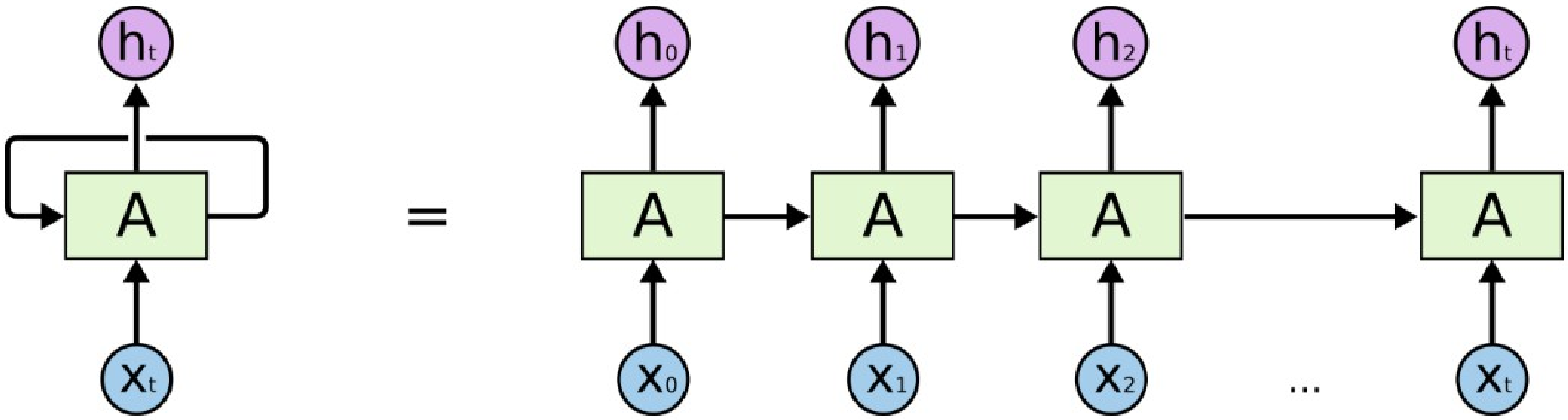
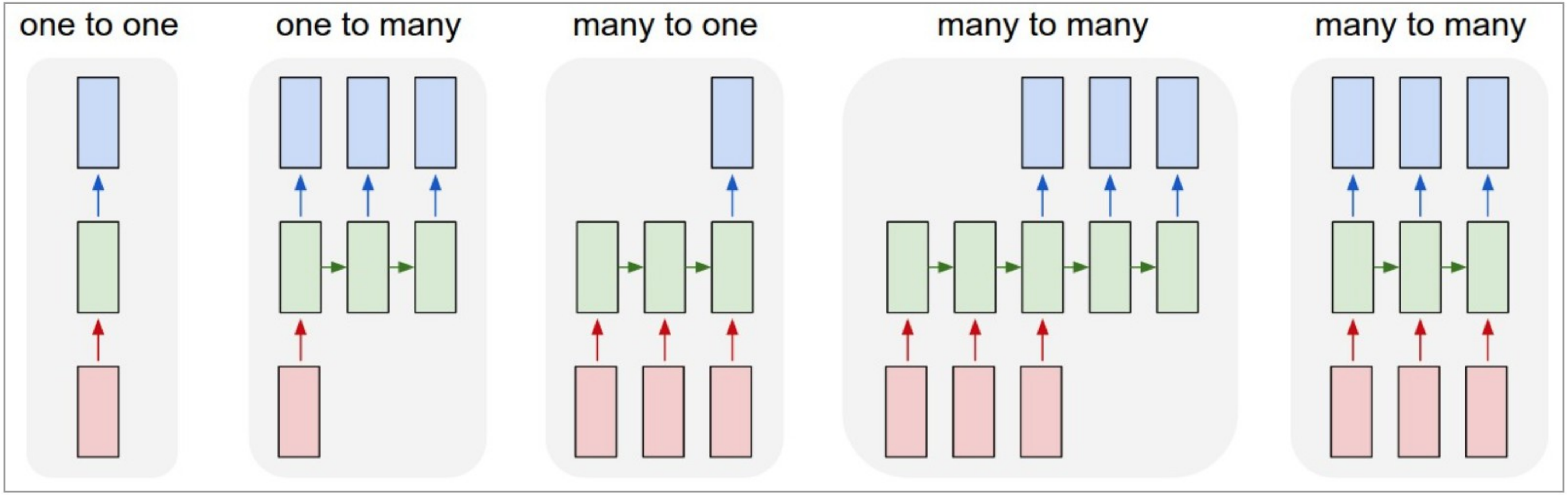
- Recurrent neural networks
- Encoder Decoder + Attention
- Characters and Copying Mechanism

