Recurrent Neural Network from Scratch

符尧 | PKU EECS | 2017.05.13

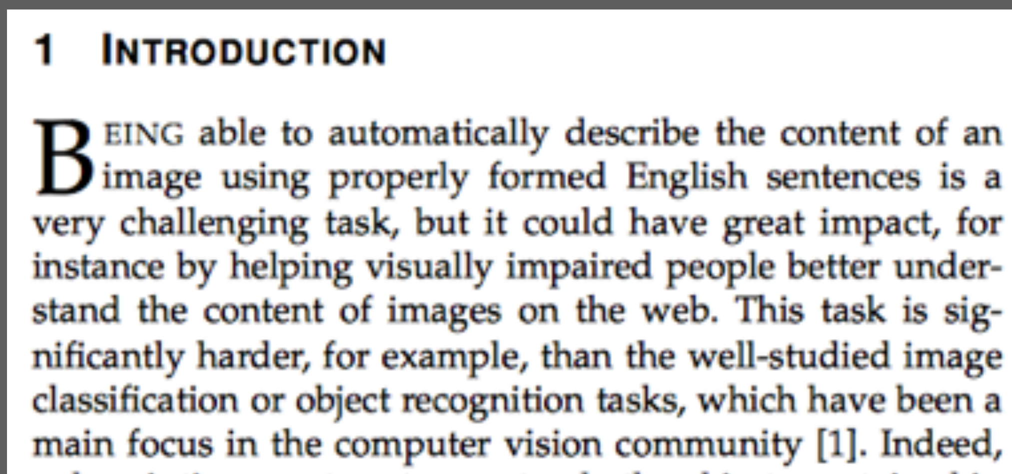
Guideline

- Sequence Processing and Why RNN
- What is RNN
- From Vanila RNN to LSTM
- Sequence Classification Case Study
- State of Art: Attention Mechanism
- RNN Playground

Guideline

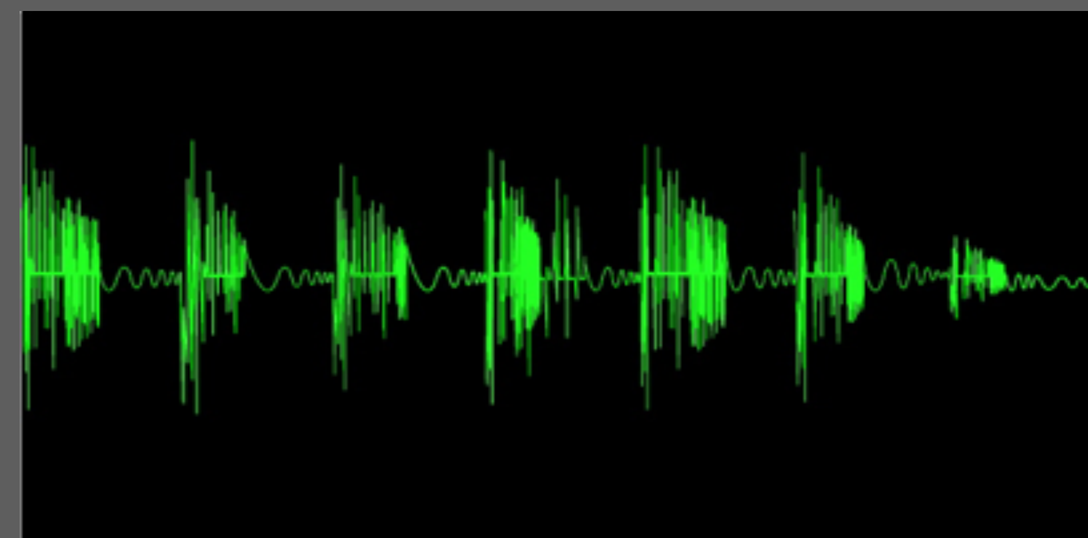
- Sequence Processing and Why RNN
- What is RNN
- From Vanila RNN to LSTM
- Sequence Classification Case Study
- State of Art: Attention Mechanism
- RNN Playground

Sequence Processing and Why RNN


1 INTRODUCTION

BEING able to automatically describe the content of an image using properly formed English sentences is a very challenging task, but it could have great impact, for instance by helping visually impaired people better understand the content of images on the web. This task is significantly harder, for example, than the well-studied image classification or object recognition tasks, which have been a main focus in the computer vision community [1]. Indeed,

Text data



Audio data

Financial data

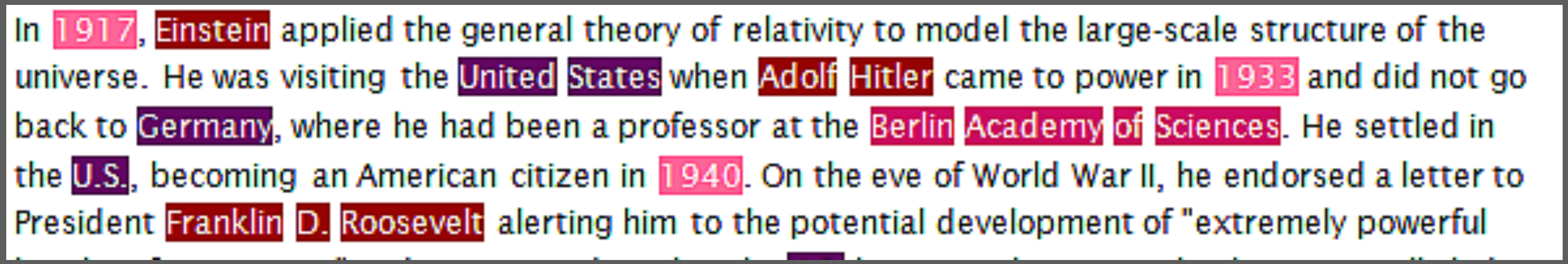
Video data

Three basic machine learning tasks

- Sequence Classification

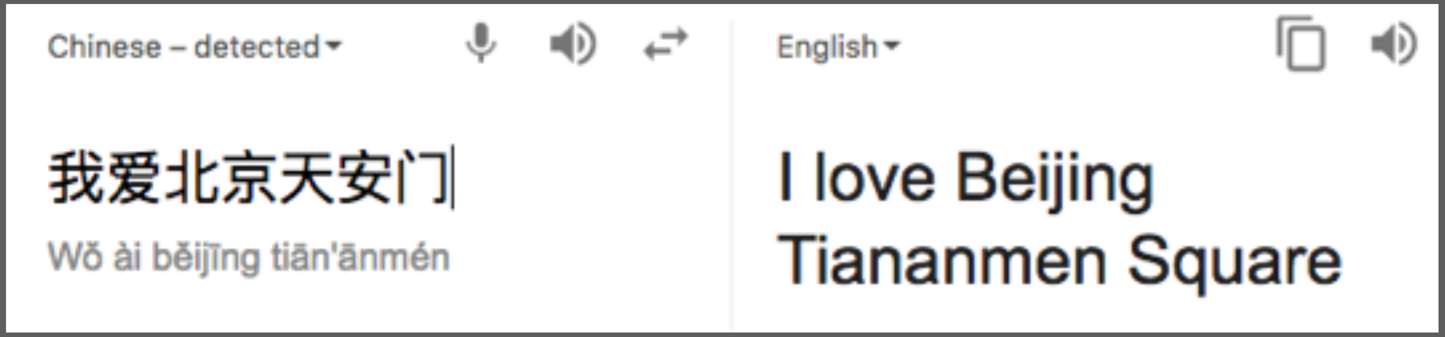
e.g: sentiment classification

- Sequence Labeling


In 1917, Einstein applied the general theory of relativity to model the large-scale structure of the universe. He was visiting the United States when Adolf Hitler came to power in 1933 and did not go back to Germany, where he had been a professor at the Berlin Academy of Sciences. He settled in the U.S., becoming an American citizen in 1940. On the eve of World War II, he endorsed a letter to President Franklin D. Roosevelt alerting him to the potential development of "extremely powerful

e.g: named entity recognition

- Sequence Generation



e.g: sequence 2 sequence, machine translation

Sequence Processing and Why RNN

Previous Approach: fixed length window

e.g: n-gram

I love Beijing Tianmen Square

I love

love Beijing

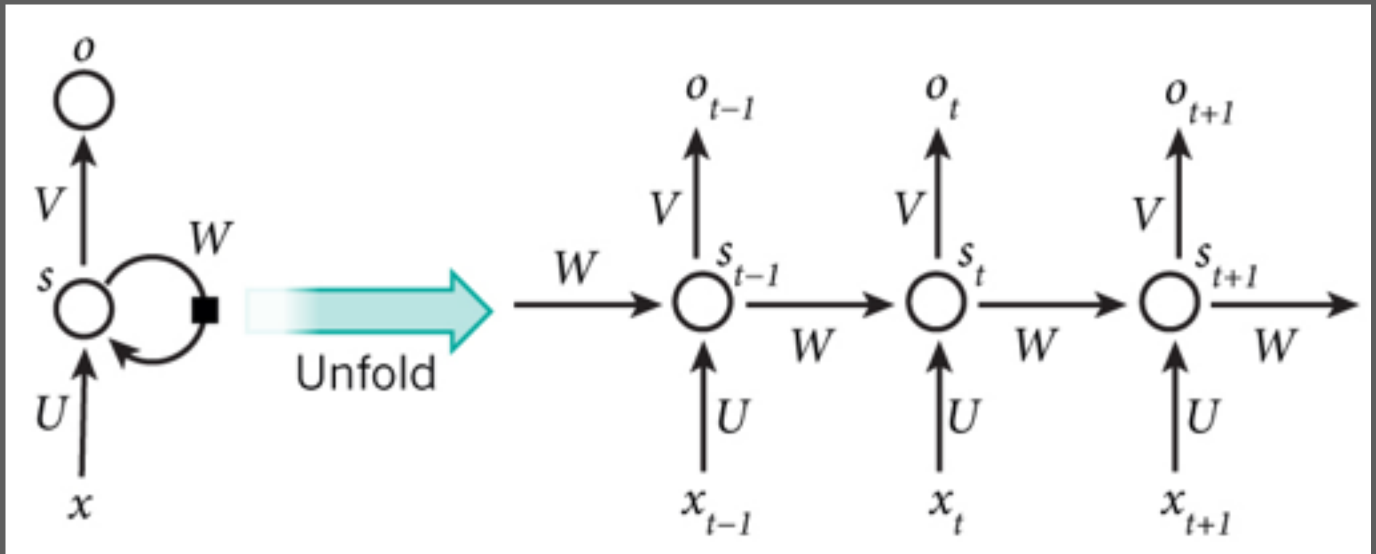
Beijing Tianmen

Tianmen Square

… Not flexible to arbitrary length dependency

can we find a way to tackle long term dependency?

