RNN, GRU, LSTM, and some NLP

Chantana chantrapornchai

HPCNC

NVIDIA GPU EDUCATION CENTER

NVIDIA DEEP LEARNING INSTITUTE

KU KASETSART UNIVERSITY

References

- https://docs.google.com/presentation/d/1UHXrKL1oTdgMLoAHHPfMM_srDO0BCyJXPmhe4DNh_G8/pub?start=false&loop=false&delayms=3000&slide=id.g24de73a70b_0_0
- http://slazebni.cs.illinois.edu/spring17/lec02_rnn.pdf
- http://slazebni.cs.illinois.edu/spring17/lec03_rnn.pdf
- https://www.coursehero.com/file/27811104/Deep-Learning-2017-Lecture6RNNppt/

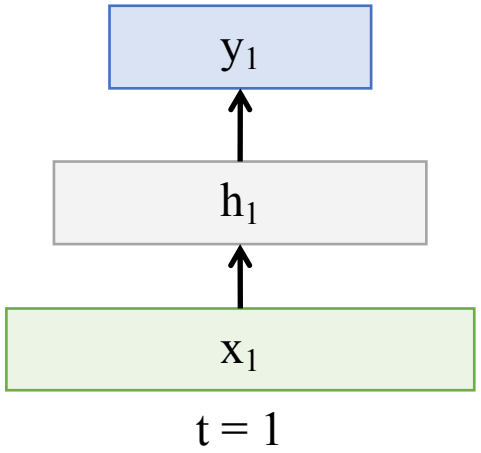
About all the three…

- LSTM and GRU are widely used. GRU is the most intuitive one.
- LSTM with peephole connection was designed to attacked basic LSTM problem, ie., loss of information.

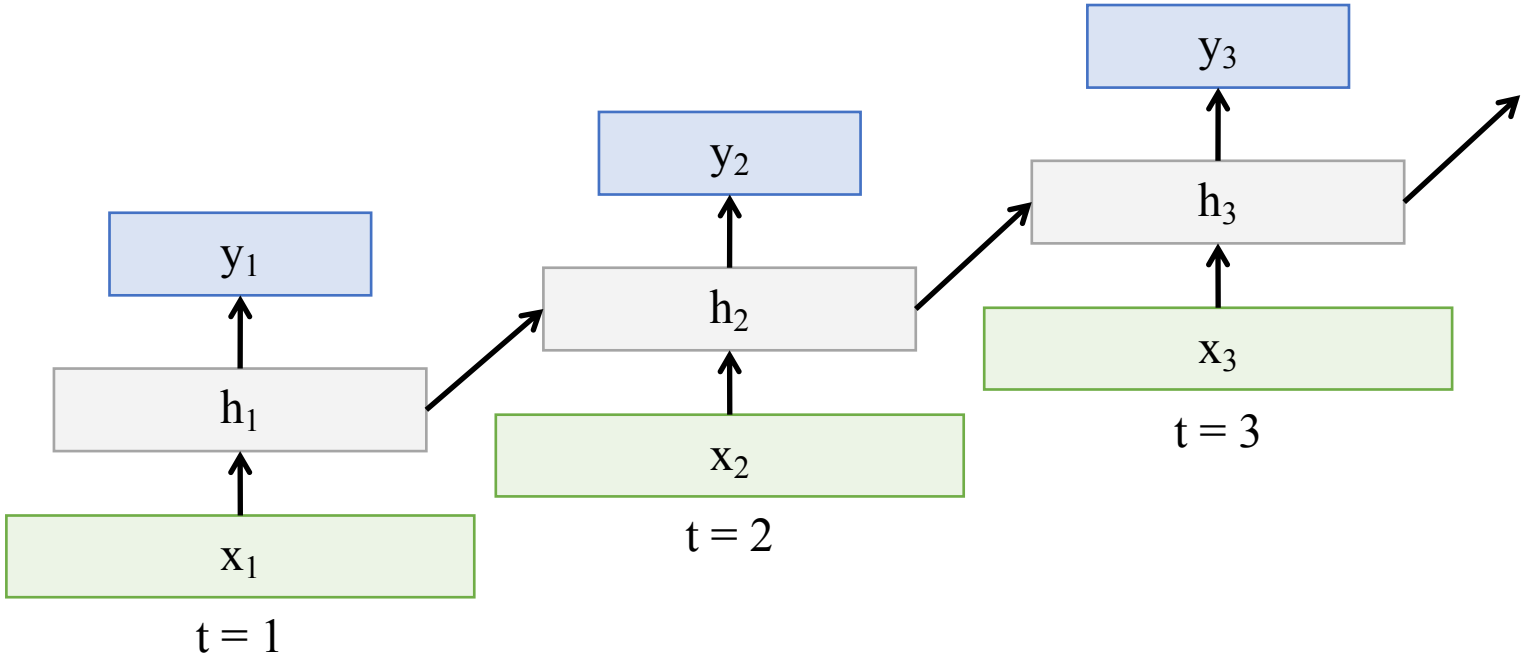
Recurrent Neural Networks: Motivtion

- Speech Recognition or Time-series Prediction require **a system to store and use context information**
  - Simple case: Output YES if the number of 1s is even, else NO
    1000010101 – YES, 100011 – NO, …
  - Like state machine….
- How to choose a fixed context window?
- Recurrent Neural Networks take the previous output or hidden states as inputs.
  The composite input at time t has some historical information about the happenings at time T < t
  - RNNs are useful as their intermediate values (state) can store information about past inputs for a time that is not fixed a priori

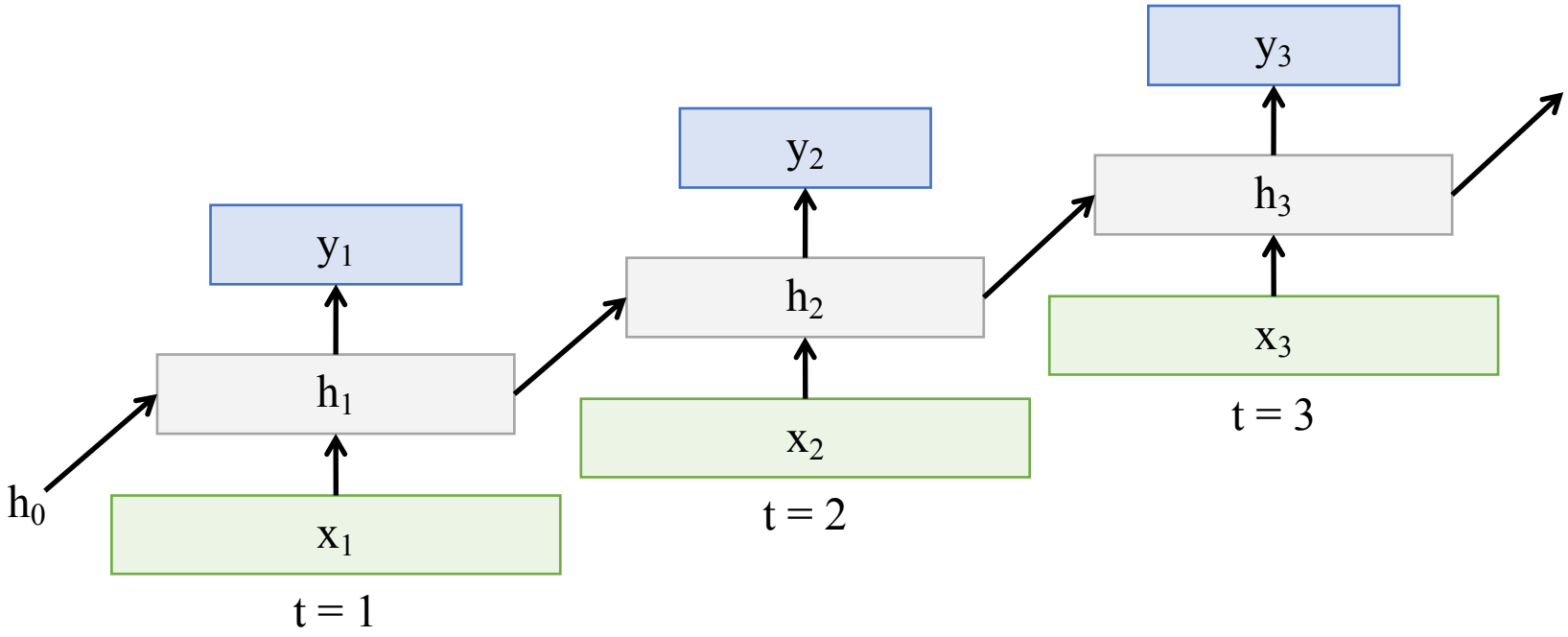
Sample Feed-forward Network: one layer



Sample RNN

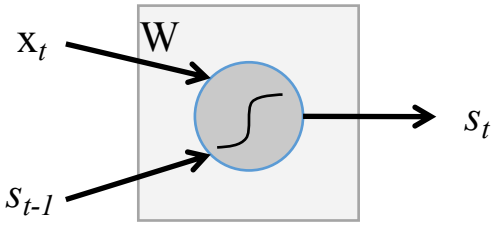


Sample RNN



RNN Cell

- Vanilla RNN is just a single layer network cell.



$$(n \times 1) \quad (n \times n)(n \times 1) \quad (n \times m)(m \times 1) \quad (n \times 1)$$

$$s_t = \varphi(Ws_{t-1} + Ux_t + b)$$

$$s_t = \tanh W \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix}$$

Vanilla RNN

Let s_t be current state
s_{t-1} be previous state

$n = 2$ (state size),
$m = 3$ (input size)

$$s_t = \varphi(W s_{t-1} + U x_t + b)$$

$$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix} = \varphi\left(\begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \end{pmatrix} + \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \end{pmatrix} \begin{pmatrix} x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right)$$

$$= \varphi\left(\begin{pmatrix} w_{11} s_{t-1}^1 + w_{12} s_{t-1}^2 \\ w_{21} s_{t-1}^1 + w_{22} s_{t-1}^2 \end{pmatrix} + \begin{pmatrix} u_{11} x_t^1 + u_{12} x_t^2 + u_{13} x_t^3 \\ u_{21} x_t^1 + u_{22} x_t^2 + u_{23} x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right)$$

$$= \varphi\left(\begin{pmatrix} w_{11} s_{t-1}^1 + w_{12} s_{t-1}^2 + u_{11} x_t^1 + u_{12} x_t^2 + u_{13} x_t^3 \\ w_{21} s_{t-1}^1 + w_{22} s_{t-1}^2 + u_{21} x_t^1 + u_{22} x_t^2 + u_{23} x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right)$$

$$= \varphi\left(\begin{pmatrix} w_{11} & w_{12} & u_{11} & u_{12} & u_{13} \\ w_{21} & w_{22} & u_{21} & u_{22} & u_{23} \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right)$$

- single layer network

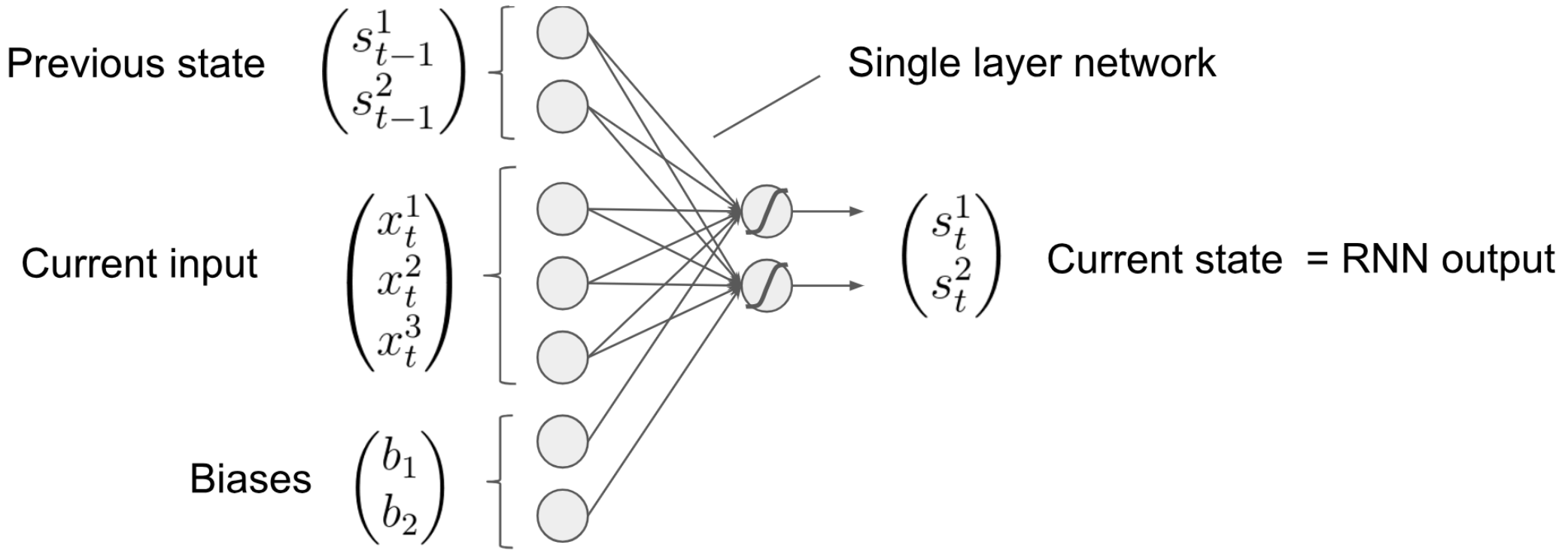
Some math

Vanilla RNN

$$s_t = \varphi(W_c[s_{t-1}, x_t] + b)$$

$$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix} = \varphi\left(\begin{pmatrix} w_{11} & w_{12} & u_{11} & u_{12} & u_{13} \\ w_{21} & w_{22} & u_{21} & u_{22} & u_{23} \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ x_t^1 \\ x_t^2 \\ x_t^3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right)$$

How it looks?



Vanilla RNN

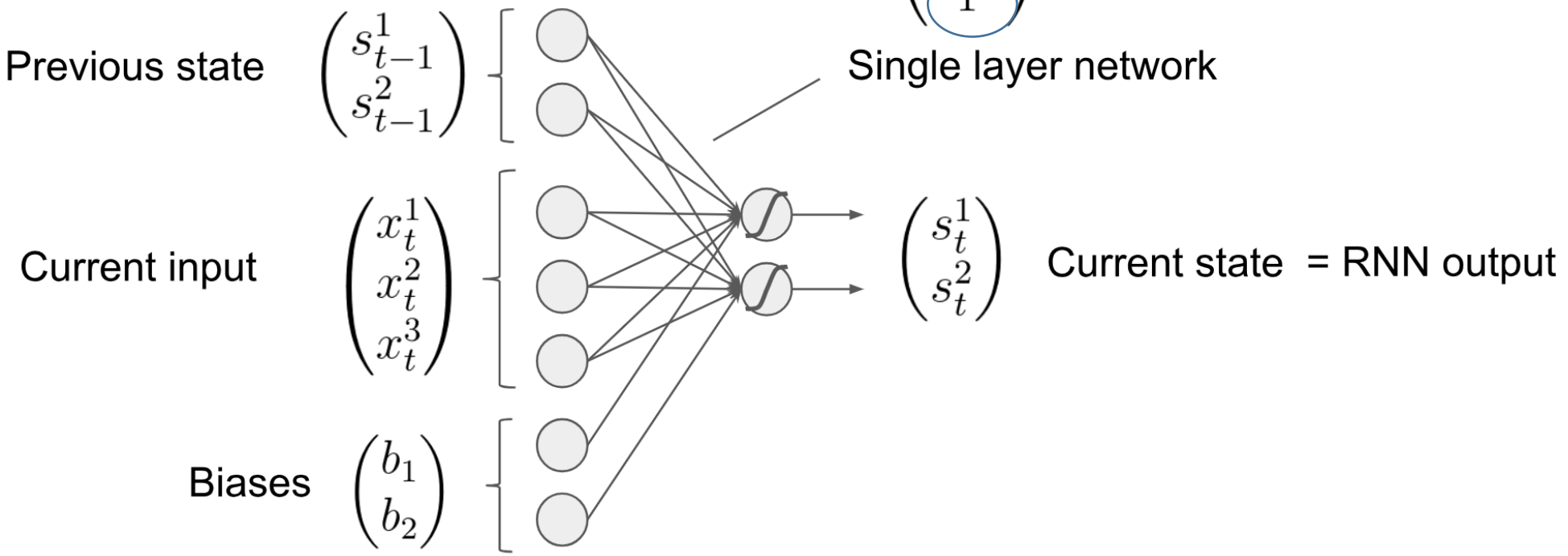
b=1

$$s_t = \varphi(W_b[s_{t-1}, x_t, 1])$$

$$s_t = \varphi(W_b[s_{t-1}, x_t])$$

But often “1” is omitted

$$\begin{pmatrix} s_t^1 \\ s_t^2 \end{pmatrix} = \varphi\left(\begin{pmatrix} w_{11} & w_{12} & u_{11} & u_{12} & u_{13} & b_1 \\ w_{21} & w_{22} & u_{21} & u_{22} & u_{23} & b_2 \end{pmatrix} \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ x_t^1 \\ x_t^2 \\ x_t^3 \\ 1 \end{pmatrix} \right)$$



Note…

$$s_t = \varphi(W s_{t-1} + U x_t + b)$$

sometimes written as

$$s_t = \varphi(W_c[s_{t-1}, x_t] + b)$$

sometimes as

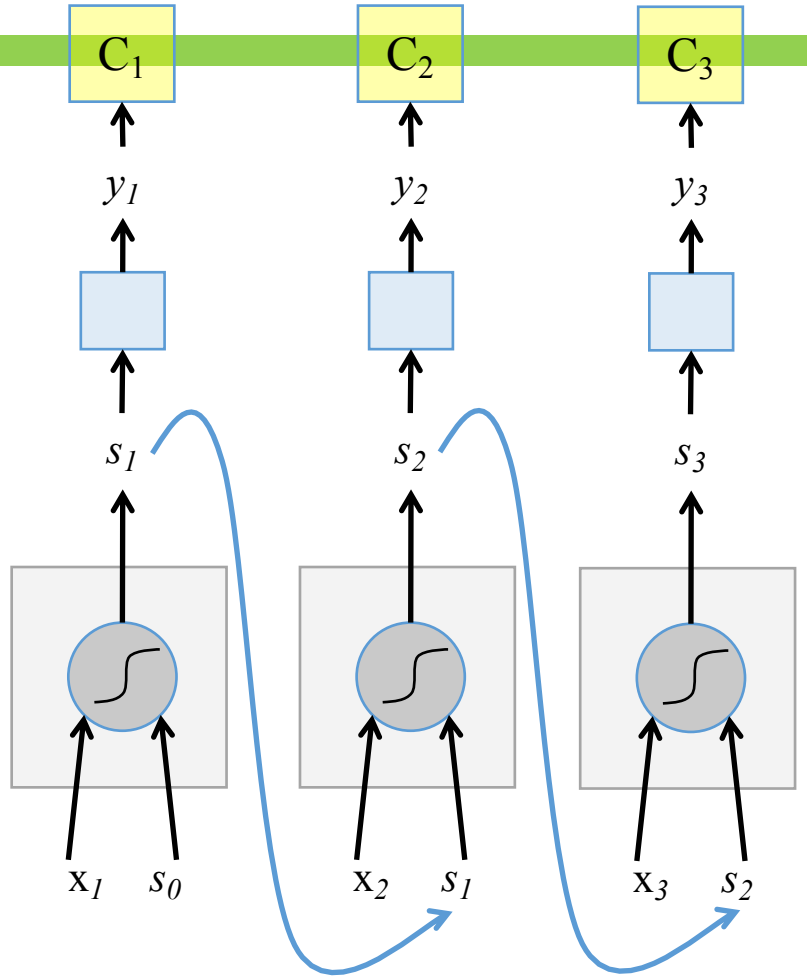
$$s_t = \varphi(W_b[s_{t-1}, x_t])$$

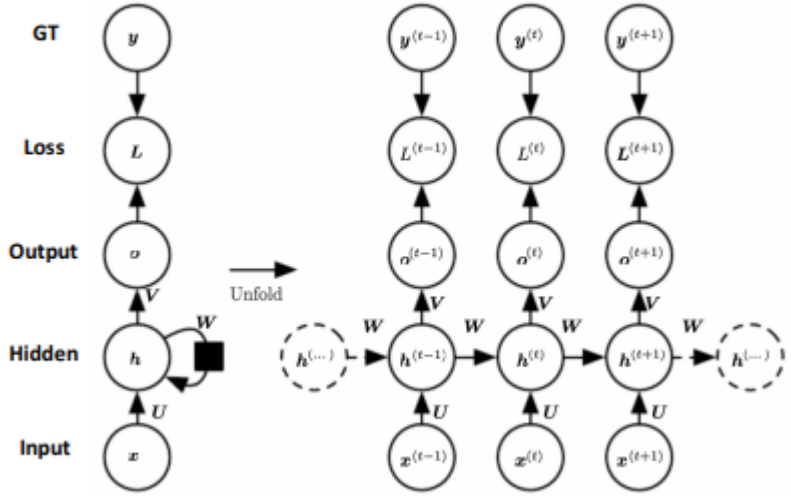
Feed forward RNN

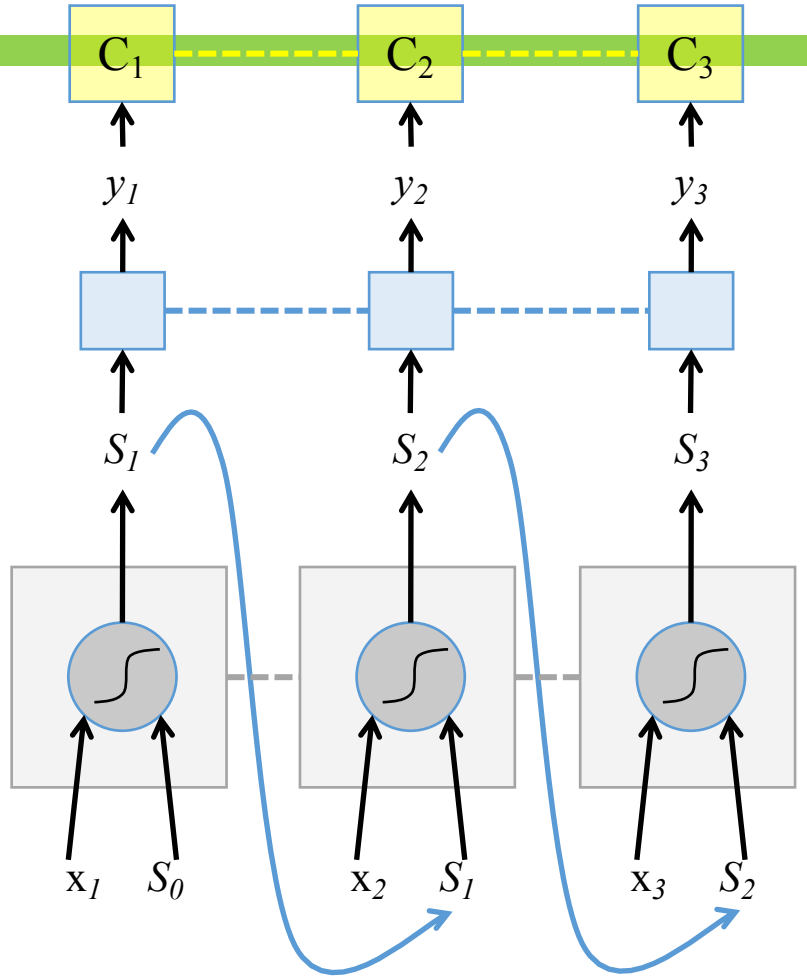
$$s_t = \tanh W \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix}$$

