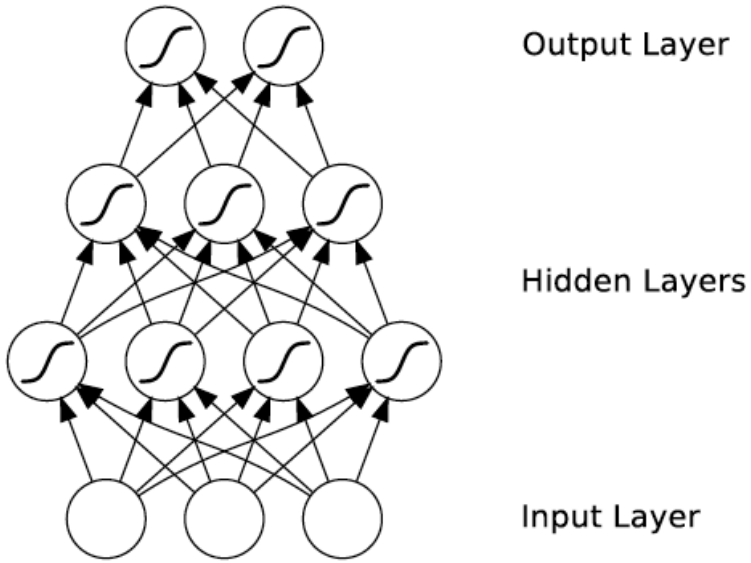
Recurrent Neural Networks

Minlie Huang
aihuang@tsinghua.edu.cn
Dept. of Computer Science and Technology
Tsinghua University
http://coai.cs.tsinghua.edu.cn/hml/

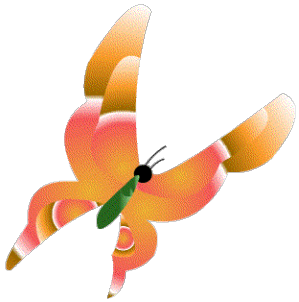
What's wrong with MLPs?

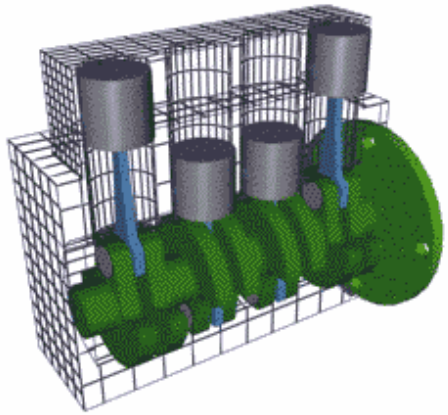
- Problem 1: Can't model sequences
  - Fixed-sized inputs & outputs
  - No temporal structure, thus no dynamics

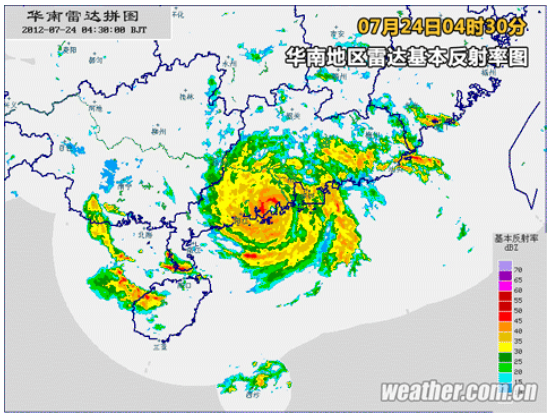
- Problem 2: Pure feed-forward processing
  - No "memory", no feedback

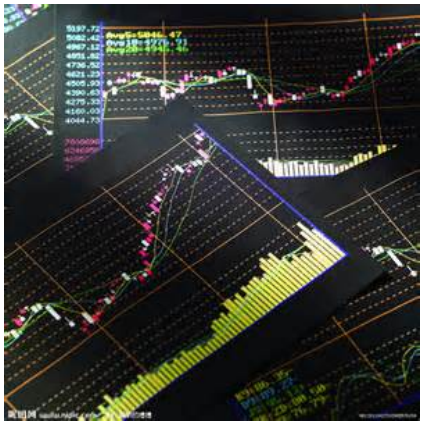


Dynamic Systems





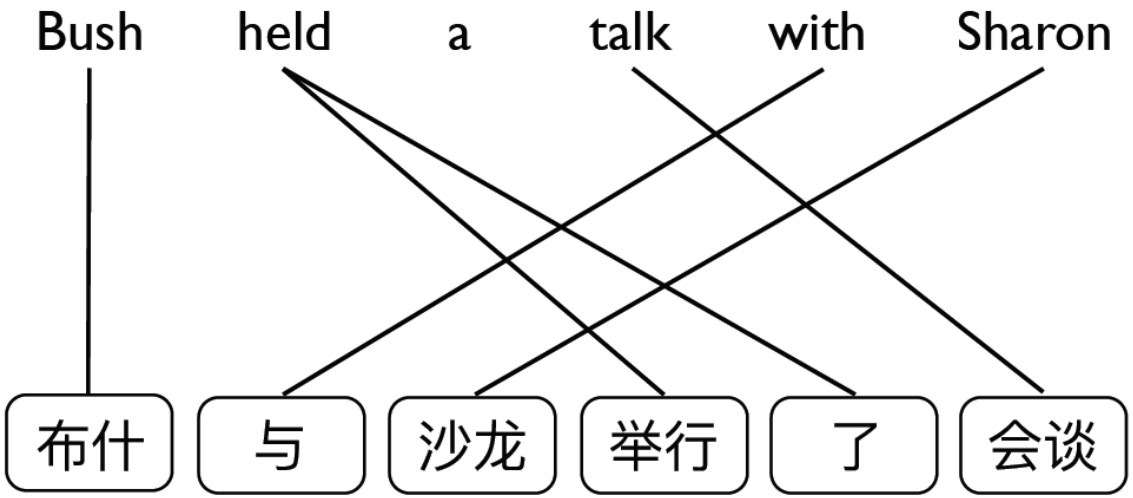




Sequences are everywhere

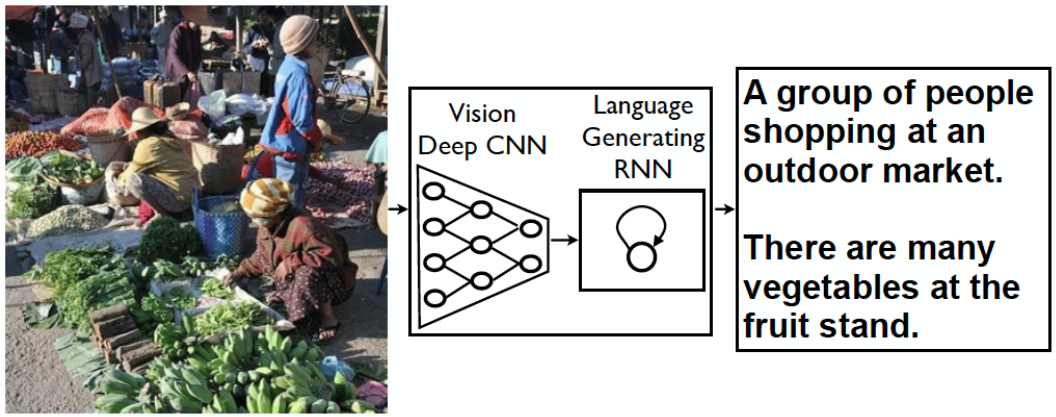
- Signal to natural language

- Translation from English to Chinese



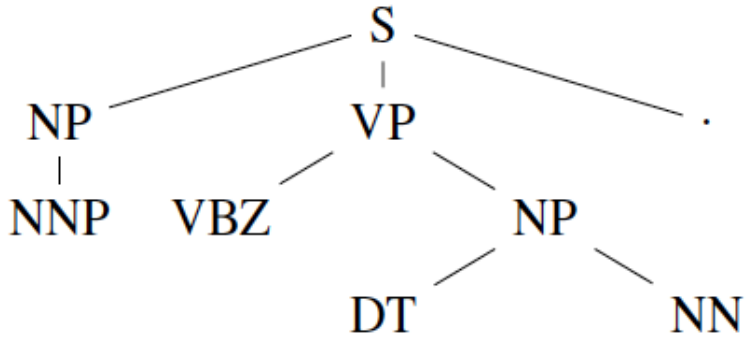
Sequences are everywhere

- Image to description



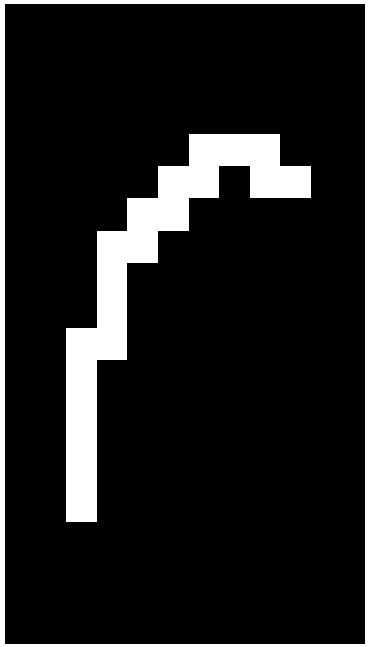
- Natural language to parsing tree

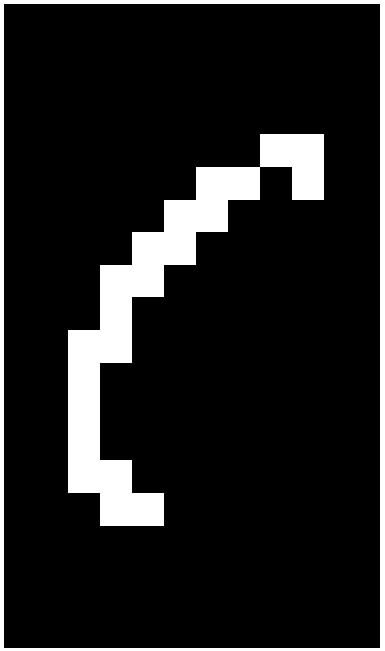
John has a dog . →



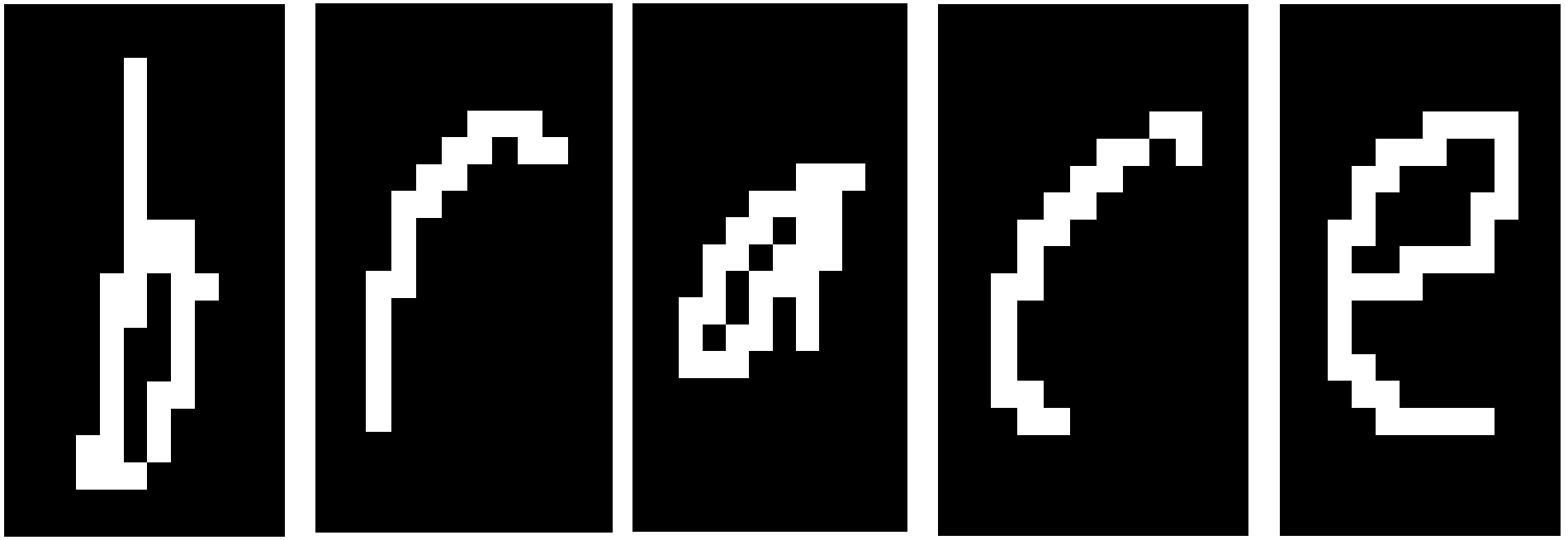
John has a dog . → (S (NP NNP)$_{NP}$ (VP VBZ (NP DT NN)$_{NP}$)$_{VP}$.)$_{S}$

Why model sequences?

