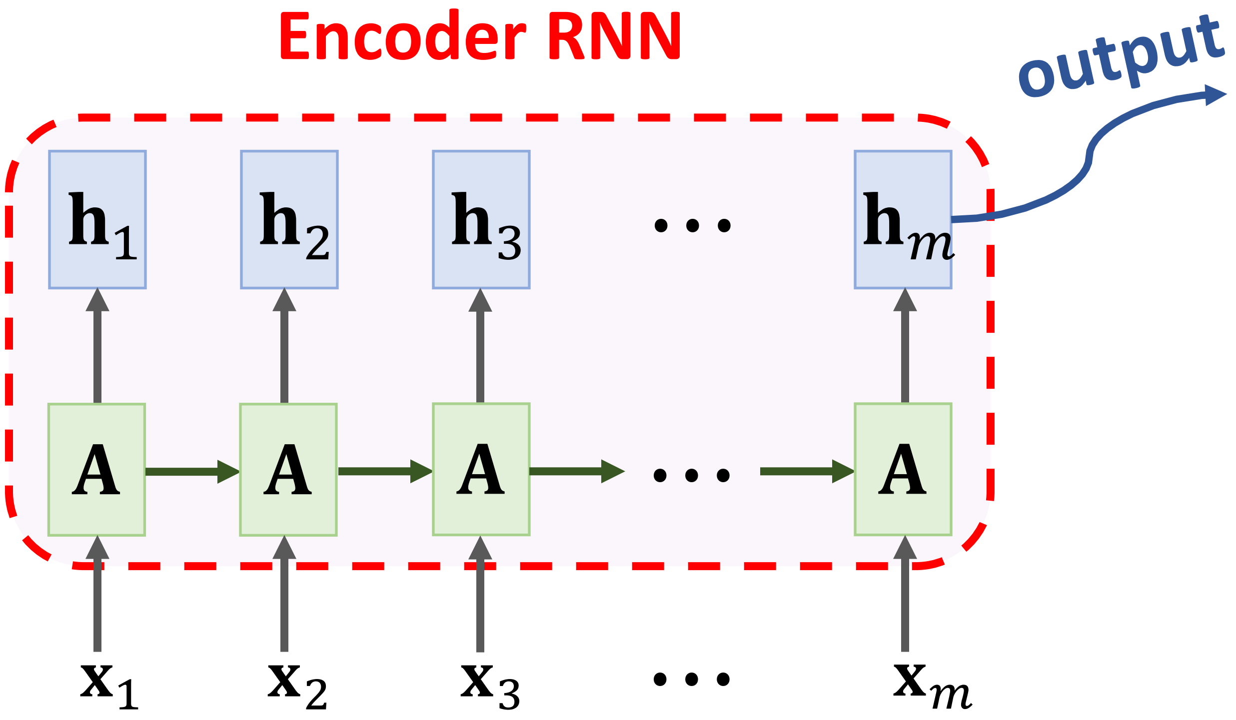
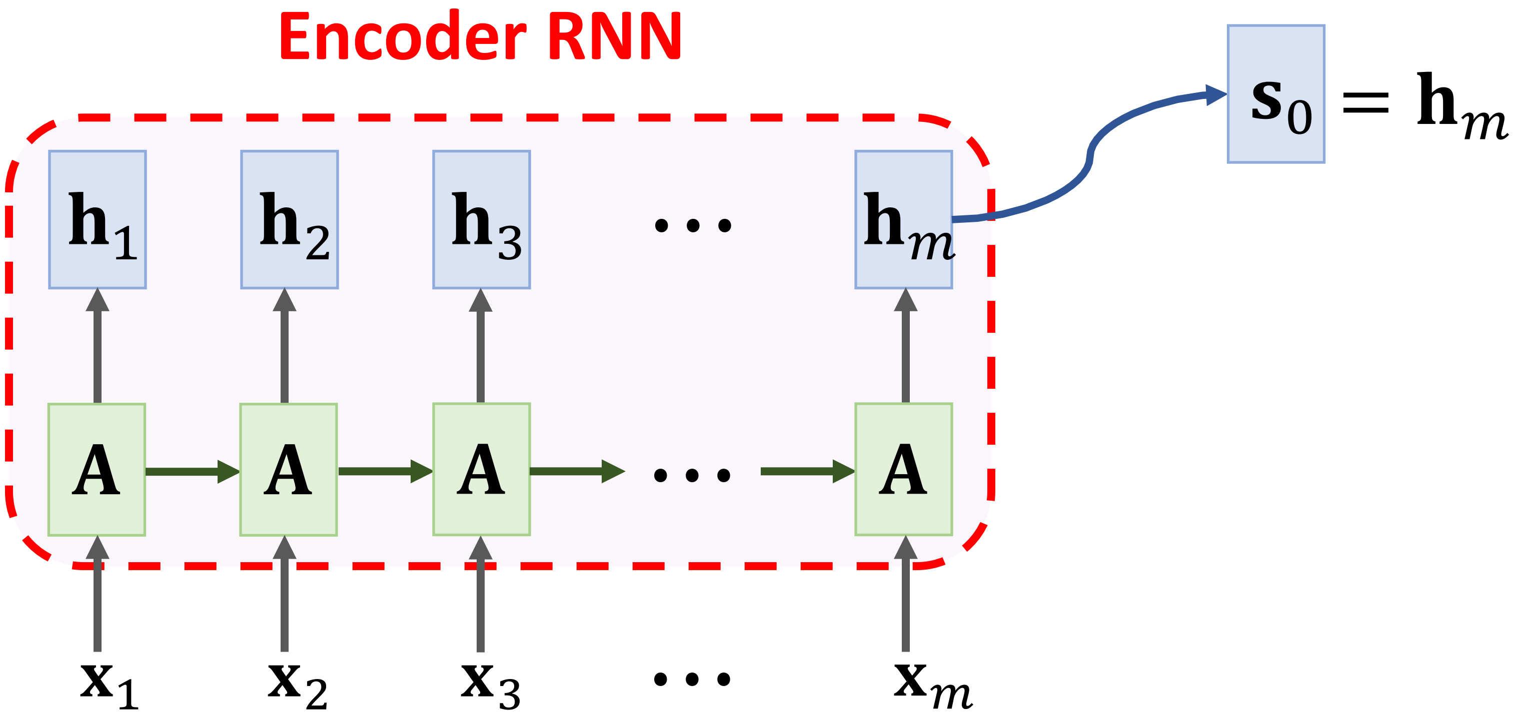
Attention