$$y_t = F(s_t)$$

$$C_t = Loss(y_t, GT_t)$$







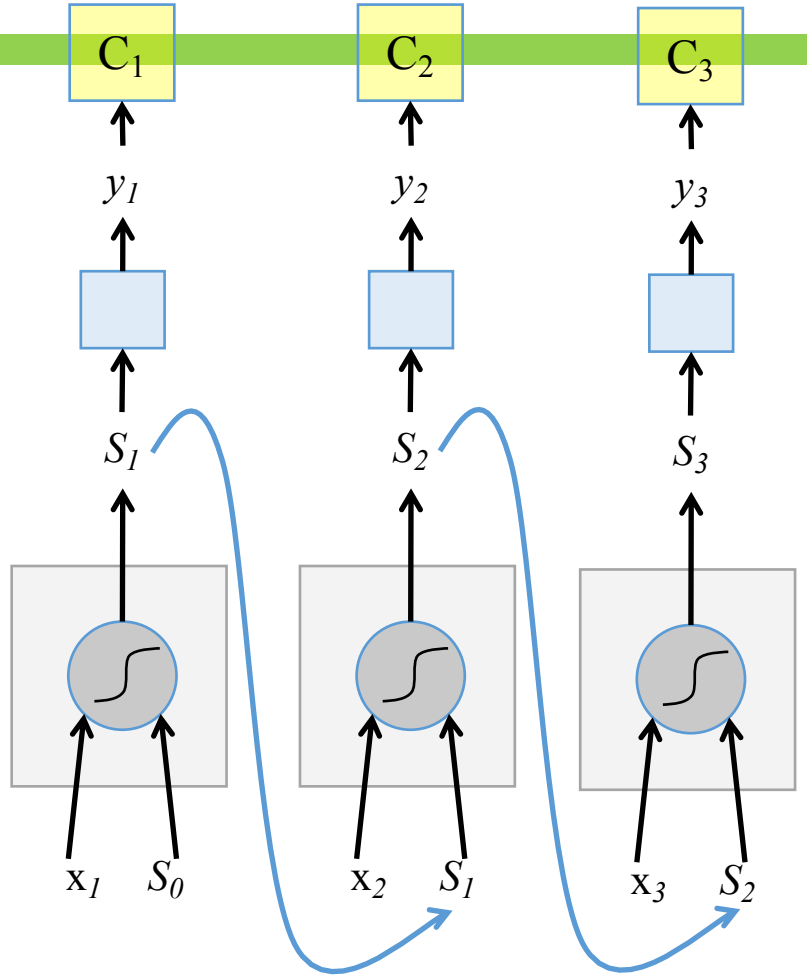
$$s_t = \tanh W \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix}$$

$$y_t = F(s_t)$$

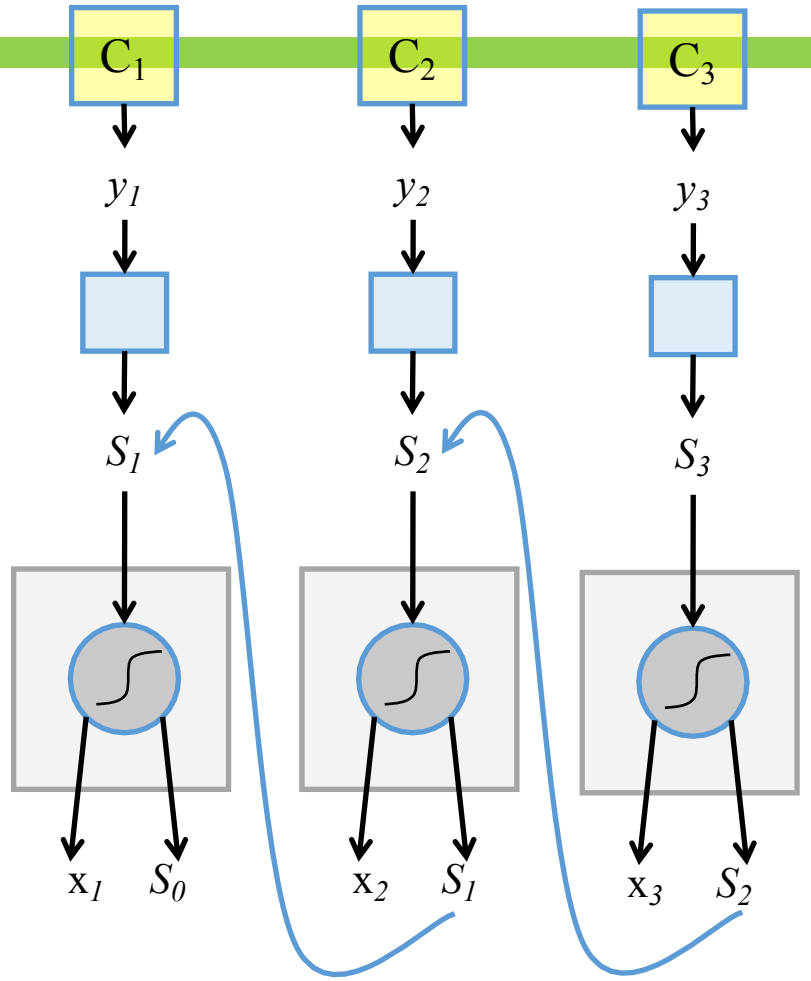
$$C_t = Loss(y_t, GT_t)$$

------ indicates shared weights

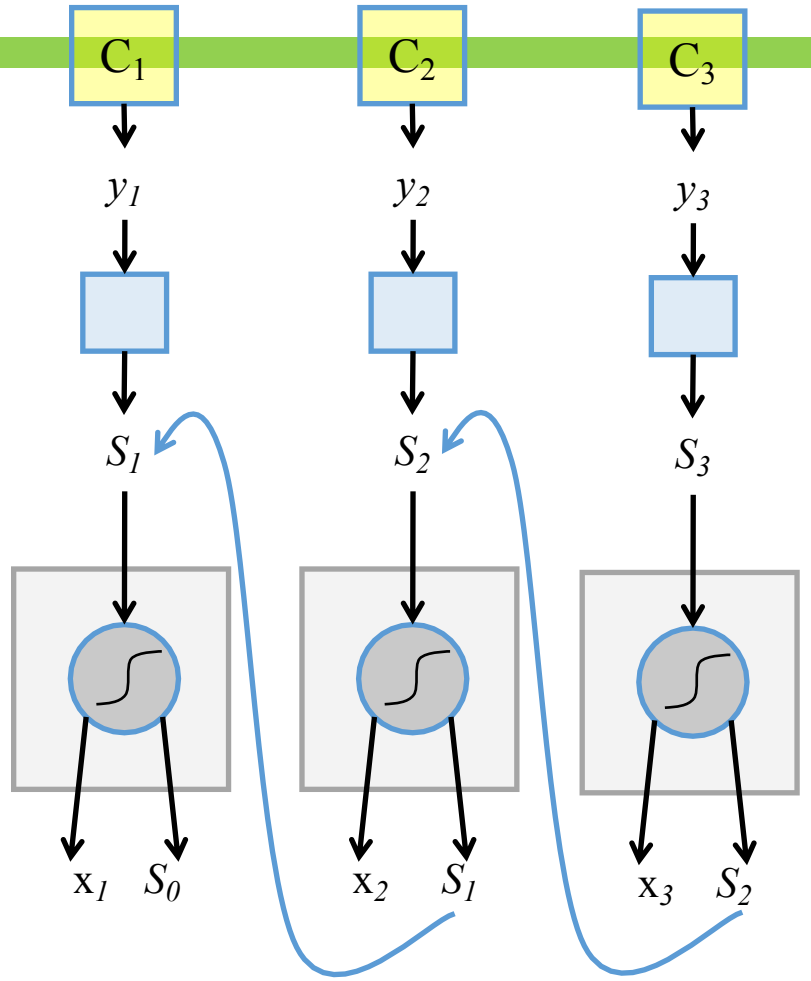
The Unfolded Vanilla RNN Forward



The Unfolded Vanilla RNN Backward



The Vanilla RNN Backward



$$s_t = \tanh W \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix}$$

$$y_t = F(s_t)$$

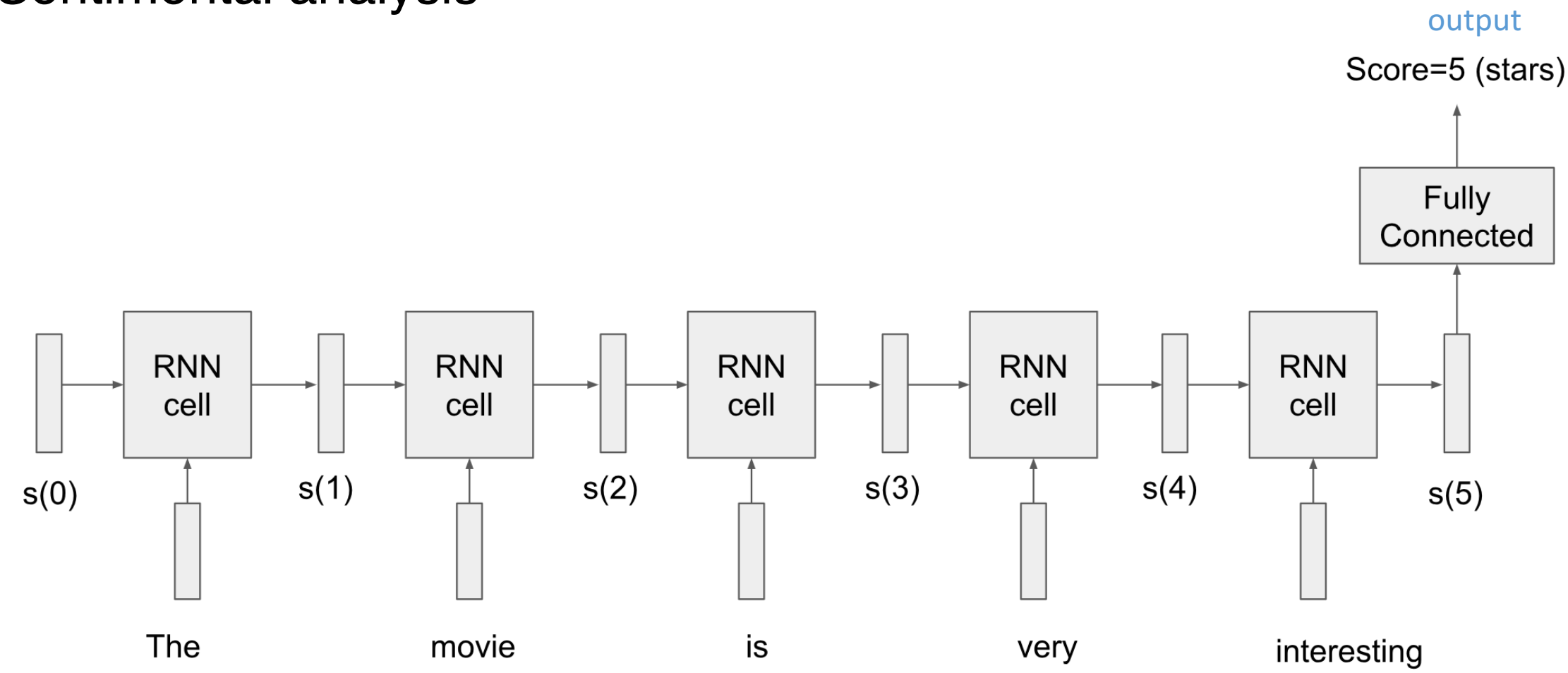
$$C_t = Loss(y_t, GT_t)$$

$$\frac{\partial C_t}{\partial S_1} = \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial S_1}\right)$$

$$= \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial S_t}\right)\left(\frac{\partial S_t}{\partial S_{t-1}}\right)\cdots\left(\frac{\partial S_2}{\partial S_1}\right)$$

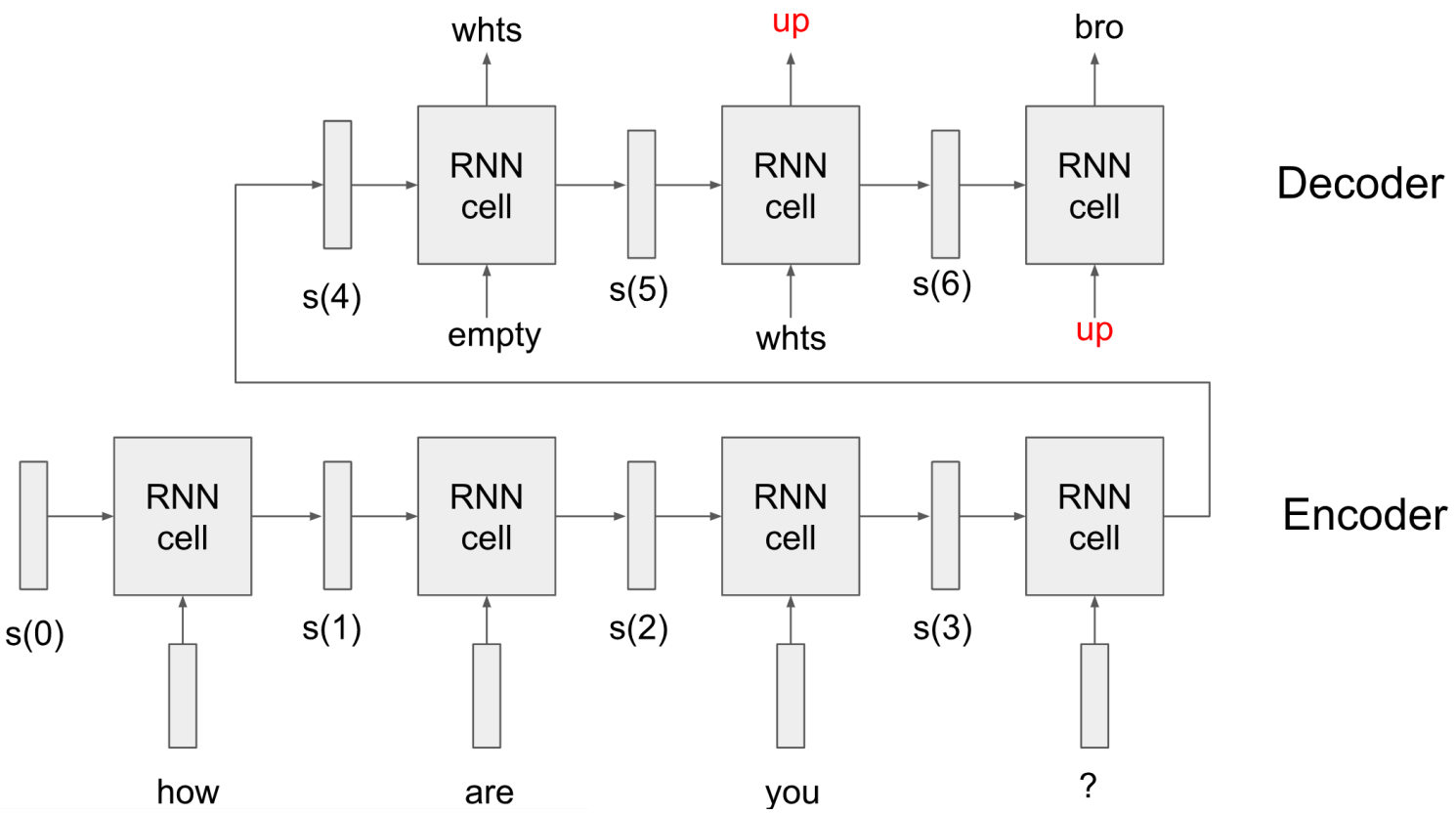
RNN Example

- Sentimental analysis



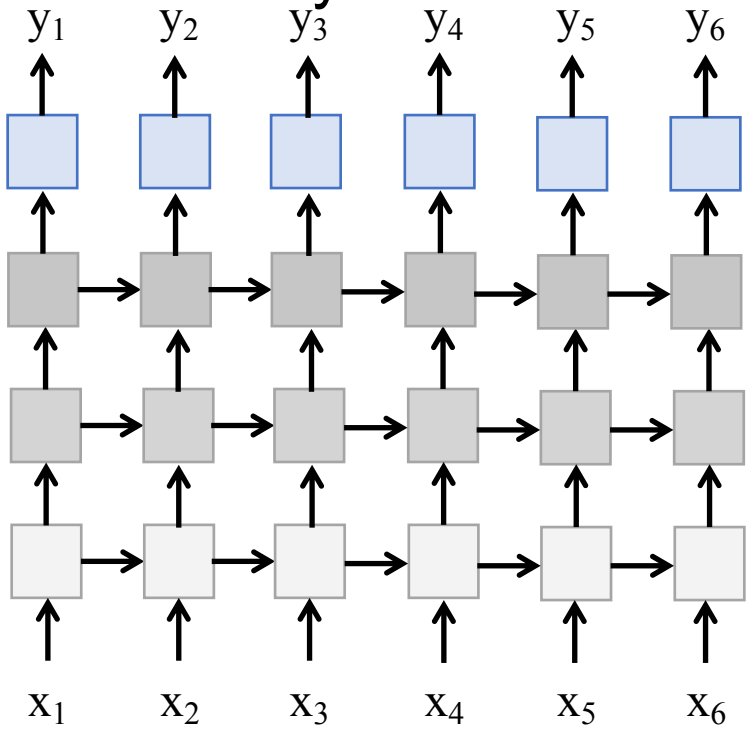
RNN Example

- English to slang



Multi-layer RNNs

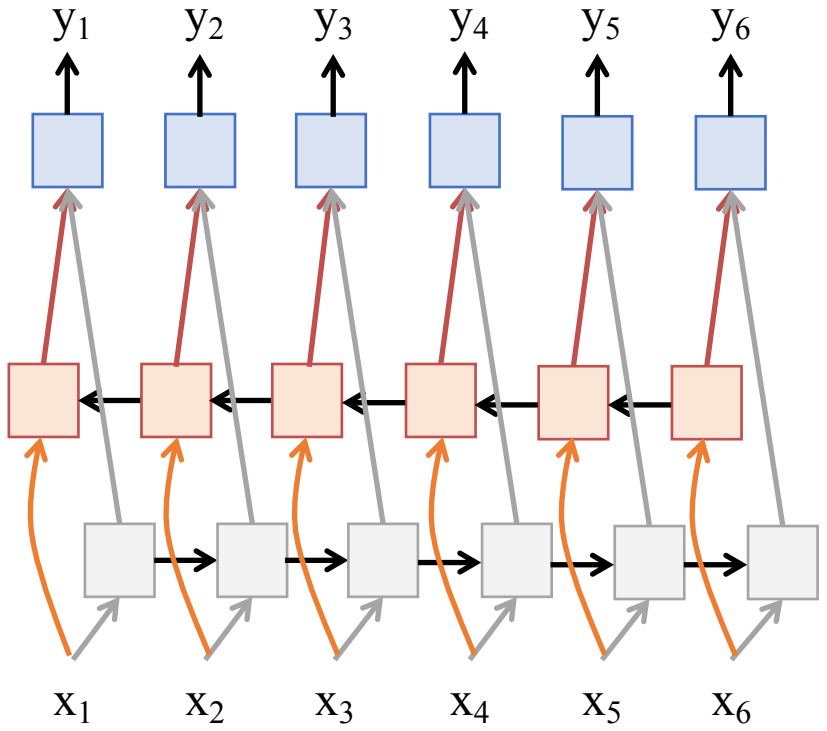
- Design RNNs with multiple hidden layers



  - Think exotic: Skip connections across layers, across time, …

Bi-directional RNNs

- RNNs can process the input sequence in forward and in the reverse direction



- Popular in speech recognition

Problems with RNN

- Information morphing
  - Inability to keep content more than certain time steps
- Gradient vanishing
- Exploding gradient

[1] On the difficulty of training recurrent neural networks, Pascanu *et al*., 2013

Information morphing

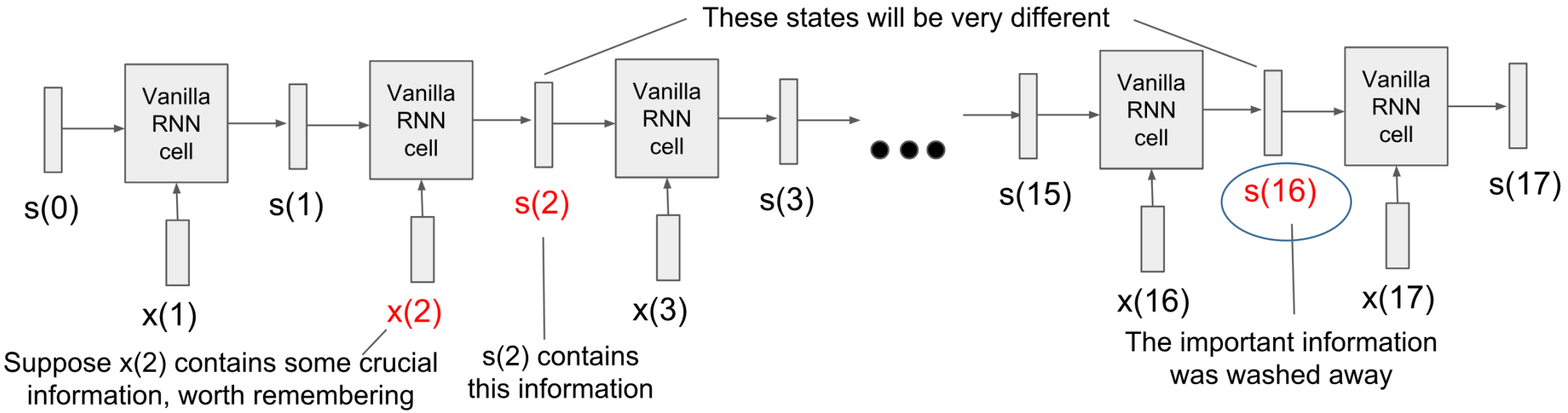
$$s_t = \varphi(W s_{t-1} + U x_t + b)$$

Nonlinearity is bad for long term
No selectivity

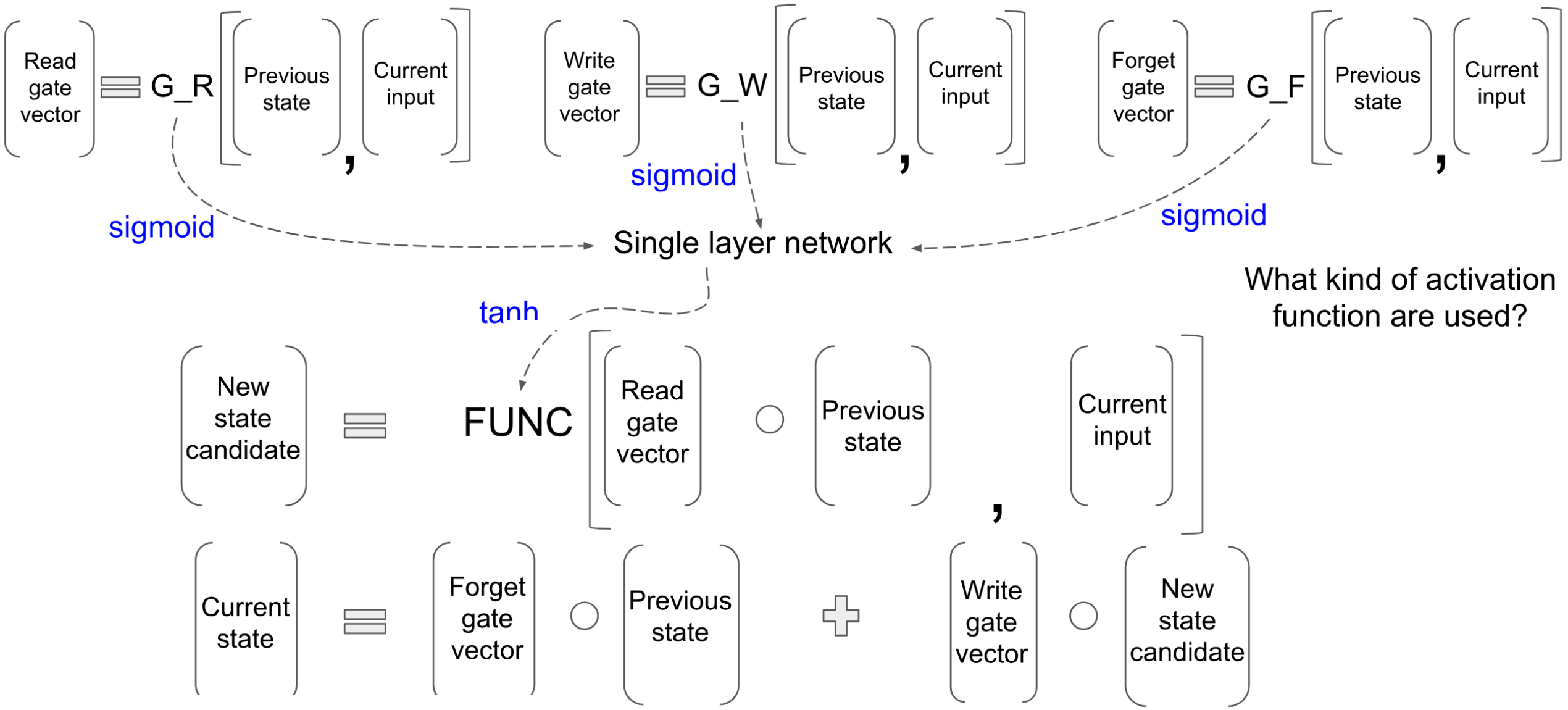
Why this happen?