Recurrent neural network schema

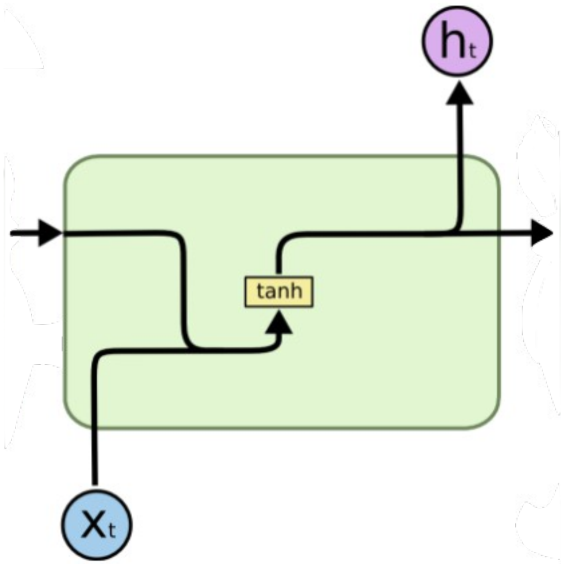
Rnn Cell **Rnn unrolled**



Recurrent neural network architectures

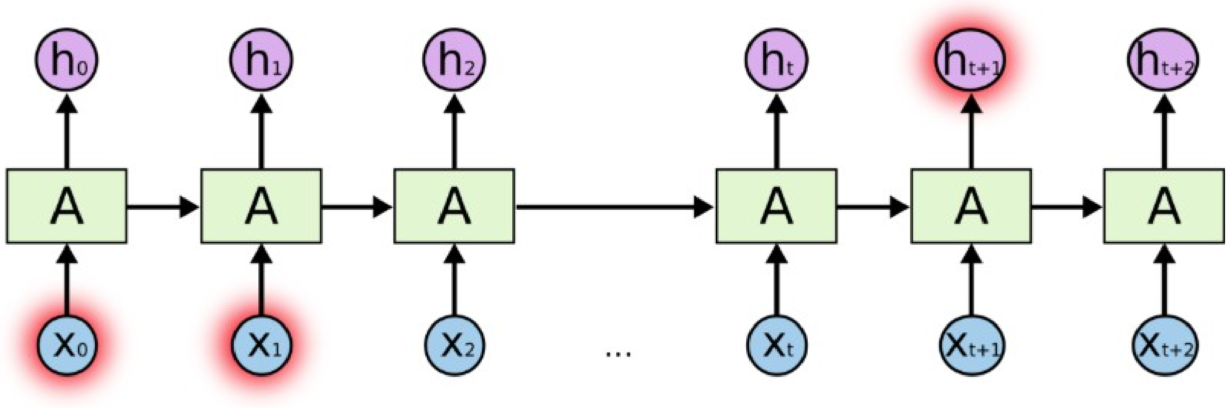


Vanilla Rnn Cell



$$h_t = tanh(Wh_{t-1} + UX_t + b)$$

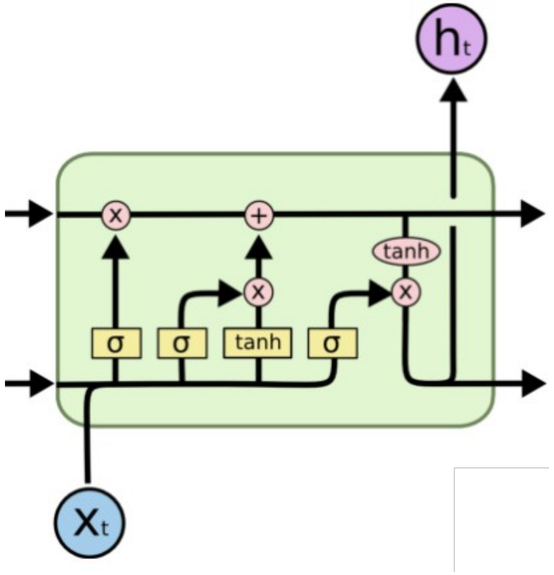
Long Term Dependency Problem



Backpropagation applied to RNN is Backpropagation through time (BPTT)

Problem: BPTT = a lot of gradient multiplications --> vanishing gradient

LSTMs: a long term dependency solution



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

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

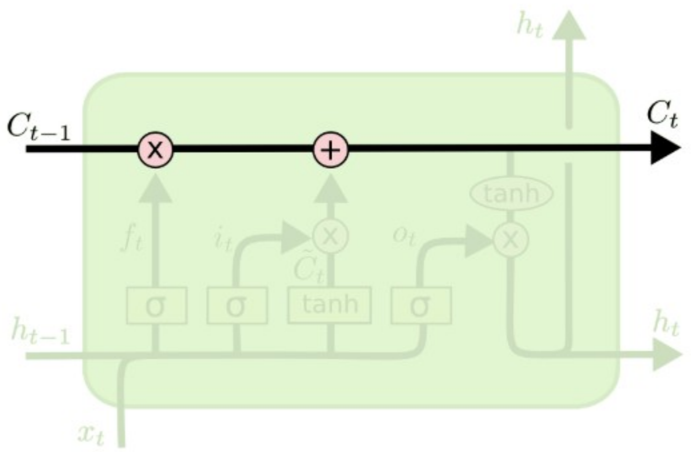
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

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

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

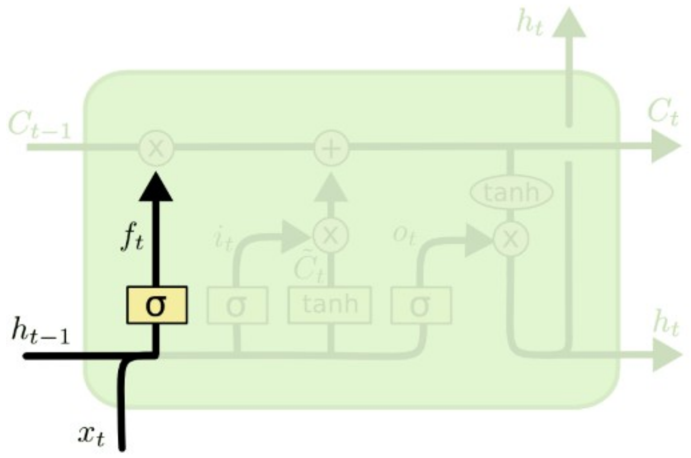
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTMs: a long term dependency solution



Cell state: let the information run free!

