Recurrent Neural Networks

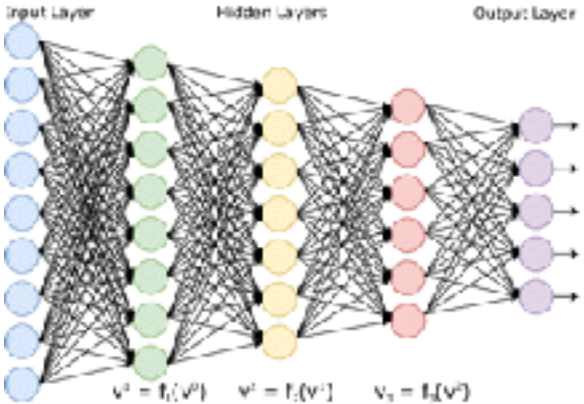
Notes

Content

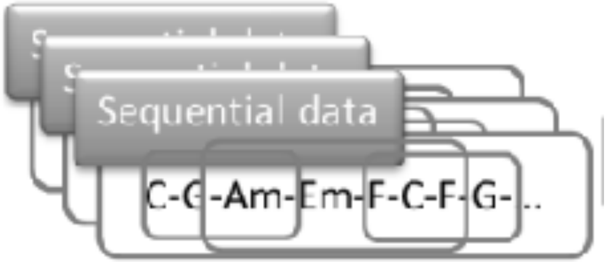
- Background (Where does DNNs lack?)
- Introduction to RNN
- Working of RNN
- Where do RNN lack?
- Introduction to LSTM
- Working of LSTM

Background

Candidate Name	Email	Department Code	Clarity of questions	Usability of Test Interface
Beecher	Beecher1970@yahoo.com	B	4	4
Kalea	Kalea1989@yahoo.com	D	6	1
Gianni	Gianni1981@	B	3	3
Clemmie	Clemmie1983@gmail.com	A	3	3
Jerel	Jerel1987@gmail.com	D	4	4
Isam	Isam1977@company.com	D	6	6
Arnav	Arnav1979@yahoo.com	A	5	5
Orrie	Orrie1976@	D	4	4
Mikel	Mikel1990@gmail.com	E	6	6
Jaida	Jaida1991@company.com	C	5	5
Chanel	Chanel1980@hotmail.com	C	3	3
Vinie	Vinie1980@hotmail.com	H	1	1

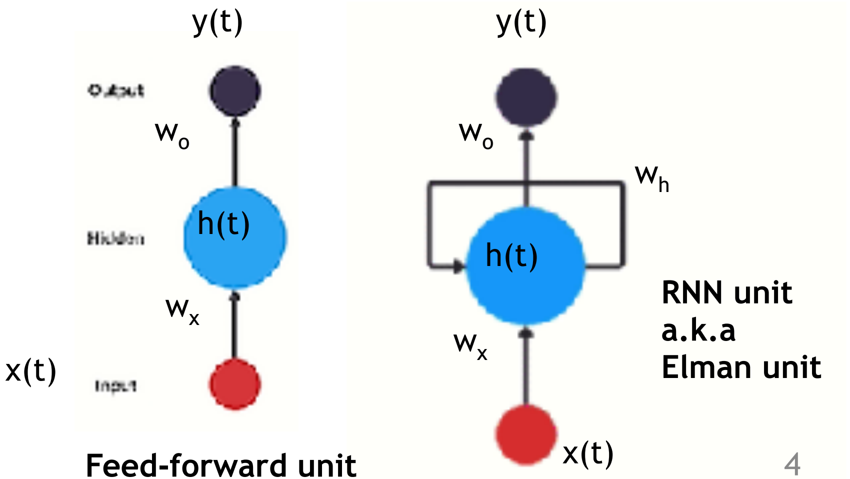


- We are used to tabular data – NxD matrix
- Suppose, a Sequence is of length T, then each sequence is of size TxD
- If there are N sequences, then the size becomes, NxTxD



Background

- used in speech recognition, language translation, stock predictions
- good at modelling sequence data
- They work by using sequential memory
  - e.g.: learning the alphabet sequence
- An RNN has a looping mechanism that acts as a highway to allow information to flow from one step to the next.



