Sequence Modeling: Recurrent and Recursive Nets

Sargur Srihari
srihari@buffalo.edu

Topics

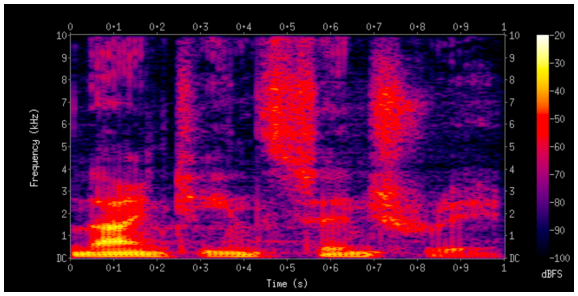
0. Sequential Data and RNN Overview
1. Unfolding Computational Graphs
2. Recurrent Neural Networks
3. Bidirectional RNNs
4. Encoder-Decoder Sequence-to-Sequence Architectures
5. Deep Recurrent Networks
6. Recursive Neural Networks
7. The Challenge of Long-Term Dependencies
8. Echo-State Networks
9. Leaky Units and Other Strategies for Multiple Time Scales
10. LSTM and Other Gated RNNs
11. Optimization for Long-Term Dependencies
12. Explicit Memory

Sequential Data Examples

- Often arise through measurement of time series
  - Acoustic features at successive time frames in speech recognition
  - Sequence of characters in an English sentence
  - Parts of speech of successive words
  - Snowfall measurements on successive days
  - Rainfall measurements on successive days
  - Daily values of currency exchange rate
  - Nucleotide base pairs in a strand of DNA

Speech/NLP tasks with sequential data

1. Sequence-to-sequence
   1. Speech recognition using a sound spectrogram
      - decompose sound waves into frequency, amplitude using Fourier transforms

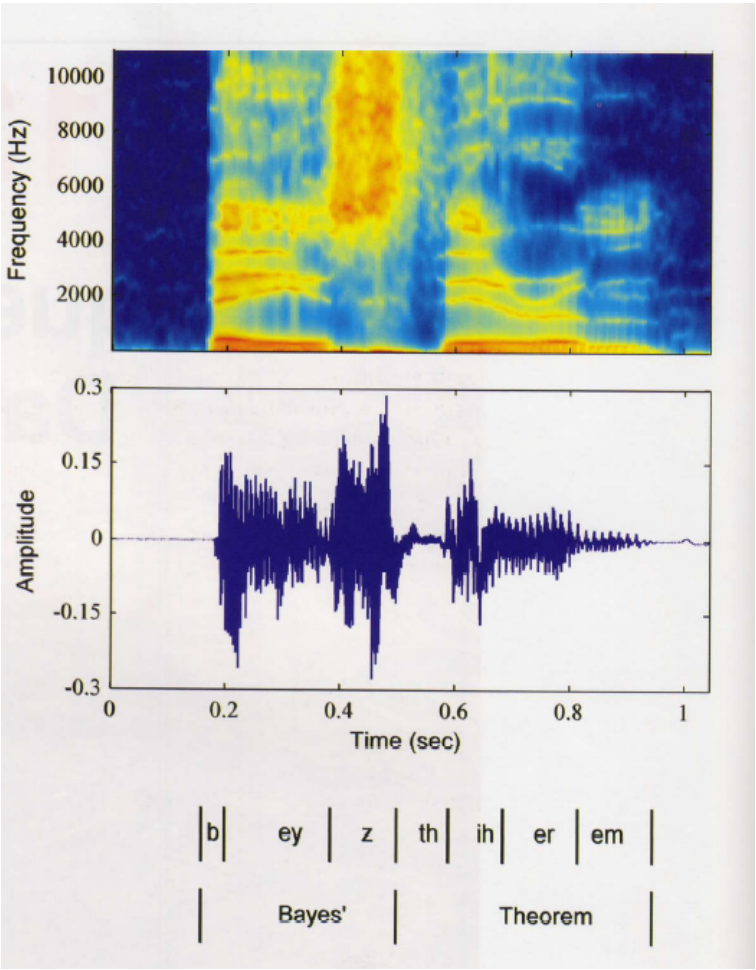


→ "Nineteenth Century"

Frequencies increase up the vertical axis, and time on the horizontal axis.
The lower frequencies are more dense because it is a male voice.
Legend on right shows that the color intensity increases with density

   1. NLP: Named Entity Recognition
      - Input: Jim bought 300 shares of Acme Corp. in 2006
      - NER: $[\text{Jim}]_{\text{Person}}$ bought 300 shares of $[\text{Acme Corp.}]_{\text{Organization}}$ in $[2006]_{\text{Time}}$
   2. Machine Translation: Echte dicke kiste → Awesome sauce
2. Sequence-to-symbol
   1. Sentiment: "*Best movie ever*" → Positive
   2. Speaker recognition: *Sound spectrogram* → Harry
   3. Textual Entailment: Positive: Text entails hypothesis, Negative: Text contradicts
      - Text "*Peter buys the car*" → Hypothesis "*Peter owns the car*"