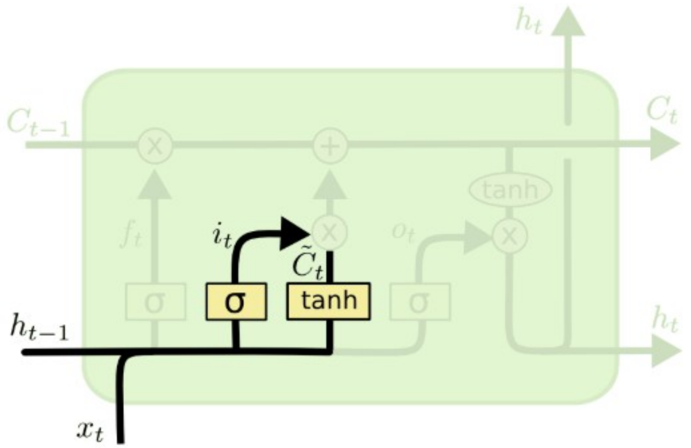
LSTMs: a long term dependency solution



Forget gate

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

LSTMs: a long term dependency solution

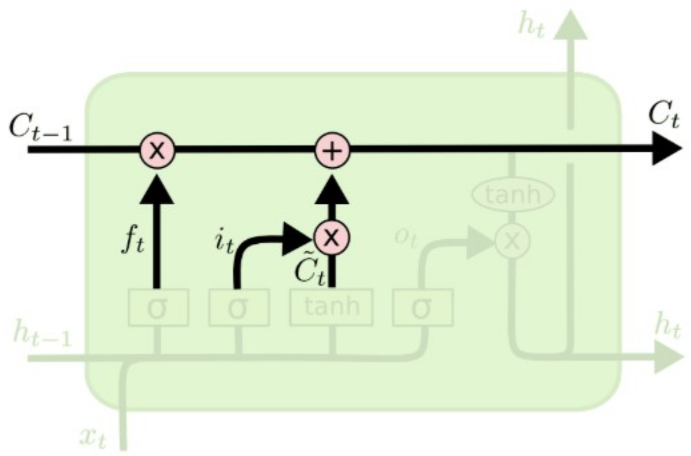


Input gate

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

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

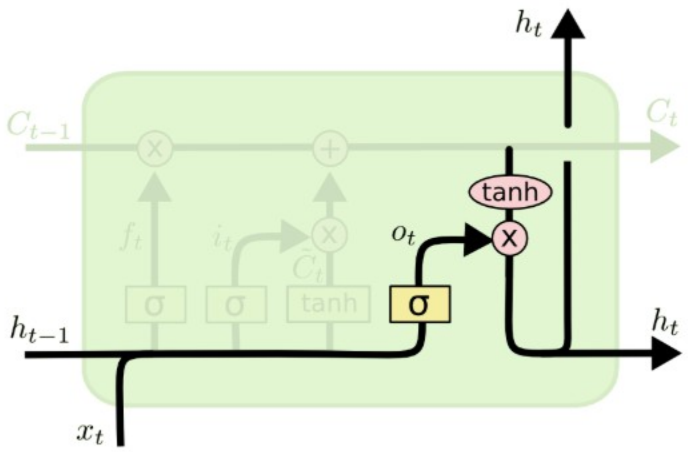
LSTMs: a long term dependency solution



Cell state update

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

LSTMs: a long term dependency solution

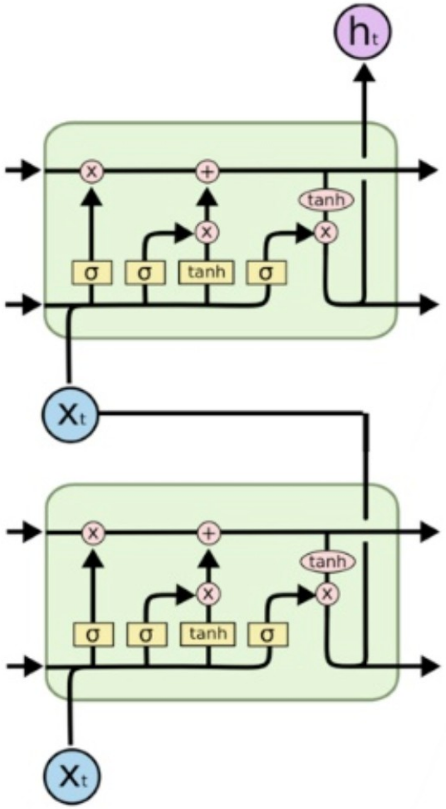


Output gate

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

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

Stacked Lstm: going deep



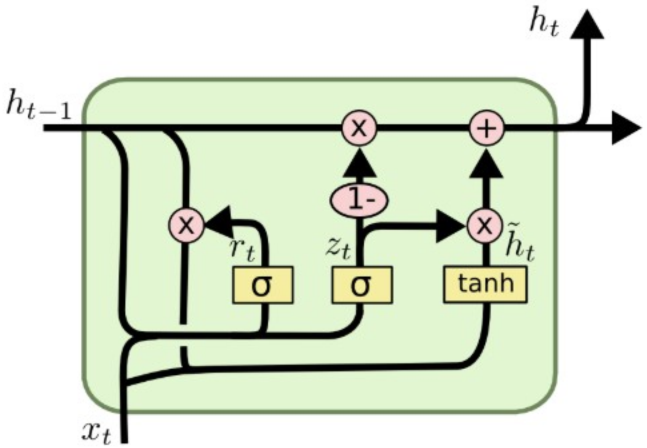
”

... building a deep RNN by stacking multiple recurrent hidden states on top of each other. This approach potentially allows the hidden state at each level to operate at different timescale

How to Construct Deep Recurrent Neural Networks, 2013, Pascanu & al.

