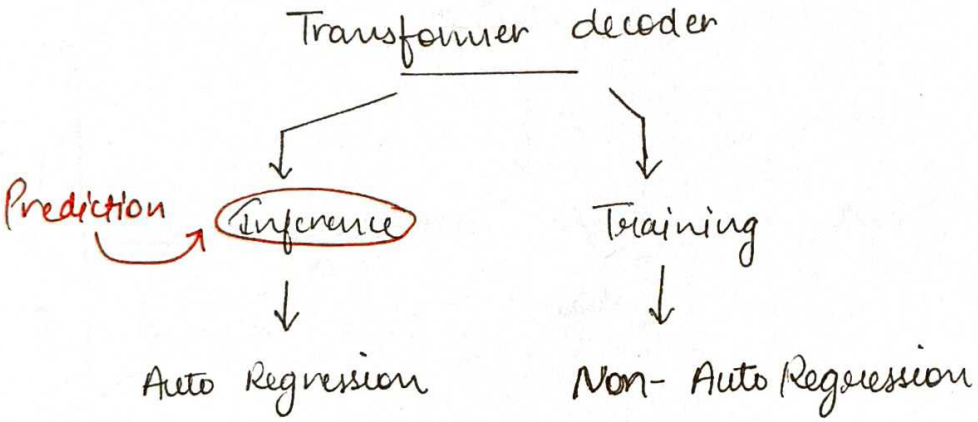
Masked Multi-Head Attention in Transformer