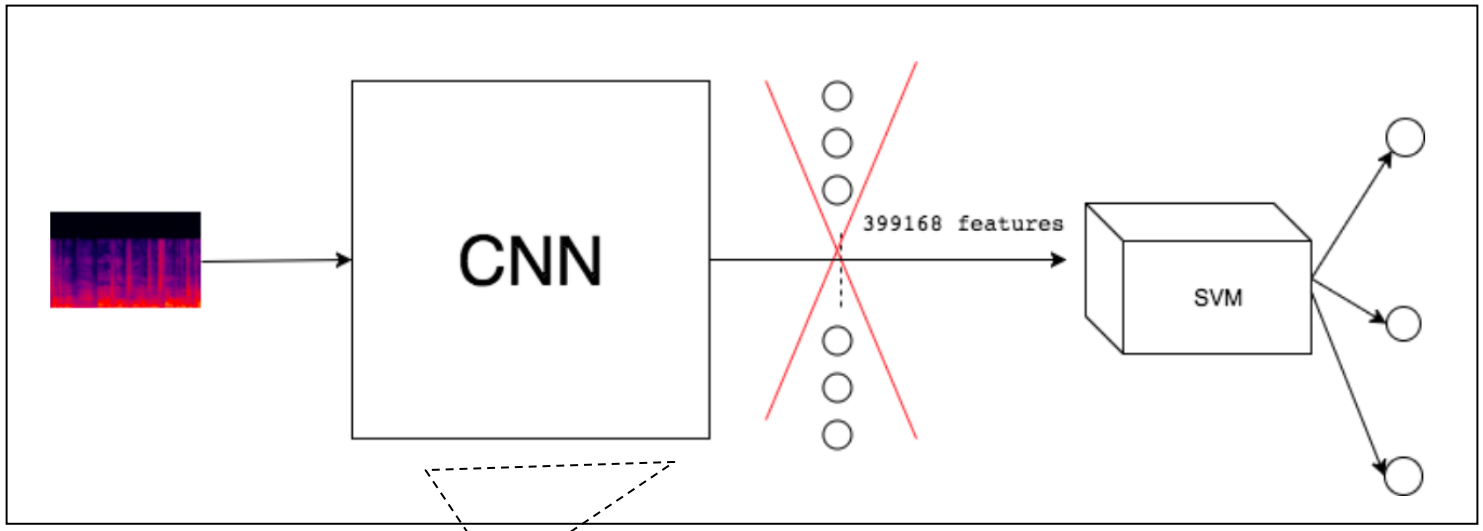
Sound Spectrogram to Word Sequence



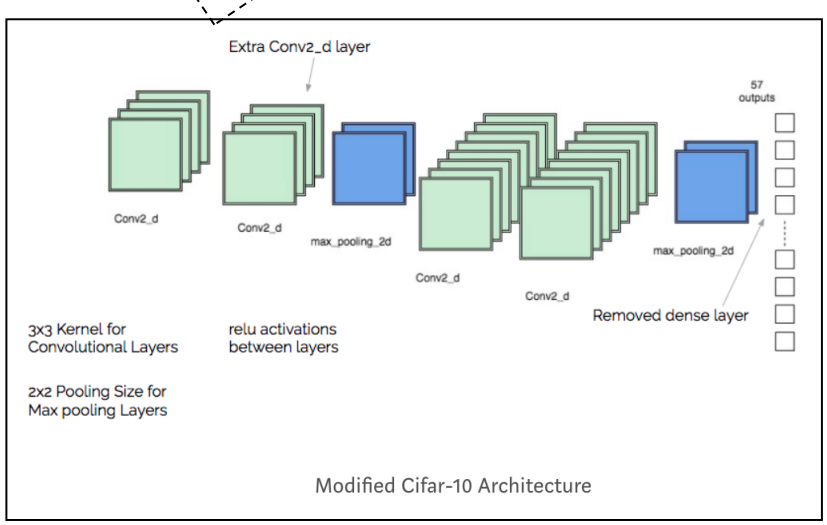
- Decompose sound waves into frequency, amplitude using Fourier transforms
- Plot of the intensity of the spectral coefficients versus time index
- Successive observations of speech spectrum highly correlated (Markov dependency)

Speaker Recognition with CNN

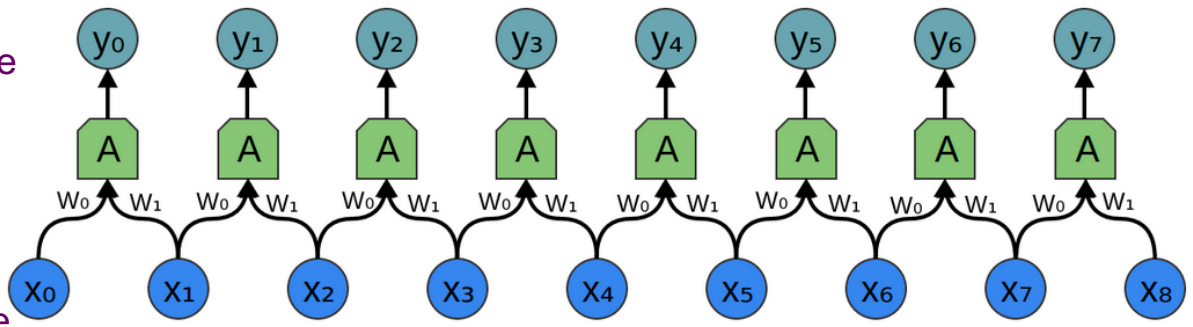
A sequence to symbol task



CNN
Feature Extractor
(Transfer Learning)



1-D CNN for sequence-sequence



Kernel $g(t)$:

$[\ldots 0, w_1, w_0, 0 \ldots]$

Note that kernel gets flipped in convolution

Equations for outputs of this network:

$$y_0 = \sigma\,(w_0 x_0 + w_1 x_1 - b)$$
$$y_1 = \sigma\,(w_0 x_1 + w_1 x_2 - b)$$

etc. upto y_8

We can also write the equations in terms of elements of a general 8×8 weight matrix W as:

$$y_0 = \sigma(W_{0,0}x_0 + W_{0,1}x_1 + W_{0,2}x_2 \ldots)$$
$$y_1 = \sigma(W_{1,0}x_0 + W_{1,1}x_1 + W_{1,2}x_2 \ldots)$$

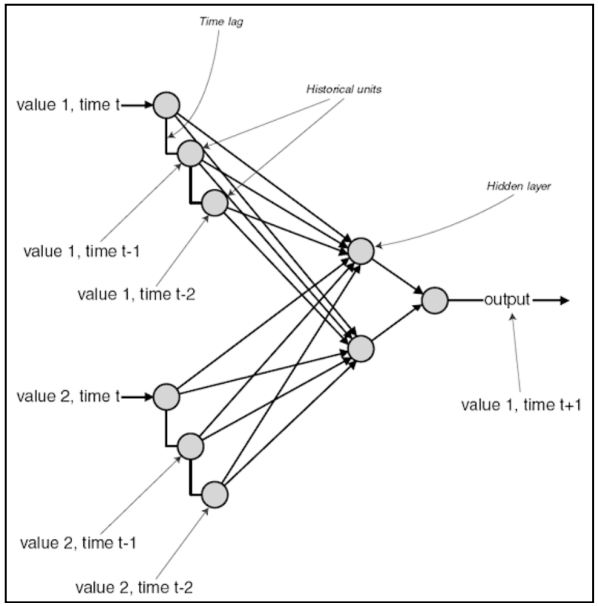
etc. upto y_8

where $W = \begin{bmatrix} w_0 & w_1 & 0 & 0 & \ldots \\ 0 & w_0 & w_1 & 0 & \ldots \\ 0 & 0 & w_0 & w_1 & \ldots \\ 0 & 0 & 0 & w_0 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{bmatrix}$

http://colah.github.io/posts/2014-07-Understanding-Convolutions/

Time Delay Neural Networks (TDNNs)