Feed-forward unit

RNN unit
a.k.a
Elman unit

- We are used to tabular data – NxD matrix
- Suppose, a Sequence is of length T, then each sequence is of size TxD
- If there are N sequences, then the size becomes, NxTxD

https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9
Deep Learning: Recurrent Neural Networks in Python - Udemy

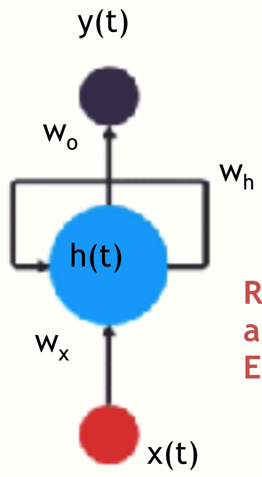
Simple Recurrent Unit

- If h(t) is an M-sized vector (i.e. M hidden units)
  - 1st unit connects back to all M units
  - 2nd unit connects back to all M units and so on
  - Therefore, W_h is an MxM matrix
- Mathematically,

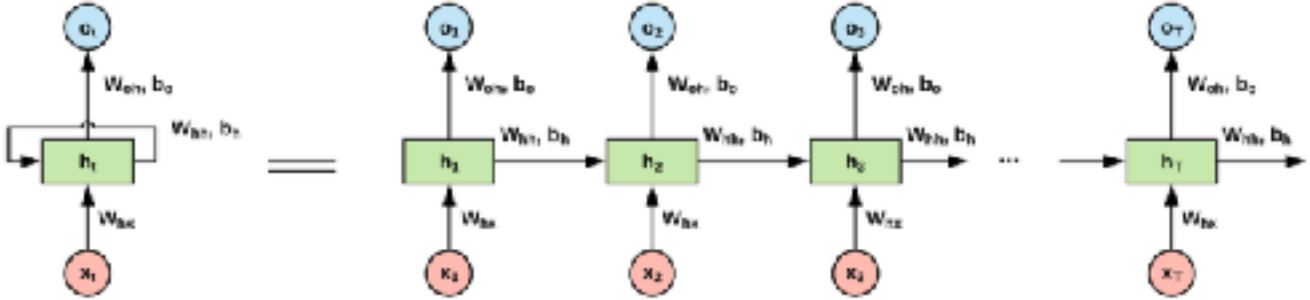
$$h(t) = f(W_h^T h(t-1) + W_x^T x(t) + b_h)$$

$$y(t) = softmax(W_o^T h(t) + b_o)$$

f can be sigmoid, relu, tanh, etc.



RNN unit
a.k.a
Elman unit

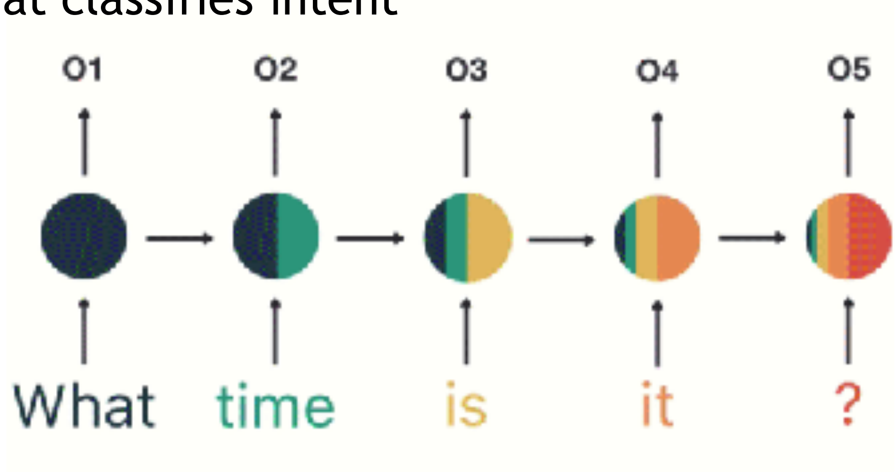


Visualizing a recurrent neural network in terms of a feed forward neural network

Example:

- Suppose we make a chatbot to classify intent
- - RNN used to encode the sequence of text
- - RNN output to a classification model that classifies intent

RNN encoding – previous step information getting incorporated in the current step



Classification based on RNN output

https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

Where do RNNs lack?