This is why RNN



Guideline

Sequence Processing and Why RNN

What is RNN

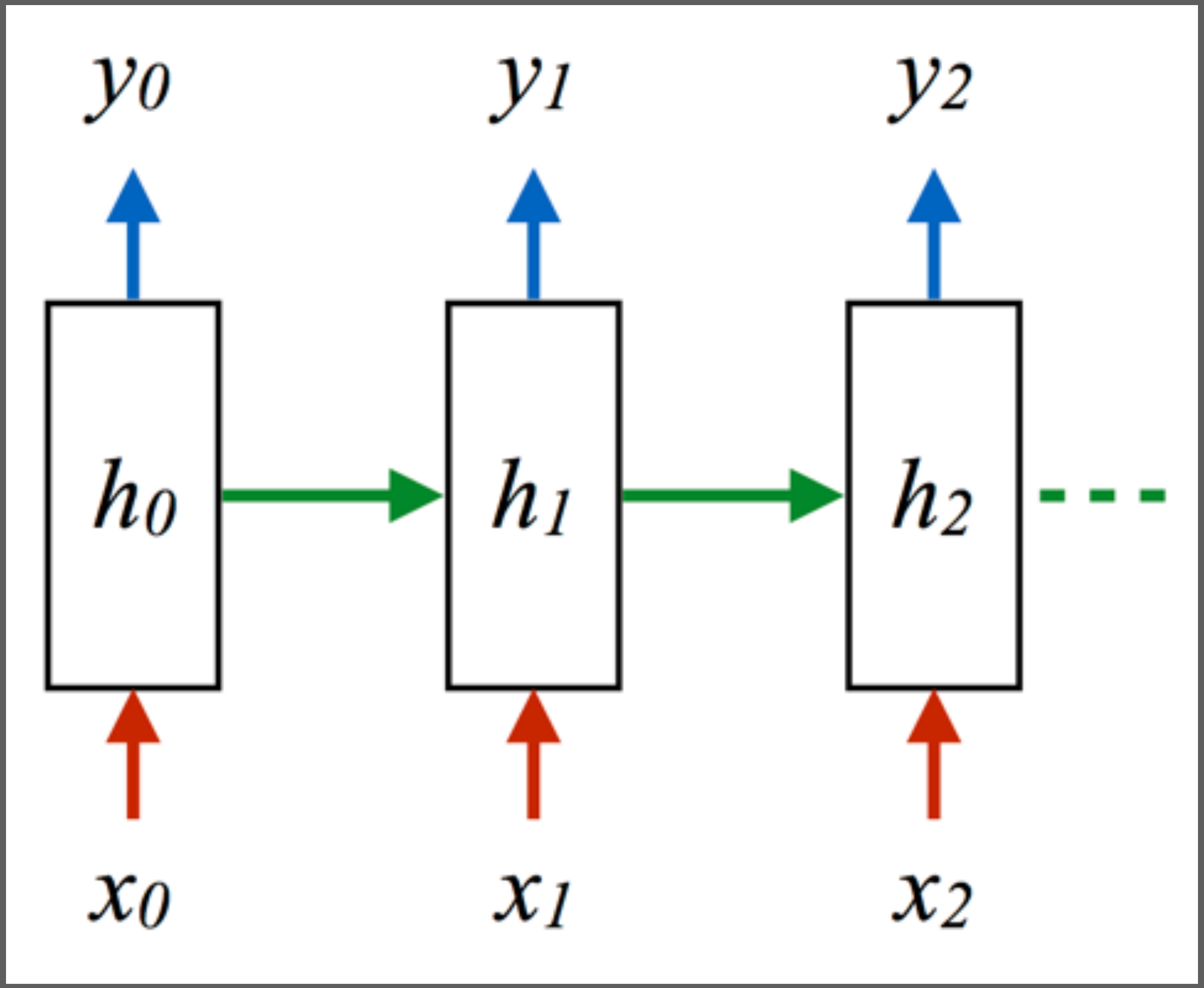
From Vanila RNN to LSTM

Sequence Classification Case Study

State of Art: Attention Mechanism

RNN Playground

What is RNN



$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

Input at timestep t: x_t

output at timestep t: y_t
  y_t depend on h_t

hidden state at timestep t: h_t
  h_t depend on x_t and h_{t - 1}

sequence input: x1, x2 … xn

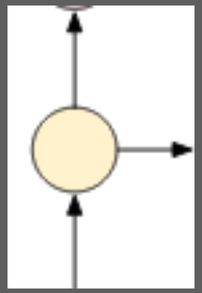
sequence output: y1, y2 … yn
  - sequence labeling

sequence encoding: hn
  - sequence classification

RNN step by step

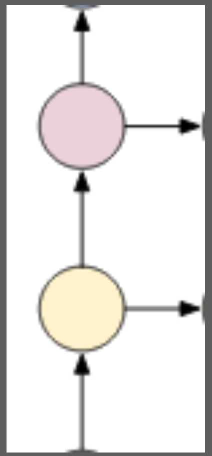
Single RNN cell:

```
# size: number of units in LSTM, i.e. input vecto
# need to be initialized
lstm_cell = tf.contrib.rnn.BasicLSTMCell(size,
  forget_bias=0.0, state_is_tuple=True)
```



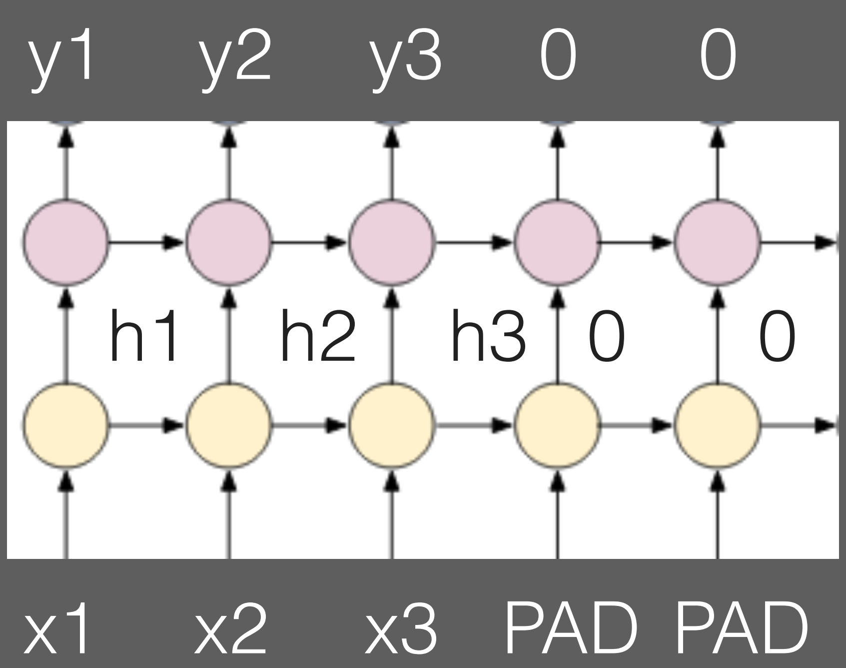
Stacked RNN cells:

```
# stacked simple RNN cell
cell = tf.contrib.rnn.MultiRNNCell(
  [lstm_cell] * config.num_layers, state_is_tuple=True)
```



Unrolled RNN cell stacks:

```
# input size: batch size * time step * embedding size
# outputs: batch size * time step * embedding size (outputs large
# tested
outputs, state = tf.nn.dynamic_rnn(cell, input_x,
  sequence_length = input_length, initial_state = initial_state)
```



Variable length? padding

Training: Truncated Back Propagation Through Time

$$\mathbf{h}_t = f(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1})$$

$$\mathbf{y}_t = g(\mathbf{W}_{hy}\mathbf{h}_t)$$

Simplification: get rid of b

$$E = c\sum_{t=1}^{T}\|\mathbf{l}_t - \mathbf{y}_t\|^2 = c\sum_{t=1}^{T}\sum_{j=1}^{L}(l_t(j) - y_t(j))^2$$

Loss function: MSE
Target: L_t, output: y_t

$$w^{new} = w - \gamma\frac{\partial E}{\partial w}$$

Weight update

Reference: 俞栋，邓力 et.al. 解析深度学习：语音识别实践

Training: Truncated Back Propagation Through Time

After ….

$$\delta_T^y(j) = -\frac{\partial E}{\partial y_T(j)}\frac{\partial y_T(j)}{\partial v_T(j)} = (l_T(j) - y_T(j))g'(v_T(j))$$

$$\delta_T^h(j) = -\left(\sum_{i=1}^{L}\frac{\partial E}{\partial v_T(i)}\frac{\partial v_T(i)}{\partial h_T(j)}\frac{\partial h_T(j)}{\partial u_T(j)}\right) = \sum_{i=1}^{L}\delta_T^y(i)w_{hy}(i,j)f'(u_T(j))$$

$$\delta_t^y(j) = (l_t(j) - y_t(j))g'(v_t(j)) \quad \text{其中，} j = 1, 2, \ldots, L$$

$$\begin{aligned}\delta_t^h(j) &= -\left[\sum_{i=1}^{N}\frac{\partial E}{\partial u_{t+1}(i)}\frac{\partial u_{t+1}(i)}{\partial h_t(j)} + \sum_{i=1}^{L}\frac{\partial E}{\partial v_t(i)}\frac{\partial v_t(i)}{\partial h_t(j)}\right]\frac{\partial h_t(j)}{\partial u_t(j)} \\ &= \left[\sum_{i=1}^{N}\delta_{t+1}^h(i)w_{hh}(i,j) + \sum_{i=1}^{L}\delta_t^y(i)w_{hy}(i,j)\right]f'(u_t(j))\end{aligned}$$

Reference: 俞栋，邓力 et.al. 解析深度学习：语音识别实践

Training: Truncated Back Propagation Through Time

We have

$$\mathbf{h}_t = f(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1})$$

$$\mathbf{y}_t = g(\mathbf{W}_{hy}\mathbf{h}_t)$$

$$\mathbf{W}_{xh}^{new} = \mathbf{W}_{xh} + \gamma \boxed{\sum_{t=1}^{T} \delta_h^t \mathbf{x}_t^T}$$

??