GRUs: Gated Recurrent Units

Easier to implement and evaluate [than LSTM]



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

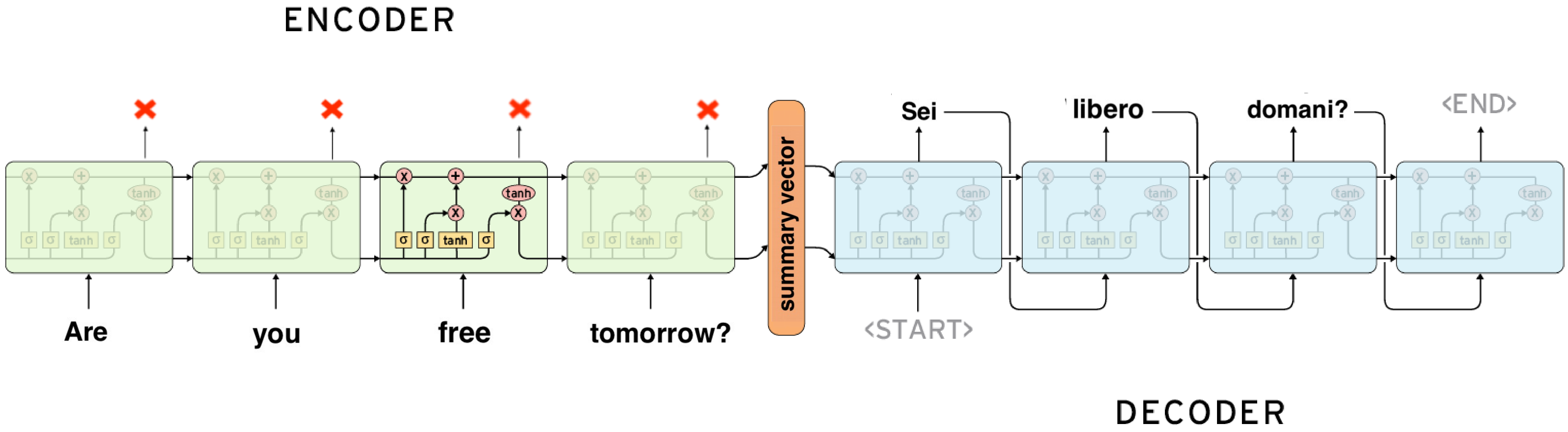
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014, Cho & al.

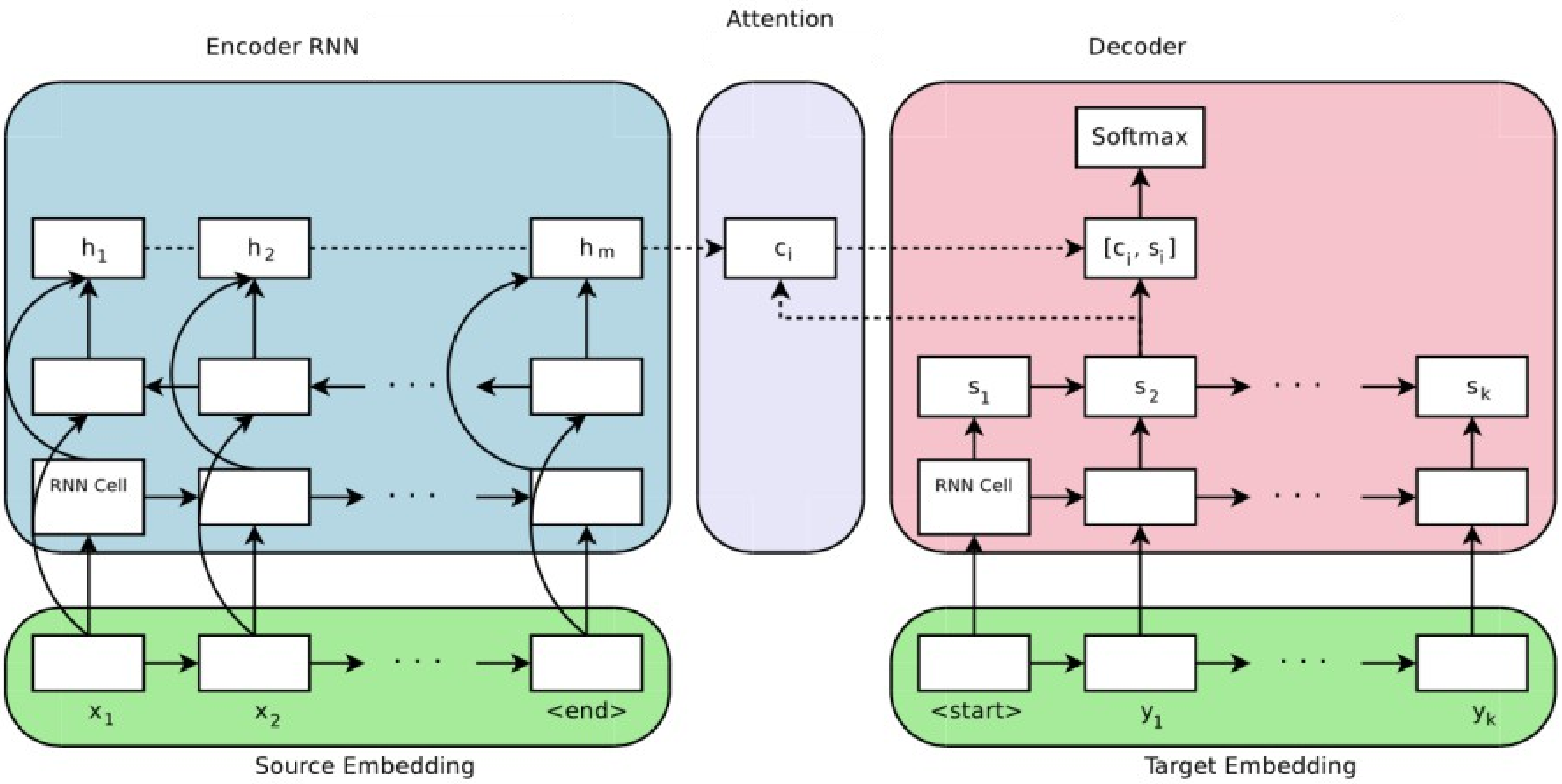
- Recurrent neural networks
- Encoder Decoder + Attention
- Characters and Copying Mechanism