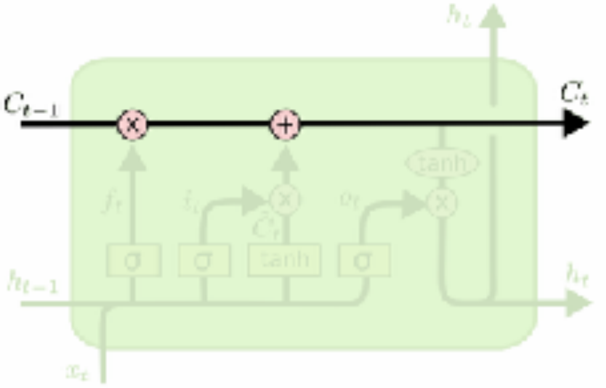
Final hidden state of RNN

- RNNs suffer from short-term memory
  - Caused by vanishing gradients
  - Training an RNN happens through Backpropagation through time.
  - The gradient values will exponentially shrink as it propagates through each time step.

  - Small gradients mean small adjustments. That causes the early layers not to learn.
  - Therefore, the RNN doesn't learn the long-range dependencies across time steps.
- LSTM and GRUs are solutions to the short term memory

https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9
https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Long Short Term Memory Network (LSTM)



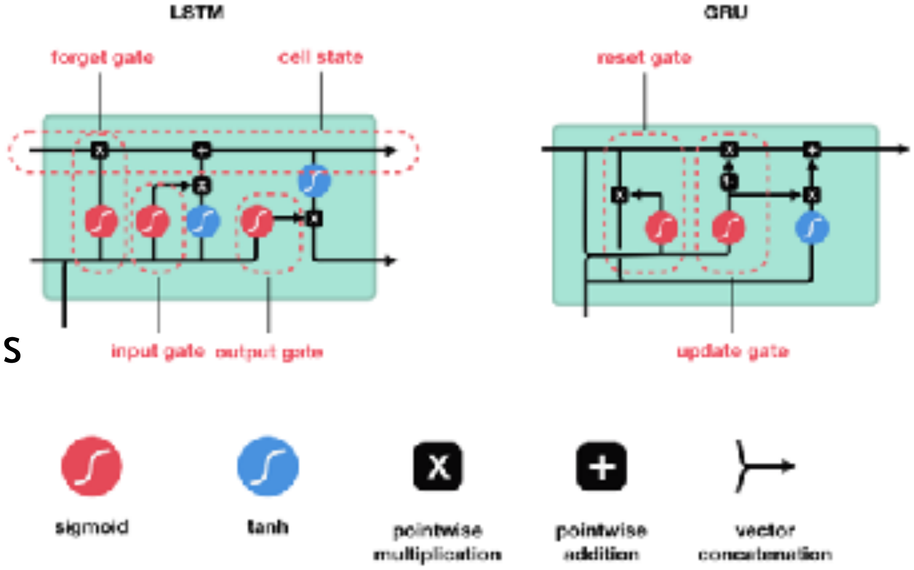
- Core concept of LSTMs are the **cell state** and the **gates**
- Cell State act as a transport highway that transfers relative information all the way down the sequence chain.
  - You can think of it as the "**memory**" of the network
  - Carries relevant information throughout the processing of the sequence
- Information gets added to or removed from the cell state via gates.
- The gates can learn what information is relevant to keep or forget during training.
- Gates contains **sigmoid** activations.
  - helpful for updating or forgetting data
  - because any number getting multiplied by 0 is 0, causing values to disappears or be "forgotten." Any number multiplied by 1 is the same value therefore that value stays the same or is "kept."