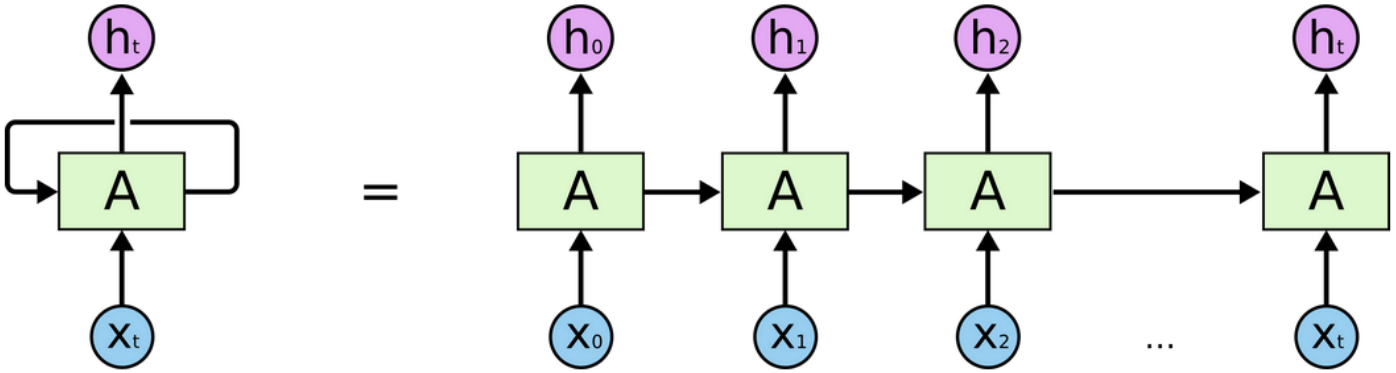
- TDNNs use convolution for a 1-D temporal sequence
  - Convolution allows shared parameters across time, but is shallow
    - Each output is dependent upon a small no. of neighboring inputs
    - Parameter sharing manifests in the application of the same convolutional kernel at each time step



A TDNN remembers the previous few training examples and uses them as input into the network.
The network then works like a feed-forward, back propagation network.

RNN vs. TDNN

- RNNs share parameters in a different way than TDNN
  - Each member of output is a function of previous members of output
  - Each output produced using same update rule applied to previous outputs
  - This recurrent formulation results in sharing of parameters through a very deep computational graph
- An unrolled RNN



Recurrent Neural Networks process sequential data

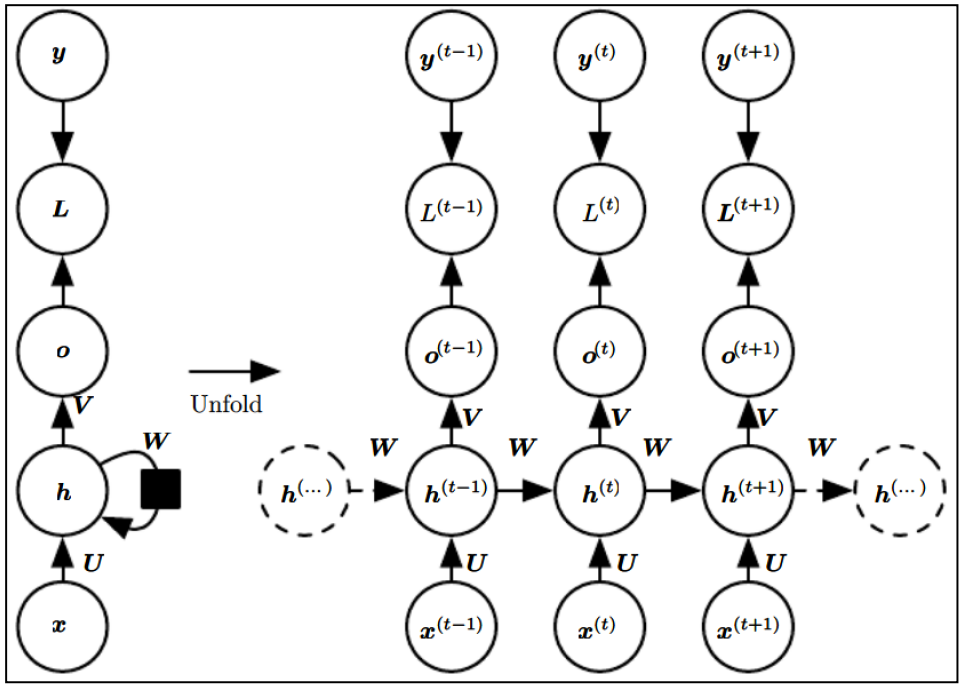
- RNNs are a family of neural nets for sequential data
- Analogy with Convolutional Neural Networks
  - Specialized architectures
    - CNN is specialized for grid of values, e.g., image
    - RNN is specialized for a sequence of values $\boldsymbol{x}^{(1)},..,\boldsymbol{x}^{(\tau)}$
  - Scaling & Variable length
    - CNNs readily scale to images with large width/height
      - CNNs can process variable size images
    - RNNs scale to longer sequences than would be practical for networks without sequence-based specialization
      - RNNs can also process variable-length sequences

RNNs share same weights across Time Steps

- To go from multi-layer networks to RNNs:
  - Need to share parameters across different parts of a model
  - Separate parameters for each value of cannot generalize to sequence lengths not seen during training
  - Share statistical strength across different sequence lengths and across different positions in time
- Sharing important when information can occur at multiple positions in the sequence
  - Given "*I went to Nepal in* 1999 " and "*In* 1999, *I went to Nepal* ", an ML method to extract year, should extract 1999 whether in position 6 or 2
  - A feed-forward network that processes sentences of fixed length would have to learn all of the rules of language separately at each position
  - An RNN shares the same weights across several time steps

Computational Graphs for RNNs

- We extend computational graphs to include cycles
  - Cycles represent the influence of the present value of a variable on its own value at a future time step
  - In a Computational graph nodes are variables/operations
  - RNN to map input sequence of $\boldsymbol{x}$ values to output sequence of $\boldsymbol{o}$ values
    - Loss L measures how far each output $\boldsymbol{o}$ is from the training target $\boldsymbol{y}$



•Forward propagation is given as follows:

For each time step t, t=1 to t=τ

Apply the following equations

$\boldsymbol{o}^{(t)}=\boldsymbol{c}+V\boldsymbol{h}^{(t)}$

$\boldsymbol{h}^{(t)}=\tanh(\boldsymbol{a}^{(t)})$

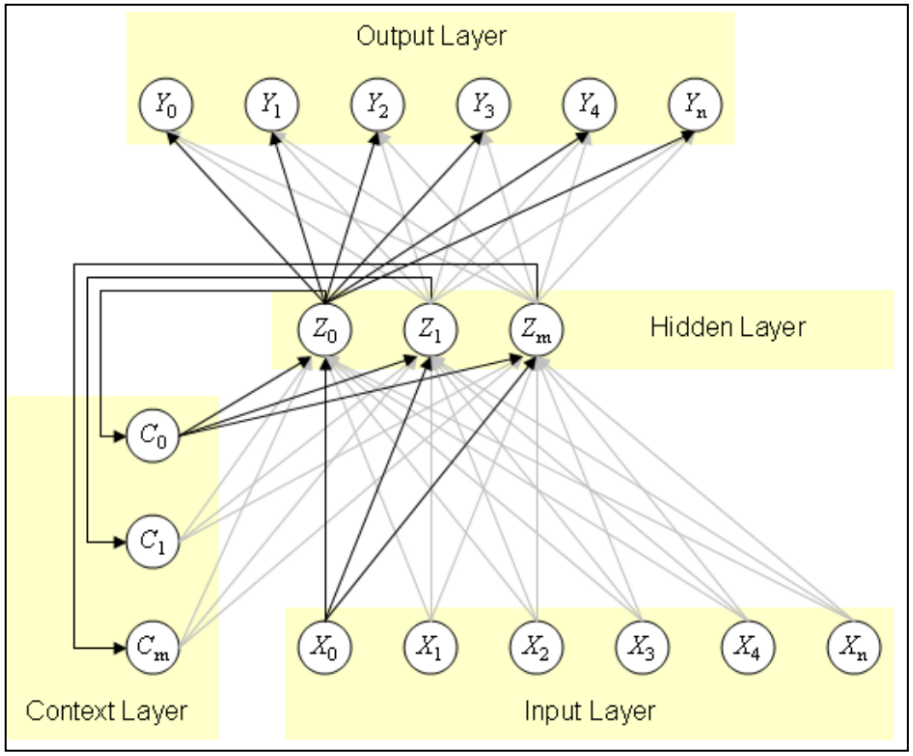
$\boldsymbol{a}^{(t)}=\boldsymbol{b}+W\boldsymbol{h}^{(t-1)}+U\boldsymbol{x}^{(t)}$

