① RNN :-

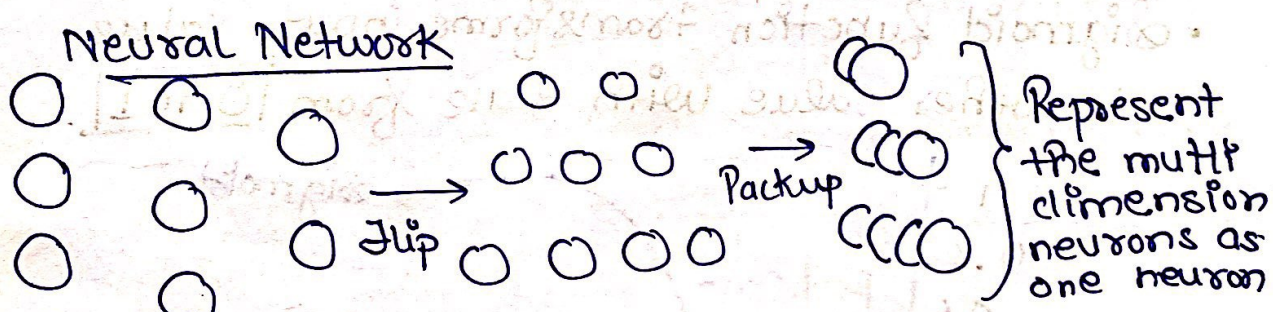
- Used for time series analysis.
- Prediction based on previous data ; not only the current data.
- Make use of sequential data.

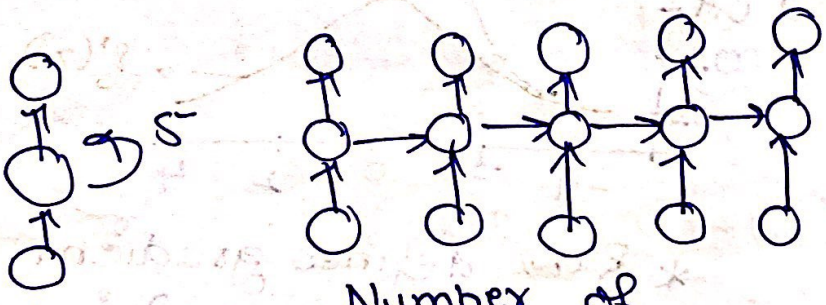
Examples :-

① Stock Price Prediction :- You need to observe the data & trend previously, to make a new prediction.