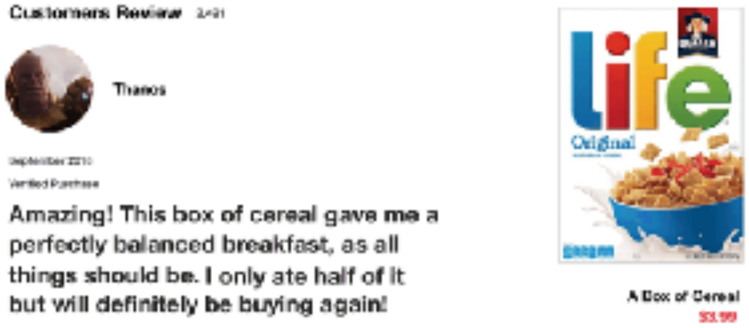
	RNN	LSTM
Time steps	✓	✓
Memory for every time step	✗	✓

LSTM and GRU

- internal mechanisms called gates that can regulate the flow of information.
- gates can learn which data in a sequence is important to keep or throw away

- Analogy towards the working of LSTM and GRUs

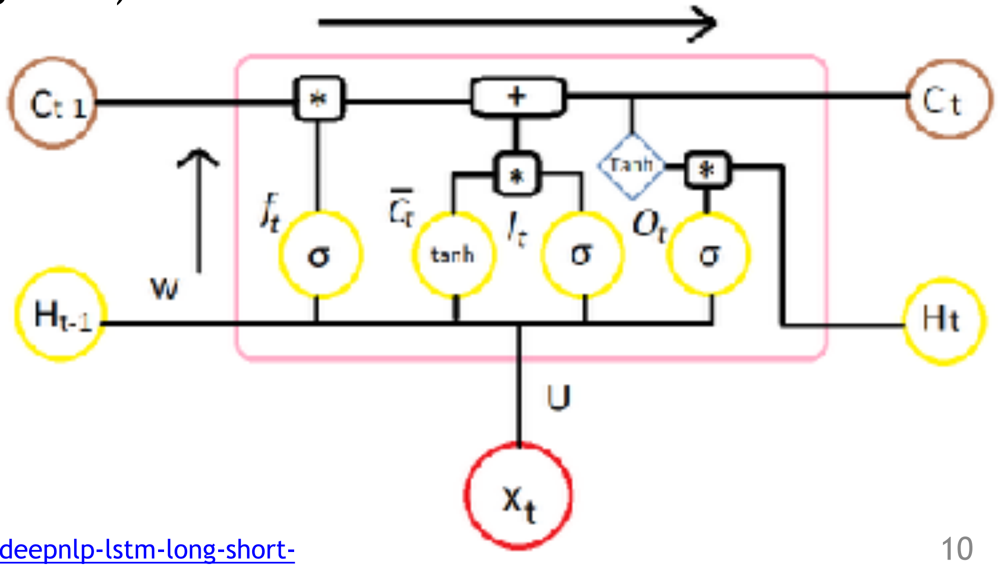




https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