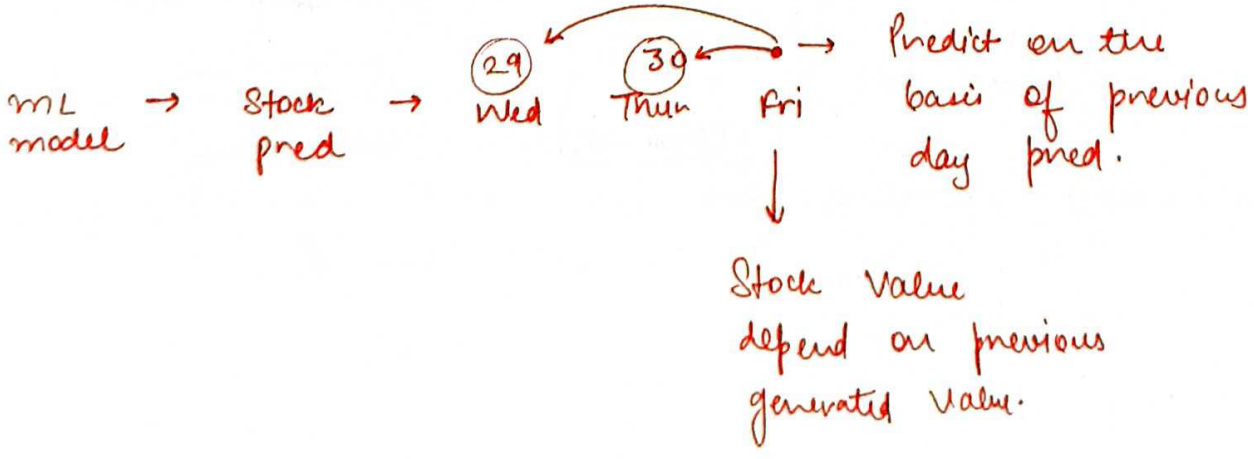
Masked Self Attention

Autoregressive models

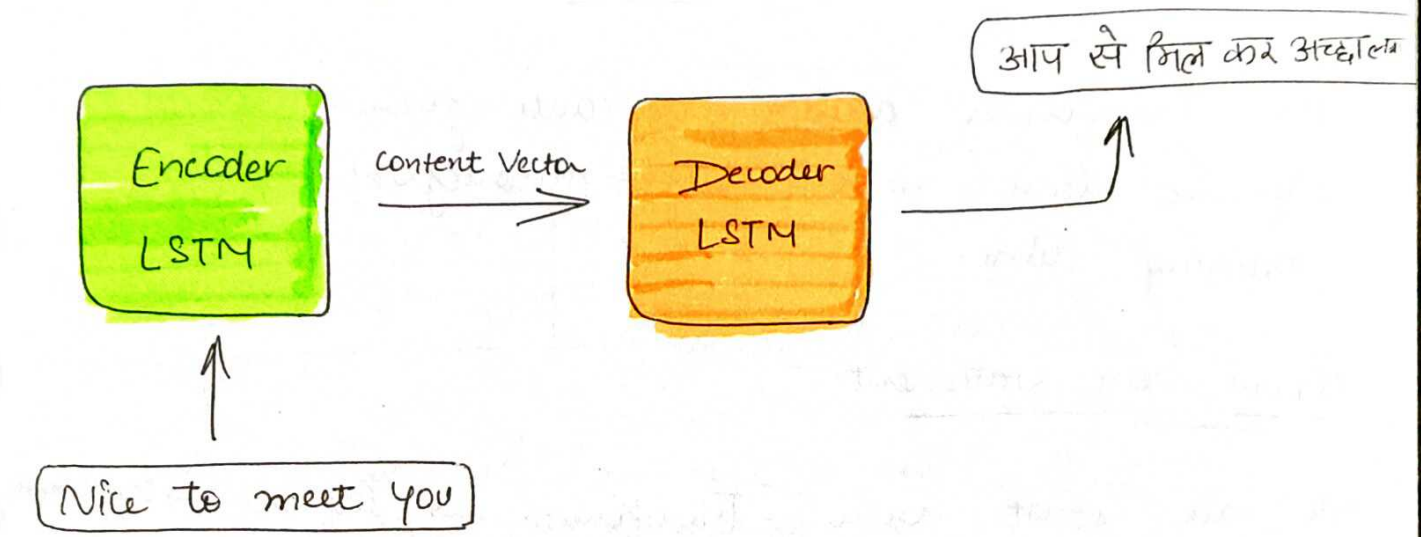
The Transformer decoder is autoregressive at inference time and non-autoregressive at training time:

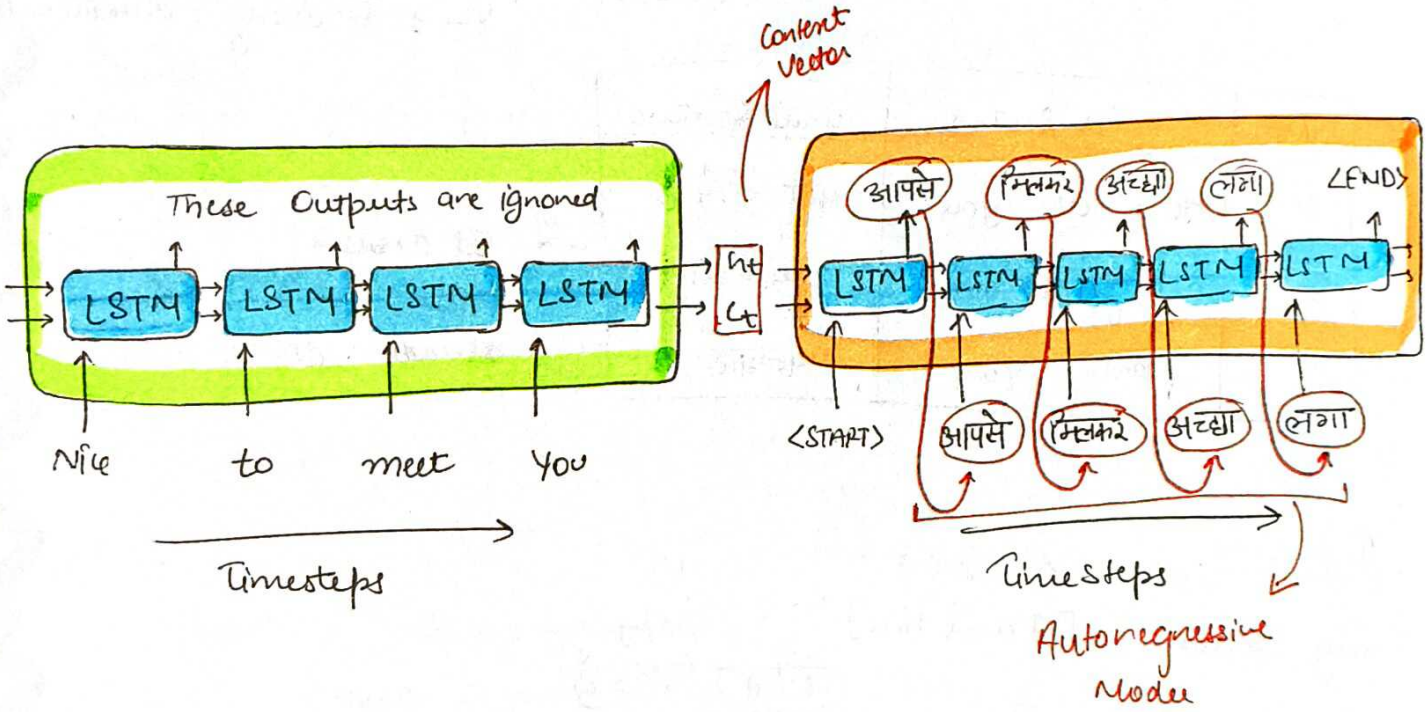


In the content of deep Learning, autoregressive models are a class of models that generate data points in a sequence by conditioning each new point on the previously generated points.



* Similar concept use in translate English to Hindi





* Why Transformer behave differently in Training time (Non-autoregressive) and Inference time (Autoregressive)

So! Because of Masked Self Attention.

Transformer as an Autoregressive Model

The Transformer decoder is autoregressive at inference time and non-autoregressive at training time.

Proove the statement

We can start with Transformer decoder → Inference → auto Regressive
→ Training → auto Regressive

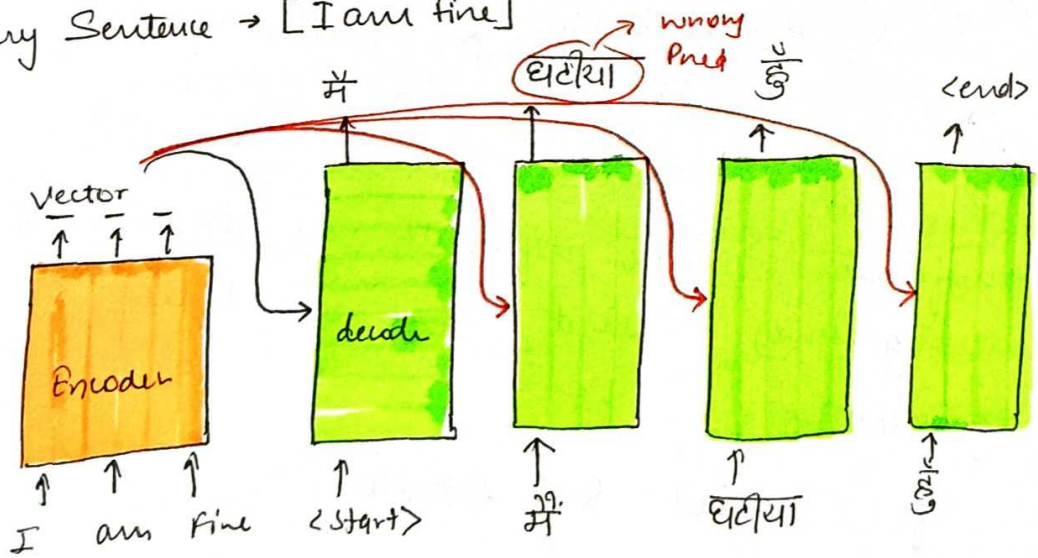
S.NO	English Sentence	Hindi Sentence
1.	How are you?	आप कैसे हैं
2.	Congratulation	बधाई हो
3.	Thank you	धन्यवाद

→ let assume Transformer Training already done.