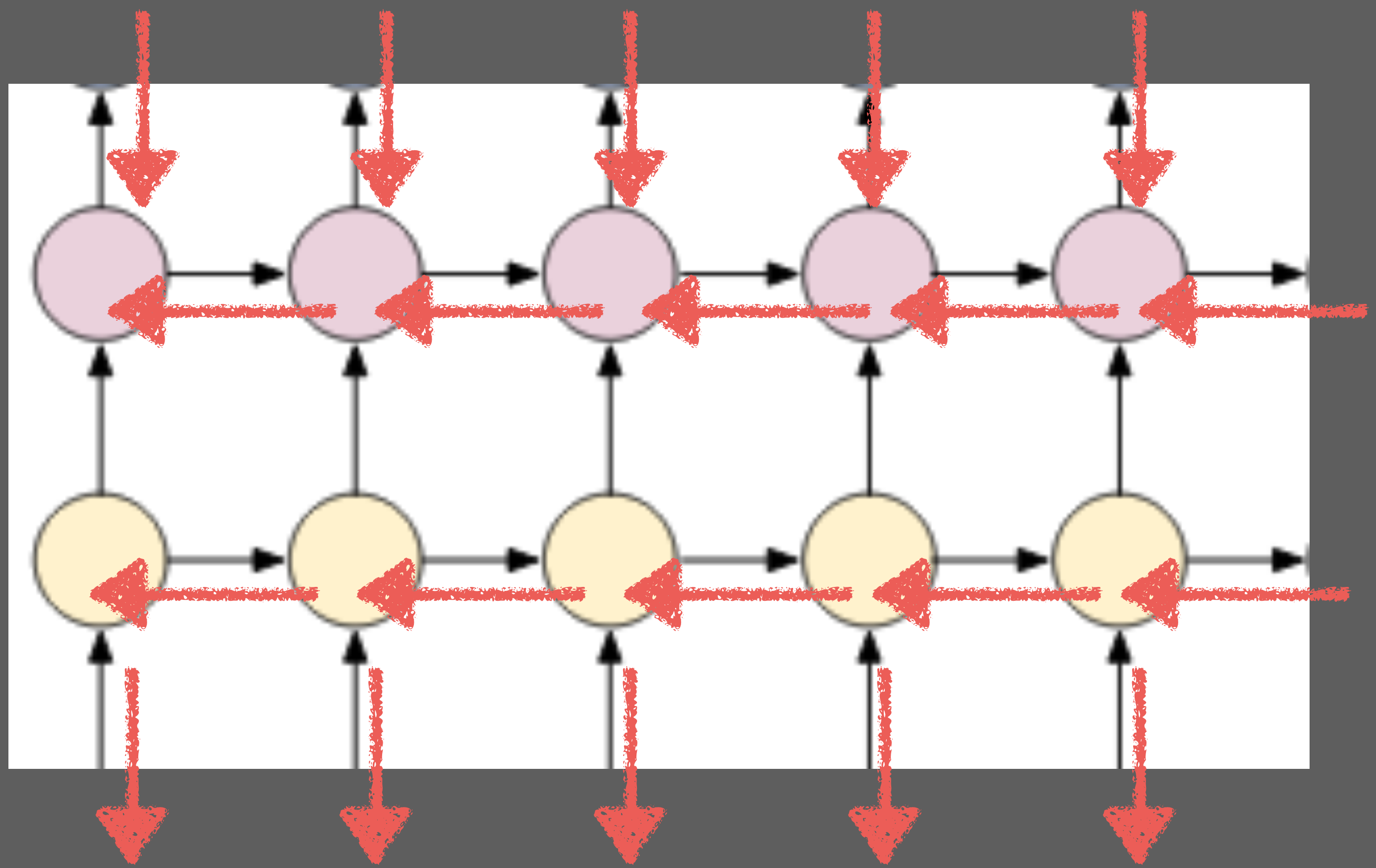
$$\mathbf{W}_{hh}^{new} = \mathbf{W}_{hh} + \gamma \sum_{t=1}^{T} \delta_h^t \mathbf{h}_{t-1}^T$$

$$\mathbf{W}_{hy}^{new} = \mathbf{W}_{hy} + \gamma \sum_{t=1}^{T} \delta_y^t \mathbf{h}_t^T$$

If pad -> ht = 0 -> W not update

Reference: 俞栋，邓力 et.al. 解析深度学习：语音识别实践

Training: Truncated Back Propagation Through Time



Error flow at BPTT: … ideal case

But in fact it is not what you think … this is why LSTM

Reference: 俞栋，邓力 et.al. 解析深度学习：语音识别实践

Guideline

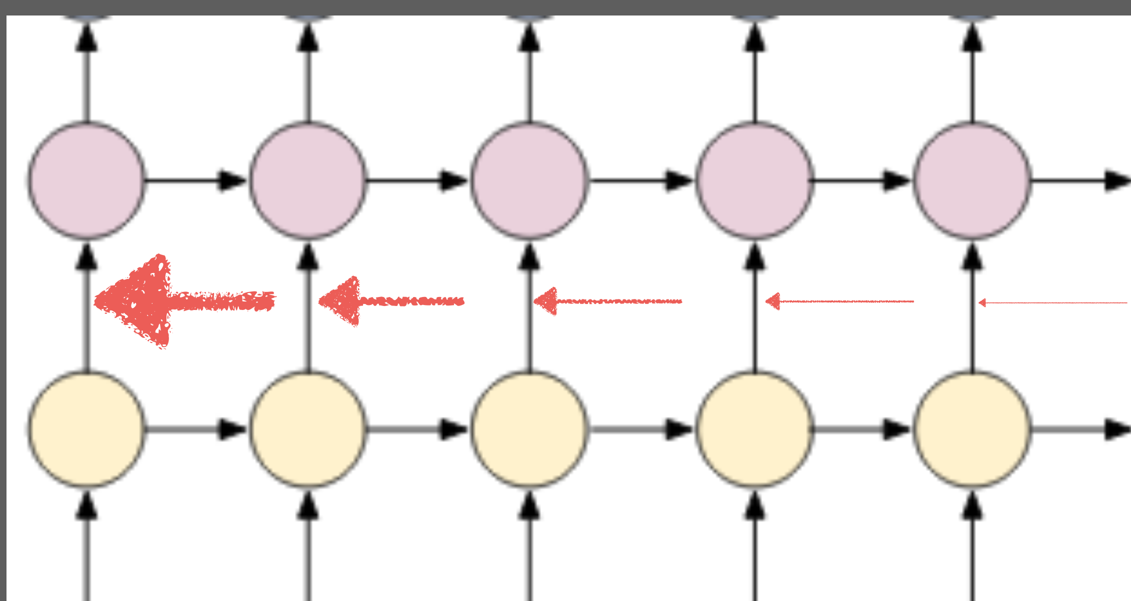
- Sequence Processing and Why RNN
- What is RNN
- **From Vanila RNN to LSTM**
- Sequence Classification Case Study
- State of Art: Attention Mechanism
- RNN Playground

Gradient Vanishing/ explosion when Back Propagation Through Time

A upper bound of gradient by theoretical analysis

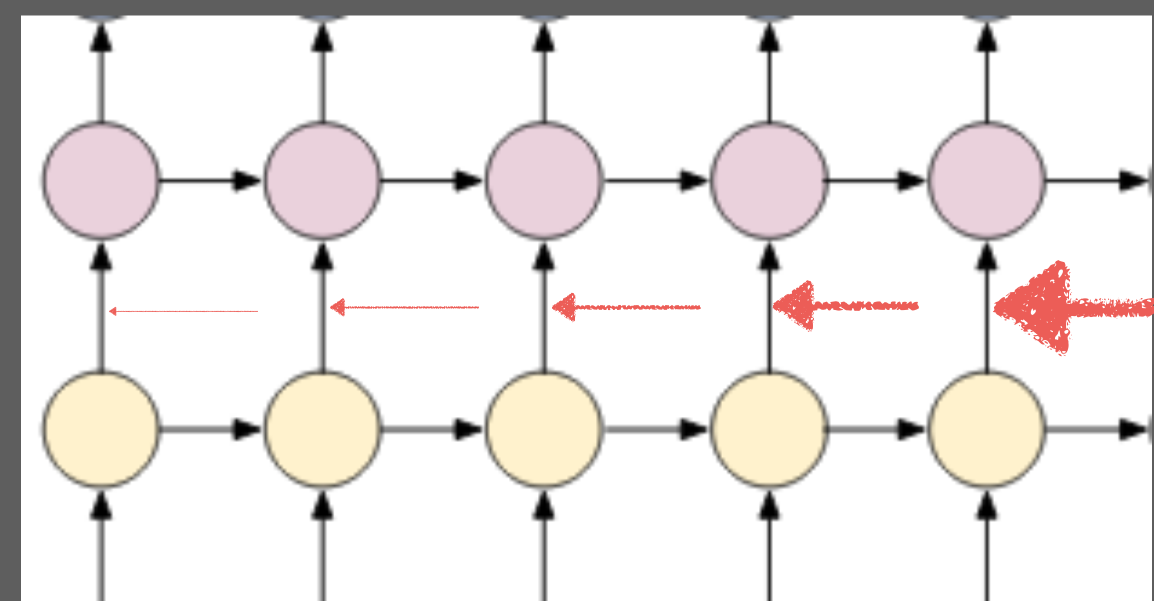
$$\left\| \frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \left(\prod_{i=k}^{t-1} \frac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{x}_i} \right) \right\| \leq \boxed{\eta^{t-k}} \left\| \frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \right\|$$

