09 Mar 2025

SELF ATTENTION

- The self attention mechanism enables each token in a sequence to attend to every other token, dynamically adjusting its representation based on its relationship to other tokens.
- This flexibility allows the transformers model to handle long-range dependencies and parallelize computations efficiently.

lets take a two sentence
(i) money bank grows
(ii) river bank flows.

for 1st sentence

money = 0.7 money + 0.2 bank + 0.1 grows
bank = 0.25 money + 0.7 bank + 0.05 grows
grows = 0.1 money + 0.2 bank + 0.7 grows

or in the term of embedding

for 2nd sentence

river = 0.8 river + 0.15 bank + 0.05 flows
bank = 0.2 river + 0.78 bank + 0.02 flows
flows = 0.4 river + 0.01 bank + 0.59 flows

or in the term of embedding

$e_m(new) = 0.7 e_m + 0.2 e_b + 0.1 e_g$ —— (a)
$e_b(new) = 0.25 e_m + 0.7 e_b + 0.05 e_g$ —— (b)
$e_g(new) = 0.1 e_m + 0.2 e_b + 0.7 e_g$ —— (c)

↳ it means.
$e_m(new)$ is makes with 0.7 time e_m, 0.2 time e_b and 0.1 time e_g

or we can say that coefficients (e.g. 0.7) is similarity between the $e_m(new)$ and e_m