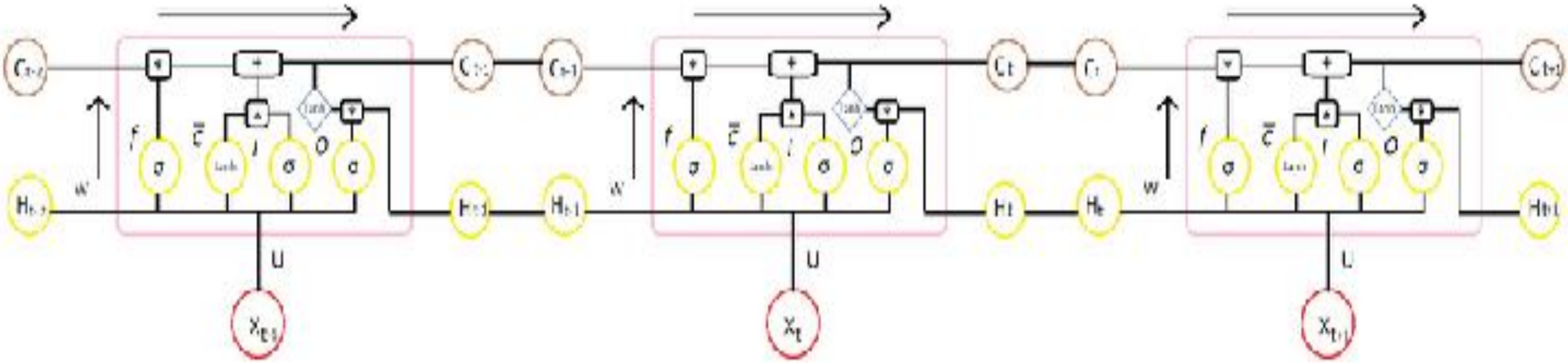
Components of LSTM

- Forget Gate **“f”** (a neural network with sigmoid)
- Candidate layer **“Cbar"**(a NN with Tanh)
- Input Gate **“I”** (a NN with sigmoid)
- Output Gate **“O”**(a NN with sigmoid)
- Hidden state **“H”** (a vector)
- Memory state **“C”** (a vector)



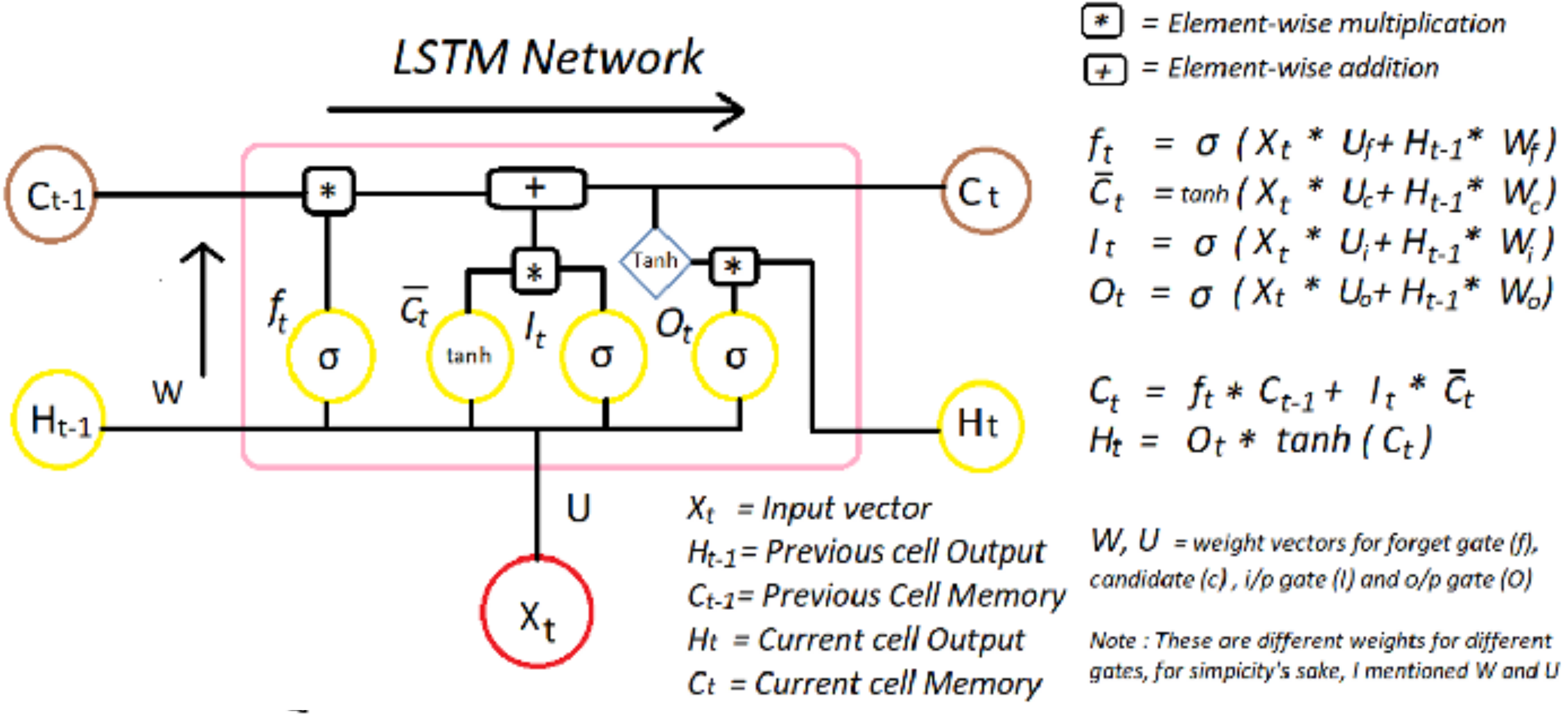
https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235

