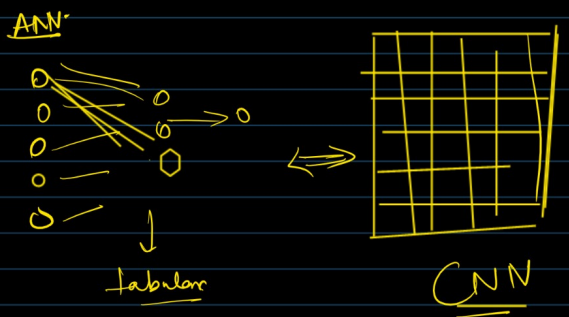
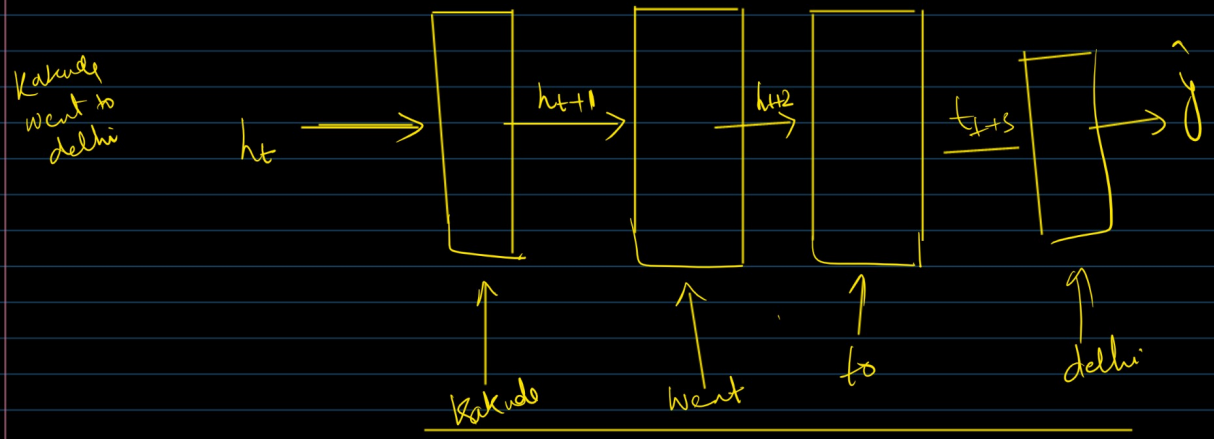
RNN.

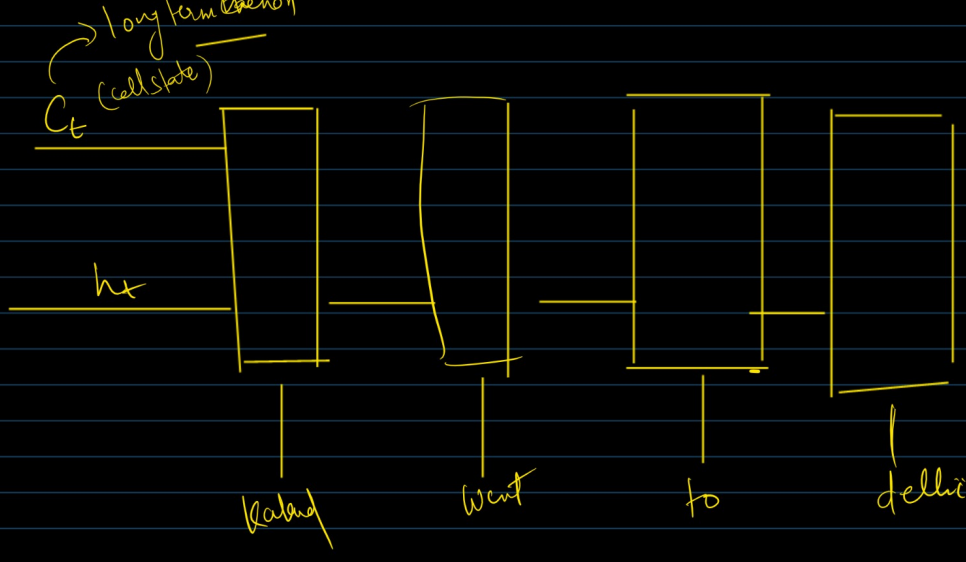


"Kakuda went to Delhi. He met Monjo. Monjo loves ice cream. Kakuda loves ice cream both love each other."

