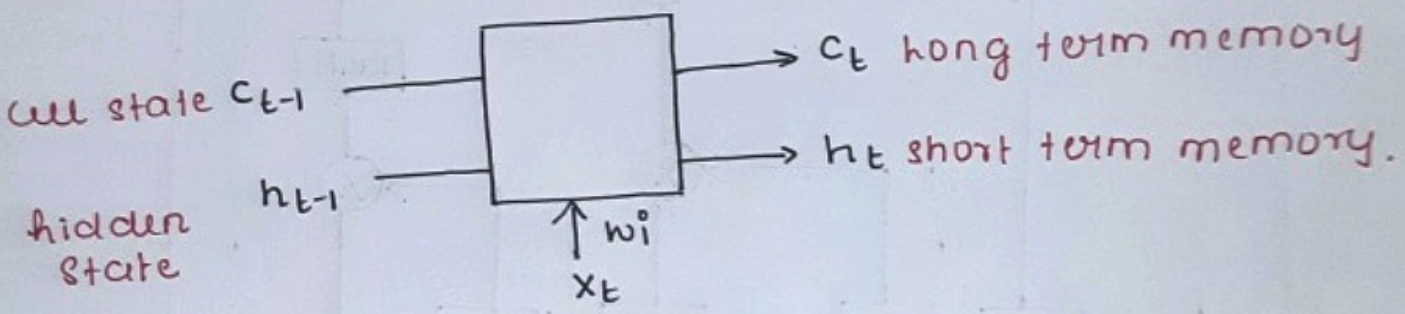
Long Short Term Memory Networks (LSTM)

It is a special kind of RNN capable of learning long term dependencies.

It processes sequences of data (text, speech, time series data) where there is need to remember information from previous time steps to make accurate predictions.

LSTMs contain a chain of memory cell that store information for a long time.

Memory cell:



consider the following example:

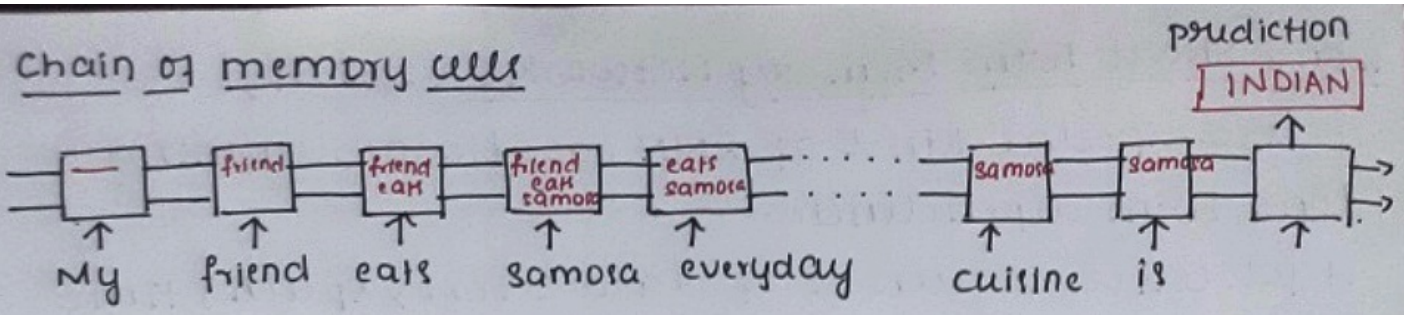
My friend eat samosa everyday. It shouldn't be hard to guess that his favourite cuisine is INDIAN

My brother likes pasta and cheese. That means his favourite cuisine is ITALIAN.