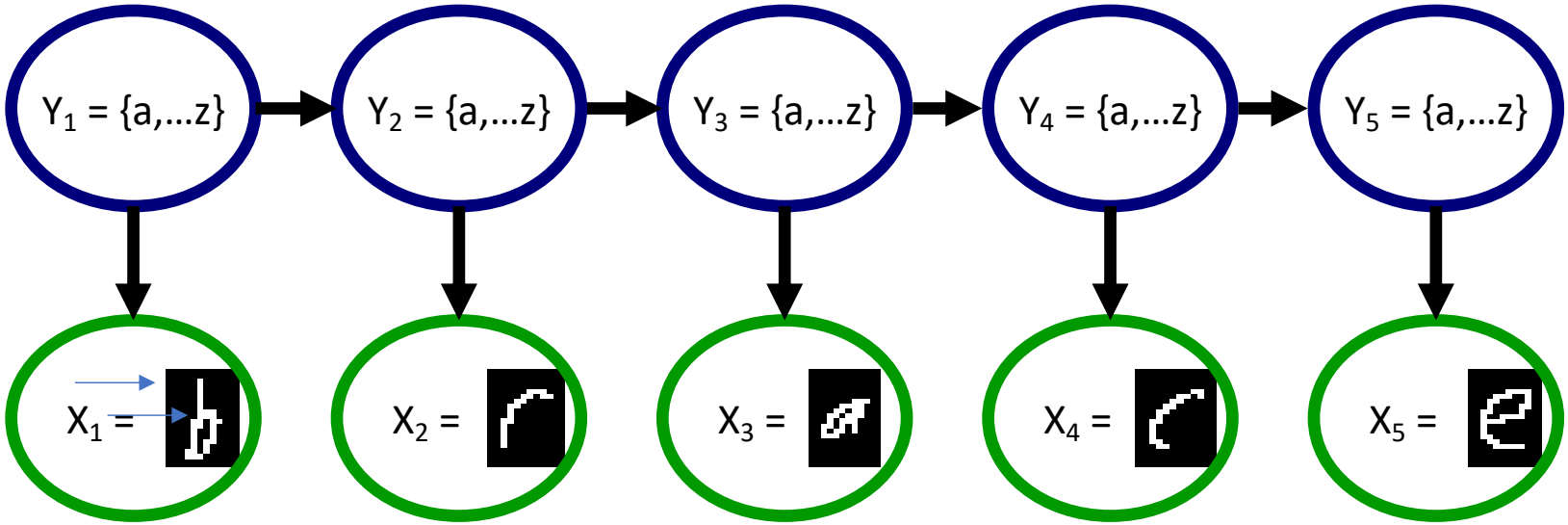




Why model sequences?



Why model sequences?

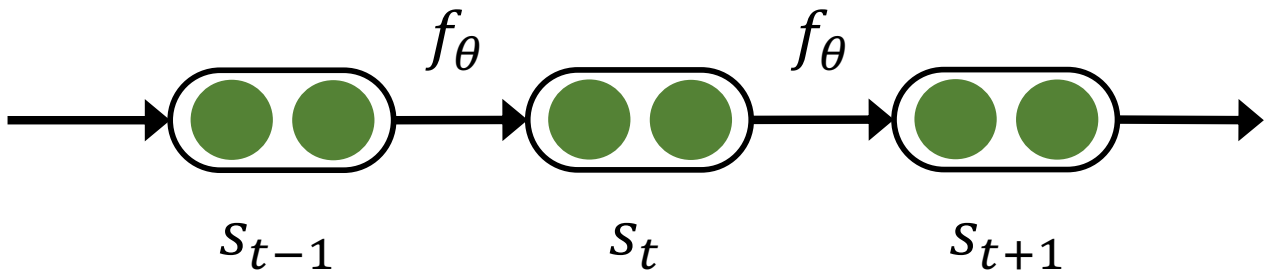


Hidden Markov Model (HMM):

$$P(X_{1\ldots n}, Y_{1\ldots n}) = P(Y_1)P(X_1|Y_1) \cdot \prod_{i=2}^{n} P(Y_i|Y_{i-1})P(X_i|Y_i)$$

How to model sequences?

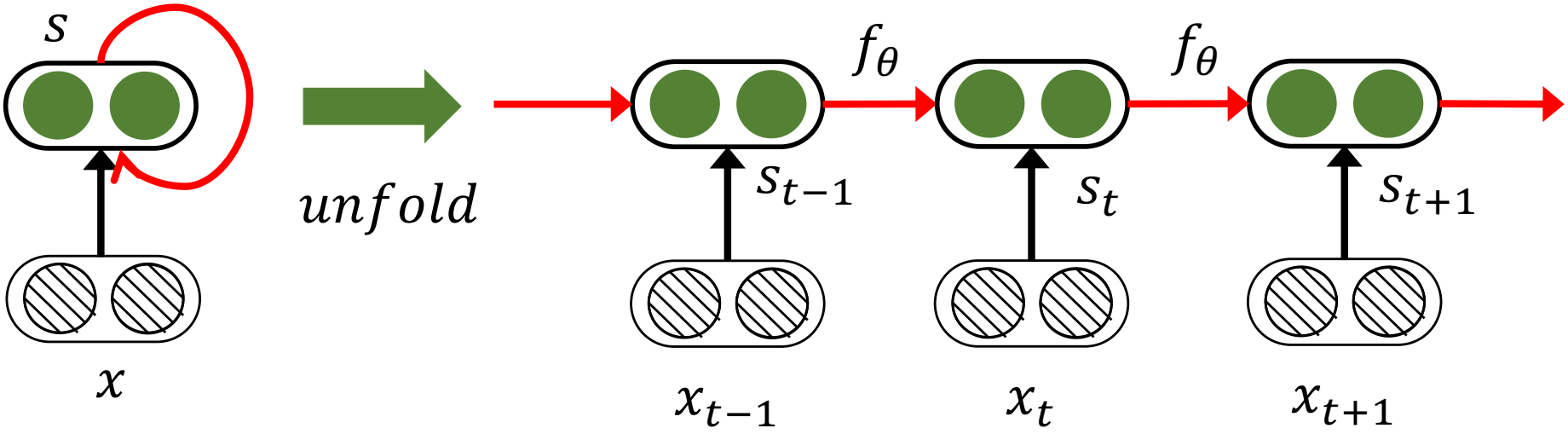
- No input (only states)



$$s_t = f_\theta(s_{t-1})$$

How to model sequences?

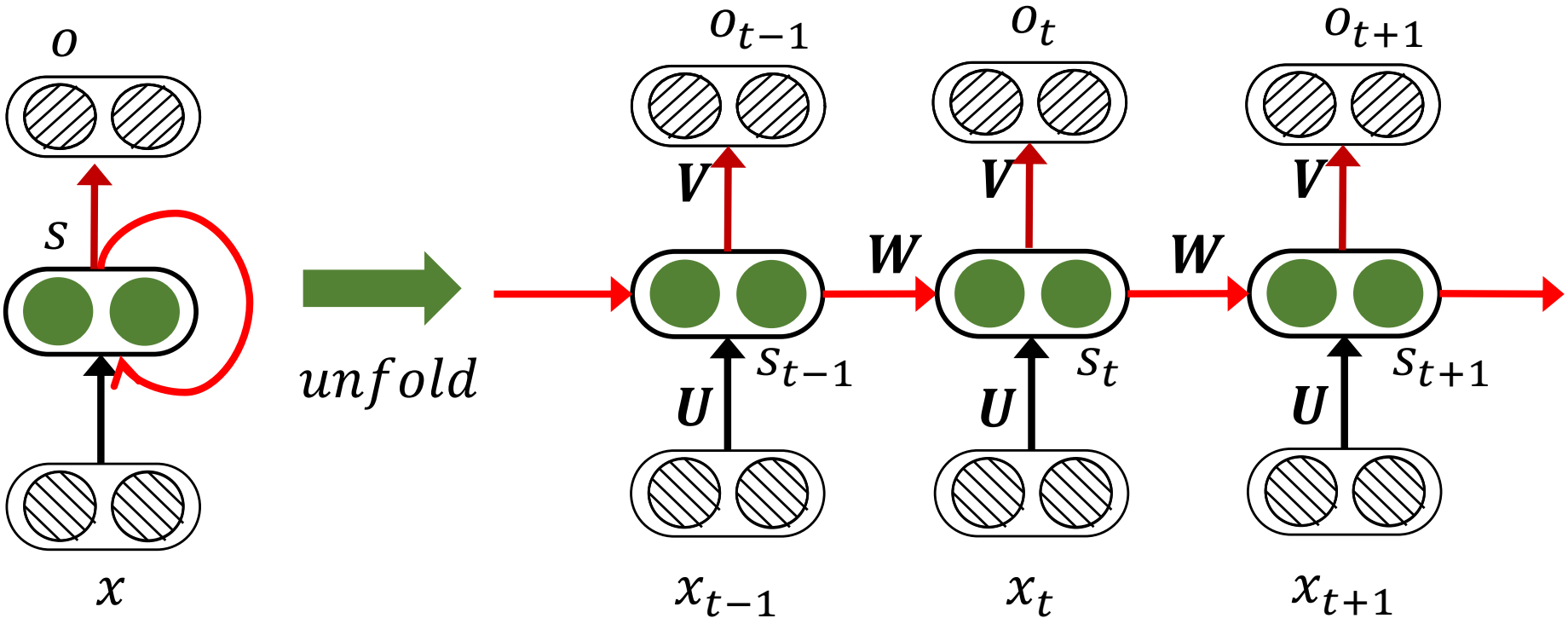
- With inputs (and states)



$$s_t = f_\theta(s_{t-1}, x_t)$$

How to model sequences?

- With inputs and outputs (and states)

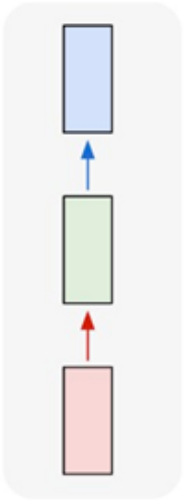


$$s_t = f(W * s_{t-1} + U * x_t)$$

$$o_t = g(V * s_t)$$

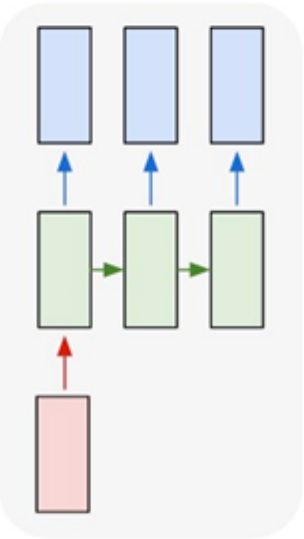
How to model sequences?

one to one



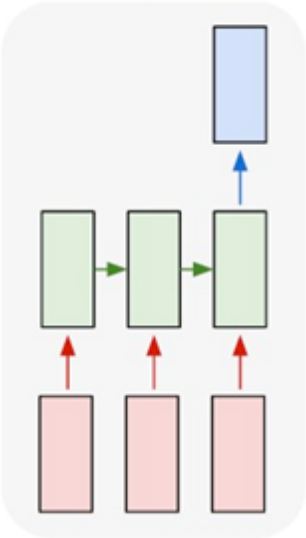
Input: No sequence
Output: No sequence
Example: “standard” classification / regression

one to many



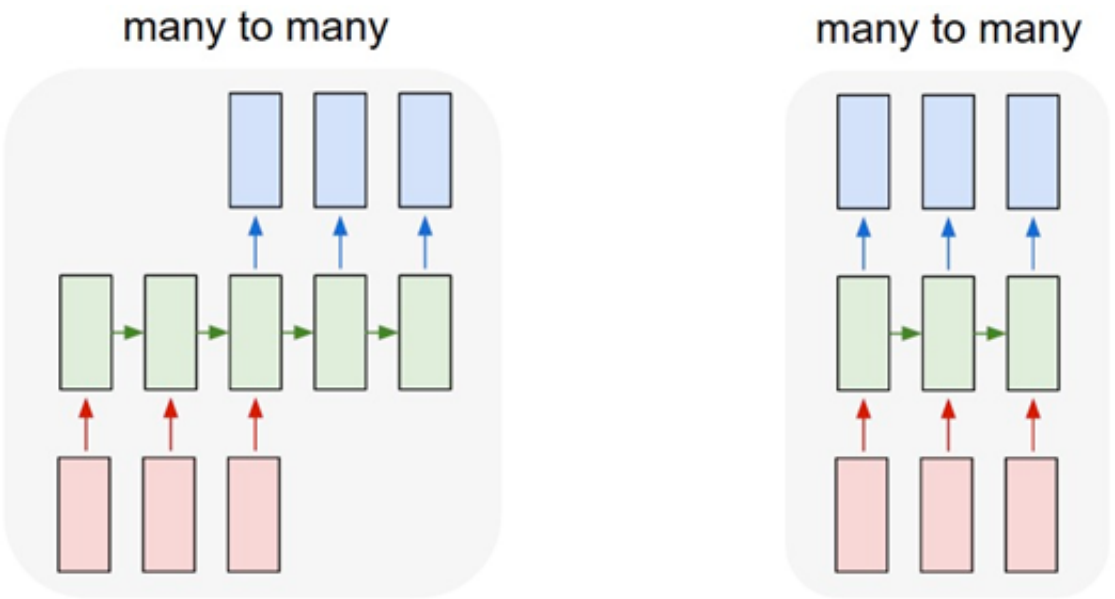
Input: No sequence
Output: Sequence
Example: Image2Caption

many to one



Input: Sequence
Output: No sequence
Example: sentence classi-fication, multiple-choice question answering

How to model sequences?