Exponential over t



Gradient vanishing over timesteps

Hard to train and
lose long term dependency

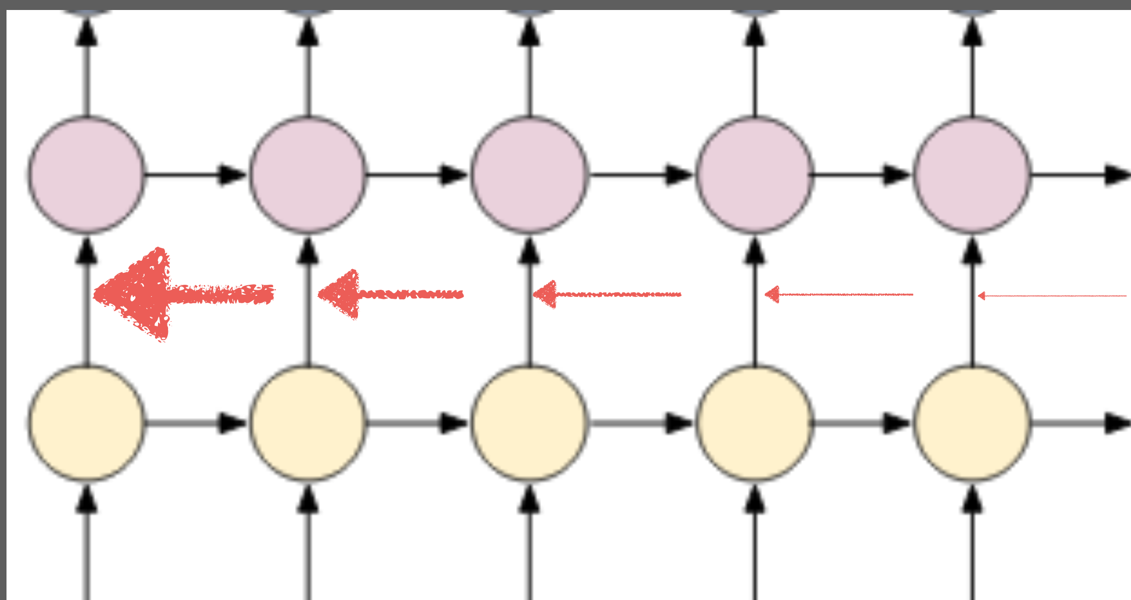


Gradient explosion over timesteps

Error flow at BPTT, … real world

Reference: Pascanu et. al. *On the difficulty of training RNN*

From Vanila RNN to LSTM



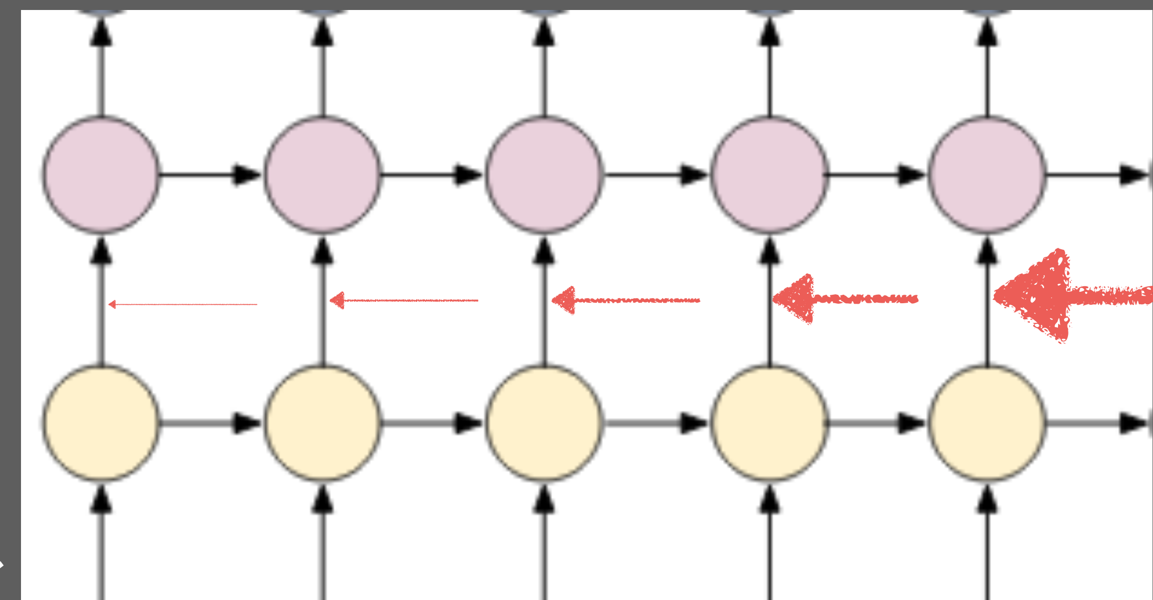
Gradient vanishing …

No very explicit solution (on vanilla RNN)

(but there exist some
optimization related solution.)

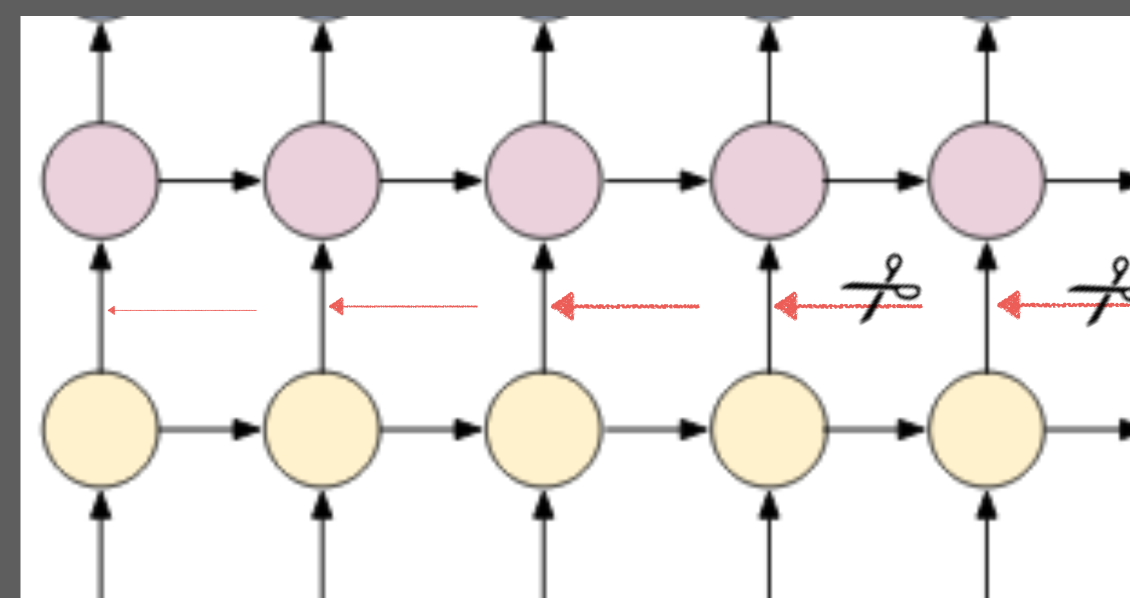
—> LSTM comes to aid here.

with long term dependency and
some more advantages



Gradient explosion

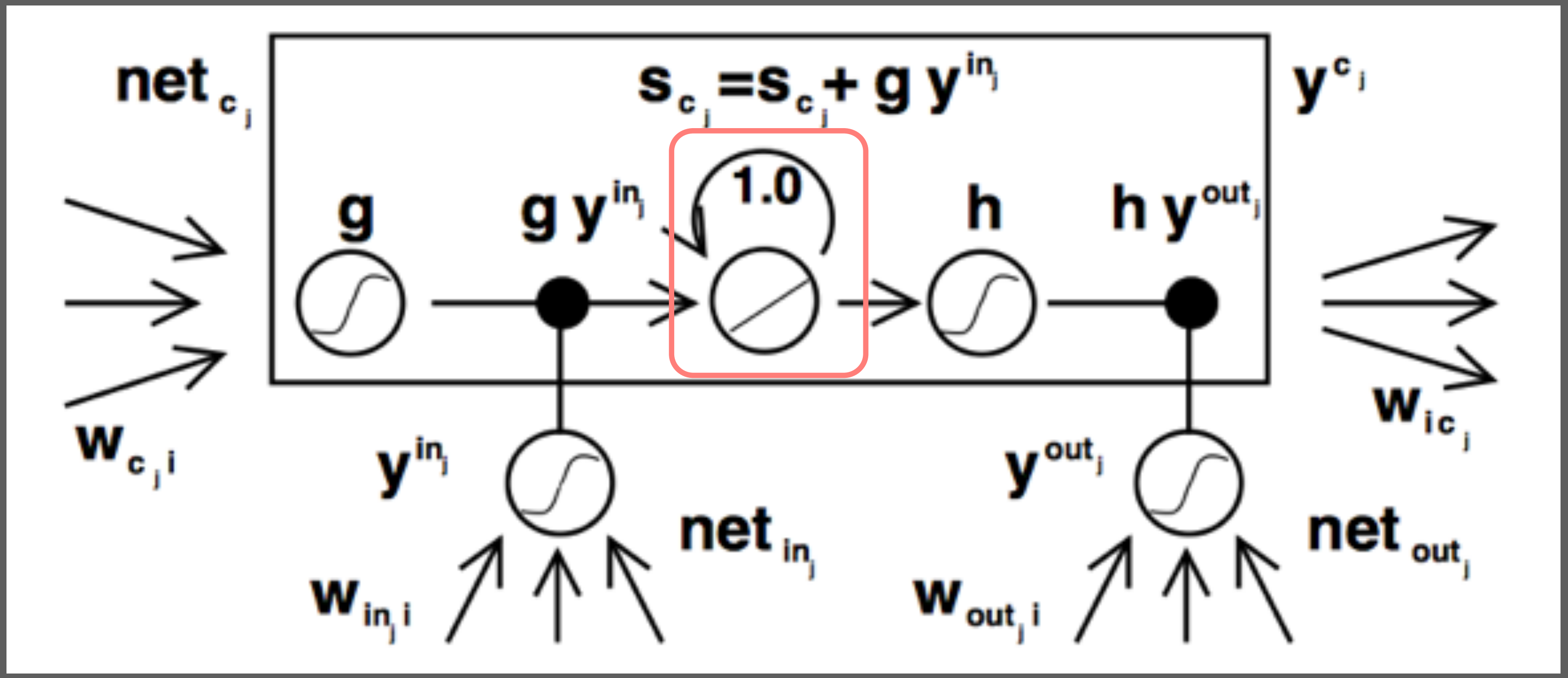
Solution



Gradient clipping: LSTM also does this

Reference: Pascanu et. al. *On the difficulty of training RNN*

LSTM has constant error flow
— by implementing gating mechanism



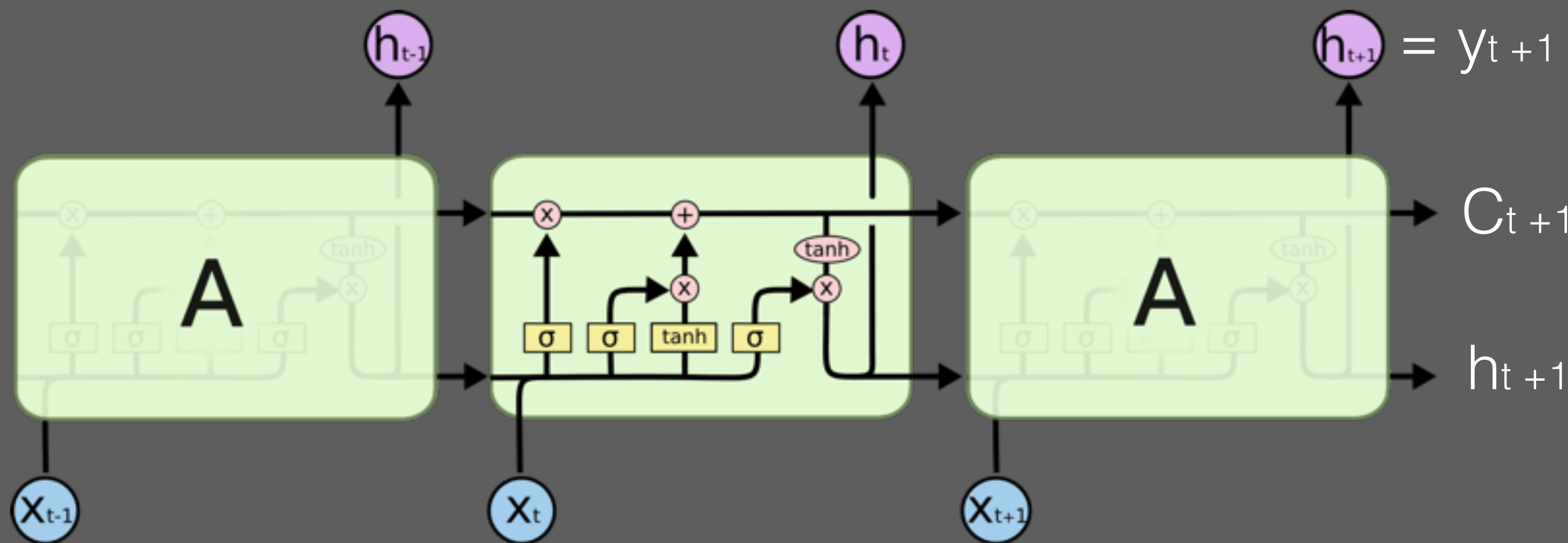
Can solve: gradient vanishing (over time)
Cannot solve: gradient explosion (but we have clipping)