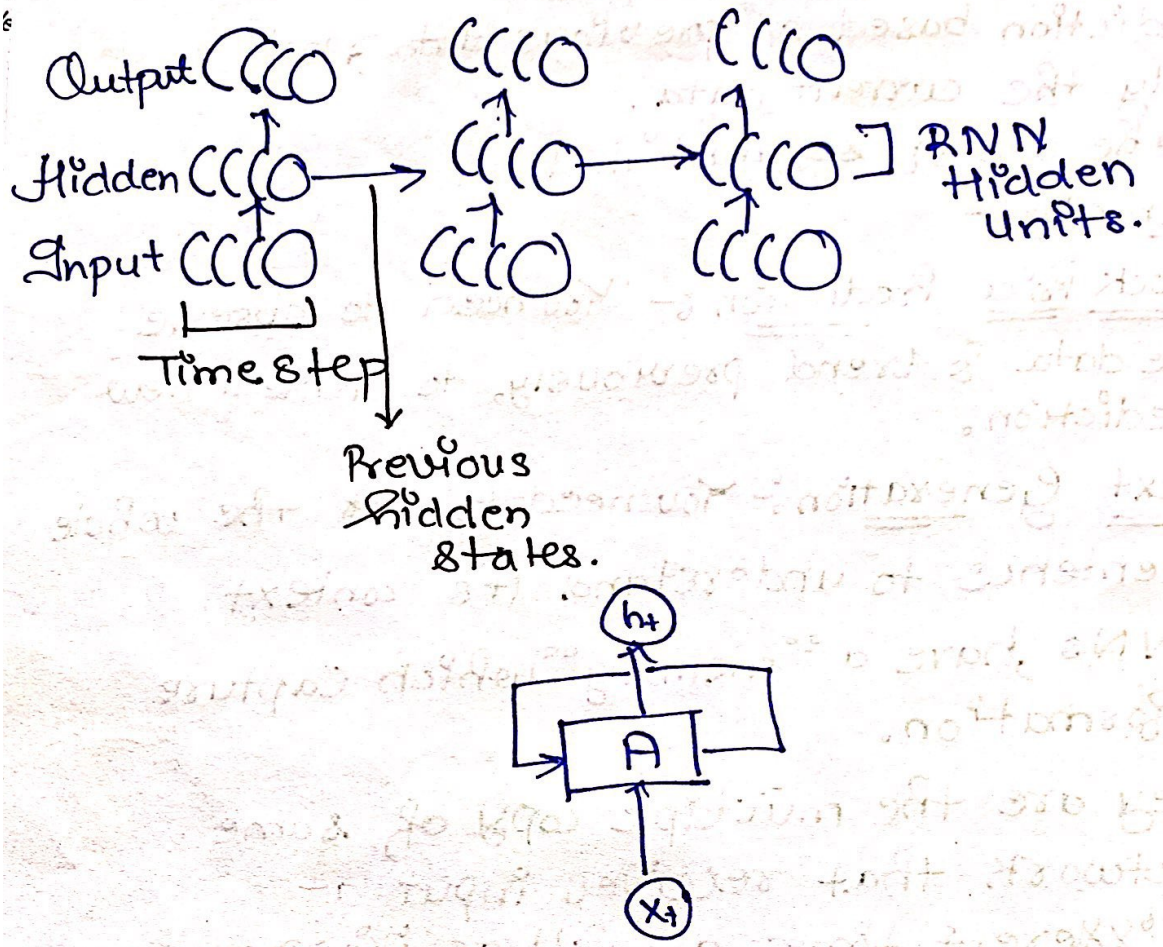
② Text Generation :- You need to look the whole sentence to understand it's context.

- RNNs, have a "memory" which capture information.
- They are the multiple copy of same network that receives input at different times as well as it's previous hidden state.



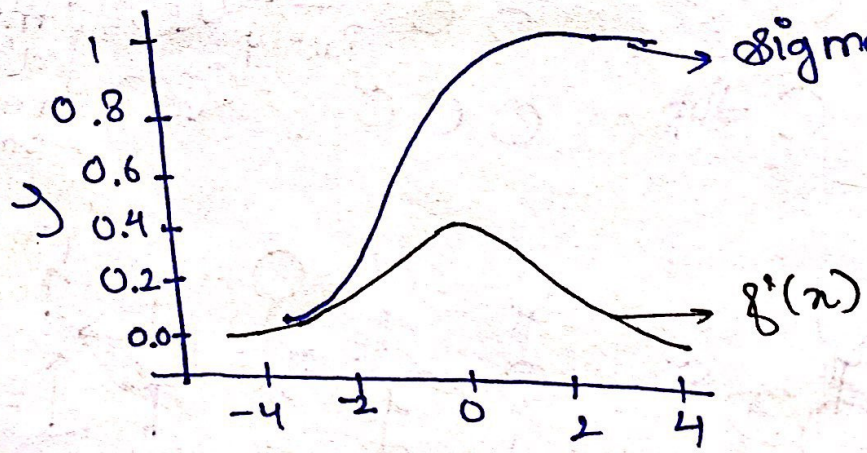


(2)



The activations in RNN's:

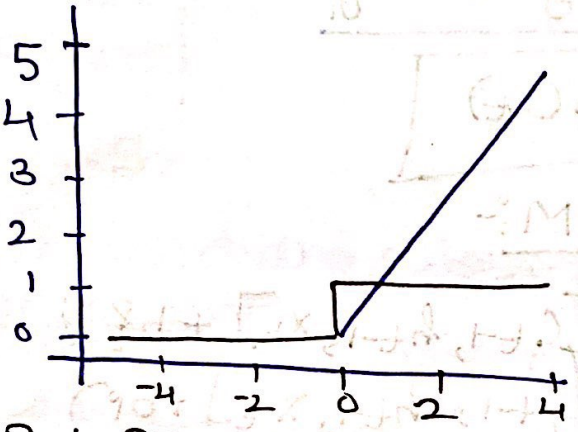
- Sigmoid function transforms input value to other value with scale from 0 to 1.