Input: Sequence, **Output**: Sequence
Example: machine translation, video captioning, open-ended question answering, video question answering

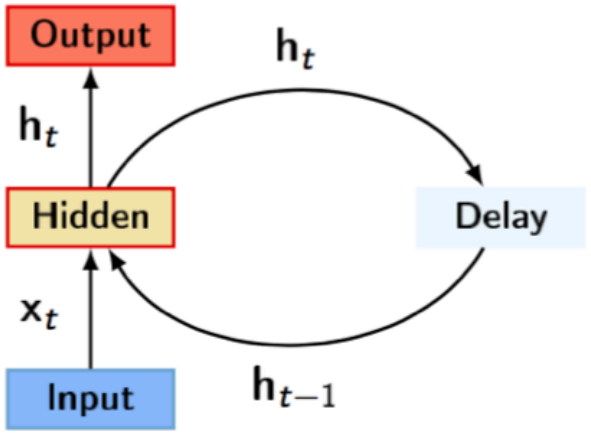
Recurrent Neural Network (RNN)

- Input Layer: A sequence $x(1:n) = (x_1, x_2, \dots, x_t, \dots, x_n)$(**each x_i is a vector**)
- Hidden Layer: RNN has recurrent hidden states whose output at each time is dependent on that of the previous time. RNN updates its recurrent hidden state $h(t)$ by

$$h_t = \begin{cases} 0 & \text{if } t = 0 \\ f(h_{t-1}, x_t) & \text{otherwise} \end{cases}$$

- Output Layer: The output layer takes as input the hidden states and outputs the prediction by

$$y_t = g_r(h_t)$$



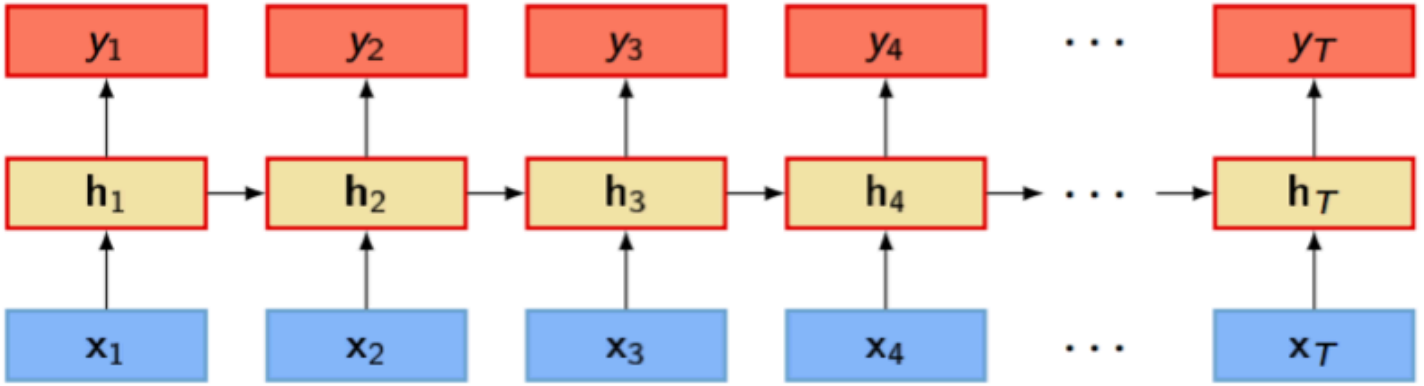
Simple Recurrent Network

- The recurrence function

$$h_t = f(Uh_{t-1} + Wx_t + b)$$

where f is non-linear function (for instance, *sigmoid* or *tanh*). Weight $U \in \mathbb{R}^{d \times d}$, weight $W \in \mathbb{R}^{d \times k}$, hidden state $h \in \mathbb{R}^d$, and input $x \in \mathbb{R}^k$

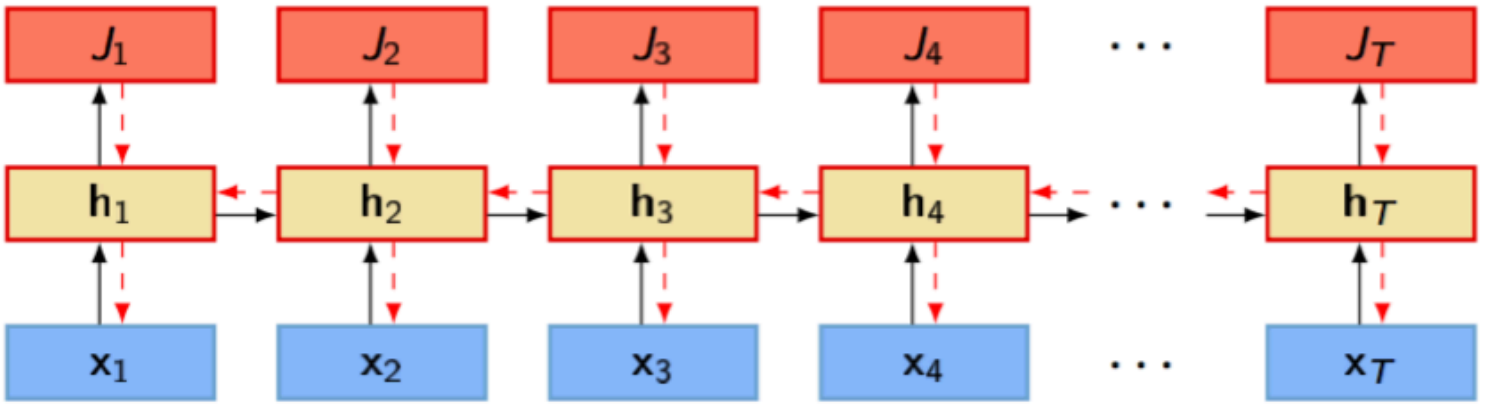
- A typical RNN structure can be illustrated as follows:



Backpropagation Through Time, BPTT

- Suppose loss at time *t* is J_t, then the total loss is $\sum_{t=1}^{T} J_t$. The gradient of *J* is

$$\frac{\partial J}{\partial U} = \sum_{t=1}^{T} \frac{\partial J_t}{\partial U} = \sum_{t=1}^{T} \frac{\partial h_t}{\partial U} \frac{\partial J_t}{\partial h_t}$$



Gradient of RNN

$$\frac{\partial J}{\partial U} = \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial h_k}{\partial U} \frac{\partial h_t}{\partial h_k} \frac{\partial y_t}{\partial h_t} \frac{\partial J_t}{\partial y_t} \tag{4}$$

$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=k+1}^{t} diag[f'(h_{i-1})] * U \tag{5}$$

$$\frac{\partial J}{\partial U} = \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial h_k}{\partial U} \left(\prod_{i=k+1}^{t} diag[f'(h_{i-1})] * U \right) \frac{\partial y_t}{\partial h_t} \frac{\partial J_t}{\partial y_t} \tag{6}$$

Matrix Calculus

$$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} & \cdots & \frac{\partial y}{\partial x_n} \end{bmatrix}$$

Figure: $y = f(x_1, x_2, ..., x_n)$

$$\frac{\partial \mathbf{Y}}{\partial x} = \begin{bmatrix} \frac{\partial y_{11}}{\partial x} & \frac{\partial y_{12}}{\partial x} & \cdots & \frac{\partial y_{1n}}{\partial x} \\ \frac{\partial y_{21}}{\partial x} & \frac{\partial y_{22}}{\partial x} & \cdots & \frac{\partial y_{2n}}{\partial x} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_{m1}}{\partial x} & \frac{\partial y_{m2}}{\partial x} & \cdots & \frac{\partial y_{mn}}{\partial x} \end{bmatrix}$$

Figure: Y is a matrix.

Matrix Calculus

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

Figure: Y, X are column vectors.

$$\frac{\partial \mathbf{F}}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial \mathbf{F}}{\partial X_{1,1}} & \cdots & \frac{\partial \mathbf{F}}{\partial X_{1,n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{F}}{\partial X_{m,1}} & \cdots & \frac{\partial \mathbf{F}}{\partial X_{m,n}} \end{bmatrix}$$

Figure: F, X are matrices.

Derive the Results

$$\overrightarrow{h}_t = f(\overrightarrow{U} * \overrightarrow{h}_{t-1} + \overrightarrow{W} * \overrightarrow{X}_t) \tag{7}$$

where weight $\overrightarrow{U} \in \mathcal{R}^{d \times d}$, weight $\overrightarrow{W} \in \mathcal{R}^{d \times k}$, hidden state $\overrightarrow{h} \in \mathcal{R}^{d}$, and input $\overrightarrow{X} \in \mathcal{R}^{k}$.

$$\begin{bmatrix} \overrightarrow{h}_t[1] \\ \overrightarrow{h}_t[2] \\ \vdots \\ \overrightarrow{h}_t[d] \end{bmatrix} = \begin{bmatrix} f(\overrightarrow{U}_{1*} \cdot \overrightarrow{h}_{t-1} + \overrightarrow{W}_{1*} \cdot \overrightarrow{X}_t) \\ f(\overrightarrow{U}_{2*} \cdot \overrightarrow{h}_{t-1} + \overrightarrow{W}_{2*} \cdot \overrightarrow{X}_t) \\ \vdots \\ f(\overrightarrow{U}_{d*} \cdot \overrightarrow{h}_{t-1} + \overrightarrow{W}_{d*} \cdot \overrightarrow{X}_t) \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_d \end{bmatrix} \tag{8}$$

where each f_i is a scalar since $\overrightarrow{U}_{i*} \cdot \overrightarrow{h}_{t-1} + \overrightarrow{W}_{i*} \cdot \overrightarrow{X}_t$ is a scalar.

$$\frac{\partial \overrightarrow{h}_t}{\partial \overrightarrow{h}_{t-1}} = \begin{bmatrix} \frac{\partial f_1}{\partial \overrightarrow{h}_{t-1}} \\ \frac{\partial f_2}{\partial \overrightarrow{h}_{t-1}} \\ \vdots \\ \frac{\partial f_d}{\partial \overrightarrow{h}_{t-1}} \end{bmatrix} = \begin{bmatrix} \overrightarrow{U}_{1*} \cdot f'(\overrightarrow{h}_{t-1})[1] \\ \overrightarrow{U}_{2*} \cdot f'(\overrightarrow{h}_{t-1})[2] \\ \vdots \\ \overrightarrow{U}_{d*} \cdot f'(\overrightarrow{h}_{t-1})[d] \end{bmatrix} \tag{9}$$

where each $f'(\overrightarrow{h}_{t-1})[i]$ is a scalar, $f'(\overrightarrow{h}_{t-1})$ is a column vector, and $\overrightarrow{U}_{i*}$ is a row vector.

$$\frac{\partial \overrightarrow{h}_t}{\partial \overrightarrow{h}_{t-1}} = \begin{bmatrix} f'(\overrightarrow{h}_{t-1})[1] & & & \\ & f'(\overrightarrow{h}_{t-1})[2] & & \\ & & \ddots & \\ & & & f'(\overrightarrow{h}_{t-1})[d] \end{bmatrix} \cdot \begin{bmatrix} \overrightarrow{U}_{1*} \\ \overrightarrow{U}_{2*} \\ \vdots \\ \overrightarrow{U}_{d*} \end{bmatrix} \tag{10}$$

$$= diag[f'(\overrightarrow{h}_{t-1})] \cdot \overrightarrow{U} \tag{11}$$

Gradient Vanishing/Explosion

Define $\gamma = ||diag[f'(h_{i-1})] * U||$,

- Exploding Gradient Problem: When $\gamma > 1$ and $t - k \to \infty$, $\gamma^{t-k} \to \infty$.
- Vanishing Gradient Problem: When $\gamma < 1$ and $t - k \to \infty$, $\gamma^{t-k} \to 0$.

There are various ways to solve Long-Term Dependency problem.

Vanishing/Exploding Gradient Problem

- Consider a ***linear*** recurrent net with zero inputs

$$h_t = W_{hh} h_{t-1} = W_{hh}^t h_0$$

- Singular value of W < 1 ⇒ gradient Vanishes
- Singular value of W > 1 ⇒ gradient Explodes

- "It is **sufficient** for the largest eigenvalue λ_1 of the recurrent weight matrix to be smaller than 1 for long term components to vanish (as t → ∞) and **necessary** for it to be larger than 1 for gradients to explode."

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." IEEE transactions on neural networks 5.2 (1994): 157-166.

Vanishing/Exploding Gradient Problem

- Gradient norm clipping (Mikolov thesis 2012; Pascanu, Mikolov, Bengio, ICML 2013)

$$\hat{g} \leftarrow \frac{\partial \mathcal{J}}{\partial \theta}$$

$$\text{IF} \quad ||\hat{g}|| > \textit{Threshold}$$

$$\text{THEN} \quad \hat{g} \leftarrow \frac{\textit{Threshold}}{||\hat{g}||} * \hat{g}$$

- Gradient propagation regularizer (avoid vanishing gradient)

Connections to MLP?

- Why is RNN a deep structure?

Connections to MLP?

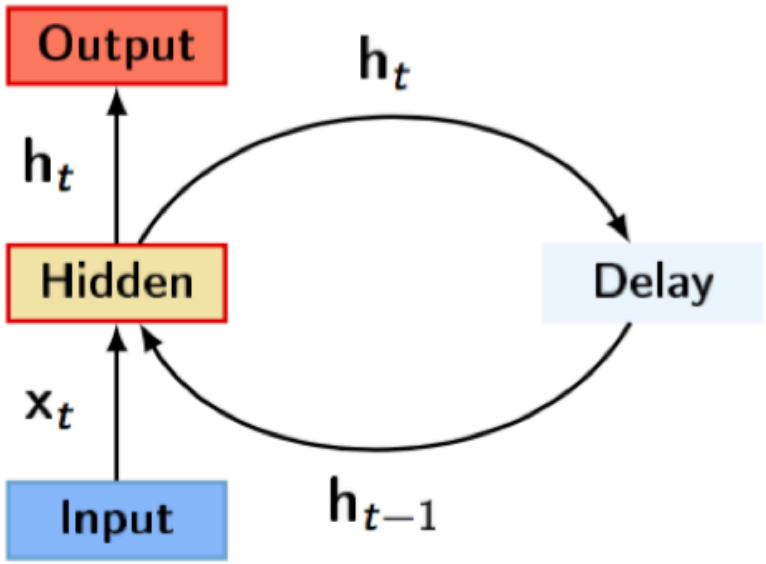


Figure: Traditional View.