Here,

e_m – embedding of money

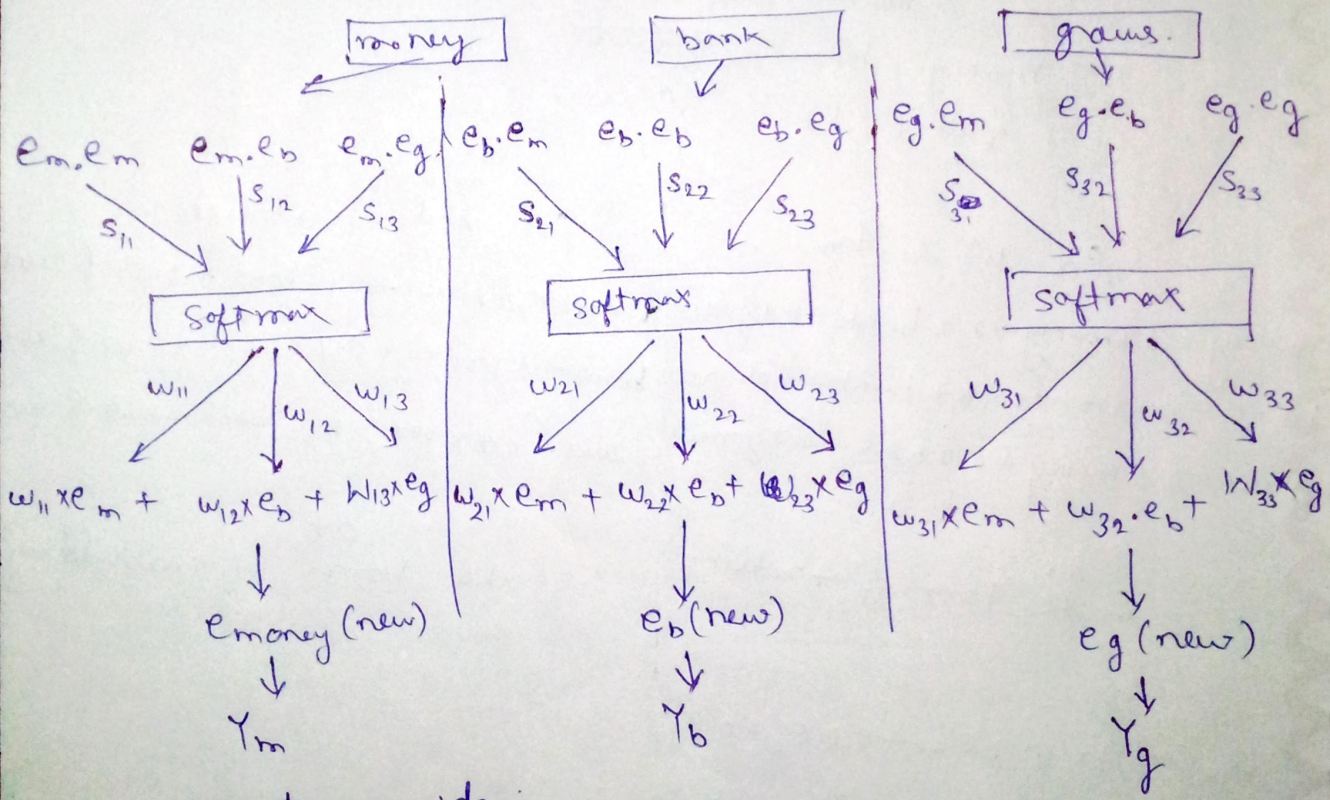
e_b – embedding of bank

e_g – embedding of grams

Here these embedding are vectors in high dimension there find the similarity between two vector we uses the dot product

$$e_b(new) = [e_b \cdot e_m^T]\, e_m + [e_b \cdot e_b^T]\, e_b + [e_b \cdot e_g^T]\, e_g$$

from equation (b)

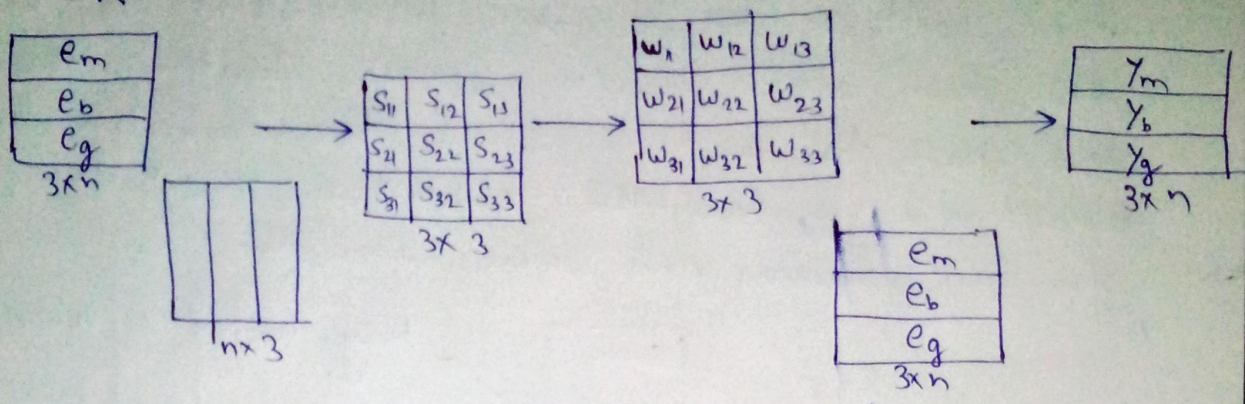


point to consider:

- This operation is a parallel operation
- There are no parameters involved
- This is generate the general contextual embedding not a task specific contextual embedding