Encoder Decoder architecture without Attention

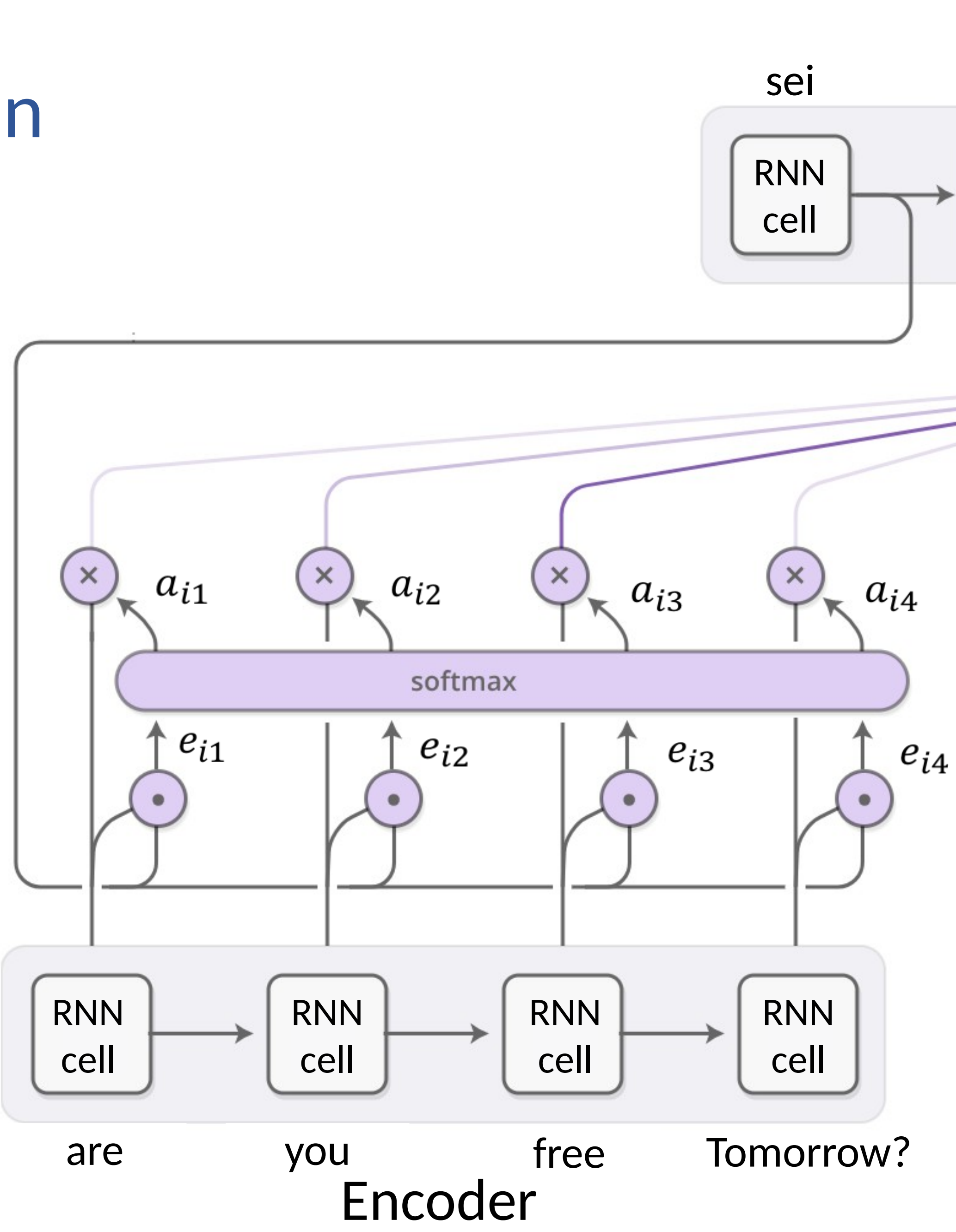


Problem: The fixed size summary vector has to keep all the semantic information!

Encoder Decoder architecture with Attention



Attention



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

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

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

Attention

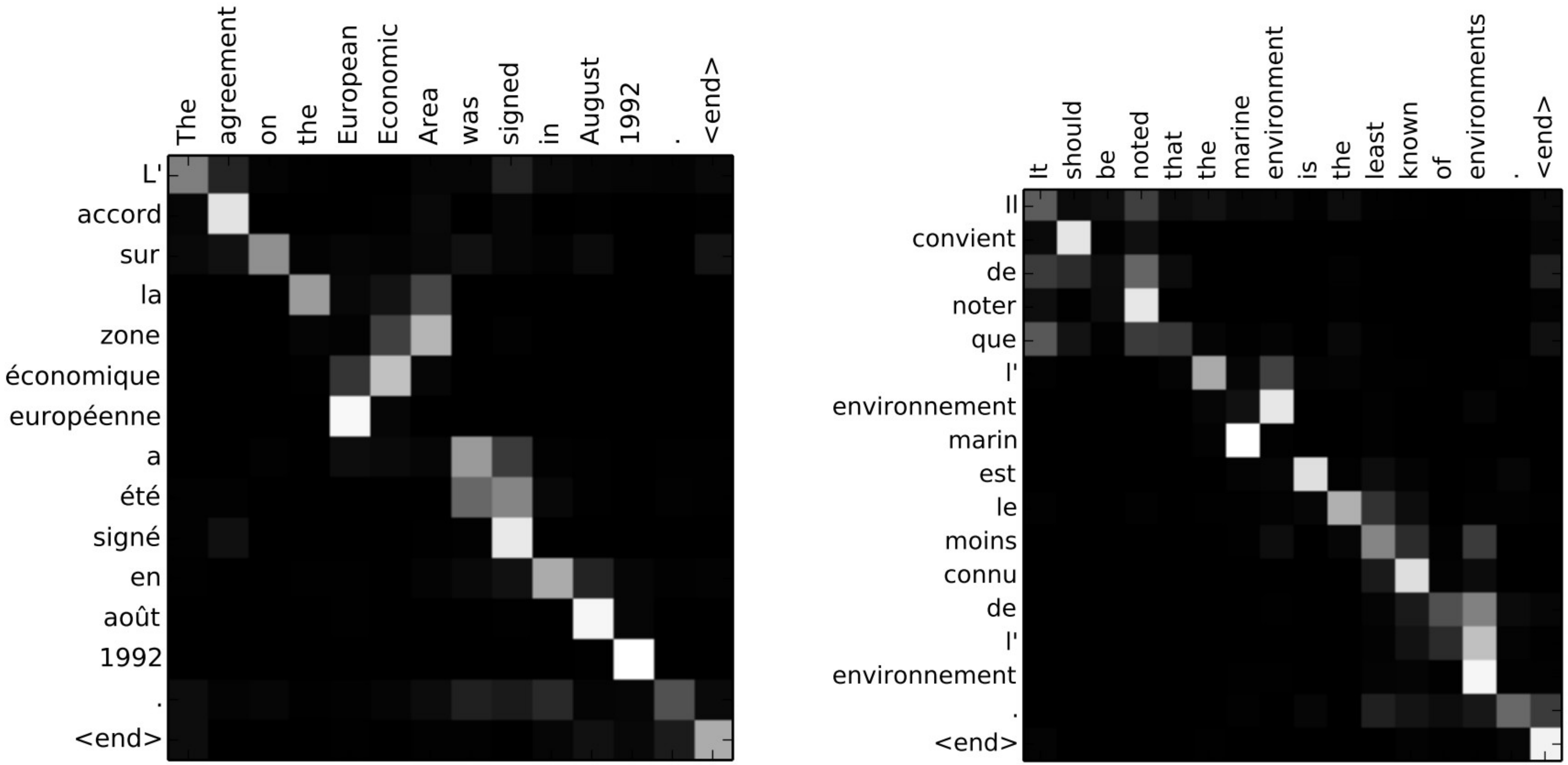
Bahdanau attention:

$$a(s_{i-1}, h_j) = v_a^\top \tanh\left(W_a s_{i-1} + U_a h_j\right)$$

Dot attention or Fast attention:

$$a(s_{i-1}, h_j) = < W_a s_{i-1} , U_a h_j >$$

Results



(a) (b)

Each pixel shows the weight α_{ij} of the annotation of the j-th source word for the i-th target word , in grayscale (0: black, 1: white).

- Recurrent neural networks
- Encoder Decoder + Attention
- Copying Mechanism and characters

Copy Mechanism

Problem: **What happens if we want a word which is not in the output vocabulary, but instead in the input sentence?**

Say we want to copy a name:

The network will probably try to use a name it saw during training…

Input -> hello, my name is Giovanni **output -> Ciao, mi chiamo Giacomo**

From words to characters

Why characters over words:

- **Smaller vocabulary (from ~ 100.000 to 70 tokens)**
- **No need for tokenization**
- **No need for delexicalization**
- **More general (same alphabet for different languages)**

Why not:

- **Difficult to align (repetitions)**
- **Characters have no meaning,**

  (embedding less useful)

E2E dataset sample

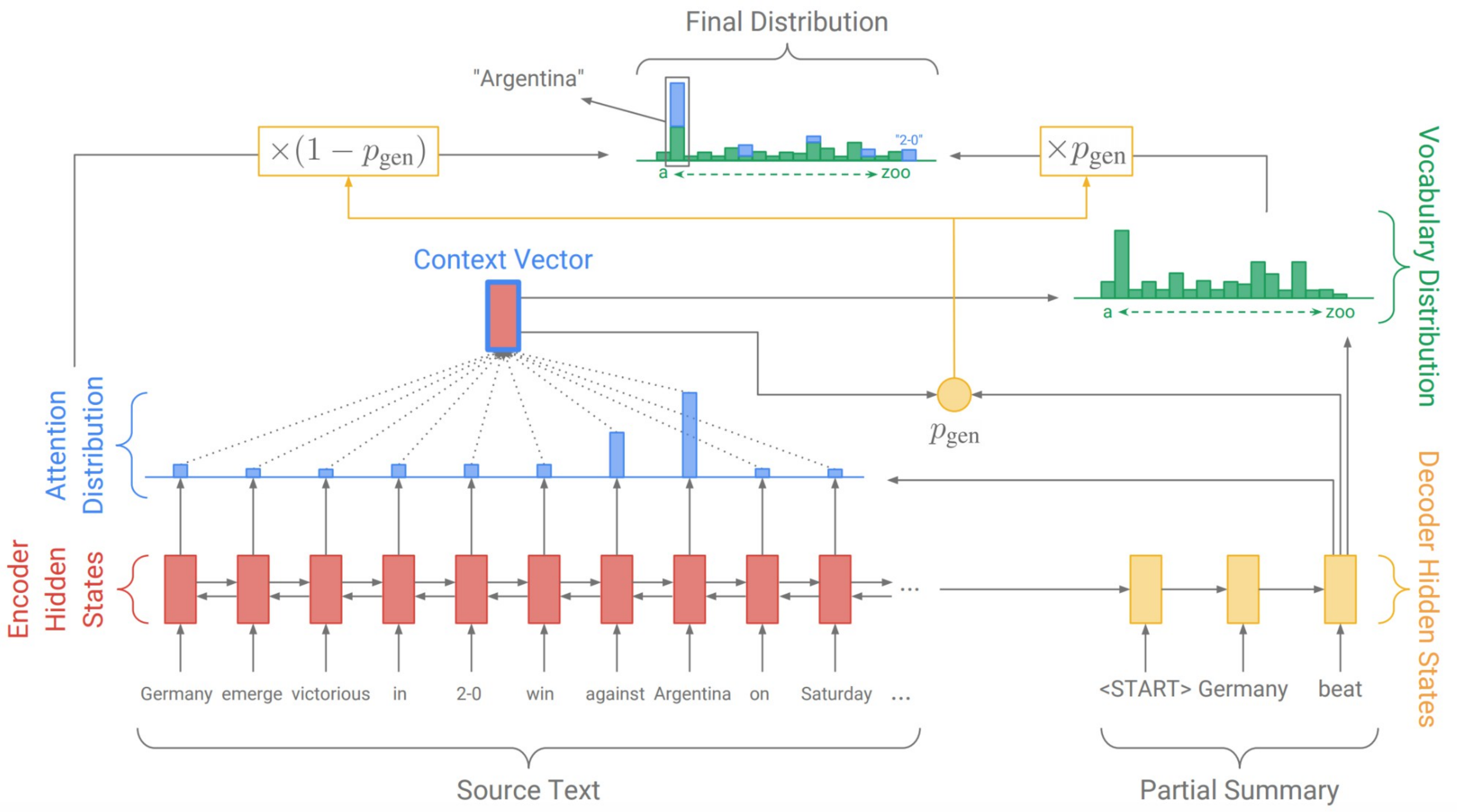
Flat MR	NL reference
name[Loch Fyne], eatType[restaurant], food[French], priceRange[less than £20], familyFriendly[yes]	Loch Fyne is a family-friendly restaurant providing wine and cheese at a low cost. Loch Fyne is a French family friendly restaurant catering to a budget of below £20. Loch Fyne is a French restaurant with a family setting and perfect on the wallet.

Copy with chars

The idea is to use a soft switch, *Pgen*, that learns to copy or generate the proper token.

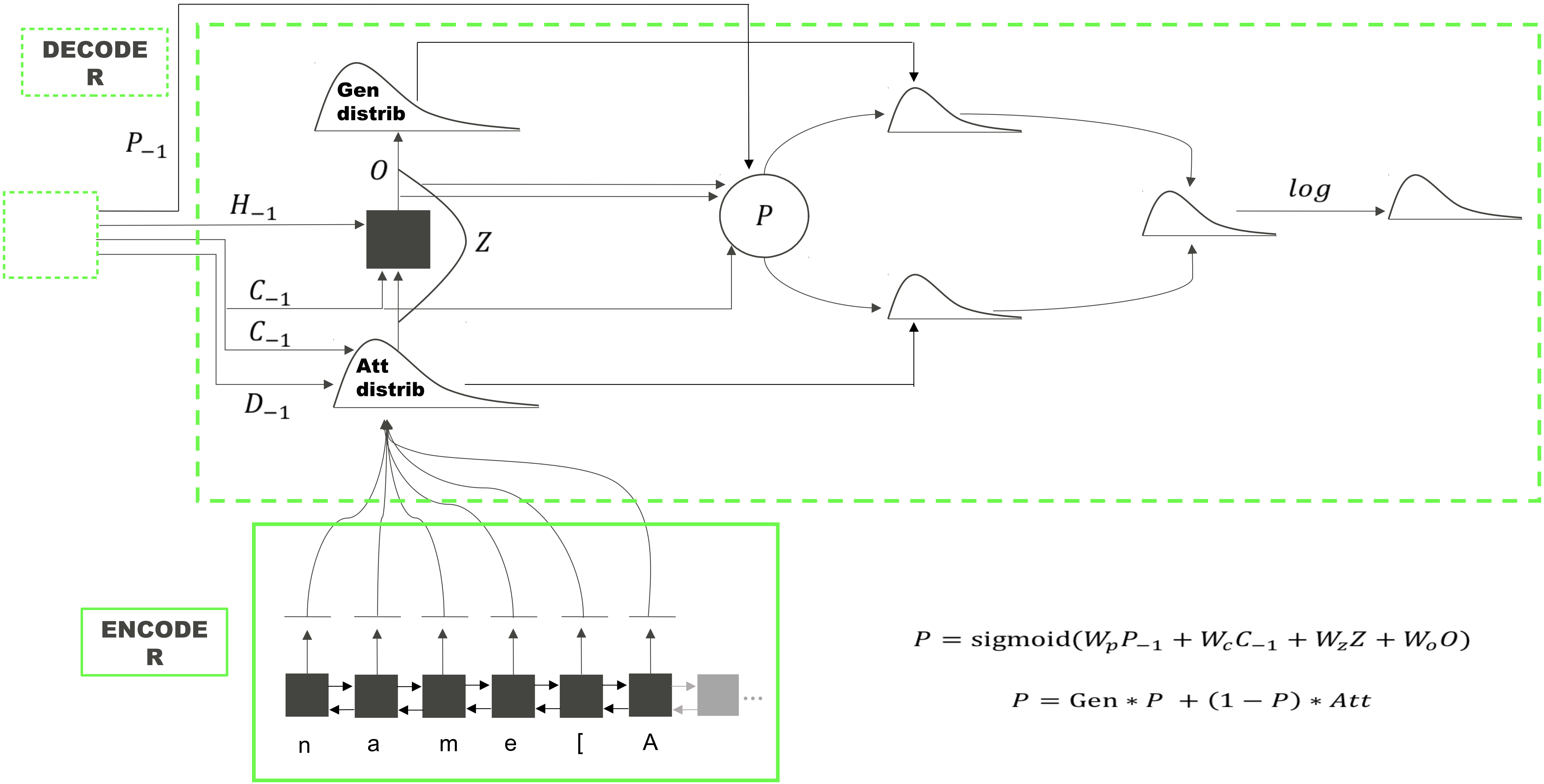
Copy = use the attention distribution.

Generate = use the standard Rnn Cell output.



$$P_{\text{final}}(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) \sum_{i:w_i=w} a_i$$

Get To The Point: Summarization with Pointer-Generator Networks, ACL 2017, Abigail See & al.



$$P = \text{sigmoid}(W_p P_{-1} + W_c C_{-1} + W_z Z + W_o O)$$

$$P = \text{Gen} * P + (1 - P) * Att$$

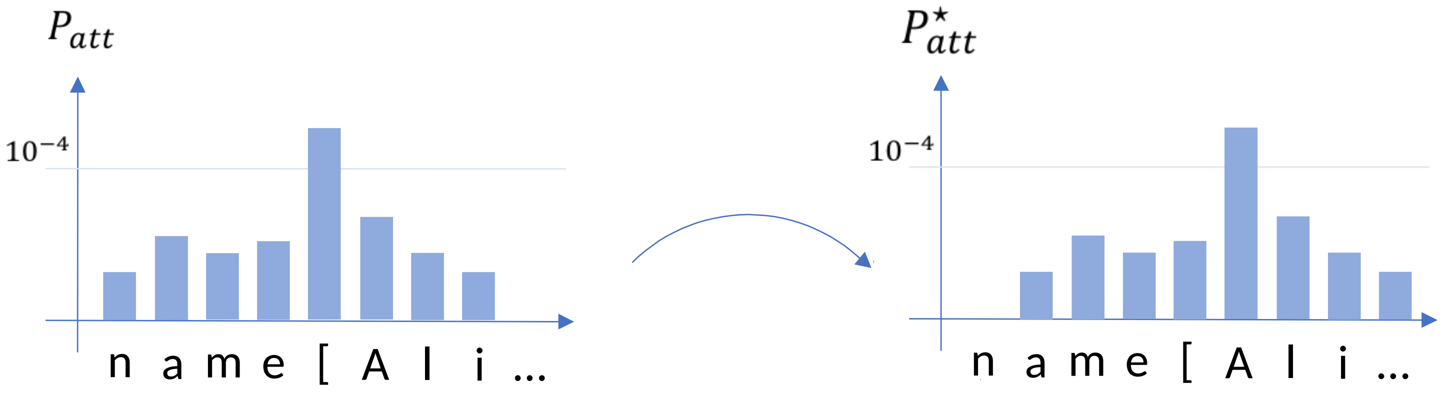
Some results

Input:
name[**La Boite en Bois**], eatType[coffee shop], food[Chinese], area[riverside], near[Crowne Plaza Hotel], familyFriendly[no]

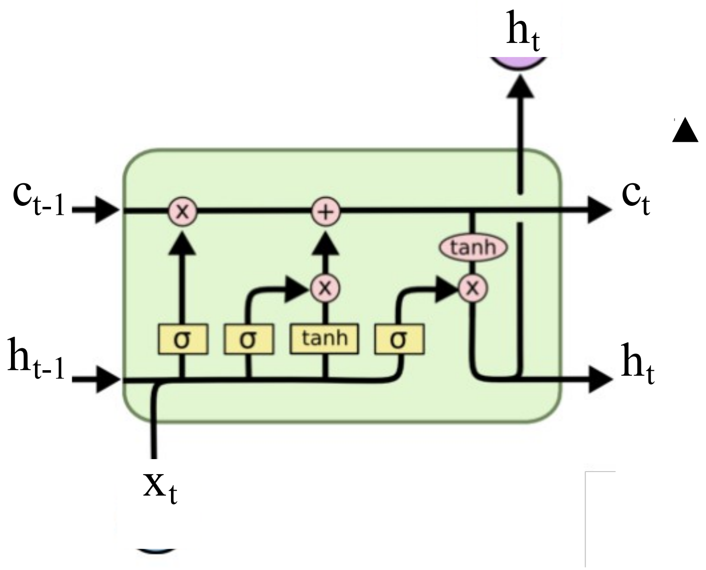
Output:
La Boite en Bois is a coffee shop that serves Chinese food and is located in the riverside area near Crowne Plaza Hotel. It is not family friendly.

Input:
name[**Tre Pomodori**], eatType[coffee shop], food[Chinese], area[riverside], near[Raja Indian Cuisine], priceRange[£30-35]

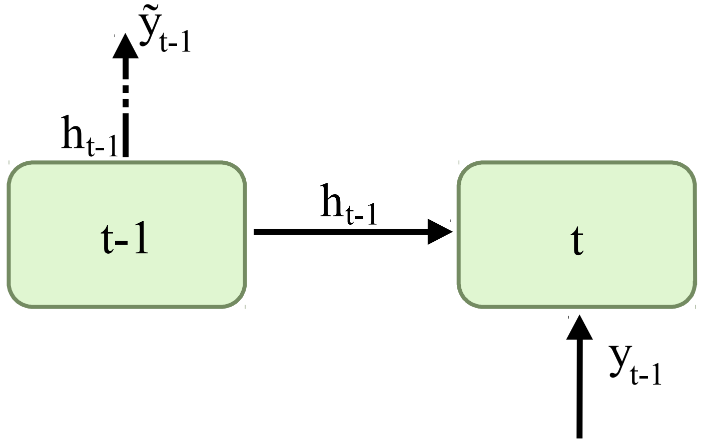
Output:
Tre Pomodori is a coffee shop that serves Chinese food in the more than £30 price range. It is located in the riverside area near Raja Indian Cuisine.



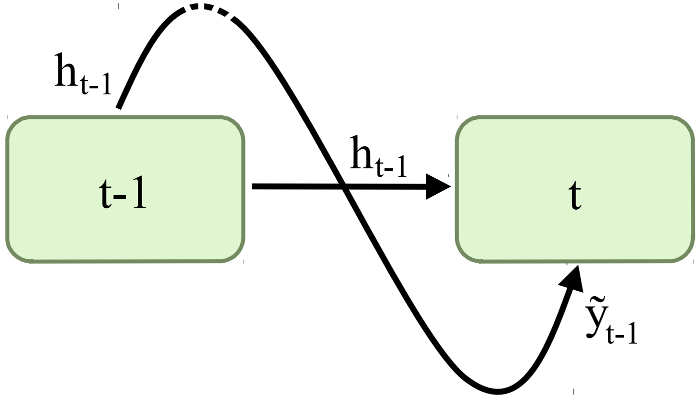
P_{att}
10^{-4}
n a m e [A l i ...
$P^{\star}_{att}$
10^{-4}
n a m e [A l i ...



h_t
c_{t-1}
c_t
tanh
σ
σ
tanh
σ
h_{t-1}
h_t
x_t



$\tilde{y}_{t-1}$
h_{t-1}
t-1
h_{t-1}
t
y_{t-1}



h_{t-1}
t-1
h_{t-1}
t
$\tilde{y}_{t-1}$