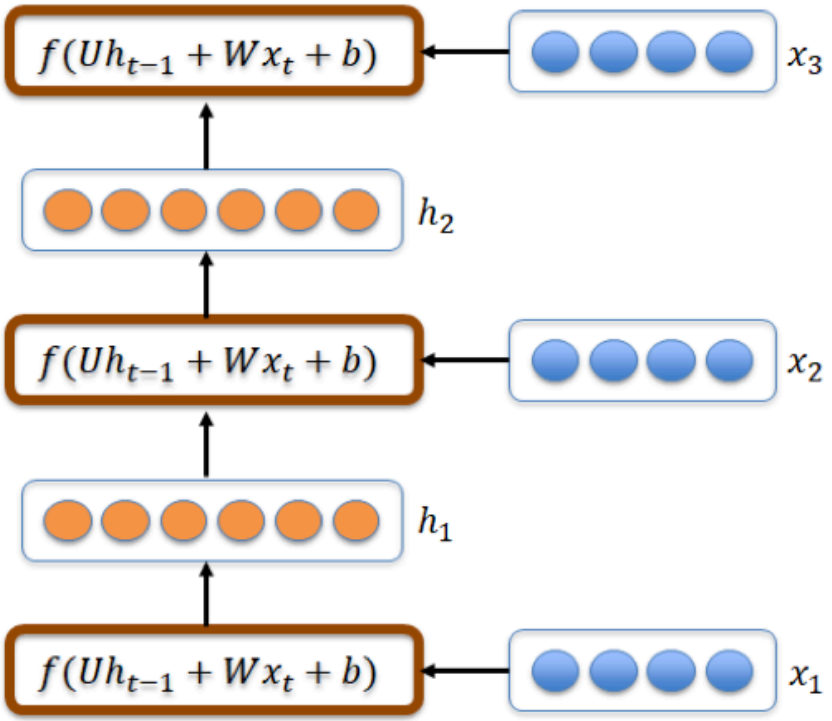


Figure: Deep View.

Long Short-Term Memory (LSTM)

Key difference to RNN: a memory cell *c* which is controlled by three gates:

- Input gate *i*:

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

- Output gate *o*:

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

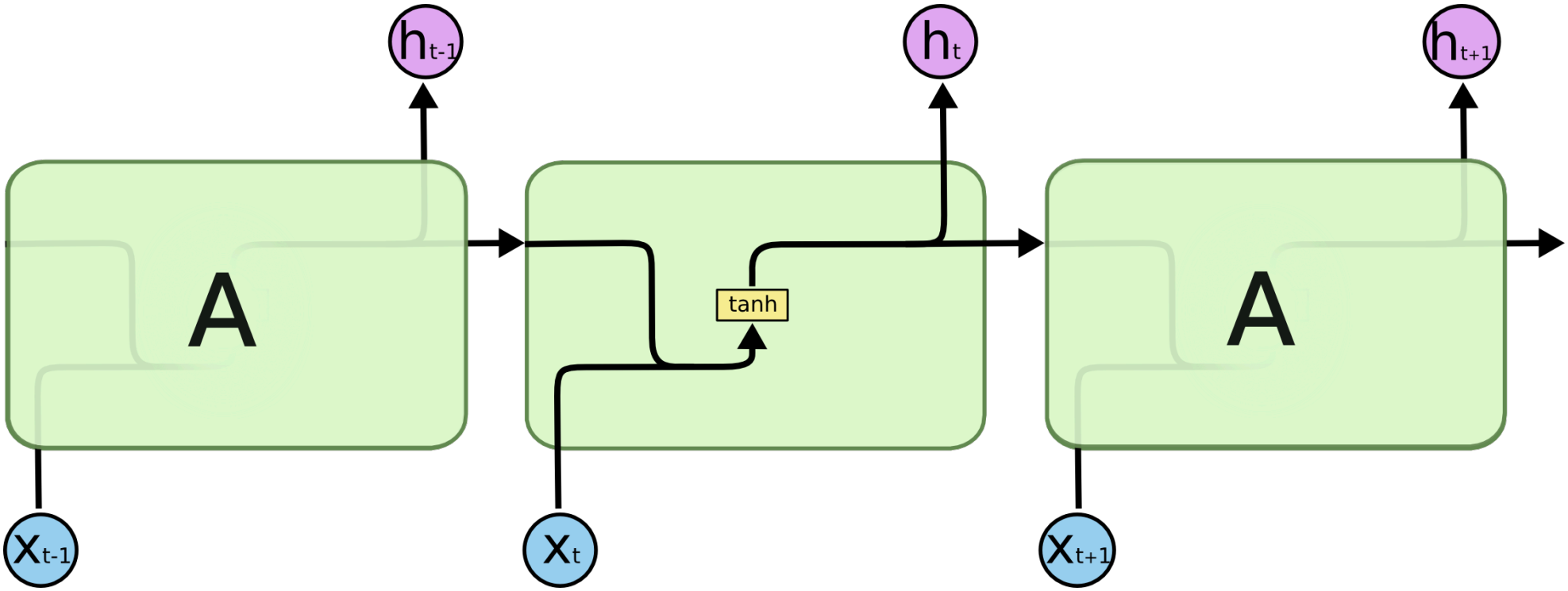
- Forget gate *f*:

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

- Update of memory cell and hidden state:

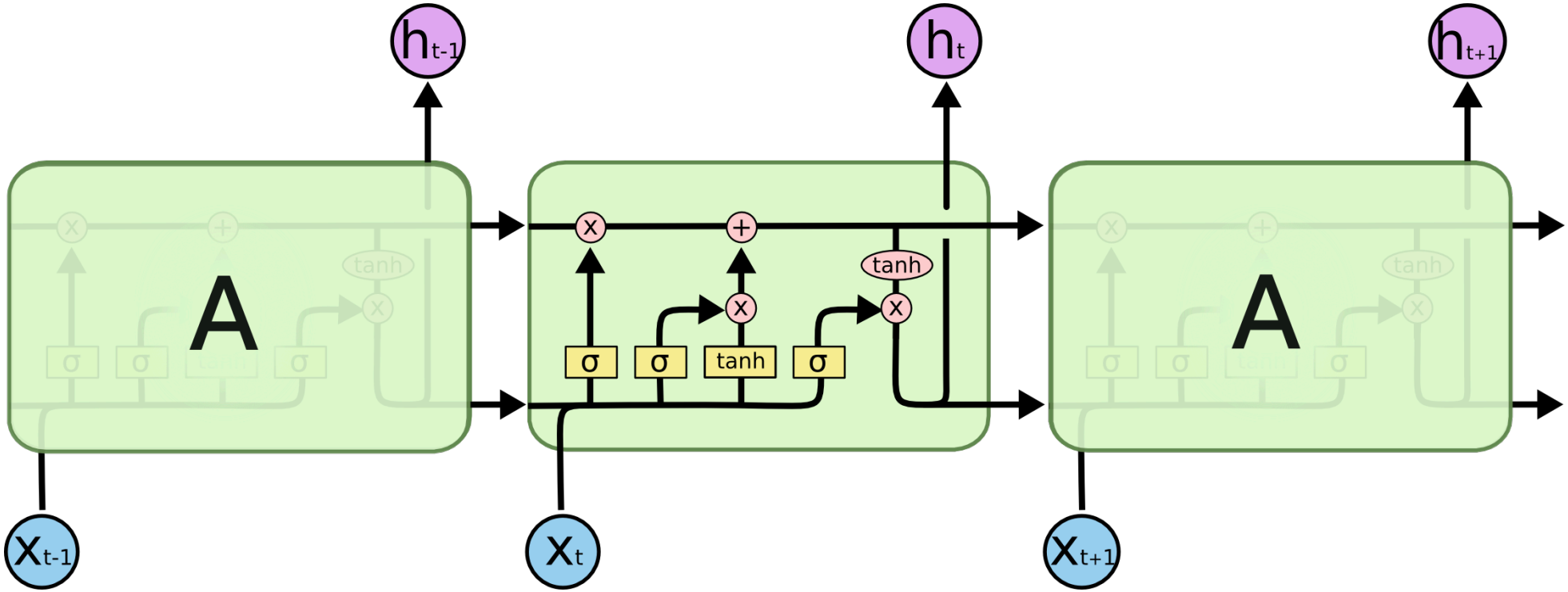
$$\widetilde{C}_t = \tanh(W_C x_t + U_C h_{t-1} + b_C)$$
$$C_t = f_t \otimes C_{t-1} + i_t \otimes \widetilde{C}_t$$
$$h_t = o_t \otimes \tanh(C_t)$$

Standard RNN vs. LSTM



The pictures are from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Standard RNN vs. LSTM



The pictures are from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Empirical Feasible Length of LSTM

RNN

100

LSTM

500

Gated Recurrent Unit (GRU)

Instead of using three gates, only two gates are employed:

- Update gate *z*:

$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$

- Reset gate *r*:

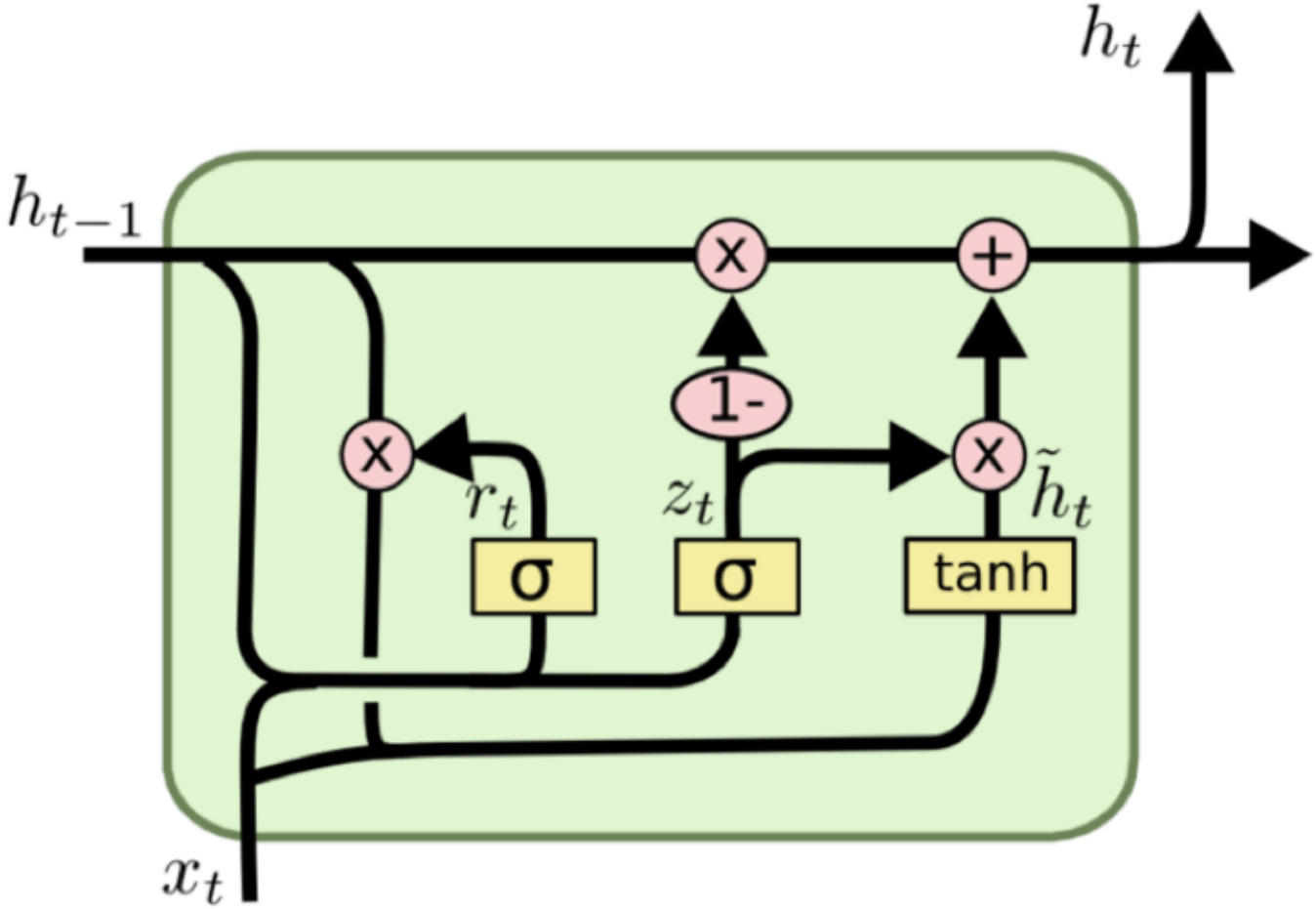
$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$

- State update:

$$\widetilde{h_t} = \tanh(W_C x_t + U(r_t \otimes h_{t-1}))$$

$$h_t = (1 - z_t) \otimes h_{t-1} + z_t \otimes \widetilde{h_t}$$

GRU Architecture

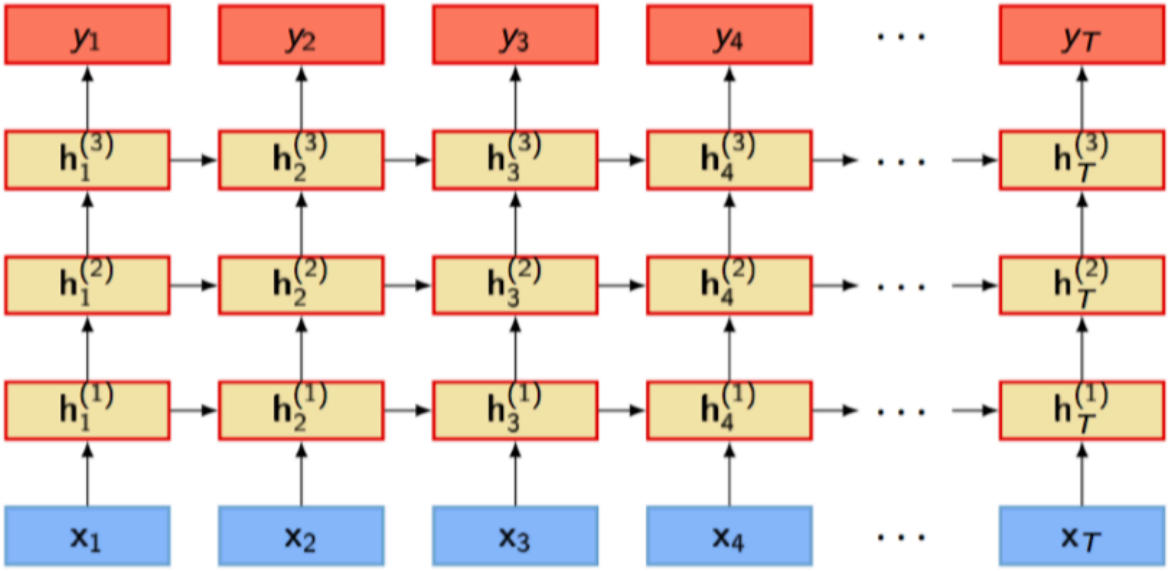


Empirical Feasible Length of GRU

RNN	LSTM	GRU
100	500	1000

Stacked RNN

- Stack multiple RNN/LSTM in vertical direction



- More common is seen that raw input (x_i) is connected to all layers as input.