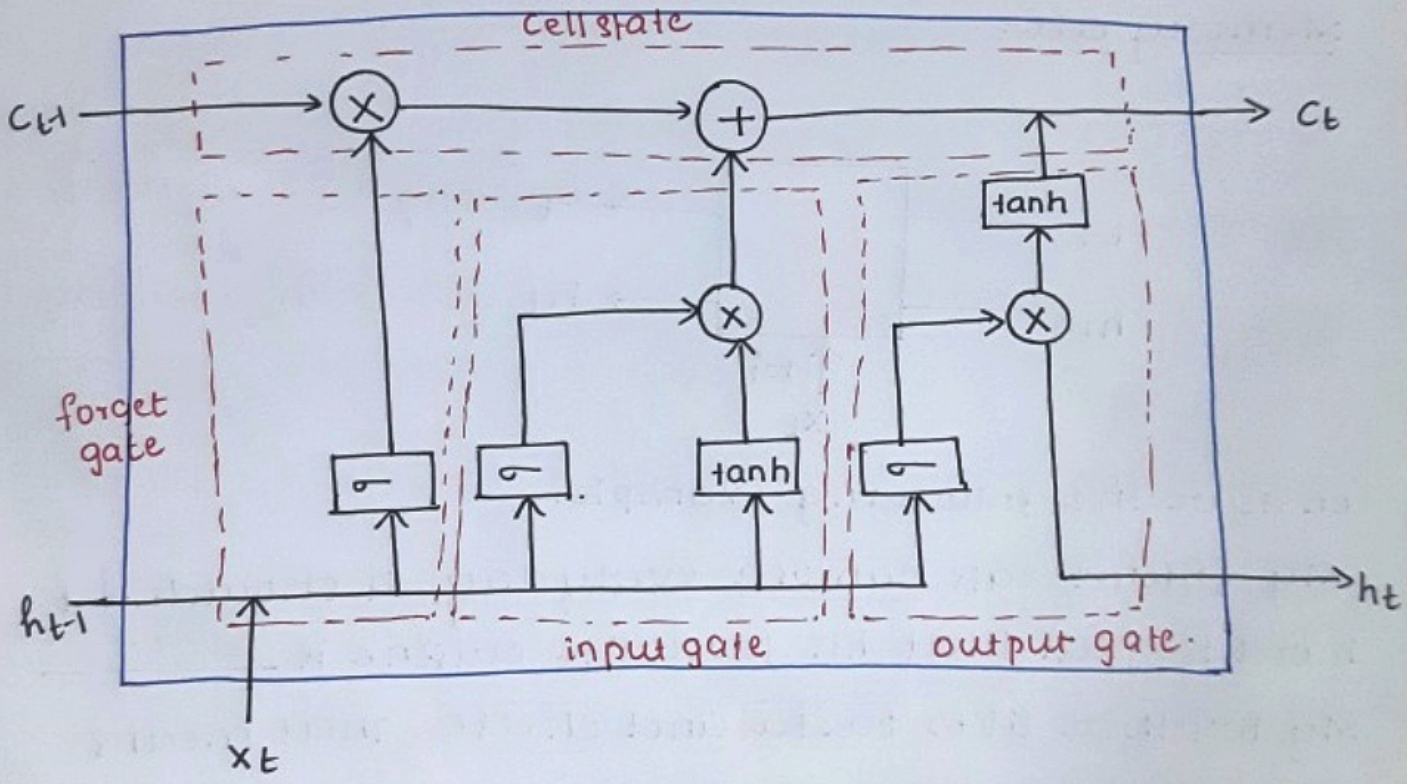
The model predict future word based on keywords stored in its memory network. Keywords are relevant words required for prediction stored based on context.

The model learns which words to select as keywords during the training process from a large dataset.

Chain of memory cells



Each memory cell in LSTM is made up of 4 neural network layers interacting in a special way: The cell state layer and three gate layers - forget gate, input gate, output gate.



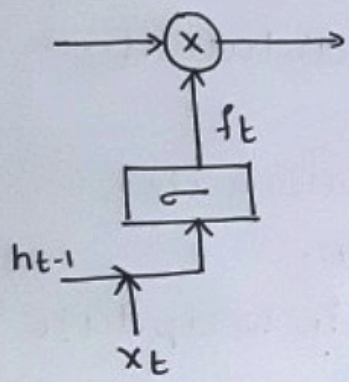
The cell state:

* It is a memory network for long term dependencies that remembers keywords.
* Cell state undergoes minor linear interactions (pointwise addition, multiplication) to alter its state based on output from gates.

The gate layers:

* The gate layers regulate the flow of information to the memory cell.
* They protect and control cell states.
* Each gate is essentially a sigmoid activation layer which gives an output of either 0 or 1. Based on this output actions are performed.
* Each gate takes previous hidden state (h_{t-1}) and current input (x_t) ~~for~~ as input. Output will be 0/1 which alters cell state.

Forget gate:



* Forget gate is responsible for determining what information should be discarded from memory cell.
* Output from forget gate:

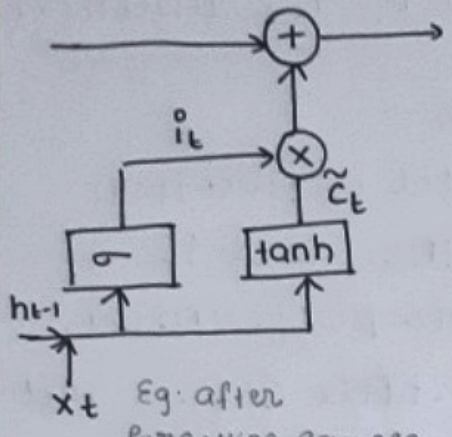
$$f_t = \sigma[W_f(h_{t-1}, x_t) + b_f]$$

Eg: when past a enters, samosa - the previous context has to be forgotten

* h_{t-1} and x_t are passed through sigmoid function (σ). If output is 0, it means context has changed and previous cell state (C_{t-1}) has to be forgotten. If output is 1, context remains same – previous state has to be retained.