Mathematical details are skipped here, see the original paper

Reference: Hochreiter et. al. *Long Short-term Memory*

From Vanila RNN to LSTM

IO of LSTM cell



x_t Input vector

h_t Output vector/ short term information

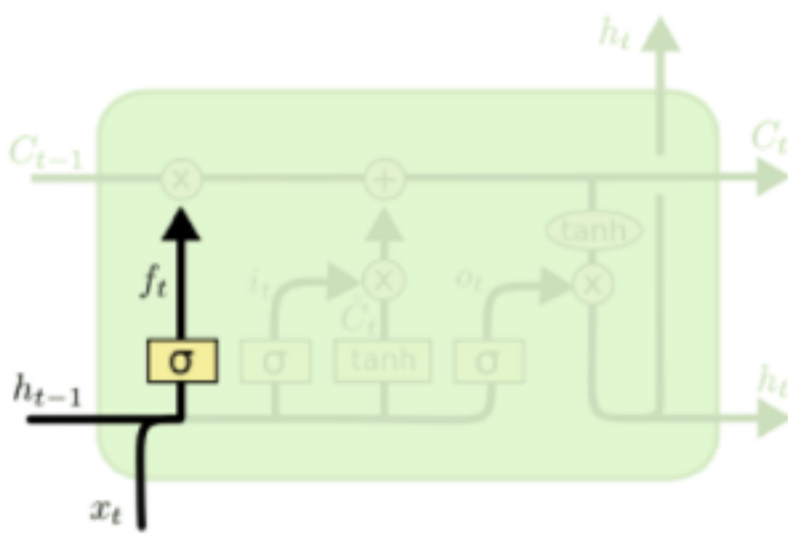
C_t Cell state/ long term information

Long Short term

Note: $h_t = y_t$

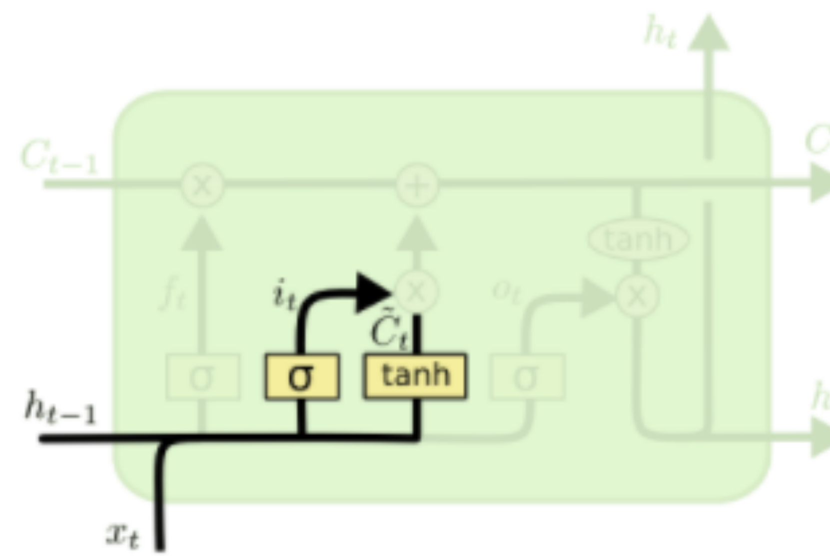
Reference: Colah's blog: *Understanding LSTM Networks*

Inside LSTM cell: gating mechanism



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

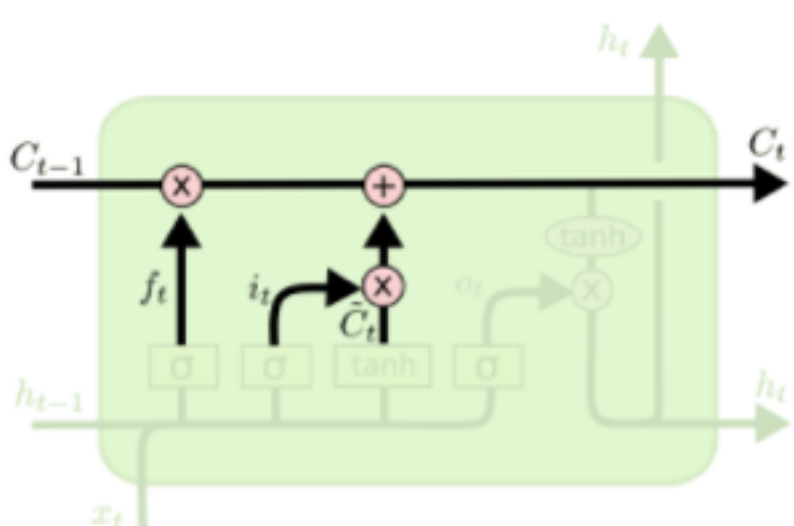
Forget gate



Input gate

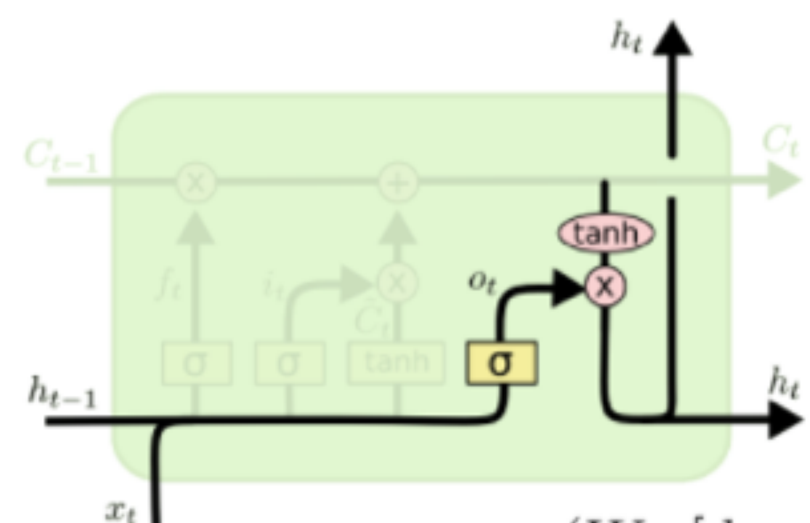
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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



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

Gating



Output gate

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

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

Reference: Colah's blog: *Understanding LSTM Networks*

On gating mechanism

All LSTM variants have
- i: input gate
- f: forget gate
- o: output gate
- c: cell state (long term state)

Gating Useful to
- learn to control information flow
- maintain long-term information
- lock gradient

