C35-NLP- Theory Session for

20/11/2022

Attention models

Encoder-Decoder → Attention Models → Transformers → Hugging Face

Conceptual Session

Python Implementation

Introduction:

Attention: most influential ideas in DL.

↳ NMT (Neural M/c Translation)

↳ Seq 2 Seq Models

↳ Encoder-Decoder architecture

English ⟶ Marathi

process i/p seq & encode/compress/ summarize info into CONTEXT/THOUGHT vector

It is initialized with context vector, using which it starts generating transformed o/p.

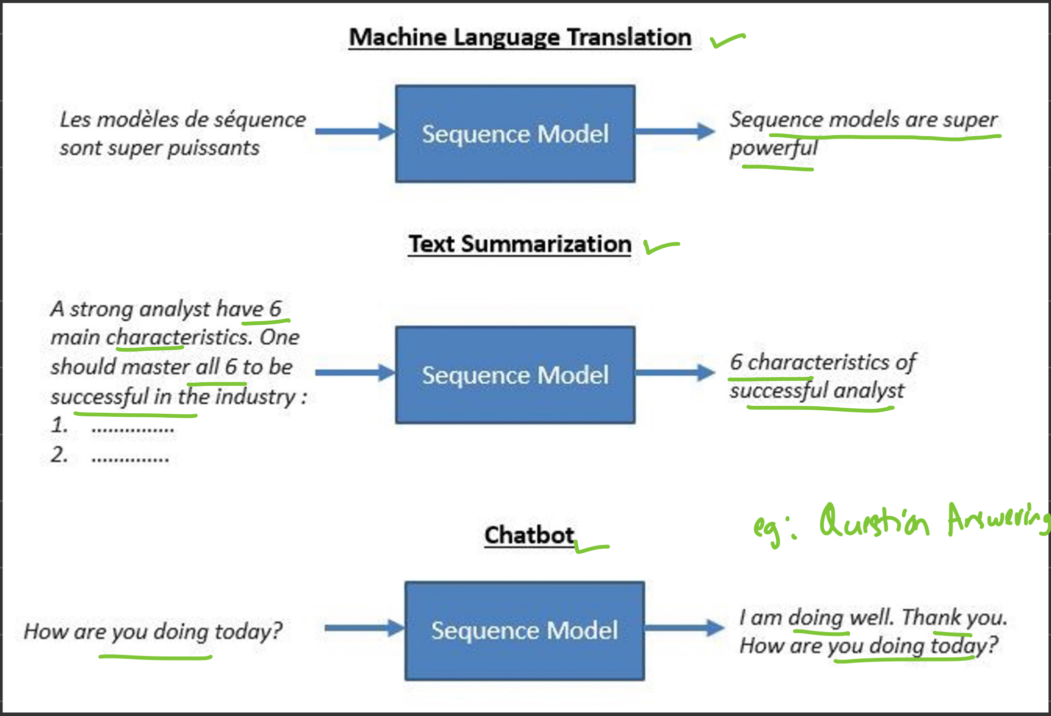
I/P English: Rahul is a good boy.

Target Marathi: राहुल चांगला मुलगा आहे.

RNN: LSTM | GRU → complex sequence → large amt of data.

Appl.: Speech recognition, NLP, Time series forecasting, etc.

Seq2Seq



Machine Language Translation ✓

Les modèles de séquence sont super puissants → Sequence Model → *Sequence models are super powerful*

Text Summarization ✓

A strong analyst have 6 main characteristics. One should master all 6 to be successful in the industry :

1. ....

2. ....