Shusen Wang

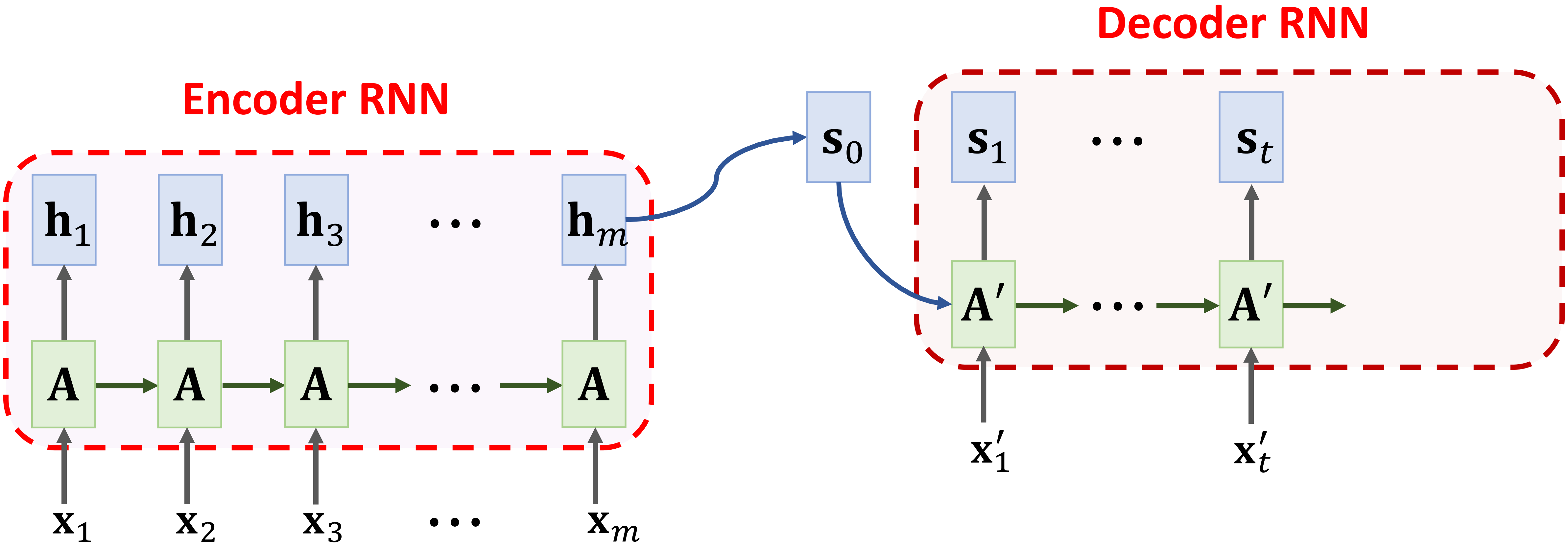
Seq2Seq Model



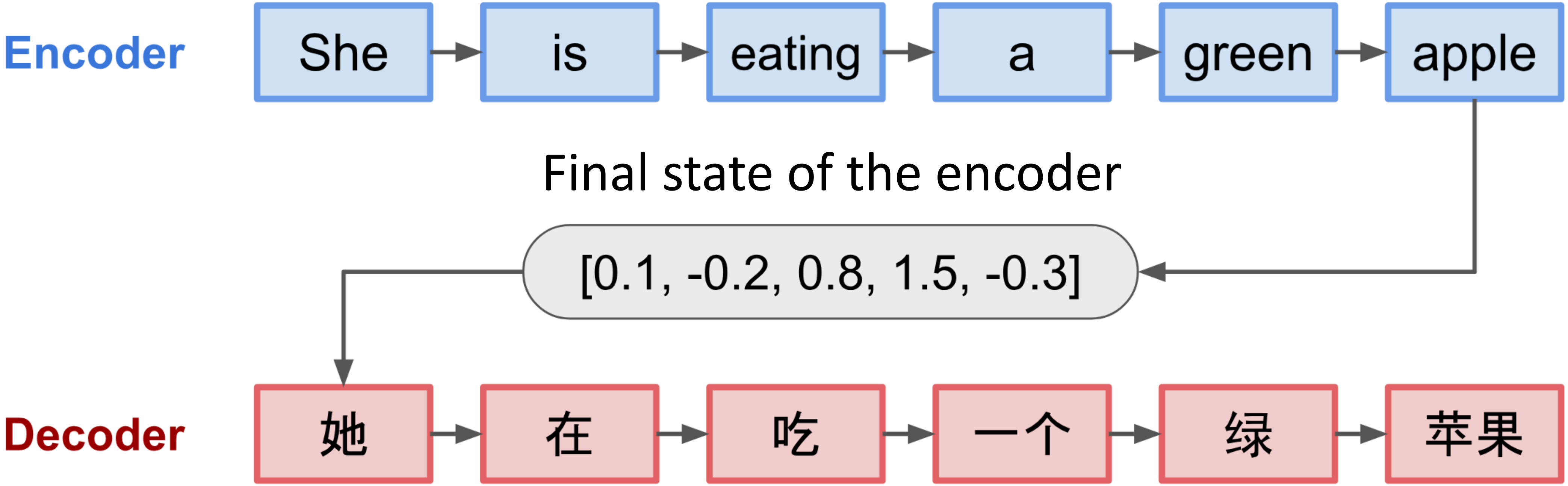
Seq2Seq Model



Seq2Seq Model