Inference

मैं बढ़िया हुँ

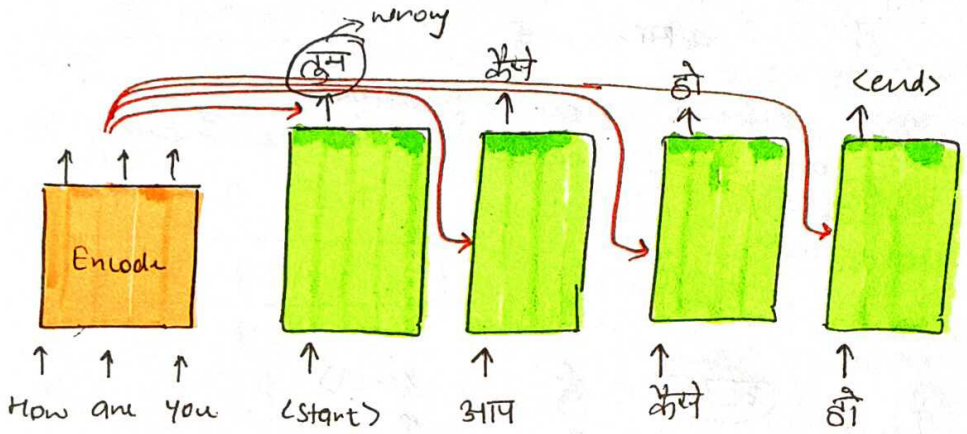
Query Sentence → [I am fine]



Let discuss Training

S.No	English Sentence	Hindi Sentence
1	How are you?	आप कैसे हैं
2	congratulation	बधाई हो
3	Thank you	धन्यवाद

training → auto regressor



Training → Autoreg

↳ Slow

Q Why Slow?

→ In decoder, Transform run 4 times → 3 words

what if there are 300 words then it will take more time.

if 1 lakh rows with 300 words then process will be more slow.

Q For Sequential data → Auto reg need?

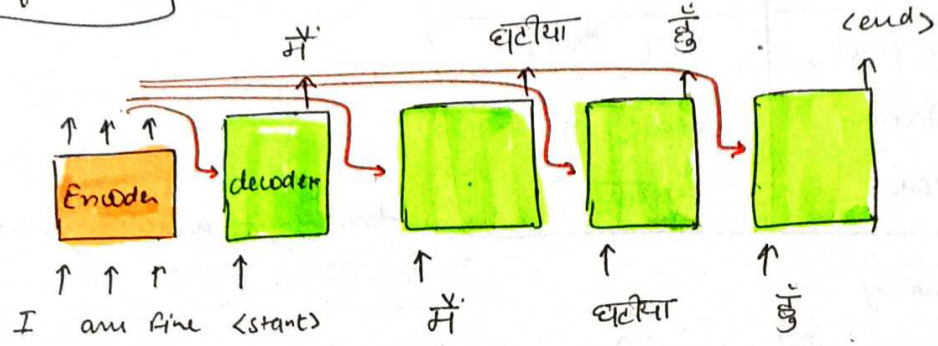
Ans → Yes and No

Yes ↳ Inference

No ↳ Training

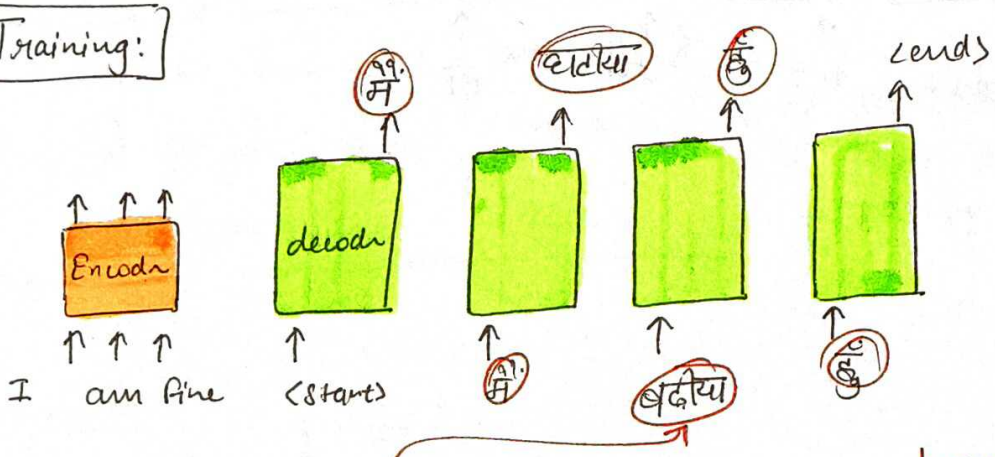
Sequential data need Auto reg → Yes

Inference:



Here, Next word are depend on previous one we can not write or send correct word as input.
So, Inference is Auto reg.