* $f'(x)$ defines gradient vanishing problem as (0.2) is very small.

③

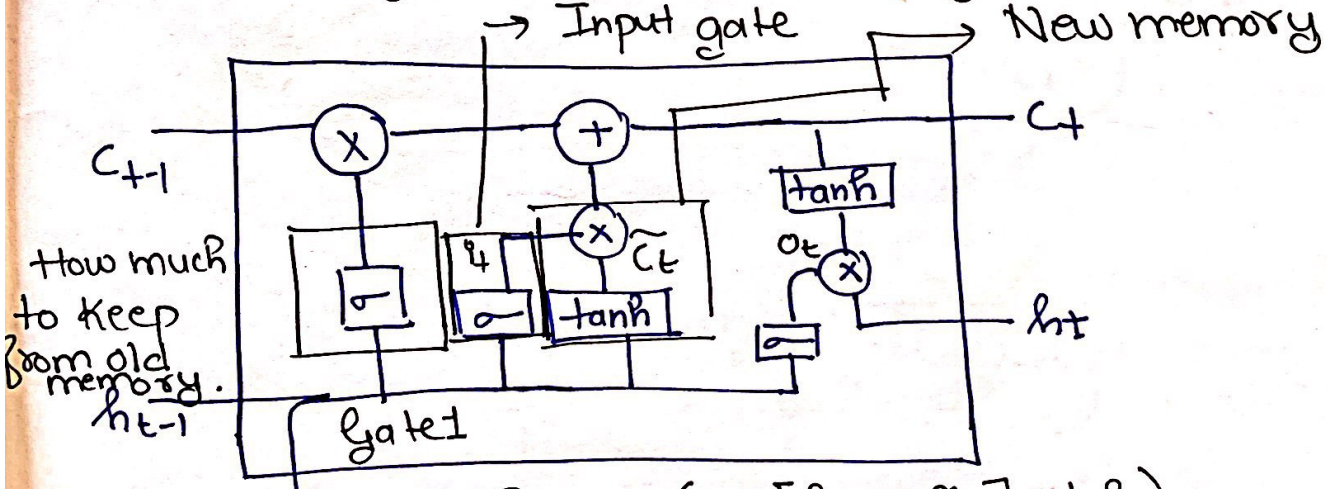
* Vanishing gradient leads to dimishing the effect of lower level layers.
* This leads to change in activation function with ReLU activation.



* But ReLU also has vanishing gradient for vanishing gradient.
* This leads to leaky ReLU.-

Leaky ReLU Derivative: $g'(z)$ → 1 if $z > 0$; 0.01 if $z < 0$

* All this follows to a solution → LSTM (Long Short Term Memory).



$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$

0 → None
1 → All
0.5 → Partial.

④

* Gate 2 controls how much to take from new memory.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

C_t: New memory; C_{t-1}: old memory; $\tilde{C}_t$: New memory.

