Transformer Architechture

Transformers

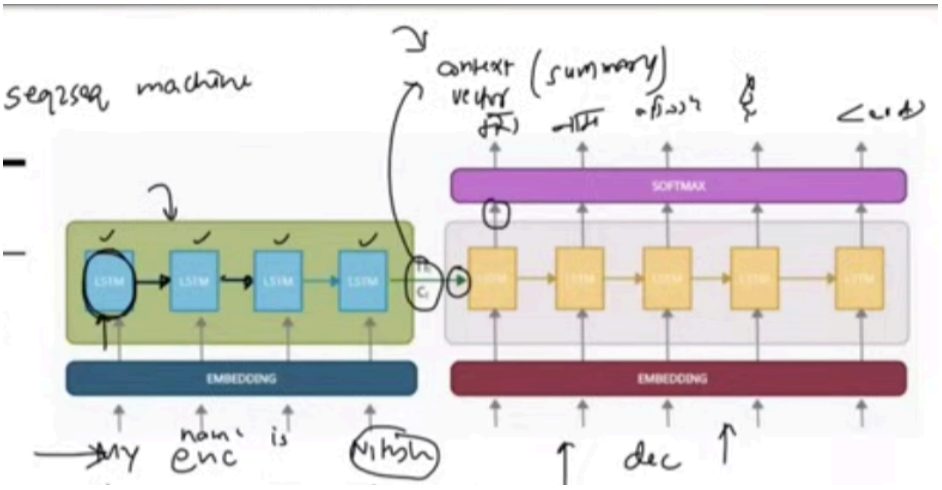
It is Neural Network Architecture that :-

1. Handles task sequence to sequence
2. Processes all the words in parallel which makes them model more scalable.

Origin Story

2014→*Sequence to Sequence learning with Neural Network.*

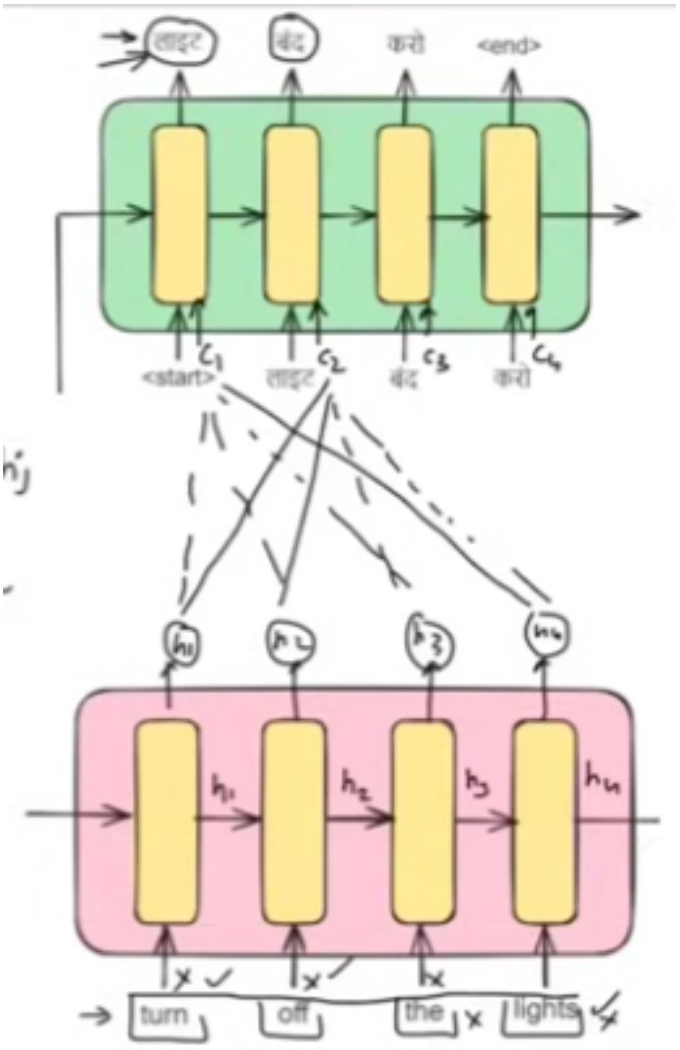
- Proposed a Architecture that worked on tasks SEQ 2 SEQ.
- Used the concept of encoder-decoder architecture.



- Works well with Small inputs but not with Long ones
- PROBLEM: The Input is stored in a Context Vectors, So The big Inputs will create a problem for the input vectors.

2015→*Neural Machine Translation.*

- Concept of Attention was Proposed.



- PROBLEM: Every other vector is connected to every another vector But They have LSTMs do they cannot process the words in Parallel.

2017→ *Attention is all you Need*

- Introduced the concept of Transformer Architecture and Self Attention.
- No use of LSTMs or RNNs and made parallel reading possible.
- Very Stable on training with huge datasets.

Advantages

- Scalability → can be trained in parallel and are quick.
- Transfer Learning→ train on large datasets and apply Fine tunning.
- Multi-model→Uses texts, images, audios to make different type of apps.
- Flexibility

    - Encoder only like BERT
    - Decoder only like GPT
- Ecosystem→Lots of libraries e.g. Hugging face
- Easily integrated with other AI techniques

Disadvantages

- They need High computational Cost (GPUs)
- require a lot of data (specially on specific domains)
- Overfitting
- Energy consumptions
- Interpretability
    - It is mostly a black box.
    - Very Difficult to tell what is going on inside the Transformers
    - So Banks and healthcare could be risky as it can give correct result but does not show how?

Future Developments

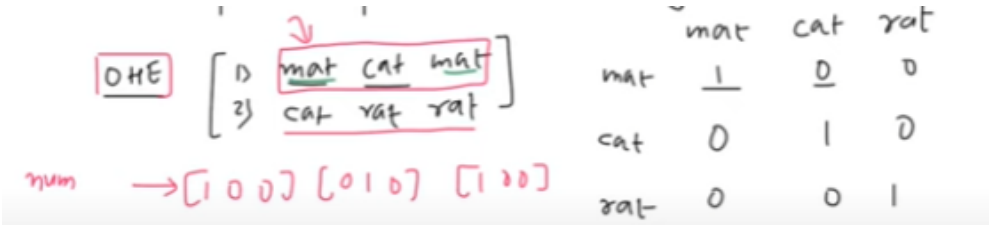
- Improved in efficiency
- Pruning
- Quantization
- Multi-model capabilities
    - Apart from text and images there can be biometric feedbacks, Time servers.
- They can go Multi-lingual.
- Solve the problem of interpretability.
- In future we can have domain specific GPTs like Dr. GPT, LegalGPT, etc.

Self Attention

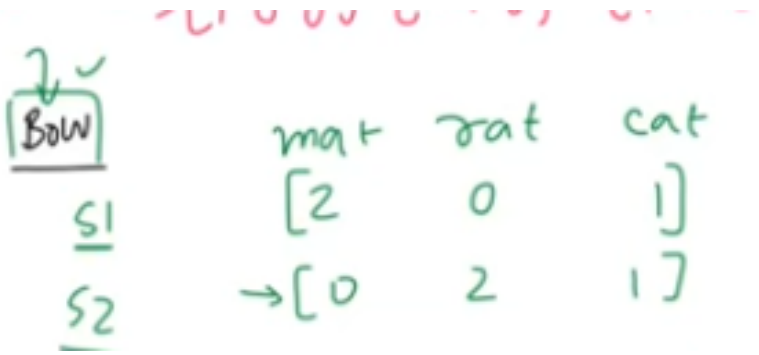
- Most important part about making an NLP

- Converting words⟶Numbers(vectors)

- Method-1



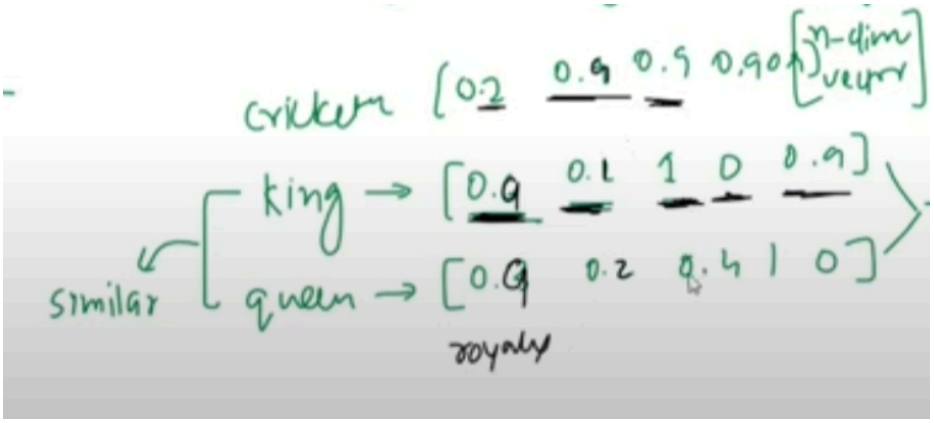
- Method-2
  - It also counts the Instances of each occurrence.



  - The problem with both the methods is that both of them are really inefficient.

Method of using Word Embedding

- This method captures the semantic meaning of the words which basically means that it captures the context of the word with respect to other words.
- STEPS
  1. Take the training data and send it to the NN.
  2. Understand the words and then convert them into N dimension vector.
  3. Every dimension has a meaning and 2 similar words have the value of the that particular dimension similar in which they are same.
  4. When plotted on a N-dimension plane the angle between same words is less as compared to the words that have distant characteristic in the dimension.



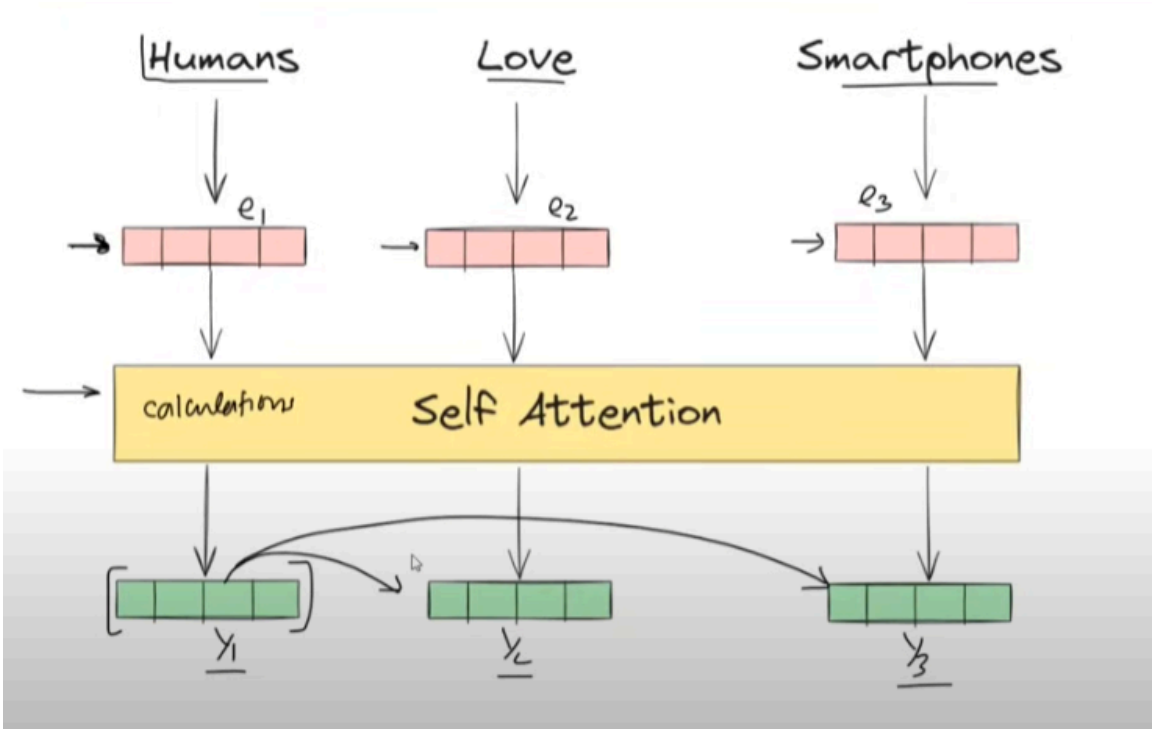
- Problems

  It captures the **Average Meaning** of the word.

  - Meaning while training Apple can be used as a fruit as well as a Company.
  - The vector has both the components.
  - [F,C] = [0.6,0.1] → Hence this represents the average meaning.

  - The main cause is because these embedding are Static Embedding.
    - We need Contextual Embeddings → These embedding analyze the entire sentence and adjust the meaning of words accordingly like in the case above increase the vector part of Fruit if the context is with regards to fruit.