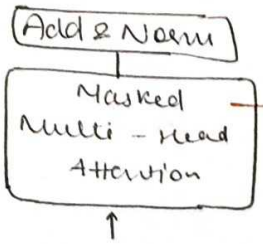
Training:



During Training, word doesn't depend on previous word. If word or previous word is wrong still next time correct word goes as input.
So, if we cannot use auto-reg still it is good. and we can perform task parallely.

Because of Parallel process

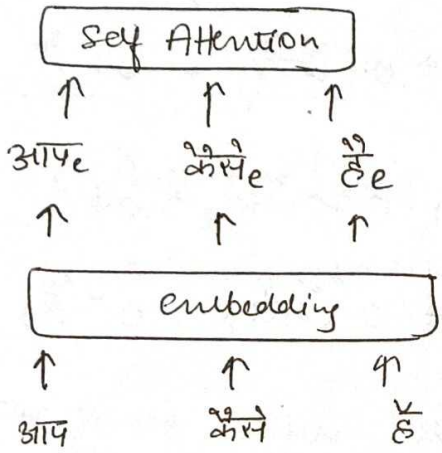
Training speed is very fast as compare to using auto-reg.



→ Let forget about Masked and focus on Multi-Head Attention.

Multi-Head Attention → Multiple Self Attention

for easy explanation → use Self Attention



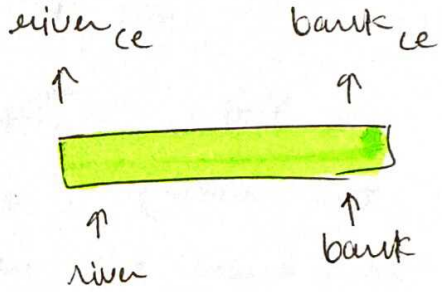
Self Attention Lecture

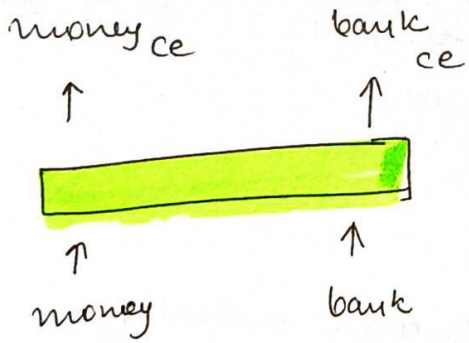
we discussed that when share or send word embedding into self Attention. Output is word contextual embedding.

eg:- river bank → content different

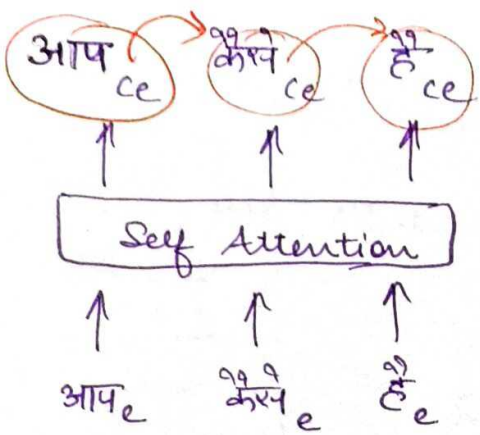
money bank →

bank → consider ki vo river ke sath use kr raha hai





bank → consider ki vo money ke sath use kr raha hai.



contextual embedding ~~distress~~ considered the particular word made with other contextual embedding.