RNN operating on a sequence

- RNNs operate on a sequence that contain vector $\boldsymbol{x}^{(t)}$ with time step index t, ranging from 1 to τ
  - Sequence: $\boldsymbol{x}^{(1)},..,\boldsymbol{x}^{(\tau)}$
  - RNNs operate on minibatches of sequences of length τ
- Some remarks about sequences
  - The steps need not refer to passage of time in the real world
  - RNNs can be applied in two-dimensions across spatial data such as image
  - Even when applied to time sequences, network may have connections going backwards in time, provided entire sequence is observed before it is provided to network

RNN as a network with cycles

- An RNN is a class of neural networks where connections between units form a directed cycle
- This creates an internal state of the network which allows it to exhibit dynamic temporal behavior
- The internal memory can be used to process arbitrary sequences of inputs

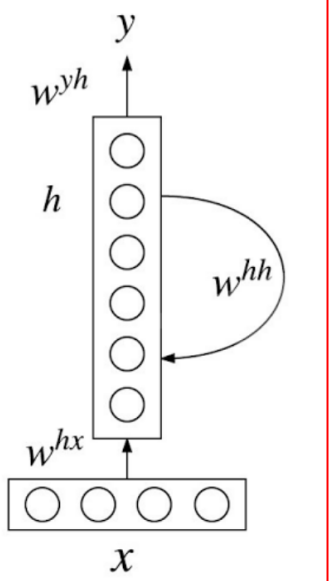


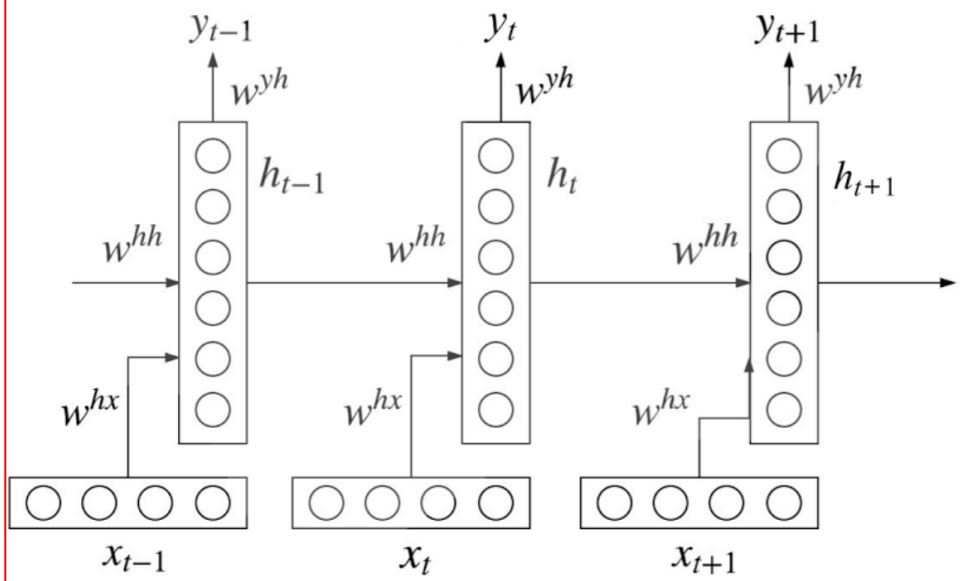
Three layer network with input ***x*** , hidden layer ***z*** and output ***y***
Context units ***c*** maintain a copy of the previous value of the hidden units