RNN



LSTM

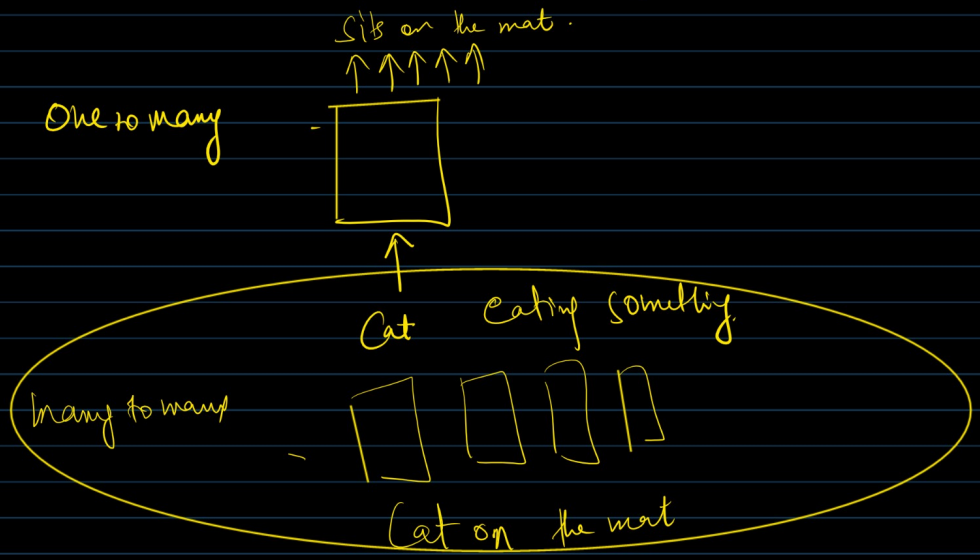


① Forget gate
② Update gate
③ Output gate

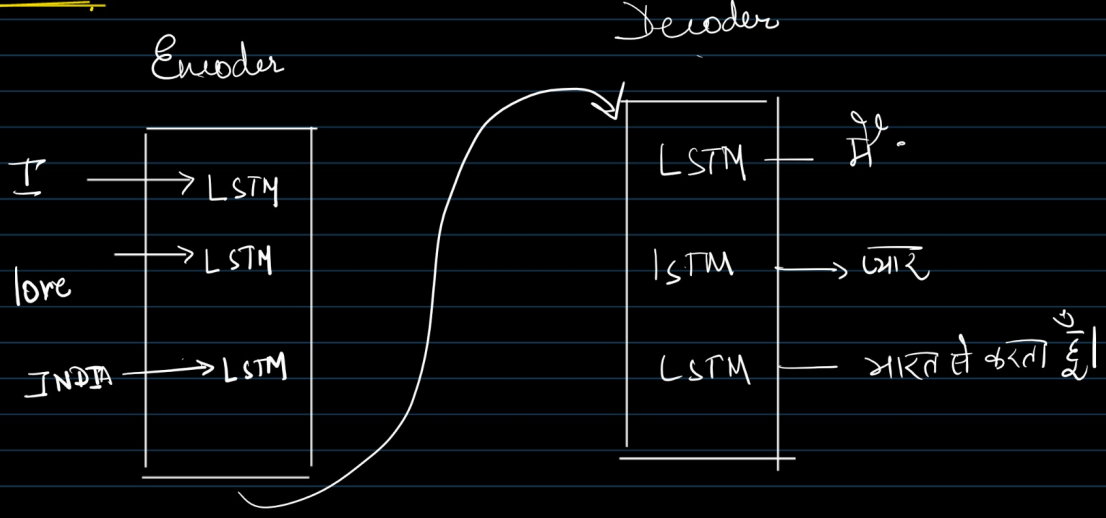
GRU → u_t / z_t → Update gate

→ Reset gate — r_t

fewer params. / easy to train



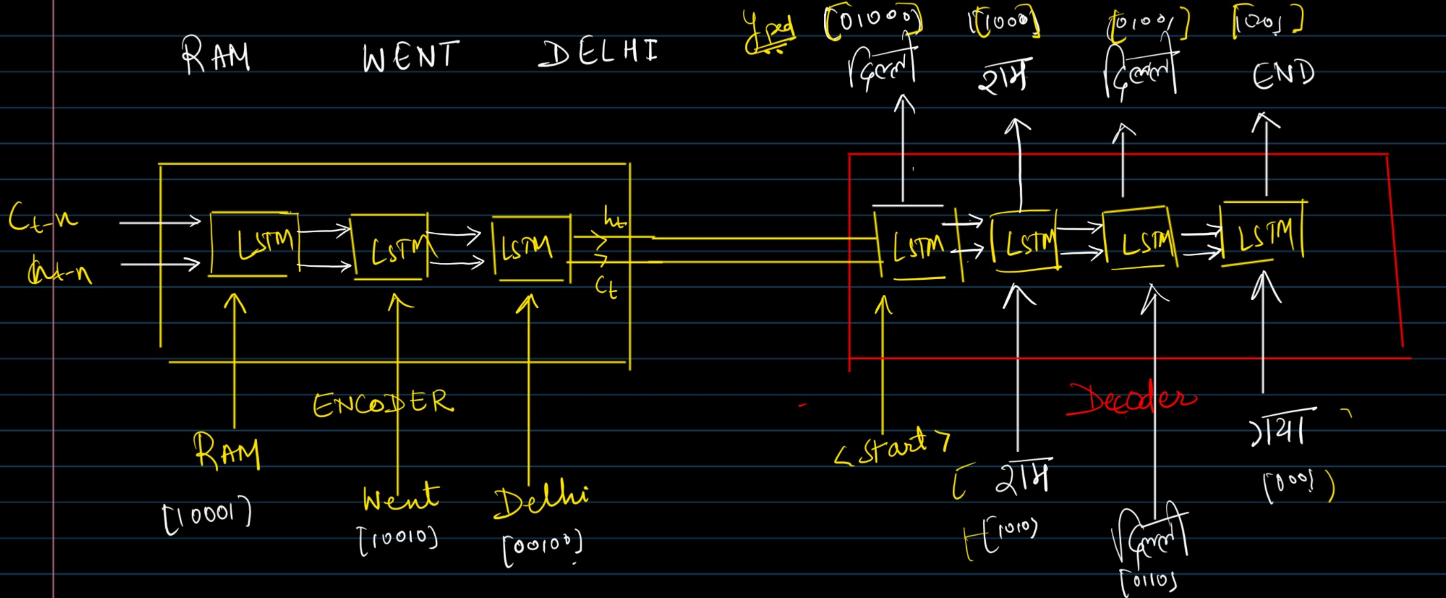
Seq 2 seq model



sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT-14 dataset, the translations produced by the LSTM