Mathematical concept

$$\text{आप}_{ce} = 0.8\ \text{आप}_e + 0.1\ \text{कैसे}_e + 0.1\ \text{हैं}_e$$

$$\text{कैसे}_{ce} = 0.15\ \text{आप}_e + 0.75\ \text{कैसे}_e + 0.1\ \text{हैं}_e$$

$$\text{हैं}_{ce} = 0.1\ \text{आप}_e + 0.3\ \text{कैसे}_e + 0.7\ \text{हैं}_e$$

→ When we write आप . आप made with 80% of आप, 1% of कैसे (ye hame current situation pata nhi hai next word कैसे Hi Hai kya) and 1% of हैं (हैं also don't know)

→ Now, आप कैसे . कैसे made with 15% of आप (we know this word), 75% of कैसे and 1% of हैं (still we don't know this word)

→ Now. आप कैसे हैं . 1% of आप (We know this word), 3% of कैसे (know this word) and हैं with 70%.

Big Problem

* For finding current token value (contextual embedding) use future token value (embedding value)

* During Training, this is fine because we have output data too.

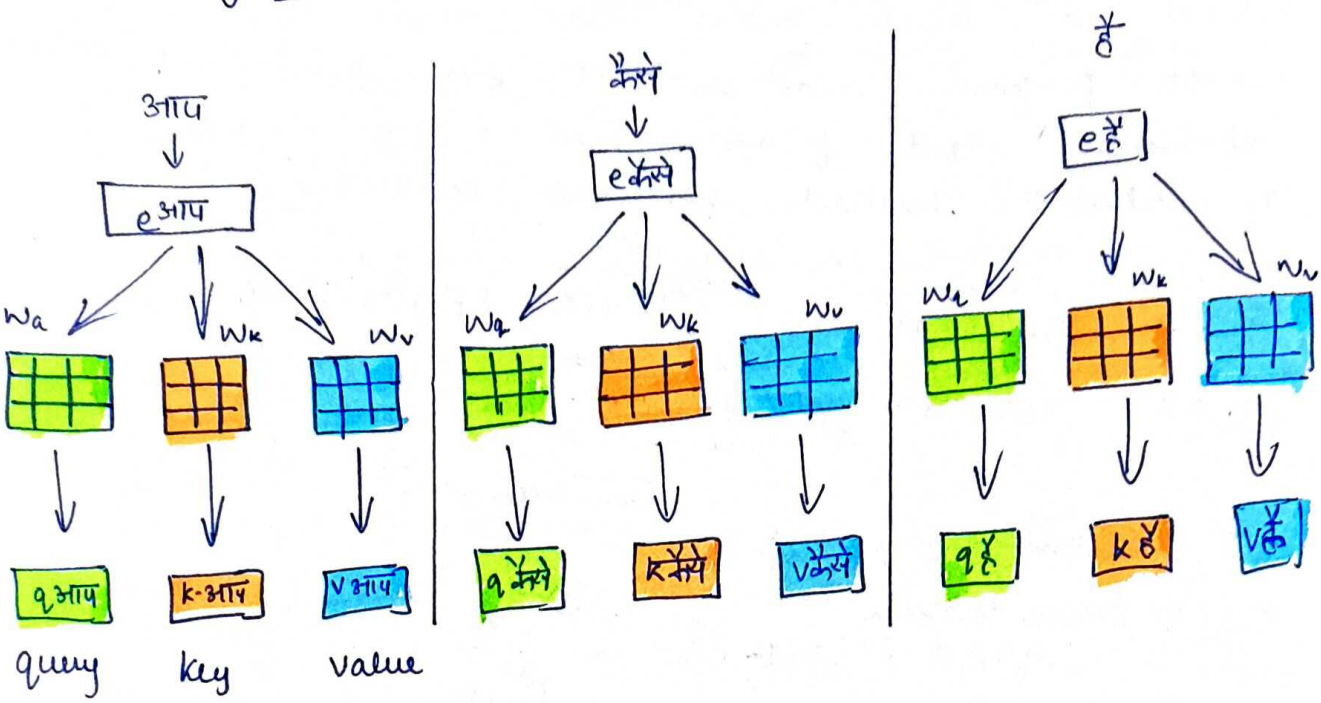
But it create trouble in Inference or Prediction.
Because we don't know next word.
Cannot perform mathematical expression. Mathematical expression need future word embedding value to calculate current contextual embedding.