Gating is wildly used:
- GRU
- Facebook Gated CNN
- and more fancy architectures

So how to build a real project? see a case study

Guideline

Sequence Processing and Why RNN

What is RNN

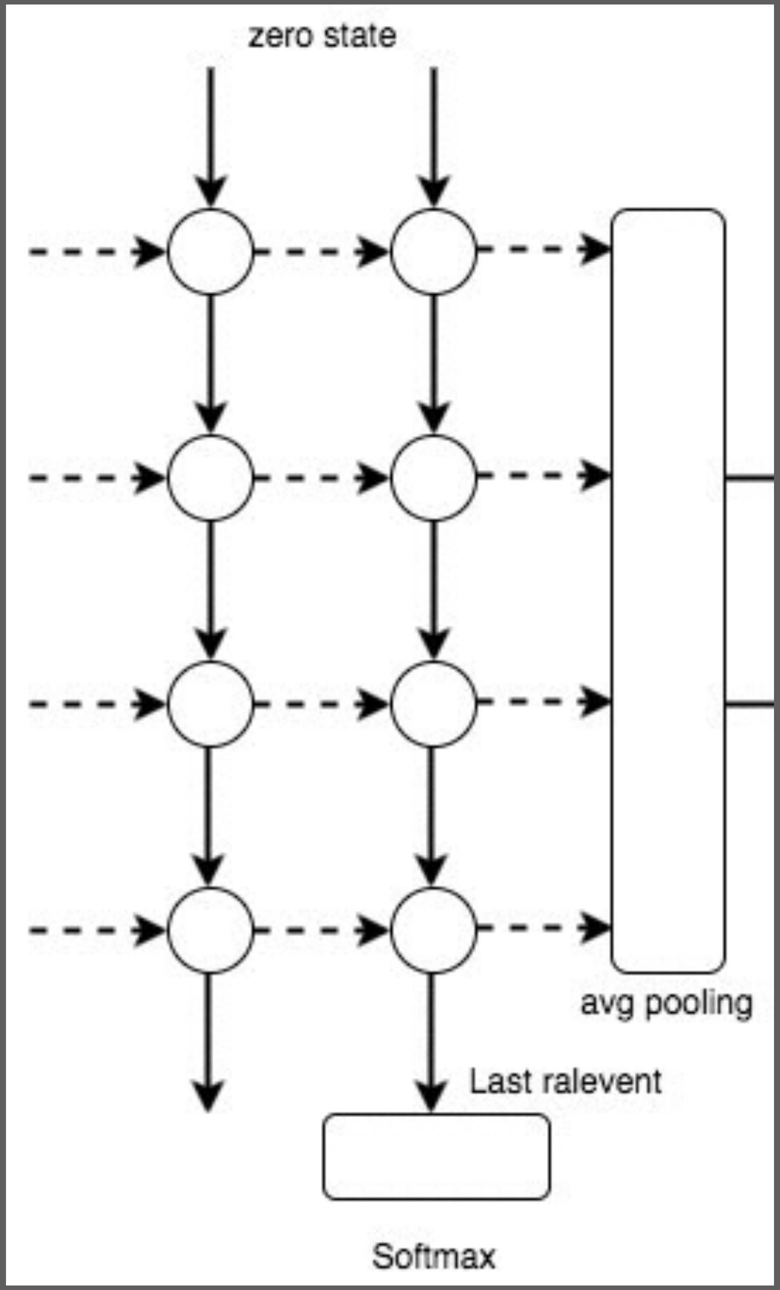
From Vanila RNN to LSTM

Sequence Classification Case Study

State of Art: Attention Mechanism

RNN Playground

Sequence Classification Case Study

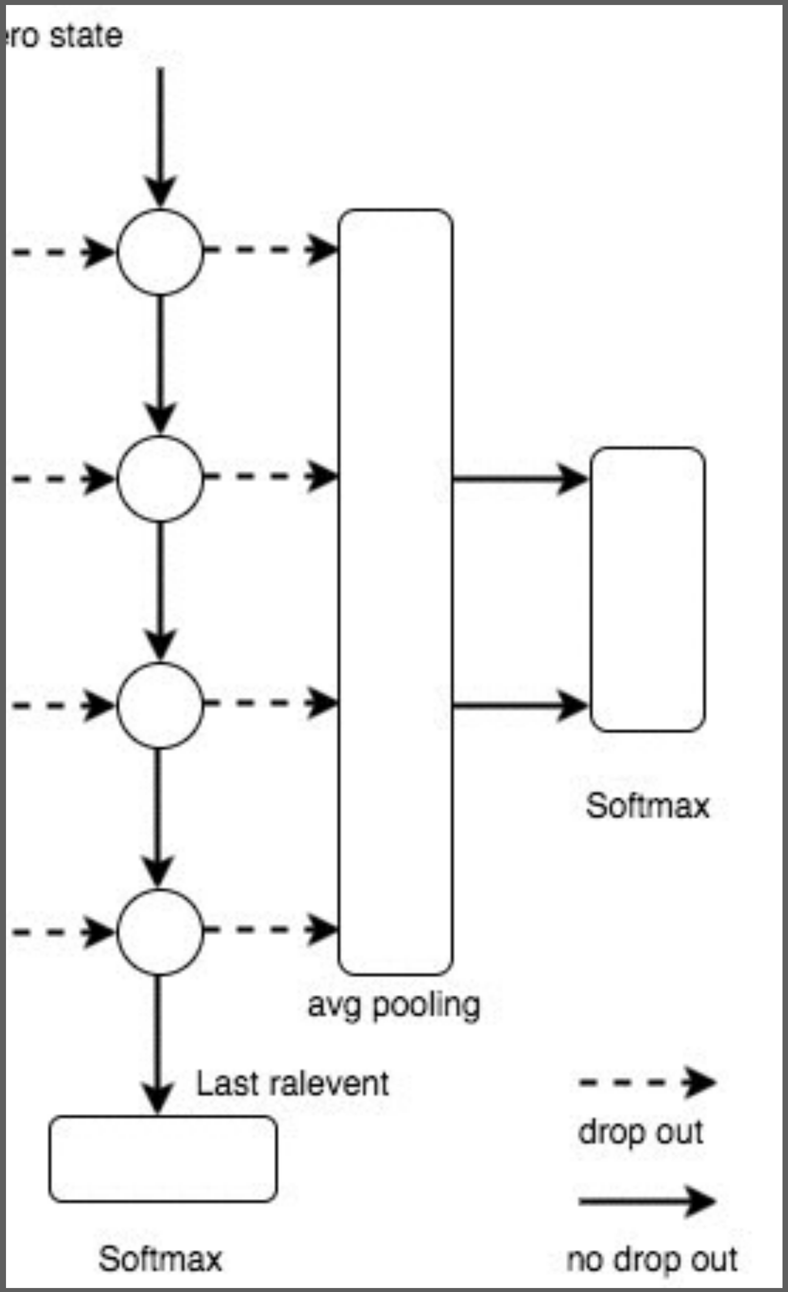



```
# LSTM cell construction
lstm_cell = tf.contrib.rnn.BasicLSTMCell(size,
  forget_bias=0.0, state_is_tuple=True)

lstm_cell = tf.contrib.rnn.DropoutWrapper(lstm_cell,
  output_keep_prob = config.keep_prob)

cell = tf.contrib.rnn.MultiRNNCell(
  [lstm_cell] * config.num_layers, state_is_tuple=True)

initial_state = cell.zero_state(batch_size, tf.float32)

outputs, state = tf.nn.dynamic_rnn(cell, input_x,
  sequence_length = input_length, initial_state = initial_state)
```

Note: dropout only on dashed lines

Reference: Zaremba et.al. *Recurrent Neural Networks Regularization*

Sequence Classification Case Study




```
# last relevant
last_relevant_output = get_last_relevant(outputs, input_length)

# average pooling
average_output = avg_pooling(outputs, input_length)

# softmax
softmax_w = tf.get_variable("softmax_w", [size, num_classes], dtype
softmax_b = tf.get_variable("softmax_b", [num_classes], dtype = tf.
logits = tf.matmul(last_relevant_output, softmax_w) + softmax_b

# predicted labels
prediction = tf.nn.softmax(logits)
```

Two approaches

- last relevant output (sentence embedding)
- average pooling (average on word embedding)
- basically they are the same
- but why average?
- Should it emphasis on certain steps?

This is why Attention mechanism.

Guideline

Sequence Processing and Why RNN

What is RNN

From Vanila RNN to LSTM

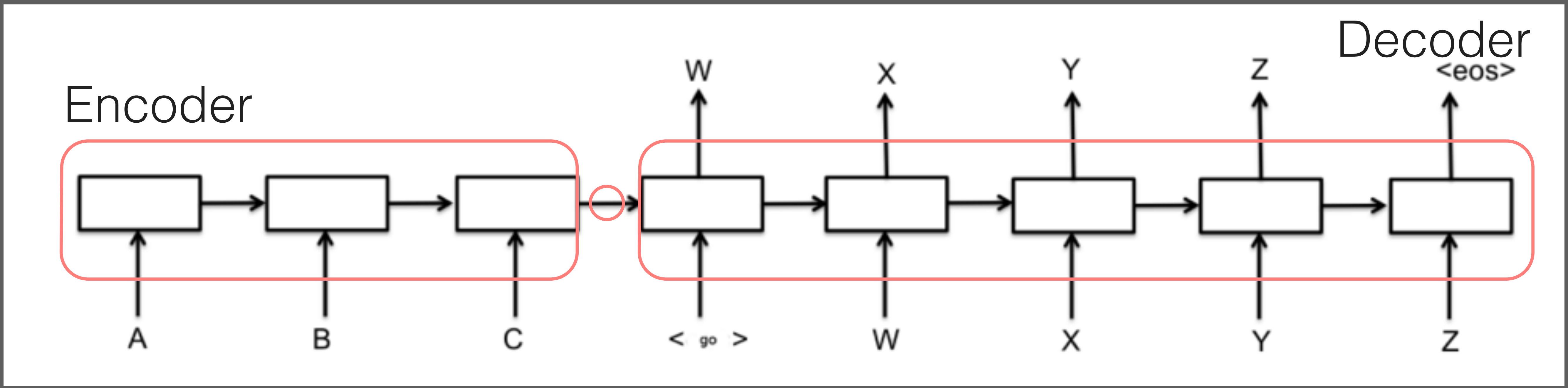
Sequence Classification Case Study

State of Art: Attention Mechanism

RNN Playground

State of Art: Attention Mechanism

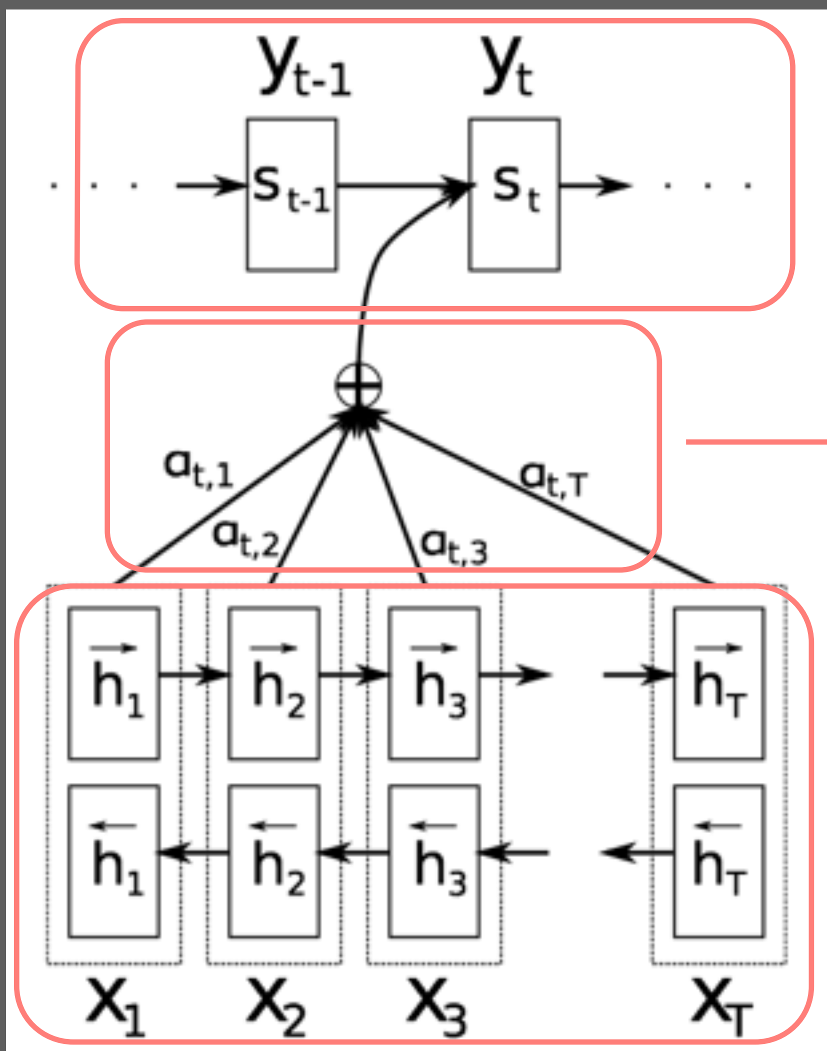
Vanilla sequence to sequence model for machine translation



two LSTMs: encoder and decoder
Encoder: source language
Decoder: target language
Use last hidden state of encoder as initial state for decoder