RNN memory should be protected (+ or – for write operation)

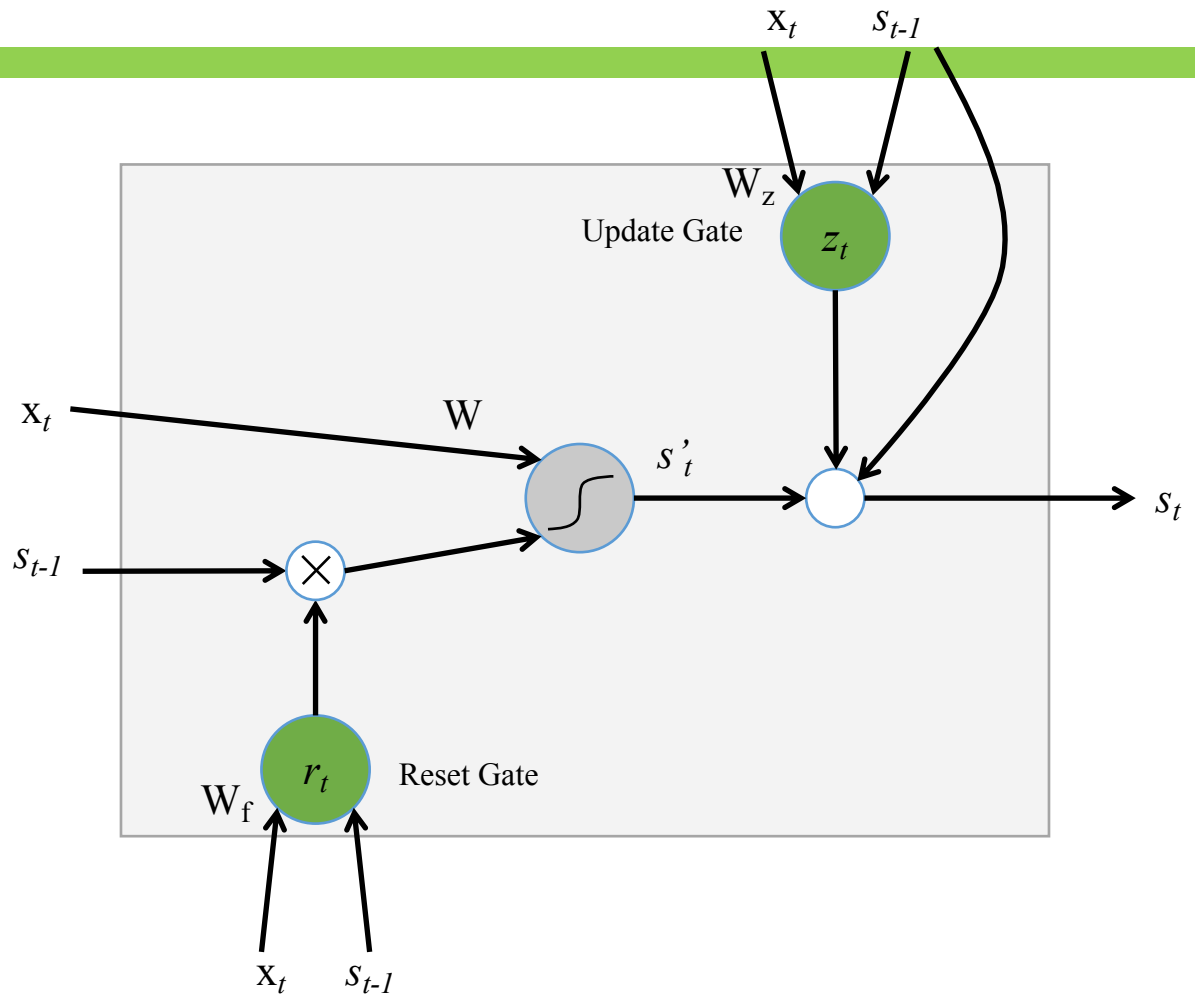
Be selective: choose what to write.



Evolution to …


Read gate vector
G_R
Previous state
Current input
Write gate vector
G_W
Previous state
Current input
Forget gate vector
G_F
Previous state
Current input
sigmoid
sigmoid
sigmoid
Single layer network
What kind of activation function are used?
tanh
New state candidate
FUNC
Read gate vector
Previous state
Current input
Current state
Forget gate vector
Previous state
Write gate vector
New state candidate

GRU—Gated Recurrent Unit



$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix} + b_f\right)$$

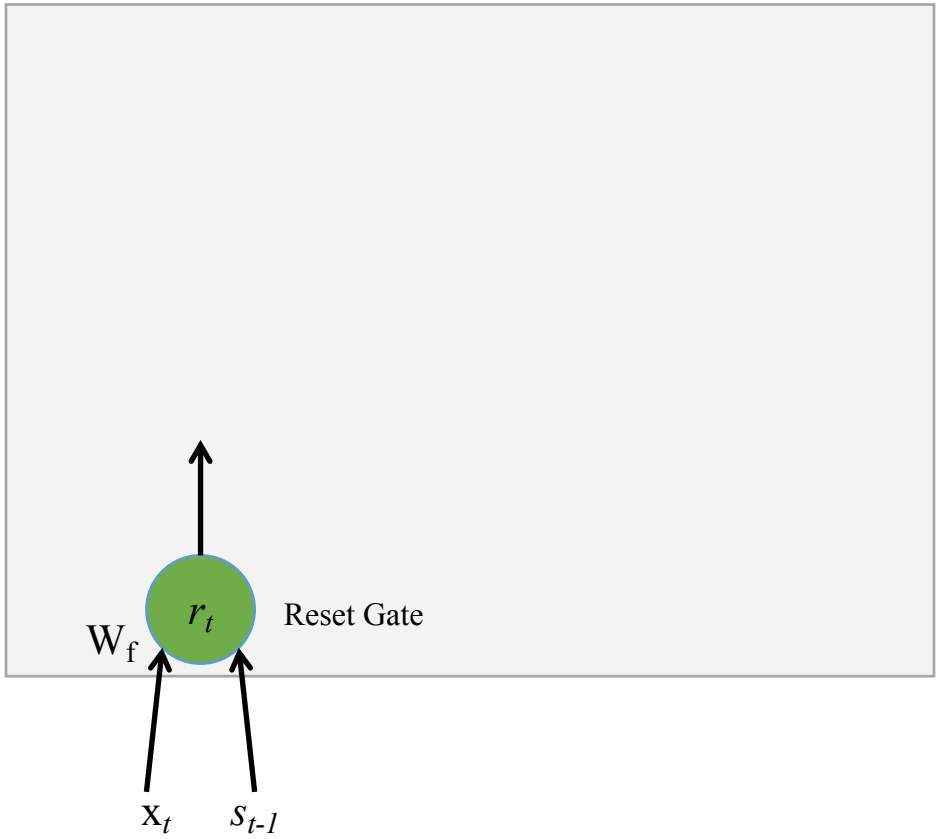
$$s'_t = \tanh W \begin{pmatrix} x_t \\ r_t \otimes s_{t-1} \end{pmatrix}$$

$$z_t = \sigma\left(W_z \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix} + b_f\right)$$

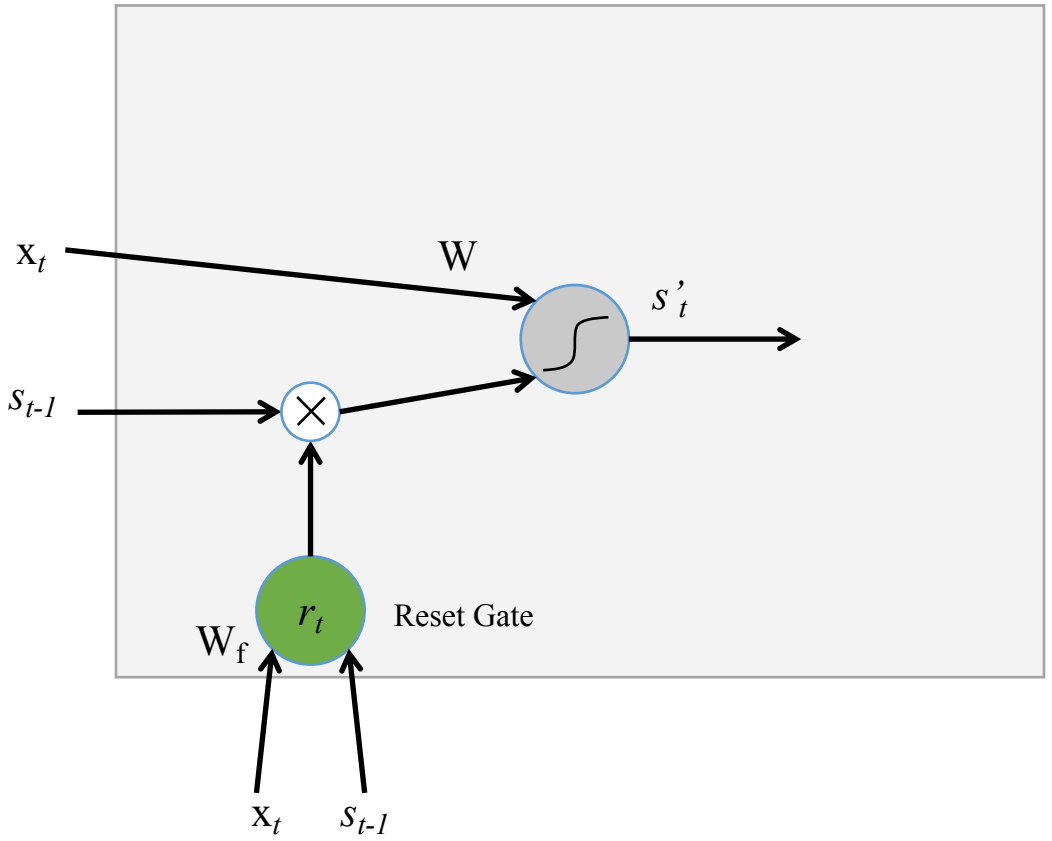
$$s_t = (1 - z_t) \otimes s_{t-1} + z_t \otimes s'_t$$

GRU

$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix} + b_f\right)$$



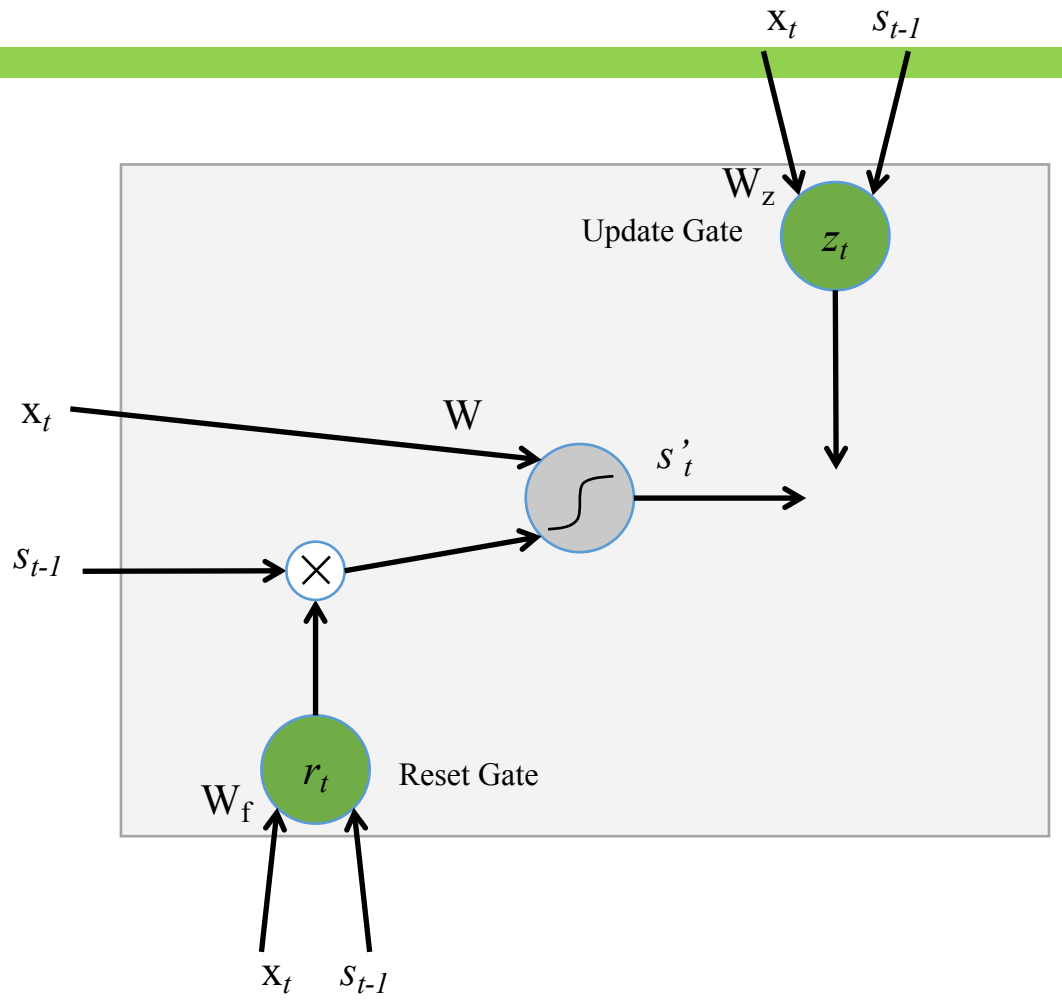
GRU



$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix} + b_f\right)$$

$$s'_t = \tanh W \begin{pmatrix} x_t \\ r_t \otimes s_{t-1} \end{pmatrix}$$

GRU

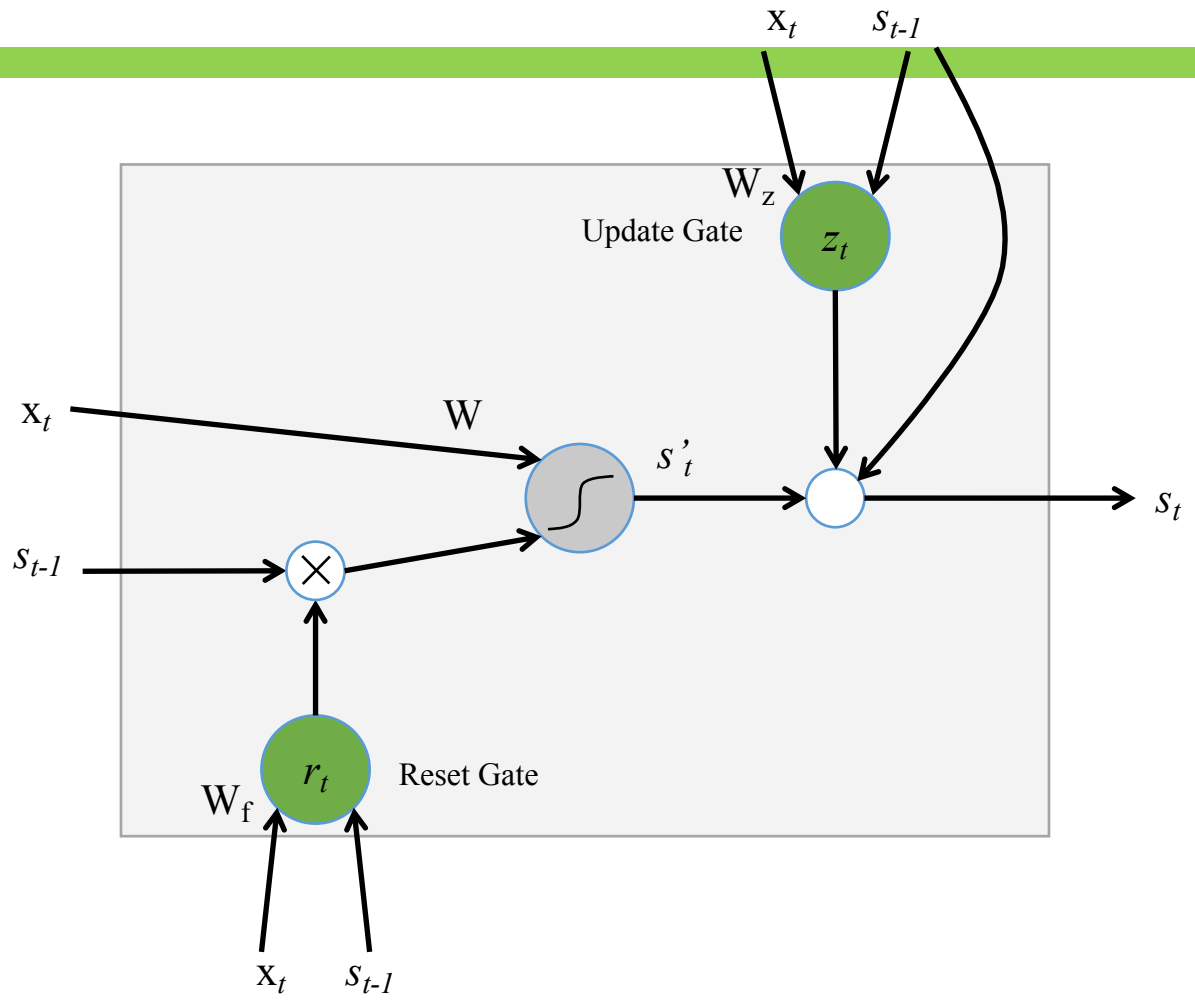


$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ S_{t-1} \end{pmatrix} + b_f\right)$$

$$S'_t = \tanh W \begin{pmatrix} x_t \\ r_t \otimes S_{t-1} \end{pmatrix}$$

$$z_t = \sigma\left(W_z \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix} + b_f\right)$$

GRU



$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix} + b_f\right)$$

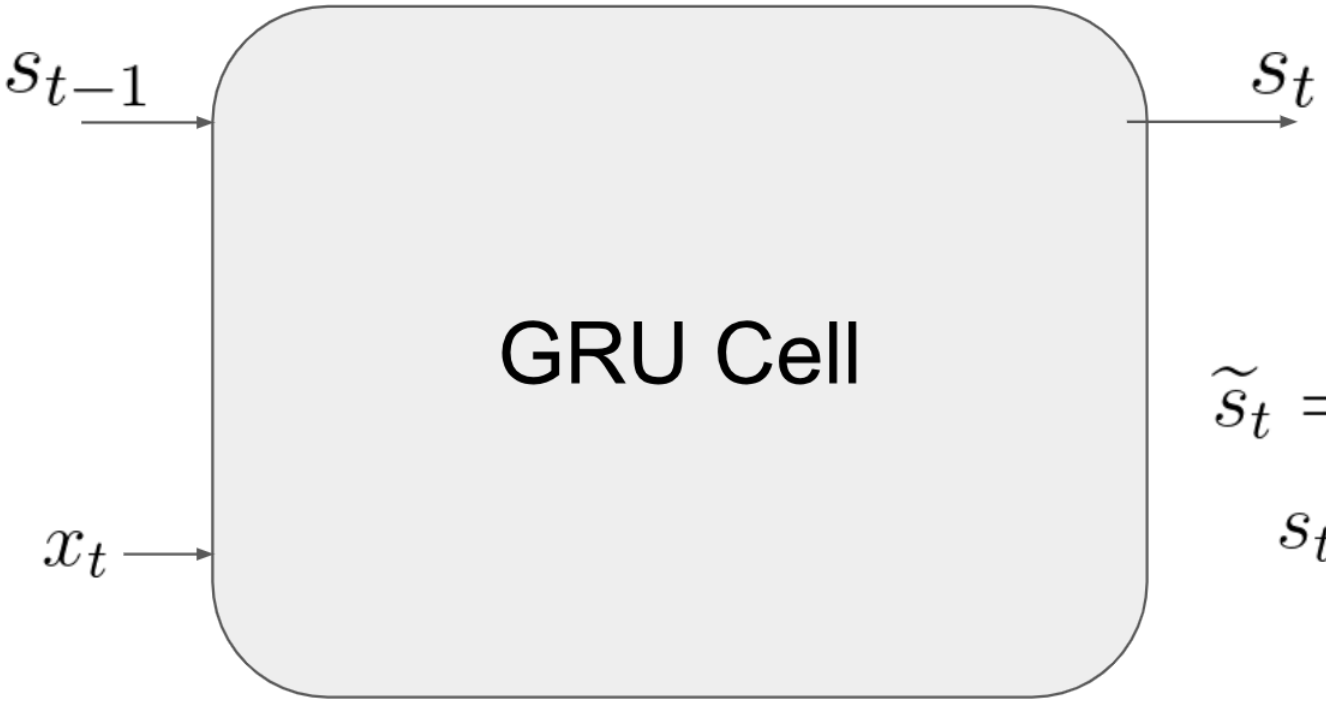
$$s'_t = \tanh W \begin{pmatrix} x_t \\ r_t \otimes s_{t-1} \end{pmatrix}$$

$$z_t = \sigma\left(W_z \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix} + b_f\right)$$

$$s_t = (1 - z_t) \otimes s_{t-1} + z_t \otimes s'_t$$

GRU

https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be



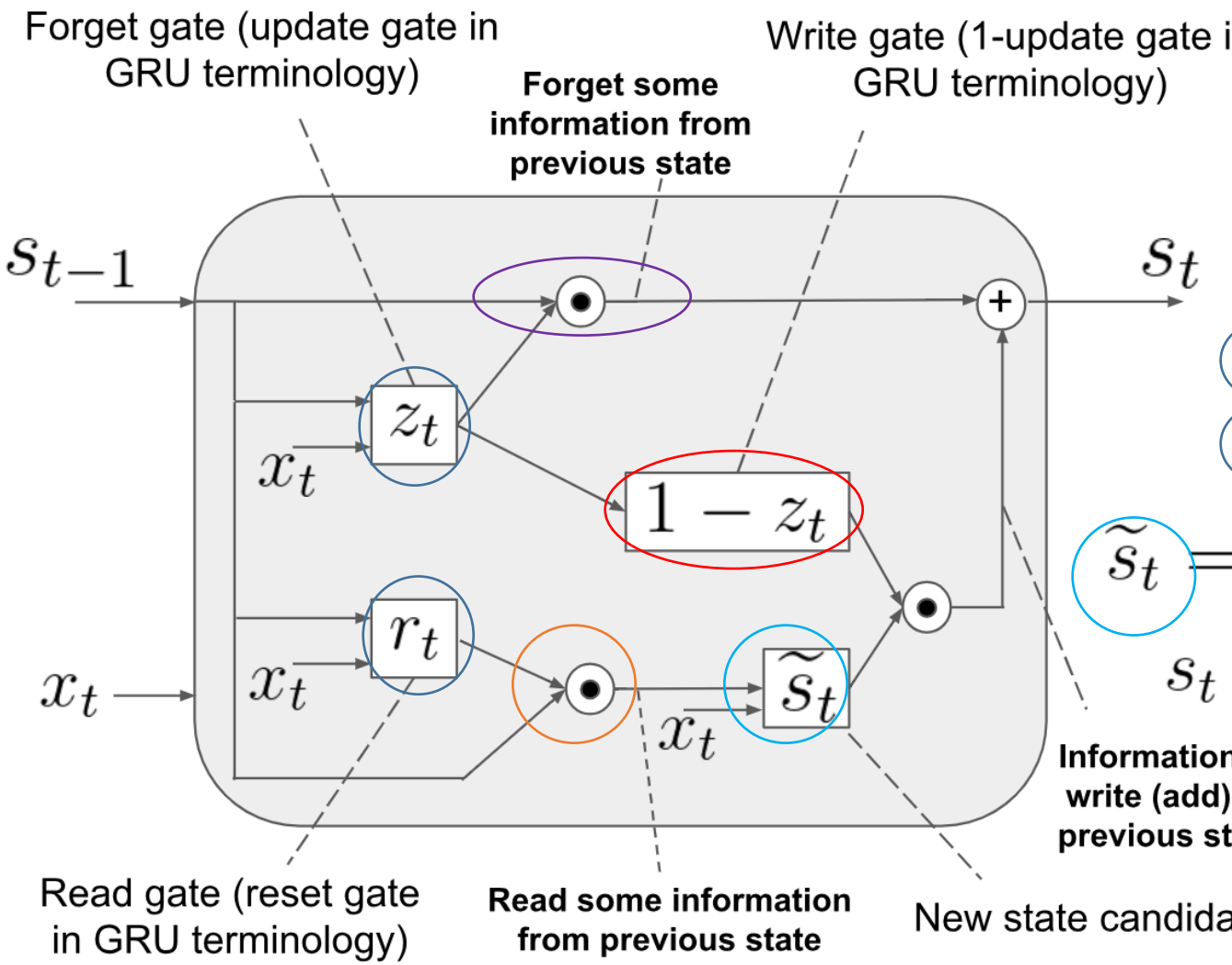
GRU equations

$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$ Read gate

$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$$ Forget gate

$$\widetilde{s}_t = \phi(W\ (r_t \circ s_{t-1}) + U x_t + b)$$ State candidate

$$s_t = z_t \circ s_{t-1} + \underbrace{(1 - z_t)}_{\text{Write gate}} \circ \widetilde{s}_t$$ Current State



GRU equations

In GRU terminology

$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$ Read gate

$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$$ Update gate

$$\tilde{s}_t = \phi(W (r_t \circ s_{t-1}) + U x_t + b)$$ State candidate

$$s_t = z_t \circ s_{t-1} + (1 - z_t) \circ \tilde{s}_t$$ Current State

Forget some previous state

Write gate

GRU equations

In GRU terminology

$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$ Reset gate

$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$$ Update gate

$$\widetilde{s}_t = \phi(W\ (r_t \circ s_{t-1}) + U x_t + b)$$ State candidate

$$s_t = z_t \circ s_{t-1} + (1 - z_t) \circ \widetilde{s}_t$$ Current State

$$\begin{pmatrix} \widetilde{s}_t^1 \\ \widetilde{s}_t^2 \\ \widetilde{s}_t^3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \circ \begin{pmatrix} s_{t-1}^1 \\ s_{t-1}^2 \\ s_{t-1}^3 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \circ \begin{pmatrix} \widetilde{s}_t^1 \\ \widetilde{s}_t^2 \\ \widetilde{s}_t^3 \end{pmatrix}$$

Update gate = vector of zeros

Replace all memory content with the State Candidate

GRU equations

In GRU terminology

$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$ Reset gate

$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$$ Update gate

$$\widetilde{s}_t = \phi(W\ (r_t \circ s_{t-1}) + U x_t + b)$$ State candidate

$$s_t = z_t \circ s_{t-1} + (1 - z_t) \circ \widetilde{s}_t$$ Current State

$$\begin{pmatrix} s^1_{t-1} \\ s^2_{t-1} \\ s^3_{t-1} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \circ \begin{pmatrix} s^1_{t-1} \\ s^2_{t-1} \\ s^3_{t-1} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \circ \begin{pmatrix} \widetilde{s}^1_t \\ \widetilde{s}^2_t \\ \widetilde{s}^3_t \end{pmatrix}$$

Update gate = vector of ones

The current state equals the previous state

Completely ignore current input and state candidate

What is LSTM?

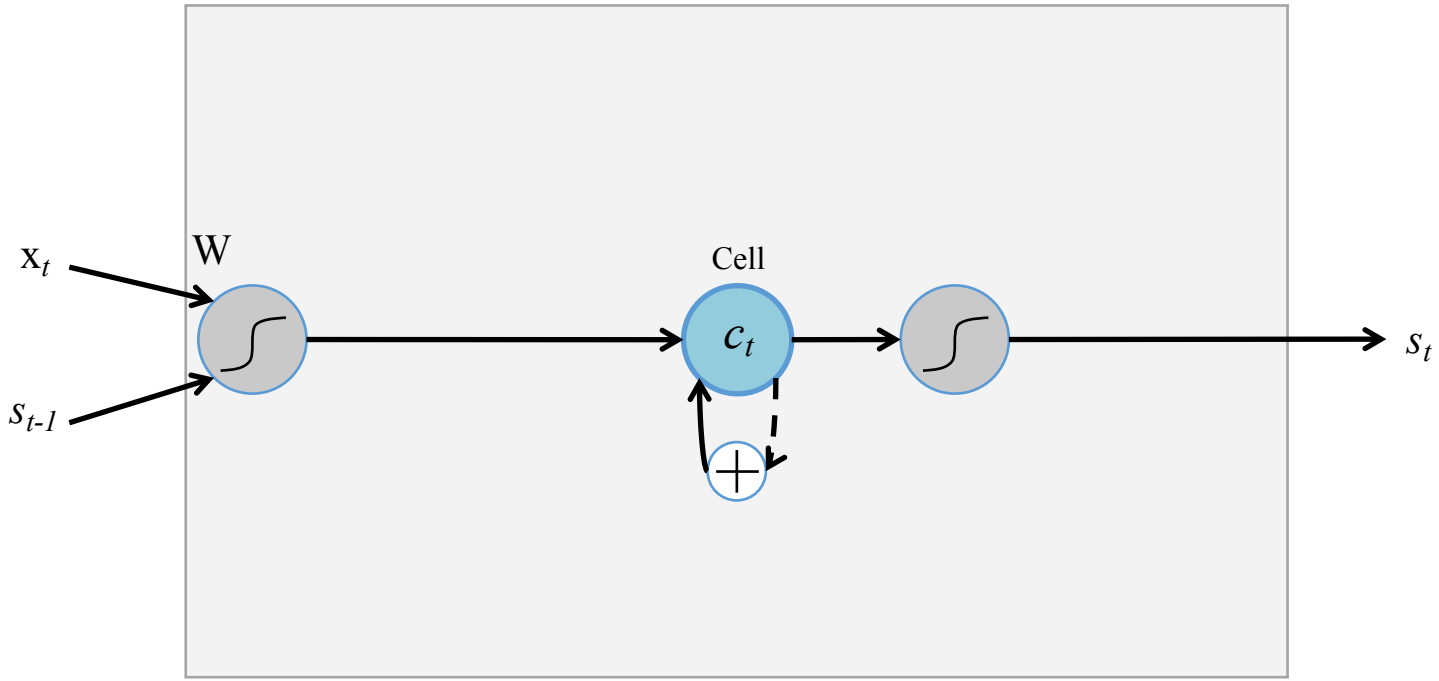
- The LSTM uses this idea of **"Constant Error Flow"** for RNNs to create a "Constant Error Carousel" (CEC) **which ensures that gradients don't decay.**

- The key component is a memory cell that acts like an accumulator (contains the identity relationship) over time

- Instead of computing new state as a matrix product with the old state, it rather computes the difference between them. Expressivity is the same, but gradients are better behaved

[1] Long Short-Term Memory, Hochreiter *et al*., 1997

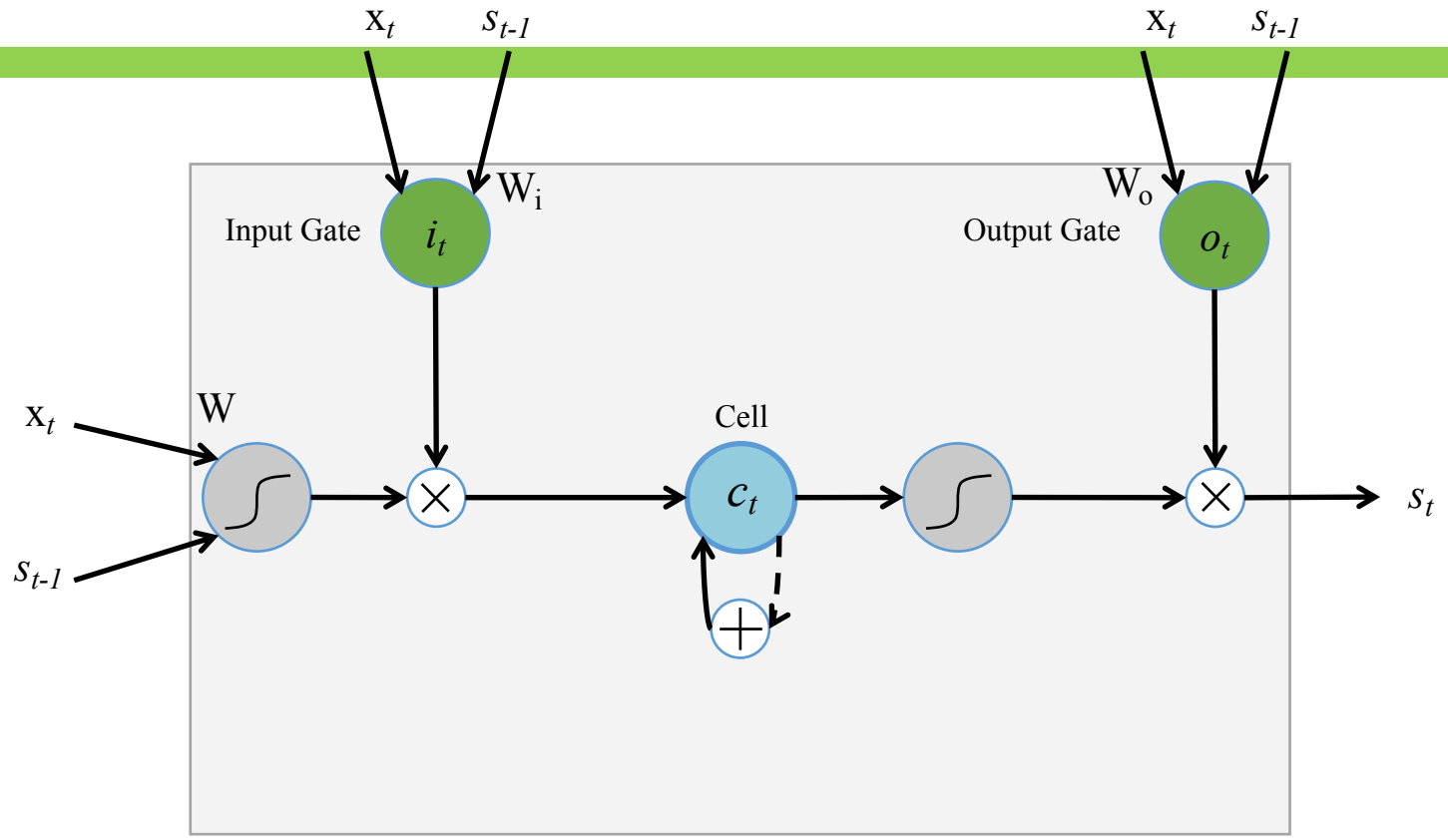
The LSTM Idea



$$c_t = c_{t-1} + \tanh W \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix} \qquad s_t = \tanh c_t$$

* Dashed line indicates time-lag

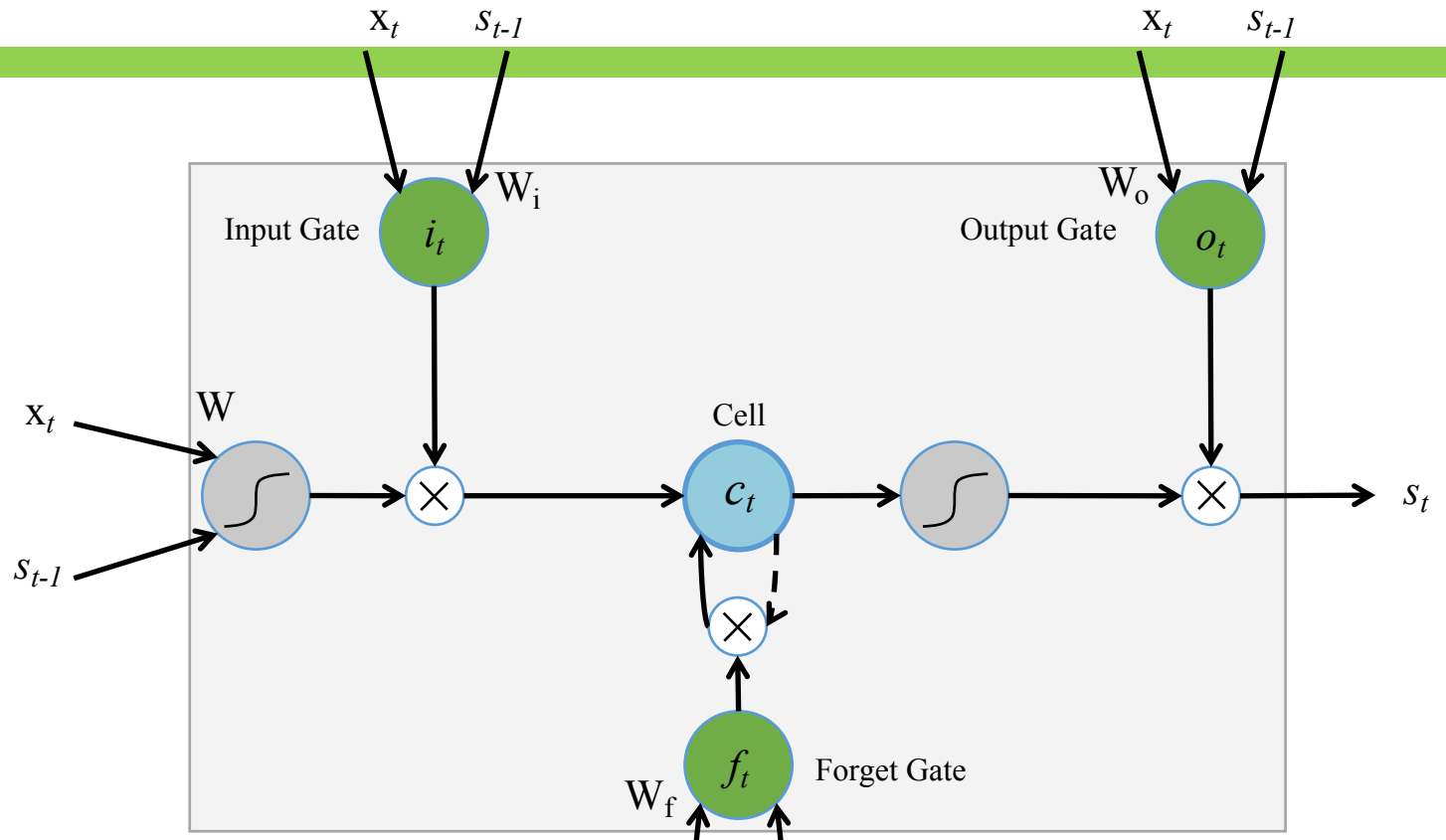
The Original LSTM Cell



$$c_t = c_{t-1} + i_t \otimes \tanh W \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix} \qquad s_t = o_t \otimes \tanh c_t \qquad i_t = \sigma\left(W_i \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix} + b_i \right)$$

Similarly for o_t

The Popular LSTM Cell



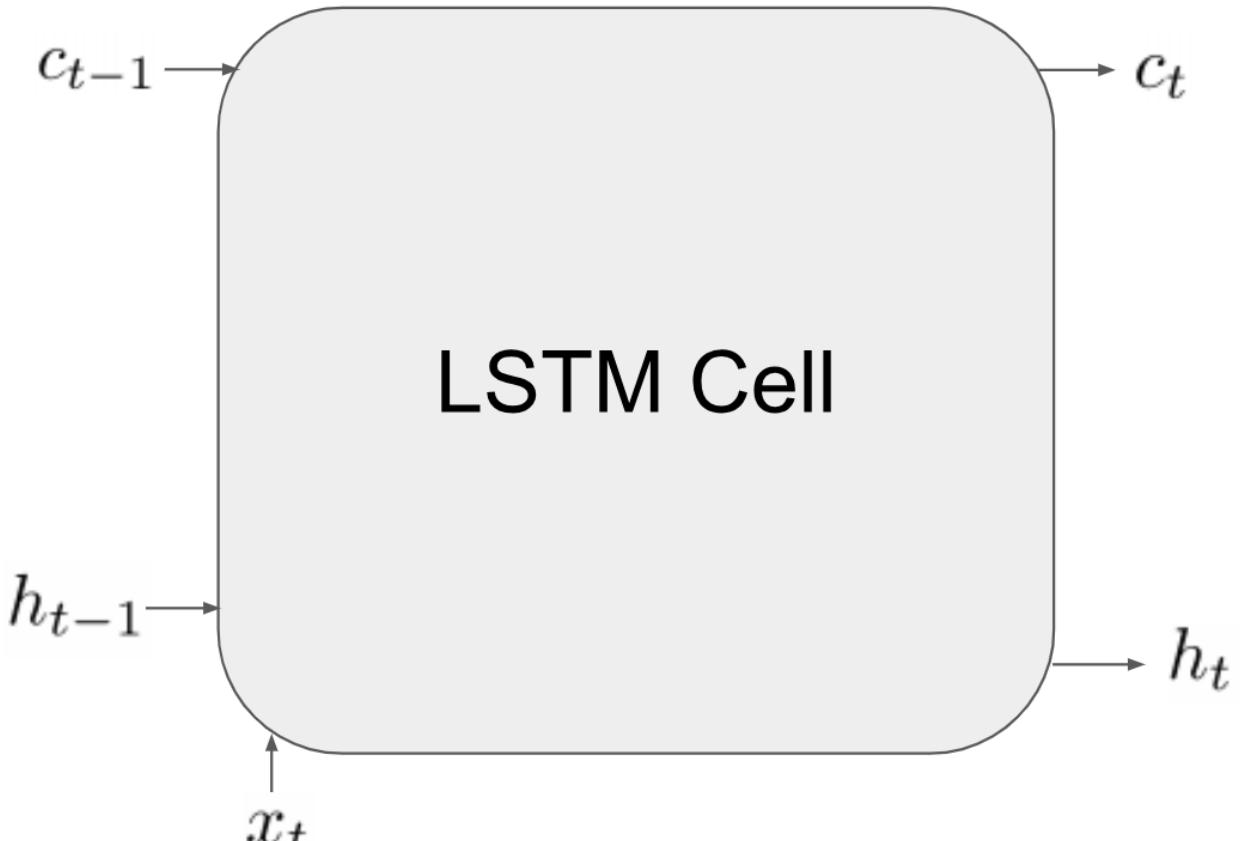
$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tanh W \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix}$$

$$f_t = \sigma\left(W_f \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix} + b_f \right)$$

Go To: [Illustrated LSTM Forward and Backward Pass]

LSTM Cell

(Some Math)



LSTM equations

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$ Input gate

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$ Forget gate

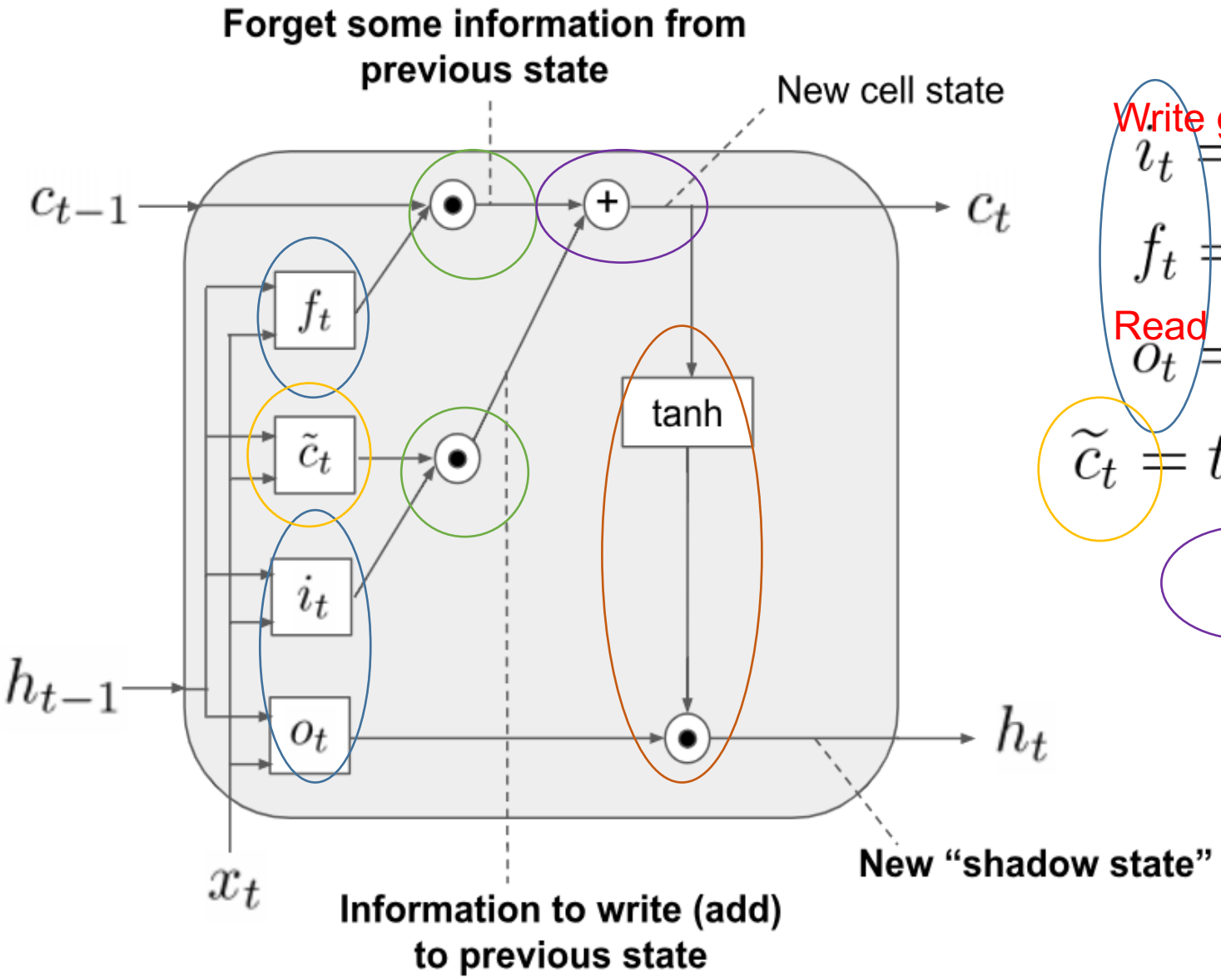
$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$ Output gate

$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$ Memory cell

$$h_t = o_t \circ tanh(c_t)$$ Shadow state

$$y_t = h_t$$ Cell Output



LSTM equations

h_{t-1}- is the gated version of memory content. What is the problem?

Write gate

We lose information. We should use shadow state to calculate.

Read

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$ Input gate

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$ Forget gate

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$ Output gate

$$\tilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$ Memory cell candidate

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$ Memory cell

$$h_t = o_t \circ tanh(c_t)$$ Shadow state

Read occurs after Write?

$$y_t = h_t$$ Cell Output

Read/Write gate?

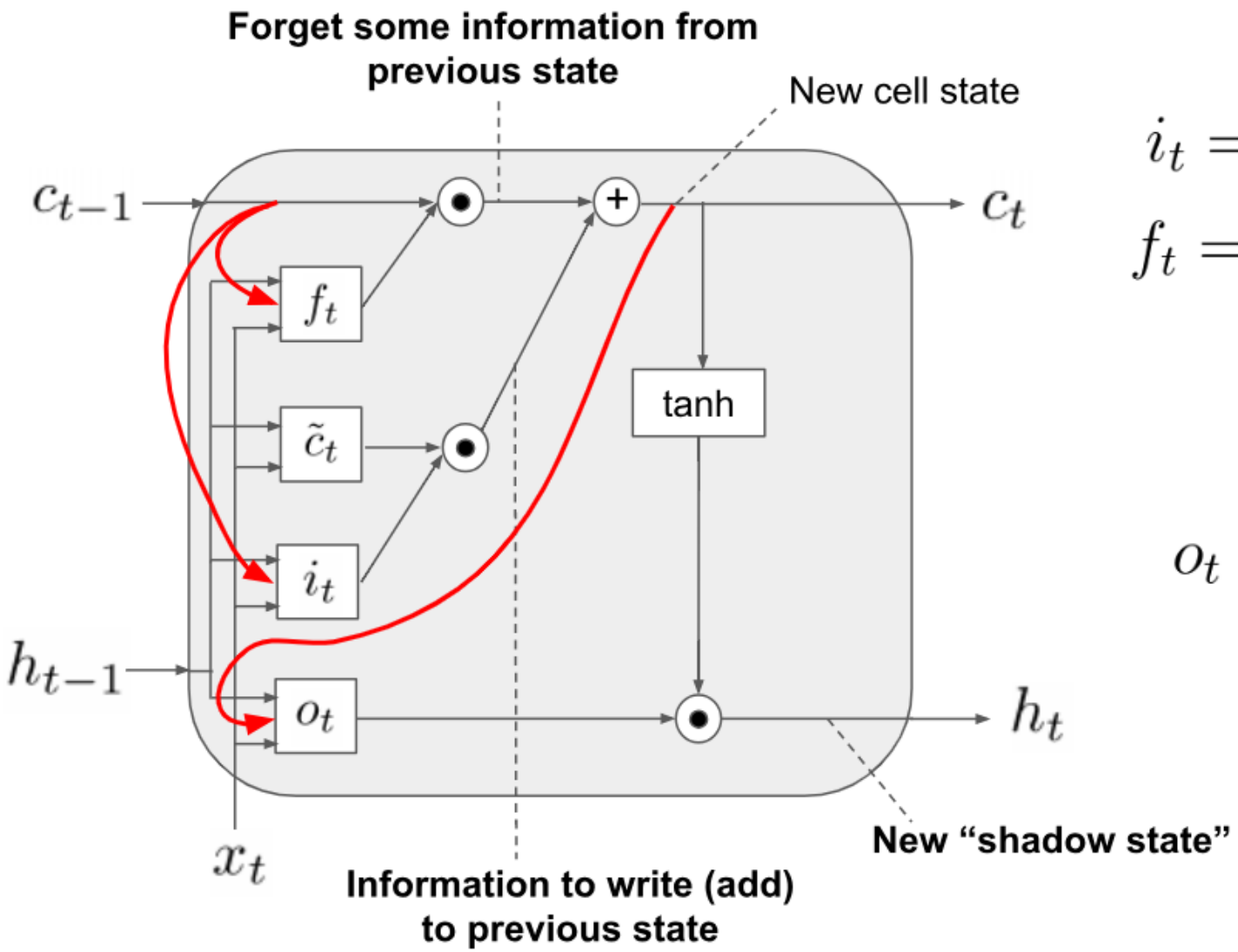
Adding peephole connection

LSTM equations

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$
$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$
$$h_t = o_t \circ tanh(c_t)$$
$$y_t = h_t$$

LSTM with peephole connections

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + P_i c_{t-1} + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + P_f c_{t-1} + b_f)$$
$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$
$$o_t = \sigma(W_o h_{t-1} + U_o x_t + P_o c_t + b_o)$$
$$h_t = o_t \circ tanh(c_t)$$
$$y_t = h_t$$



LSTM with Peephole connections

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + P_i c_{t-1} + b_i)$$

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + P_f c_{t-1} + b_f)$$

$$\widetilde{c}_t = tanh(W h_{t-1} + U x_t + b)$$

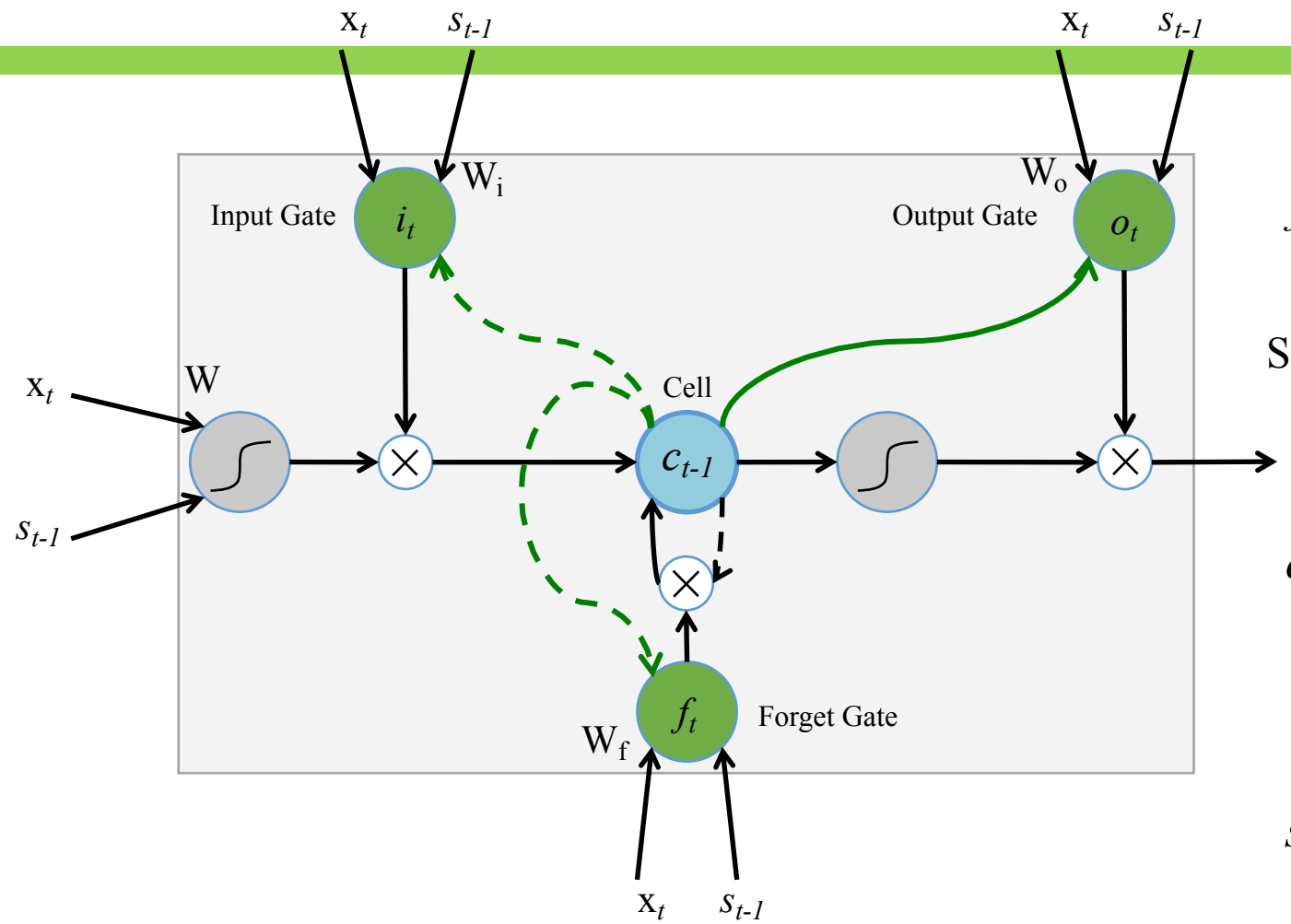
$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t$$

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + P_o c_t + b_o)$$

$$h_t = o_t \circ tanh(c_t)$$

$$y_t = h_t$$

Peephole LSTM



$$f_t = \sigma\left(W_f \begin{pmatrix} x_t \\ s_{t-1} \\ c_{t-1} \end{pmatrix} + b_f\right)$$

Similarly for i_t, o_t (uses c_t)

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tanh W \begin{pmatrix} x_t \\ s_{t-1} \end{pmatrix}$$

$$s_t = o_t \otimes \tanh c_t$$

* Dashed line indicates time-lag

Peephole LSTM

- Gates can only see the output from the previous time step, which is close to 0 if the output gate is closed. However, these gates control the CEC cell.

- Helped the LSTM learn better timing for the problems tested – Spike timing and Counting spike time delays

Recurrent nets that time and count, Gers *et al.*, 2000

Other minor variants

- Coupled Input and Forget Gate $f_t = 1 - i_t$

- Full Gate Recurrence

$$f_t = \sigma\left(W_f \begin{pmatrix} x_t \\ s_{t-1} \\ c_{t-1} \\ i_{t-1} \\ f_{t-1} \\ o_{t-1} \end{pmatrix} + b_f \right)$$

LSTM: A Search Space

- Tested the following variants, using Peephole LSTM as standard:
  1. No Input Gate (NIG)
  2. No Forget Gate (NFG)
  3. No Output Gate (NOG)
  4. No Input Activation Function (NIAF)
  5. No Output Activation Function (NOAF)
  6. No Peepholes (NP)
  7. Coupled Input and Forget Gate (CIFG)
  8. Full Gate Recurrence (FGR)
- On the tasks of: e.g.
  - Timit Speech Recognition: Audio frame to 1 of 61 phonemes
  - IAM Online Handwriting Recognition: Sketch to characters
  - JSB Chorales: Next-step music frame prediction

LSTM: A Search Space Odyssey, Greff *et al.*, 2015

LSTM: A Search Space

- The standard LSTM performed reasonably well on multiple datasets and none of the modifications significantly improved the performance
- Coupling gates and removing peephole connections simplified the LSTM without hurting performance much
- The forget gate and output activation are crucial
- Found interaction between learning rate and network size to be minimal – indicates calibration can be done using a small network first

LSTM: A Search Space Odyssey, Greff *et al.*, 2015

Note for GRU

- A very simplified version of the LSTM
- Merges forget and input gate into a single ‘update’ gate
- Merges cell and hidden state

- Has fewer parameters than an LSTM and has been shown to outperform LSTM on some tasks

Milestones summary

Vanilla RNN

Late 1980s - backpropagation through time to train Vanilla RNN

LSTM

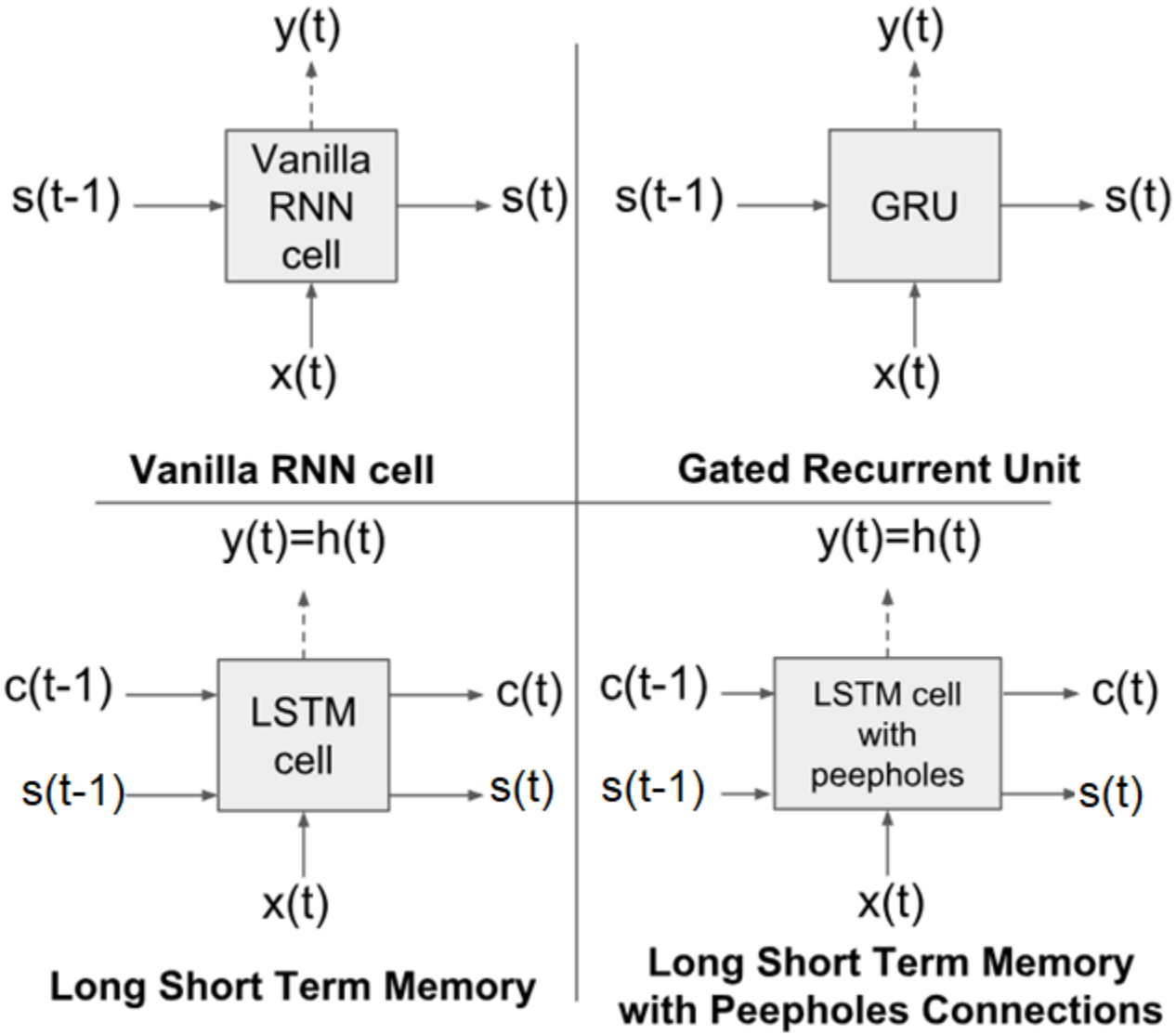
1997 - Long Short-Term Memory (S.Hochreiter, J.Schmidhuber)

LSTM with Peepholes

2000 - Recurrent nets that time and count (F.A. Gers ; J. Schmidhuber)

GRU

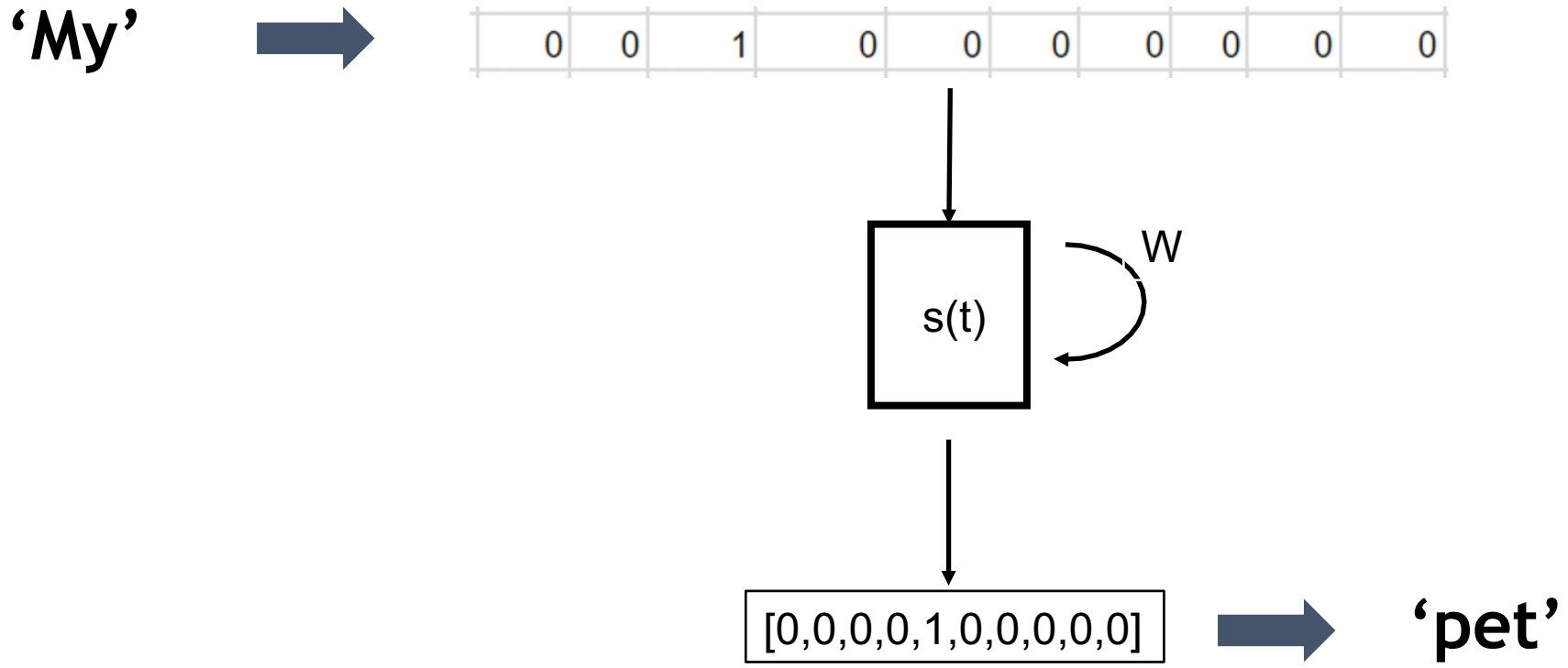
2014 - Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation (Kyunghyun Cho, Yoshua Bengio, and others)



Applications

Language generation

Notes

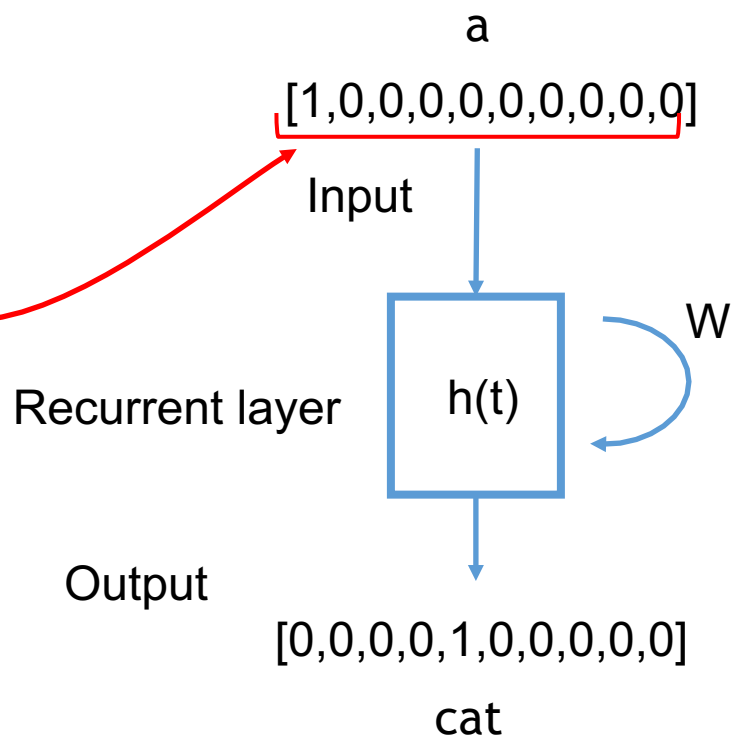


Using RNN

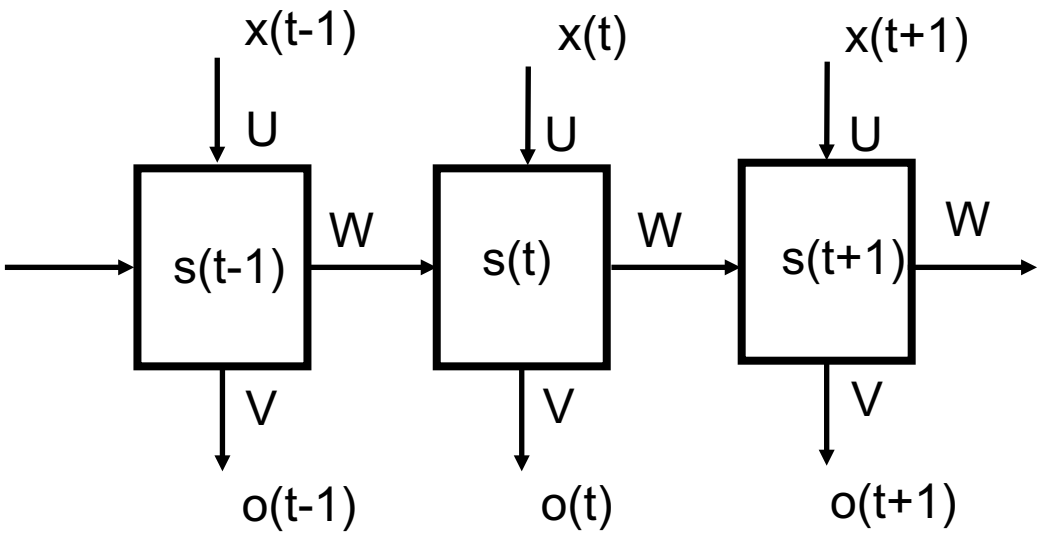
Word prediction example

a	1	0	0	0	0	0	0	0	0	0
cat	0	0	0	0	1	0	0	0	0	0
is	0	0	0	1	0	0	0	0	0	0
on	0	0	1	0	0	0	0	0	0	0
the	0	1	0	0	0	0	0	0	0	0
grass	0	0	0	0	0	0	0	0	1	0

One-hot coding



a	the	on	is	cat	park	play	swing	grass	sitting
0	1	2	3	4	5	6	7	8	9



Unrolled Recurrent Layer

[0 , 4 , 3 , 2 , 1 , 8]

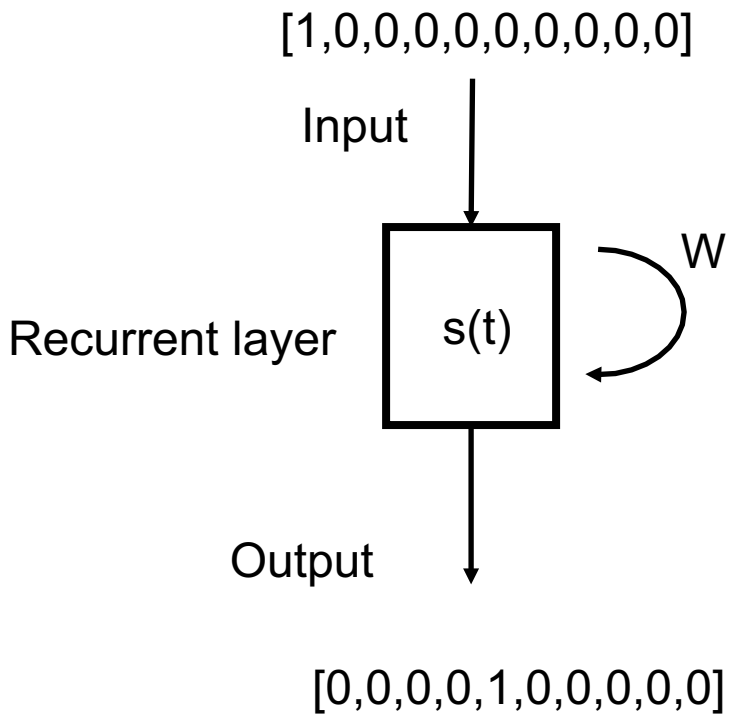
A cat is on the grass.

RNNs learn by reducing the error between their predicted next word and the actual next word in a corpus. RNNs are structured to "remember" the words that led to their prediction.

Time series information

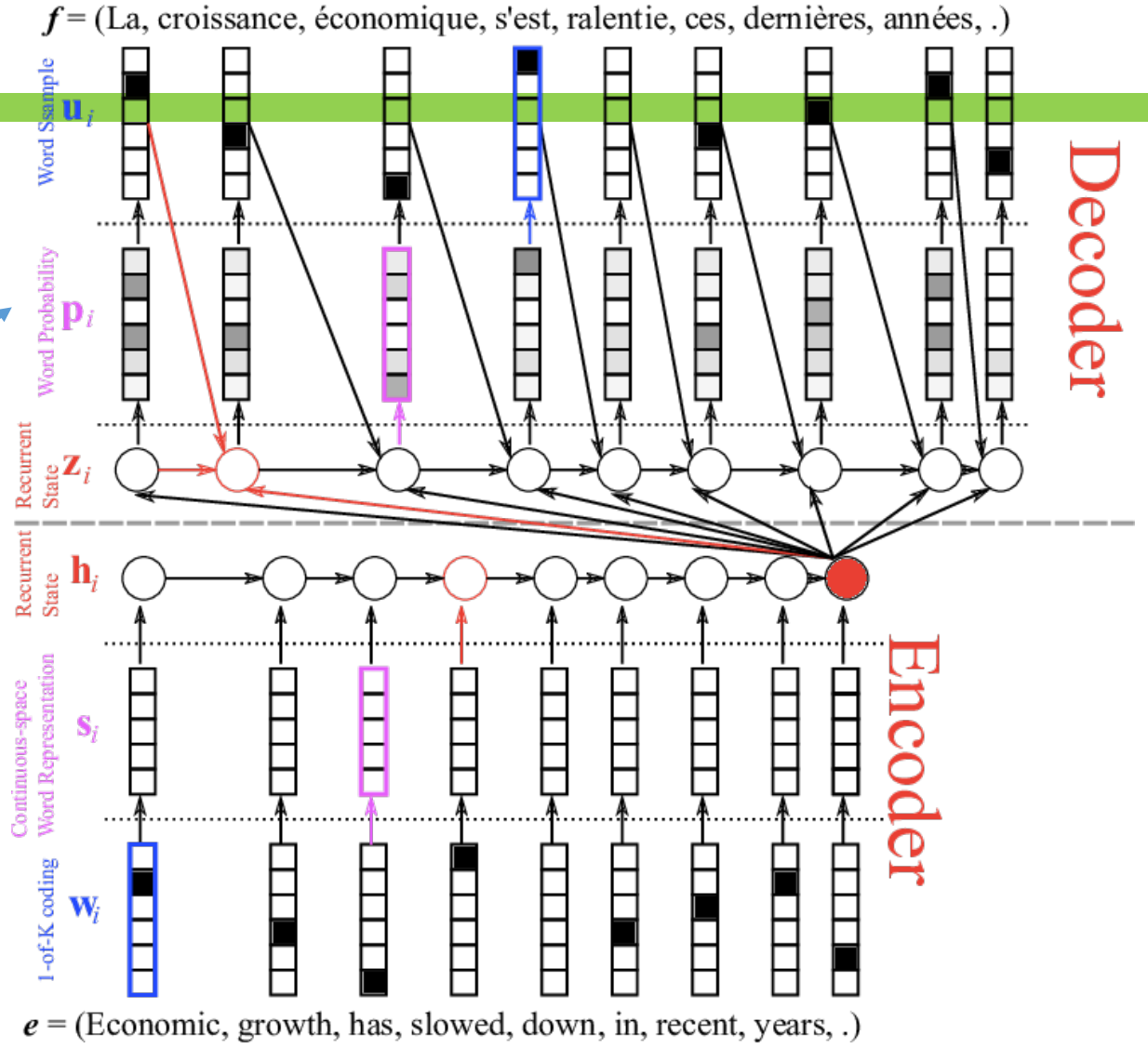
Recurrent neural networks are a popular approach

Demonstrated effectiveness with sentence and code creation as well as language translation

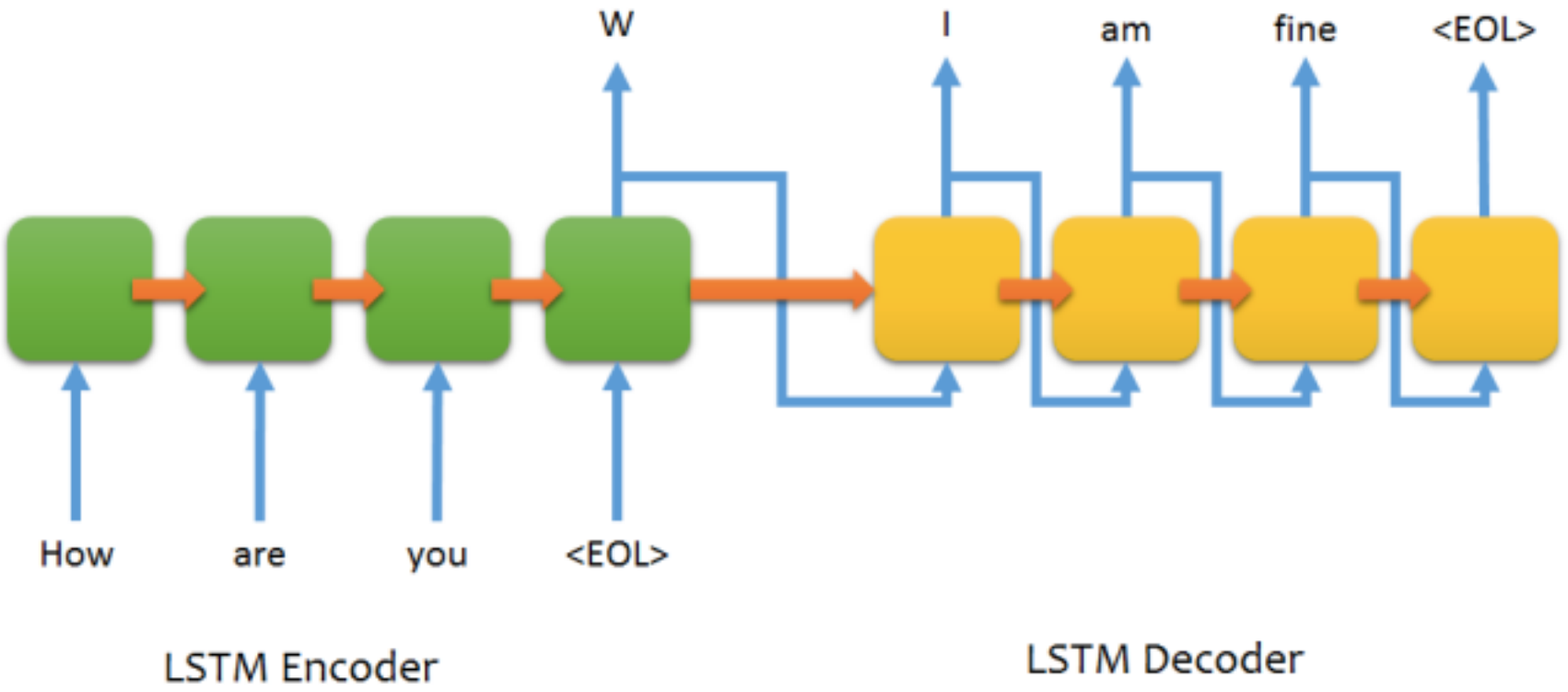


Language translation

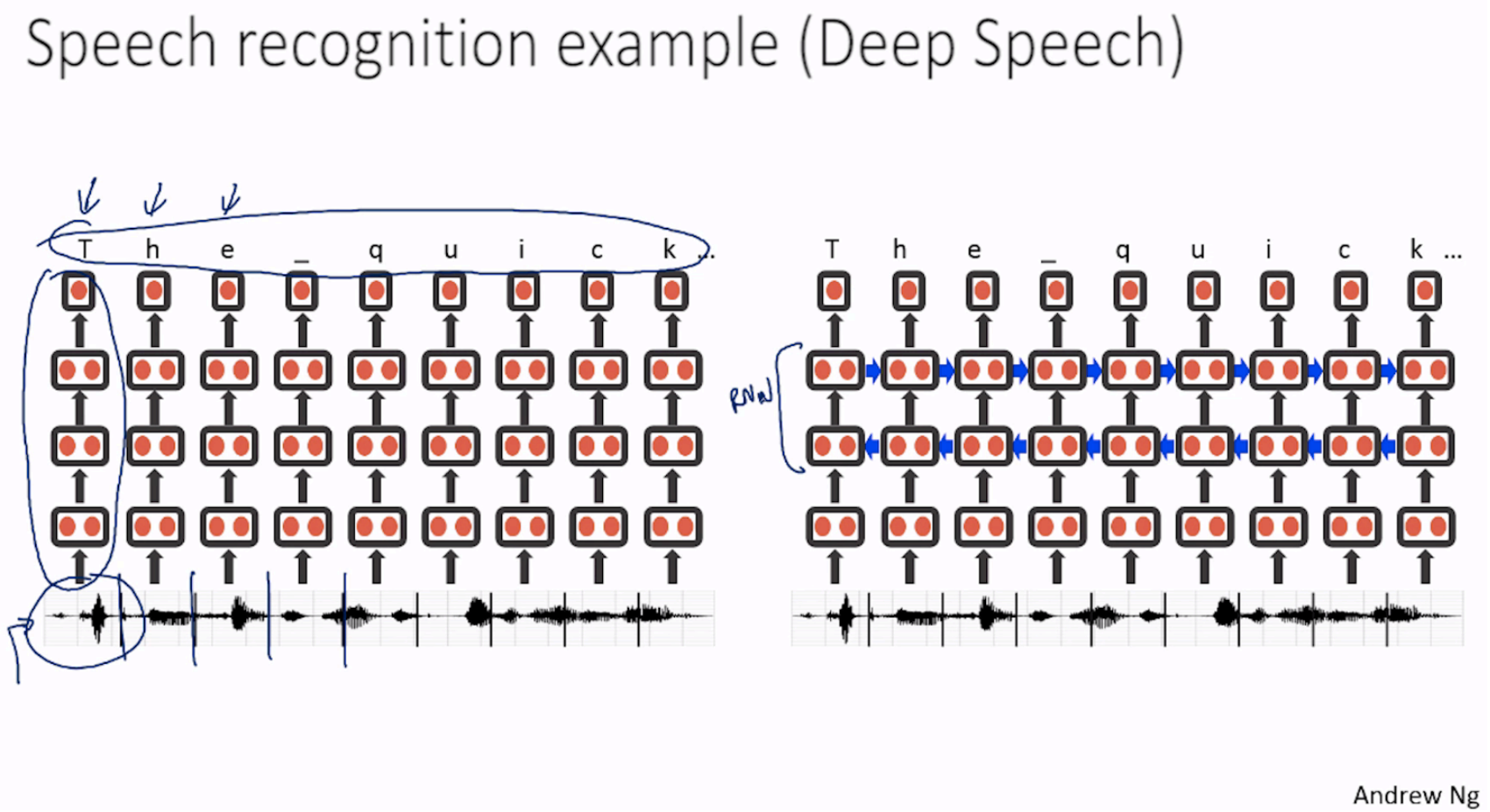
LSTM



Seq2Seq chatbot model



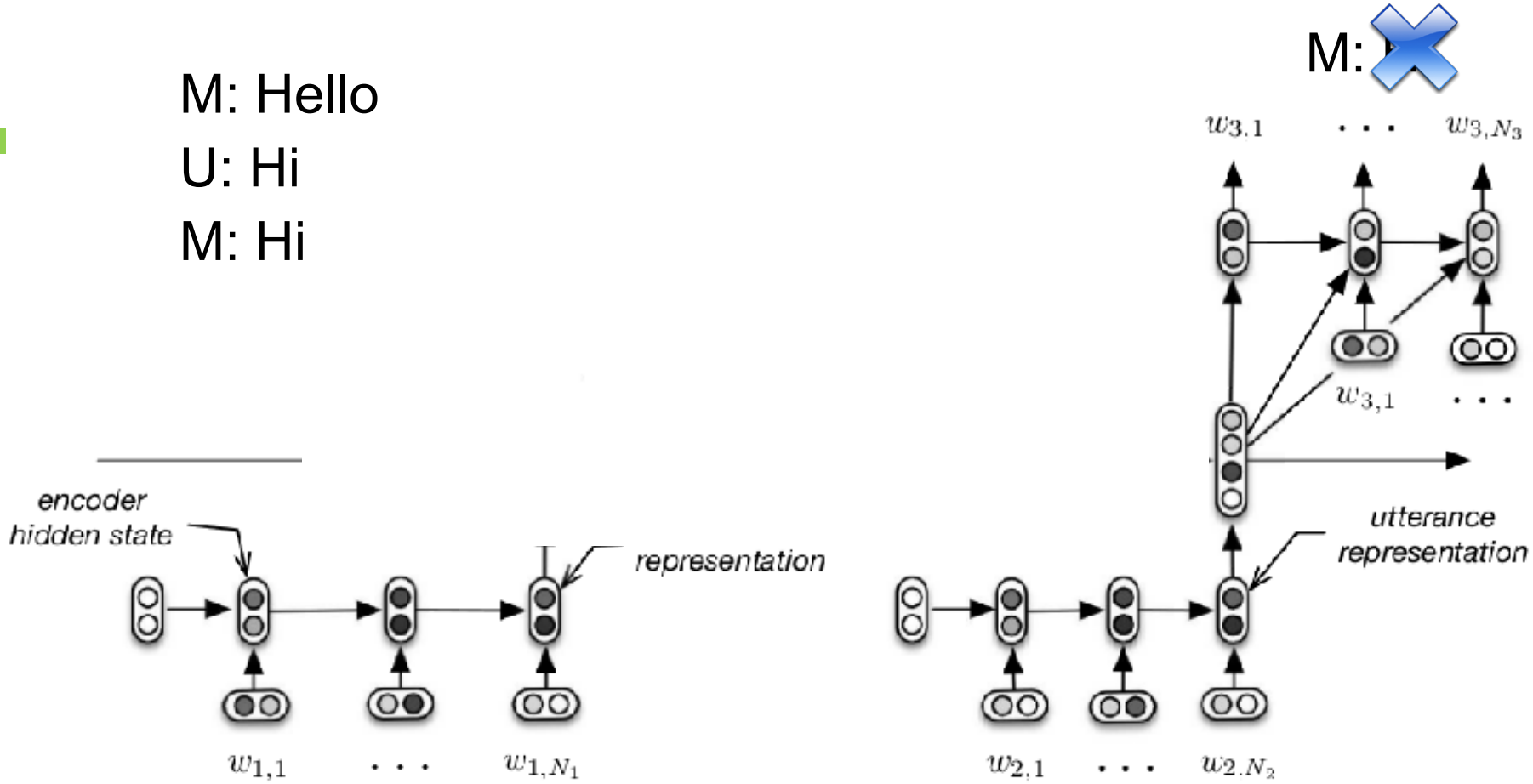
Baidu speech recognition



M: Hello

U: Hi

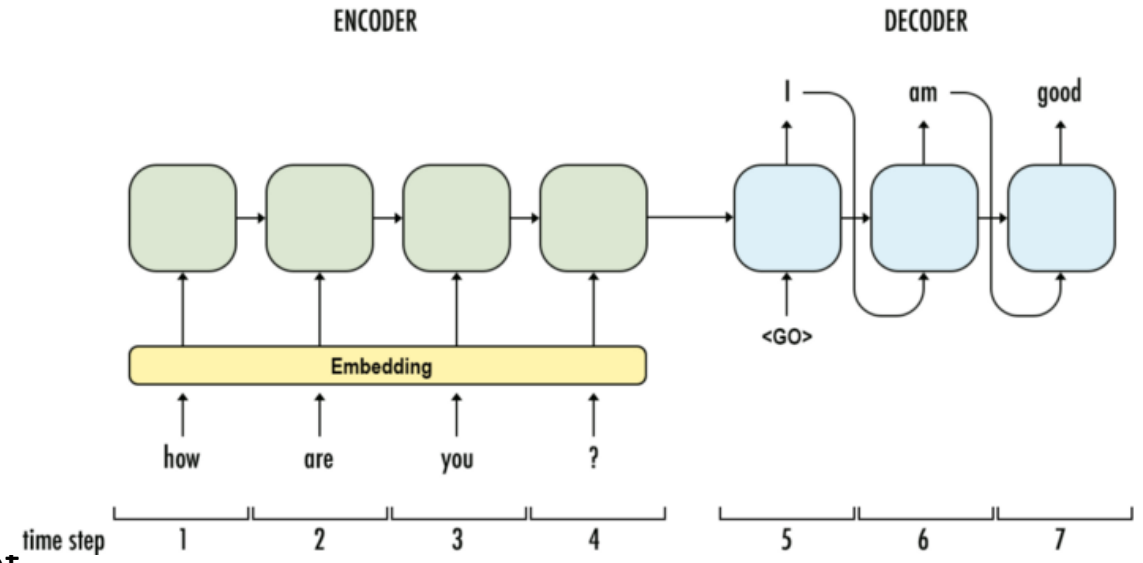
M:



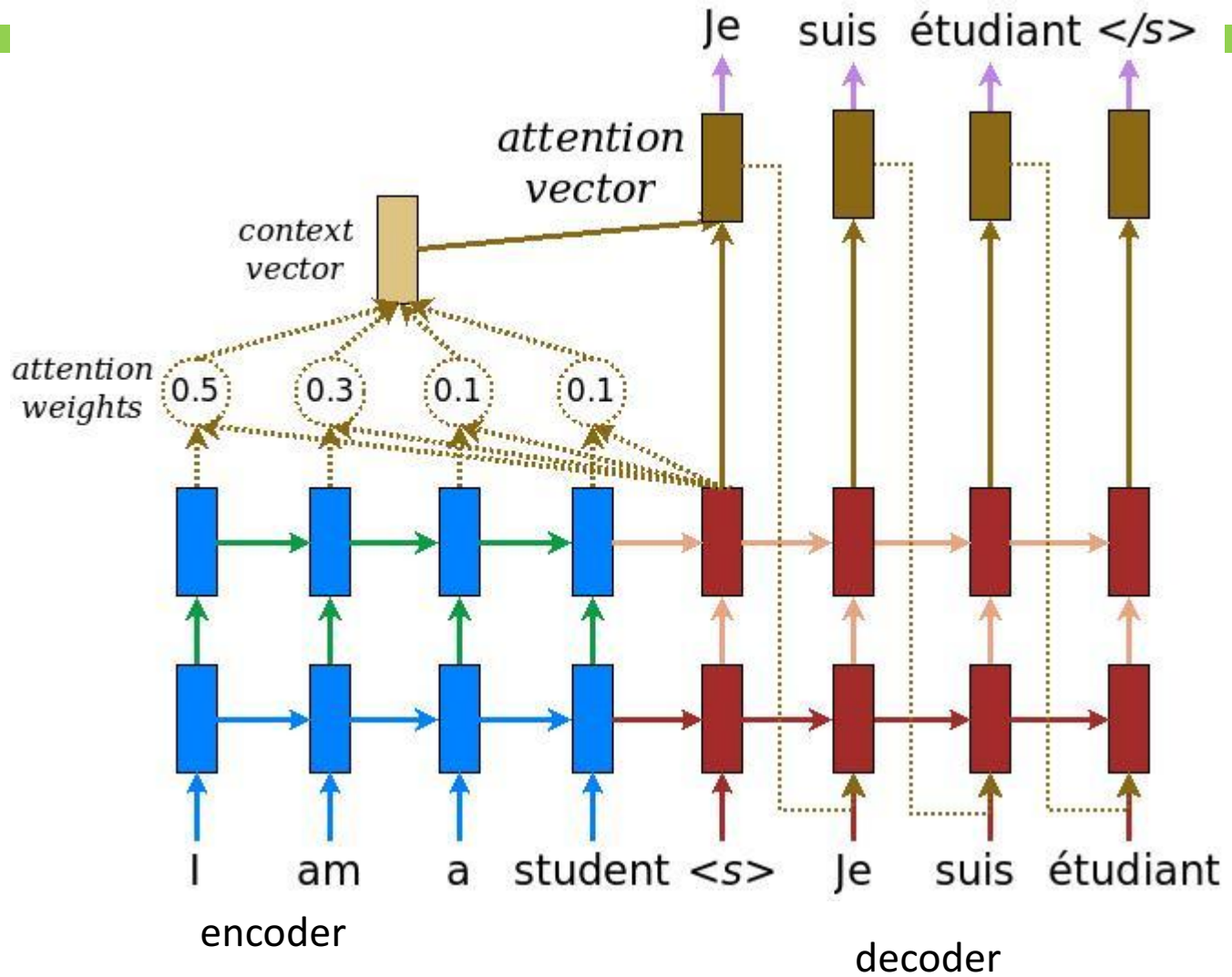
Serban, Iulian V., Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau, 2015 "Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models.

Attention

- It is a vector; output of dense layer using softmax.
- It allows machine to look over the information in the original sentence and generates proper words according to the current word and context.



https://medium.com/syncedreview/a-brief-overview-of-attention-mechanism-13c578ba9129

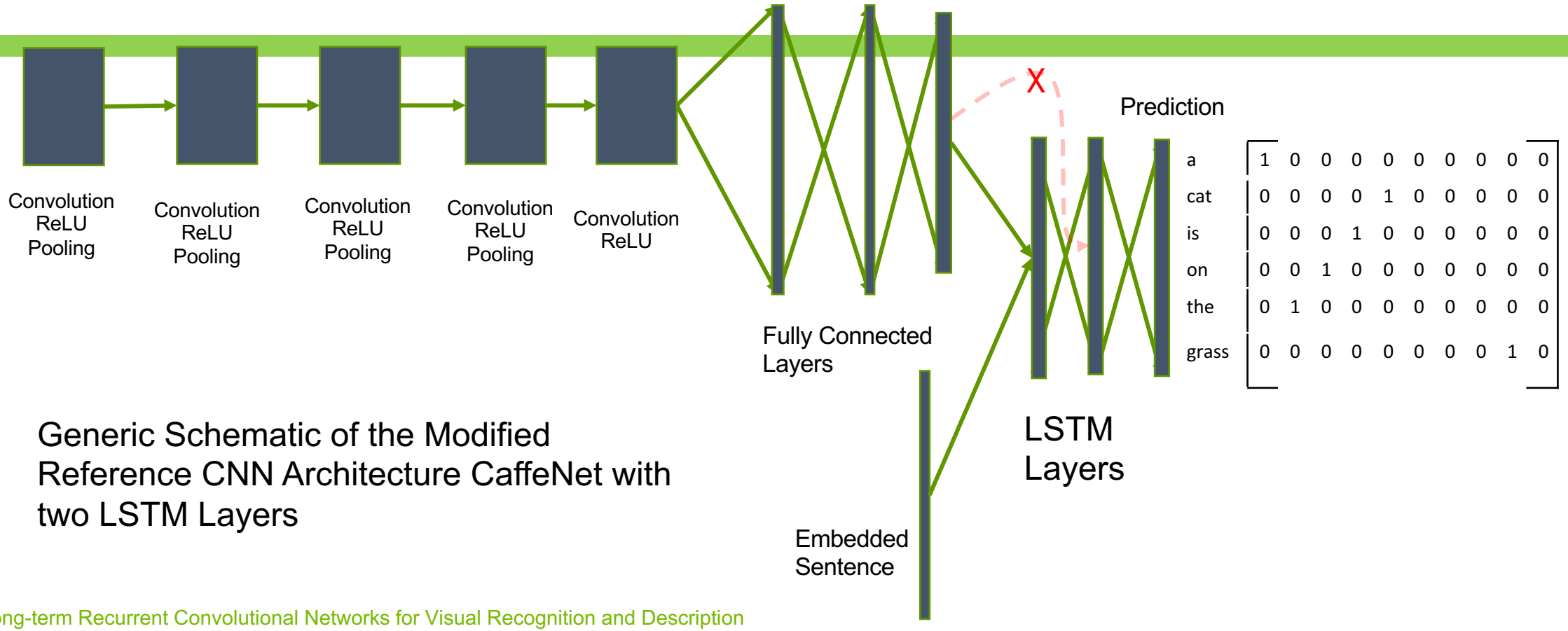


$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]} \quad (1)$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]} \quad (2)$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_{\boldsymbol{c}}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad \text{[Attention vector]} \quad (3)$$

Image captioning



Generic Schematic of the Modified Reference CNN Architecture CaffeNet with two LSTM Layers

Long-term Recurrent Convolutional Networks for Visual Recognition and Description
Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, Trevor Darrell

Example results

CaffeNet	A white bird standing on top of a sandy beach.
VGG	A small bird standing on the ground.

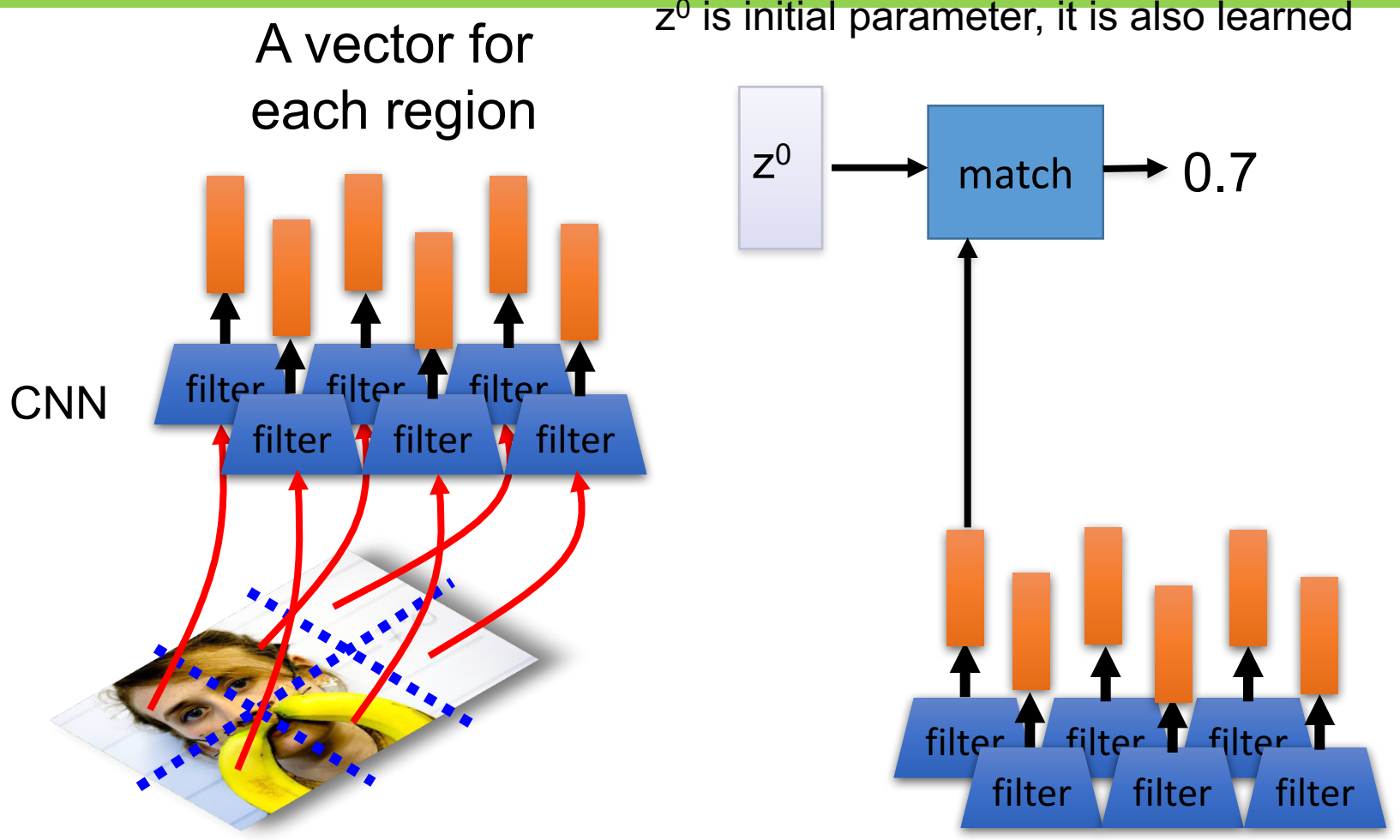
CaffeNet	A white cat sitting on a chair.
VGG	A white and white cat laying on a white chair.

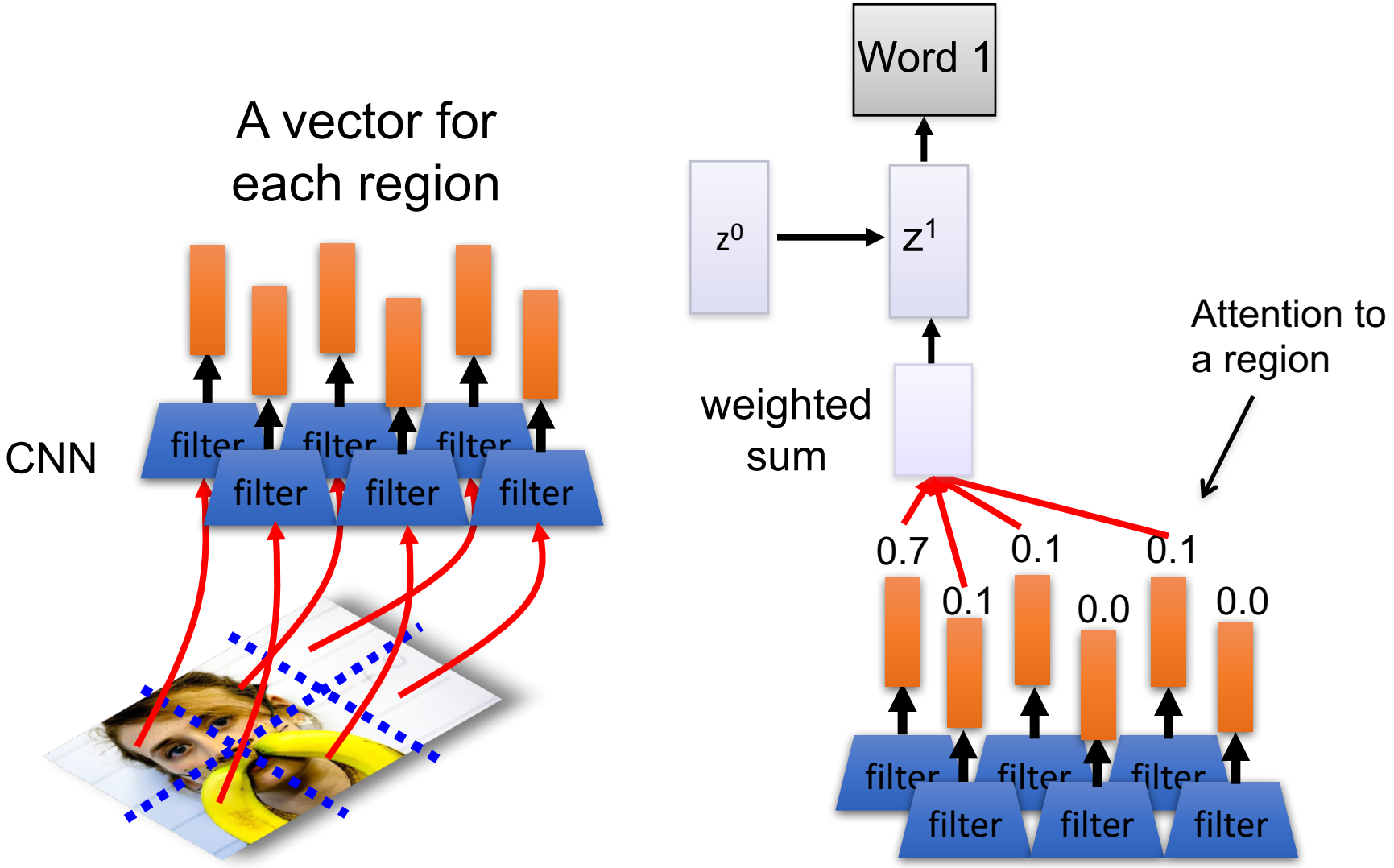
CaffeNet	A white horse standing in a lush field of grass.
VGG	A white horse standing in a field next to a fence.

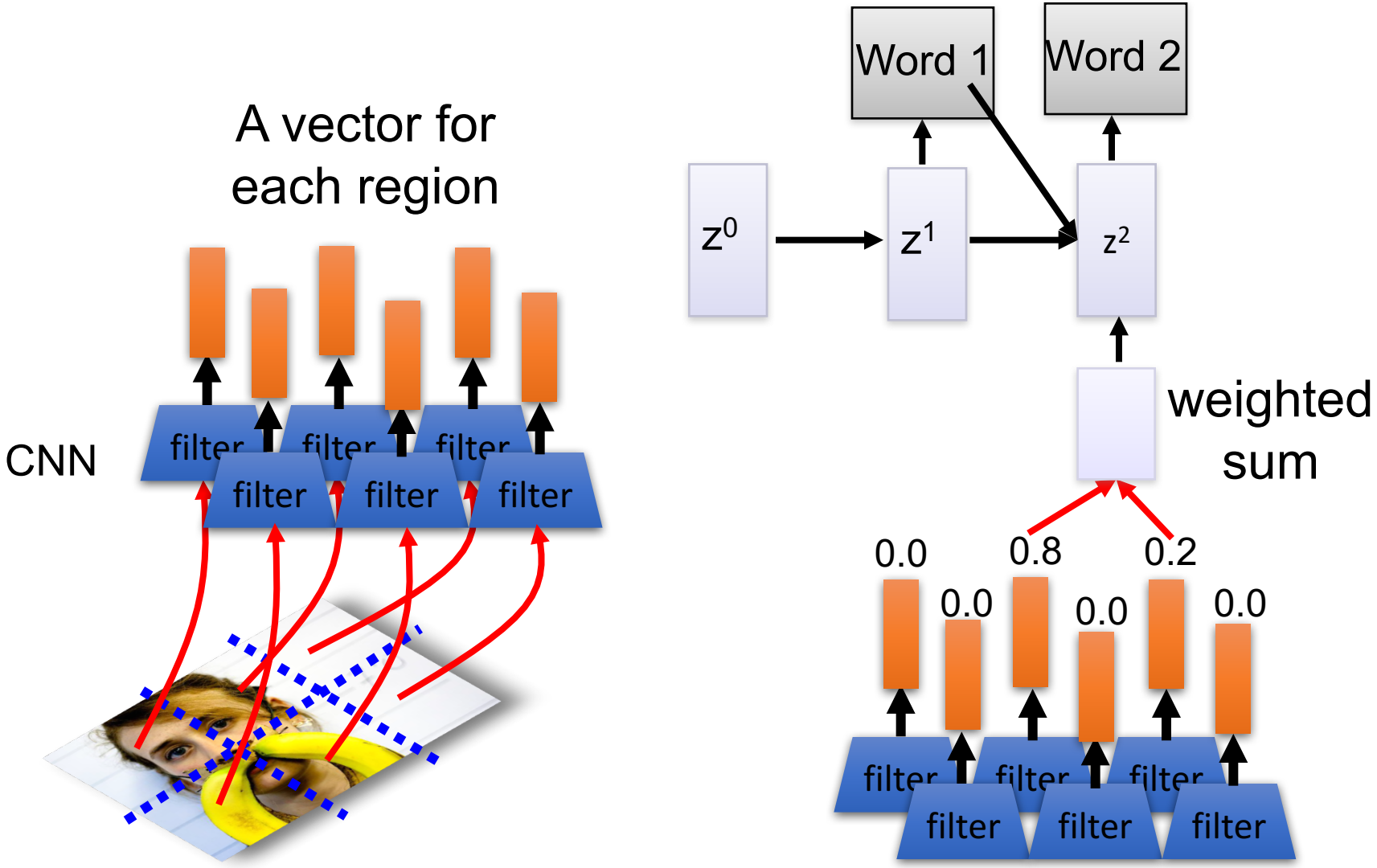
CaffeNet	A bunch of bananas that are on a table.
VGG	A close up of a bunch of white flowers.

*Results shown here were generated using work from this paper.
Long-term Recurrent Convolutional Networks for Visual Recognition and Description
Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, Trevor Darrell

Image caption generation using attention (From CY Lee lecture)




A vector for each region
CNN
filter
filter
filter
filter
filter
filter
filter
Word 1
z^0
z^1
weighted sum
Attention to a region
0.7
0.1
0.1
0.1
0.0
0.0
filter
filter
filter
filter
filter
filter
filter


A vector for each region
CNN
filter
filter
filter
filter
filter
filter
filter
Word 1
Word 2
z^0
z^1
z^2
weighted sum
0.0
0.8
0.2
0.0
0.0
0.0
filter
filter
filter
filter
filter
filter

Video captioning

Process

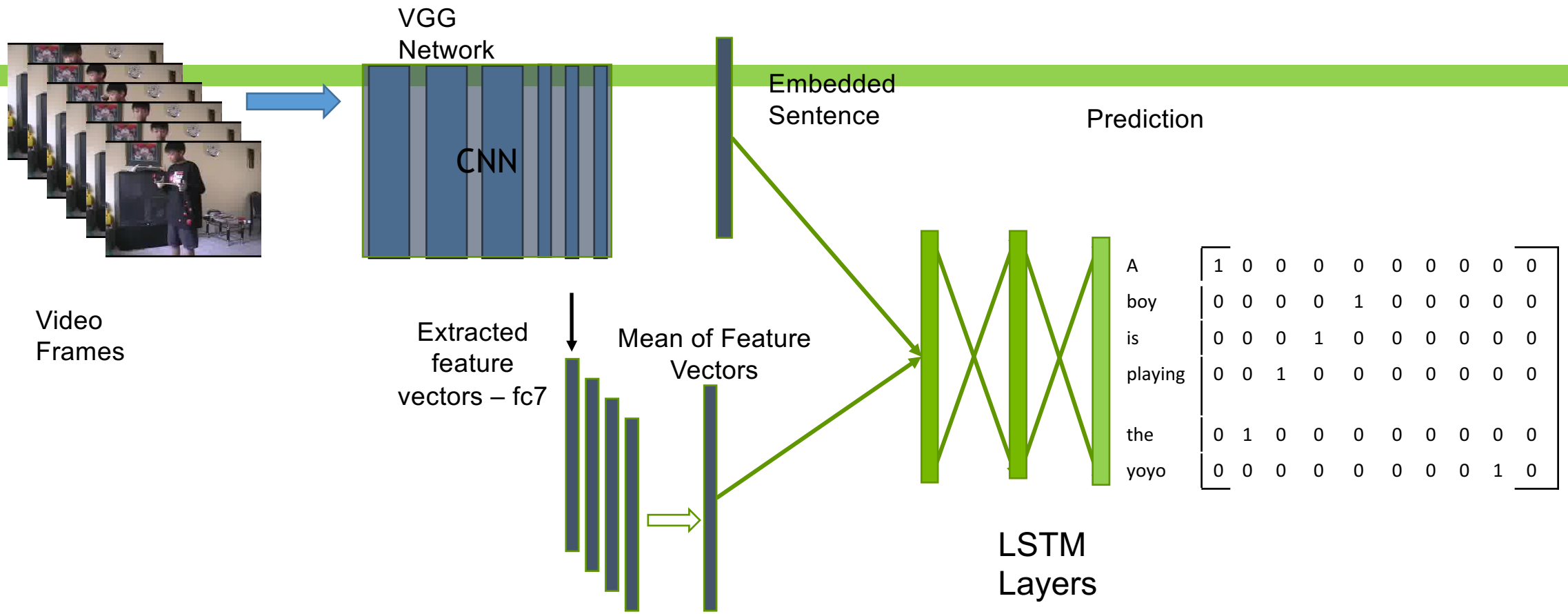
- Import libraries
- Evaluate videos and captions
  - Create a mean vector of a single clip
  - This will generate a high-level representation of each frame from layer fc7
- Align captions with feature maps
  - Will work with a subset of the data
- Predict next word for captions
  - Parse, tokenize, etc.

Video captioning

Process

- Architect the network (RNN)

- Train / build model
- Evaluate a training image & captions
- Generate a caption for a validation image



Translating Videos to Natural Language Using Deep Recurrent Neural Networks
Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond Mooney, Trevor Darrell, Kate Saenko

Results

A man is riding a horse

A animal is eating

A dog is standing

MSVD - Collecting Highly Parallel Data for Paraphrase Evaluation
David L. Chen and William B. Dolan

Translating Videos to Natural Language Using Deep Recurrent Neural Networks
Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond Mooney, Trevor Darrell, Kate Saenko

Time series prediction: LSTM

Stock price prediction

Rainfall forecasting

….. To be continue

LAB

Prerequisites

- Terminology: One-hot coding, word-embedding
- Tools: nltk, genism, glove_python

One-hot coding

a	1	0	0	0	0	0	0	0	0	0
cat	0	0	0	0	1	0	0	0	0	0
is	0	0	0	1	0	0	0	0	0	0
on	0	0	1	0	0	0	0	0	0	0
the	0	1	0	0	0	0	0	0	0	0
grass	0	0	0	0	0	0	0	0	1	0

Word embedding: Vocabulary representation

https://towardsdatascience.com/3-silver-bullets-of-word-embedding-in-nlp-10fa8f50cc5a

- Three common ones:
  - word2vec: trained on google news and provided by Google. Based on 100 billion words from Google News data, they trained model with 300 dimensions.
  - glove: provided by Stanford NLP team. Stanford provides various models from 25, 50 , 100, 200 to 300 dimensions base on 2, 6, 42, 840 billion tokens.
  - fastText: released by Facebook which provides 3 models with 300 dimensions. One of the pre-trained model is trained with subword. For example, “difference”, it will be trained by “di”, “dif”, “diff” and so on.

word2vec

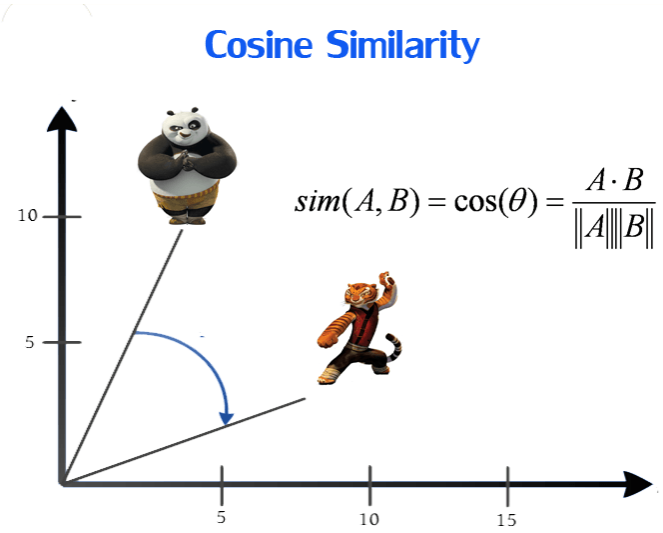
- Consider 2 sentences:

Have a good day.

Have a great day.

- Objective is to have words with similar context occupy close spatial position. The cosine of the angle between such vectors should be close to 1, i.e. angle close to 0.

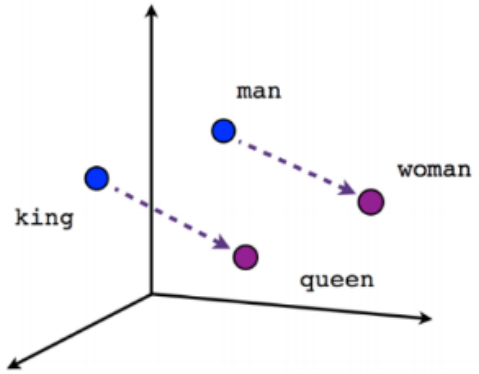
https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa



word2vec

http://web2.cs.columbia.edu/~blei/seminar/2016_discrete_data/readings/MikolovSutskeverChenCorradoDean2013.pdf

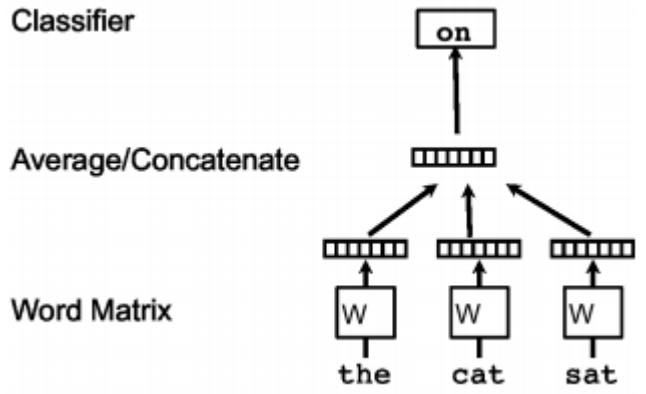
- Origin in paper https://arxiv.org/abs/1301.3781
- It is numeric representation for each word, that will be able to capture such relations between the word and other words.
  - E.g. France and Paris will be more closely related than France and power.
  - Relations may be : synonym, antonym, analogy, etc.
- Its algorithm is done by Continuous Bag-of-Words model (CBOW) and the Skip-Gram model.