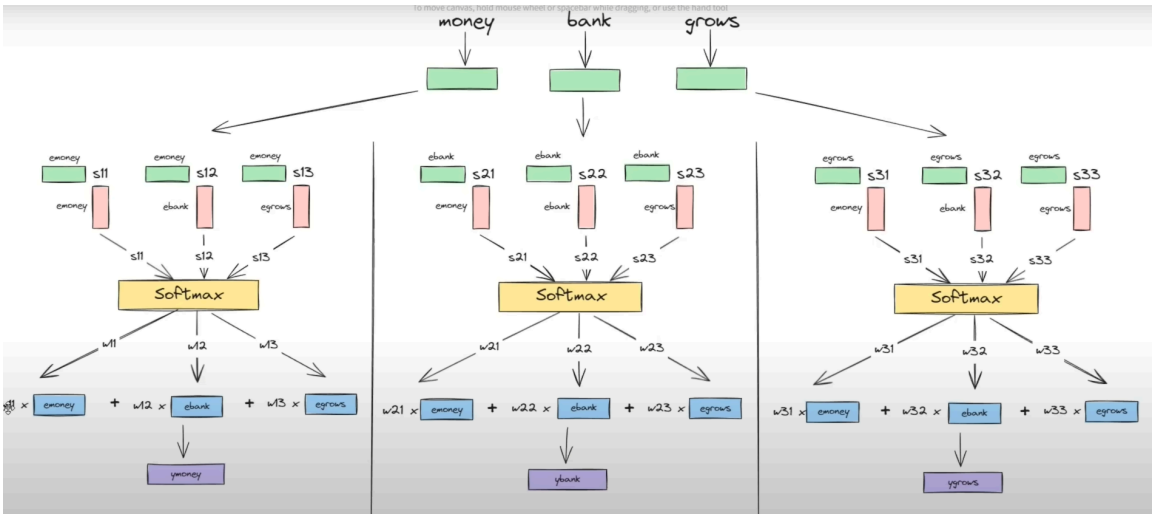
- SELF ATTENTION SOLVES THIS PROBLEM.
  - The Words (here Human,Love,Smartphones) are converted into static embeddings.
  - Then they are put inside the self Attention layers.
  - Calculation happen there and static embedding are converted into dynamic contextual embedding.



Frist Principle Approach for Self Attention

- S1→Money bank grows
  - 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 bank flows
  - 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)
- Conclusion→
  - Now in this way the word bank is same but it has a different meaning in both the sentences.
  - Money′ → new embedding which is the addition of 0.7 times the static embedding of Money, 0.2 times that of bank and 0.1 times of grows. (Instead of words computer will actually have numbers)
  - 0.7 and other numbers → They show similarity between the word embedding.

        - So, 0.7 is actually the dot products between the Money vector and the Money vector.
- Diagramic View

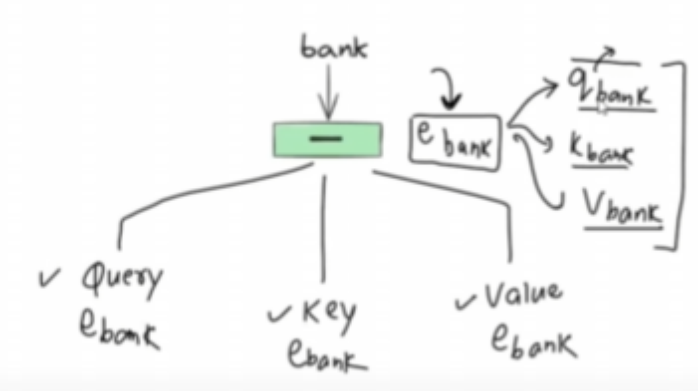


    - The above diagram shows how the equation comes to play in FPA.
        - S11,S12...and other represent the dot product of the embeddings.
        - These dot products go to the Softmax here the calculation happen and the numbers all the numbers are decreased keeping the equivalence in mind such that the the addition of all(S11,S12,S13) is 1.
- Points to consider
    - All the tasks can work in parallel.
    - There are no learning parameters involved.
        - Meaning the embedding that are formed are general contextual embedding(GCE) not task specific contextual embedding(TSCE)
        - e.g. S → Piece of cake
            - Now when this is meant to be changed to Hindi

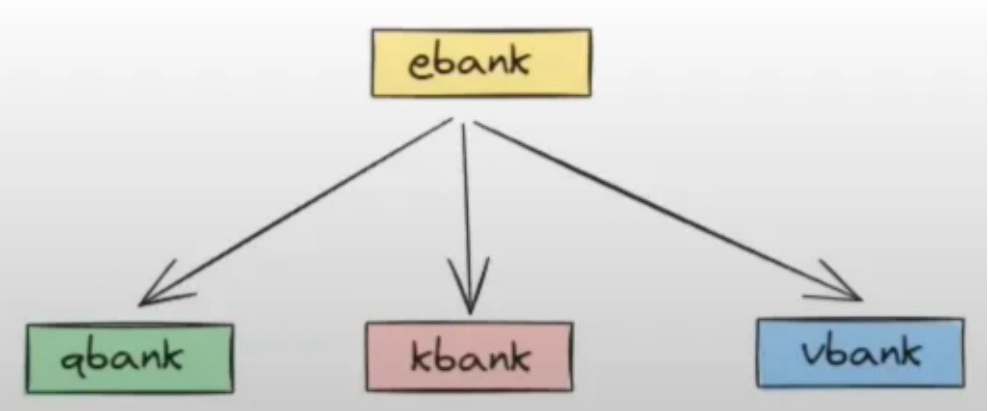
              we can say (cake ka tukda) but this sentence also means (bohot assan kaam).

Query, Key and Value

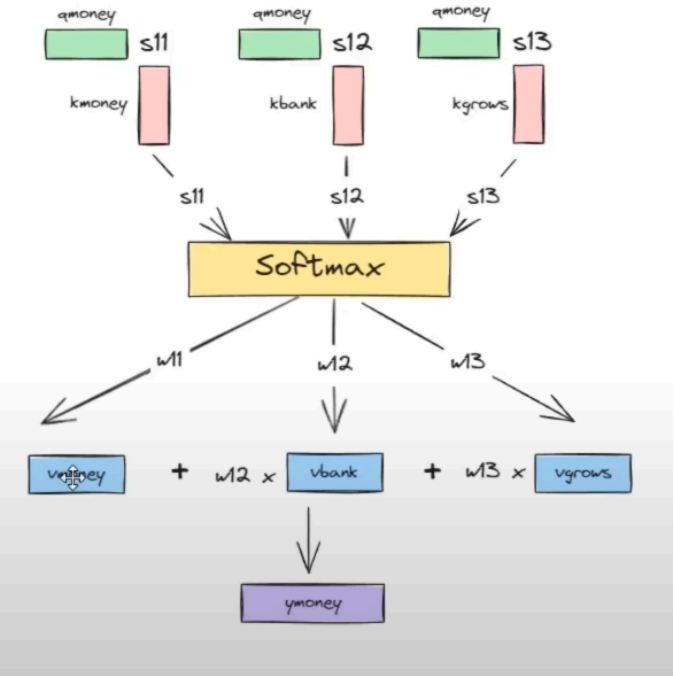
- Every word embedding is used in three ways



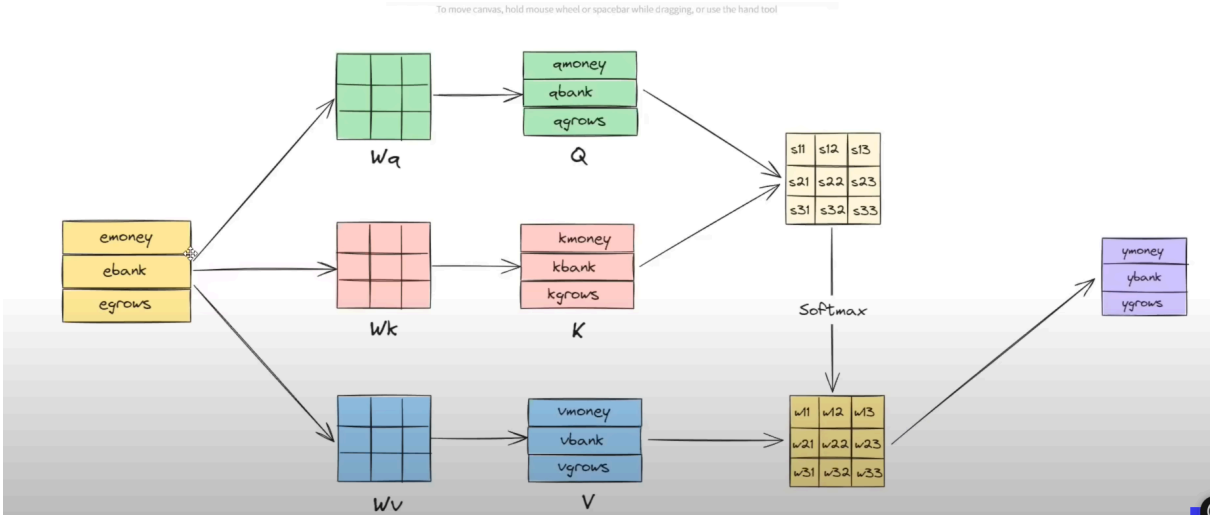
- Problem → Separation of Concern
  - If all three are made to do from one vector it will lose its exact inference.
- Solution →



  - Divide them into three components each component has its own part which makes the tasks more accurate and more efficient.



- How to make the three components from EV?
  - Main thing is By Using and Training with Data.
  - Mathematically making new vectors from one vector can be done with mainly 2 techniques in Linear Algebra
    - Linear Scaling (increasing the magnitude)
    - Linear Transformation (multiplying the vector with Matrix)
  - We use Linear Transformation in most of the cases but the question is → Where do you get the matrix from?
    - Data is used so Initially we use random matrix(weights) and we get some result → then translation happens → mistakes are found → keeping mistakes in mind the number in the matrix(weights) change
    - The above process happens again and again until the correct weights are found.
- Summary



Scaled Dot Product Attention

- d(k) → dimensionality of key vector

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- Why is d(k) used in the equation?
  - This is because of the nature of Dot Products

- In the equation above There is the dot product of these vectors Q(green) and K^T (red).
- The answer is a 3*3 matrix which basically happened when we did dot product of individual vectors and we get 9 number.
- Now as there are numbers it means that we can calculate variance also and the nature of dot products say that

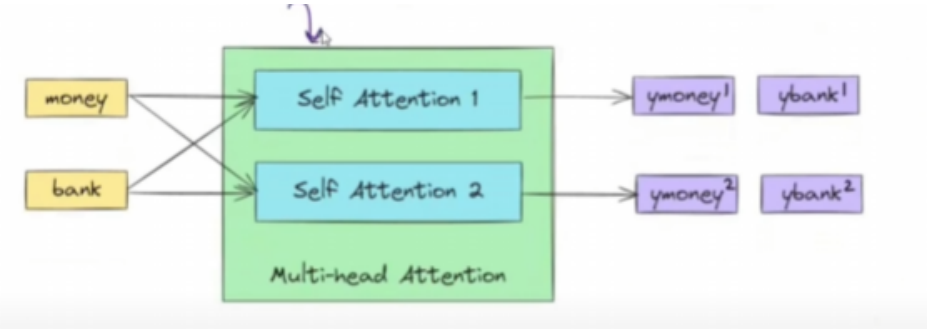
        - Higher the dimension higher the variance.
        - This is problem because if the 3*3 matrix have high variance then the softmax will push it to high probability hence the numbers after softmax application will have huge gap.
        - Now as the model is completely based on training trough back propagation if there are huge numbers difference while training the data the whole focus will go to the big numbers and the small numbers will lose their gradient hence the parameter they are attached to will have a malfunction and make training inefficient.
    - Hence we divide Q(green) and K^T (red) by d(k) to reduce the variance while calculating the softmax.
        - Dimension and variance have linear relationship.
        - X → var(X) and Y = CX
        - var(Y) = C^2var(X)
        - hence the d(k) is inside the root.

Problem

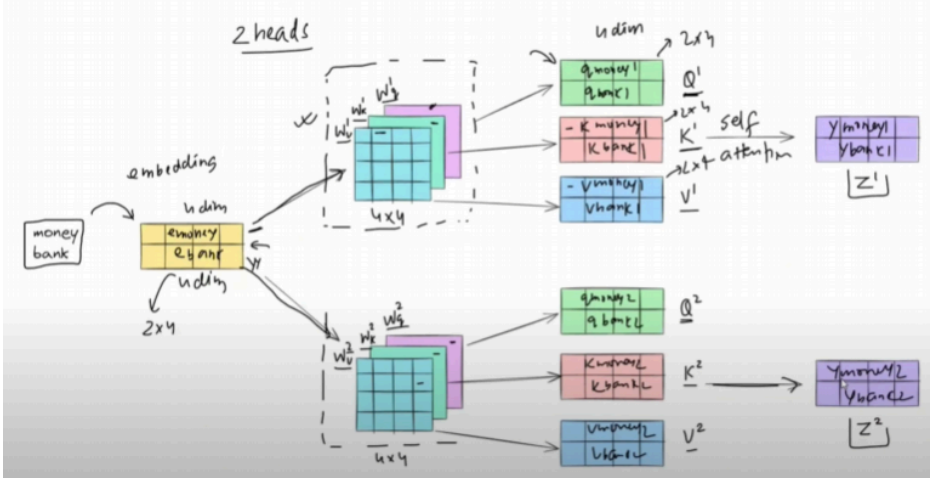
- The man saw the boy with a camera.
    - This sentence is very ambigious because it has 2 meaning.
- The problem with self attention is that it captures only one meaning hence doesnot look at the sentence with multiple perspectives.
    - Multihead Attention Solves this problem

Multihead Attention

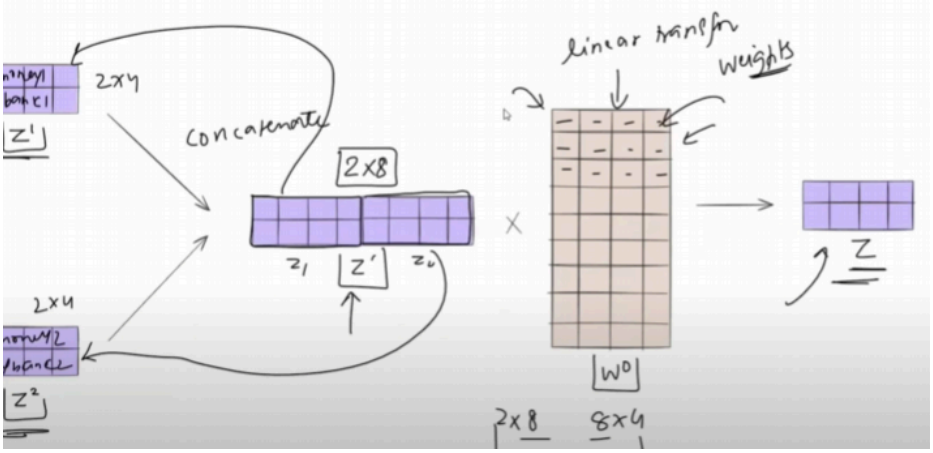
- The concept in its own remain the same but the main part is that we use more than one self attenion to capture all differnent perspectives.