RNN parameters

Folded network with cycles

Unfolded sequence network with three time steps

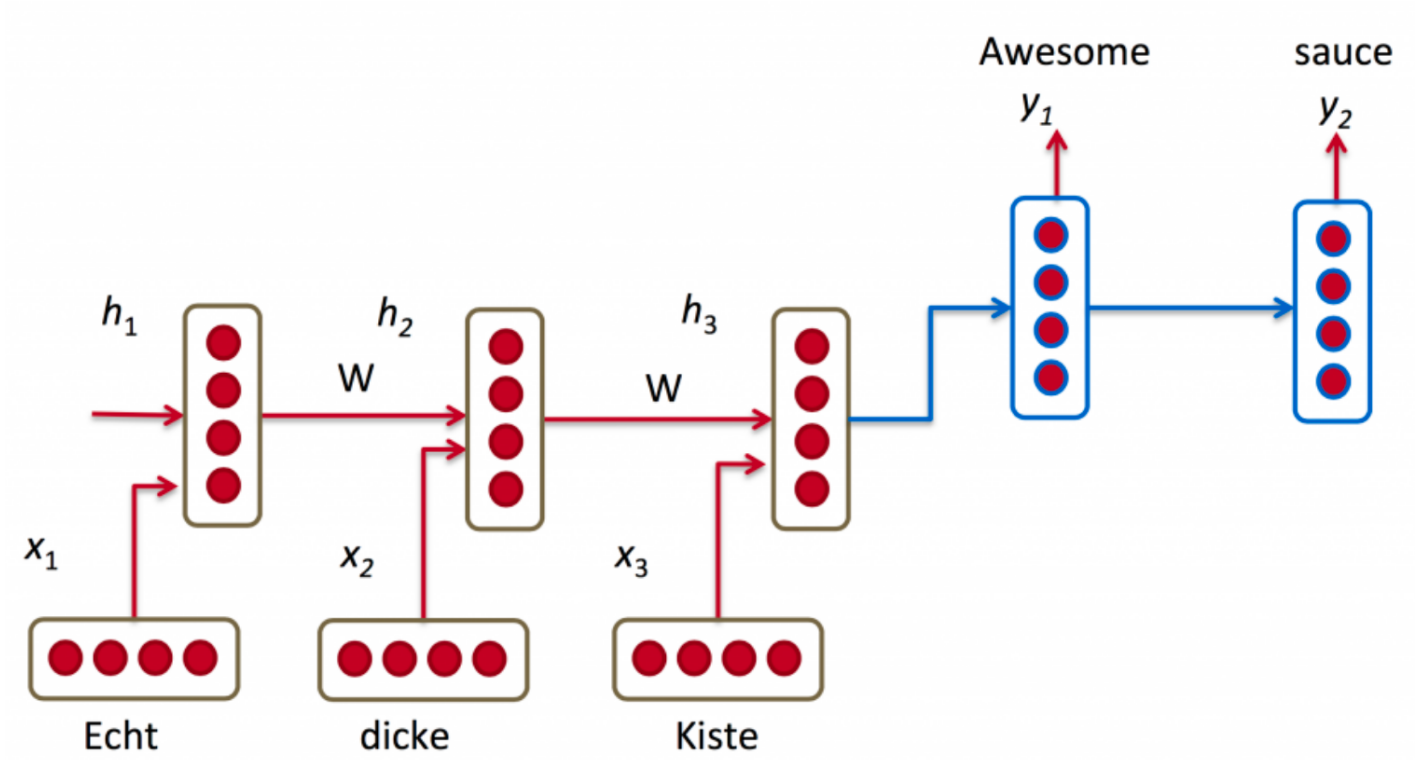




$$h_t = f(w^{hh} h_{t-1} + w^{hx} x_t)$$
$$y_t = \text{softmax}(w^{yh} h_t)$$

Unlike a feedforward neural network, which uses different parameters at each layer, RNN shares the same parameters $(\mathrm{W}^{hx}, \mathrm{W}^{hh}, \mathrm{W}^{yh})$ across all steps

RNN for Machine Translation

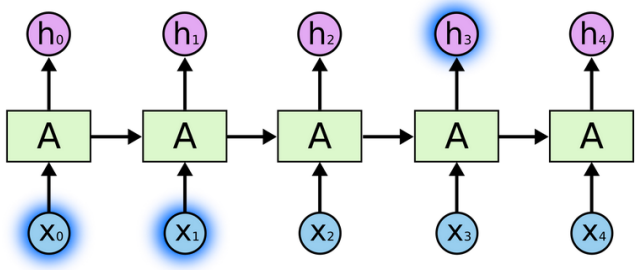


Limitation of length of time steps

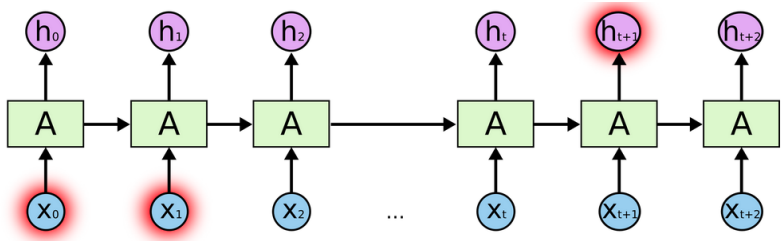
- Length of time steps is determined by length of input
  - e.g., if word sequence to be processed is a sentence of six words, RNN would be unfolded into a neural net with six time steps or layers.
    - One layer corresponds to a word
- Theoretically, RNN can make use of the information in arbitrarily long sequences
  - In practice, RNN is limited to looking back only a few steps due to the vanishing gradient or exploding gradient problem

Problem of Long-Term Dependencies

- Easy to predict last word in "the clouds are in the *sky*,"
  - When gap between relevant information and place that it's needed is small, RNNs can learn to use the past information

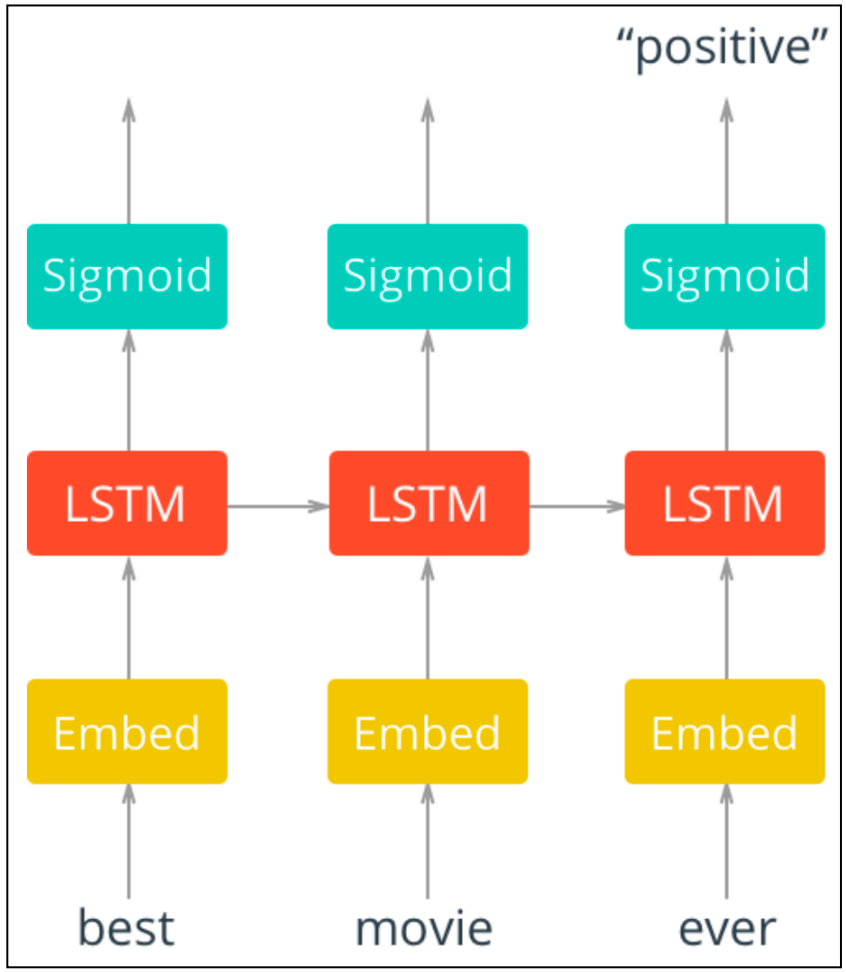


- "I grew up in France… I speak fluent *French*."
  - We need the context of France, from further back.
  - Large gap between relevant information and point where it is needed



- In principle RNNs can handle it, but fail in practice
  - LSTMs offer a solution

RNN for sentiment analysis



Embedding Layer

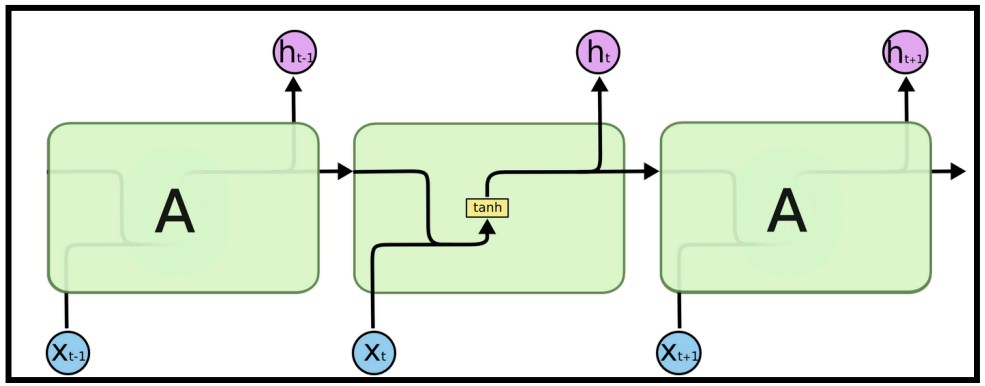
We pass in words to an embedding layer

Options

1. Actually train up an embedding with Word2vec
2. It is good enough to just have an embedding layer and let network learn the embedding table on its own.