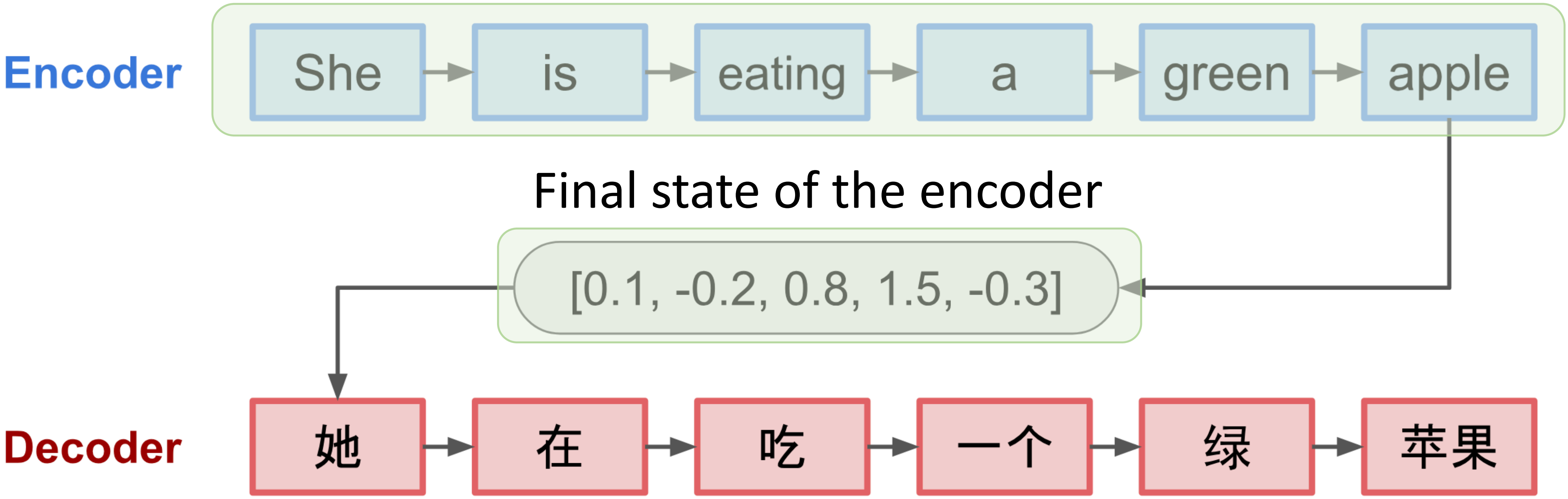
Seq2Seq Model



The figure is from blog lilianweng.github.io

Seq2Seq Model

Shortcoming: The final