LSTM Full time steps



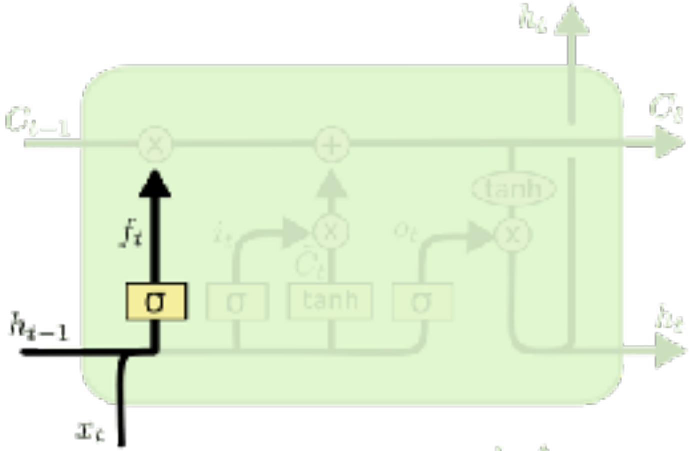
https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235

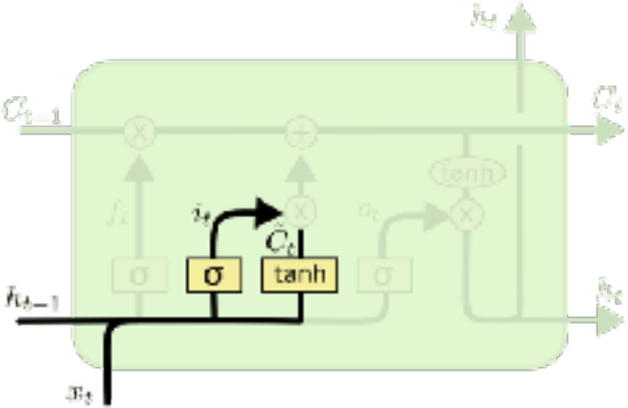
LSTM Flow

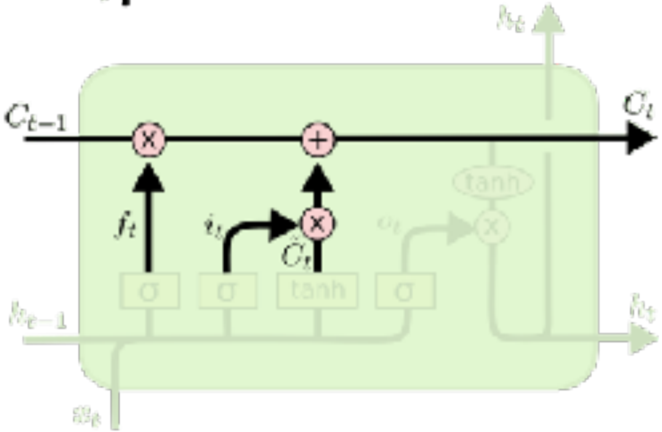


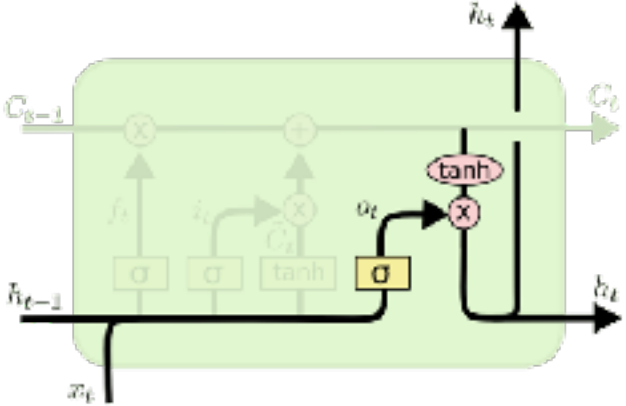
https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235

LSTM Flow





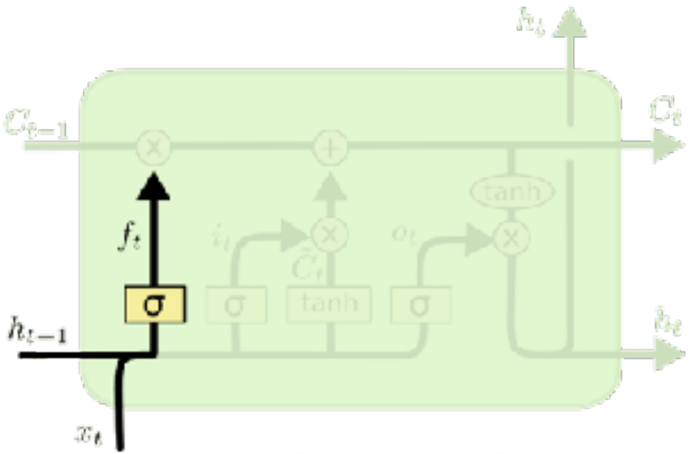




https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235

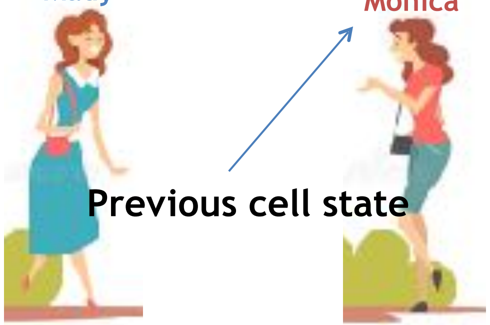
Thank You

Step-by-Step LSTM Walkthrough: Step 1



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

- The first step in LSTM is to decide what information we're going to throw away from the cell state
- Done using the "forget" gate