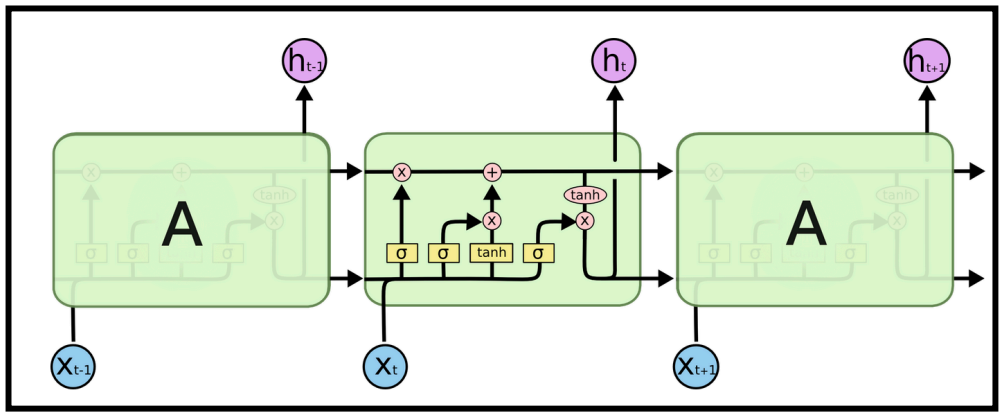
LSTM:

RNN vs LSTM



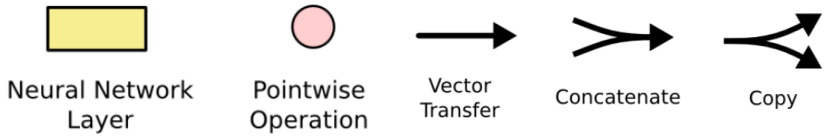
RNN

Repeating module has a simple structure
Such as a tanh layer

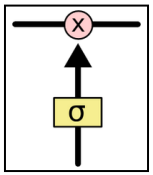


LSTM

Repeating module has four interacting Layers, with notation:

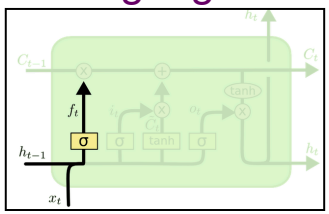


Three gates of the form



Sigmoid is 0 or 1
Allows input to go through or not

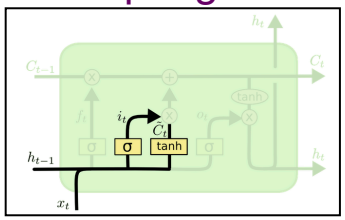
Forget gate



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

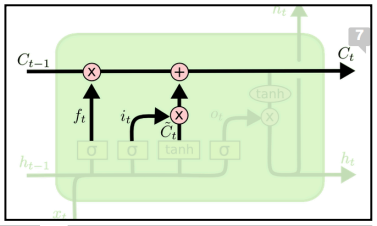
Forget gender of old subject

Input gate



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

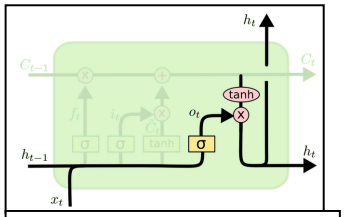
Input gender of new subject



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

Actually drop old
Add new

Output gate



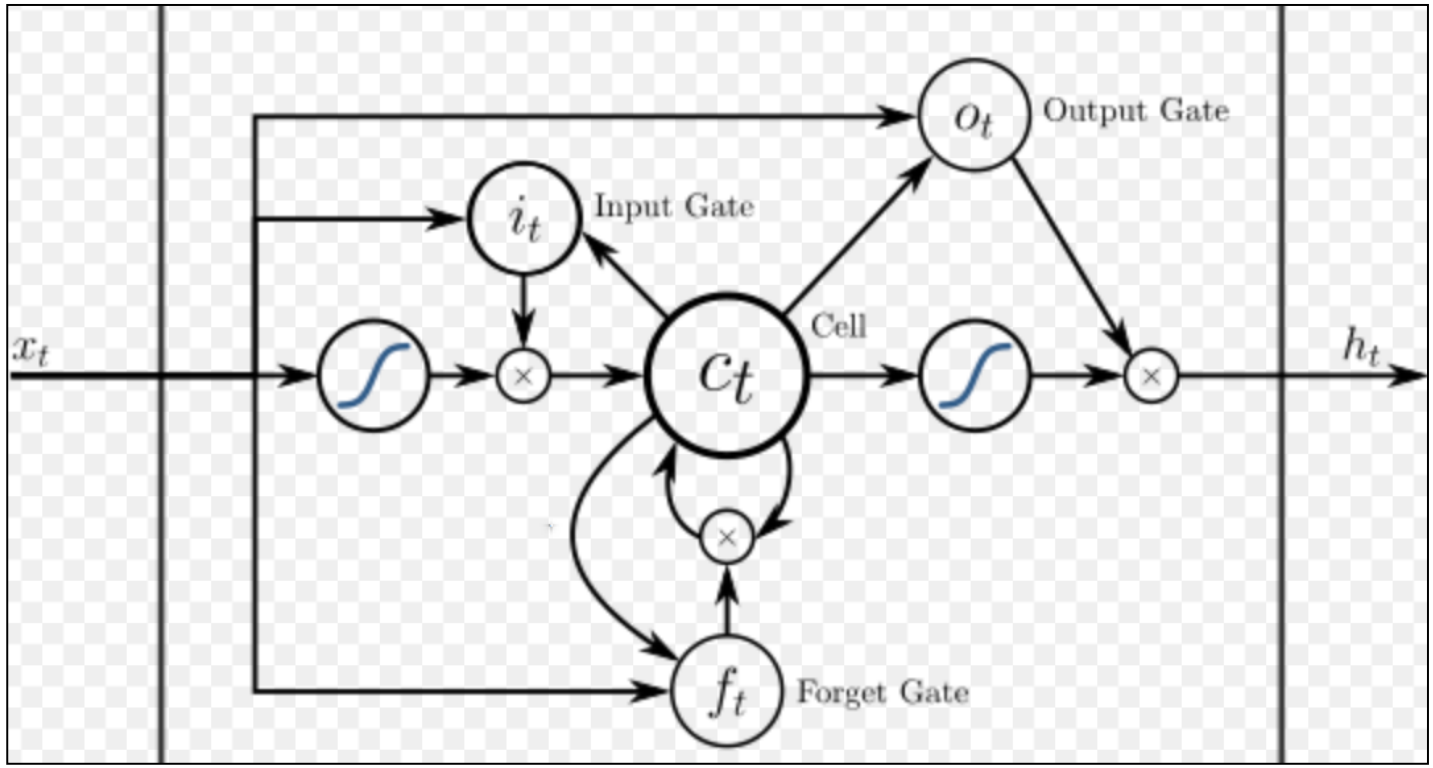
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Whether subject is singular or plural