* The pointwise multiplication operation is used to forget or retain previous state (C_{t-1}) based on f_t

Input gate:



Eg: after forgetting samosa, pasta has to be added.

* Input gate is responsible for determining what new information should be added to memory cell.
* Output from tanh function: (blw -1 to 1)

$$\tilde{C}_t = \sigma[W_c(h_{t-1}, x_t) + b_c]$$

These represent candidate values that can be added to cell state (new potential keywords).

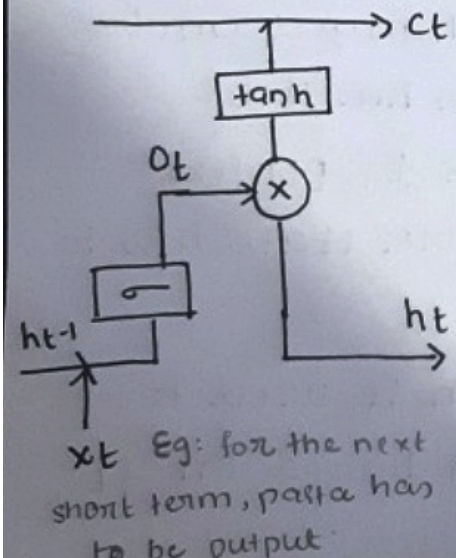
* Output from sigmoid gate: (blw 0 to 1) → add new information; ↙ don't add information

$$i_t = \sigma[W_i(h_{t-1}, x_t) + b_i]$$

This represents how significant each candidate is. (depending on context).

* The product $\tilde{C}_t \cdot i_t$ is the output of input gate which determines the new information.
* The pointwise multiplication uses $\tilde{C}_t \cdot i_t$ to update previous cell state C_{t-1}.

Output gate:



Eg: for the next short term, pasta has to be output

* The output gate is responsible for determining what is outputted from memory cell.
* Output from output gate:

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

$$h_t = o_t * \tanh(C_t)$$

↳ (blw 0 & 1) ↙ don't output; ↓ output

* o_t decides what part of cell state we need to output.
* h_t selectively outputs relevant information required for current task.

Advantages of LSTM:

* Long term memory retention:
  LSTM can retain information for a long time and predict output based on context
* Handles Vanishing Gradient / Exploding Gradient:
  Long term dependency avoids these problems and weights can be successfully updated.
* Versatility:
  LSTM can be used for a variety of NLP tasks, speech recognition, time series analysis etc.
* Ability to handle variable length sequences:
  can handle dynamic sequences of input

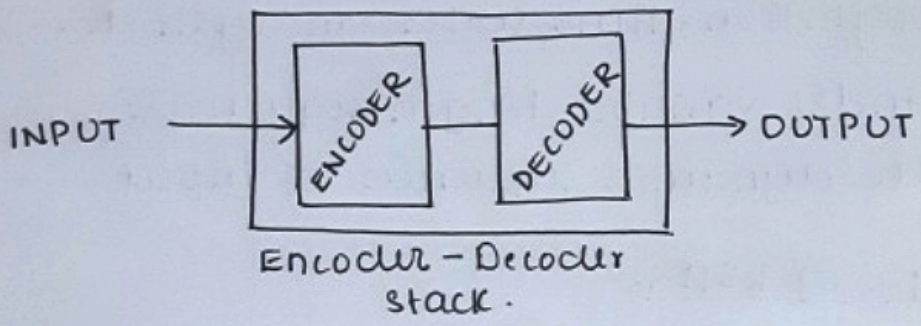
Disadvantages of LSTM:

* Complexity:
  The LSTM architecture is more complex than that of traditional RNN, making it difficult to train and optimize.
* computationally expensive:
  complexity makes it computationally more expensive and unsuitable for speed and efficiency-critical applications.
* Overfitting:
  Model is prone to overfitting if it becomes too complex or if there is insufficient training data which lead to poor generalization performance.