- It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}.



Previous cell state

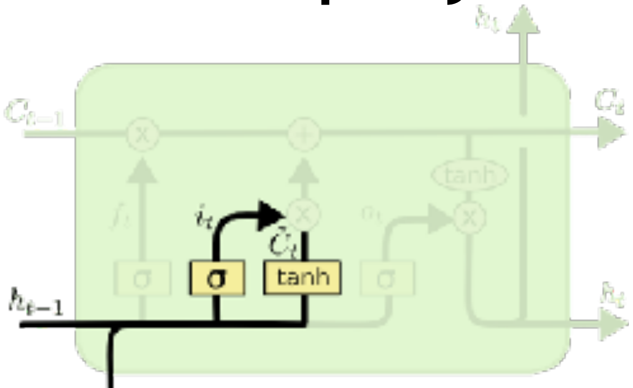
Previous hidden state

Current input

What to forget?
Monica

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

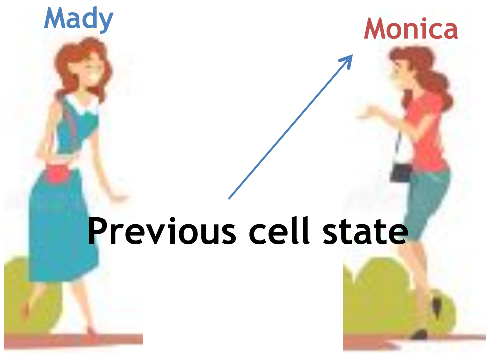
Step-by-Step LSTM Walkthrough: Step 2



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

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

- The next step is to decide what new information we're going to store in the cell state.
- This has two parts:
  - a sigmoid layer called the "input gate layer" decides which values we'll update.
  - Next, a tanh layer creates a vector of new candidate values, $\hat{C}_t$, that could be added to the state.