→ Sequence Model → *6 characteristics of successful analyst*

Chatbot ✓

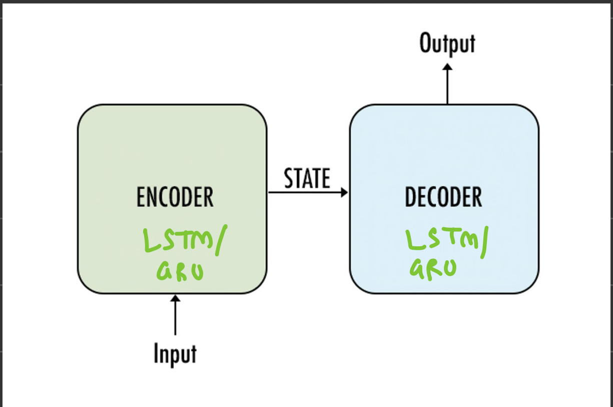
eg: Question Answering

How are you doing today? → Sequence Model → *I am doing well. Thank you. How are you doing today?*

Prerequisites:

1. Fundamental concepts in ML & NN
2. Linear Algebra & Prob.
3. Working knowledge of LSTM.

Encoder-Decoder Architecture:



Encoder: read i/p seq.
↓
summarize info as Internal state vectors (Hidden state & cell state).

o/p of encoder are discarded & we only preserve the Internal state.

Decoder: I/p is final states of encoder

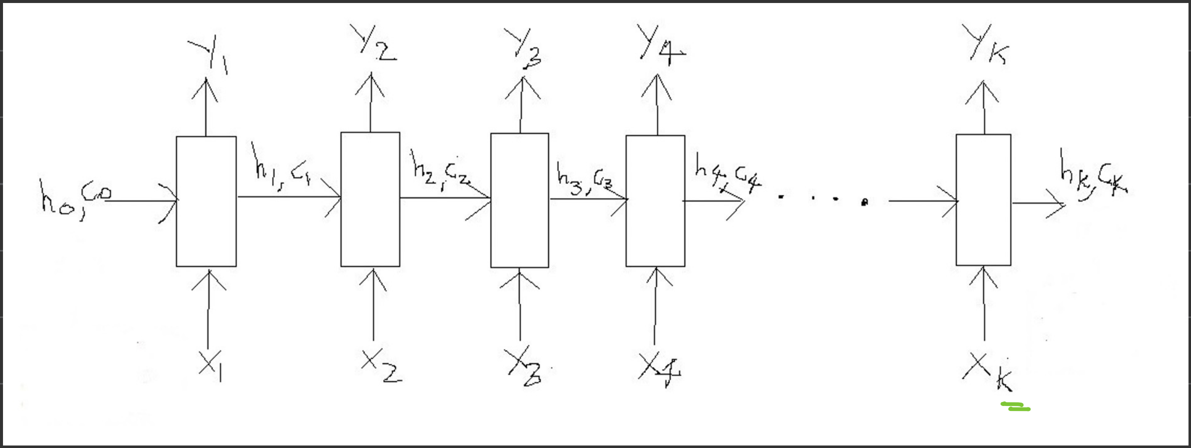
Using this, Decoder starts generating o/p seq.

Decoder Behaves differently in
- Training (uses Teacher-forcing)
- Inference (i/p to decoder at each time step is o/p from previous time step).

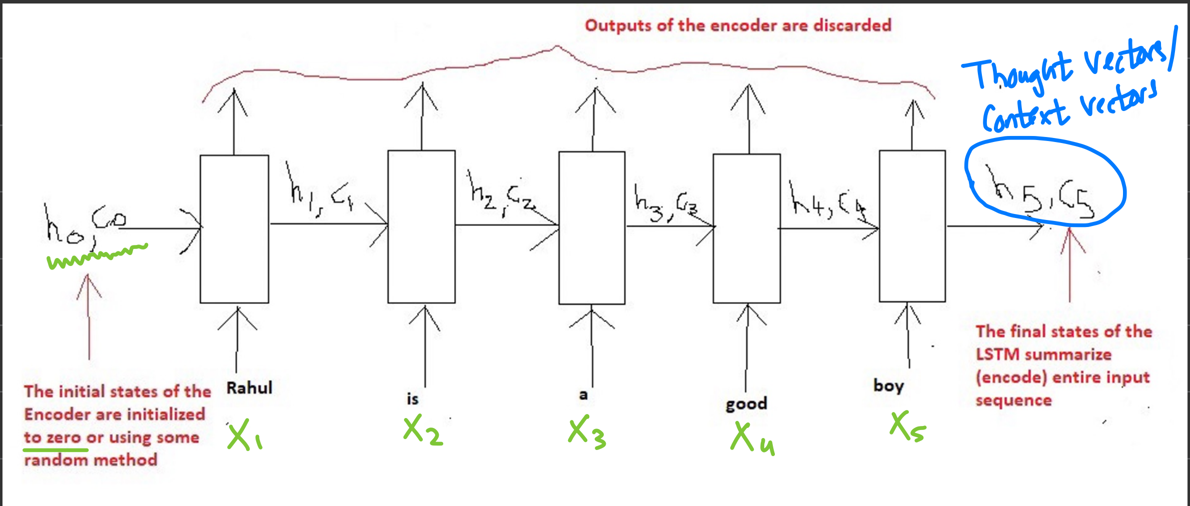
Encoder LSTM:

3 components:

1. X_i → i/p seq. at time step i
2. h_i & c_i → internal states
3. y_i → o/p seq. at time step i



Word Level NMT:


Outputs of the encoder are discarded

Thought vectors/ Context vectors

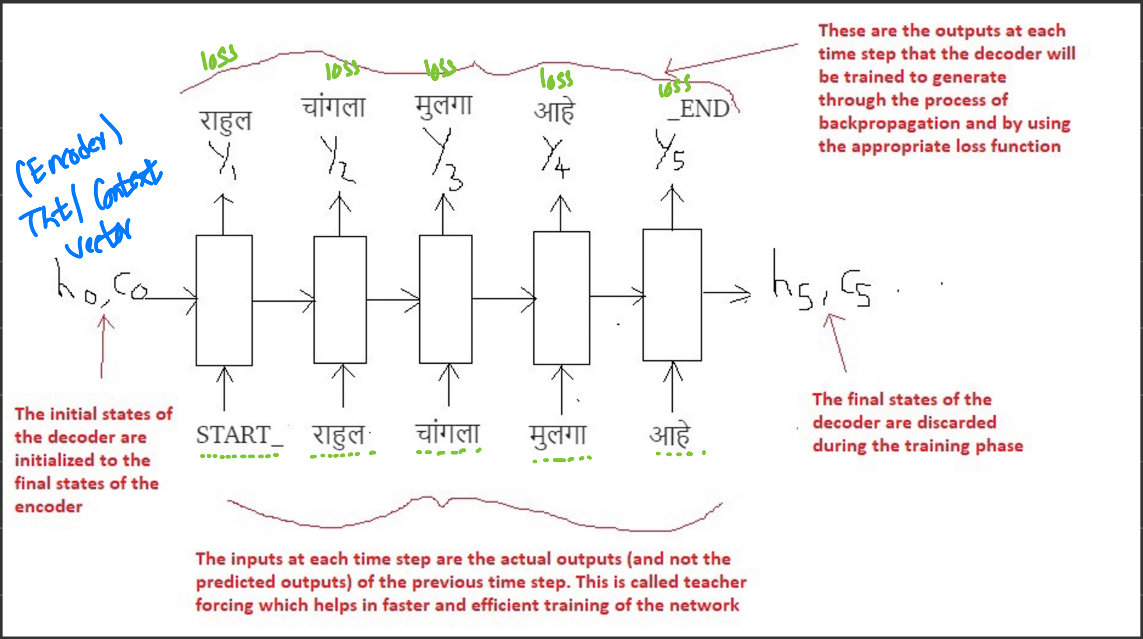
The initial states of the Encoder are initialized to zero or using some random method

The final states of the LSTM summarize (encode) entire input sequence

h_i & c_i: They remember what the LSTM has read (learned) till now

y_i: o/p (predictions) of LSTM at each time step)
y_i is prob. dist over entire vocabulary

Decoder LSTM _ Training Mode.



These are the outputs at each time step that the decoder will be trained to generate through the process of backpropagation and by using the appropriate loss function

The initial states of the decoder are initialized to the final states of the encoder

The final states of the decoder are discarded during the training phase

The inputs at each time step are the actual outputs (and not the predicted outputs) of the previous time step. This is called teacher forcing which helps in faster and efficient training of the network

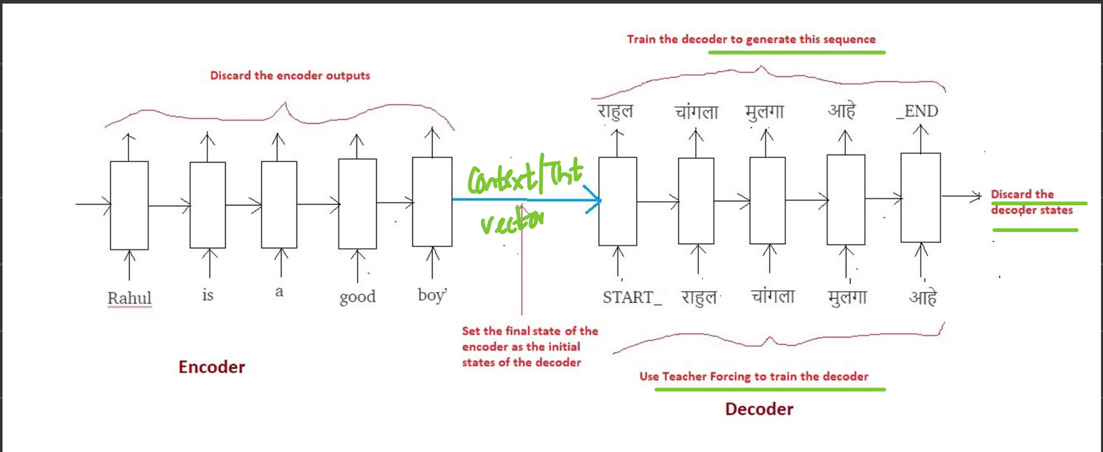
Goal: Train the (TEACH) decoder to o/p राहुल चांगला मुलगा आहे.

"START_ राहुल चांगला मुलगा आहे _END"

Teacher Forcing: Here i/p at each time step is given as actual o/p (& not the predicted output) from the previous time step

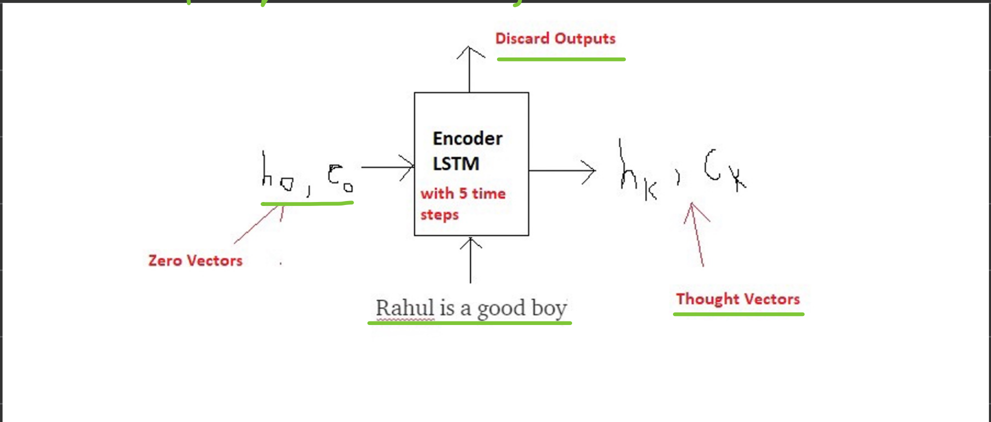
- helps to train more fast.

https://machinelearningmastery.com/teacher-forcing-for-recurrent-neural-networks/


Train the decoder to generate this sequence
Discard the encoder outputs
राहुल
चांगला
मुलगा
आहे
_END
Context/Tht vector
Discard the decoder states
Rahul
is
a
good
boy'
START_
राहुल
चांगला
मुलगा
आहे
Set the final state of the encoder as the initial states of the decoder
Encoder
Use Teacher Forcing to train the decoder
Decoder

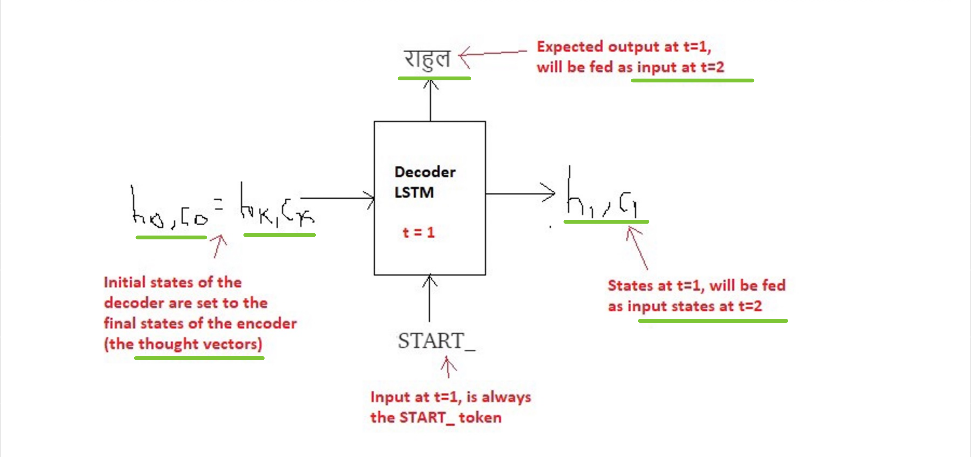
Decoder LSTm - Inference mode:

1. Encode i/p seq. into Thought vectors.



2. Start generating o/p seq. in a loop, word by word.

At $t=1$,



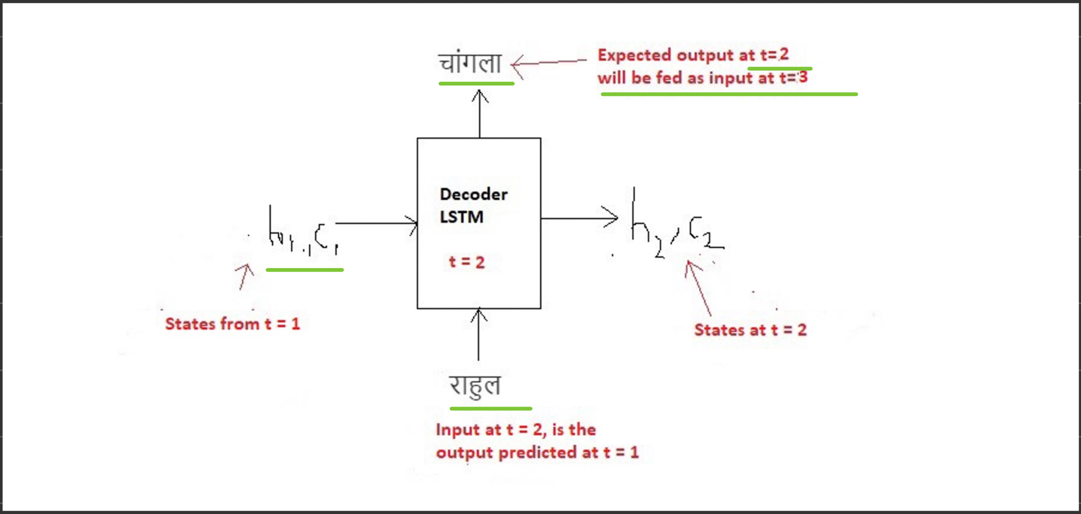
Expected output at t=1, will be fed as input at t=2

Initial states of the decoder are set to the final states of the encoder (the thought vectors)

States at t=1, will be fed as input states at t=2

Input at t=1, is always the START_ token

At t=2,



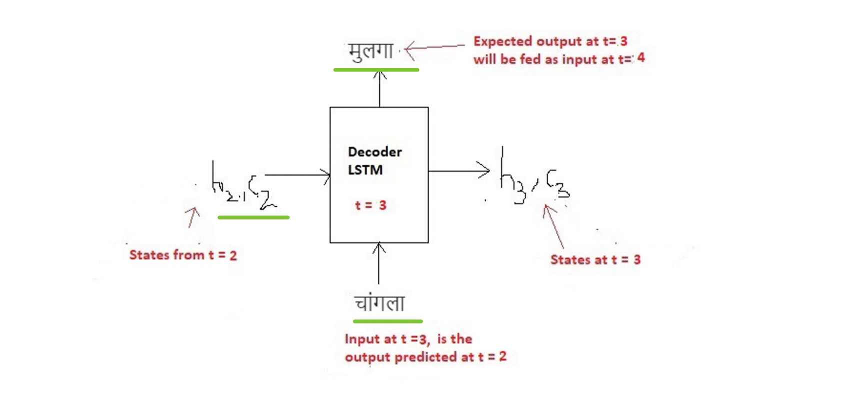
Expected output at t=2 will be fed as input at t=3

States from t = 1

States at t = 2

Input at t = 2, is the output predicted at t = 1

At t=3,



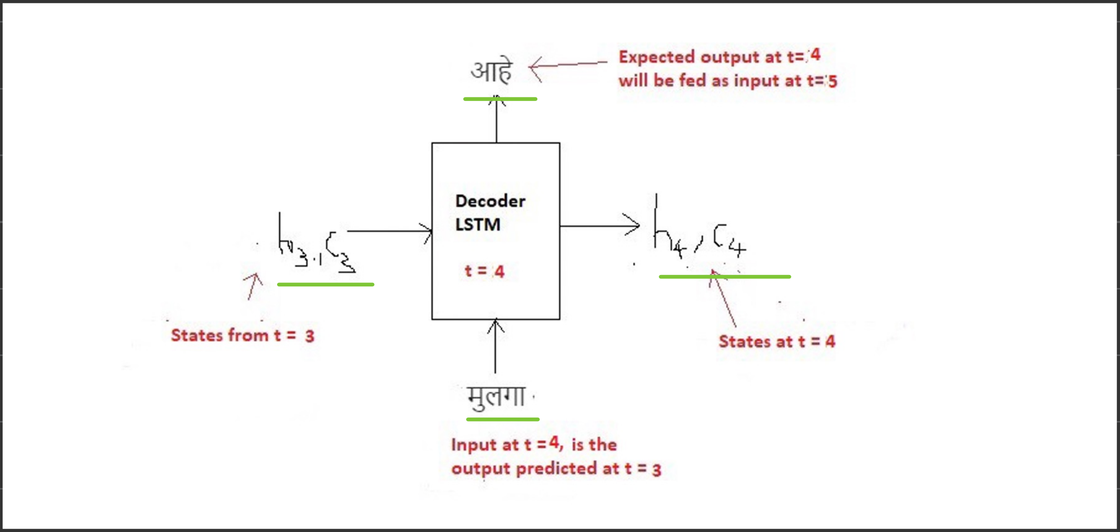
Expected output at t= 3 will be fed as input at t= 4

States from t = 2

States at t = 3

Input at t =3, is the output predicted at t = 2

At t=4,



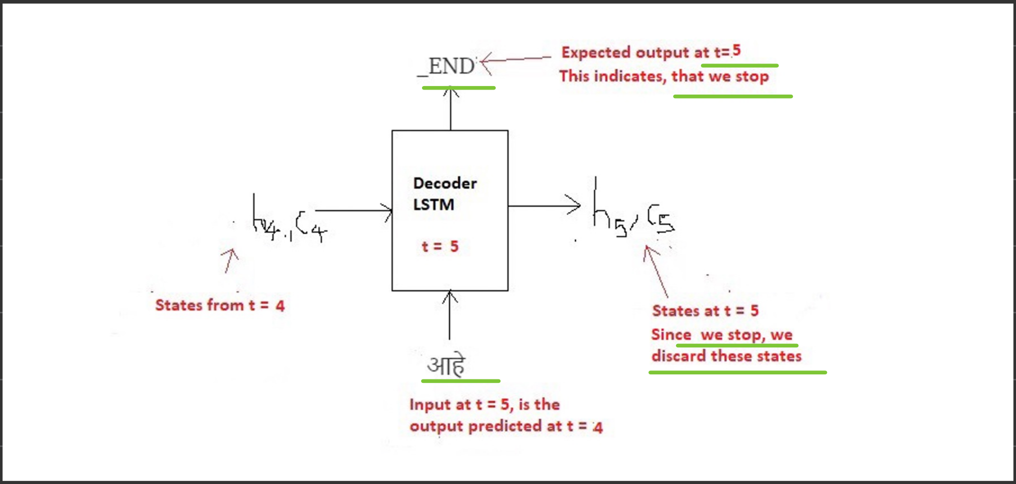
Expected output at t=4
will be fed as input at t=5

States from t = 3

States at t = 4

Input at t =4, is the
output predicted at t = 3

At t=5,



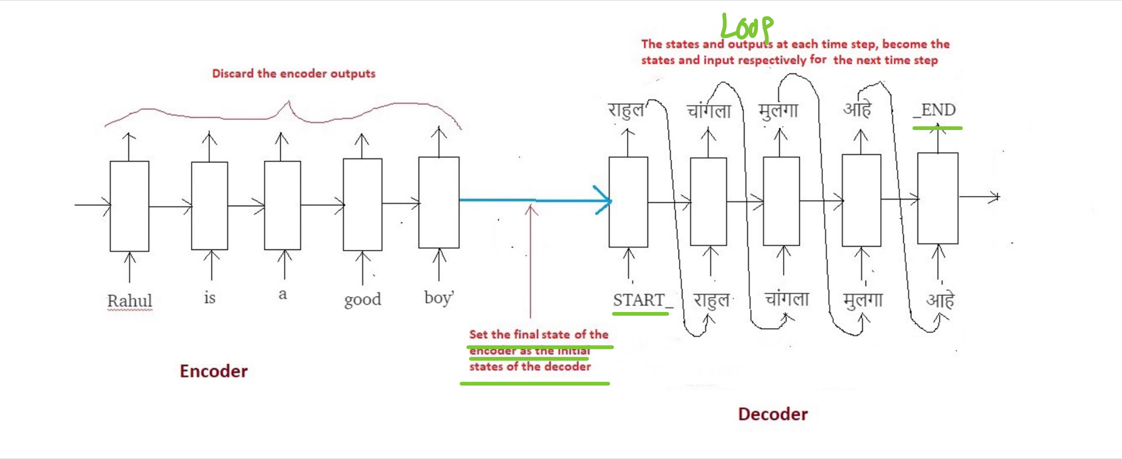
Expected output at t=5
This indicates, that we stop

States from t = 4

States at t = 5
Since we stop, we
discard these states

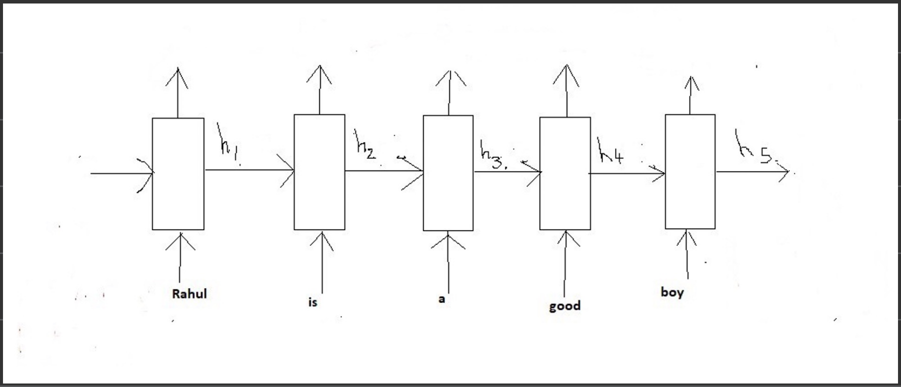
Input at t = 5, is the
output predicted at t = 4

Inference Algorithm:

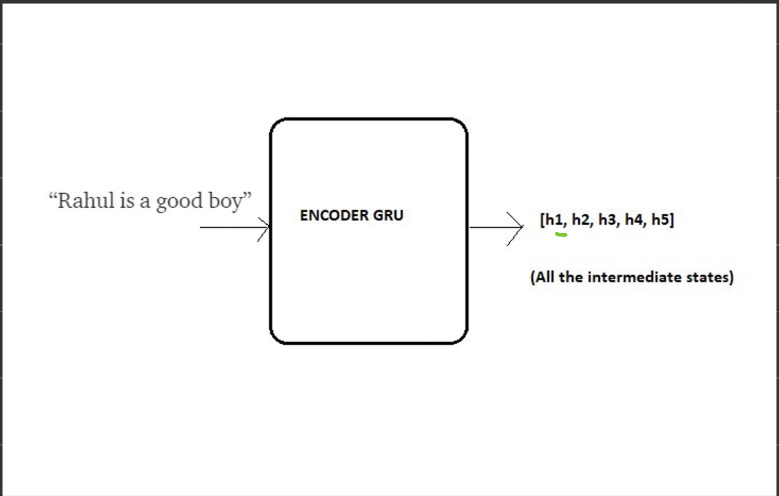


Loop
The states and outputs at each time step, become the states and input respectively for the next time step
Discard the encoder outputs
Rahul
is
a
good
boy'
Encoder
Set the final state of the encoder as the initial states of the decoder
राहुल
चांगला
मुलगा
आहे
_END
START_
राहुल
चांगला
मुलगा
आहे
Decoder

Why name Attention?

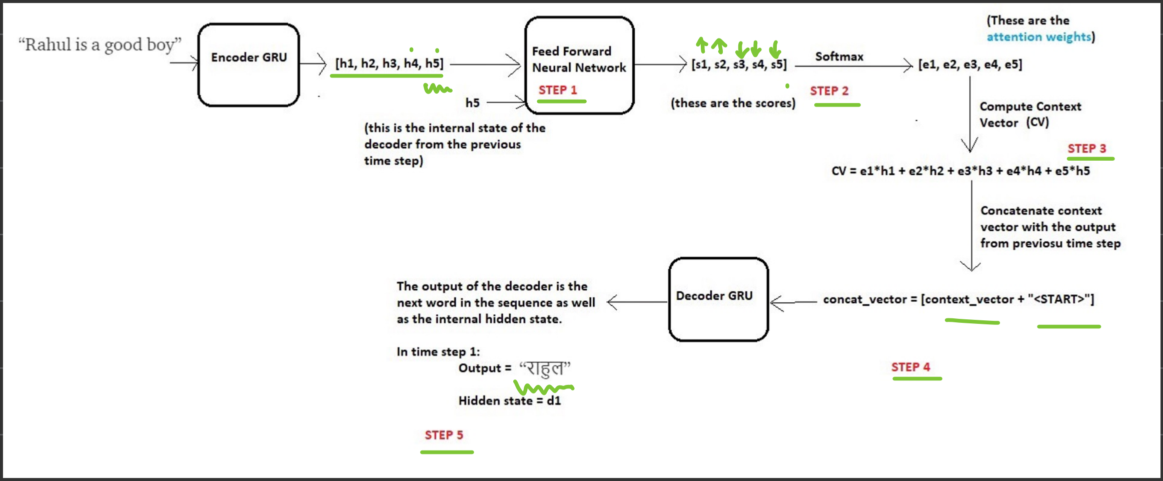


Encoder GRU:



How does Attention work?

Decoding at time step1.



(this is the internal state of the decoder from the previous time step)

CV = e1*h1 + e2*h2 + e3*h3 + e4*h4 + e5*h5

concat_vector = [context_vector + "<START>"]

The output of the decoder is the next word in the sequence as well as the internal hidden state.

In time step 1:
Output = "राहुल"
Hidden state = d1

Compute a score

"राहुल"

Rahul

Compute attention weights: Apply softmax e1,e2,e3, e4,e5
[0-1]

$e_1+e_2+e_3+e_4+e_5=1$

$e_1=0.75,\ e_2=0.20,\ e_3=0.02,\ e_4=0\ 2,\ e_5=0.01$

Compute Context vector $= e_1 \times h_1 + e_2 \times h_2 + e_3 \times h_3 + e_4 \times h_4 + e_5 \times h_5$

Concatenate context vector with o/p of prev. step