state is incapable of remembering a **long** sequence.



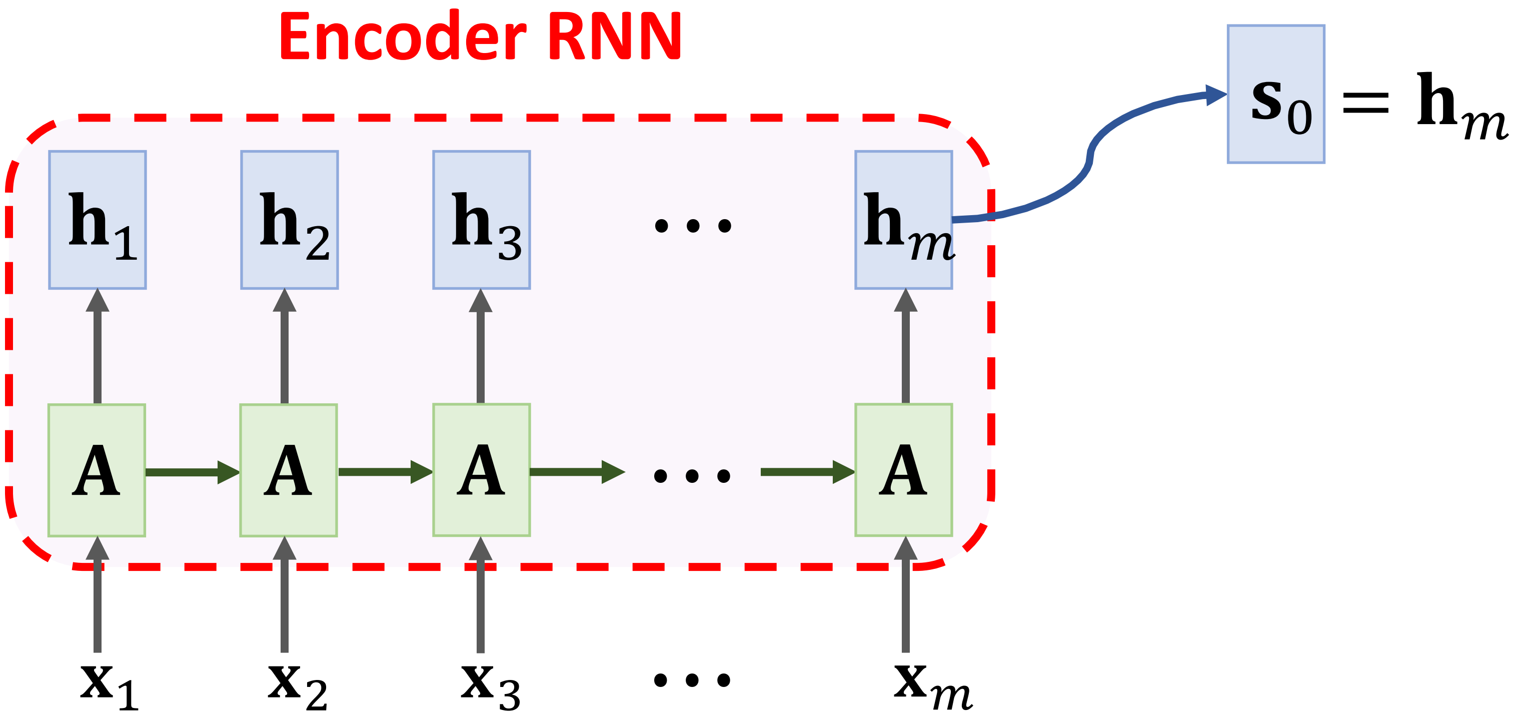
The figure is from blog lilianweng.github.io