Sequence to Sequence LSTM (Seq2Seq)

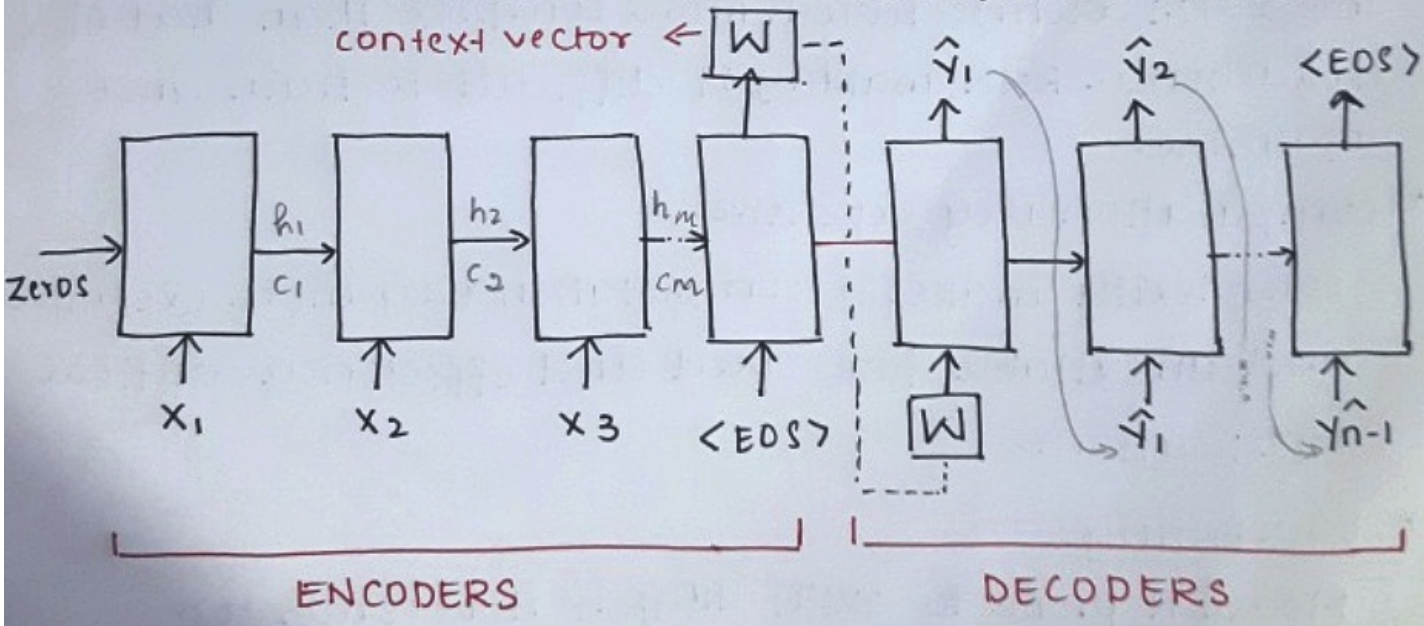
Sequence to sequence LSTM is a neural network architecture designed to handle tasks where input and output are both variable length sequences. It is very useful in applications such as language translation, image captioning, speech recognition etc.

Seq2Seq architecture is composed of 2 main components: Encoder and Decoder.



Encoder - Decoder stack.

LSTM Architecture for sequence to sequence Model:



INPUT sequence: $X = \langle x_1, x_2, \cdots x_m \rangle$

OUPUT sequence: $Y = \langle y_1, y_2, \cdots y_n \rangle$

PREDICTED OUTPUT sequence: $\hat{Y} = \langle \hat{y}_1, \hat{y}_2, \cdots \hat{y}_n \rangle$

Encoder:

* The encoder takes input sequence and converts it into a fixed length vector called the context vector (W), which captures relevant information from input. (Context vector contains hidden state h_t, cell state (c_t))
* Encoder is usually a bidirectional LSTM which process input sequence both forwards and backwards to capture both past and future context.
* The output of encoder is a summary of input sequence - its hidden state

$$h_t = f(W_{hh} h_{t-1} + W_{hx} x_t)$$

Decoders:

* The decoder takes context vector (W) produced by Encoder LSTM and generates output sequence one token at a time.
* The decoder is initialized with W. At each time step t, a new token $\hat{y}_t$ is generated based on hidden state and previous token $\hat{y}_{t-1}$
* Hidden state computation:

$$h_t = f(W_{hh} h_{t-1})$$

* Predicted output:

$$y_t = \text{softmax}(W_s h_t)$$

* The predicted output $\hat{y}_t$ is a probability distribution which helps determine final output.

Training:

* During training, encoder and decoder are jointly trained to minimize loss between predicted and actual output sequence.
* This can be done using loss functions like cross entropy.
* weights are updated during back propagation to optimize model.

Advantages of seq2seq LSTM:

* can handle variable length sequences:
  It processes sequences of data of variable length that is very useful in applications like machine translation.
* Long term dependency:
  Bidirectional RNN can capture both past and future context

Disadvantages of seq2seq LSTM:

* Computationally expensive:
  complexity makes training process computationally expensive
* It cannot handle very long sequences:
  As length of input sequence increases, translational accuracy for the model decreases
* Requires large training set:
  To avoid overfitting and improve generalization, model has to be trained on large, diverse data.