→ Ram went delhi → राम दिल्ली गया

→ Shyam came home → श्याम घर आया



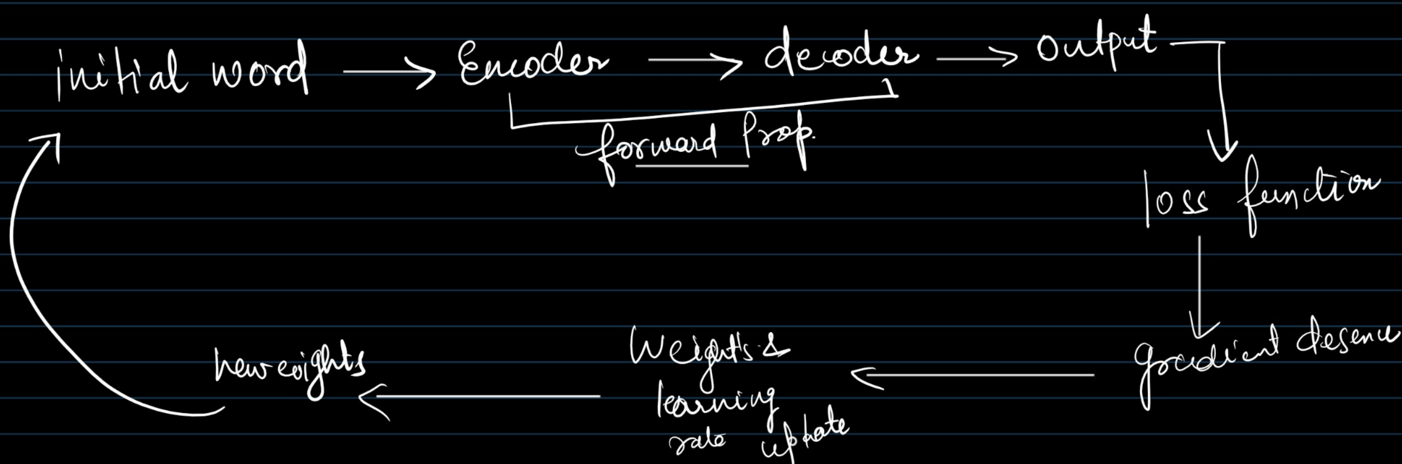
→ Shyam Came home

there is difference

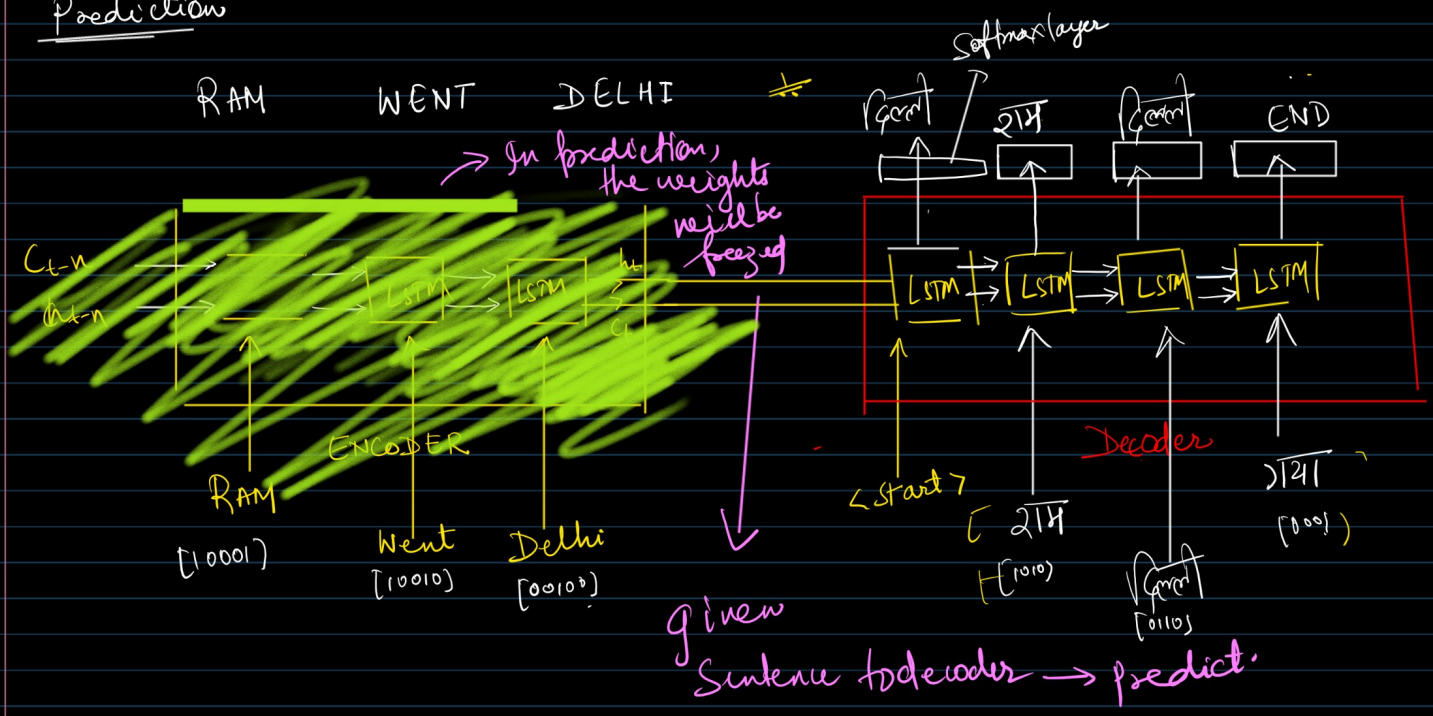
Actual → राम दिल्ली गया
pred → दिल्ली राम दिल्ली