Previous cell state

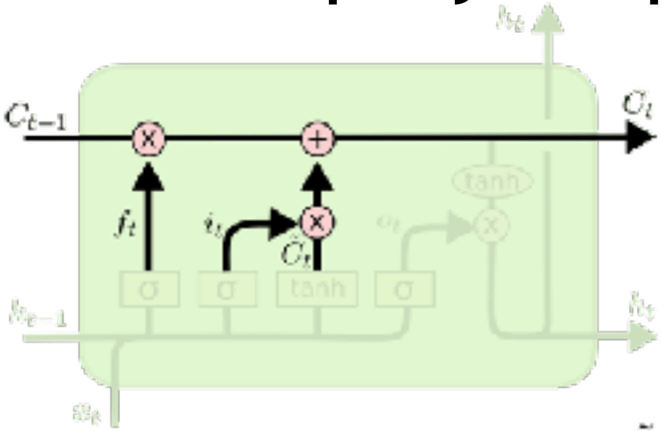
Previous hidden state

Current input

What's new?
Richard

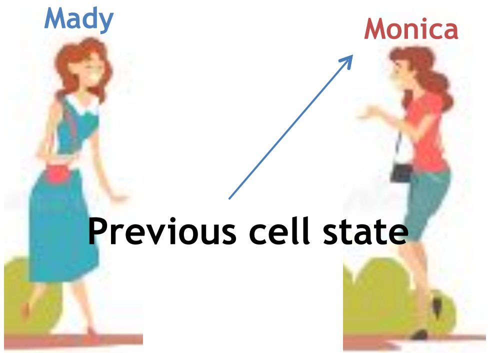
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Step-by-Step LSTM Walkthrough: Step 3



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

- Now, Update the old cell state, C_{t-1}, into the new cell state C_t
- We multiply the old state by f_t, forgetting the things we decided to forget earlier.
- Then we add $i_t * \hat{C}_t$



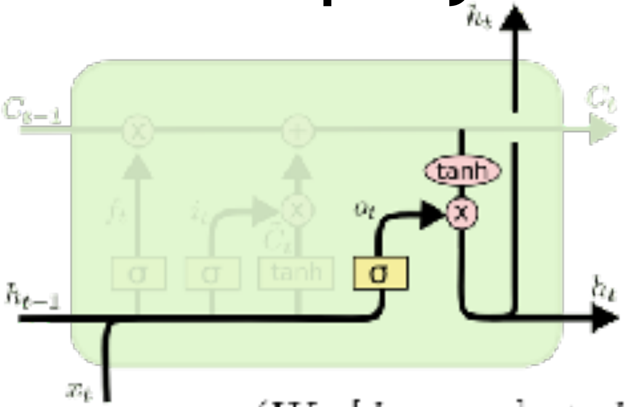
Previous hidden state

Current input

Updated cell state?
Richard

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Step-by-Step LSTM Walkthrough: Step 4

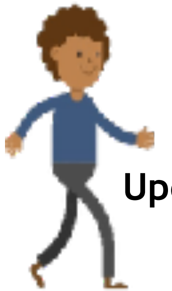


$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$

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

- Getting the new hidden state
- First, we run a sigmoid layer which decides what parts of the cell state we're going to output.
- Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

Richard



Updated Cell state

Previous hidden state

Richard

Current input

New hidden state?
Richard, Walks
Mady, Monica,
Walk, Room

https://colah.github.io/posts/2015-08-Understanding-LSTMs/