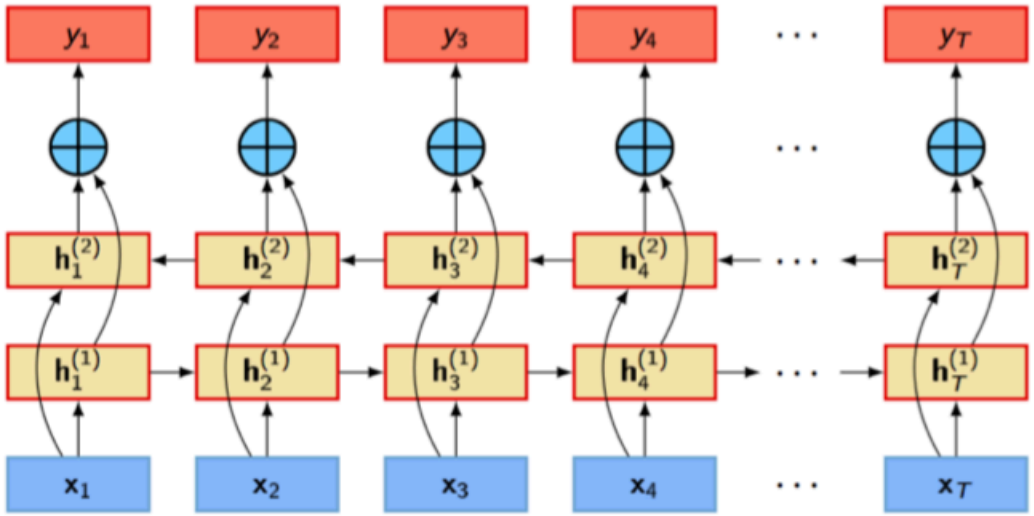
Bidirectional RNN

- Modeling the prefix context and suffix context at the same time.

$$h_t^{(1)} = f(U^{(1)} h_{t-1}^{(1)} + W^{(1)} x_t + b^{(1)})$$

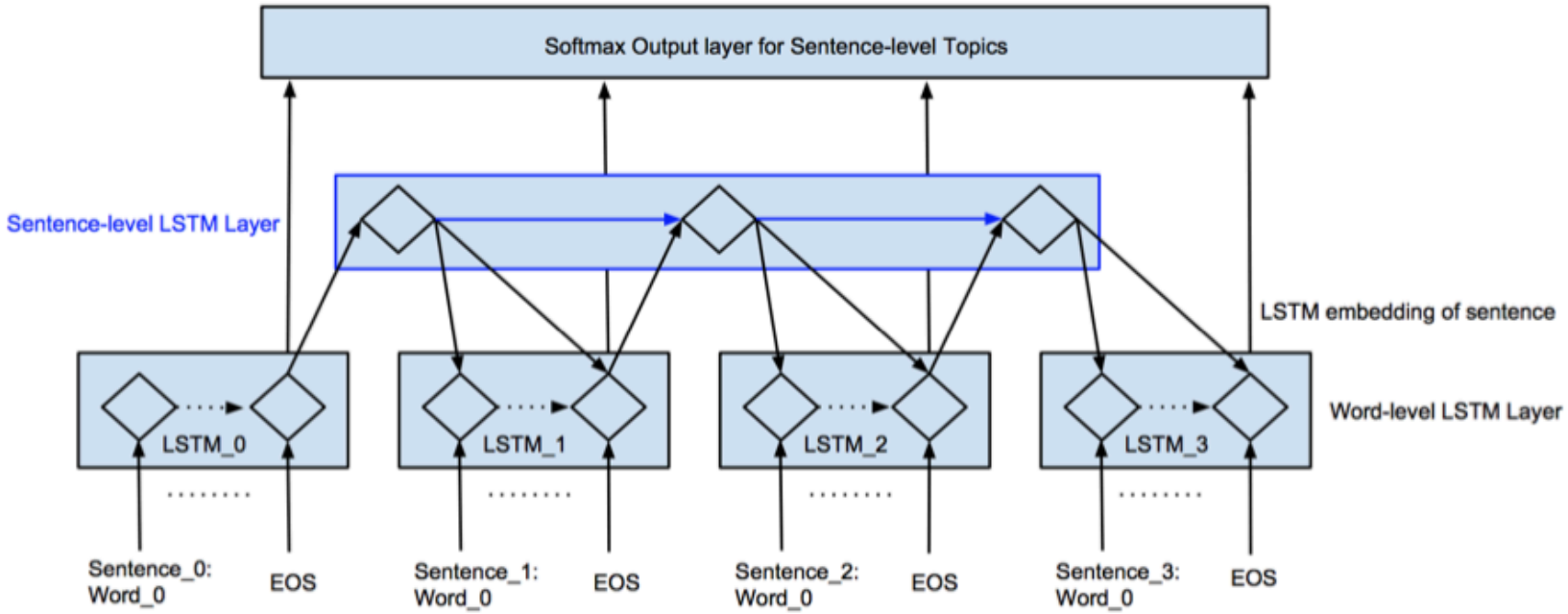
$$h_t^{(2)} = f(U^{(2)} h_{t+1}^{(2)} + W^{(2)} x_t + b^{(2)})$$

$$h_t = h_t^{(1)} \oplus h_t^{(2)}$$



Hierarchical LSTM

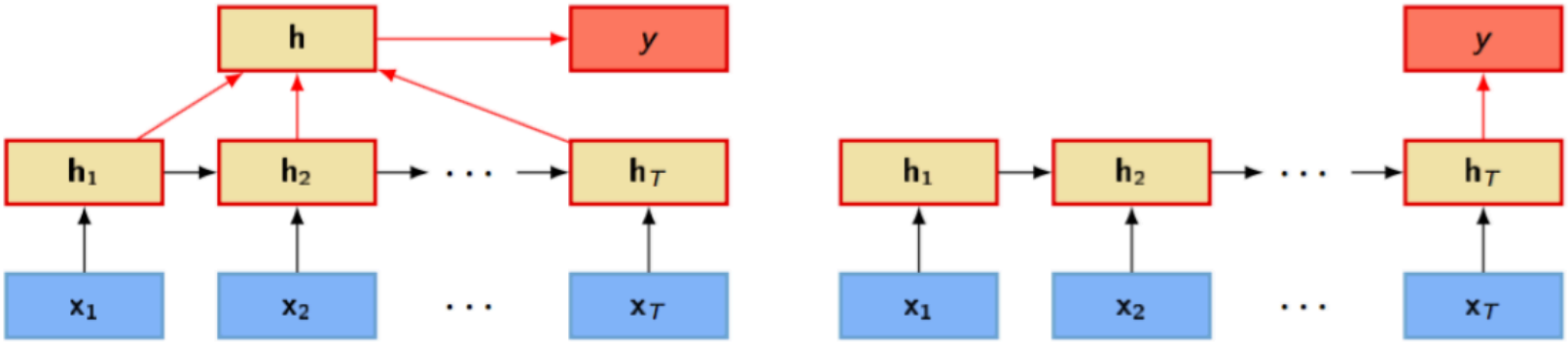
- Hierarchical LSTM (low-level LSTMs are taken as input to the high-level LSTM)



Ghosh S, Vinyals O, Strope B, et al. Contextual LSTM (CLSTM) models for Large scale NLP tasks[J]. arXiv prepint arXiv:1602.06291, 2016.

Application: Classification

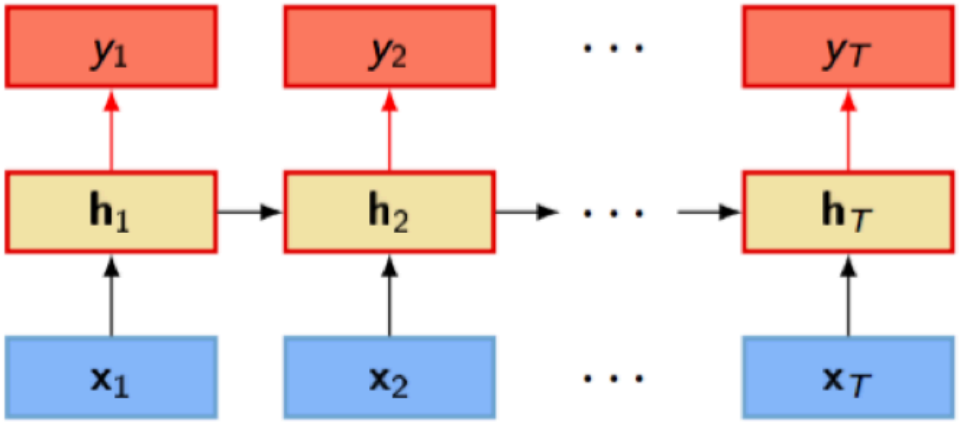
- Applicable to many classification problems:
  - Text Classification
  - Sentiment Classification



- The left: mean (can be enhanced with attention mechanisms)
- The right: last

Application: Sequence Labeling

- Sequence Labeling, such as Chinese word segmentation, part-of-speech tagging, semantic role labeling



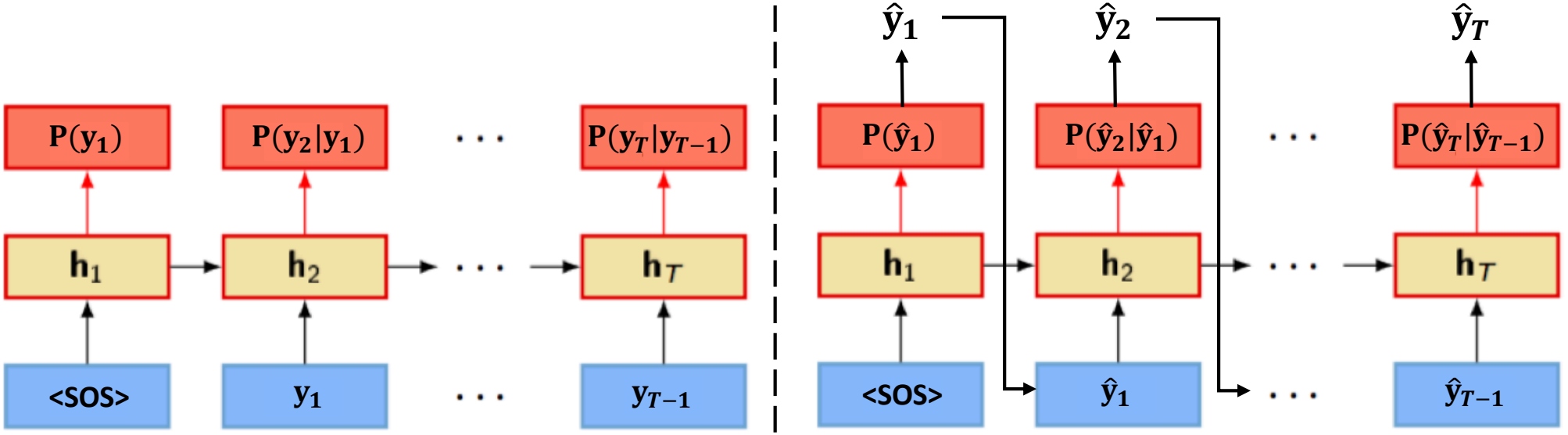
- Linguistic Tools:
  - Natural Language Toolkit (NLTK)
  - Stanford CoreNLP
  - THU Lexical Analyzer for Chinese (THULAC)

Application: RNNLM

- Text generation
- Language modelling:

$$P(Y) = P(y_1, y_2, \dots, y_T) = \prod_{t=1}^{T} P(y_t|Y_{<t}) = \prod_{t=1}^{T} \text{softmax}\left(W_0 h_t + b_0\right)|_{y_t}$$

- Optimization: $\theta = \text{argmax}_\theta \, P(Y;\theta) = \text{argmax}_\theta \sum_{t=1}^{T} \log P(y_t|Y_{<t};\theta)$



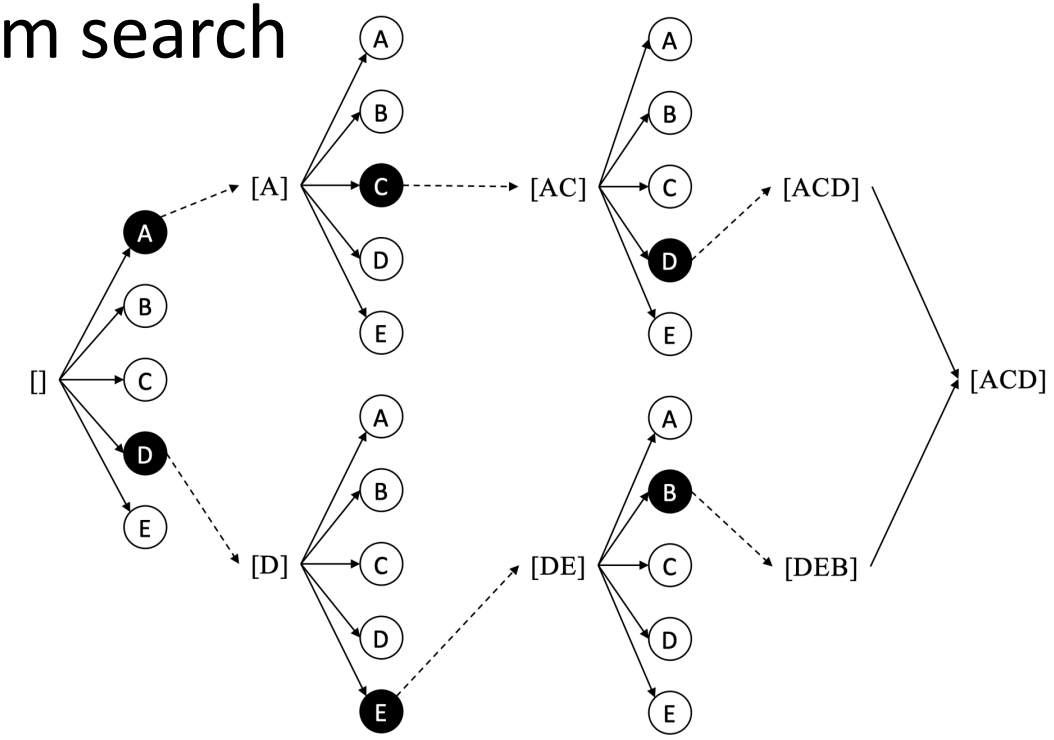
Training: Teacher Forcing

Inference: Free-run

Application: RNNLM

- Decoding algorithms: generation from $P(y_t|Y_{<t})$ by search
  - Greedy search