(i) architecture of general contextual embedding of each word

OR

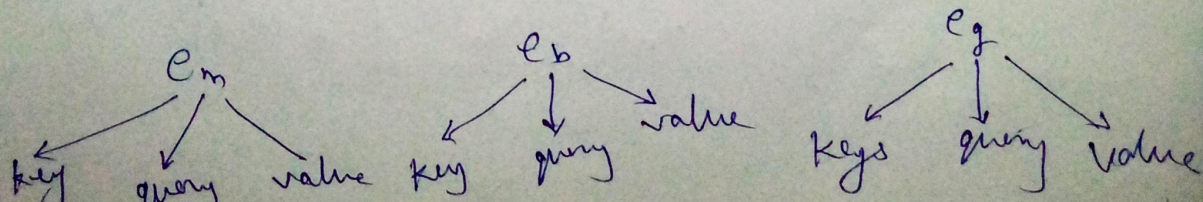


Here 3 number of word in sentence
n is size of embedding vector

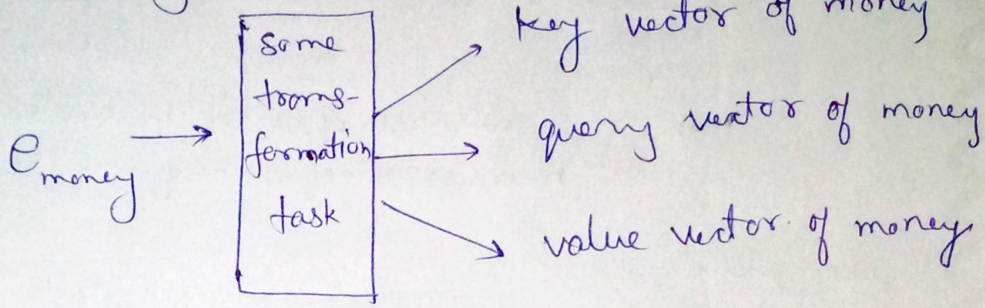
example:

in general contextual embedding	in task specific contextual embedding
piece of cake – केक का टुकड़ा	piece of cake – बहुत आसान काम
break a leg – टांग तोड़ दो	break a leg – शुभकामनाएँ

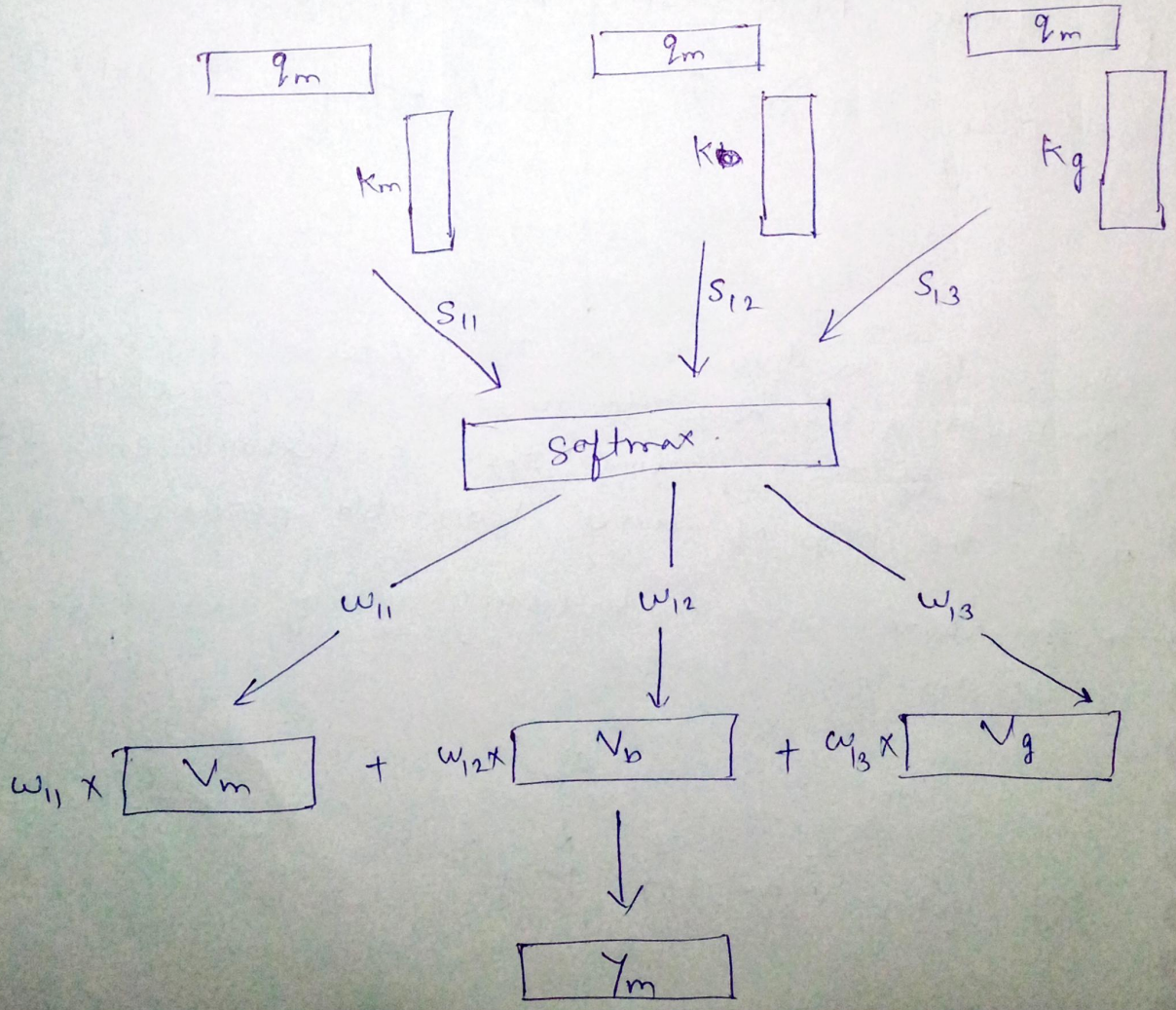
- Now, due to drawback of general contextual embedding we have to improve the architecture of the finding the contextual embedding with the help of some learnable parameters.
- in the above word embedding of any word is use a three ways.