Loss function → Categorical cross entropy

$$-\sum y_i \log y_{i\,pred}$$

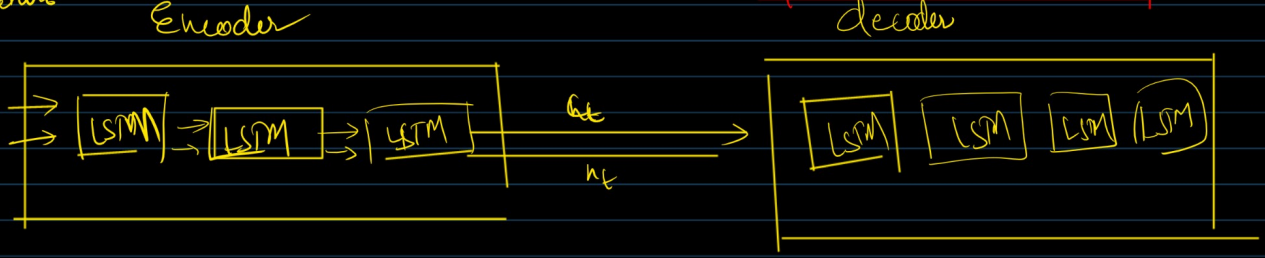


Prediction



given Sentence to decoder → predict.

Architecture

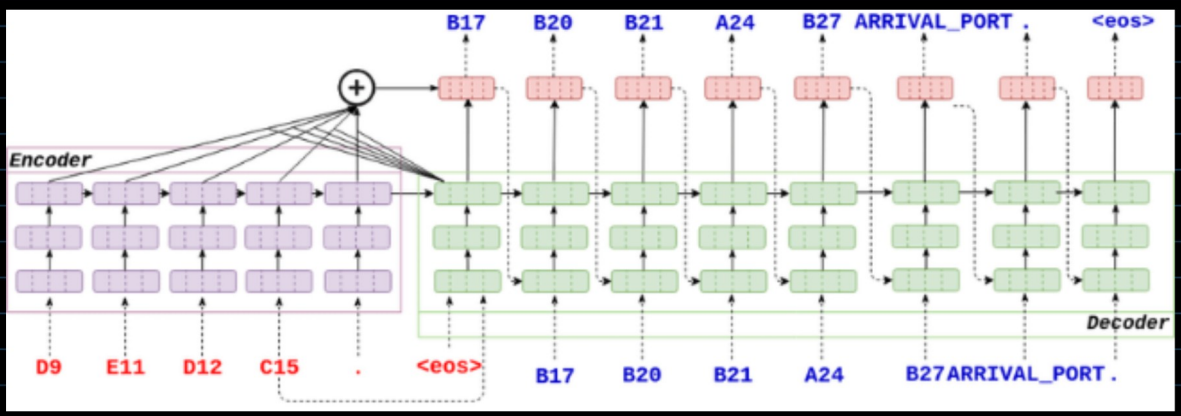


Embeddings

Ram went Delhi

Embedding

<start> आई फ्रेंच गया <end>



Use Case

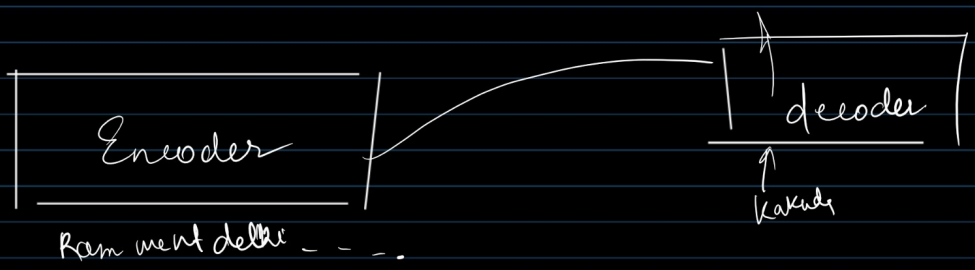
→ Text summarization
→ Captioning
→ QnA

Disadvantage of Seq2Seq models

* 30-50 words → performs well → the performance starts decreasing
* Doesn't understand long term context.

Test Case

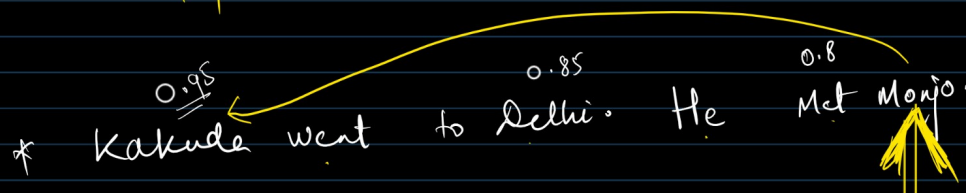
"Kakuda went to Delhi. He met Monjo. Monjo loves icecream. He loves ice cream both love each other." Both got married, After 5 years they had a child, the child had a child after 30 years".