Seq2Seq Model with Attention

Original paper:

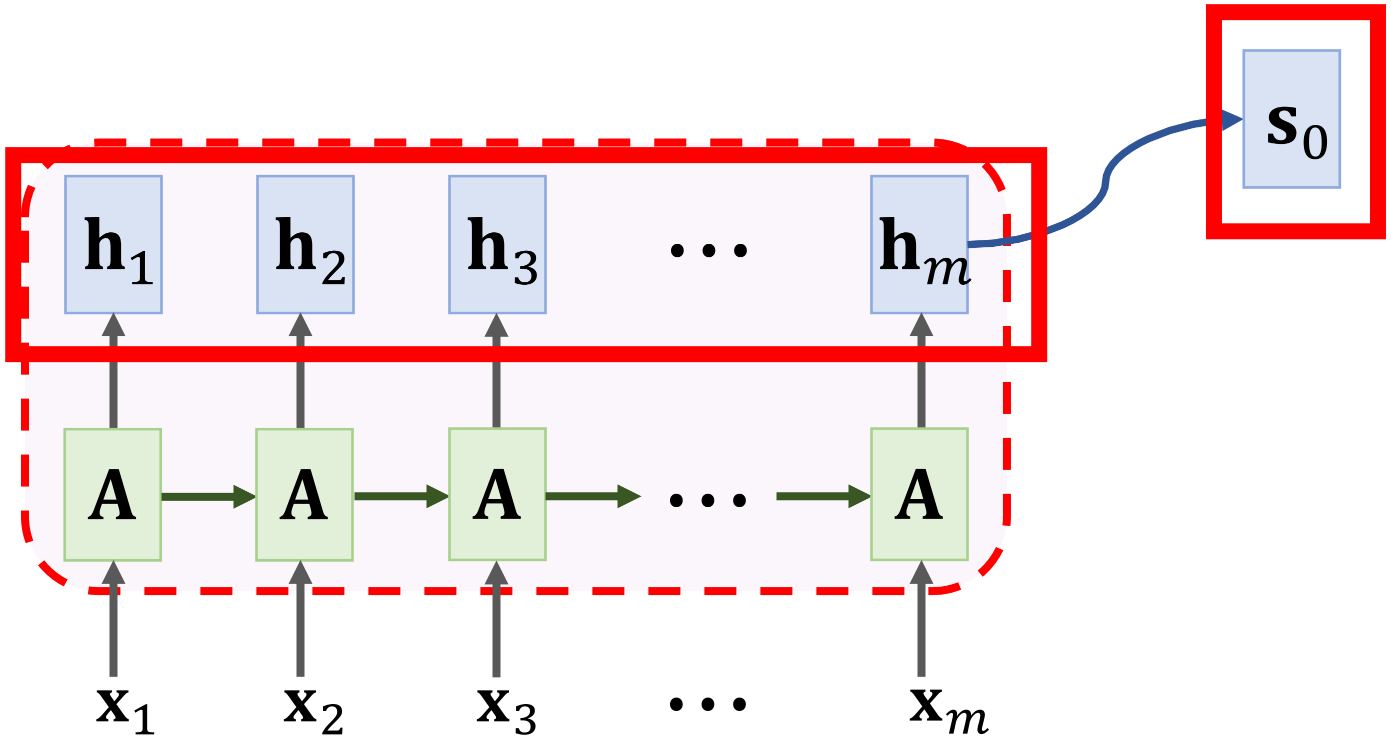
- Bahdanau, Cho, & Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