State of Art: Attention Mechanism

Attention mechanism



Decoder

$$s_i = f(s_{i-1}, y_{i-1}, \boxed{c_i}).$$

Attention table aij

and context vector ci

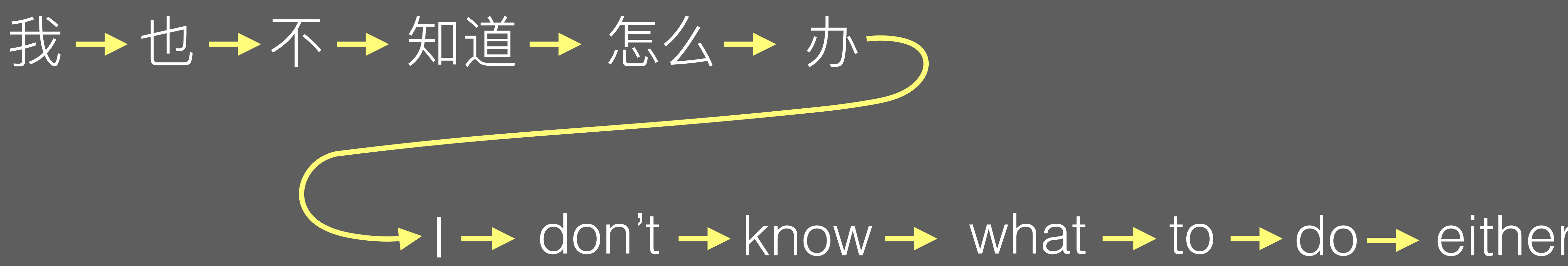
Encoder

Context vector is a weight average over all encoder's hidden states

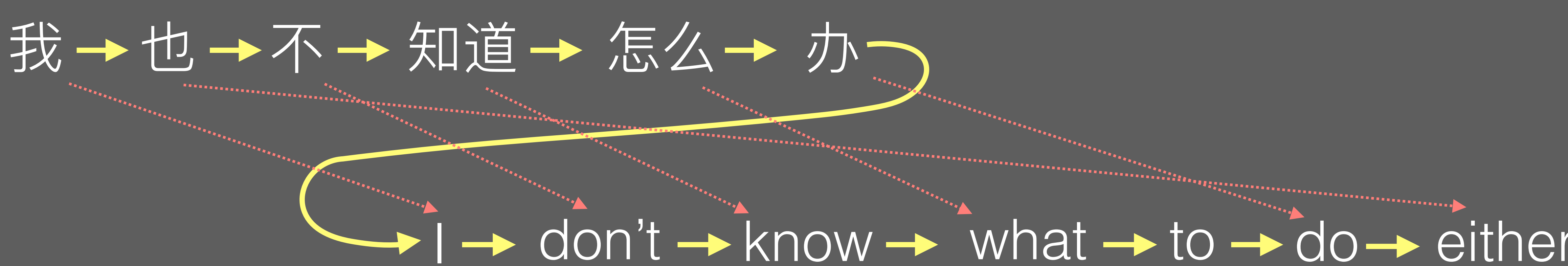
Reference: Bahdanau et. al.: *Neural Machine Translation by Jointly Learning to Align and Translate*

State of Art: Attention Mechanism

An example:



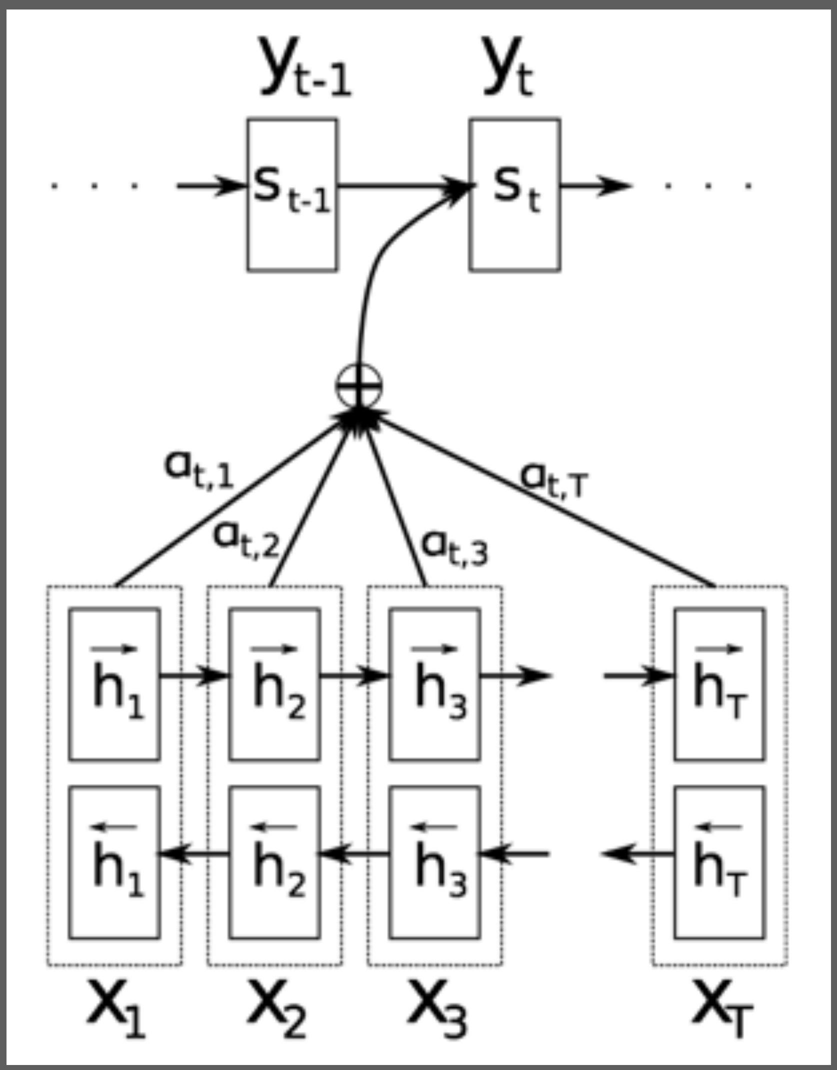
Vanilla seq2seq



seq2seq with attention

Reference: Bahdanau et. al.: *Neural Machine Translation by Jointly Learning to Align and Translate*

All we need to compute is context vector ci



$$e_{ij} = a(s_{i-1}, h_j)$$

e: energy
a: feed forward network

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

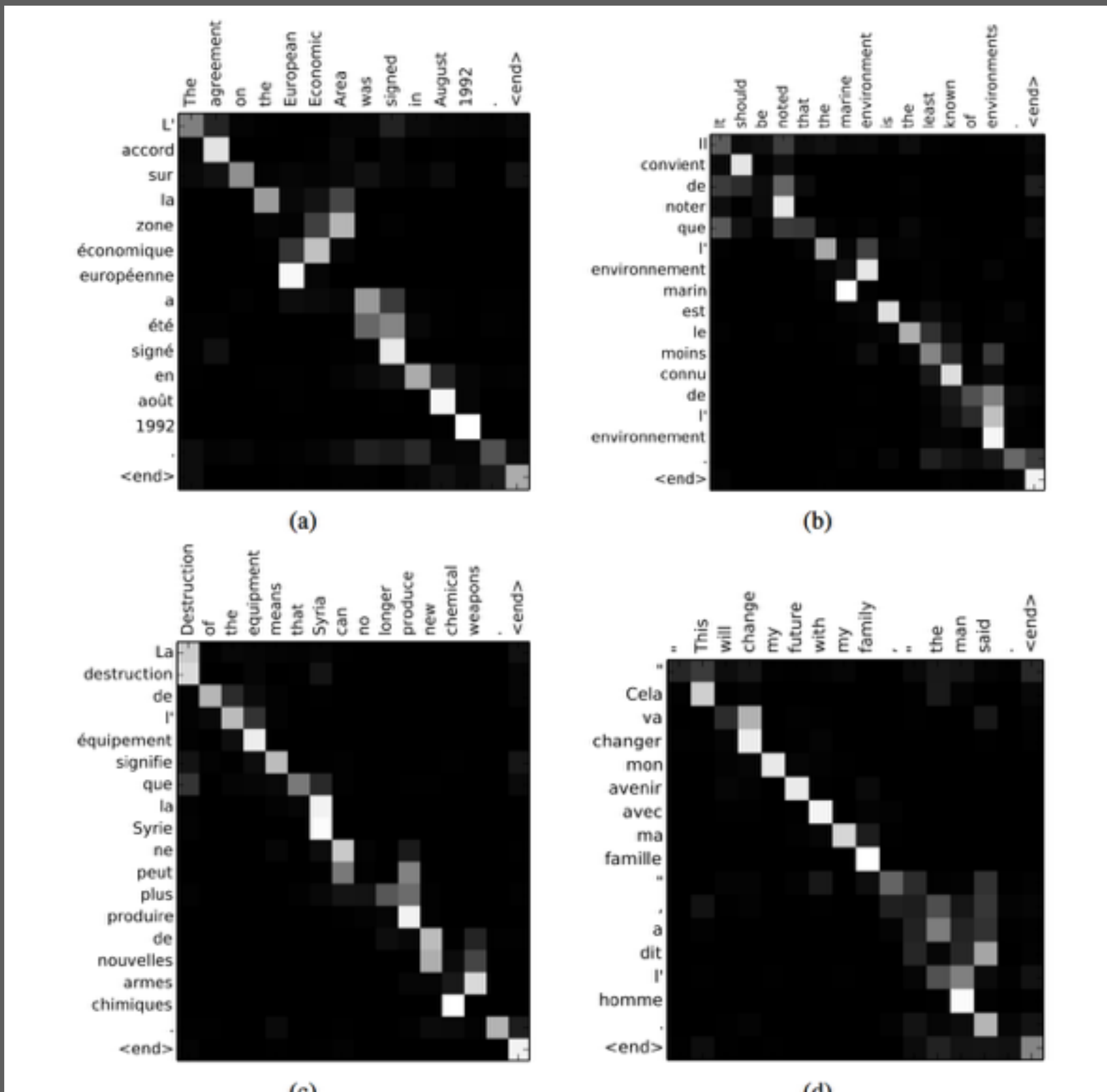
ai: attention vector to generate yi
aij: to generate yi, put weight on input j

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

ci: weight average on every hidden state hj

Reference: Bahdanau et. al.: *Neural Machine Translation by Jointly Learning to Align and Translate*

State of Art: Attention Mechanism



Attention table visualization

Performance evaluation:
BLEU score
a customized metrics for MT

use your own google translate to see!