An LSTM variant

- A common LSTM unit is composed of
  - a **cell**, an **input gate**, an **output gate** and a **forget gate**

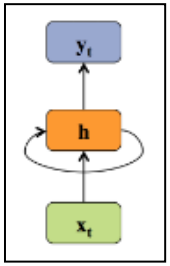
    The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell.
- A peephole LSTM is shown below



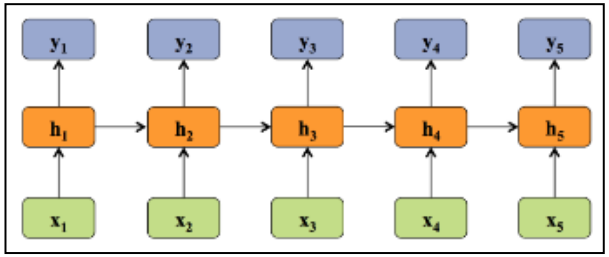
Summary of Neural Sequential Models

Recurrent Neural Network

RNN



Unrolled RNN



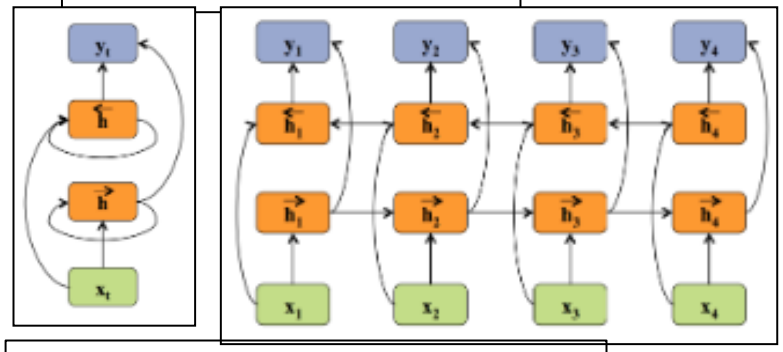
Definition

$$\begin{aligned} \text{inputs} &: x = (x_1, x_2, ..., x_T), x_i \in \mathbb{R}^I \\ \text{hidden units} &: h = (h_1, h_2, ..., h_T), h_i \in \mathbb{R}^J \\ \text{outputs} &: y = (y_1, y_2, ..., y_T), y_i \in \mathbb{R}^K \\ \text{nonlinearity} &: \mathcal{H} \end{aligned}$$

Activation Functions

$$\begin{aligned} h_t &= \mathcal{H}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \\ y_t &= W_{hy}h_t + b_y \end{aligned}$$

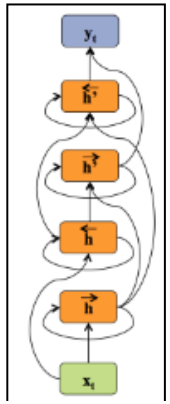
Bidirectional RNN



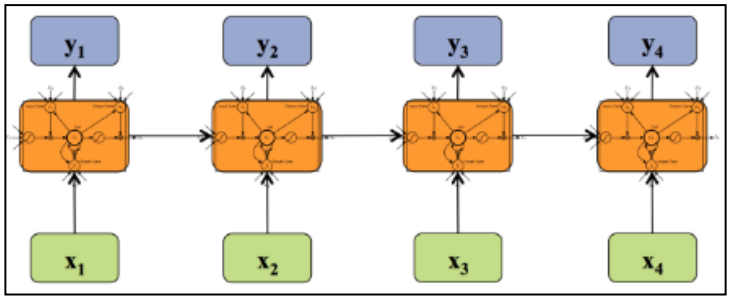
$$\begin{aligned} \text{inputs} &: x = (x_1, x_2, ..., x_T), x_i \in \mathbb{R}^I \\ \text{hidden units} &: \overrightarrow{h} \text{ and } \overleftarrow{h} \\ \text{outputs} &: y = (y_1, y_2, ..., y_T), y_i \in \mathbb{R}^K \\ \text{nonlinearity} &: \mathcal{H} \end{aligned}$$

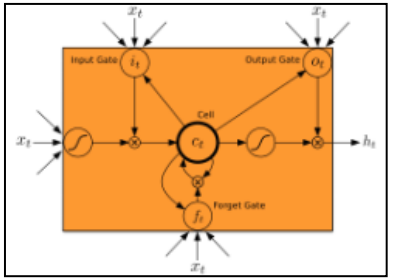
$$\begin{aligned} \overrightarrow{h}_t &= \mathcal{H}(W_{x\overrightarrow{h}}x_t + W_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}) \\ \overleftarrow{h}_t &= \mathcal{H}(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t-1} + b_{\overleftarrow{h}}) \\ y_t &= W_{\overleftarrow{h}y}\overleftarrow{h}_t + W_{\overrightarrow{h}y}\overrightarrow{h}_t + b_y \end{aligned}$$

Deep Bidirectional RNN



LSTM





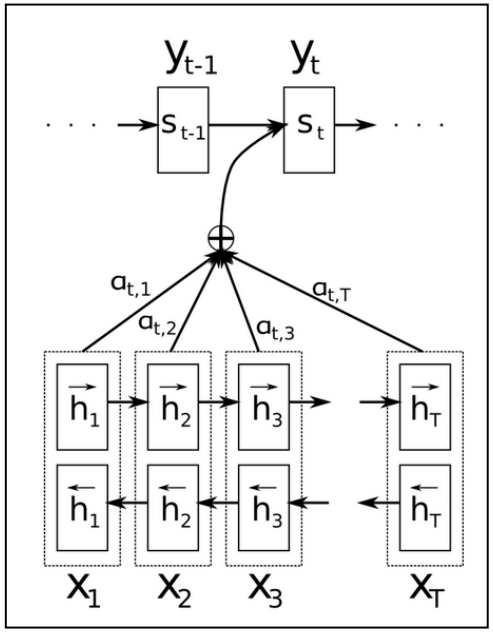
Source: http://www.cs.cmu.edu/~epxing/Class/10708-16/note/10708_scribe_lecture27.pdf