Male-Female

CBOW

- CBOW creates a sliding window around current word, to predict it from “context” — the surrounding words. Each word is represented as a feature vector. After training, these vectors become the word vectors.

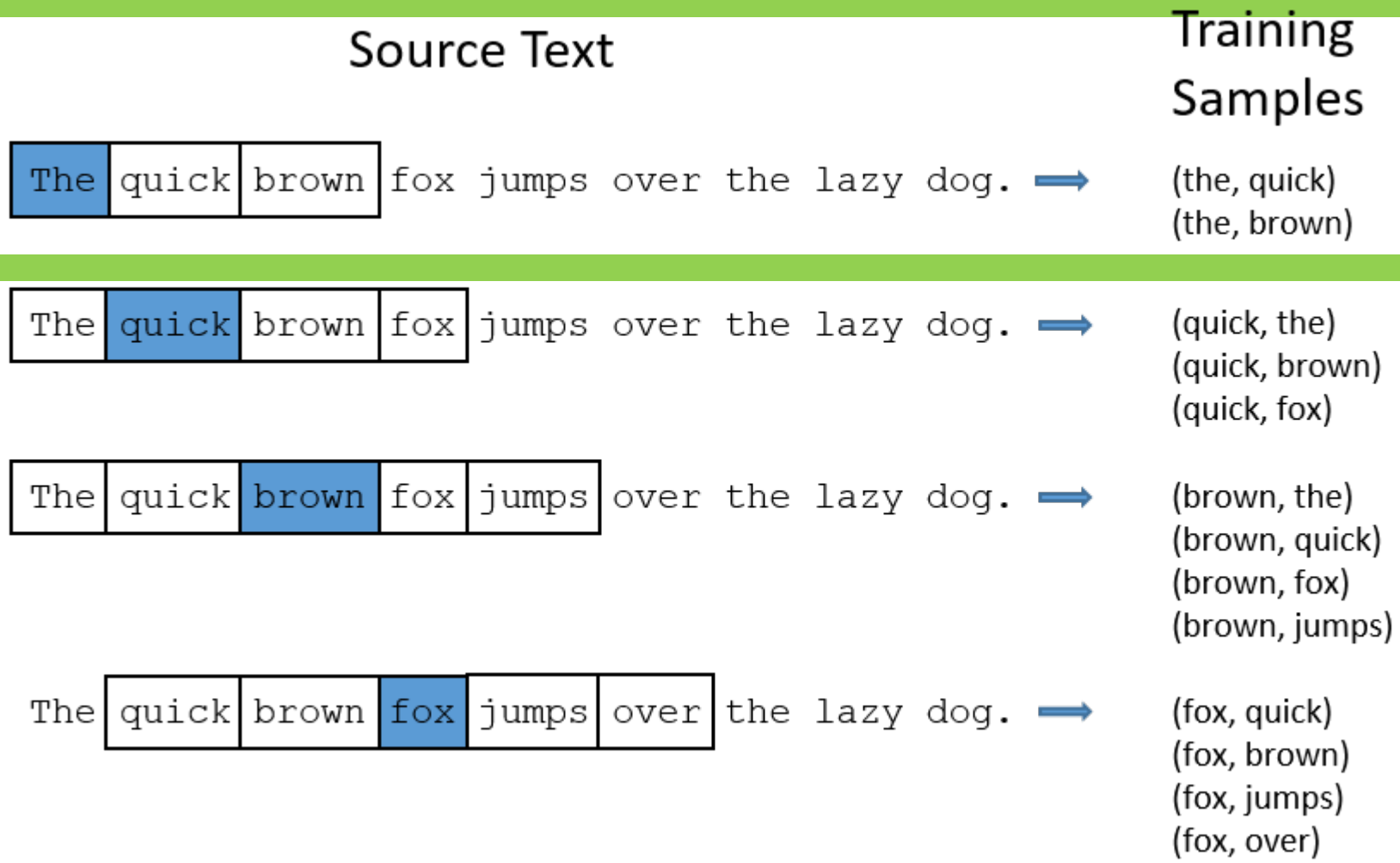


  - Vectors which represent similar words are close by different distance metrics.

https://medium.com/scaleabout/a-gentle-introduction-to-doc2vec-db3e8c0cce5e

Skip gram

- Instead of predicting one word each time, use 1 word to predict all surrounding words (“context”). Skip gram is much slower than CBOW, but considered more accurate with infrequent words.

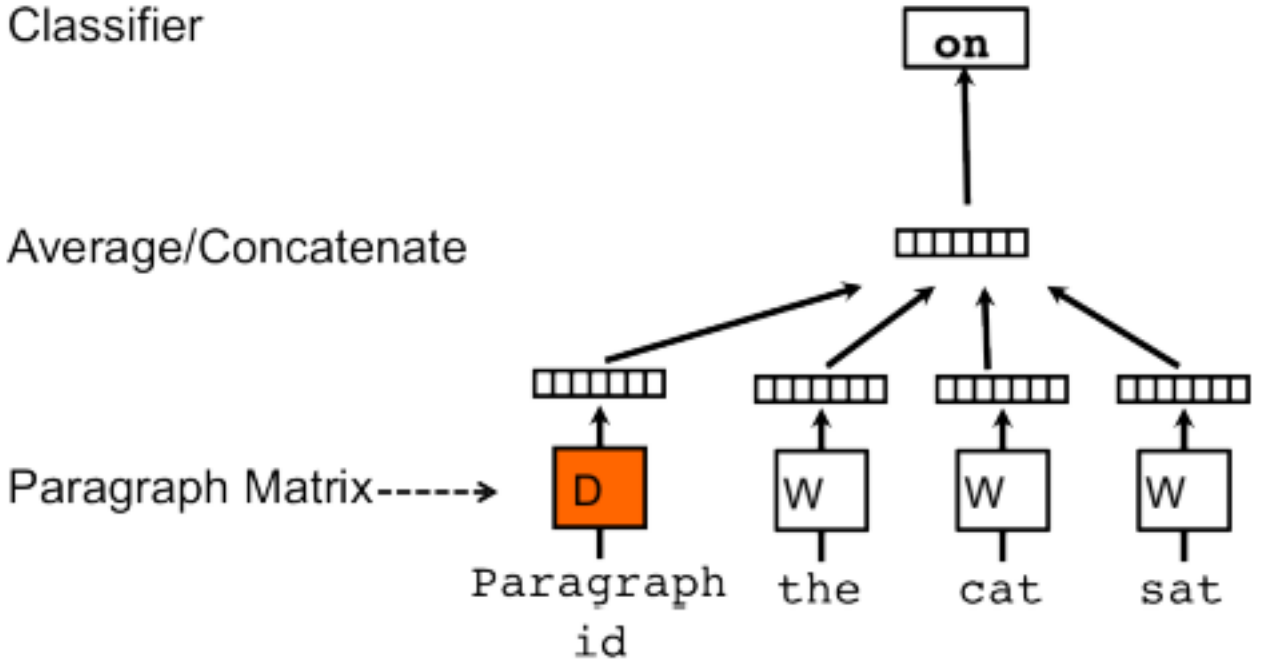


http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Doc2vec (by Mikilov and Le)

https://cs.stanford.edu/~quocle/paragraph_vector.pdf

- Create a numeric representation of a document.
- Use the word2vec model, and added another vector (Paragraph ID). (PV-DBOW)
- The document vector intends to represent the concept of a document.



Example: Video tagging.

Example: Classifying user complaint

- Using doc2vec

https://towardsdatascience.com/multi-class-text-classification-with-doc2vec-logistic-regression-9da9947b43f4

- data set https://catalog.data.gov/dataset/consumer-complaint-database

https://github.com/susanli2016/NLP-with-Python/blob/master/Doc2Vec%20Consumer%20Complaint_3.ipynb

Gensim library

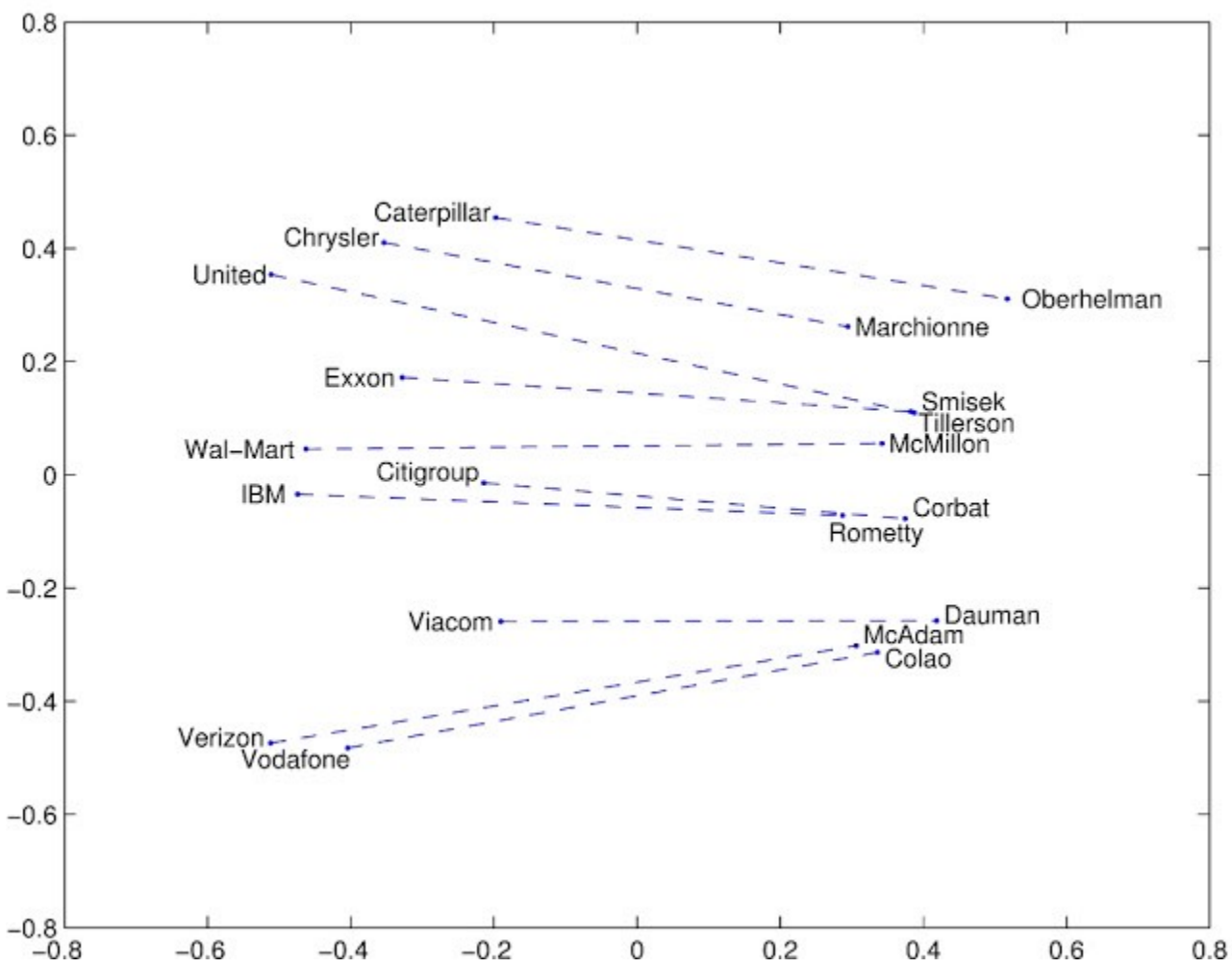
- Create dictionary, corpus
- Download API, dataset : glove
- Create word model, phase model, sentence model, document model (doc2vec), topic model (LDA,LSI)
- Train word2vec model
- Use pretrained model
- Compute TF/IDF, cosine similarity
- Text summarize using TextRank

https://www.machinelearningplus.com/nlp/gensim-tutorial/

GloVe: Global Vectors for Word Representation

https://nlp.stanford.edu/projects/glove/

- Aggregate global word-word co-occurrence matrix from a corpus. The resulting embeddings show interesting linear substructures of the word in vector space.
- Building your glove model:
  - Prepare corpus (nltk)
  - Train your lines
  - Save model



https://medium.com/@japneet121/word-vectorization-using-glove-76919685ee0b

TextRank

https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf

Build on PageRank

- Understanding PageRank
- Initialization:

webpage	links
w1	[w4, w2]
w2	[w3, w1]
w3	[ ]
w4	[w1]

1. Probability of going from page i to j, i.e., M[i][j], is initialized with **1/(number of unique links in web page w_i)**
2. If there is no link between the page i and j, then the probability will be initialized with **0**
3. If a user has landed on a dangling page, then it is assumed that he is equally likely to transition to any page. Hence, M[i][j] will be initialized with **1/(number of web pages)**

M =

	w1	w2	w3	w4
w1				
w2				
w3				
w4				

P(w1 to w2)

M =

	w1	w2	w3	w4
w1	0	0.5	0	0.5
w2	0.5	0	0.5	0
w3	0.25	0.25	0.25	0.25
w4	1	0	0	0

Finally, the values in this matrix will be updated in an iterative fashion to arrive at the web page rankings.

TextRank

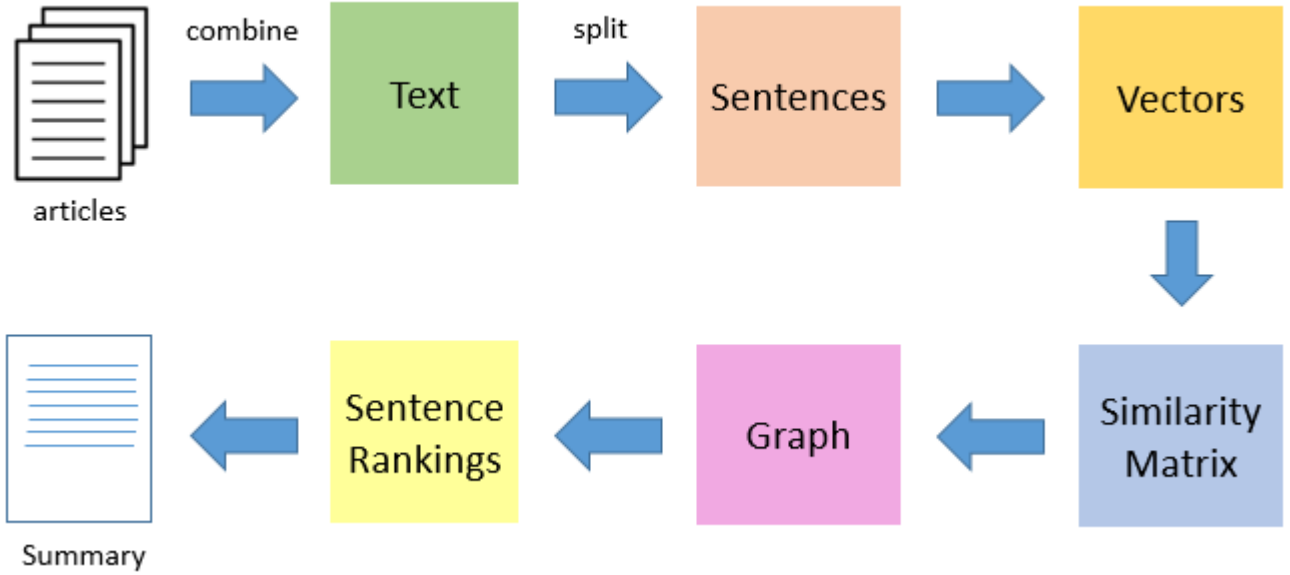
TextRank is an extractive and unsupervised text summarization technique.

- In place of web pages, we **use sentences.**
- Similarity between any two sentences is used as an equivalent to the web page transition probability.
- The similarity scores are stored in a square matrix, similar to the matrix M used for PageRank

https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/

Idea

- The first step would be to concatenate all the text contained in the articles.
- Then split the text into individual sentences.
- In the next step, find vector representation (word embeddings) for each and every sentence.
- Similarities between sentence vectors are then calculated and stored in a matrix.
- The similarity matrix is then converted into a graph, with sentences as vertices and similarity scores as edges, for sentence rank calculation.
- Finally, a certain number of top-ranked sentences form the final summary.



Text generation:

https://github.com/WillKoehrsen/recurrent-neural-networks

Prerequisite: Data download patent abstract from:
http://www.patentsview.org/querydev

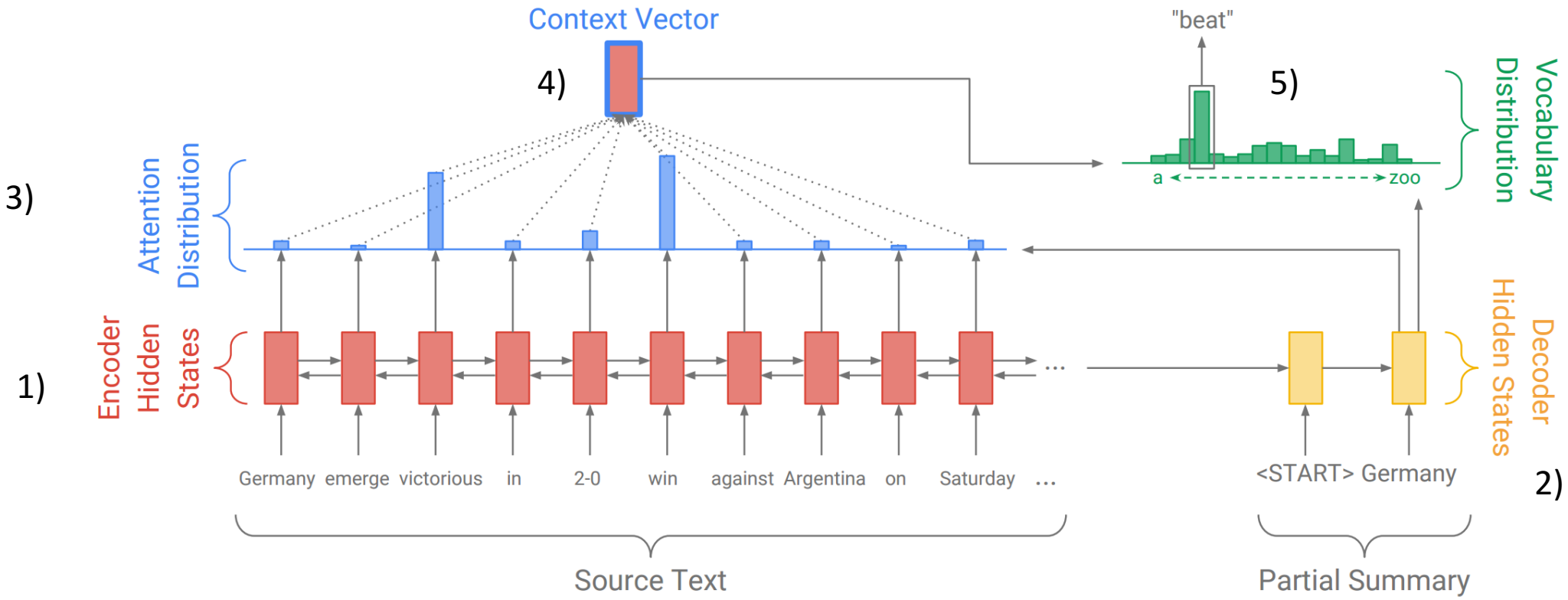
- Read in training data: thousands of "neural network" patents
- Convert patents to integer sequences: tokenization
- Create training dataset using next word following a sequence as label
- Build a recurrent neural network using word embeddings and LSTM cells
- Load in pre-trained embeddings
- Train network to predict next word from sequence
- Generate new abstracts by feeding network a seed sequence
- Repeat steps 2 - 7 using pre-trained embeddings
- Try different model architecture to see if performance improves
- For fun, create a simple game where we must guess if the output is human or computer!

Text Summarization: Two types

- Extractive summarization
  - Easy, correct grammar but cannot be practical for some cases, eg. Summarizing the whole novel.
- Abstractive summarization
  - More difficult

http://www.abigailsee.com/2017/04/16/taming-rnns-for-better-summarization.html

RNN: sequence-to-sequence model with attention



http://www.abigailsee.com/2017/04/16/taming-rnns-for-better-summarization.html

LAB1: Create embedding

- Using word2vec
  - gensim
- Using glove
  - glove-python

Basic: nltk library, nltk corpus

Specify: location of documents for corpus, specify vocab
Preprocess: read data from document as a list of line string. Chop each string to words to become a list of word list.

LAB 2: https://github.com/mishraji-8097/Extractive-Text-Summarization

Extractive Text Summarization

- Using TextRank algorithm
  - Based on Pagrank algorithm
  - https://medium.com/@shivangisareen/text-summarisation-with-gensim-textrank-46bbb3401289

LAB 3:

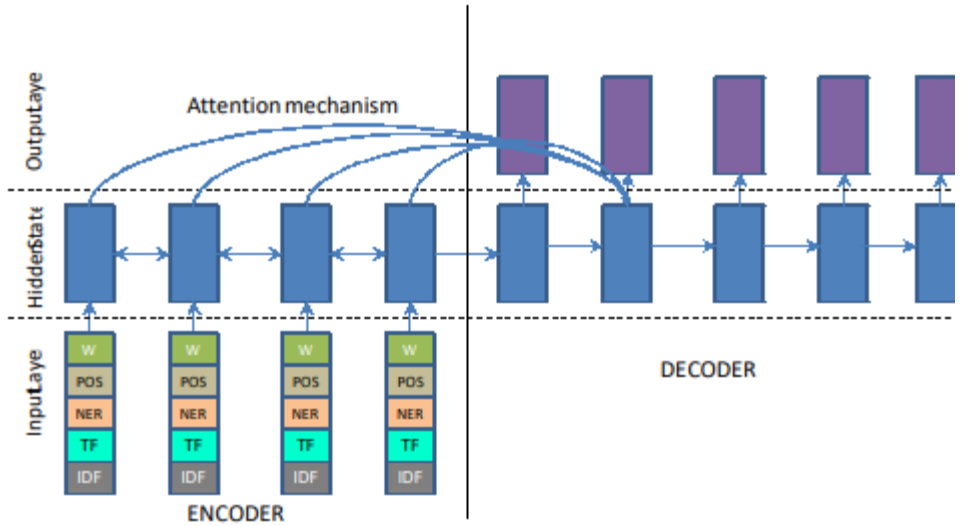
https://github.com/JRC1995/Abstractive-Summarization

Abstractive text summarization

- Download pretrained-glove embedding https://nlp.stanford.edu/projects/glove/
- Download dataset : amazon review https://www.kaggle.com/snap/amazon-fine-food-reviews
- Implementation of word2vec, vec2word, np_nearest_neighbour

Preprocessing steps

- Create vocab_limit,embed_limit
- Insert SOS, EOS marker
- Clean text / save clean text
- Look at stats of data
- Convert to embedding (numeric)
- Create model, train /save model



https://arxiv.org/abs/1602.06023

  - Model is complex: use bidirectional RNN in encoding layer and attention in decoding layer.

https://towardsdatascience.com/text-summarization-with-amazon-reviews-41801c2210b

More example on this.

- https://hk.saowen.com/a/7764607489f9100e4ad9d5586de2464b64399f875e6390b18eb26e1f38598822