    $\hat{y}_t = \text{argmax}_y P(y|\hat{Y}_{<t}, X)$
  - Beam search



Application: RNNLM

- Decoding algorithms: generation from $P(y_t|Y_{<t})$ by sampling
  - Random sampling $\hat{y}_t \sim P(y|\hat{Y}_{<t}, X)$
  - Tempered sampling
  - Top-k sampling

$$\mathcal{V}^{(k)} = \text{argmax}_{\mathcal{V}^{(k)}} \sum_{y \in \mathcal{V}^{(k)}} P(y|\hat{Y}_{<t}, X)$$

$$\tilde{P}(y_t|\hat{Y}_{<t}, X) = \begin{cases} \frac{P(y_t|\hat{Y}_{<t}, X)}{\sum_{y \in \mathcal{V}^{(k)}} P(y|\hat{Y}_{<t}, X)}, & y_t \in \mathcal{V}^{(k)} \\ 0, & \text{otherwise} \end{cases}$$

$$\hat{y}_t \sim \tilde{P}(y_t|\hat{Y}_{<t}, X)$$

  - Top-p sampling (nucleus sampling)

Language Models

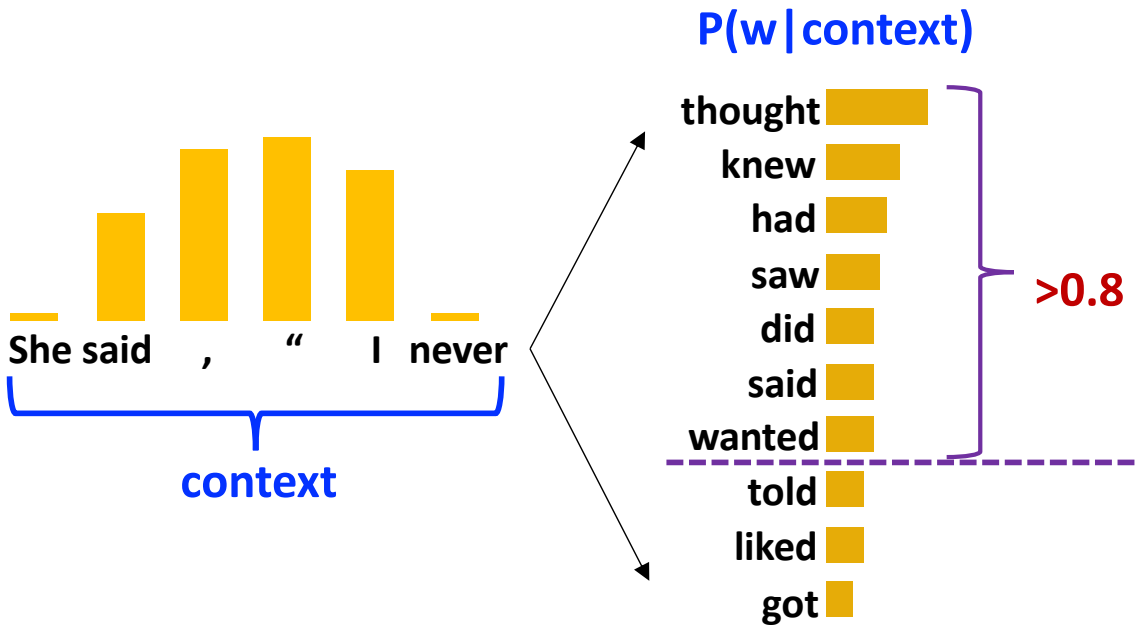
- Language modeling problem: $\boldsymbol{P(Y|X) = \prod_t P(y_t|y_1 y_2 \cdots y_{t-1}, X)}$
  - **X** ---Input
  - **Y** ---Output to be generated
- Optimizing with maximum likelihood estimation (MLE): $\boldsymbol{max \sum_{(X,Y)} log P(Y|X)}$
- Auto-regressive text generation
  - Each time sample $\boldsymbol{y_t}$ from $\boldsymbol{P(y|y_1 y_2 \cdots y_{t-1}, X)}$
  - Feed $\boldsymbol{y_t}$ into the input again to generate $\boldsymbol{y_{t+1}}$

$$\boldsymbol{P(y|y_1 y_2 \cdots y_{t-1}, X) = \mathbf{softmax}\{\mathbf{W} * s_t\}} \in \mathrm{R}^{|V|}$$

Sampling

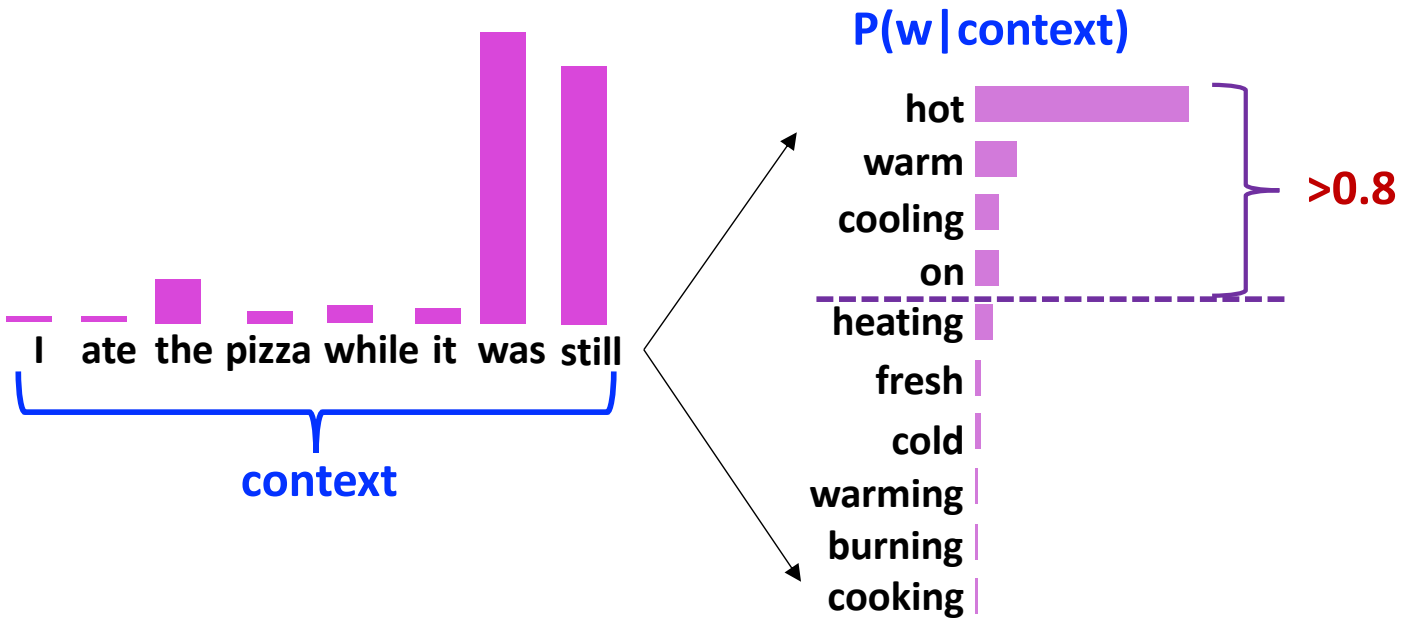
- Generation through sampling from the distribution $y_t \sim P(y|y_1 y_2 \cdots y_{t-1}, X),\ y \in V, |V| \sim 50,000$
- Sampling can lead to **long-tail noises (rare words)**
  - Remarkably degrading the generation quality
- Possible Solutions
  - Greedy sampling: choosing the word with the largest prob. each time
  - Beam search then re-rank
  - Top-***k*** sampling from the ***k*** words with the largest ***k*** probs.
  - Top-***p*** sampling from the top words whose prob. sum is larger than ***p***.

Nucleus Sampling (Top-*p*)



Holtzman et al. 2019. The Curious Case of Neural Text Degeneration

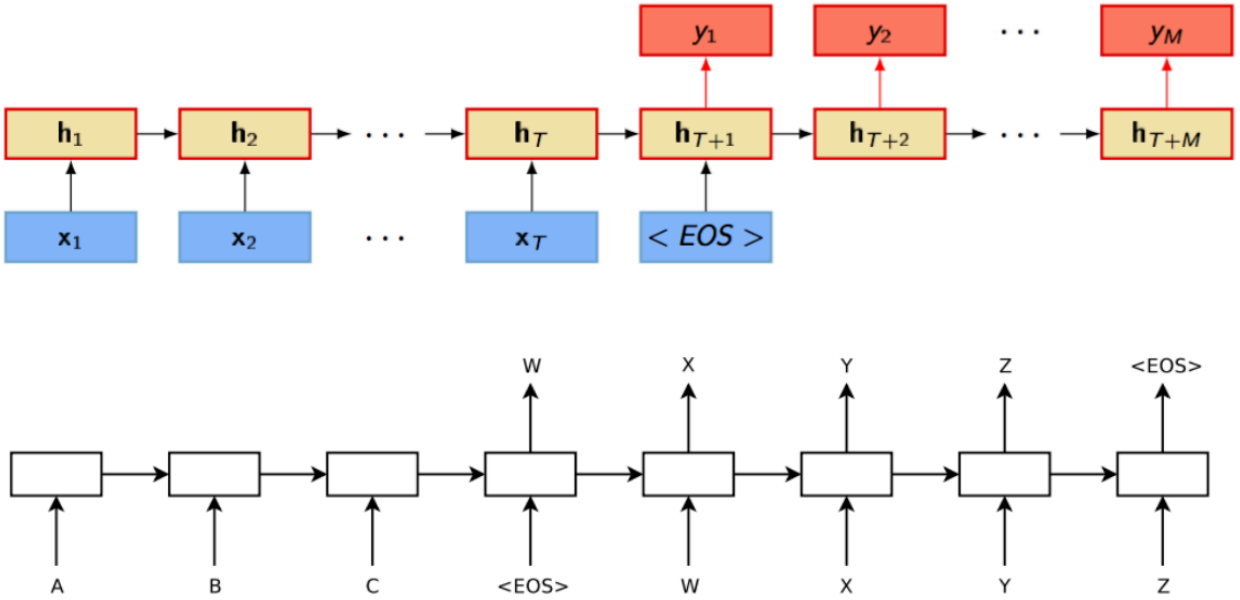
Nucleus Sampling (Top-*p*)



Holtzman et al. 2019. The Curious Case of Neural Text Degeneration

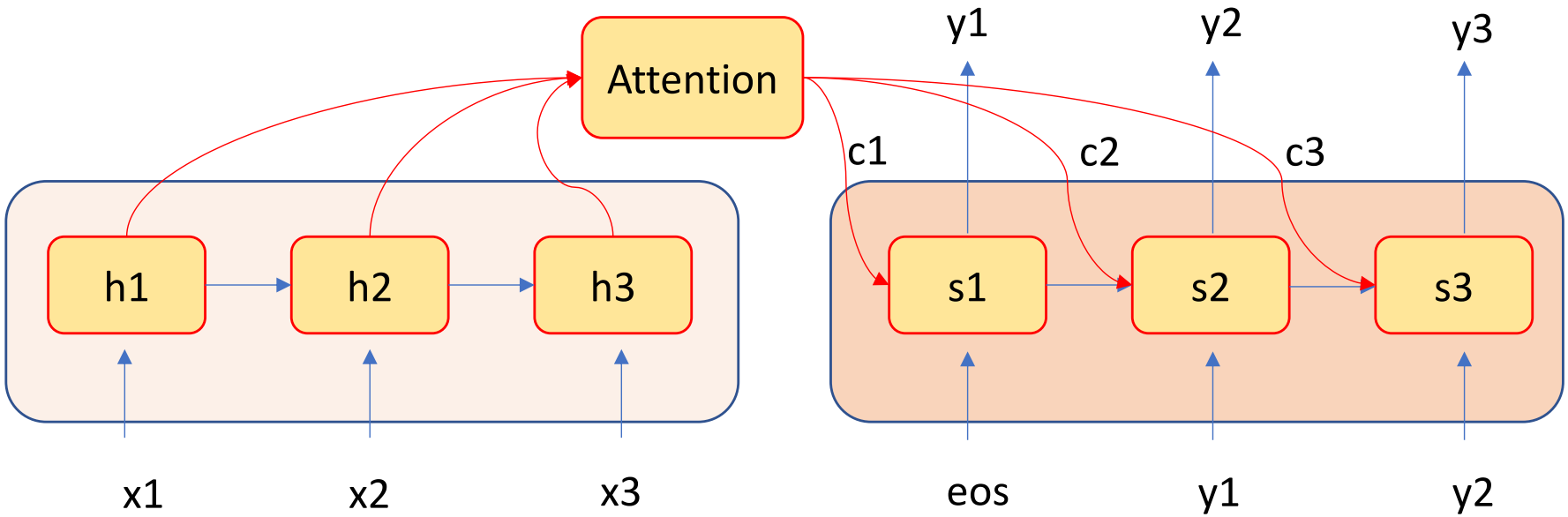
Application: Sequence to Sequence Learning

- Machine Translation
- Sequence to sequence generation (for dialogue, question answering, etc.)
- Image captioning: from image to text



Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks[C]//NIPS. 2014: 3104-3112

Encoder-Decoder Framework

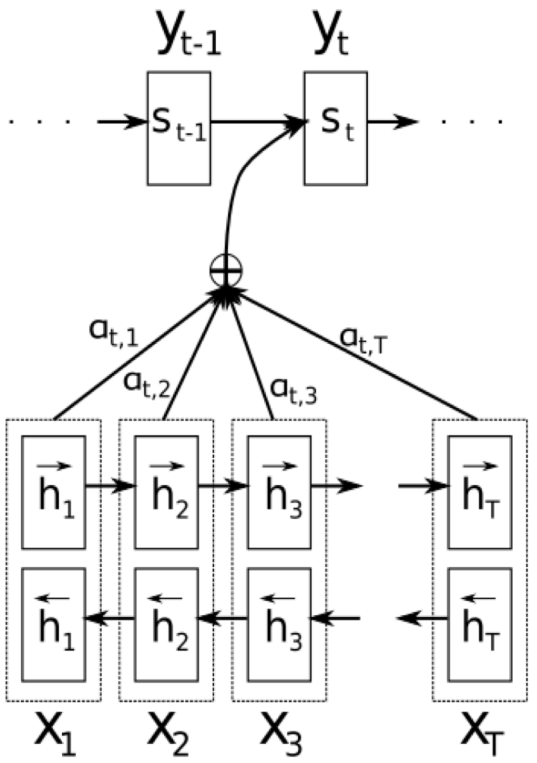


What is Attention?