Attention Mechanisms

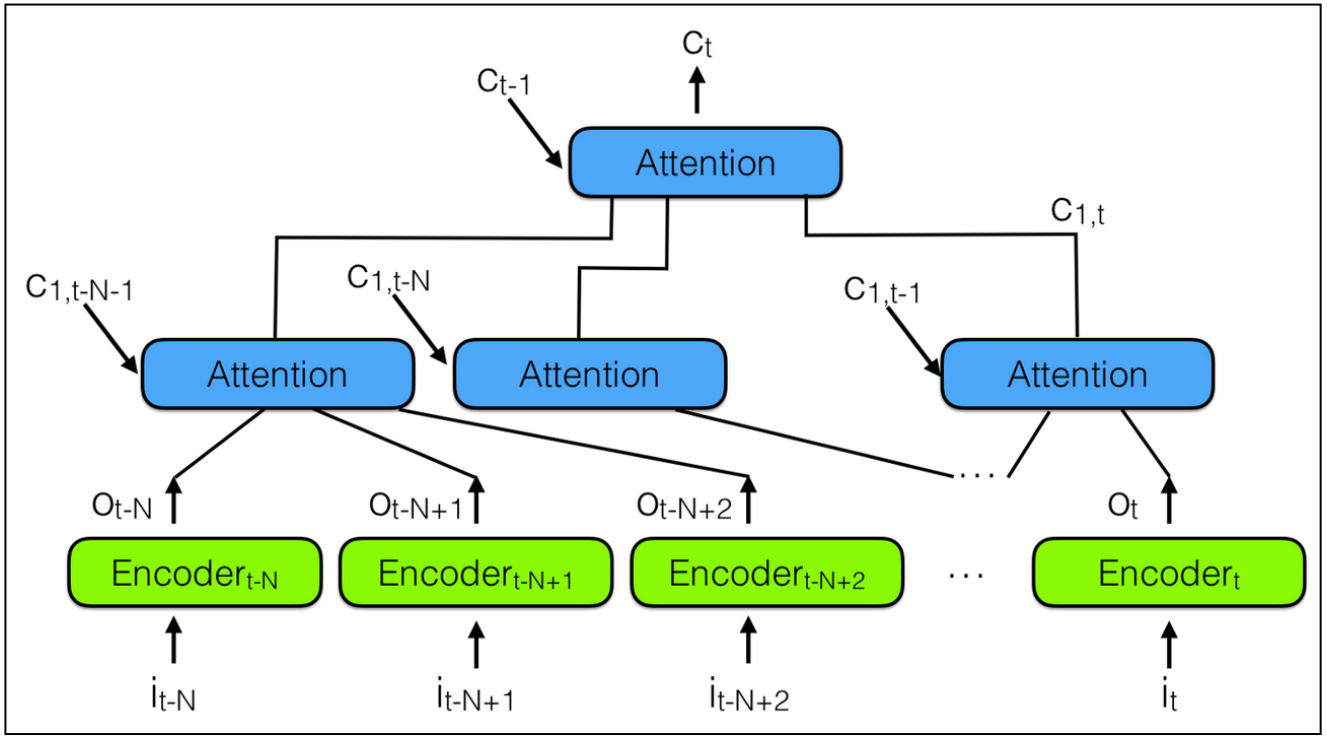
- Long range dependencies are still a problem with LSTMs
- With an attention mechanism we no longer try encode the full source sentence into a fixed-length vector
  - Rather, we allow the decoder to "attend" to different parts of the source sentence at each step of the output generation
- Attention is simply a vector, often the outputs of dense layer using softmax function.

Attention Mechanism in Translation

- y : translated words
- x : source words.
  - Use of bidirectional RNN is unimportant
- Each decoder output word now depends on
  - weighted combination of all the input states,
    - not just the last state.
  - a's are weights for each input state
    - if $a_{3,2}$ is large, decoder pays attention to second state in source sentence while producing third word of target
    - a's are normalized to sum to 1



Hierarchical neural attention encoder

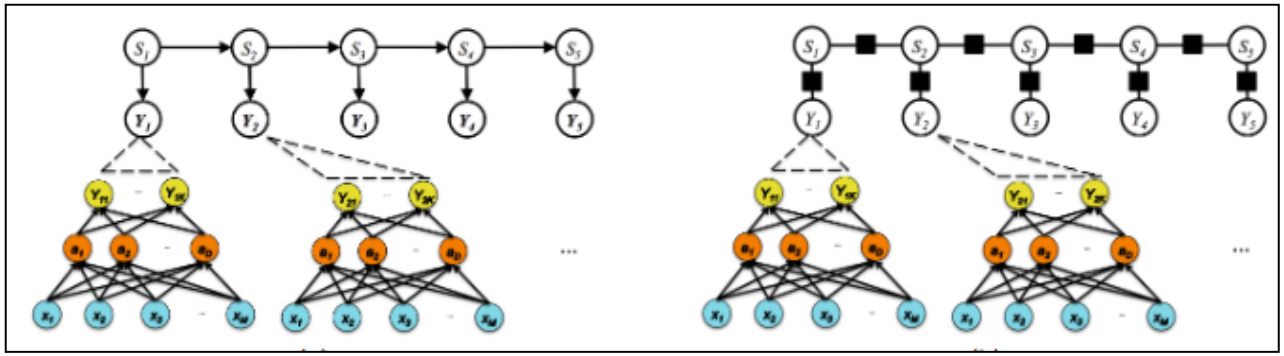


Deep Learning and Graphical Models

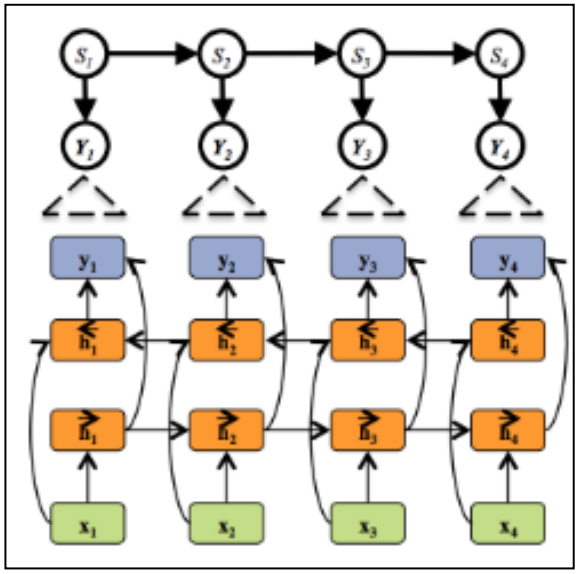
- In deep learning:
  - Tasks of interest:
    - Classification
    - Feature learning
  - Method of learning
    - Backpropagation and gradient descent
- In graphical models:
  - Tasks of interest:
    - Transfer learning
    - Latent variable inference
  - Methods of learning
    - Parameter learning methods
    - Structure learning methods
- Hybrid graphical models combine the two types of models
  - They are trained using backpropagation

Hybrid Graphical Models and Neural Networks

- Hybrid NN and HMM



- Hybrid RNN+HMM



Hybrid CNN+CRF