Transformer 1

Friday, March 14, 2025 6:32 PM

▶ Why transformer was needed?

- Can't use large dataset to train, cause the data is sequential and It will take too much time to train the model.
- As we can't train use large dataset, we cannot use a model for transfer learning

▶ What is word embeddings?

- Convert a word into a "n" dimensional vector(Fixed length)
- Word embeddings can capture the semantic meaning of a word
- Here semantic meaning can be visualize by draw the words in a "n" dimensional vector

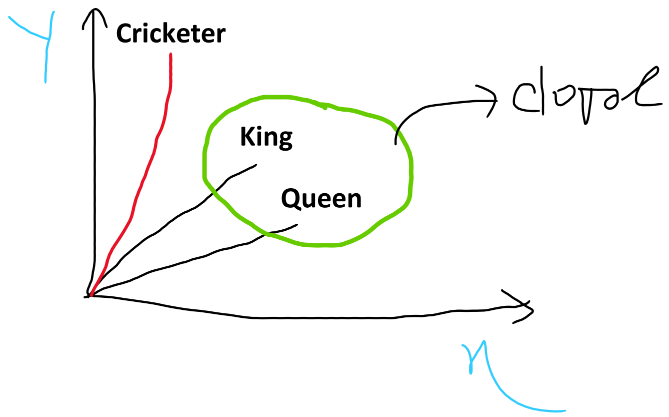
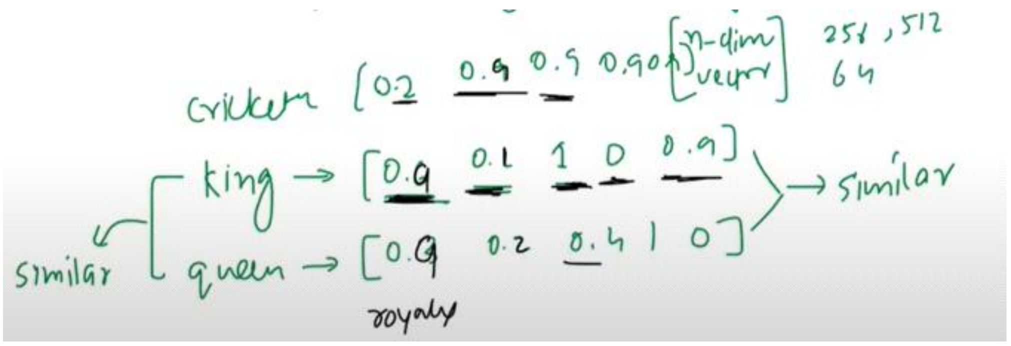


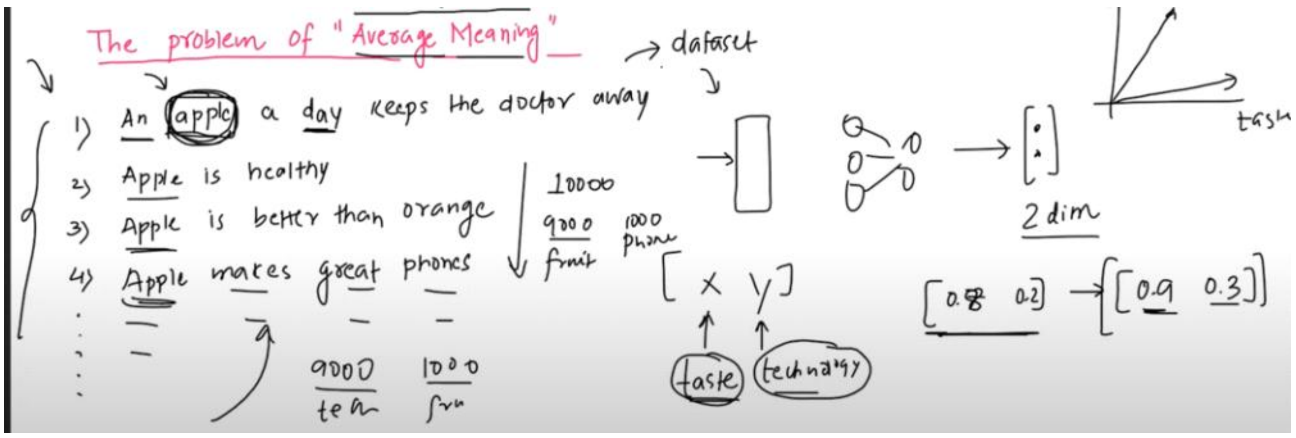
Fig: After placing the embedding vector of the words



Cricketer [0.2 0.9 0.5 0.90] n-dim vector
256, 512
64
king → [0.9 0.1 1 0 0.9]
queen → [0.9 0.2 0.4 1 0]
similar
similar
royalty

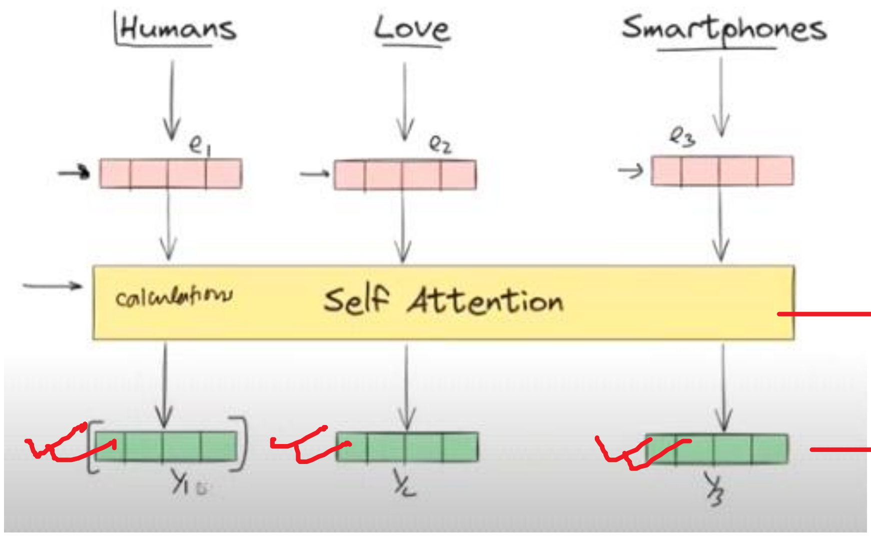
Here the value of each index of the vector represent a aspect, let just think the first index representing the Royality. Where the value of queen and king is high but for the cricketer it is low. Another index represents human........

▶ What was the problem with static embedding?


The problem of "Average Meaning"
→ dataset
1) An apple a day keeps the doctor away
2) Apple is healthy
3) Apple is better than orange
4) Apple makes great phones
10000
9000 fruit
1000 phone
[x y]
taste
technology
9000 tea
1000 fru
2 dim
[0.8 0.2] → [0.9 0.3]
task

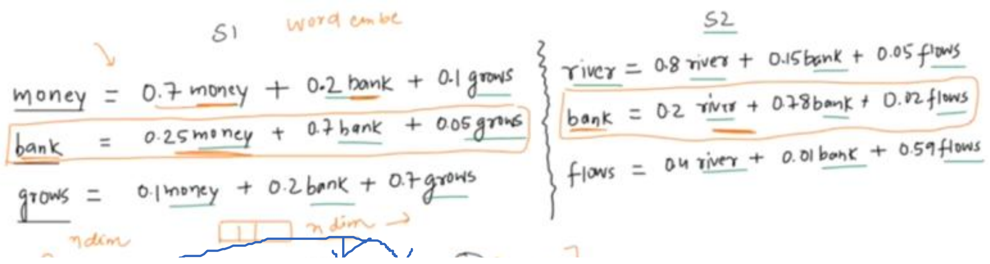
- When there is a dataset with a word "apple" and most of them are talking about the apple fruit, the embedding vector will be partially manipulated.
- That's why we need self-attention

► What is Self-Attention and how it works?



Embedding's with context

- After giving the input sentence, the input goes into the self-attention process where the embedding for the words changes with the meaning of the whole sentence



S1 word embe

money = 0.7 money + 0.2 bank + 0.1 grows

bank = 0.25 money + 0.7 bank + 0.05 grows

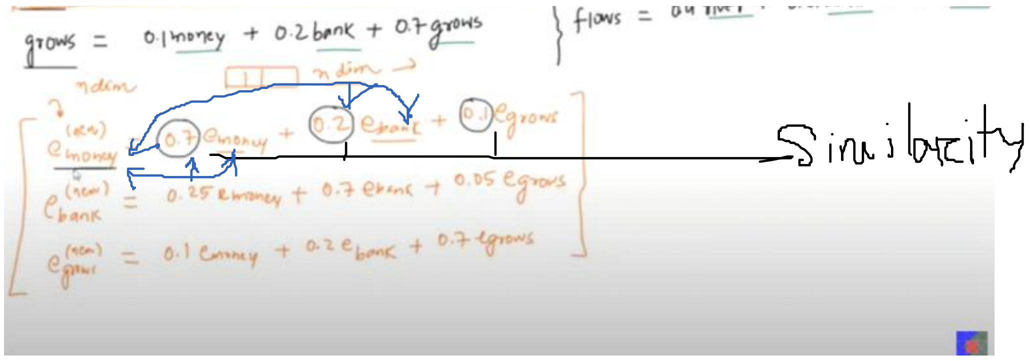
grows = 0.1 money + 0.2 bank + 0.7 grows

S2

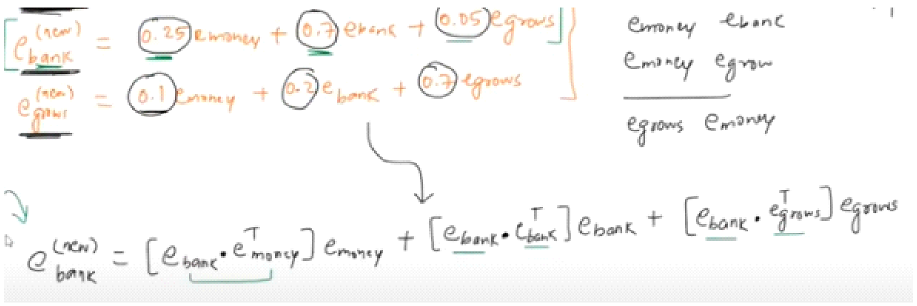
river = 0.8 river + 0.15 bank + 0.05 flows

bank = 0.2 river + 0.78 bank + 0.02 flows

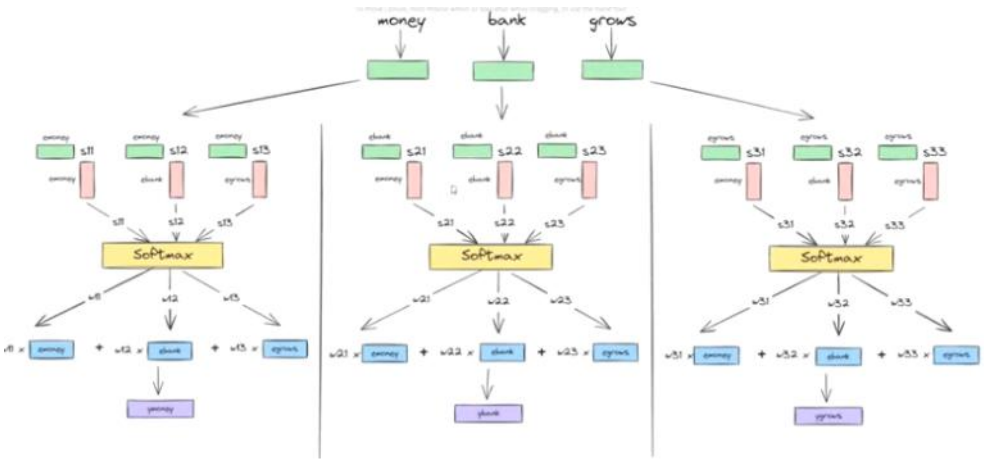
flows = 0.4 river + 0.01 bank + 0.59 flows


grows = 0.1 money + 0.2 bank + 0.7 grows
flows =
n dim
e(new) money = 0.7 e money + 0.2 e bank + 0.1 e grows
e(new) bank = 0.25 e money + 0.7 e bank + 0.05 e grows
e(new) grows = 0.1 e money + 0.2 e bank + 0.7 e grows
Similarity

- In the self-attention process the embedding of a word is the summation of the multiplication of the word embedding of that sentence and the similarity with the input word and the current word

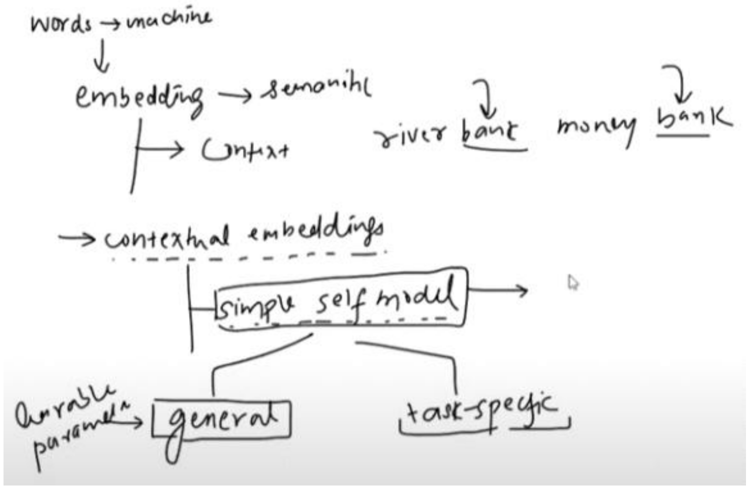

e(new) bank = 0.25 e money + 0.7 e bank + 0.05 e grows
e(new) grows = 0.1 e money + 0.2 e bank + 0.7 e grows
e money e bank
e money e grows
e grows e money
e(new) bank = [e bank · e money T] e money + [e bank · e bank T] e bank + [e bank · e grows T] e grows

- Now we can do this parallelly by vector multiplication cause there are no dependency between the vectors or any input

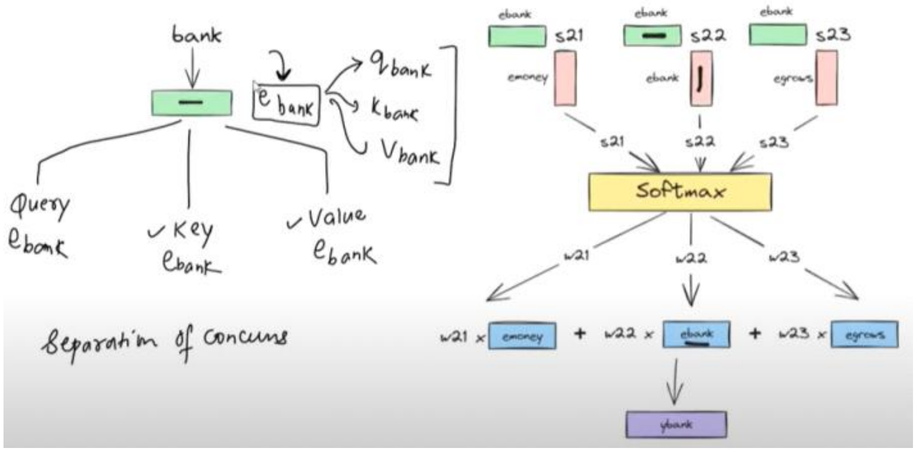


- But there are some issues with that structure, the embeddings we are getting are general embeddings. (cause there are no learnable parameters)
- If we try to translate the sentence "Break a leg" that does not mean want to break the leg.
- But our embedding result will be like that.