* At each state of decoder during time we are accessing all the information.

⇓

Weight that we are seeing is distributed and previously occuring words might become less important.



* Since we are human being, Our mind understands this that Kakuda is very very important.

⇓

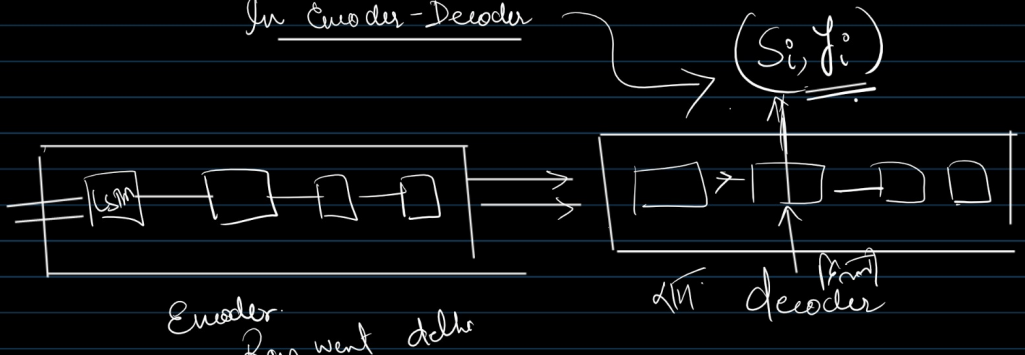
Soln ⇒ Context word Understanding

⇓

By giving more Attention to important keywords.

Attention Model

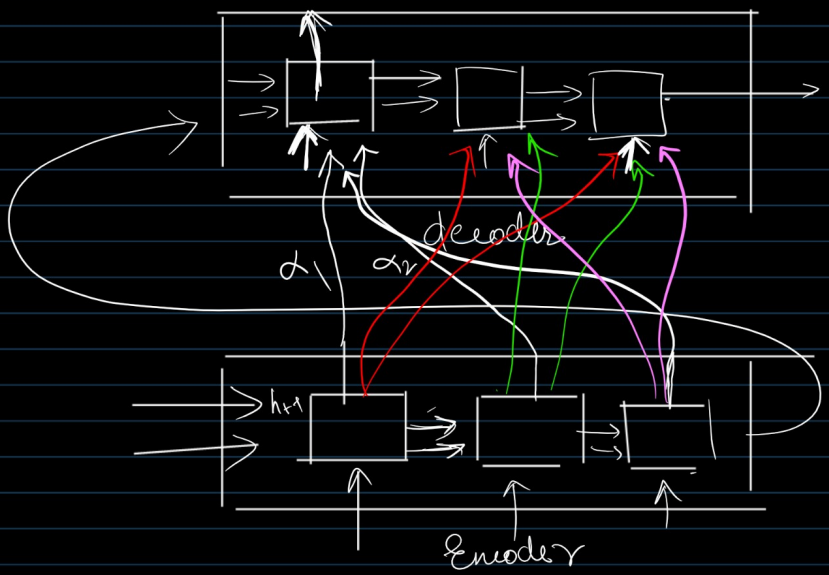
In Encoder-Decoder



Attention Mechanism (S_i, y_i, C_i)

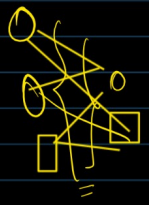
→ attention input

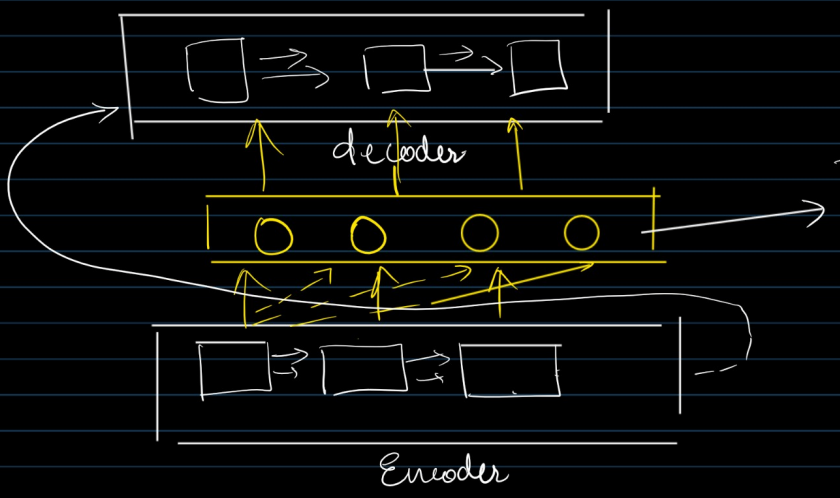
→ this is a vectors with each word's importance wrt