SimpleRNN + Attention



SimpleRNN + Attention

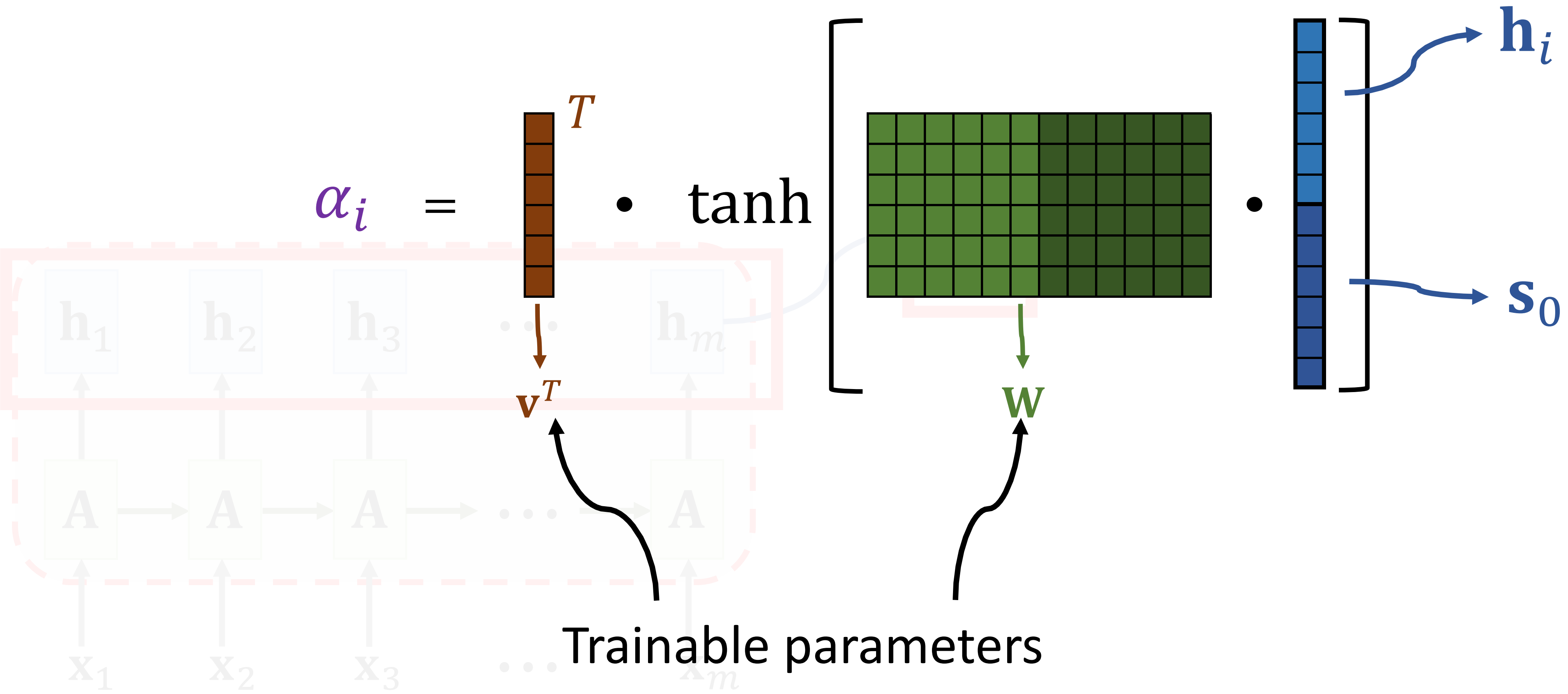
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_0)$



SimpleRNN + Attention

Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_0)$

One option (used in the original paper):



SimpleRNN + Attention

Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_0)$

One option (used in the original paper):

$$\alpha_i = \mathbf{v}^T \cdot \tanh\left[\mathbf{W} \cdot \begin{bmatrix} \mathbf{h}_i \\ \mathbf{s}_0 \end{bmatrix}\right]$$

Then **normalize** $\alpha_1, \cdots, \alpha_m$ (so that they sum to 1):

$$[\alpha_1, \cdots, \alpha_m] = \text{Softmax}\left([\alpha_1, \cdots, \alpha_m]\right)$$

SimpleRNN + Attention

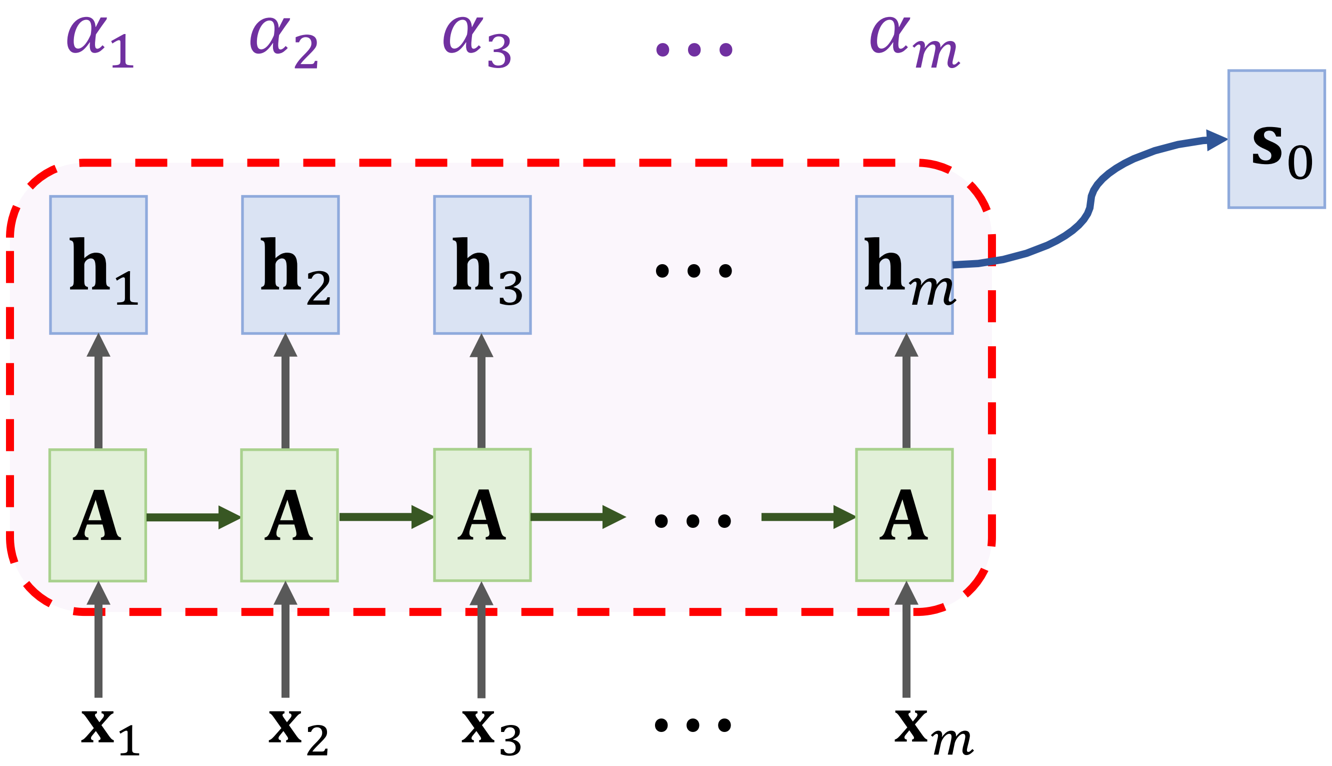
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_0)$

Another option (more popular; the same to Transformer):

