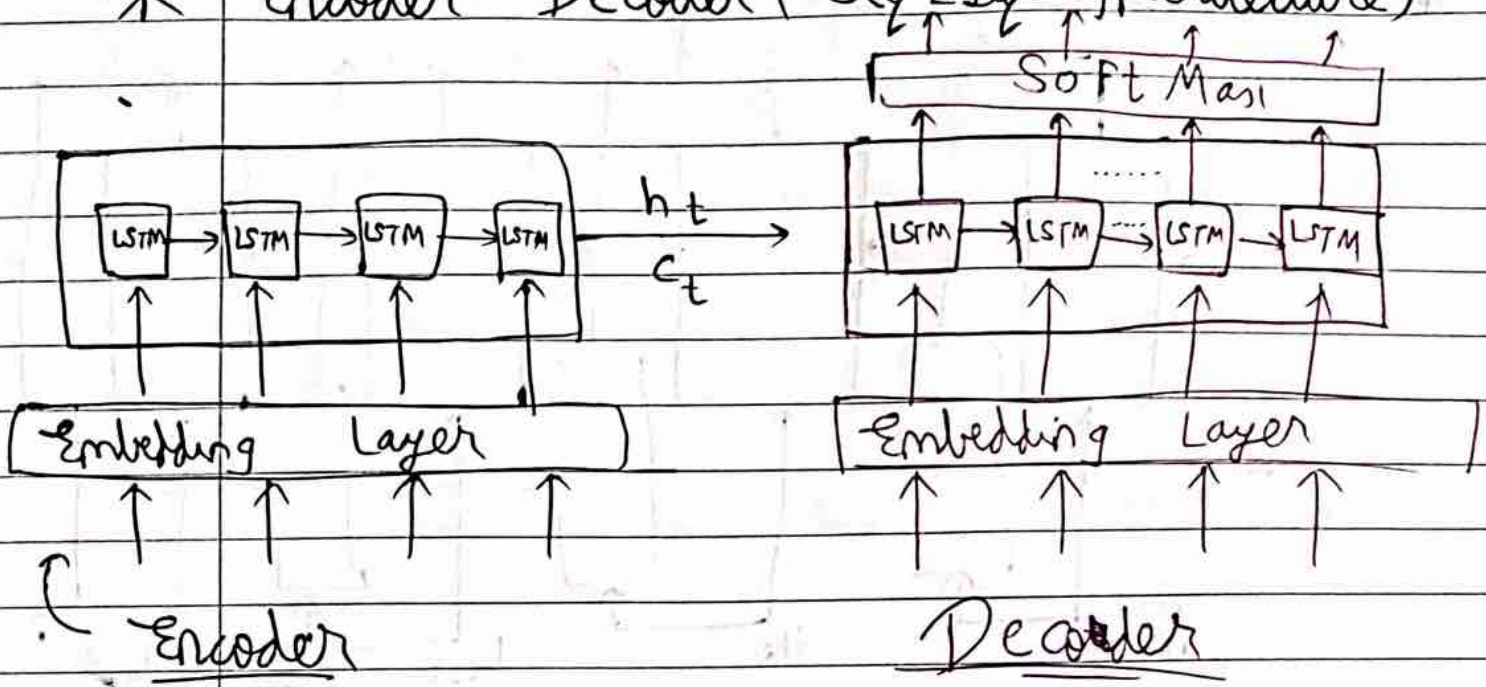
* Encoder - Decoder (Seq 2 seq Architecture)