Reference: Bahdanau et. al.: *Neural Machine Translation by Jointly Learning to Align and Translate*

Guideline

- Sequence Processing and Why RNN
- What is RNN
- From Vanila RNN to LSTM
- Sequence Classification Case Study
- State of Art: Attention Mechanism
- RNN Playground

Wrap-up: from sequence to graph

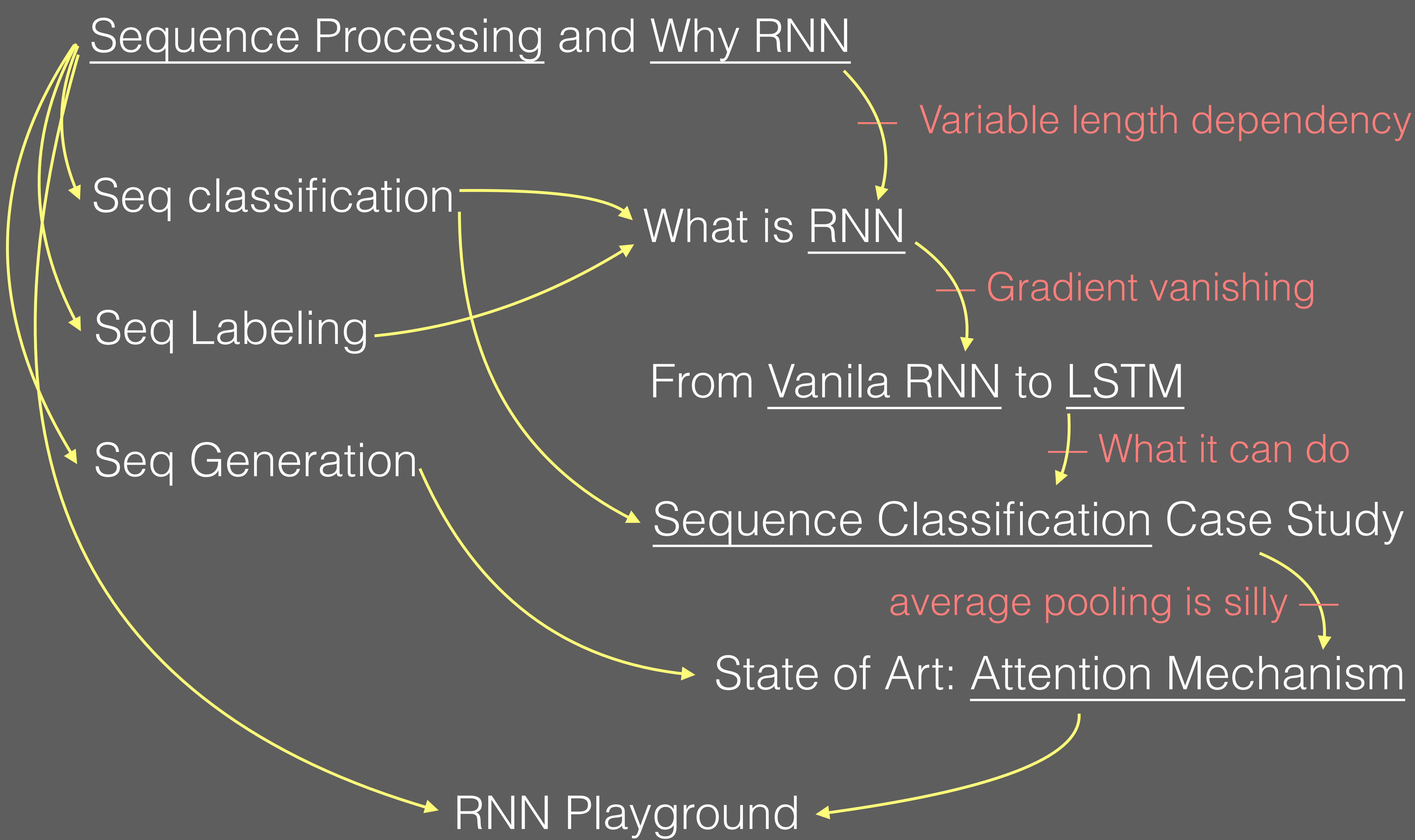
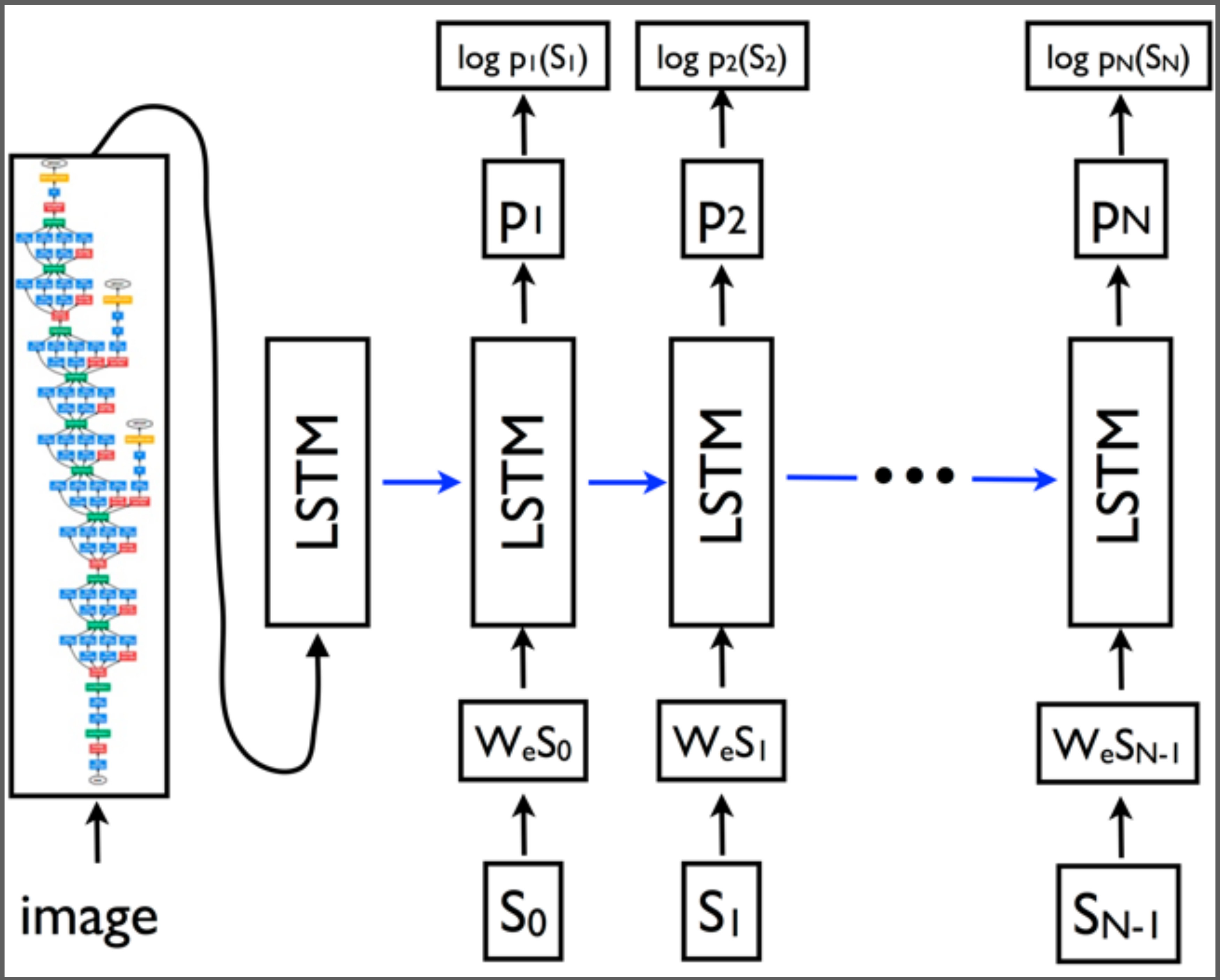


Image Caption: Im2seq

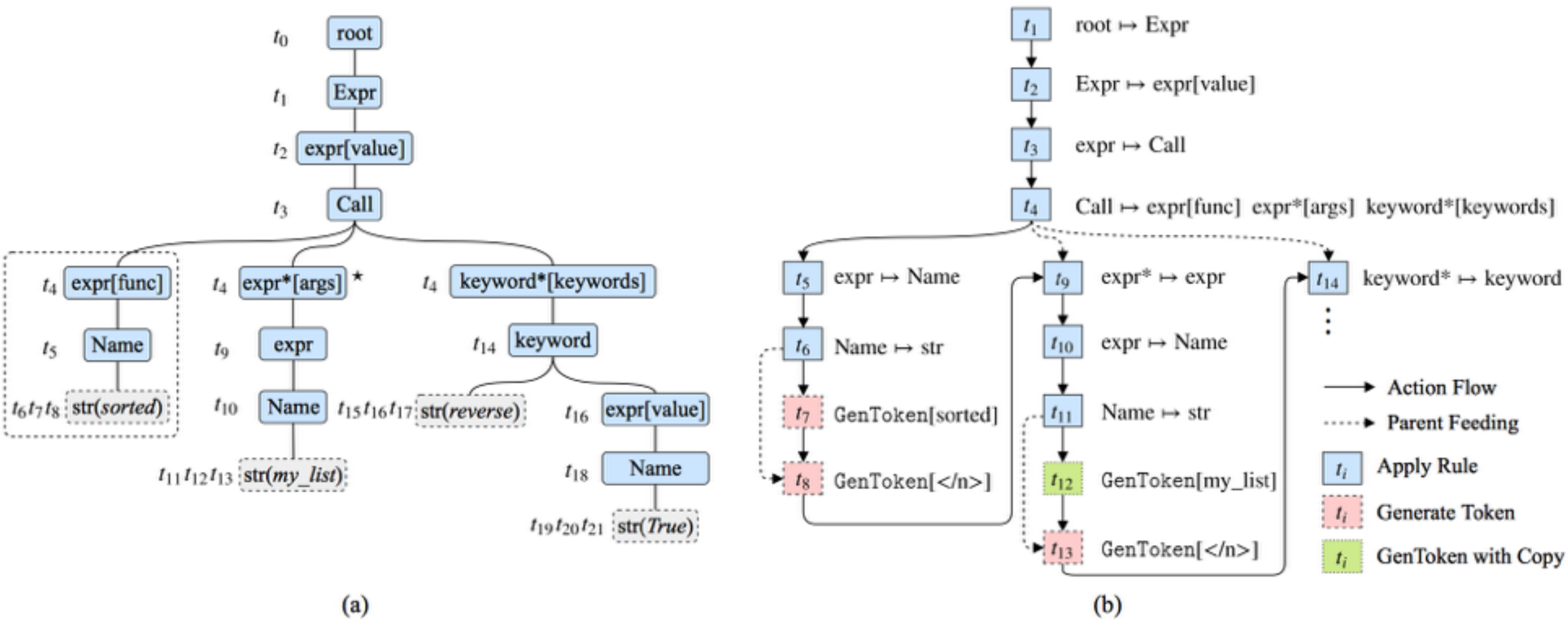


Encoder: googleNet

Decoder: LSTM

Reference: Vinyals et. al.: *Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge*

Code generation: seq2tree



Input: `sort my_list in descending order`

Code: `sorted(my_list, reverse=True)`

Sequence to sequence with connection to father node

Reference: Yin et. al.: *A Syntactic Neural Model for General-Purpose Code Generation*

Build your own! do not need to be from scratch!

References

- 俞栋，邓力 et.al. 解析深度学习：语音识别实践

- Pascanu et. al. *On the difficulty of training RNN*

- Hochreiter et. al. *Long Short-term Memory*

- Colah’s blog: *Understanding LSTM Networks*

- Zaremba et.al. *Recurrent Neural Networks Regularization*

- Bahdanau et. al.: *Neural Machine Translation by Jointly Learning to Align and Translate*

- Vinyals et. al.: *Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge*

- Yin et. al.: *A Syntactic Neural Model for General-Purpose Code Generation*