- The word money and bank both go through 2 self attention making 2 differnet money and bank embedding that can be used to make different meanings.



- The word money and bank are transferred to a 2 differnt group of matrices to get 2 different (Q,K,V). This 2 group of (Q,K,V) are used to make 2 differnet contextrual embeddings (Z^1, Z^2) .



- (Z^1, Z^2) undergo concatanation and then the size of the input and the output should be similar hence the concatanated vector is multiplied by a

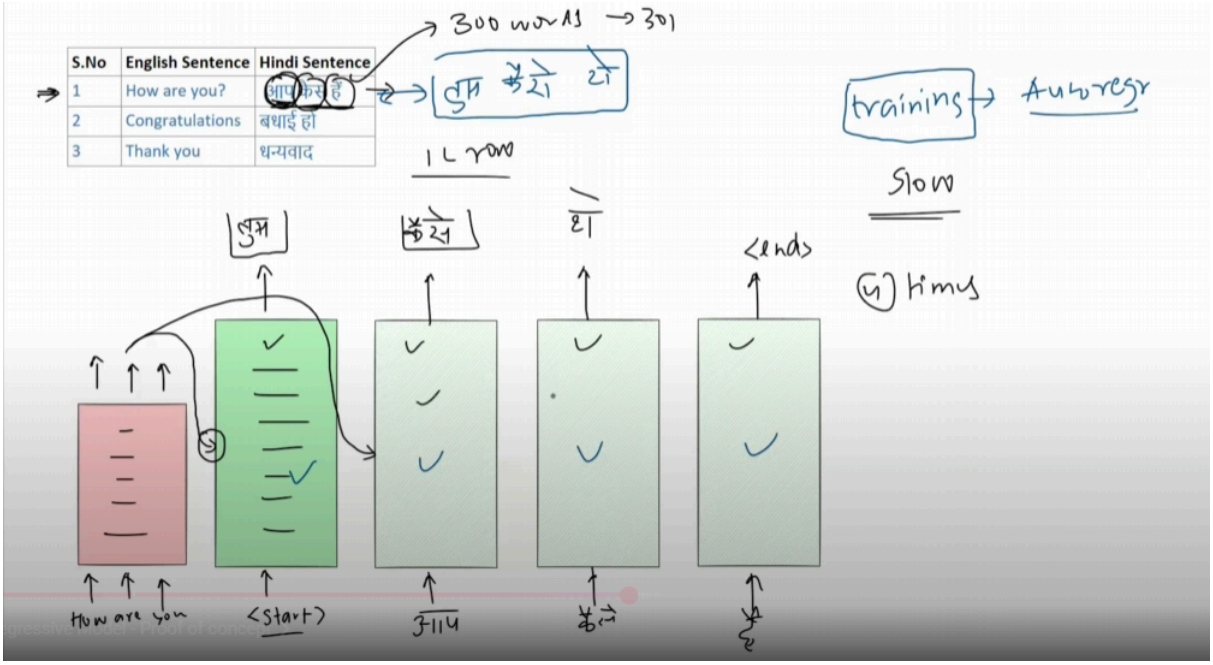
another matrix as shown which causes linear transformation and makes the final vector Z.

- BONUS POINT
  - In the actual papers of ***Attention is All you need*** the excample they have used has 8 different self attention hence 8 perspectives have been analized.
  - The inital vector is 2*512 istead of 2*4 that we used and then the same process happens.
  - The Q,K,V vectors are 2*64 because the matrix used to make them into Q,K,V are 512*64. Hence the dimension are reduced.
    - To reduce the total computations.
  - Here the concatanated vector is actually 2*512 but it still goes through transformation with w 512*512 matrix and we get an output of 2*512 matrix.

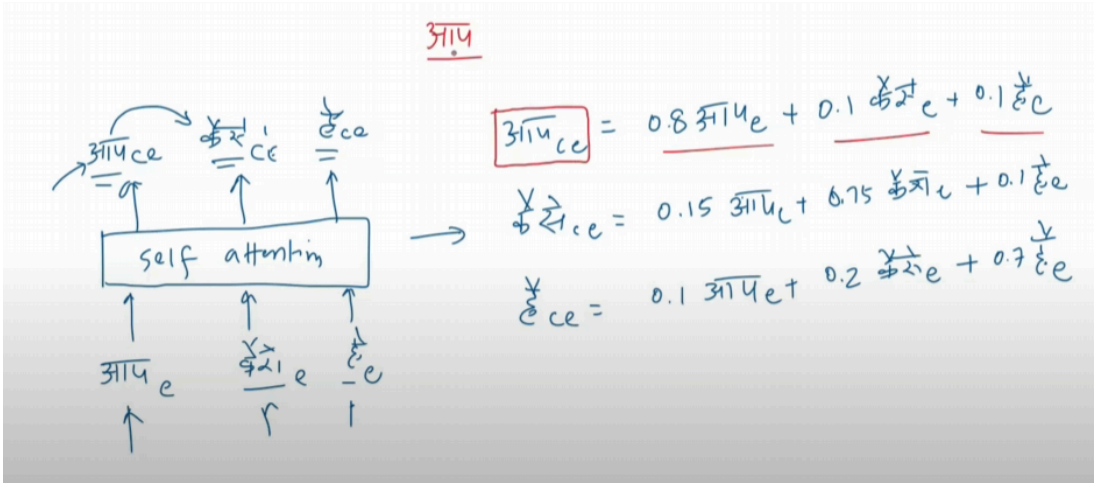
Masked Self Attention

- We need **masked self-attention** mainly to make **autoregressive decoding possible** in Transformers
- *"The Transformer decoder is autoregressive during inference time and non-autoregressive at training time"*
  - Inference → Prediction Time
  - Auto Regressive → class of models that generate Data points in sequence by conditioning each new point on the previously generated points. (In context to deep learning)

Training Process



- Training is Auto Regressive meaning it takes the previous input to generate the next steps.
  - Problem → Make the process very very slow.
- Training can be done in a Non-Auto Regressive (PARALLEL) way also because of the concept called “Teacher Forcing”.
  - Where you have to force the value from the data insted of the previous step during the initial steps of training.
  - Problem → The way self attention works to make the GCE it uses the components from the tokens of the next word to make the token for the previous word BUT…. during training When you are working with “AAP” you dont know the token values of KAISE and HAI.
  - We can do this while training because we have the data sets are available and hence we can use them.
- IN ML YOU CANNOT DO SOMETHING IN TRAINING BUT NOT IN PREDICTION
  - Model will give great outputs while training when it will get exposed to the real time data.

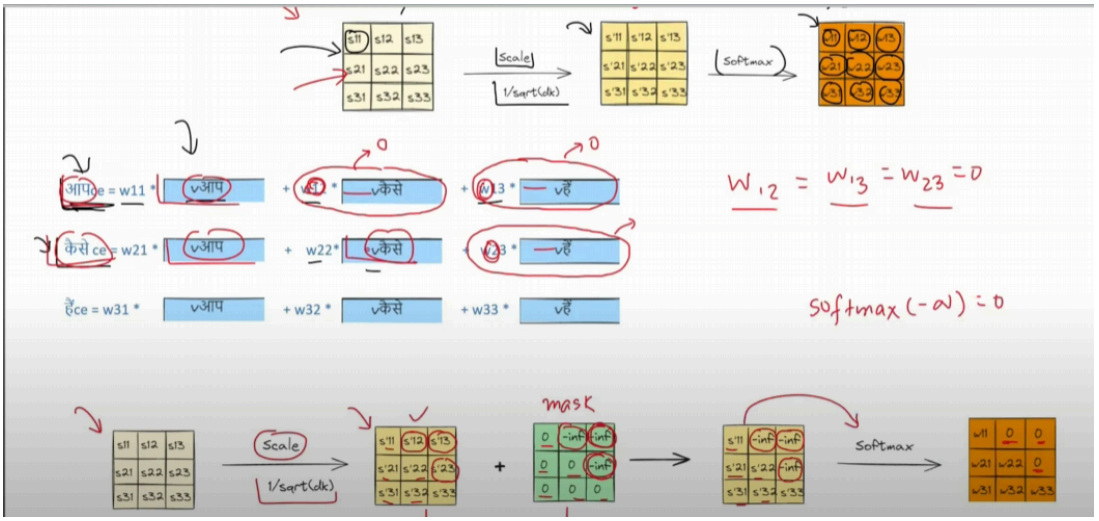


Summary of the Problem

- Auto-Regressive training make the training process very very slow.
- Non Auto-Regressive training make the process really quick but leaks a lot data is not reliable.

Solution (MSA)

- If we look at the equation of the GCE we see that in making the GCE of AAP we donot need the embedding of KAISE and HAI. While doing KAISE we donot need the embedding of HAI.
    - Hence we just want to make them Zero.

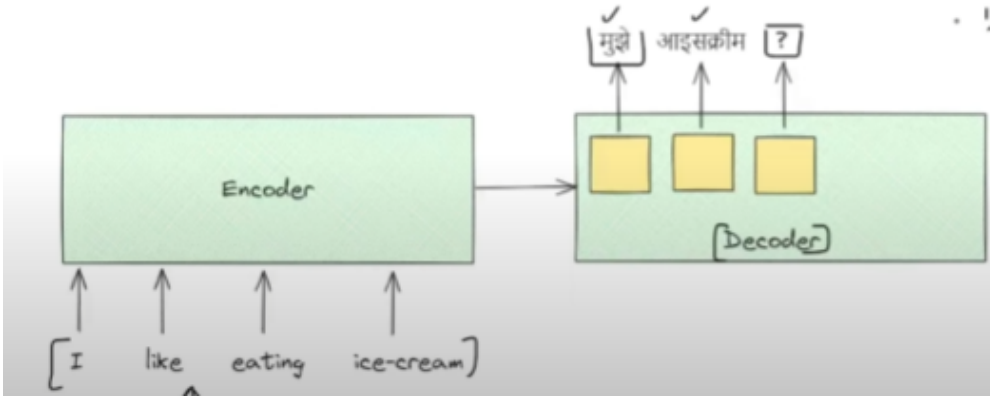


    - HOW??
        - We add a Mask matrix(Green matrix) to the matrix that we after doing the scalling dot product operation. ***SOFTMAX make any***

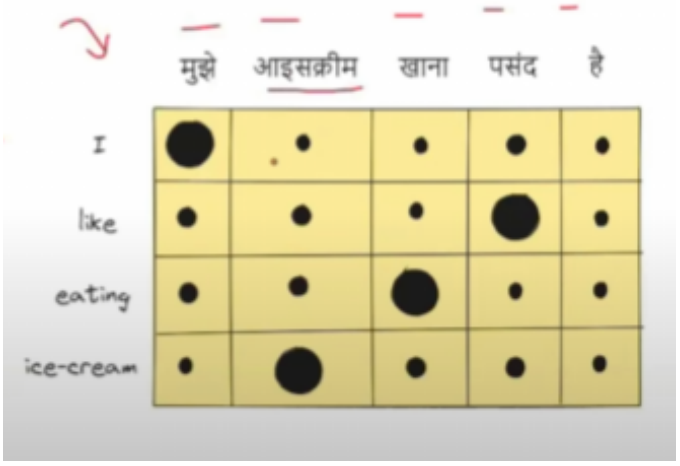
number tending to -ve infinity to ZERO

Cross Attention

- It is a mechanism used in tranformer architecture, particularly in tasks involving sequence-to-sequence data like transition or summarization.
- It allows a model to focus on different parts of an input sequence when generating an output sentence.
- Words are Generated in AR way in the decoder.
  - At t=1 we got “MUJHE” and “ICE CREAM” at t=2.
  - BUT What defines the word coming at t=3.



  - What are the previous words.?
    - How do we get this → self attention
  - And, What is there in the input sentence.? or How is the input sentence in english related to sequence sentence in Hindi.?