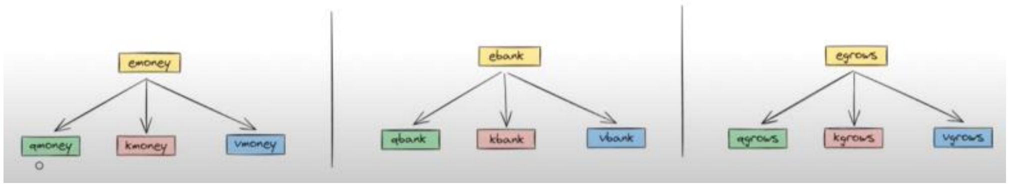
- So we need task specific embedding so that it can understand the true meaning on the sentence or specific task.



- But the question is how to use learnable parameters here?
- We are using the same vector for the vector dot product
- But we can convert the vector embedding to three embedding depend on their work

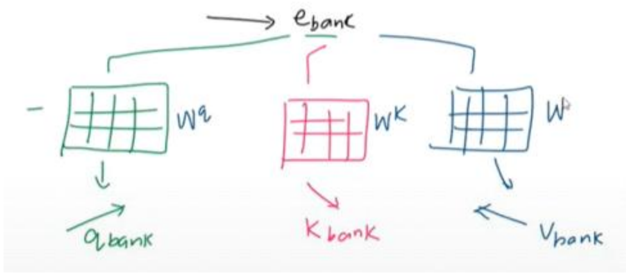


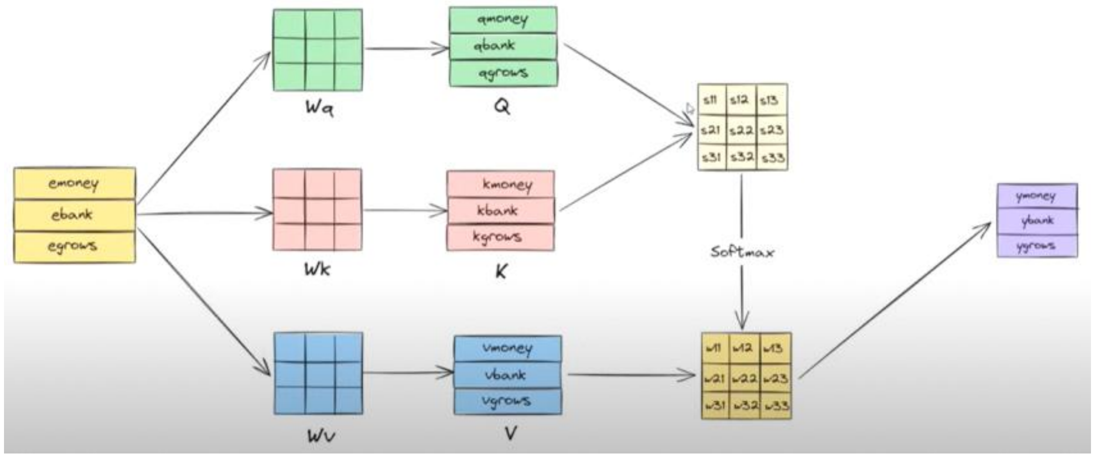
- From the figure above the green vector and the other vector getting the similarity where green vector is the query vector "G" and the other one is the key vector "X" eq: G(X) = Y
- And the last vector is the value vector "Y"



- But how to get these three vectors from one?
- If we have three "Special" vectors and we multiply with these vectors without embedding vector maybe we can get the three different vectors.

- But how can we get these vectors?
- At first these will be random vectors but throughout the training the value of these vectors will be change.





- On this end we will get the embedding vectors with self-attention and context