$$o_t = \sigma(W \cdot [h_{t-1}, x_t] + b_o$$

→ output

Hidden state

$$h_t = o_t * \tan(L_t)$$

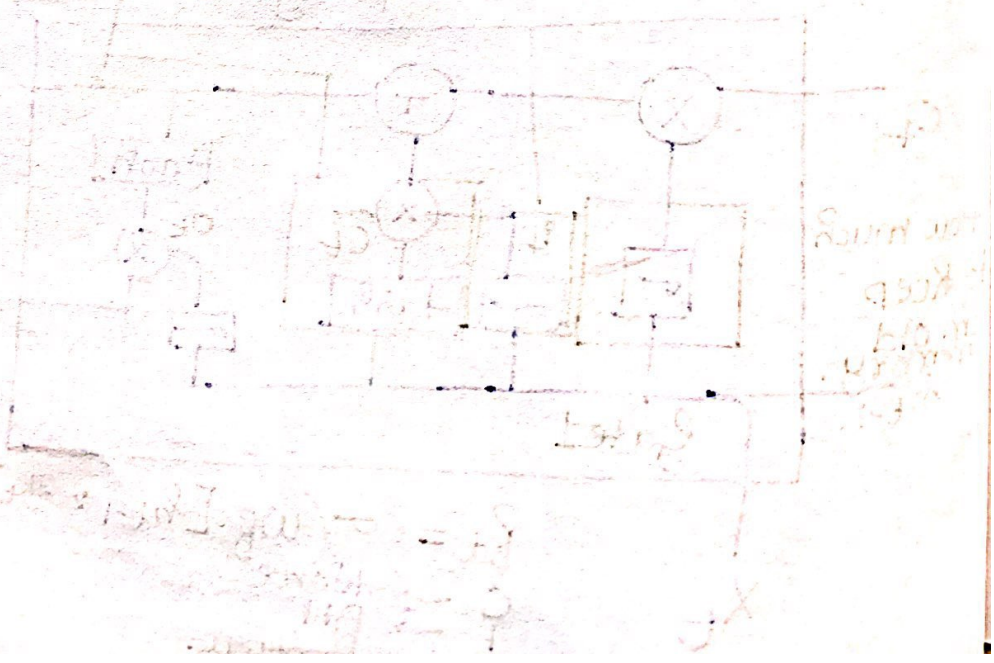
Conclusion of LSTM:-

①
$$f_t = \sigma(W_f . [C_{t-1}, h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i . [C_{t-1}, h_{t-1}, x_t] + b_i)$$
$$o_t = \sigma(W_o . [C_t, h_{t-1}, x_t] + b_o)$$

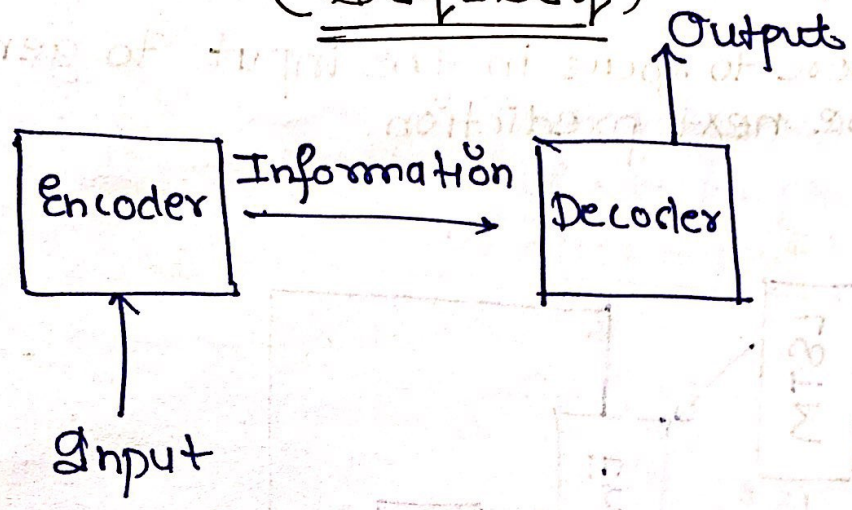
Some LSTM variants.

② Use of coupled forget and input gates.

$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C}_t$$



(5) Sequence to Sequence (Seq2Seq)



* The input and output length can be different.
  ↓
  for example Machine Translation