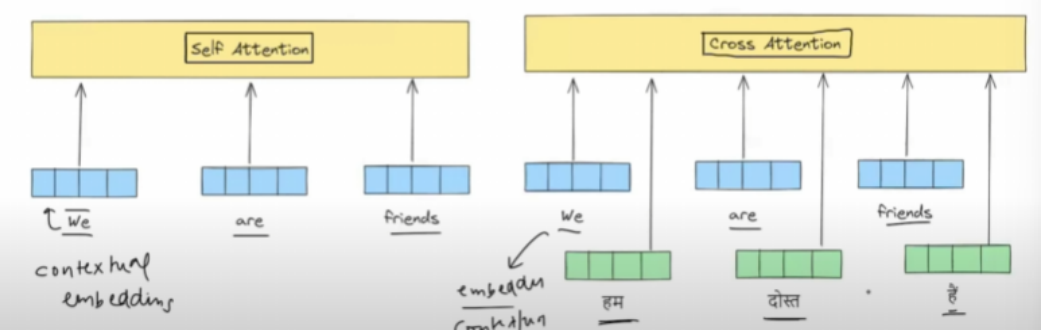


- We want to find out that how well is “Like” related to ‘Pasand”.

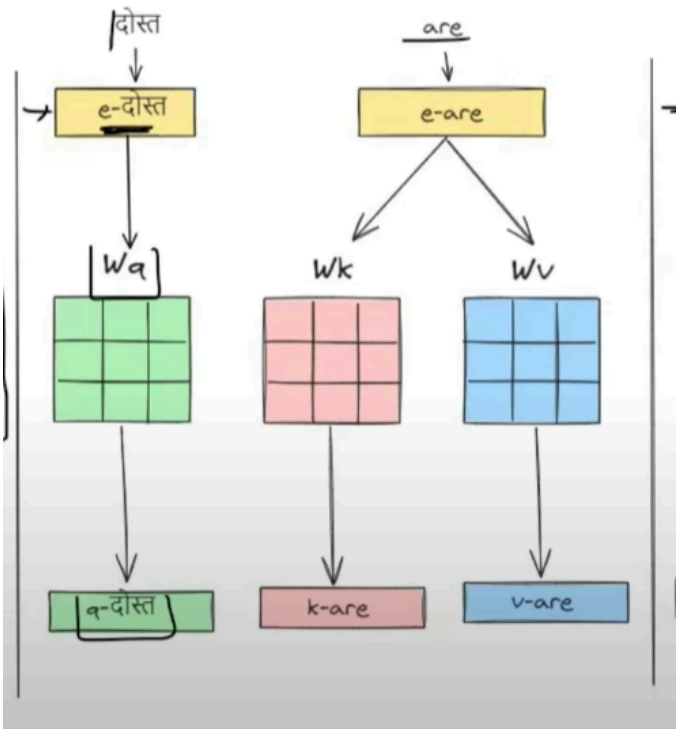
- When you are finding similarity in one sequence it is done with Self Attention. WHERAS If you want to do the same with 2 differnet sequences then it is done with CROSS ATTENTION.

Self Attention VS Cross Attention

- Inputs
  - Self Attention → CE of input from only the one sequence
  - Cross Attention → CE of input from both the sequences



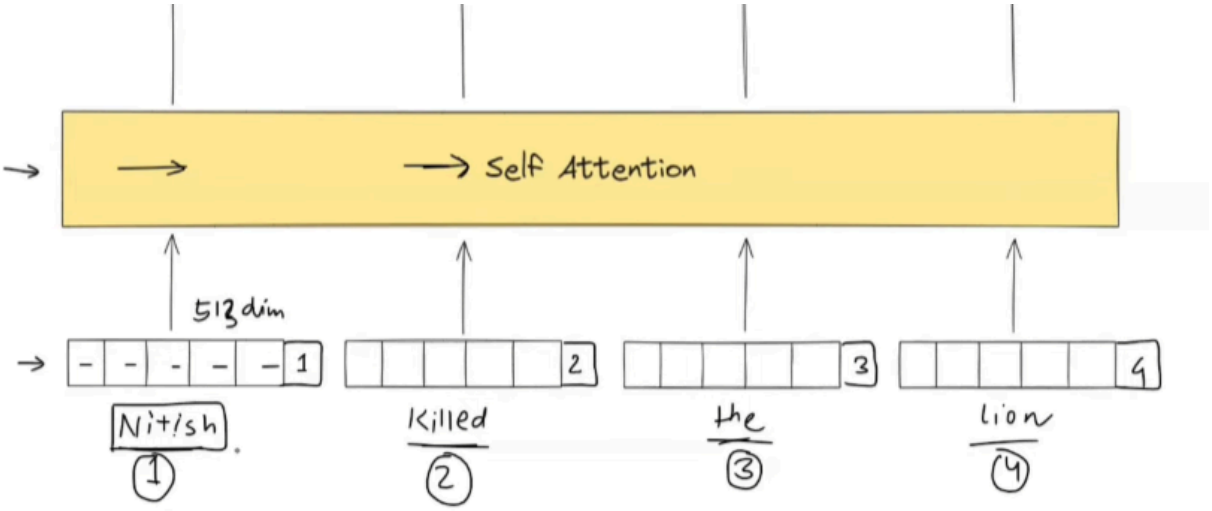
- Working
  - Self Atention → (Q,K,V) vectors are made from the one vector.
  - Cross Attention → Q vector is made from the input that Decoder gets basically (OUTPUT SEQUENCE) and the (K,V) vectors come from the Encoder (INPUT SEQUENCE).



Positional Encoding

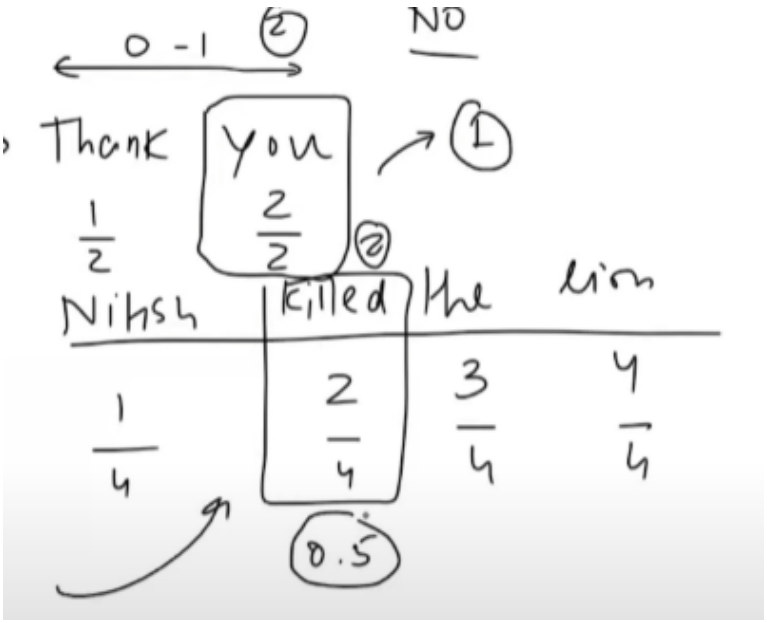
- One of the perks of SA is that it process all the words in parallel.
    - Problem: The position of the words become abigious as all the words are processed in one go.

First Principle Approach for PE



- We could just start with adding the position of the word to end of the vector by adding a dimension.

- PROBLEMS:
  - UNBOUNDED:
    - There is no Upper limit, Hence there could be like 100000 words.
      - As NN work on the logic of back propogation and hates huge numbers will create instability and make the gradient unstable.
    - SOLUTION:
      - Normalization → It will not work as the value of 2 same position in 2 differnet words will become different and there will be no consistency



  - DISCRETE NOS:
    - The here used are Discrete and the Neutral network want Continous number as it perfers them as discrerte number often disturb teh gradient slope.
  - RELATIVE POSITION:
    - We can figure out the absoulte position of the word but the relative position of the word is not difficult to process.
    - Why is this important?
      - The model doesn't care if the subject is at position 5 and the verb is at position 6 (a difference of +1), or if the subject is at

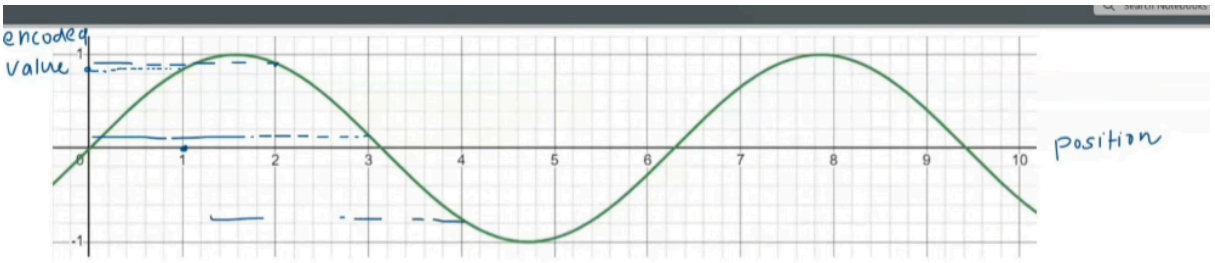
position 105 and the verb is at position 106 (also a difference of +1).

- The *relationship* is the same: the verb immediately follows the subject.
- The linear relationship in positional encoding makes it easy for the model to "see" this consistent relative distance.

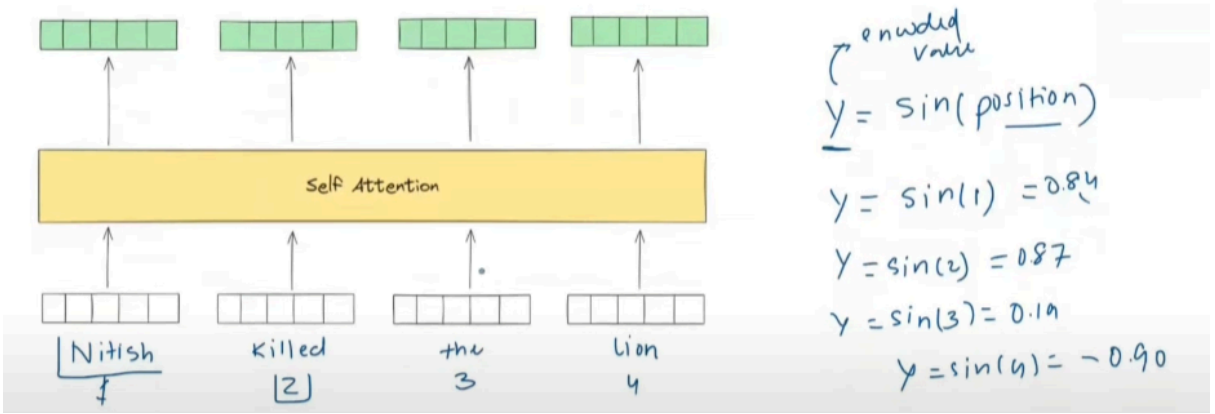
- SOLUTION:
  - What we need here is a function that is BOUNDED, CONTINOUS and PERIODIC.
  - This signals us to Trignometric Functions.

SINE and COSINE

x→position of the word



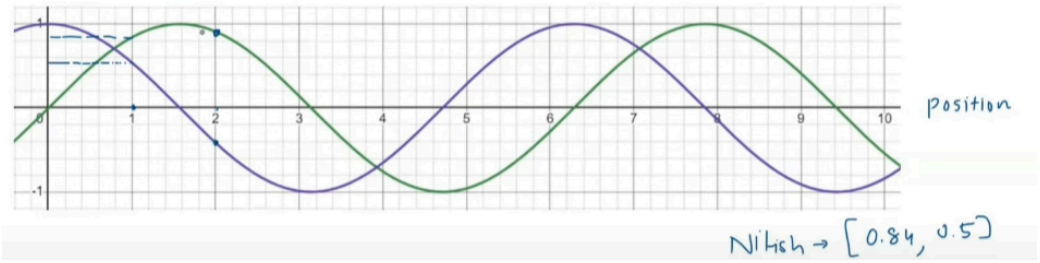
Y=sin(x)



- PROBLEM:
  - Preiodicity → 2 values of x will have the same value of y and hence if will give this indication to the SA that both the words have the same

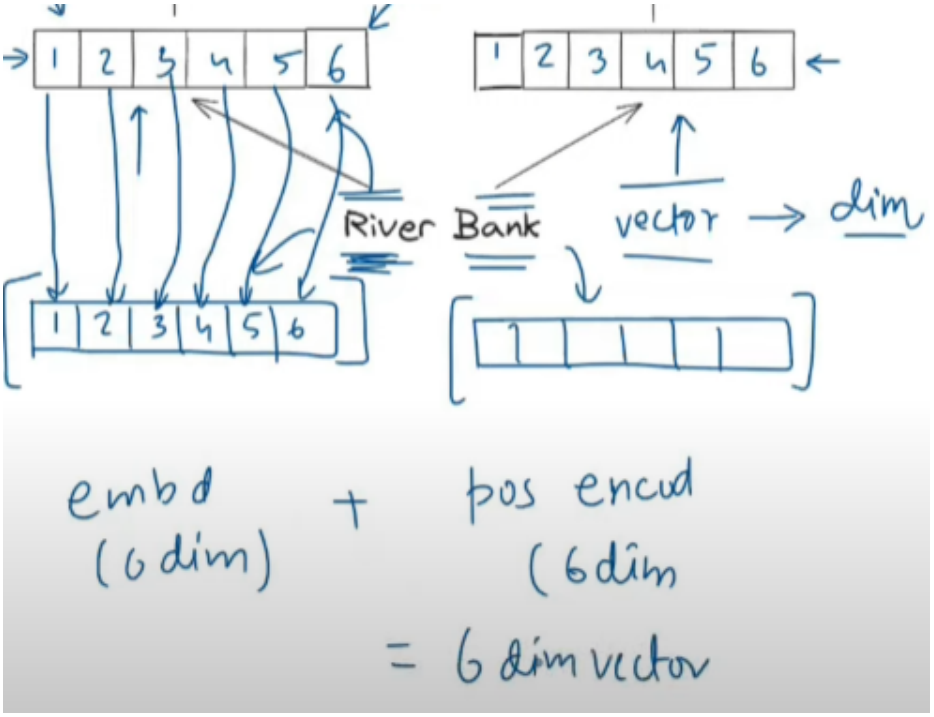
position.

- SOLUTION:
  - MULTIPLE FUNC → We can use both sine function and cosine function.
    - In this case the value will be vectors instead of scalar.



    - This will decrease the number of time the value but will repeat but will not eradicate the problem.
  - So, to completely erradicate this problem we add another pair of Trignometric functions that are Sin(x/2) and Cos(x/2) and we keep on adding this pair.

- Positional encoding will be set of number in form of Vectors.
- The dimension the PE vector will be same as the dimension of the Embedding vector.



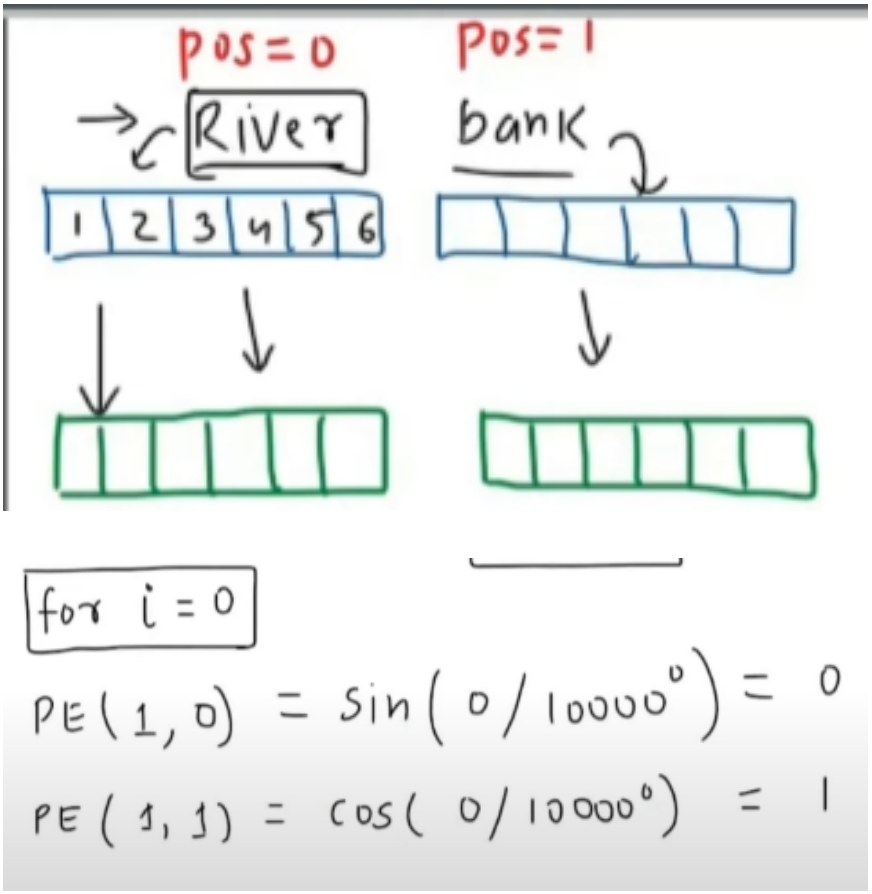
- We add the vectors and not concatanate them becasue concatanation will increase the size of the vector and it make things slow and difficult for the self attention,
- How are these values determined?
  - So we use Sine and Cosine pairs everytime we need to add a dimension,
    - In the above example → (position = x)
      - For x=1 (basically first word)
        - 1 → sin(1)
        - 2 → cos(1)
        - 3 → sin(1/2)
        - 4 → cos(1/2)
        - 5 → sin(1/3)
        - 6 → cos(1/3)

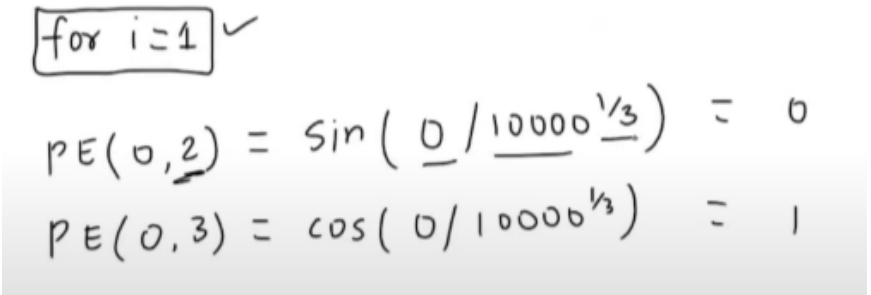
- The formula used to find the values in the PE vector

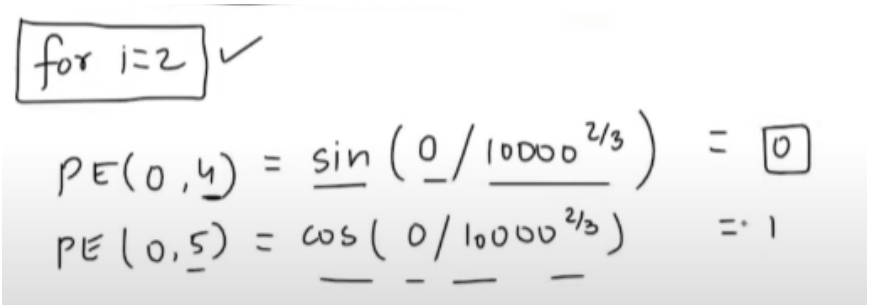
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

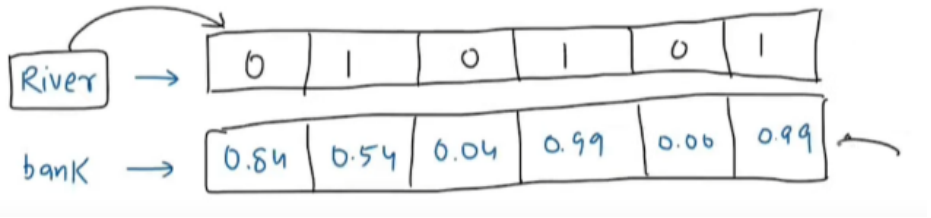
- pos = position on the word (they start from 0)
- i = [0, d/2]
- d = dimensionality of the model

- Excample:



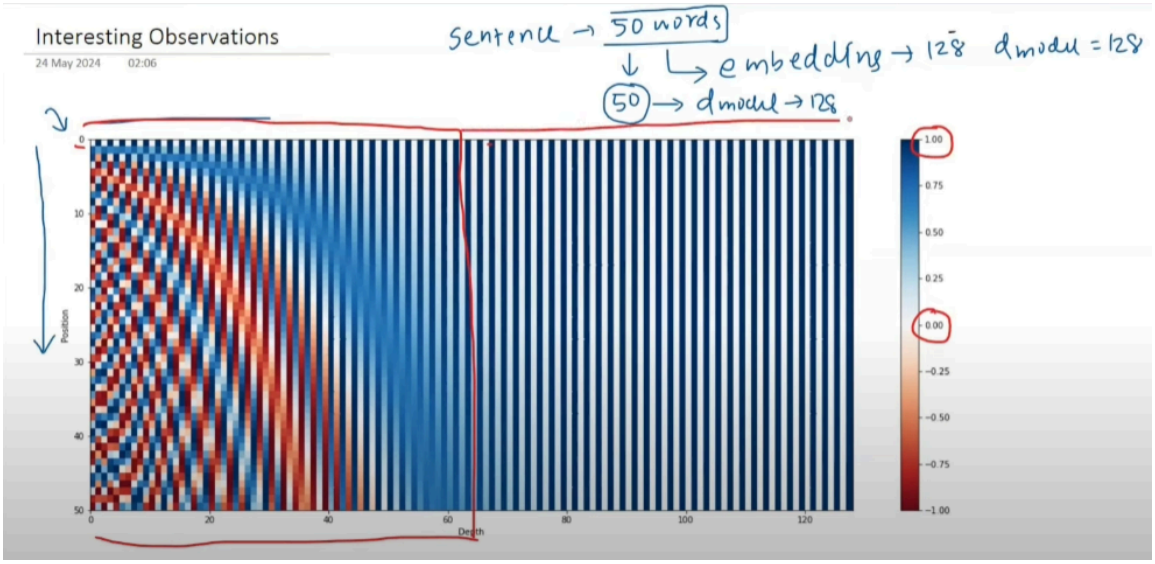






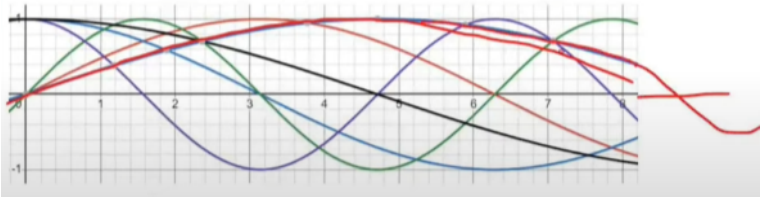
PE Graph (Interesting Obs)

- For every new Sine Cosine pair is added the frequency of the pair is decreased.



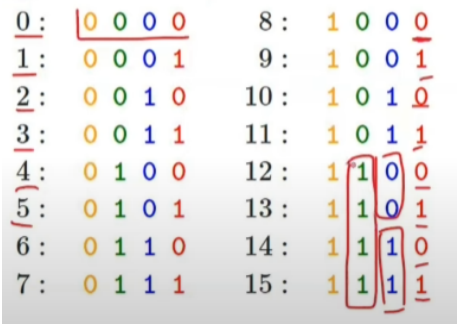
- Observation
  - For lower dimensions the color change very frequenty whereas the color for the higher dimension is either blue and white (0,1).
    - WHY?
      - There are only 50 words hence to a certain point the colors change, If more words are added to the sentence then the

colors will change for the higher dimension becasue as the frequency is decreased.

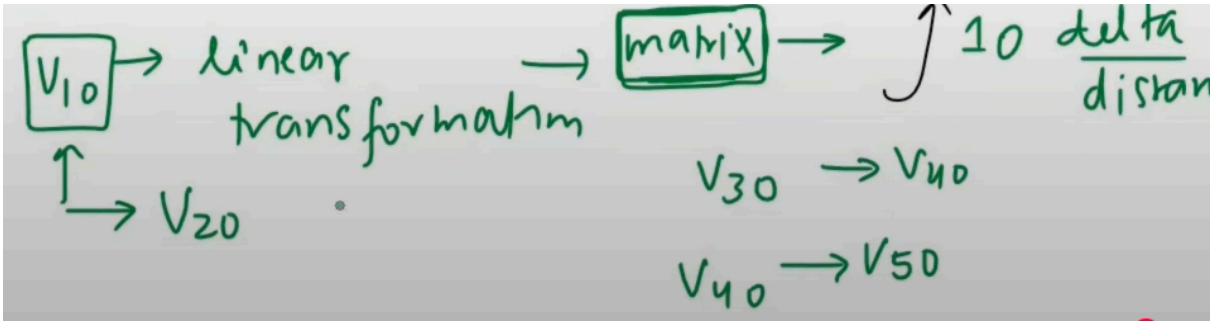


- Imagine like as the words increase the length on x-axis on out graph increase and then the value of sine and cosine that did not cut the x-axis will also cut it and hence creating more value.

- Similarity to Binary Encoding
  - The bits at the ones place change alternatively where as when you move to the tens place the bits change after 2 occurances and on the hundreds place they change after 4 apperances.
  - Here only 4 bits are shown but if these numbers are shown in 8 bits the higher places will have all 0s.
  - Hence, Positional encoding uses the same logic but in continoue numbers.

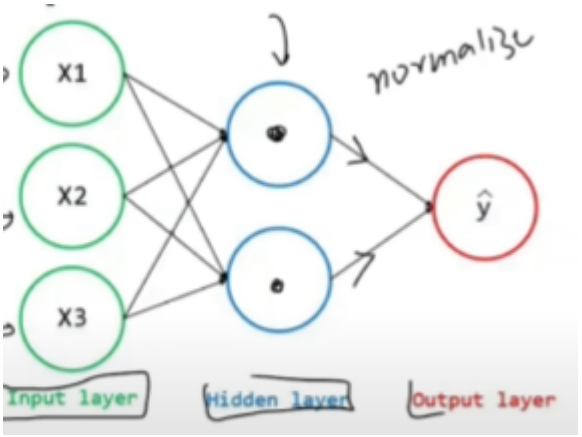


- There exists a linear transformation T(k) that can transform the positional encoding of position (t) to the positional encoding of position t+k:
- e.g. There is a certain matrix when multied with Vector V_x will take you to V_10x.
- There could be a matrix to move from x→7x or 5x.



Normalization

- Normalization in deep learning refers to the process of transforming data or model outputs to have specifiv properties, typically a mean of zero and a variance of one.
- What do we normalize ?
  - We normalize the inputs that we have sent in the input layers.
  - We also normalize the output we get after the activation of the hidden layers.

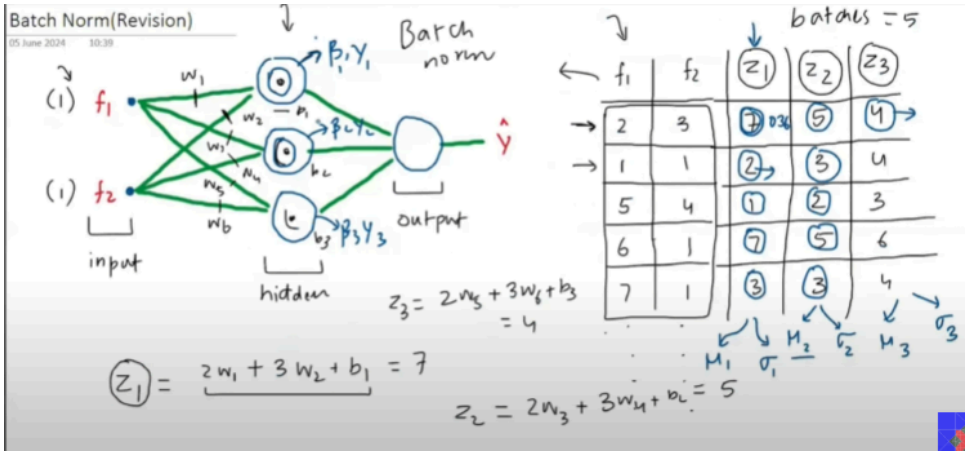


- Why do we do it ?
  - Improves Training Stability
    - Accelerate the training process by reducing the likelihood of extreme values that can casue gradient to explode or vanish.
  - Faster convergence
    - Models tend to converge more quickly because the gradient have more consistent magnitudes. This allows for more stable updates during BP.

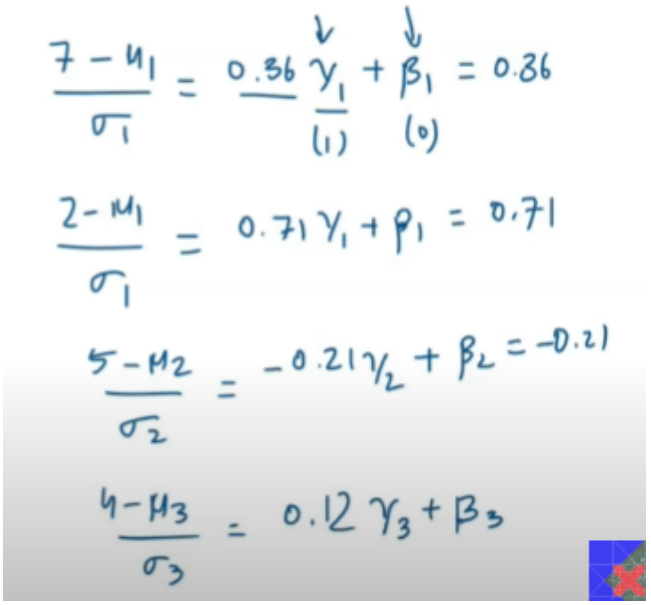
- Internal Covariate shift
  - It refers to the change in distribution of layers input during training.
    - Imagine a neural network with 10 layers. As you train the model:
      - Layer 3 changes its weights.
      - That causes the distribution of outputs from layer 3 to shift.
      - Layer 4 receives inputs from layer 3, so now **layer 4 must adapt to a different input distribution**.
  - Normalization helps to reduce this shift and make the training process more robust.

Batch Normalization

- How does it work?
  - f1 and f2 are the inputs.
  - w1,w2,w3,w4,w5,w6 are attention weights that are used to calculate the activation of the hidden layers.
  - z1,z2,z3 are bacically activation of the hidden layers.
  - The mean and variance of the activation are calculated for normalization.
  - THE NORMALIZATION HAPPENS IN COLOUM FORM. (ACROSS BATCH)



  - y1,y2,y3,y4 and b1,b2,b,3,b4 are basically the values that are we used while training so we can use any data.

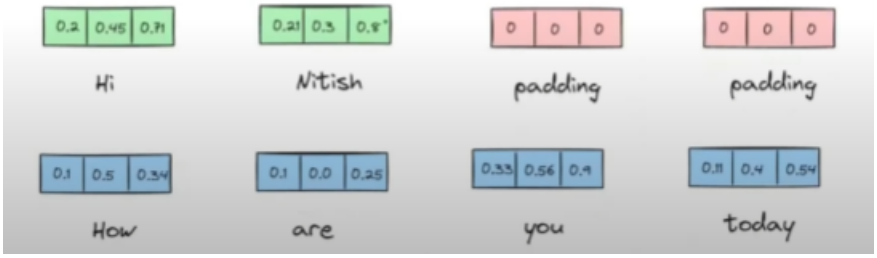


Why we don't use BN?

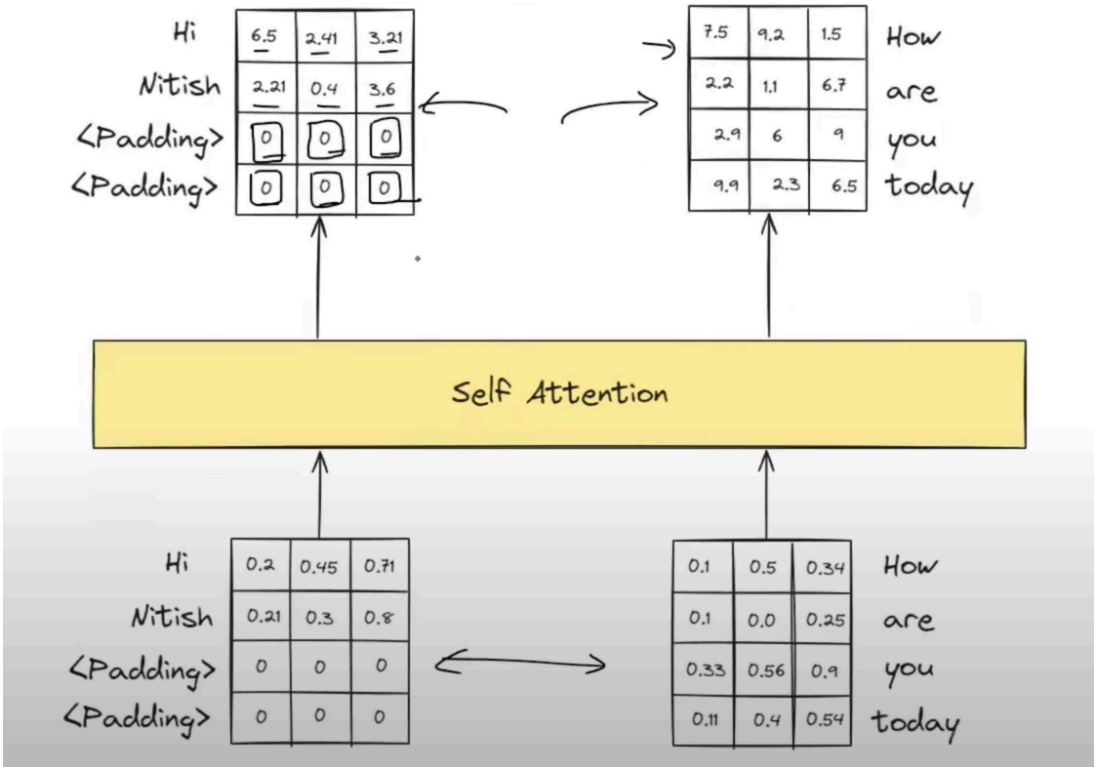
- The main problem with batch normalization is that it does not work well with Self attention.
    - Lets suppose we have these 4 sentences and we will use them in pairs.

Review	Sentiment
Hi Nitish	1
How are you today	0
I am good	0
You?	1

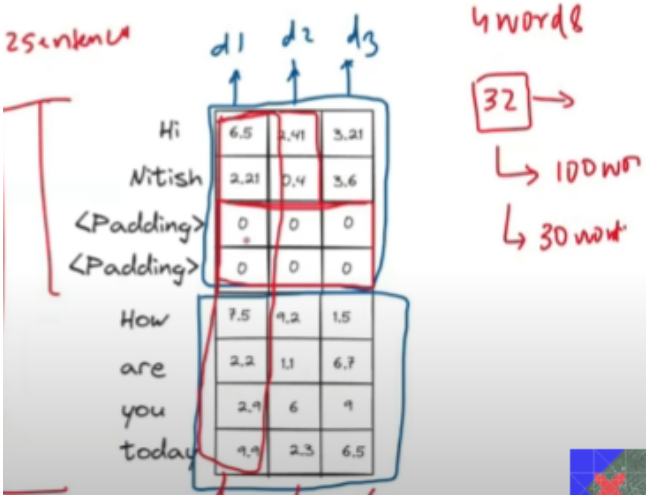
    - As the number of vectors for each sentence should be same we will use padding vectors.



    - We will put both of the sentence to the self attention and the general contextual embedding.

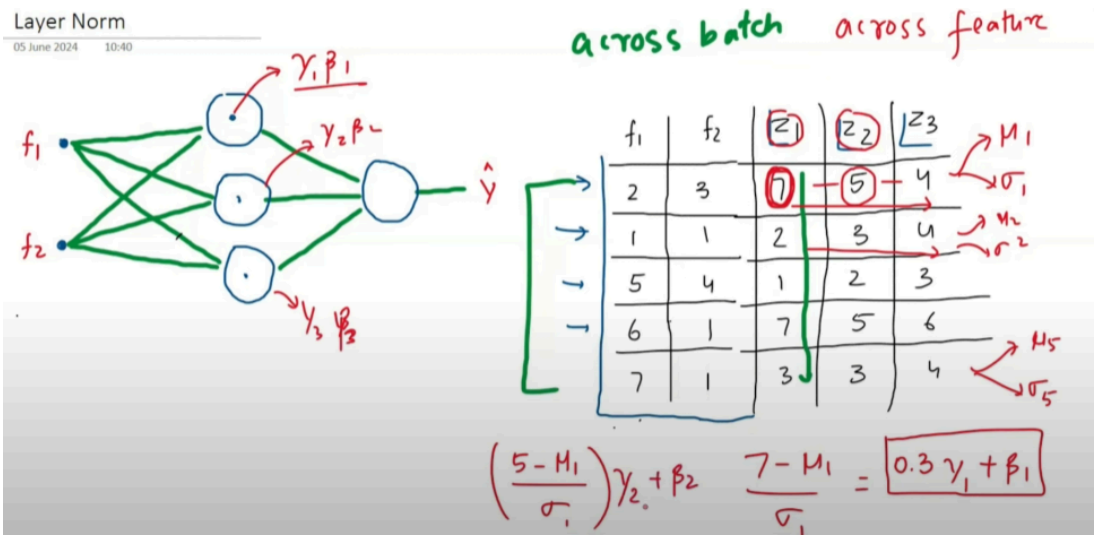


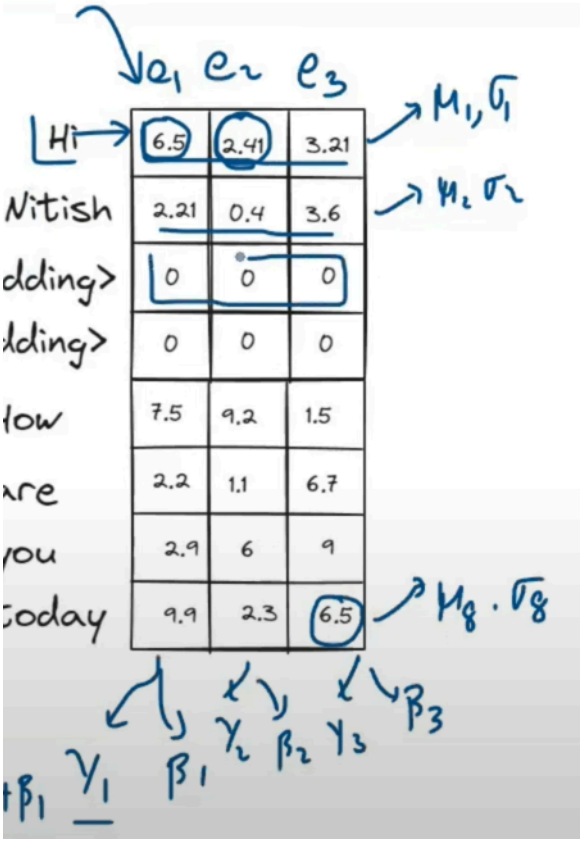
- These GCE are then stacked together and then it is normalized.
- PROBLEM:
  - The padding layer → As there are just zeros that we taken solely for the reason make the sentences of the same length. If there are like 32 sentences and lets say one has 100 words and the other on an avg have 30 words.
  - Then there will be so many padding vectors the it will make things slow and not optimum.



Layer Normalization

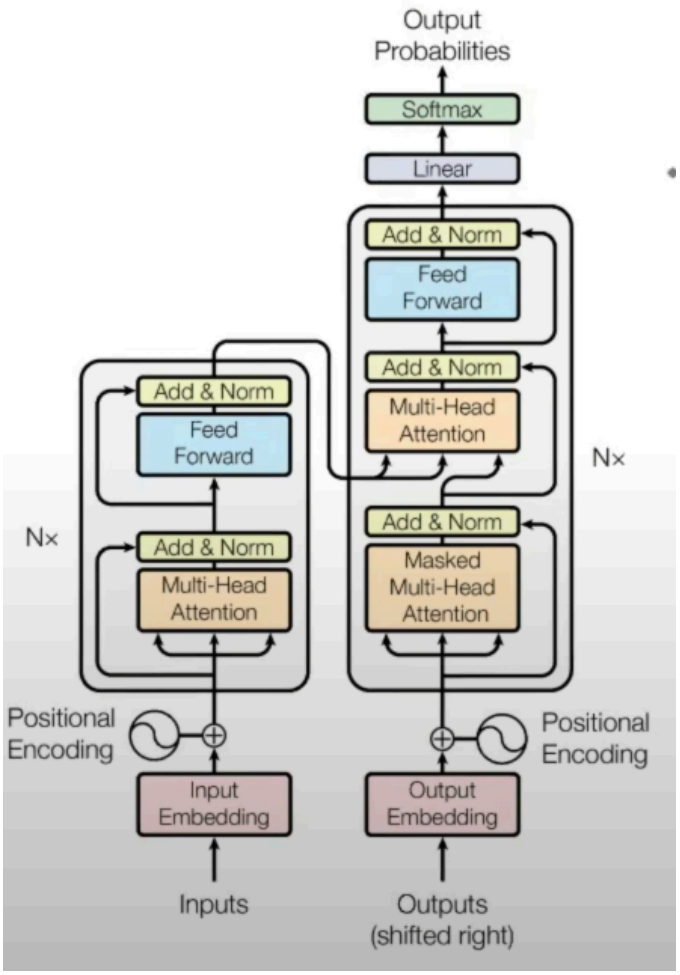
- How does it work?
  - THE NORMALIZATION HAPPENS IN ROW FORM. (ACROSS FEATURES)
  - The mean and the variance will be calculated across the row instead of the coloumn.



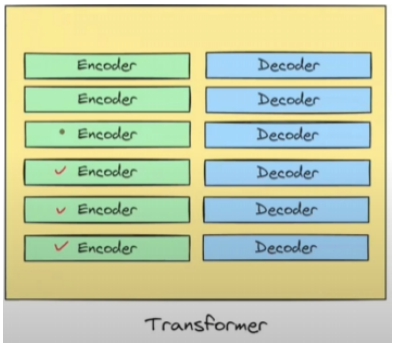


Transformer architecture

- This is actually how a Transformers look like.

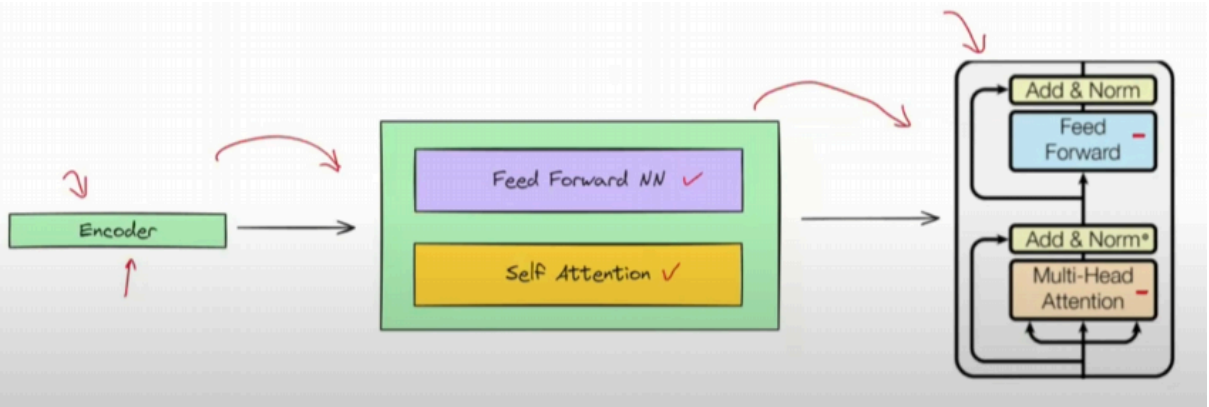


- This is the more simpler version of Transformers.
  - There are mutliple Encoders and decoders in the one transformers. Specifically the paper *"Attention is all you need"* they had mentioned 6 because they got the best results.



- Why more than 1 though?
  - Human languague are very complex and for computers to understand. Hence they need models that have high representational powers.

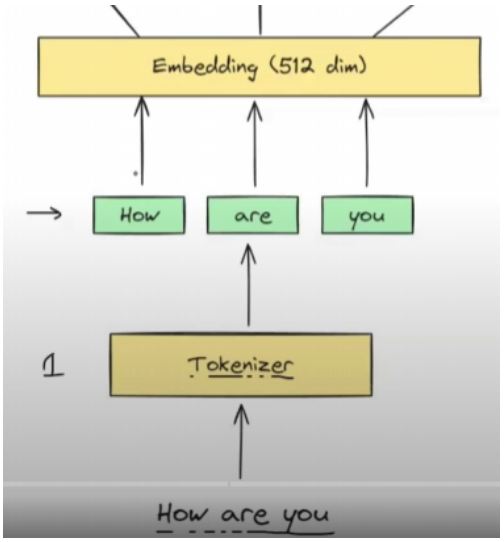
Encoder Architecture

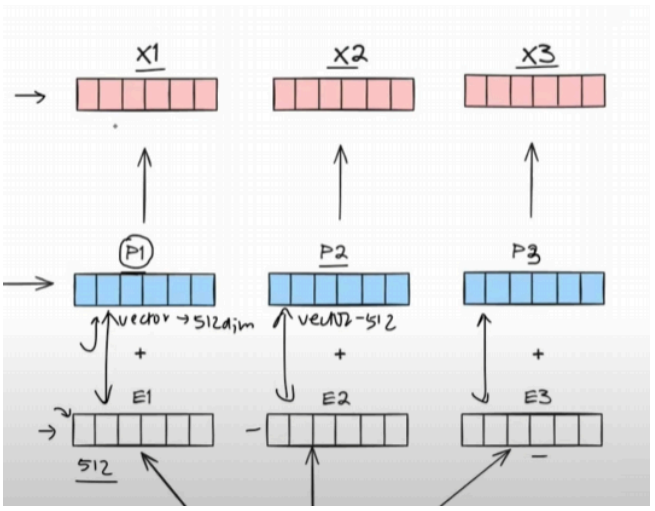


- Between the main input and the final output of the transformers there are 6 of these attached in a sequence and where the output of one becomes the input for another.

Input

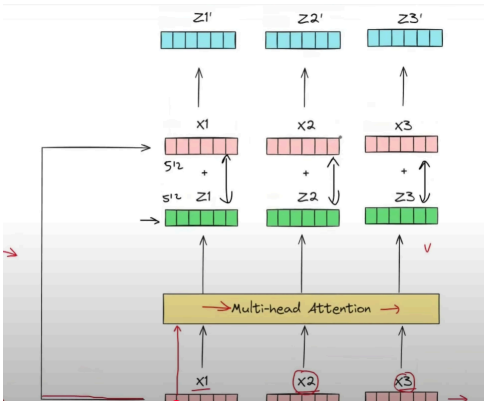
- Inputs are divided into 3 steps.
  - Tokenizer → words are converted into numbers(tokens).
  - Embedding → Each word in made into a **512 dimension vector** (Word embedding)
  - Positioning → Word embeddinga are attached to Positional encoding vectors. (WE+ PE)
  - X1,X2,X3 → Properly positioned embedding.
    - Final input that Enter the Encoders.

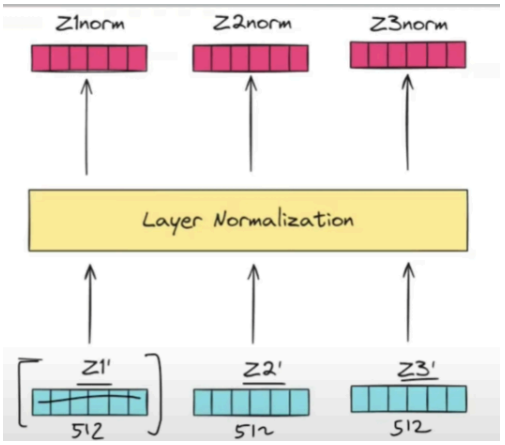




Self Attention

- Self Attention → Inputs are added and convert them to Task specific contextual embedding. (Z1,Z2,Z3)
- Residual Connections → The connection that takes the Positional Word embeddings to the Task specific Word embeddings.
    - These are then added and converted to Z1′,Z2′,Z3′.
    - These are normalized and converted to Z1norm,....

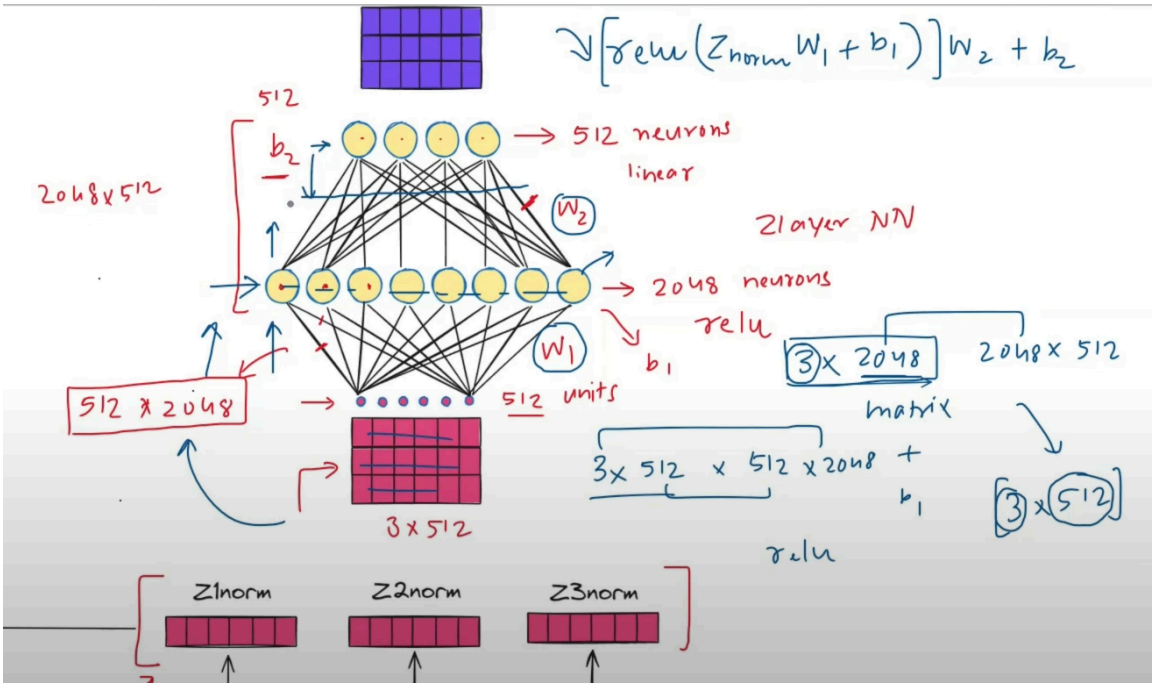




FFNN

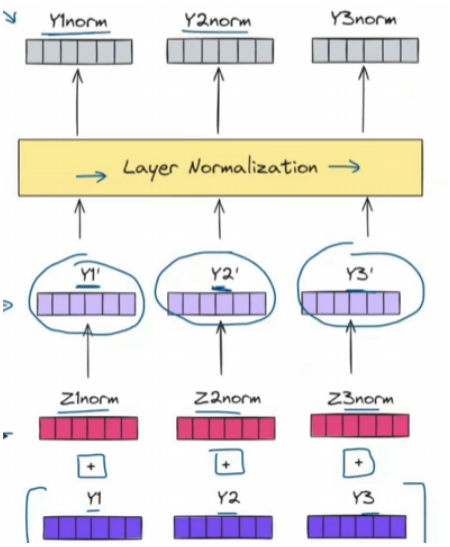
- The Z1norm, Z2norm and Z3norm are stacked to each other. (3*512)

- There is a set of 2048 neurons and all of the sentences directly go to the neurons.[(3*512) . (512*2048)→Weights)]



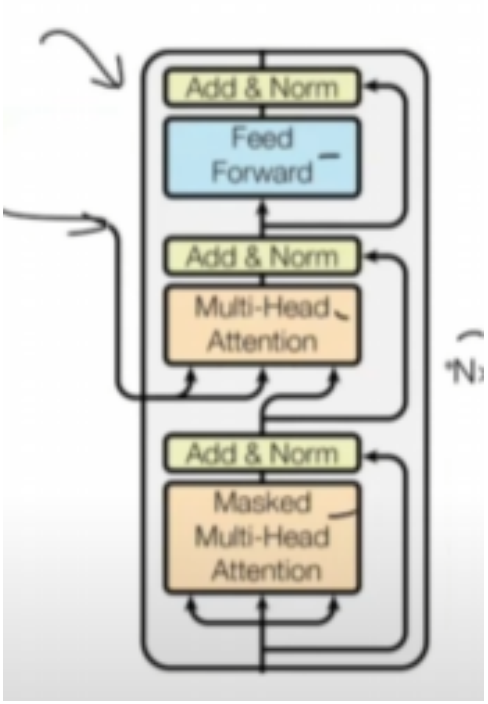
- The next set is of 512 neurons [(3*2048) . (2048*512) → Weights].

  - Summary of What is happening in FFNN
- Finally, we get the Y1,Y2,Y3 vectors(blue one's).



- We add the Z1norm,.. vectors to the Y1,.. and then they are normalized the same way.
- Y1norm,... are added to the next Encoder and then the same thing happens and then they go to the another encoder.
- **EVERY ENCODER BLOCK HAS ITS OWN DIFFERENT PARAMETERS AND CHANGED DURING BACK PROPOGATION.**

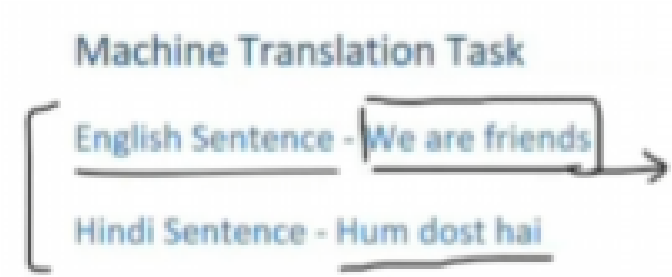
Decoder Architecture



- Between the main input and the final output of the transformers there are 6 of these attached in a sequence and where the output of one becomes the input for another.

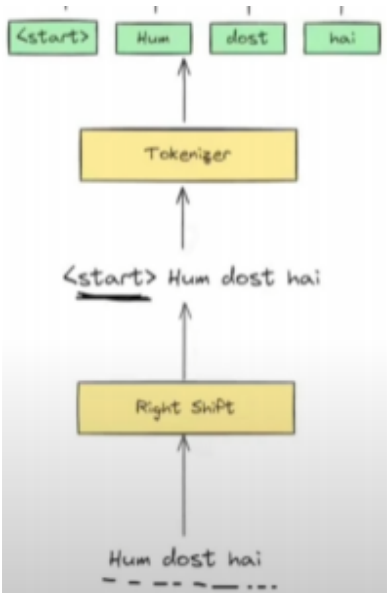
Setup

- English Sentence is sent to the Encoder to make the embedding and then they are added to the Decoder.
- Hindi Sentence becomes the Input for the Decoder

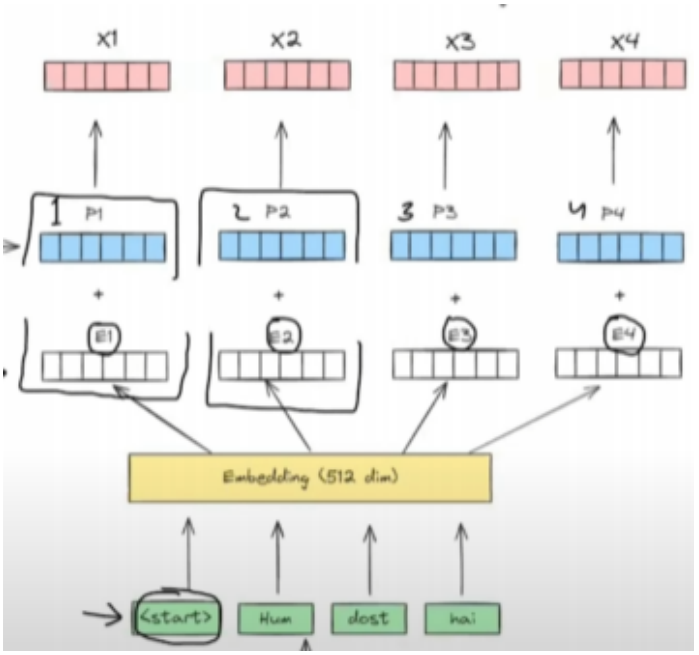


Input

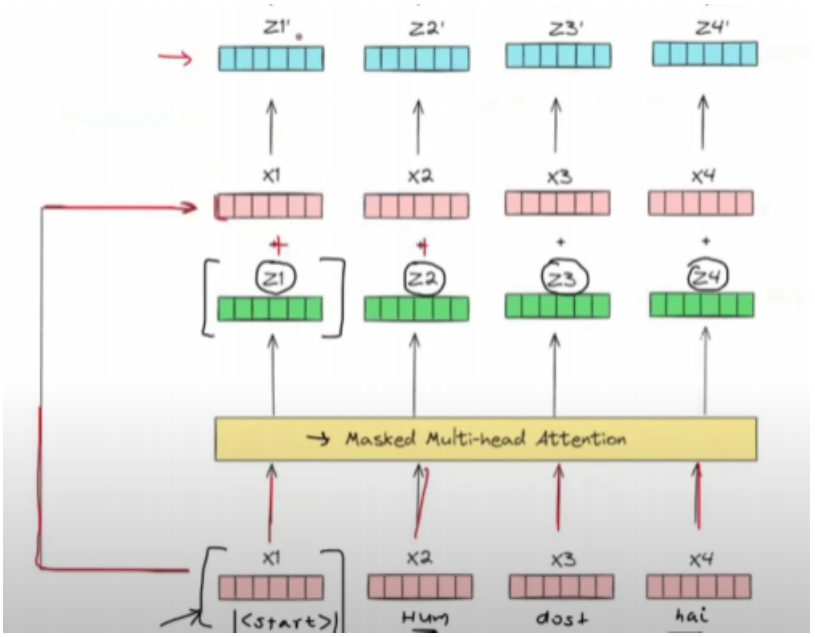
- Input is divided into 4 steps.
  - Shifting → There is a right shift added to the sentence to add the <Start>.
  - Tokenization → Words are converted into numbers(tokens).



- Embedding → Each word in made into a **512 dimension vector** (Word embedding)
- Positioning → Word embeddinga are attached to Positional encoding vectors. (WE+ PE).

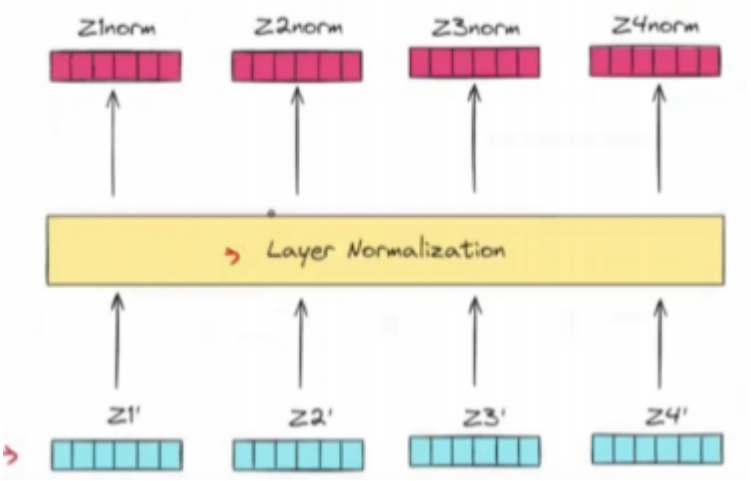


Masked Multi-Attention

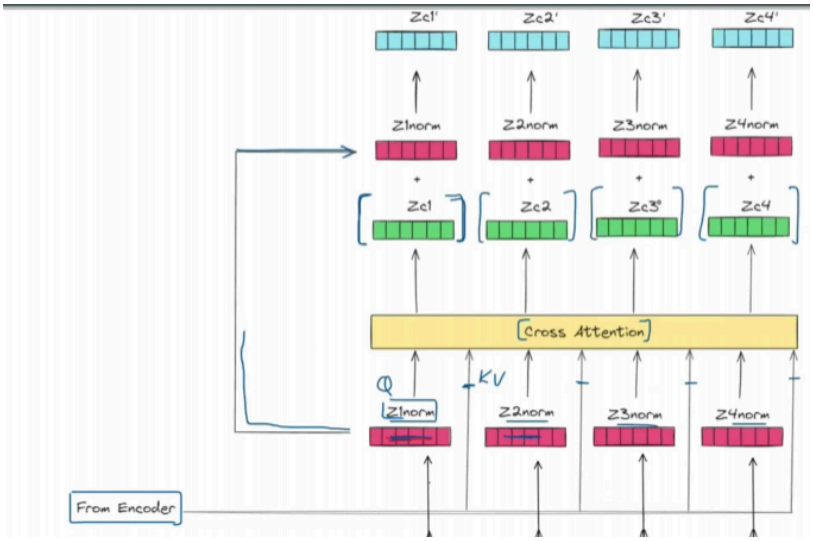


- Masked Multi-Attention → Generate embedding in a way that it does not contradict our own training process and the inference process.

- These are then normalized.

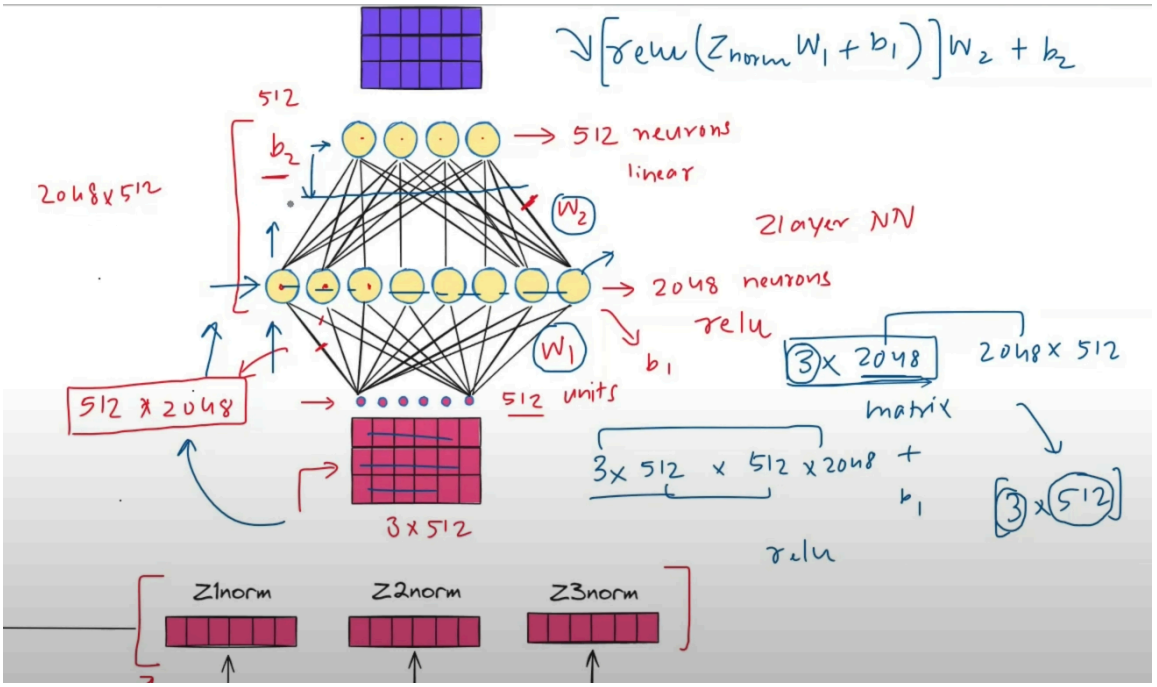


Cross Attention



FFNN

- The Z1norm, Z2norm and Z3norm are stacked to each other. (3*512)
- There is a set of 2048 neurons and all of the sentences directly go to the neurons.[(3*512) . (512*2048)→Weights)]

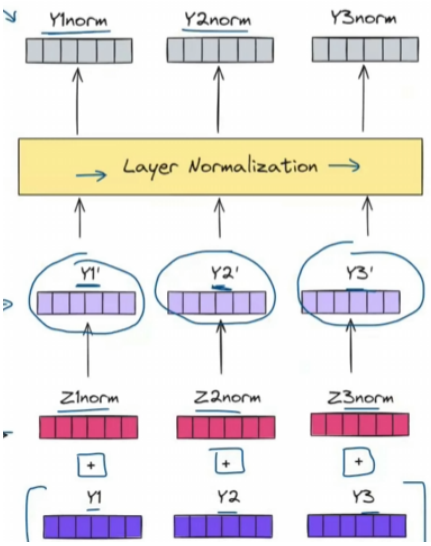


- The next set is of 512 neurons [(3*2048) . (2048*512) → Weights].

$$\left[\text{relu}\left(Z_{norm} W_1 + b_1\right)\right] W_2 + b_2$$

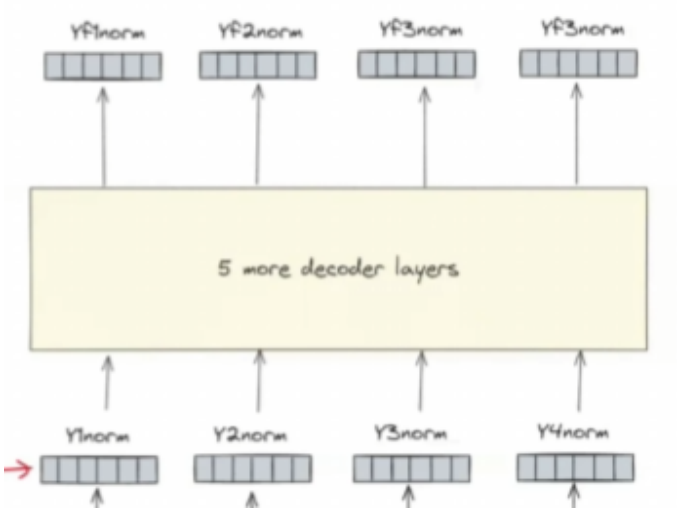
  - Summary of What is happening in FFNN

- Finally, we get the Y1,Y2,Y3 vectors(blue one's).



- We add the Z1norm,.. vectors to the Y1,.. and then they are normalized the same way.

- Y1norm,... are added to the next Decoder and then the same thing happens and then they go to the another decoder.



Output

- Number of neurons in the layer will be the same as the number of unique word that are inserted into the Decoder.