Code Completion with Neural Attention and Pointer Networks

Jian Li, Yue Wang, Irwin King, and Michael R. Lyu
The Chinese University of Hong Kong

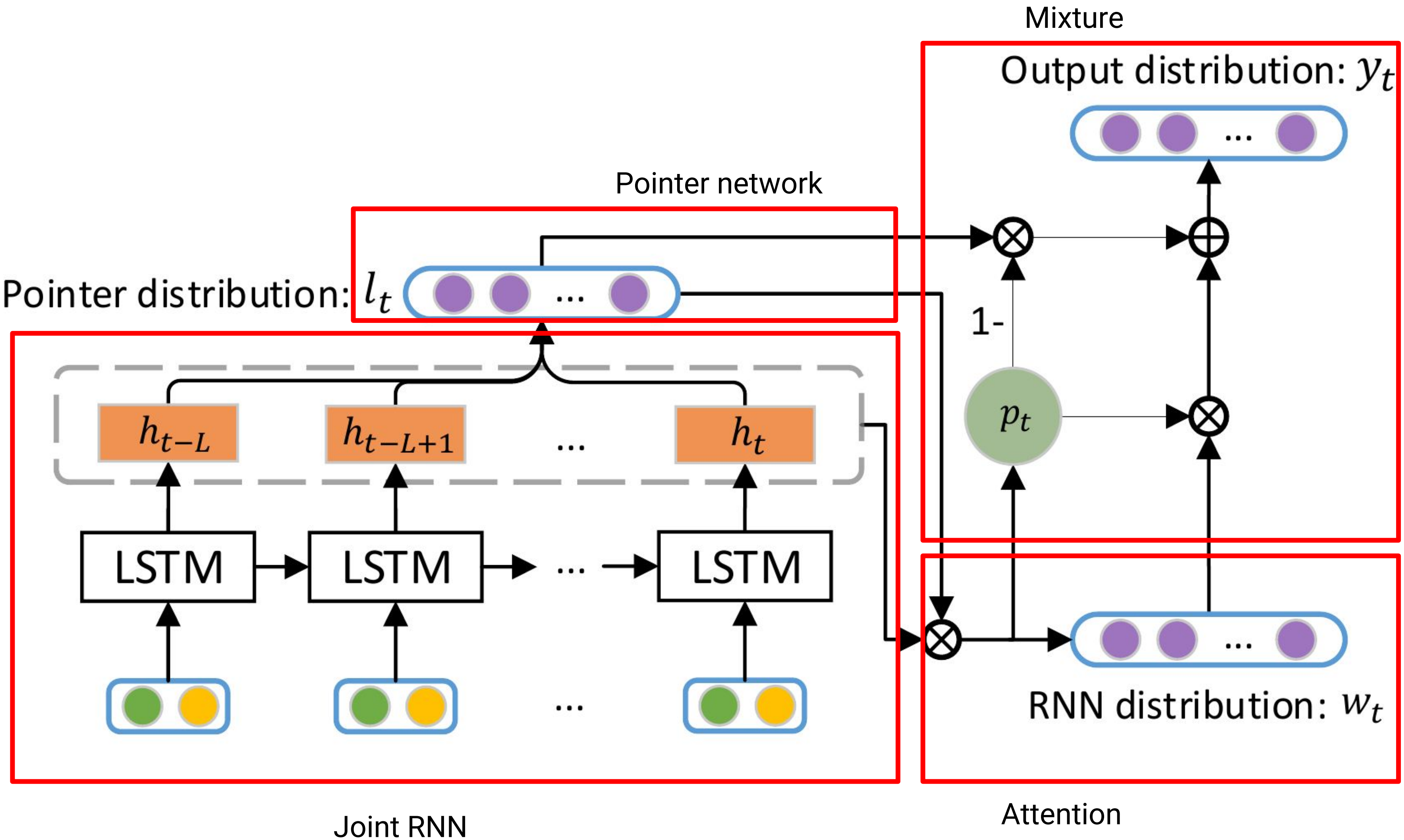
Presented by Ondrej Skopek

Goal: Predict out-of-vocabulary words using local context

(illustrative image)

Credits: van Kooten, P. neural_complete. https://github.com/kootenpv/neural_complete. (2017).

Pointer mixture networks



Credits: Li, J., Wang, Y., King, I. & Lyu, M. R. Code Completion with Neural Attention and Pointer Networks. (2017).

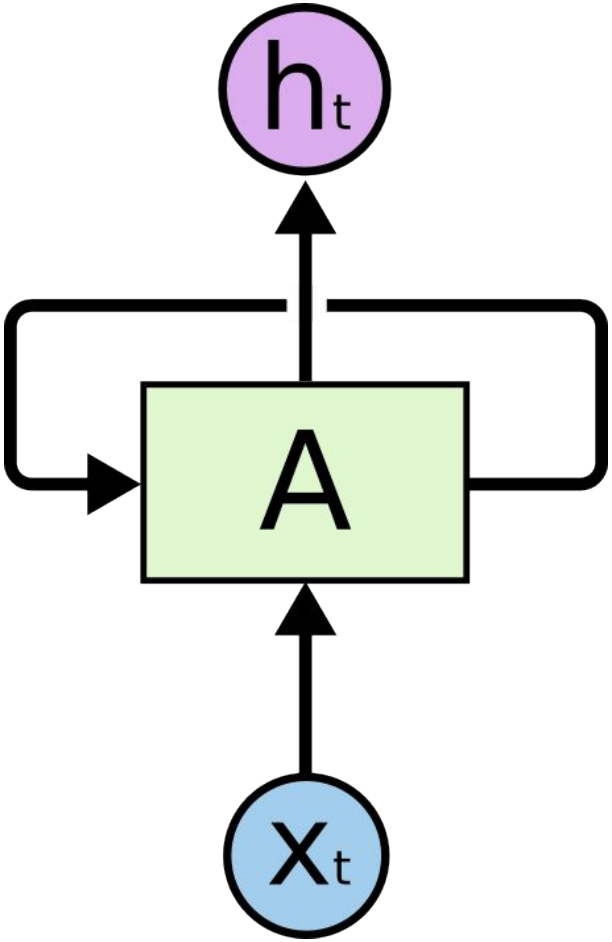
Outline

- Recurrent neural networks
- Attention
- Pointer networks

- Data representation
- Pointer mixture network

- Experimental evaluation
- Summary

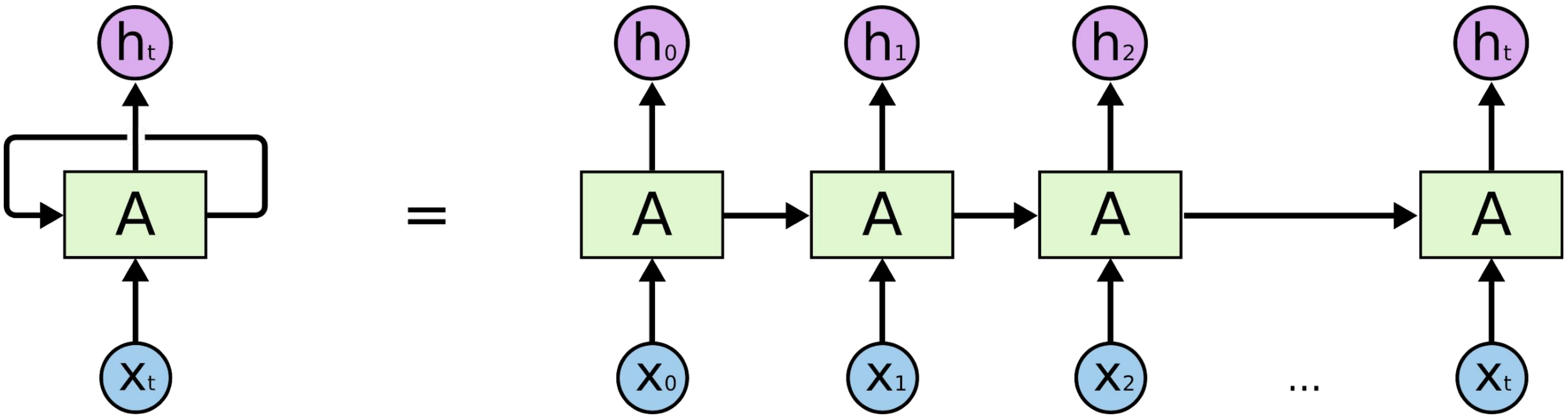
Recurrent neural networks



$$h_t = A(h_{t-1}, x_t)$$

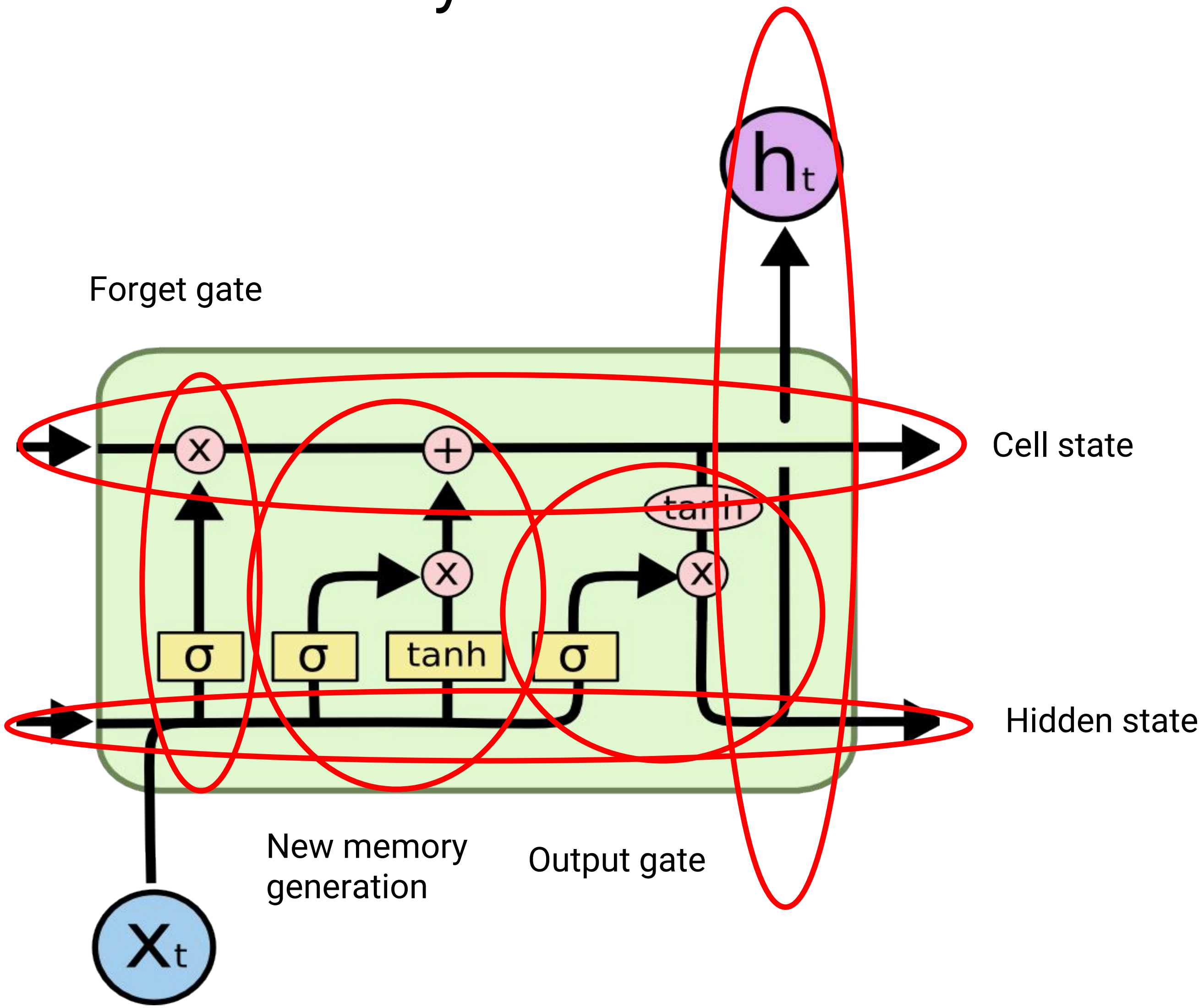
Credits: Olah, C. Understanding LSTM Networks. colah's blog (2015).

Recurrent neural networks – unrolling



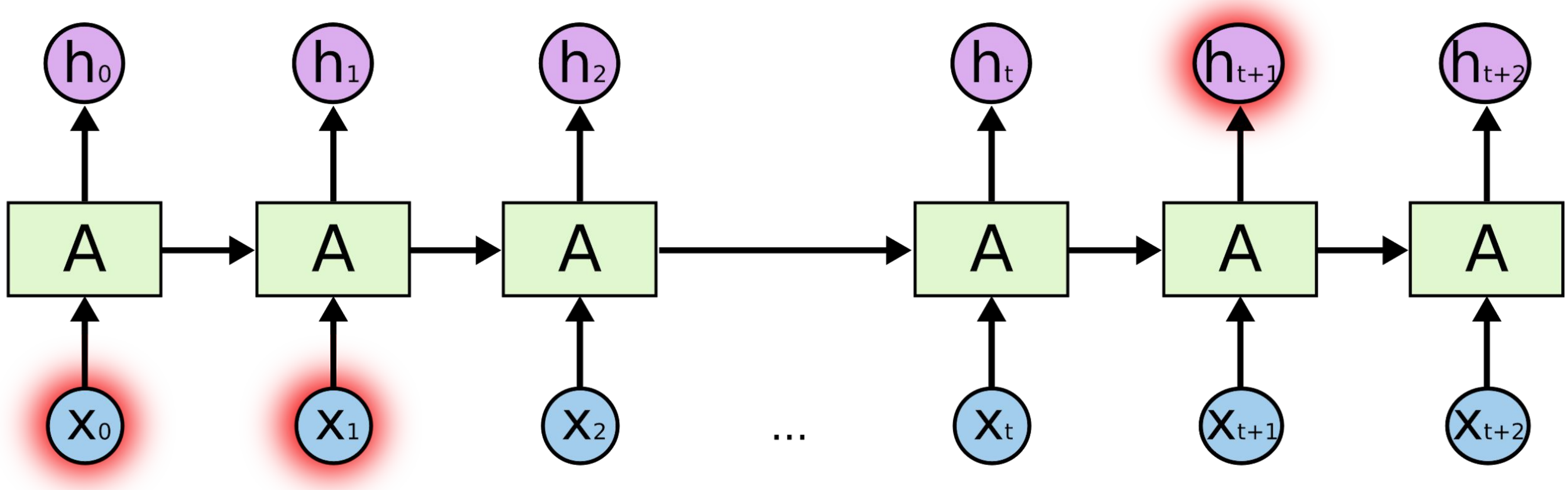
Credits: Olah, C. Understanding LSTM Networks. colah's blog (2015).

Long Short-term Memory



Credits: Hochreiter, S. & Schmidhuber, J. Long Short-term Memory. Neural Computation 9, 1735–1780 (1997).
Olah, C. Understanding LSTM Networks. colah's blog (2015).

Recurrent neural networks – long-term dependencies

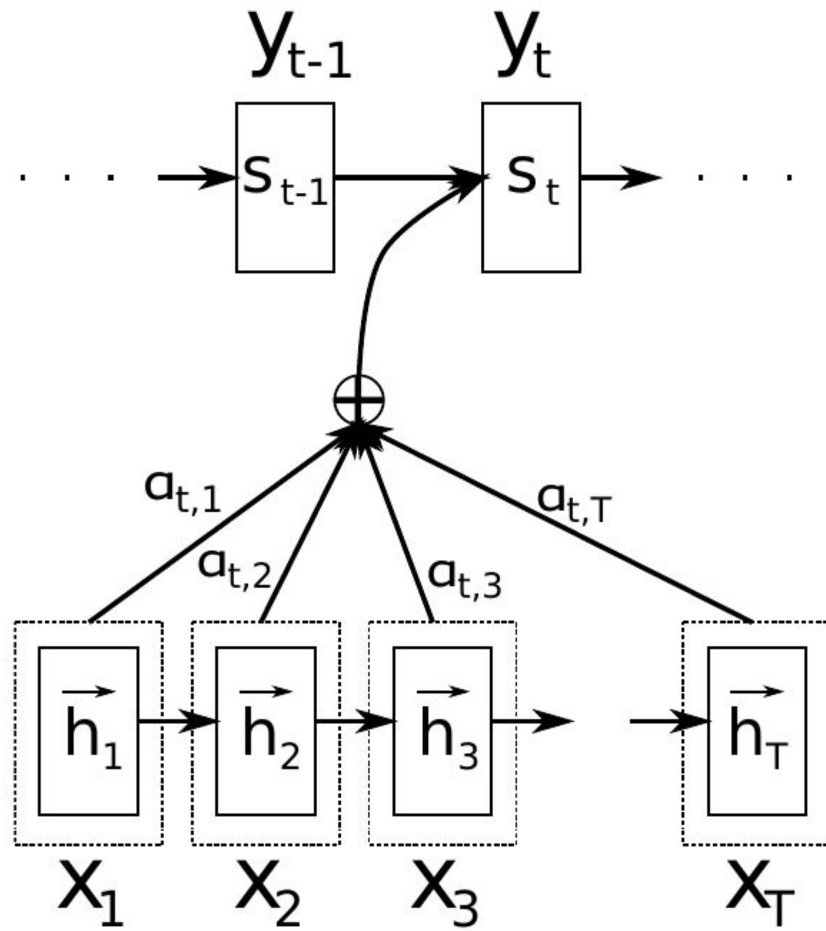


Credits: Olah, C. Understanding LSTM Networks. colah's blog (2015).

Attention

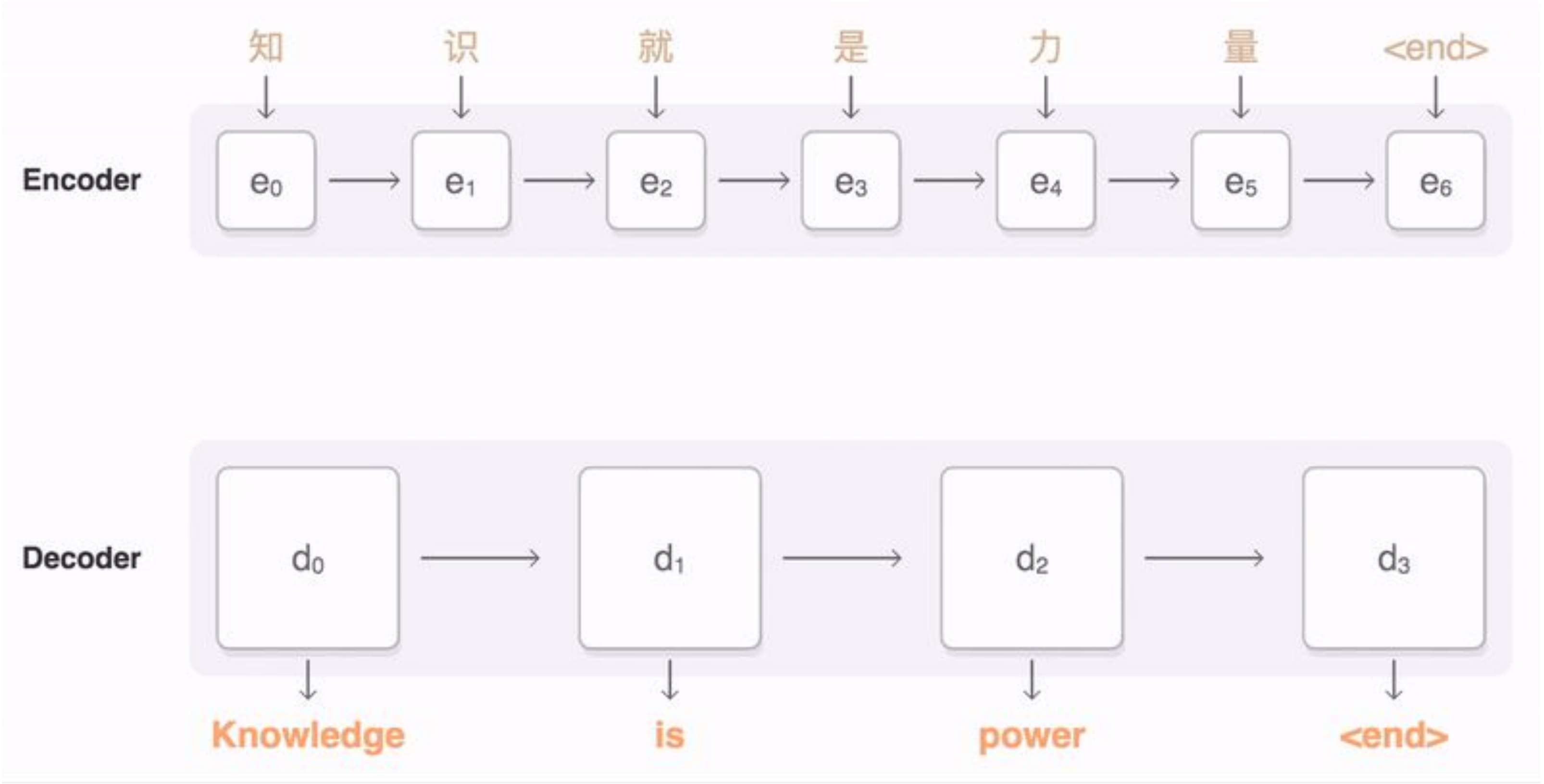
- Choose which context to look at when predicting
- Overcome the hidden state bottleneck

$$A^i_j = v^T \tanh(W_h h_j + W_s s_i)$$
$$\alpha^i = \mathrm{softmax}(A^i)$$
$$c_i = \sum_{j=1}^{n} \alpha^i_j h_j$$



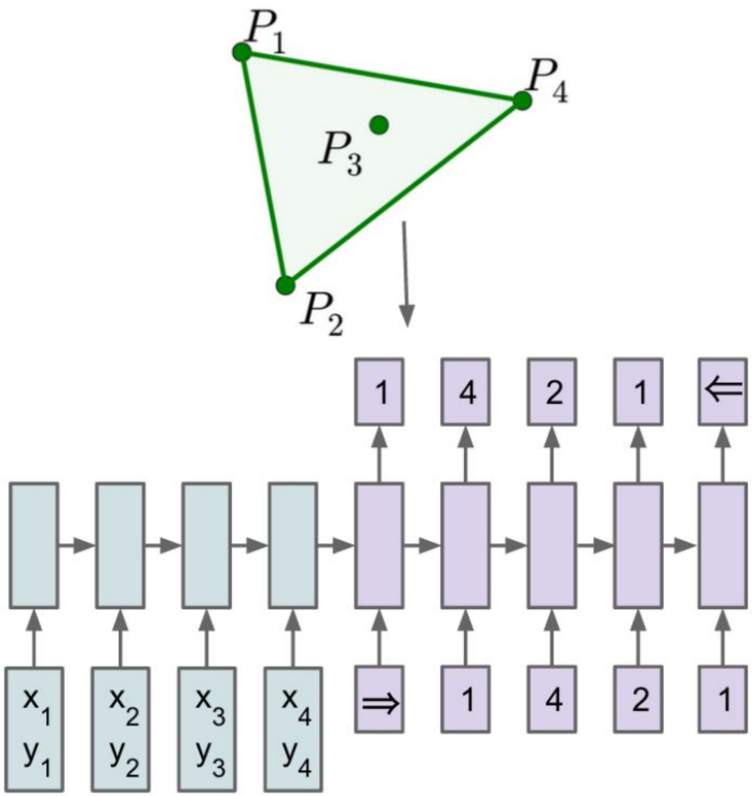
Credits: Bahdanau, D., Cho, K. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. (2014).

Attention (cont.)

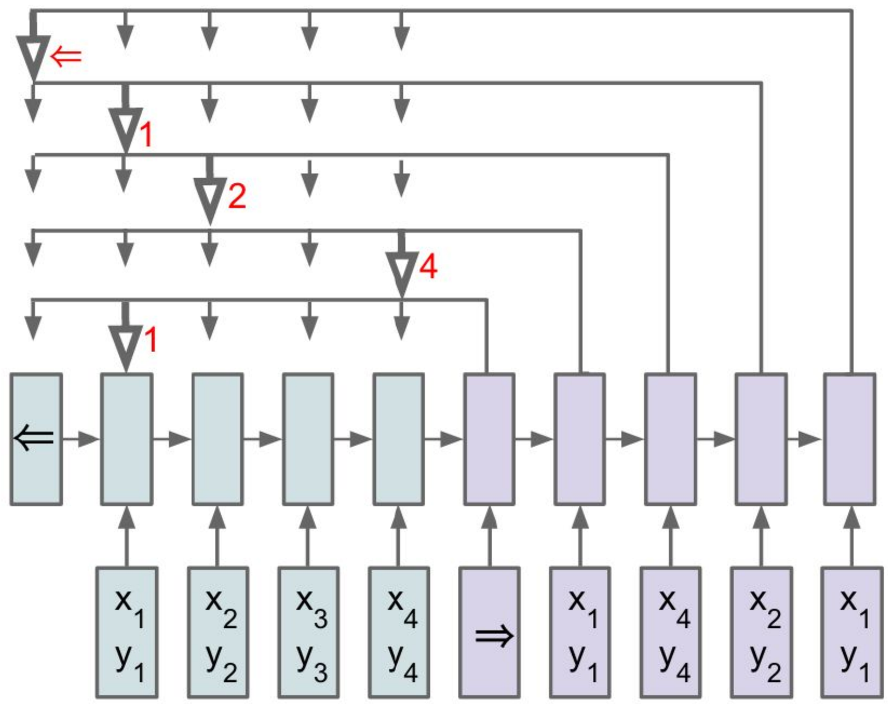


Credits: QI, X. Seq2seq. https://xiandong79.github.io/seq2seq-基础知识. (2017).
Bahdanau, D., Cho, K. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. (2014).

Pointer networks



(a) Sequence-to-Sequence



(b) Ptr-Net

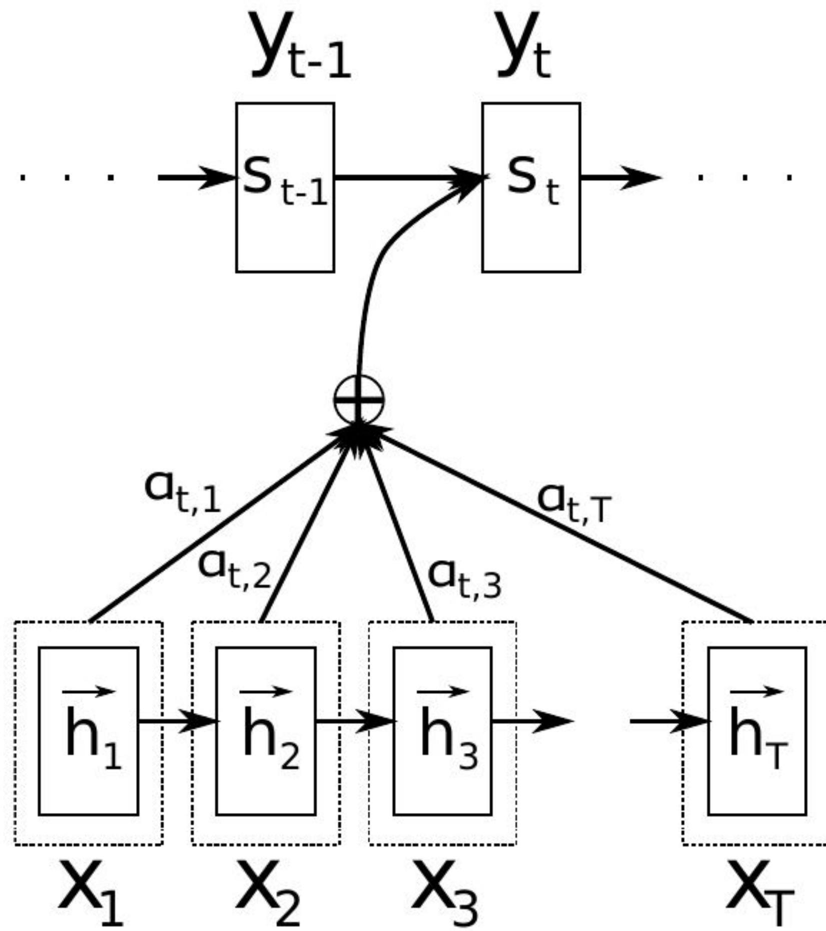
Credits: Vinyals, O., Fortunato, M. & Jaitly, N. Pointer Networks. (2015).

Pointer networks (cont.)

- Based on Attention
- Softmax over a dictionary of **inputs**
- Output models a conditional distribution of the next output token

$$A_j^i = v^T \tanh(W_h h_j + W_s s_i)$$

$$p(C_i | C_1, \ldots, C_{i-1}, \mathcal{P}) = \text{softmax}(A^i)$$



Credits: Vinyals, O., Fortunato, M. & Jaitly, N. Pointer Networks. (2015).
Bahdanau, D., Cho, K. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. (2014).

Outline

- Recurrent neural networks
- Attention
- Pointer networks

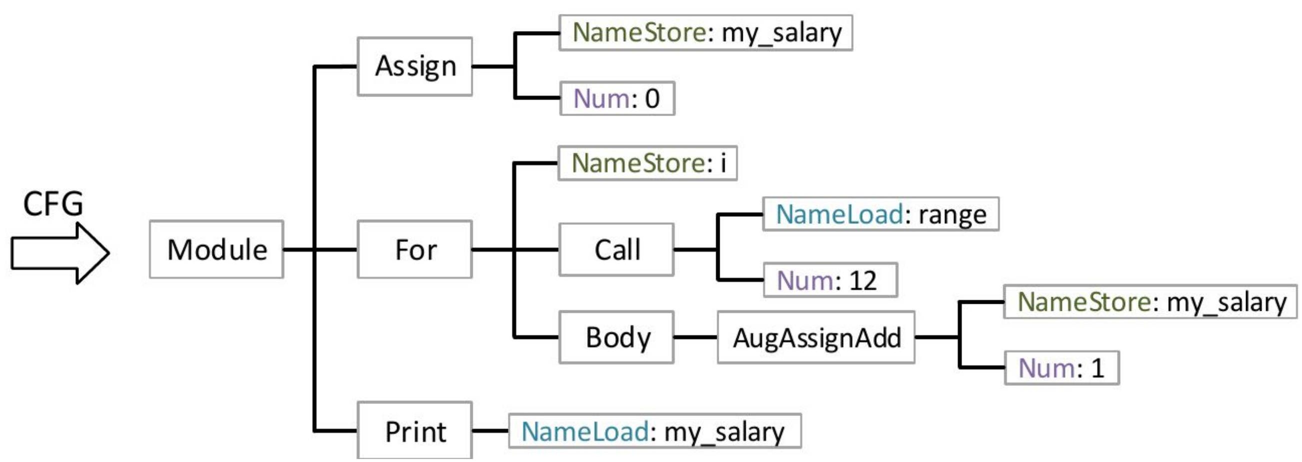
- Data representation
- Pointer mixture network

- Experimental evaluation
- Summary

Data representation

- Corpus of Abstract Syntax Trees (ASTs)
  - Parsed using a context-free grammar
- Each node has a type and a value (`type:value`)
  - Non-leaf value: `EMPTY`, unknown value: `UNK`, end of program: `EOF`
- Task: Code completion

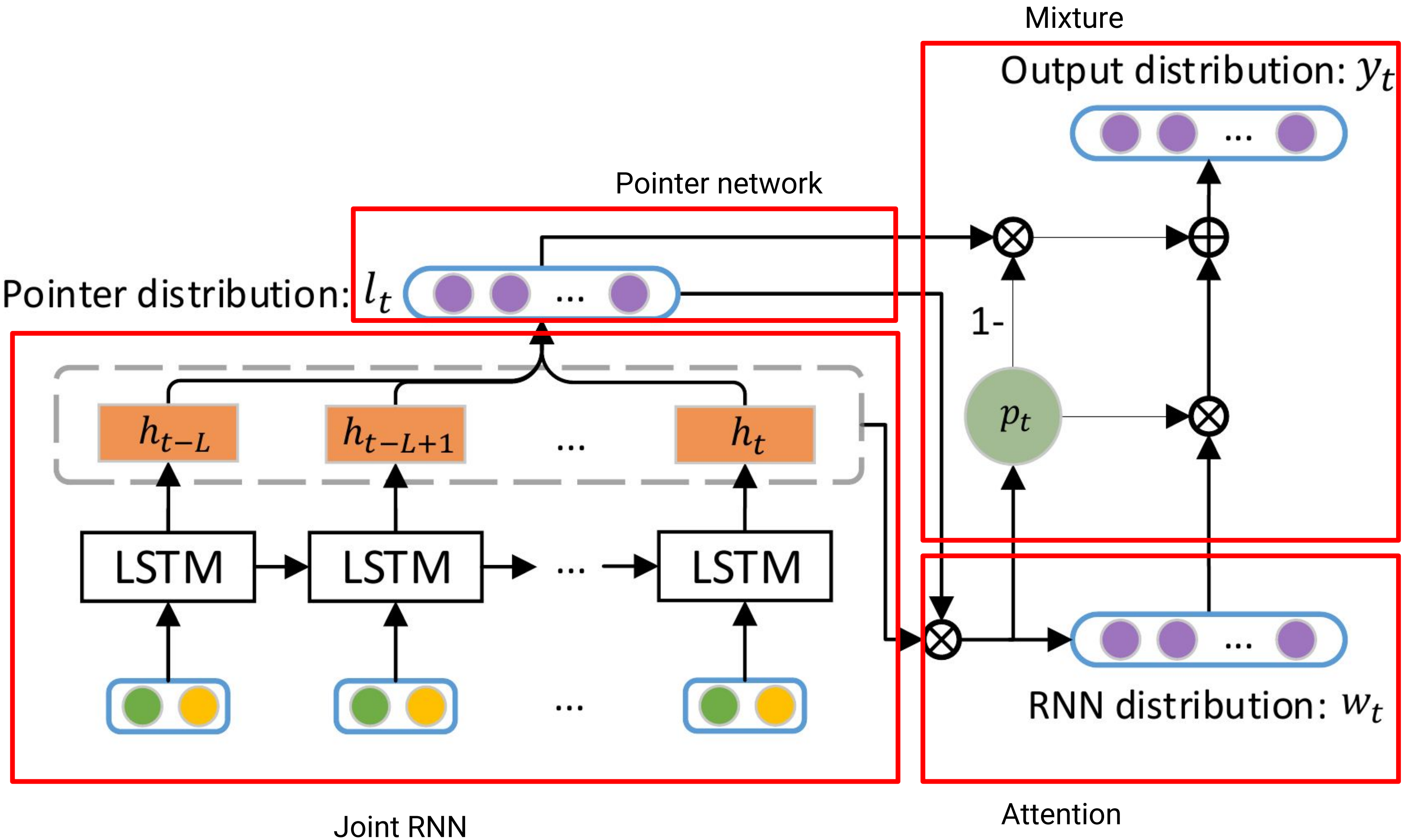
```
my_salary = 0
for i in range(12):
    my_salary += 1
print(my_salary)
```



- Task after serialization: Given a sequence of words, predict the next one

Credits: Li, J., Wang, Y., King, I. & Lyu, M. R. Code Completion with Neural Attention and Pointer Networks. (2017).

Pointer mixture networks

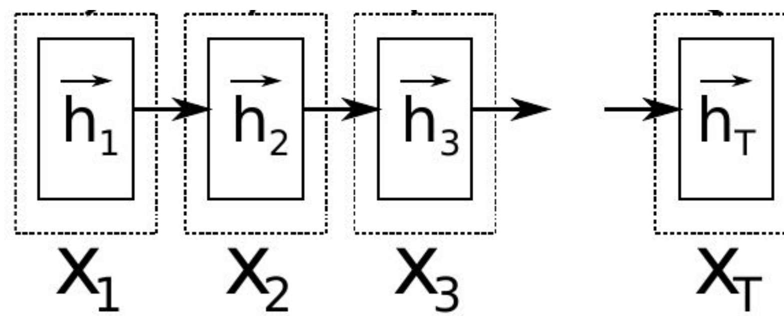


Credits: Li, J., Wang, Y., King, I. & Lyu, M. R. Code Completion with Neural Attention and Pointer Networks. (2017).

RNN with adapted Attention

- Intermediate goal
  - Produce two distributions at time t

$$w_t \in \mathbb{R}^V$$
$$l_t \in \mathbb{R}^L$$



- RNN with Attention (fixed unrolling)
  - L – input window size (L = 50)
  - V – vocabulary size (differs)
  - k – size of hidden state (k = 1500)

$$M_t = [h_{t-L}, \ldots, h_{t-1}] \in \mathbb{R}^{k \times L}$$
$$A_t = v^T \tanh\big(W_m M_t + (W_h h_t) 1_L^T\big)$$
$$\alpha_t = \mathrm{softmax}(A_t)$$
$$c_t = M_t \alpha_t^T$$

Credits: Vinyals, O., Fortunato, M. & Jaitly, N. Pointer Networks. (2015).
Bahdanau, D., Cho, K. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. (2014).

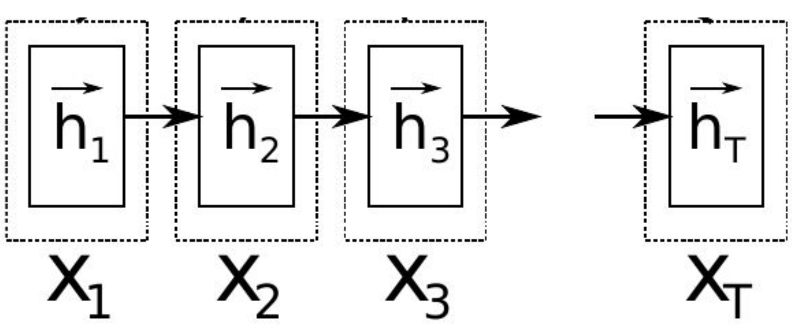
Attention & Pointer components

- Attention for the "decoder"
  - Condition on both the hidden state and context vector

$$G_t = \tanh(W_g[h_t; c_t])$$
$$w_t = \text{softmax}(W_v G_t + b_v)$$
$$\text{where } W_g \in \mathbb{R}^{k \times 2k}, W_v \in \mathbb{R}^{V \times k}$$

- Pointer network
  - Reuses Attention outputs

$$l_t = \alpha_t$$



Credits: Vinyals, O., Fortunato, M. & Jaitly, N. Pointer Networks. (2015).
Bahdanau, D., Cho, K. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. (2014).

Mixture component

- Combine the two distributions into one

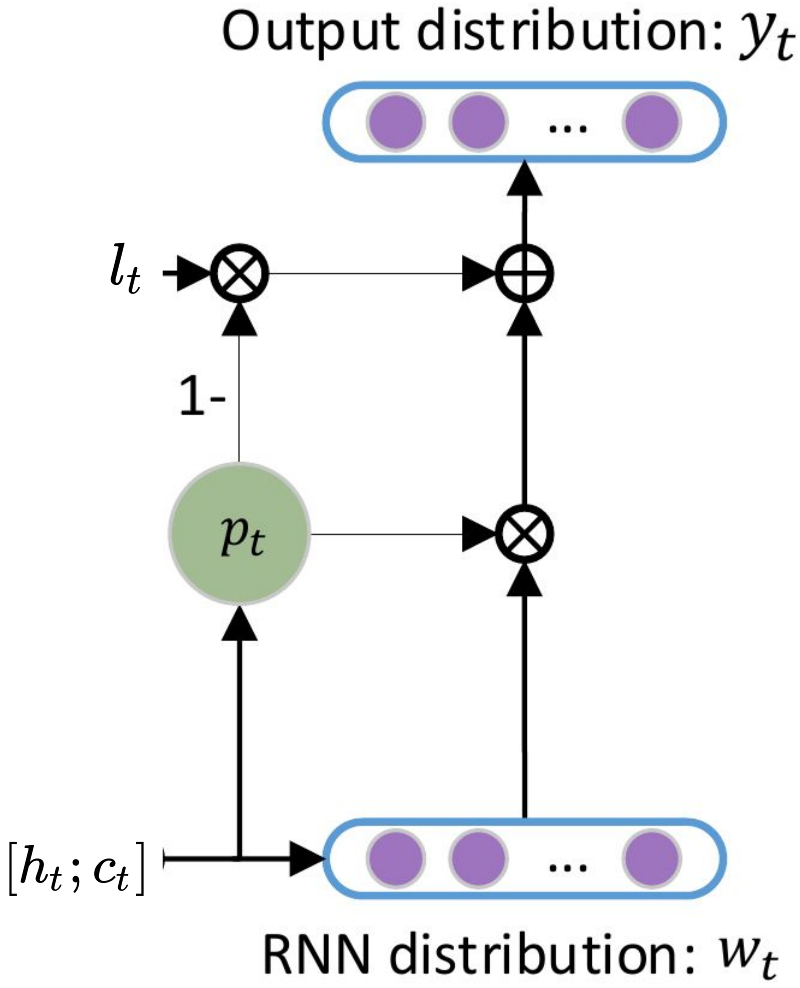
$$w_t \in \mathbb{R}^V$$
$$l_t \in \mathbb{R}^L$$

- Using

$$p_t = \sigma(W_p[h_t; c_t] + b_p)$$
$$y_t = [p_t w_t; (1 - p_t) l_t]$$

where

$$W_p \in \mathbb{R}^{2k \times 1},\ b_p \in \mathbb{R}$$



Credits: Li, J., Wang, Y., King, I. & Lyu, M. R. Code Completion with Neural Attention and Pointer Networks. (2017).

Outline

- Recurrent neural networks
- Attention
- Pointer networks

- Data representation
- Pointer mixture network

- Experimental evaluation
- Summary

Experimental evaluation

Data

- JavaScript and Python datasets
  - http://plml.ethz.ch
- Each program divided into segments of 50 consecutive tokens
  - Last segment padded with EOF
- AST data as described beforehand
  - Type embedding (300 dimensions)
  - Value embedding (1200 dimensions)
- No unknown word problem for types!

Table 1: Dataset Statistics

	JS	PY
Training Queries	$10.7 * 10^7$	$6.2 * 10^7$
Test Queries	$5.3 * 10^7$	$3.0 * 10^7$
Type Vocabulary	95	329
Value Vocabulary	$2.6 * 10^6$	$3.4 * 10^6$

Model & training parameters

- Single-layer LSTM, unrolling length 50
- Hidden unit size 1500
- Forget gate biases initialized to 1
- Cross-entropy loss function
- Adam optimizer (learning rate 0.001 + decay)
- Gradient clipping (L2 norm [0, 5])
- Batch size 128
- 8 epochs
- Trainiable initial states
  - Initialized to 0
  - All other parameters ~ Unif([-0.05, 0.05])

Credits: Li, J., Wang, Y., King, I. & Lyu, M. R. Code Completion with Neural Attention and Pointer Networks. (2017).

Experimental evaluation (cont.)

Training conditions

- Hidden state reset to trainable initial state only if segment from a different program, otherwise last hidden state reused
- If label UNK, set loss to 0 during training
- During training and test, UNK prediction considered incorrect

Labels

- Vocabulary: K most frequent words
- If in vocabulary: word ID
- If in attention window: label it as the last attention position
    - If not, labeled as UNK

	JS			PY		
Vocabulary Size (OoV Rate)	1k (20%)	10k (11%)	50k (7%)	1k (24%)	10k (16%)	50k (11%)
Vanilla LSTM	69.9%	75.8%	78.6%	63.6%	66.3%	67.3%
Attention-enhanced LSTM (ours)	71.7%	78.1%	80.6%	64.9%	68.4%	69.8%
Pointer Mixture Network (ours)	**73.2%**	**78.9%**	**81.0%**	**66.4%**	**68.9%**	**70.1%**

Credits: Li, J., Wang, Y., King, I. & Lyu, M. R. Code Completion with Neural Attention and Pointer Networks. (2017).

Comparison to other results

	JS		PY	
	TYPE	VALUE	TYPE	VALUE
Vanilla LSTM	87.1%	78.6%	79.3%	67.3%
Attention-enhanced LSTM (ours)	**88.6%**	80.6%	**80.6%**	69.8%
Pointer Mixture Network (ours)	-	81.0%	-	**70.1%**
LSTM ($\boxed{\text{Liu et al. 2016}}$)	84.8%	76.6%	-	-
Probabilistic Model (Raychev et al. 2016)	83.9%	**82.9%**	76.3%	69.2%

	JS_1k	PY_1k
Pointer Random Network	71.4%	64.8%
Attention-enhanced LSTM	71.7%	64.9%
Pointer Mixture Network	**73.2%**	**66.4%**

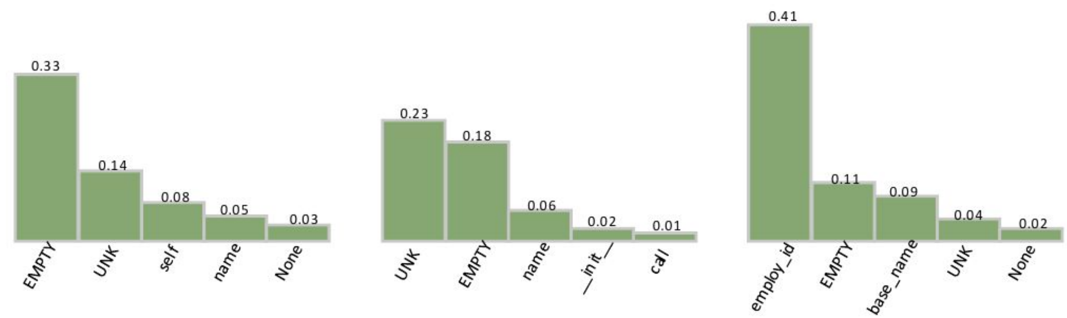
Credits: Li, J., Wang, Y., King, I. & Lyu, M. R. Code Completion with Neural Attention and Pointer Networks. (2017).

Example result

```
class Operator(Employee):
  def __init__(self, name, employee_id):
    super(Operator, self).__init__(name, Rank.OPERATOR)
    self.employee_id = employee_id

  def _dispatch_call(self, call, employees):
    for employee in employees:
      employee.take_call(call)

  def record_path(self, base_name):
    return os.path.join(base_name, str(self.___?___))
```



(a) Vanilla LSTM (b) Attention-enhanced LSTM (c) Pointer Mixture Network

Credits: Li, J., Wang, Y., King, I. & Lyu, M. R. Code Completion with Neural Attention and Pointer Networks. (2017).

Summary

- Applied neural language models to code completion
- Demonstrated the effectiveness of the Attention mechanism
- Proposed a Pointer Mixture Network to deal with the out-of-vocabulary values

Future work

- Encode more static type information
- Combine the two distributions in a different way
- Use both backward and forward context to predict the given node
- Attempt to learn longer dependencies for out-of-vocabulary values (L>50)

Credits: Li, J., Wang, Y., King, I. & Lyu, M. R. Code Completion with Neural Attention and Pointer Networks. (2017).

Thank you for your attention!