Machine Translation:-

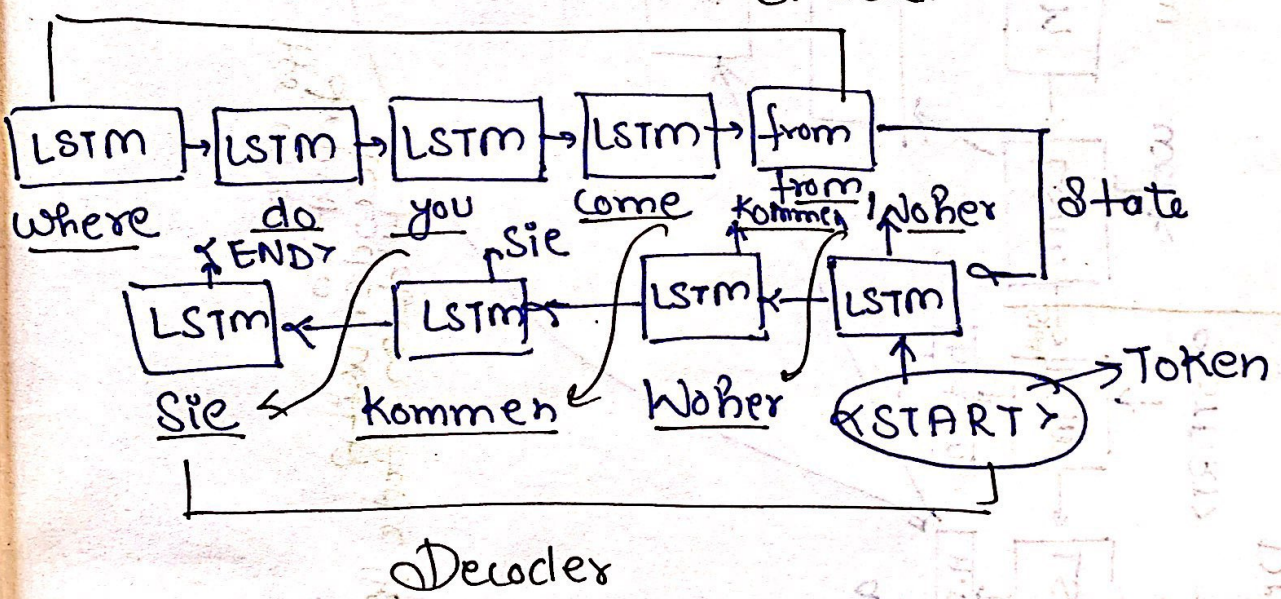


Diagram Using Testing Time

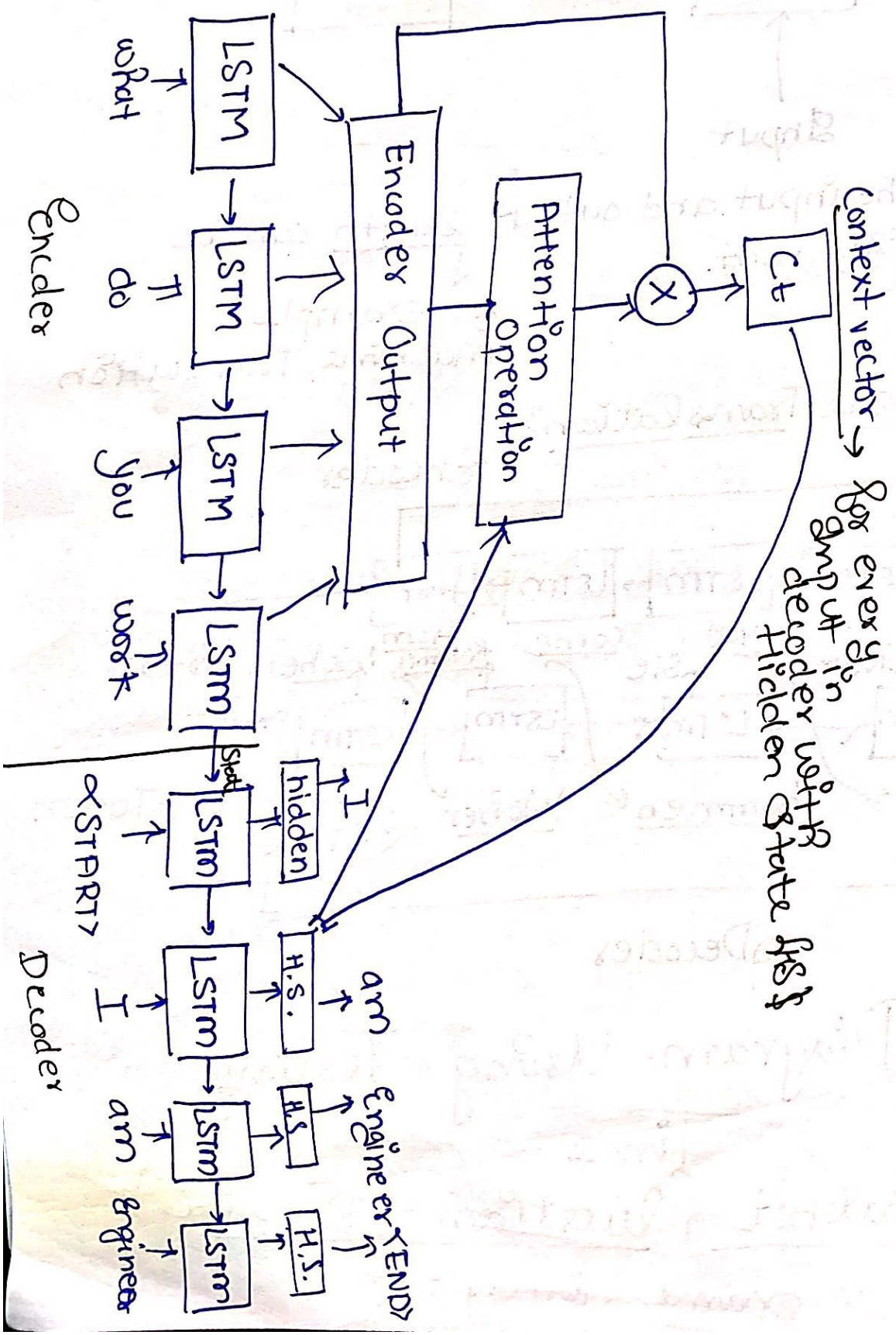
Chatbot {Question - Answer}

Same concept

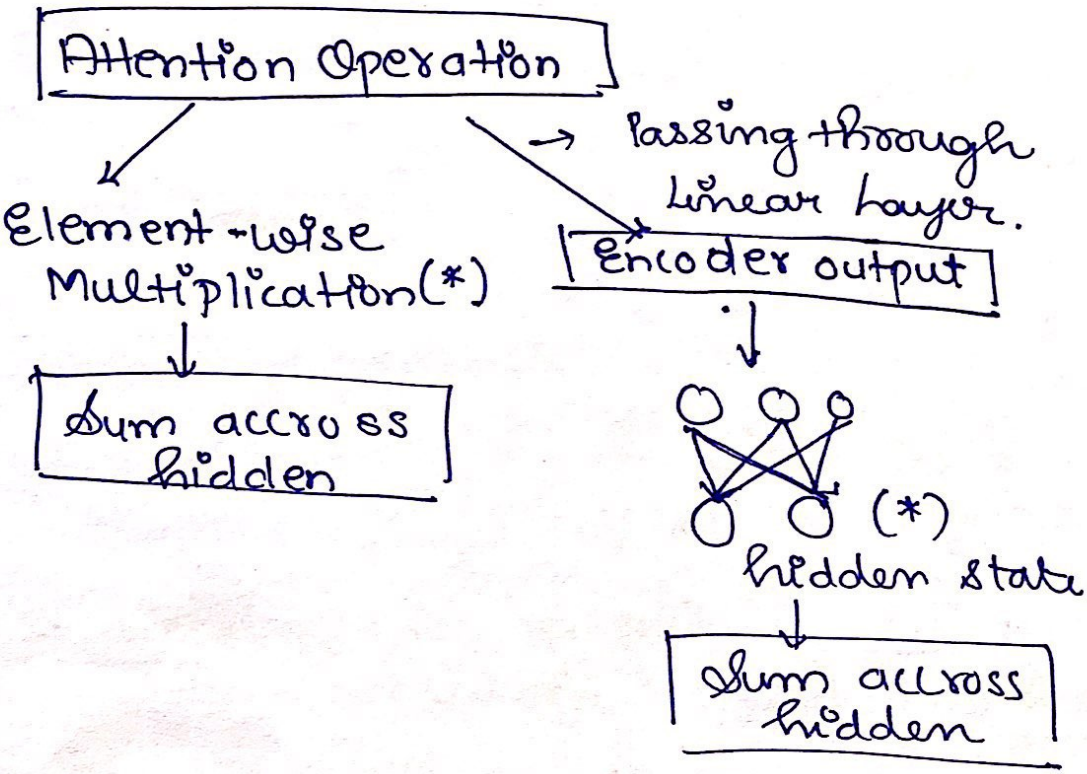
⑥

Attention Mechanism:

* where to focus in the input to generate the next prediction.



(7)



(8)

Pytorch :-

- It has dynamic computational graphs which is useful in RNN(s).

* Torch Tensors :-

tensor.shape → Provides the shape; its an attribute

tensor.size() → Provides the shape; its a method; we can pass arguments in it.

tensor.view() → Its for reshaping, [-1] is used for inferred dimension.

torch.rand() → For creating tensors with random numbers 0-1.

torch.randn() → where n stands for normal distribution.

torch.randint() → random integers with range, shape as arguments

etc.