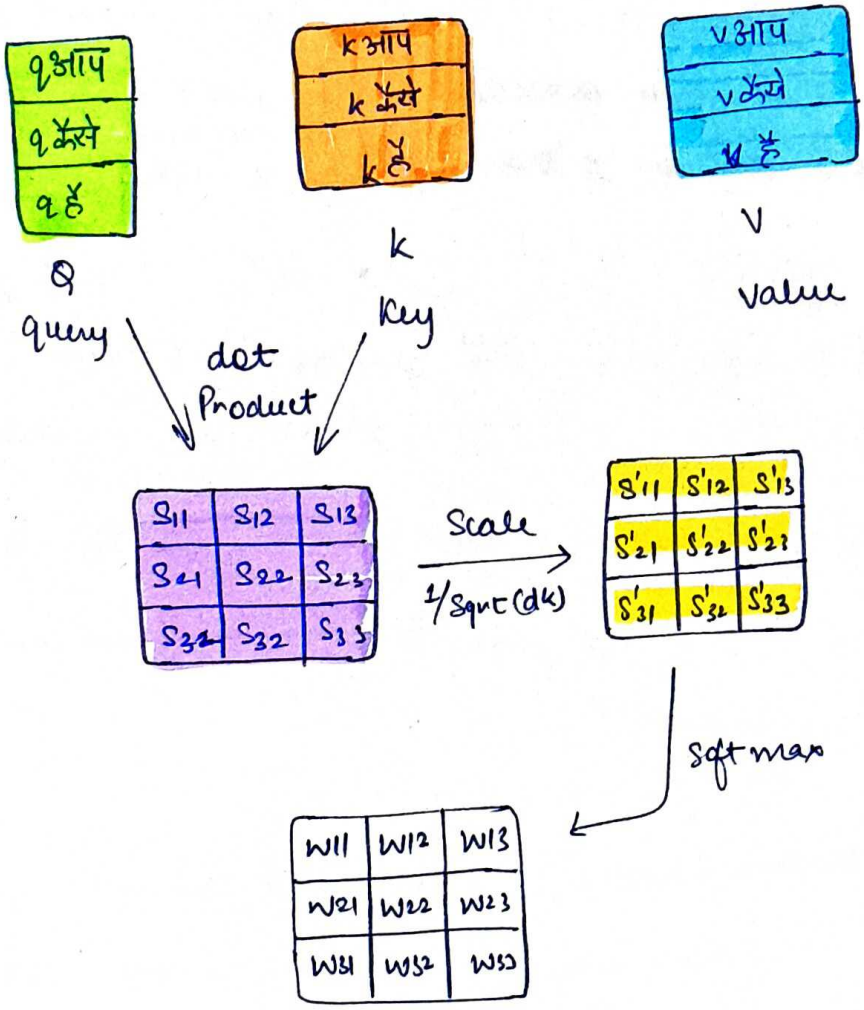
* This all scenario is **DATA LEAKAGE**

Auto-Reg	Non-Auto-Reg
→ No Data leakage	→ Data Leakage (in Inference)
→ Slow	→ Fast

* Any solution of this problem?
Yes → Self Attention is How part.

Finding the answer





आप$_{ce}$ = W_{11} * v आप + W_{12} * v कैसे + W_{13} * v ?

कैसे$_{ce}$ = W_{21} * v आप + W_{22} * v कैसे + W_{23} * v ?

है$_{ce}$ = W_{31} * v आप + W_{32} * v कैसे + W_{33} * v ?

→ We don't want W_{12} * v कैसे and W_{13} * v ? to finding आप$_{ce}$. Because आप$_{ce}$ time we don't know next word.

→ कैसे$_{ce}$ → don't want W_{23} * v ? this because at that we don't know next word.

So, convert $W_{12} = W_{13} = W_{23} = 0$

* Using Mask to convert into 0.