- in given sentence the embedding of given word have contain lot of given information to use the given word for use a specic function then we have to perform some transformation on word and produce the subvector. (three) example for key, query and value. example



- now find the contextual embedding of "money"



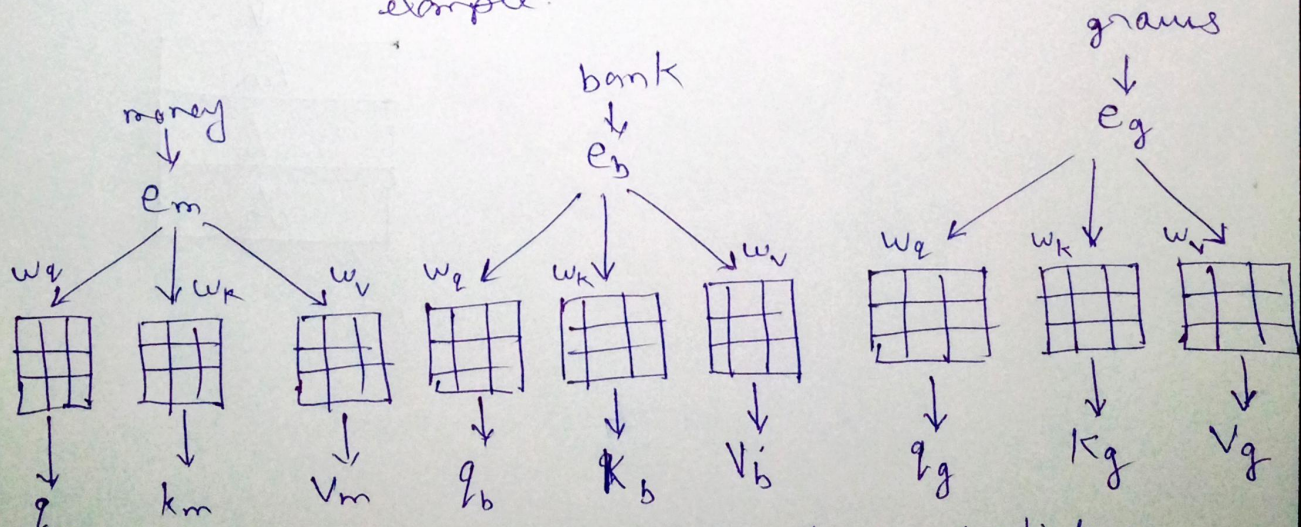
- similare way of structure of other other words also ("bank", "graw").

Here,

q_m – query's vector of money word
k_m – key's vector of money word.
k_b – key's vector of bank word.
k_g – key's vector of graw word.
v_m – value's vector of money word.
v_b – value's vector of bank word
v_g – value's vector of graw word.

- Nau let's find what type of transformation/operation is perform to find the these vectors of each word (key, query, value).

- magnitude (scaling)
- linear transformation ✓
(multiply vector (e_m, e_b, e_g) with matrix)
example.



value inside the matrix is randomly initialize.
they will update the value for further iterations

(ii) Architecture of task specific contextual embedding of each word