Chatbot Implementation

PyTorch

PART 1: DATA PREPROCESSING:-

* Download the data
* visualize some data
* Convert dataset in the form of readable dictionary.
* Arrange the dataset into question-answers pair.
* Write the question-answers pair into a text file.
* visualize some lines of our file.
* Processes the words.
* Create a vocabulary class
* define a constructor with

```
self. name
self. word2index = { }
self. word2count = { }
self. index2word = { PAD - - - . }
self. num_words = 3.
```

* define methods in the vocabulary class for adding sentence & words to the vocabulary.
* Remove words below a certain count threshold.
* Process the text.
* Turn the unicode string to plain ASCII.
* Normalize the string using regular expression.

* Filter the pairs according to the length.
* Add the pairs to the vocabulary class.
* Trim the rare words.

PART 2: DATA PREPARATION:-

* Each word is represented as index

hello	how	are	you	today
can	you	say	that	0
you	are	good	0	0
well	ok	0	0	0
sure	0	0	0	0

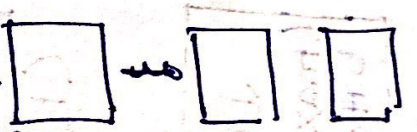
(batch_size, max_length

↓ transpose

hello	can	you	well	sure
how	you	are	ok	0
are	say	good	0	0
you	that	0	0	0
today	0	0	0	0
5	4	3	2	1

(max_length(seq), batchsize)

pack padd seq



hello	how	are
can	you	say
you	are	good
well	ok	
sure		

Batchwise processing

* Take the indexes of each word from a sentence.
* Zero padding to make the correct size.
* Binary matrix for our tensor.

(11)

PART 3: BUILDING THE MODEL :-

* nn.Module is a class includes all neural networks.

(12)

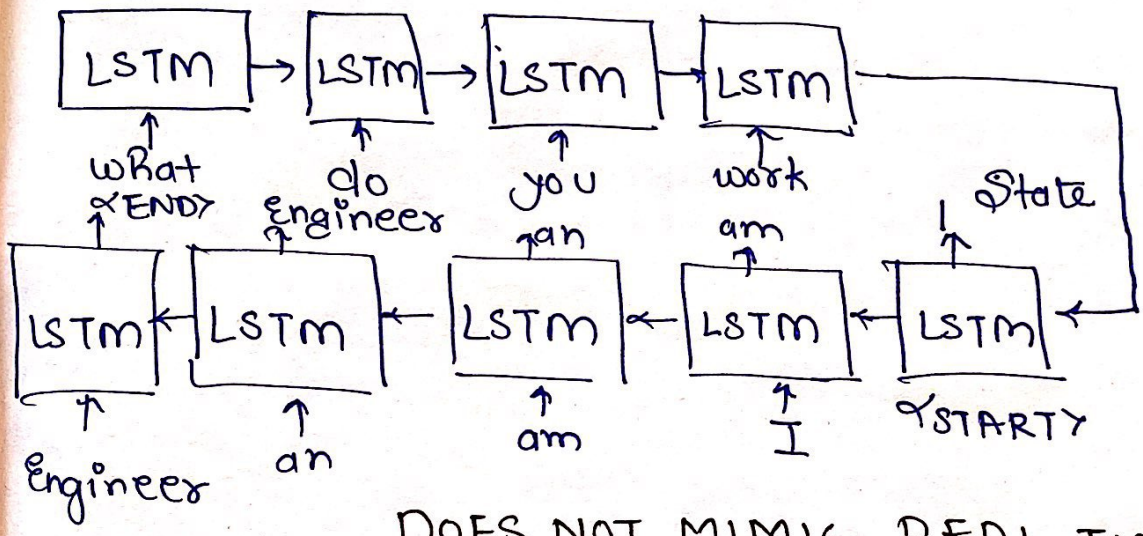
encoder output ← $\bar{h_s}$; hidden state → h_t

$$score(h_t, \bar{h_s}) = \begin{cases} h_t^T \bar{h_s} & \text{dot} \\ h_t^T W_a \bar{h_s} & \text{general} \\ v_a^T \tanh(W_a [h_t ; \bar{h_s}]) & \text{concat.} \end{cases}$$

* Dropout reduces dependency of neurons.

PART 4: TRAINING THE MODEL :-

Teacher Forcing



DOES NOT MIMIC REAL TIME SITUATIONS