- Differentiate entities by its importance
  - spatial attention is related to location
  - temporal attention is related to causality

$$\sum \alpha_i x_i$$

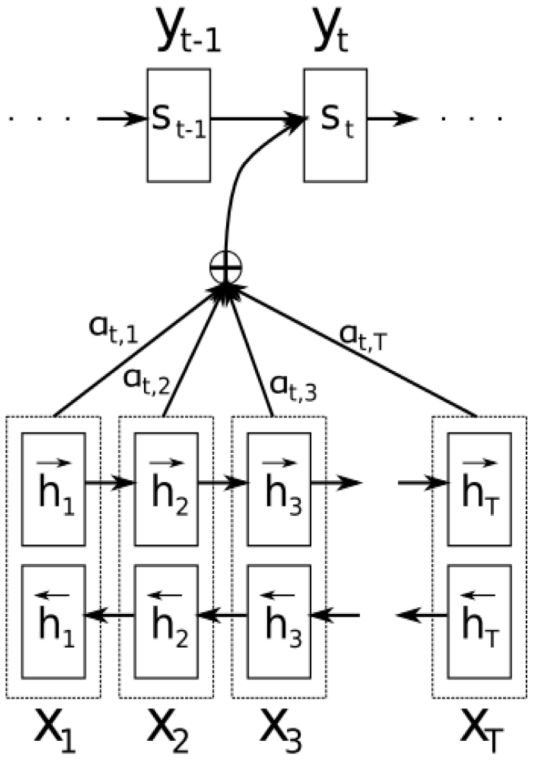
$$0 \leq \alpha_i \leq 1$$



NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE ICLR2015

What is Attention?

- To decode state s_t, attend to encoder states $h_{1\sim j}$
  - Similarity function: $e_{tj} = a(s_{t-1}, h_j)$
    - $a(\cdot)$---a parameterized function (MLP, or bilinear)
  - Softmax: $\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})}$
  - Context: $c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j$
  - Merge: $s_t = f(s_{t-1}, y_{t-1}, c_t)$

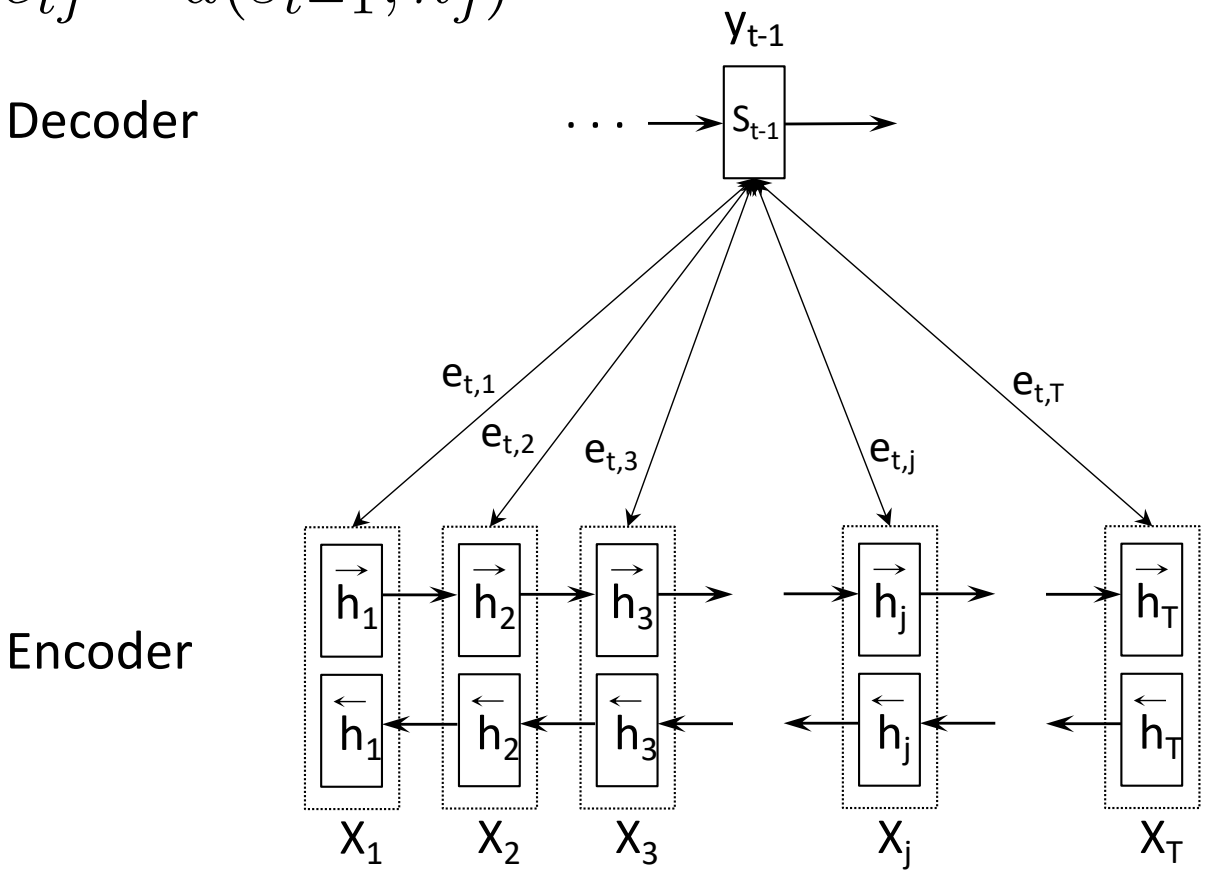


Attention mechanism

One step decoding:

$$e_{tj} = a(s_{t-1}, h_j)$$

$$e_{tj} = s_{t-1} \cdot h_j$$
$$e_{tj} = s_{t-1} W_a h_i$$
$$e_{tj} = V_a \cdot \tanh(W_a s_{t-1} + U_a h_j)$$

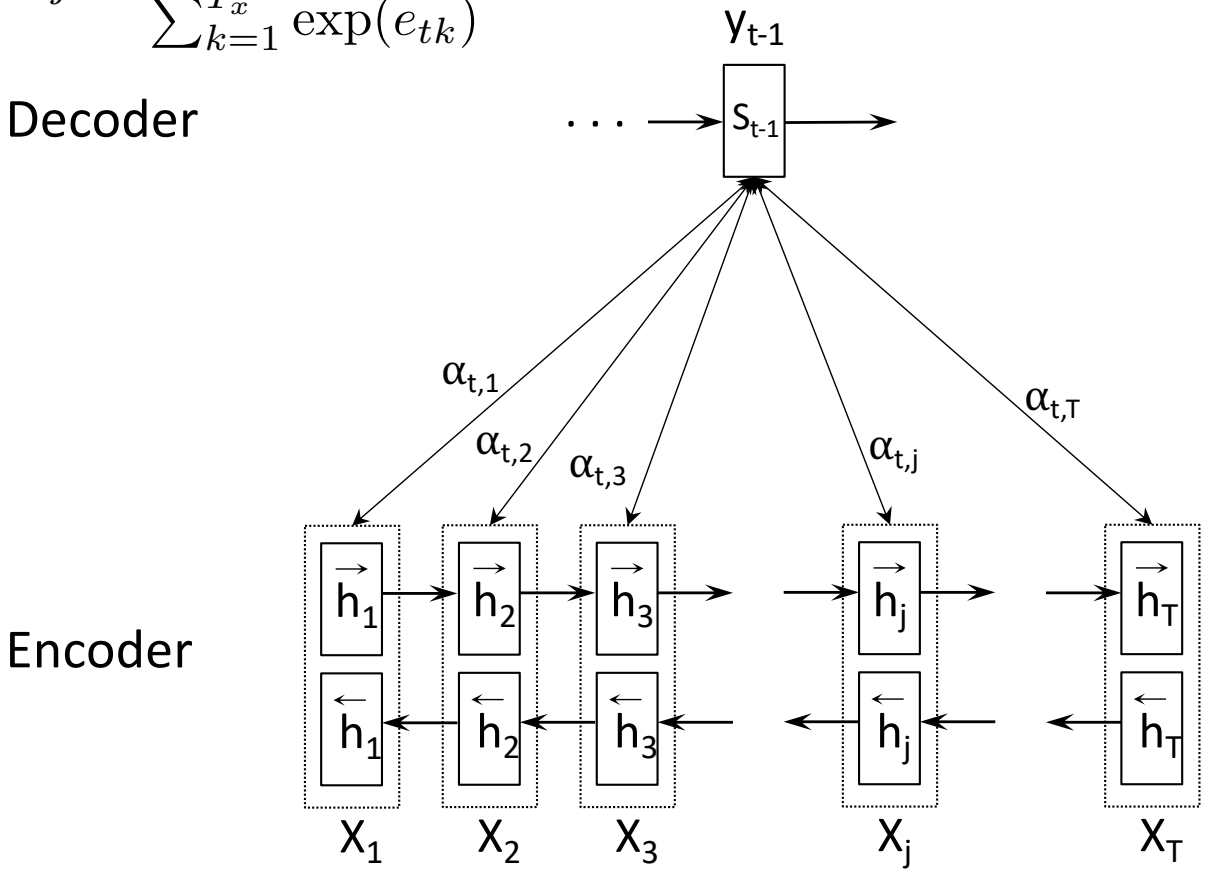


NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE ICLR2015

Attention mechanism

One step decoding:

$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})}$$



NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE ICLR2015

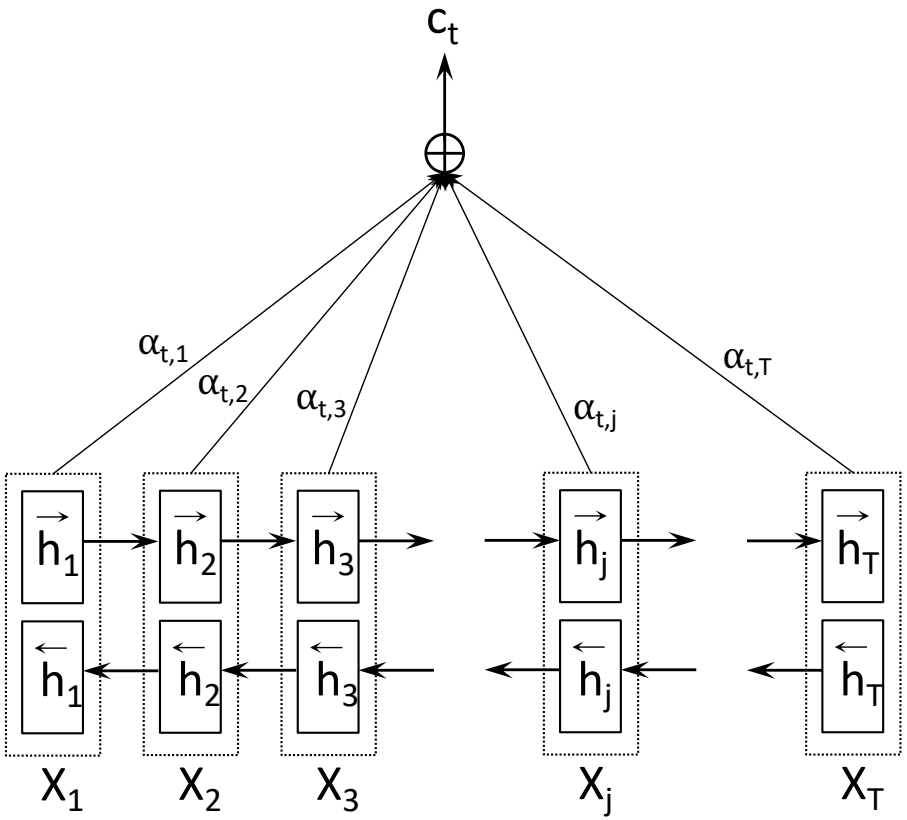
Attention mechanism

One step decoding:

$$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j$$

Decoder

Encoder

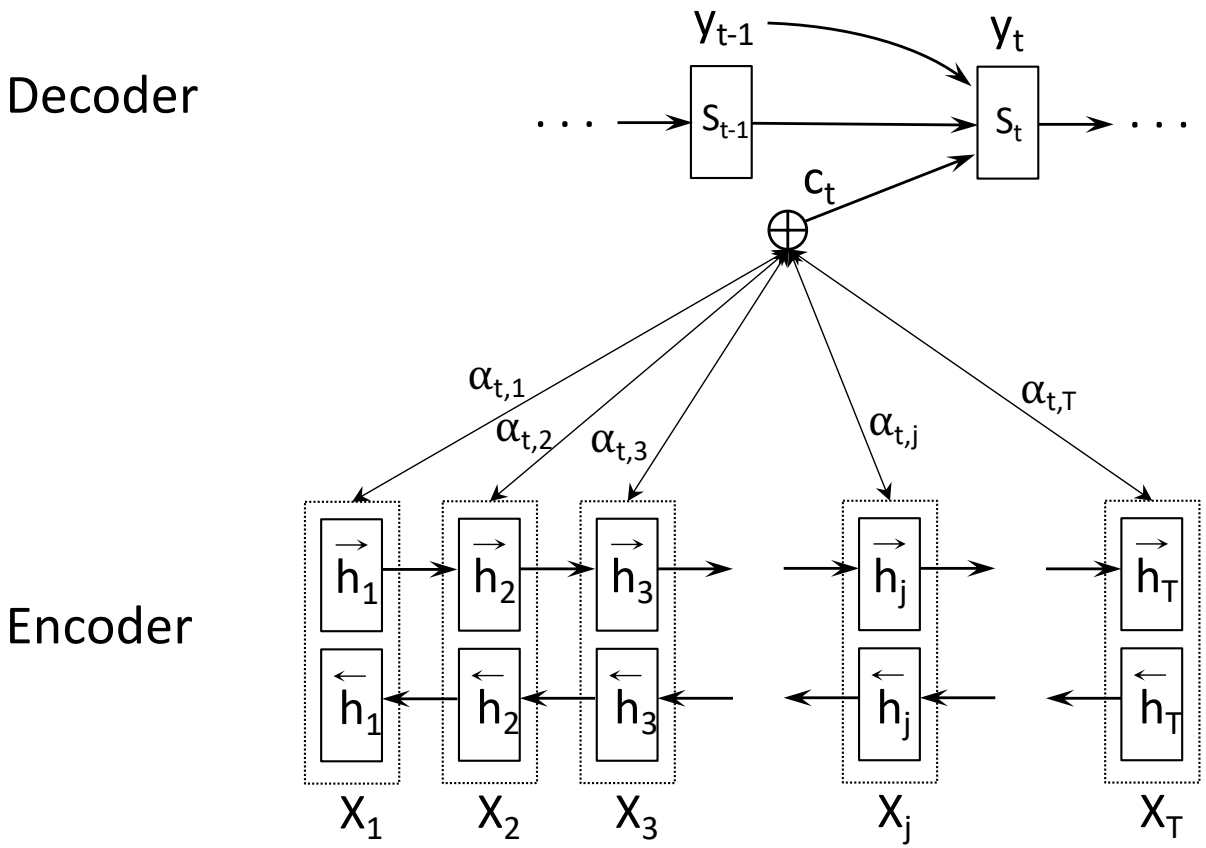


NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE ICLR2015

Attention mechanism

One step decoding: $P(y_t|Y_{<t}) = \text{softmax}(W * s_t)$

$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$



NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE ICLR2015

Attention mechanism (Luong)

One step decoding:

$$P(y_t|Y_{<t}) = \text{softmax}(W * \widetilde{s_t})$$

$$\widetilde{s_t} = \tanh(W_c(s_t \oplus c_t))$$

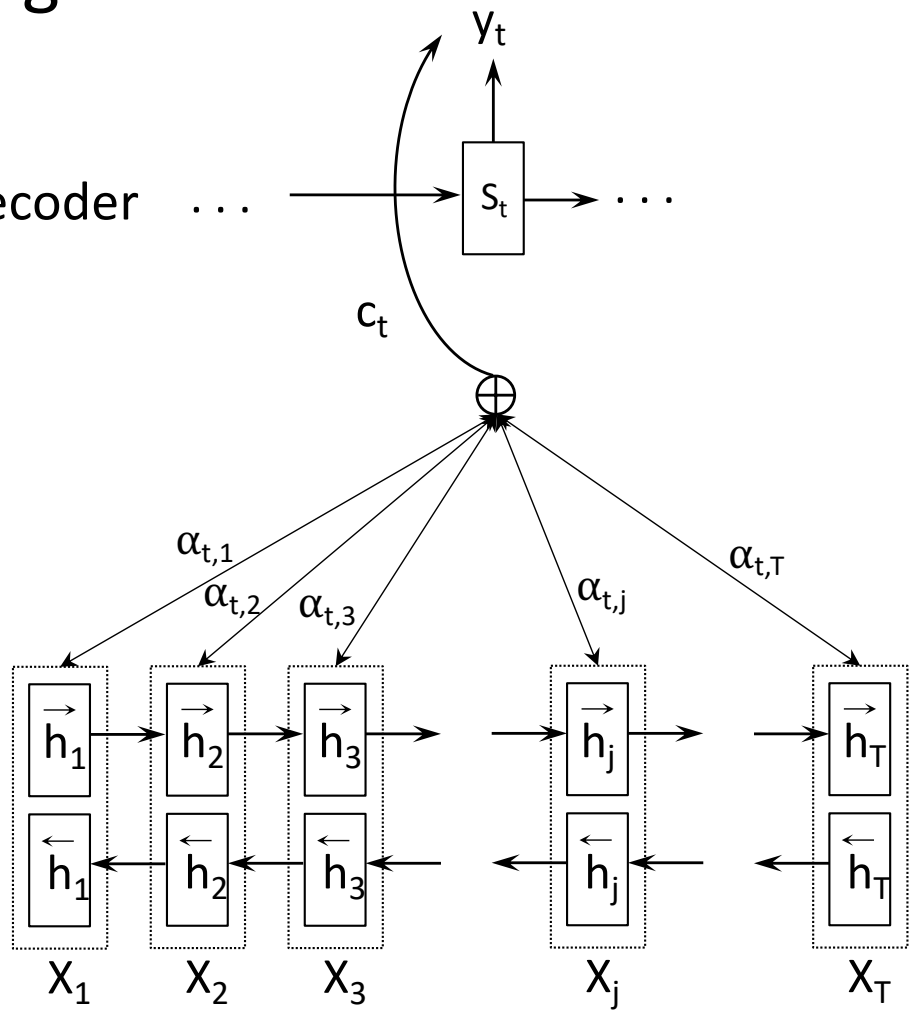
$$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j$$

$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})}$$

$$e_{tj} = a(s_t, h_j)$$

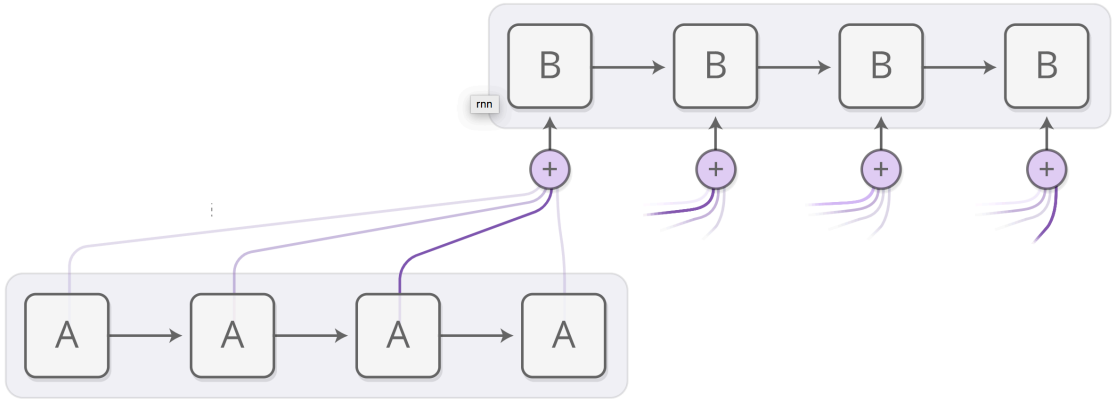
$$s_t = f(s_{t-1}, y_{t-1})$$

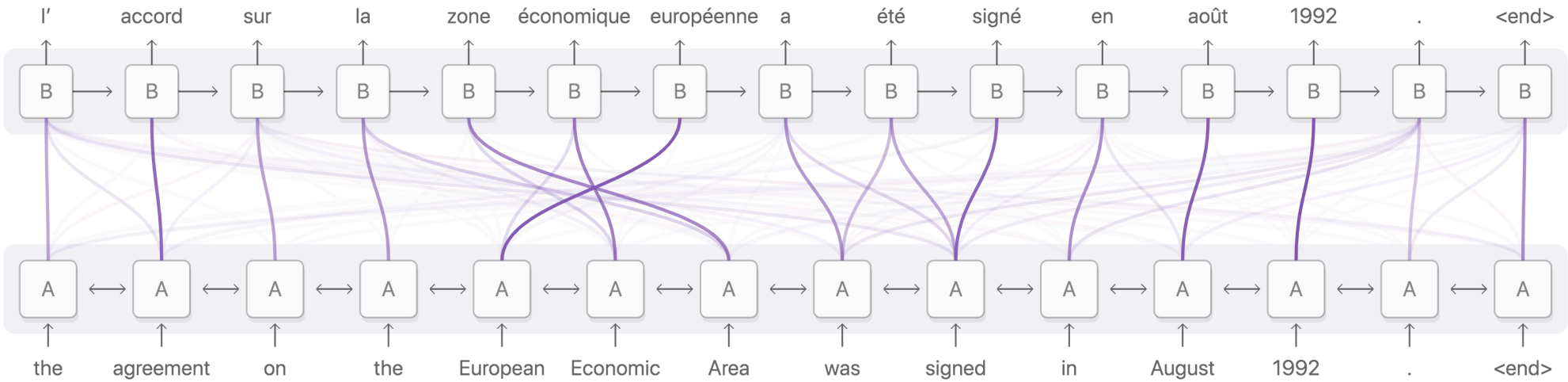
Encoder



NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE ICLR2015

Example: NMT

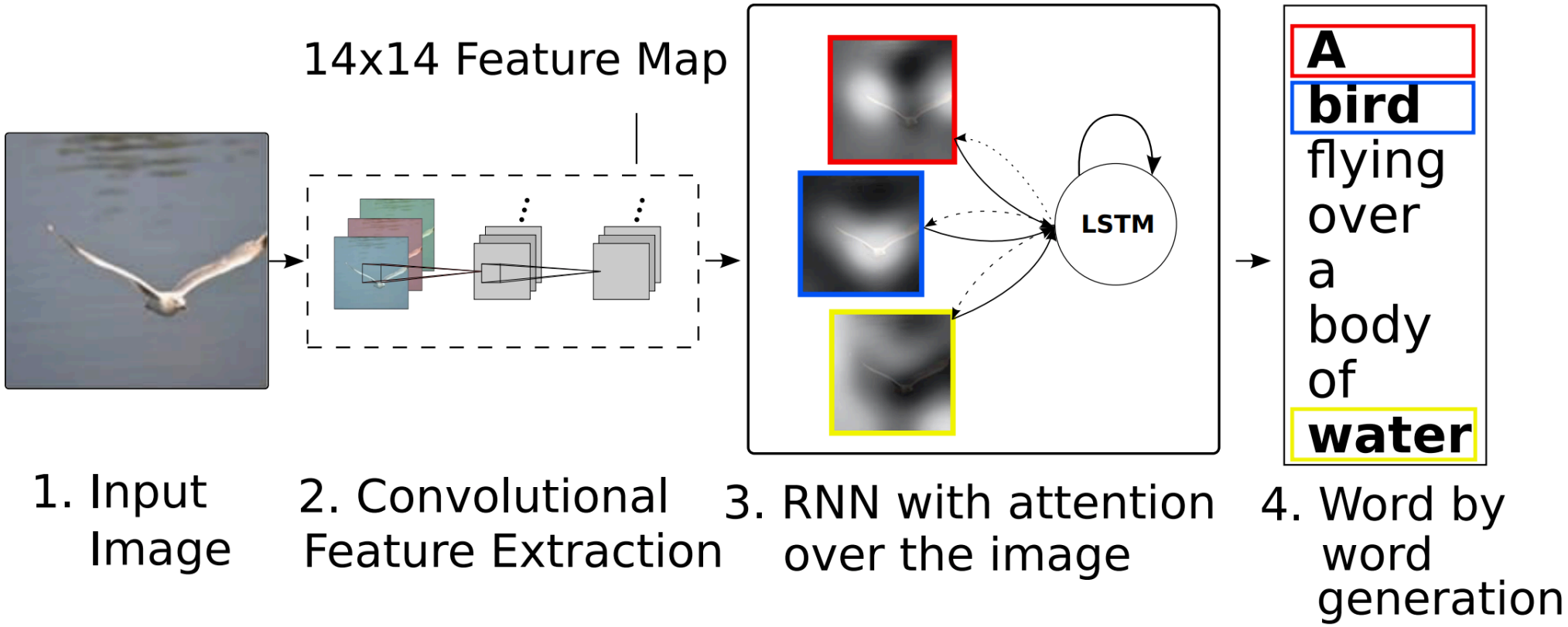




https://distill.pub/2016/augmented-rnns/#attentional-interfaces

Example: Image Captioning

- from image to text (with attention)



Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In ICML.

Summary

- RNN for sequence modeling
  - RNN
  - LSTM
  - GRU
- Application
  - Classification
  - Sequence Labeling
  - RNNLM
  - Sequence-to-sequence learning
  - Application: MT, Image Captioning
- Attention

Pioneers

Jürgen Schmidhuber

Long short-term memory

S Hochreiter, J Schmidhuber - Ne

Learning to store information ove
takes a very long time, mostly be
review Hochreiter's (1991) analys

☆ 99 被引用次数：22608

Yoshua Bengio, 2018

Probabilistic models of sequences: In the 1990s, Bengio combined neural networks with probabilistic models of sequences, such as hidden Markov models

High-dimensional word embeddings and attention:

Generative adversarial networks: