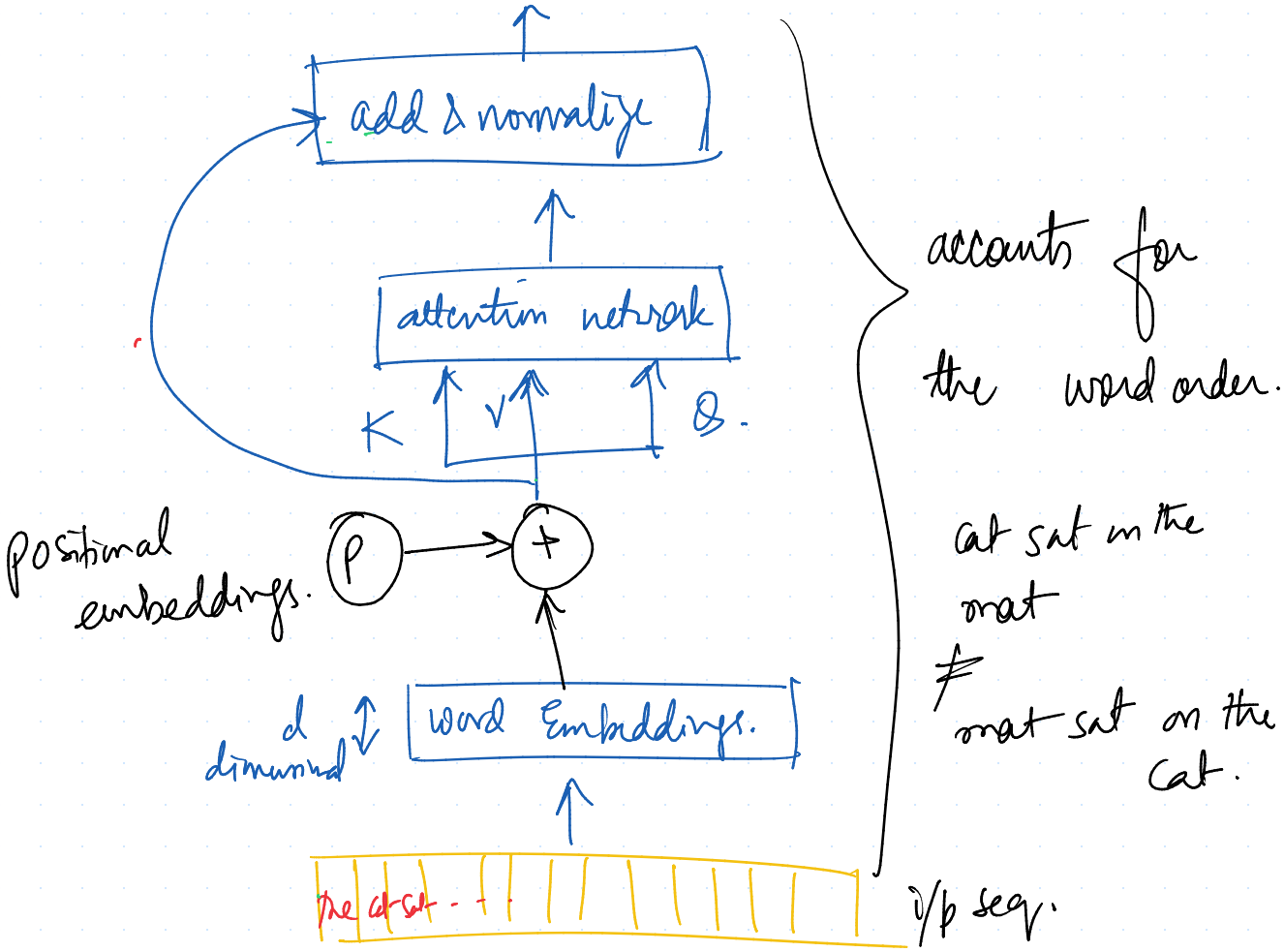
We learn the word embeddings at the bottom, when we go from the bottom to the top.



Skip connections are introduced

Word order are not accounted for yet.

Positional Embeddings



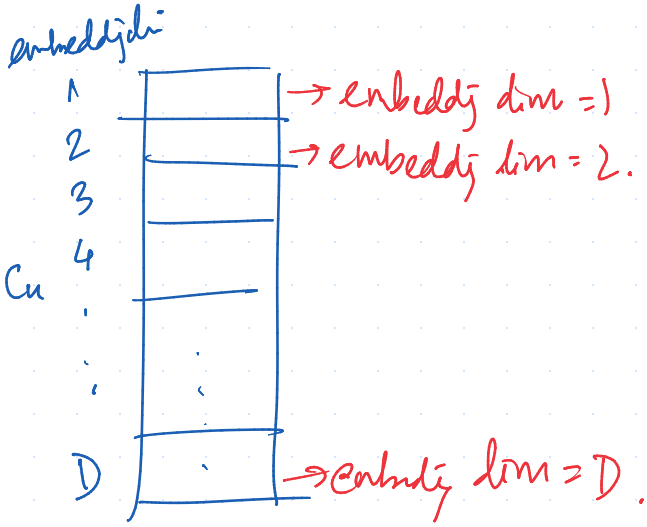
Positional embeddings.

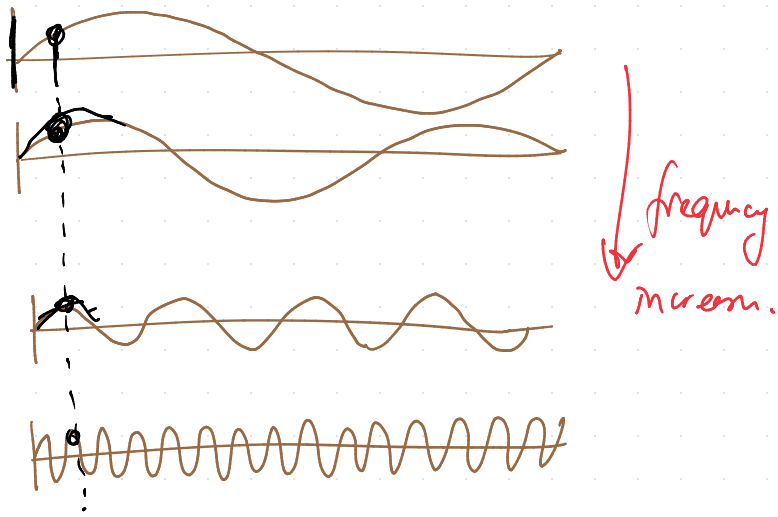
$$PE_{(pos,2i)} = \sin\left(pos / 10000^{2i/d}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(pos / 10000^{2i/d}\right)$$

} d-dimensional positional embeddings.

→ represented by sine & cosine waves.





Sine-wave

closer word will get more embedding context than the later words.

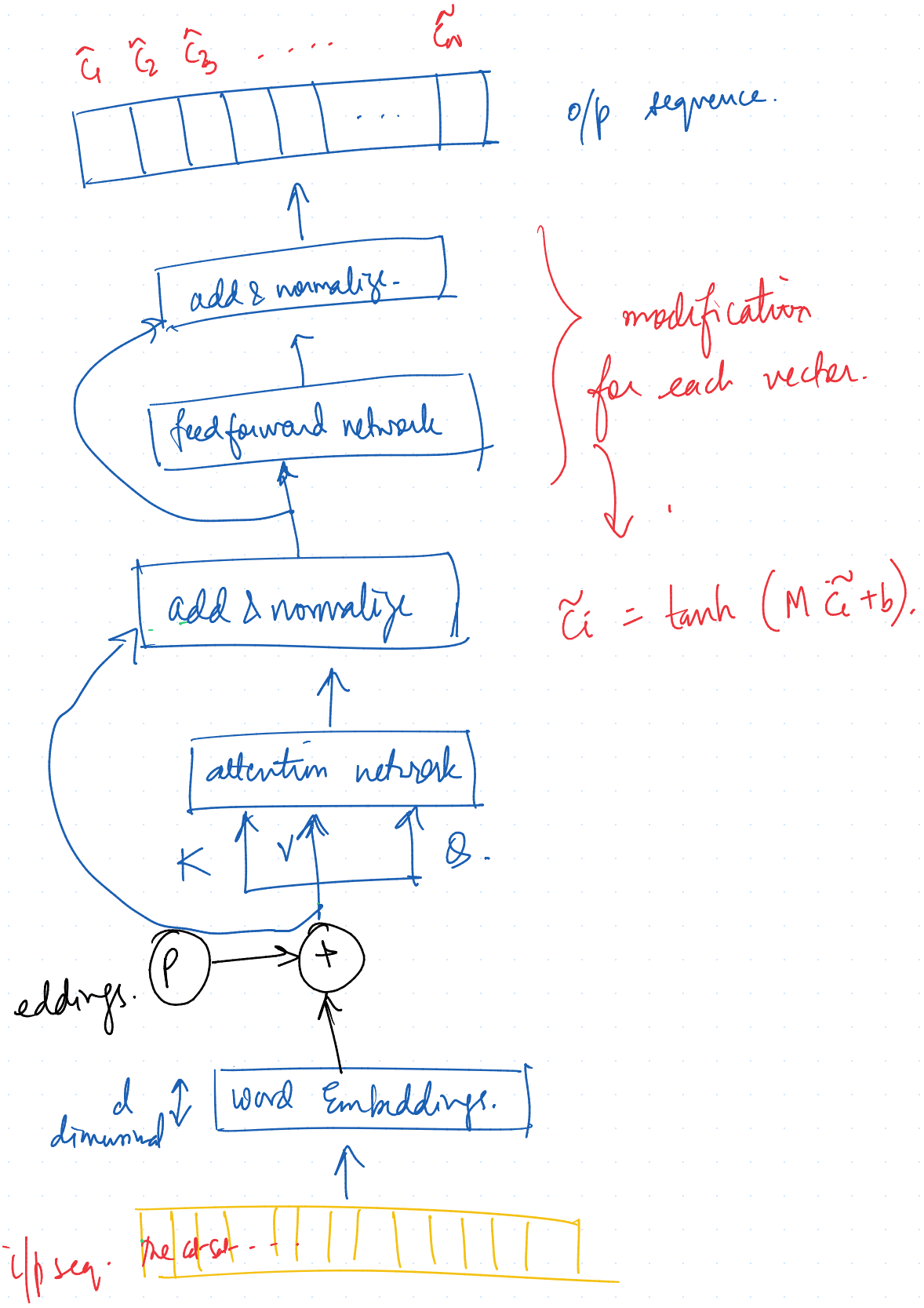
$\sin\left(pos / 10000^{2i/d}\right)$

$\downarrow \frac{pos}{\uparrow (10000)^{\frac{2i}{10}}}$

$\uparrow \frac{pos}{\downarrow (10000)^{\frac{2i}{10000}}}$.

☆☆

check positional embeddings from blogs etc in the internet.


$\tilde{c}_1$ $\tilde{c}_2$ $\tilde{c}_3$ . . . . $\tilde{c}_n$
o/p sequence.
add & normalize.
feedforward network
modification for each vector.
add & normalize
$\tilde{c}_i = \tanh(M\tilde{c}_i + b)$.
attention network
K
V
Q.
P
+
eddings.
d dimensional
word Embeddings.
i/p seq.

Words that are nearby positionally will have similar positional embeddings, therefore their inner products will be high.

Words that are far apart → word positional embeddings will be different → hence → -ve inner products hence they will not be related.