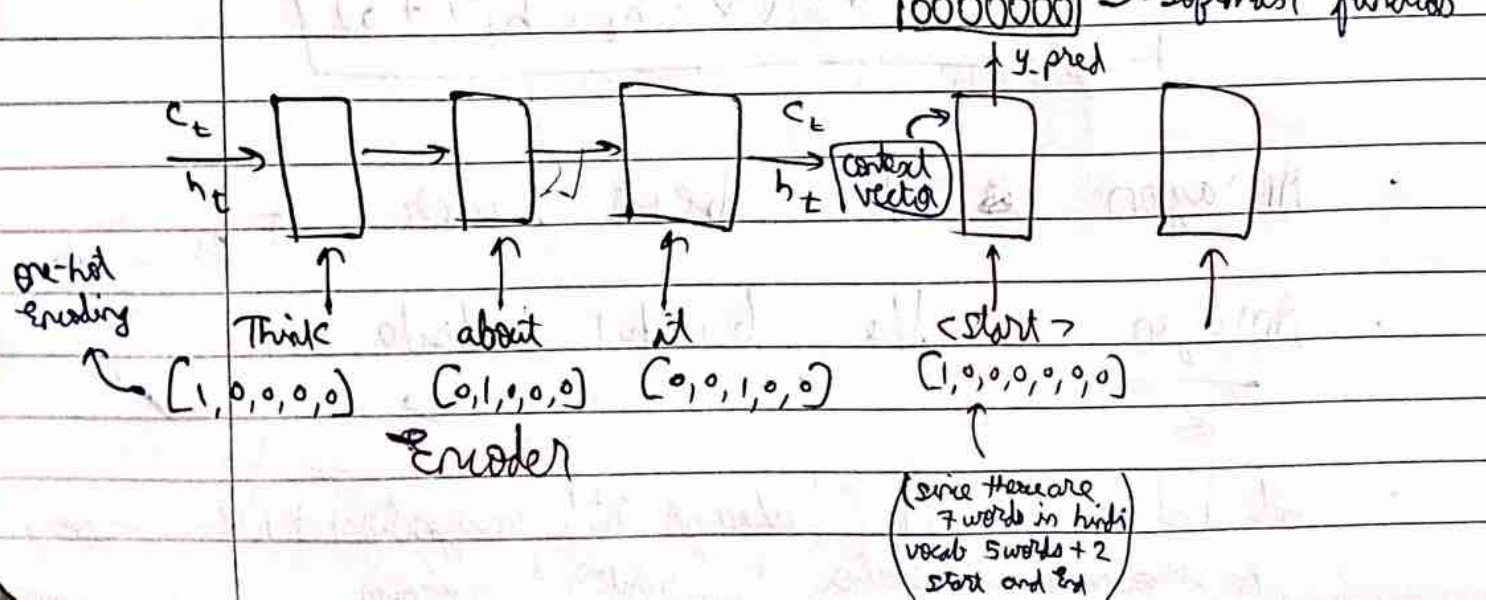


* Training the Architecture using Back propagation with an example.

1) English 2) 3)	Hindi 3)
1) (think about it)	2) (सोच लो)
2) (come in) 4) 5)	(अंदर आओ) 4) 5) 6)

1) <start>
7) <end>

row 1 - [Think about it → सोच लो]



highest → (मी) → word

__/__/____

Let's say y_pred = [0.2, 0.1, 0.3, 0.2, 0.1, 0.1, 0]

y_true = [0, 1, 0, 0, 0, 0, 0]

↳ सोच

Note :-

- Our output is 'मी' word instead of 'सोच' word
- In Decoder, the output of previous cell is given as input to its next cell at time for timestamp (t+1).
- But, while training process, even if the previous cell calculates it's output (correct or Incorrect), we would still send 'True_labelled value' to the current cell.
- Eg :- We got out as (मी) but correct word was (सोच), so we will given (सोच) as our new input to the next cell and not (मी) as our input. This is called **teacher forcing method**.
- This helps in faster training and fast convergence to the true value.

* **Calculating Loss and performing back propagation**

Eg :-

		Loss	
y_true	[0, 1, 0, 0, 0, 0, 0] सोच (padding to match output dim)	[0, 0, 1, 0, 0, 0, 0] (मी) (padding)	[0, 0, 0, 0, 0, 0, 1] <end> (padding)
y_pred	[0.2, 0.1, 0.3, 0.2, 0.1, 0] मी	[0.1, 0.1, 0.1, 0.15, 0.2, 0.4, 0.05] जा आ	[0.1, 0.1, 0.1, 0.2, 0.2, 0.3] <end>

- We would use Crossentropy Loss () function

$$L = -\sum_{i=1} y_i^{(true)} \log(y_i^{pred})$$

$$L_{t=1} = -1 \times \log 0.1 - 0 \times \log 0.2 - 0 \times \log 0.3 \ldots 0 \times \log 0$$

$$= -1 \times \log 0.1 = 1 \quad \therefore L_{t=1} = 1$$

__/__/____

$L_{t=2} = -1 \times \log 0.1 = 1$

$L_{t=3} = -1 \times \log 0.3 = 0.31$ (When output is correct Loss ↓)

∴ Total Loss = 2.31

∴ average Loss = 0.75

- Now we will calculate gradients based on Loss function and update weights

※.

(Initial wts) → forward prop → output → Loss

↓

new weights ← learning rate ← weight update (Adam) ← gradient

※. Note:-

1) While making prediction we will provide input to current cell in "decoder" as output of previous cell of decoder unlike in training phase..... (which is obvious).

* Improvements

1) Embedding :- We won't use one-hot encoding, instead use embedding layer which is a low dimensional dense vector layer, using pretrained (word2vec / Glove)

2) Use Deep LSTMs :- Instead of using single LSTM layers use deep LSTM layers, stacked upon each other :: It helps in :-
   i) long term dependency (as it has many context vectors unlike using single LSTM layer
   ii) Layered represent (It helps ~~in understanding~~ hierarchical understanding of paragraphs i.e from words → sentences → paragraphs)
   iii) If parameters are more (when used deepLSTM), it ensures good accuracy

3) Reversing the input :- In encoder, we can provide input in reverse direction too, to improve model performance (but it works in very few cases only.)

   Input → Think about it
   Reverse → it about Think

* Encoder

- Purpose :- Understand the input sequence and compress its meaning
- How :- 1) Takes input (e.g. an English sentence)
  2) Processes it word by word (using RNN/LSTM)
  3) Converts the entire input sequence into a context vector (a fixed size representation)

* Decoder :-

- Purpose :- Generate the output sequence based on the encoder's context
- How :- 1) Takes the context vector from the encoder as input
  2) Predicts the output sequence one token at a time (e.g. in French)
  3) Each prediction is based on the context and previously generated tokens

* Example :-

Input :- "I am happy"
Encoder :- Understands and encodes this into a context vector
Decoder :- Uses this vector to generate output like "Je suis heureux" (French translation)