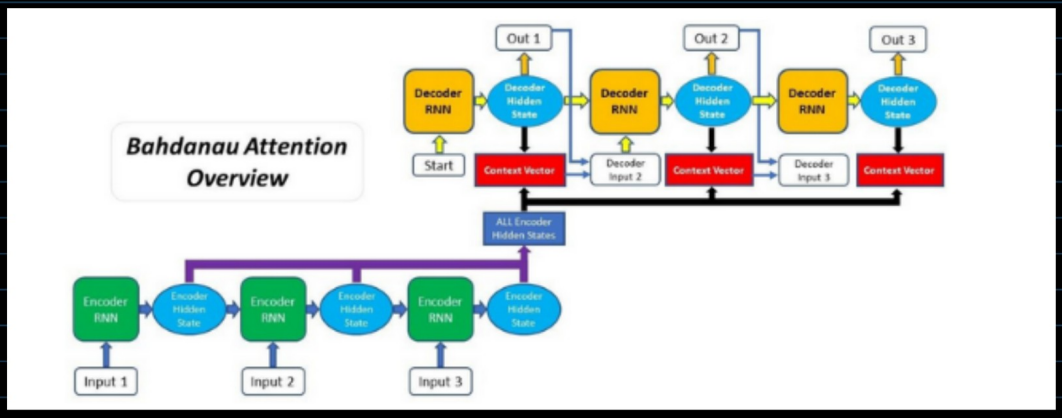
$C_i = \alpha_{ij} \times h_j$

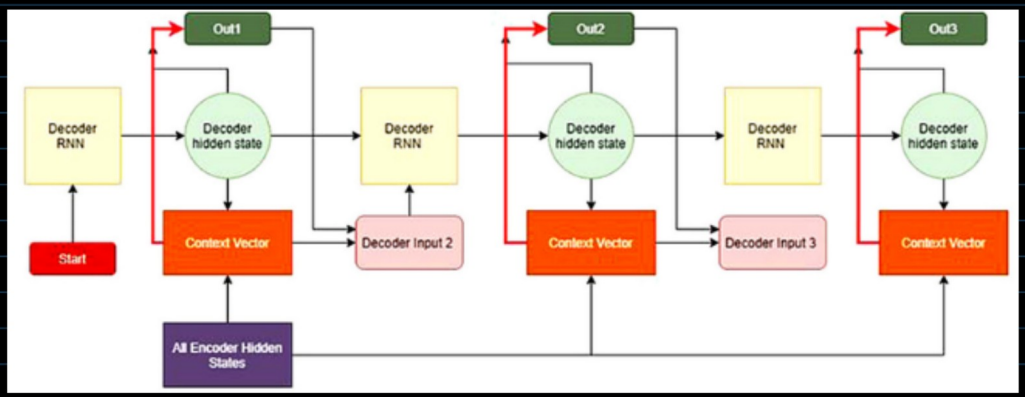
Similarity score / Assignment score





	राम	दिल्ली	गया
Ram	α_{11}	α_{12}	α_{13}
went	α_{21}	α_{22}	α_{23}
delhi	α_{31}	α_{32}	α_{33}





ANN – tabular data, CNN – Image

Sequ – RNN → LSTM → GRU

Seq2seq → Encoder-decoder → Attention mechanism

Summary

Encoder – RNN | LSTM | GRU

decoder – RNN

Attention – extra layer → It allowed decoder to look at all encoder hidden states.

↓

disadvantage

→ Slow training. → More parameter.

→ Limited Scalability

2017 → Attention is All you need

↓

2018 → transfer learn in NLP.

GPT → Generative Pre-training

transfer learning

CV → famous in CV.

(car) → RNN-101 → pretrained model

→ data train → prediction

→ truck

2018 → transfer learning was introduced.

\+

2017 → Transformer

→ GPT – 2018 → Dec.

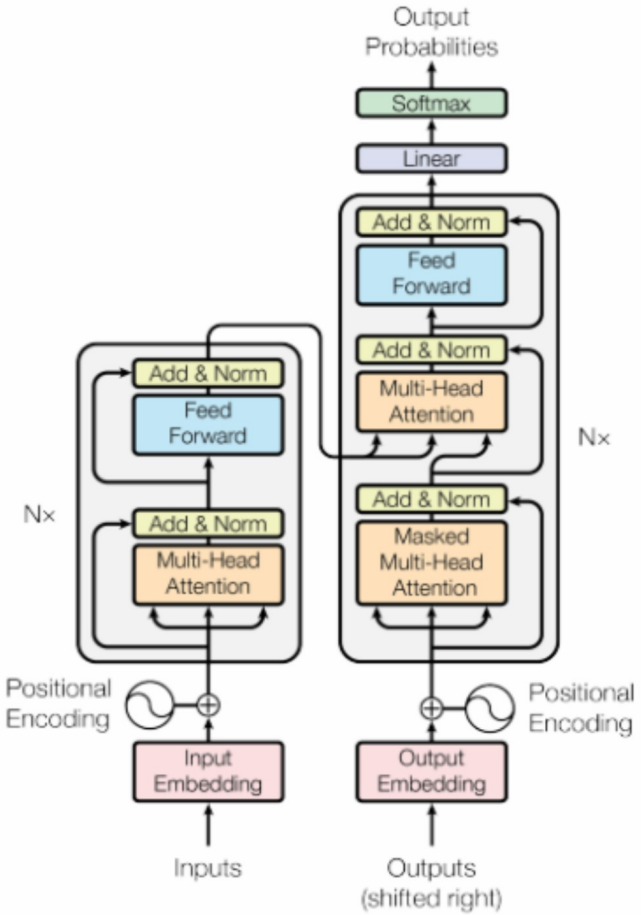


Figure 1: The Transformer - model architecture.

Why Transformers

→ RNN/LSTM took lot of training time & not scalable

Changes in transformers

→ removed RNN/LSTM entirely

→ Kept Self attention as the core Computation

→ parallel computation

→ fast (GPU friendly)

Static Embedding → I went to Bank
[1000] [0001] [1000] [0001]

dynamic Embedding

I went to Bank. The river Bank was clean
[0001] [0001]

So we need contextual Embedding.

Self Attention

Ram went Bank. Bank river was clean.

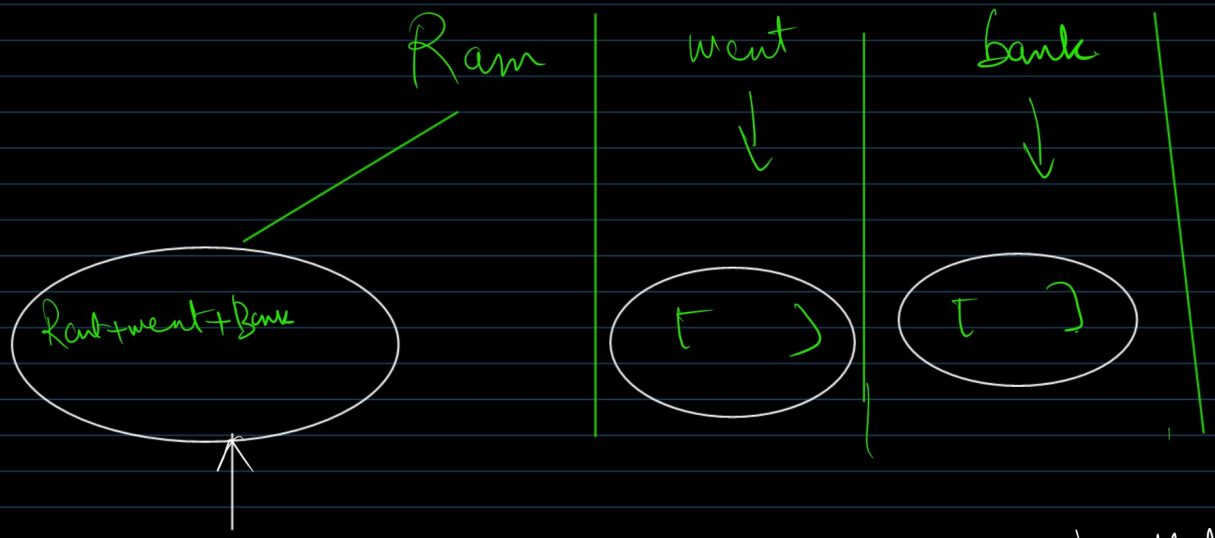
bank = 0.5 × Ram + 0.2 × went + 0.3 × bank

[0.3 0.5 0.2 0.1]

bank = 0.2 × Bank + 0.5 × River + 0.2 × was × 0.1 clean

[0.6 0.5 0.4 0

Both embeddings will be different.



→ operation is parallel.
So training it will be very cost but on a cost of losing sequence.

* Contextual embeddings → became very powerful.
+
positional encoder (add order)

Q → Query (what I am)

K → Key (what I contain)

V → value (what information I pass)

looking for

input → weight Q

$Q \rightarrow X W_Q$

$X \rightarrow$

$K = X W_K$

$V = X W_V$

$$\text{Attention Score} = \frac{QK^T}{\sqrt{d_k}}$$

$\sqrt{d_k}$ → gradient stability

Multi head mechanism

$W_Q\ W_K\ W_V \longrightarrow$ learns different relation.

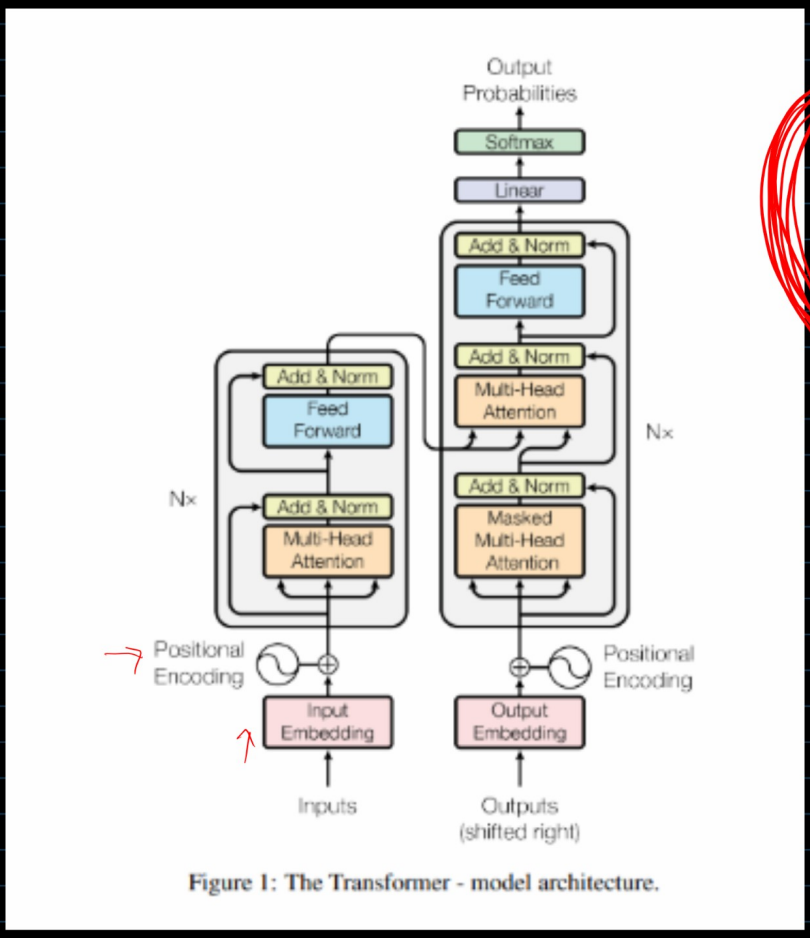


Figure 1: The Transformer - model architecture.

Encoder

① Input Embedding + positional Embeddings

② Multi-head Self attention → Residual + normalization

③ FFN (dense layer)

↓

(Residual)

↓

(Yact - Yp) (cost function)

↓

Contextual Embedding

② Decoder

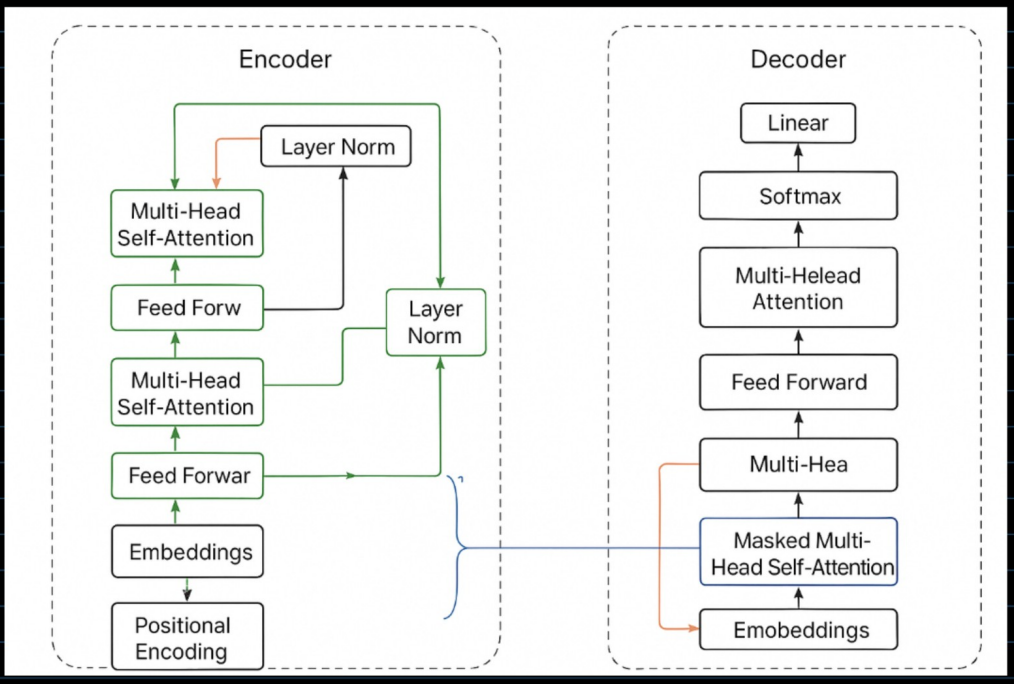
→ Output Embedding + positional Embedding.

→ Multi head Attention

→ Cross attention

→ FFN (Dense layer)

→ output.



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

Bert → Bidirectional Encoder Representation from Transformer

Before Transformer → Word2Vec (static)

Transformer → It processes the complete data in one go.

→ Contextual Embedding

→ Processed from left to Right (GPT) and right to left

Bert

↳ bidirection context problem got solved

river bank money bank

→ only Encoder part of Transformer is Bert.

Bert → RoBERTa.
→ ALBERT
→ DISTILBERT
→ SpanBERT
→ SciBert ⟶ Scientific text

ViT → Vision transformer

CV → YoloVII or MaskRCNN

⇓

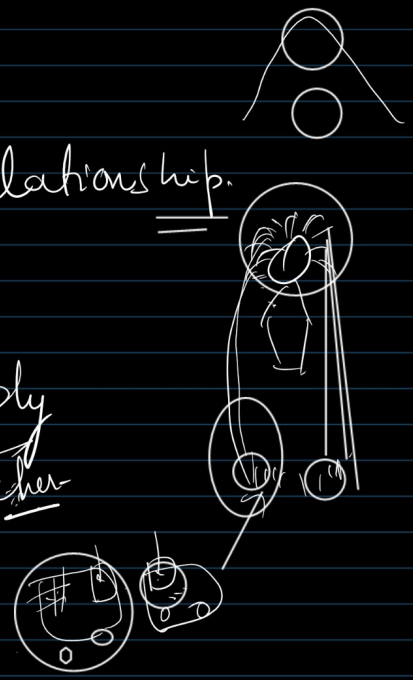
CNN

↓

Struggles with global relationship.

⇓

Transformer in NLP → apply on image patches



Idea

ViT → Vision transformer → treats an image as a sequence of patches, just like a sentence is a sequence of words

Analogy

- Sentence → Words → Embedding → Transformer.
- Image → Patches → Embedding → Transformer.

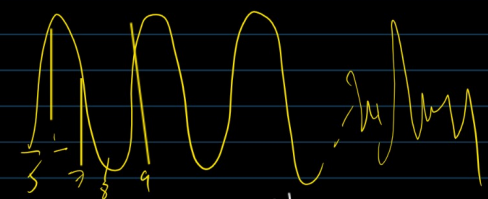
disadvantage

↳ large dataset for pre-training

DeiT

BeiT

Image (H×W×C)
↓ (split into patches)
patches.
↓
Embedding
↓
Transformer.



→ long seq of amplitude

Fourier Transform, Spectrogram

RNN, LSTM → Wav2vec