1. Linear maps:
   - $\tilde{\mathbf{h}}_i = \mathbf{W}_h \cdot \mathbf{h}_i$.
   - $\tilde{\mathbf{s}}_0 = \mathbf{W}_s \cdot \mathbf{s}_0$.
2. Inner produce:
   - $\alpha_i = \tilde{\mathbf{h}}_i^T \cdot \tilde{\mathbf{s}}_0$.
3. Normalization:
   - $[\alpha_1, \cdots, \alpha_m] = \text{Softmax}\left([\alpha_1, \cdots, \alpha_m]\right)$

SimpleRNN + Attention

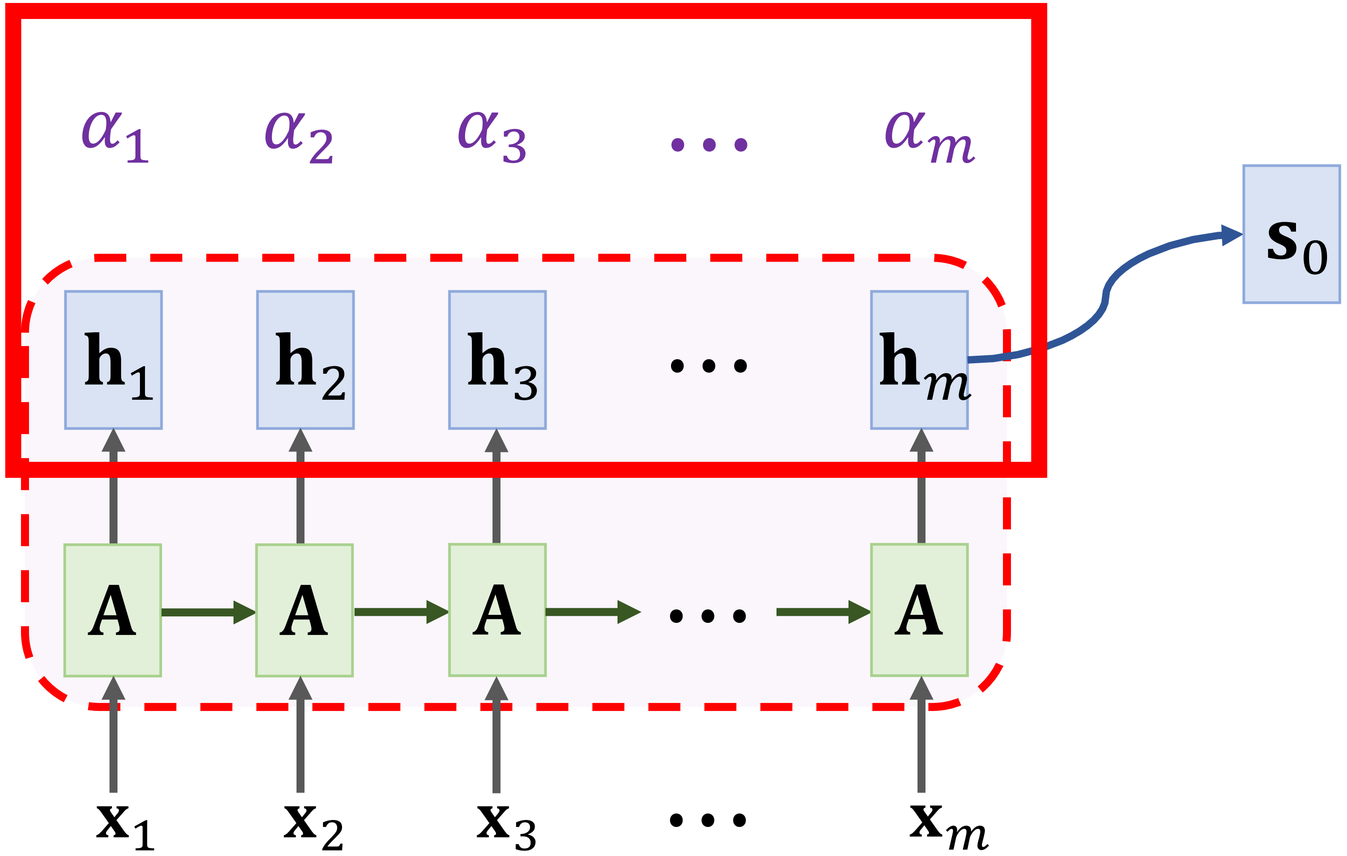
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_0)$



SimpleRNN + Attention

Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_0)$

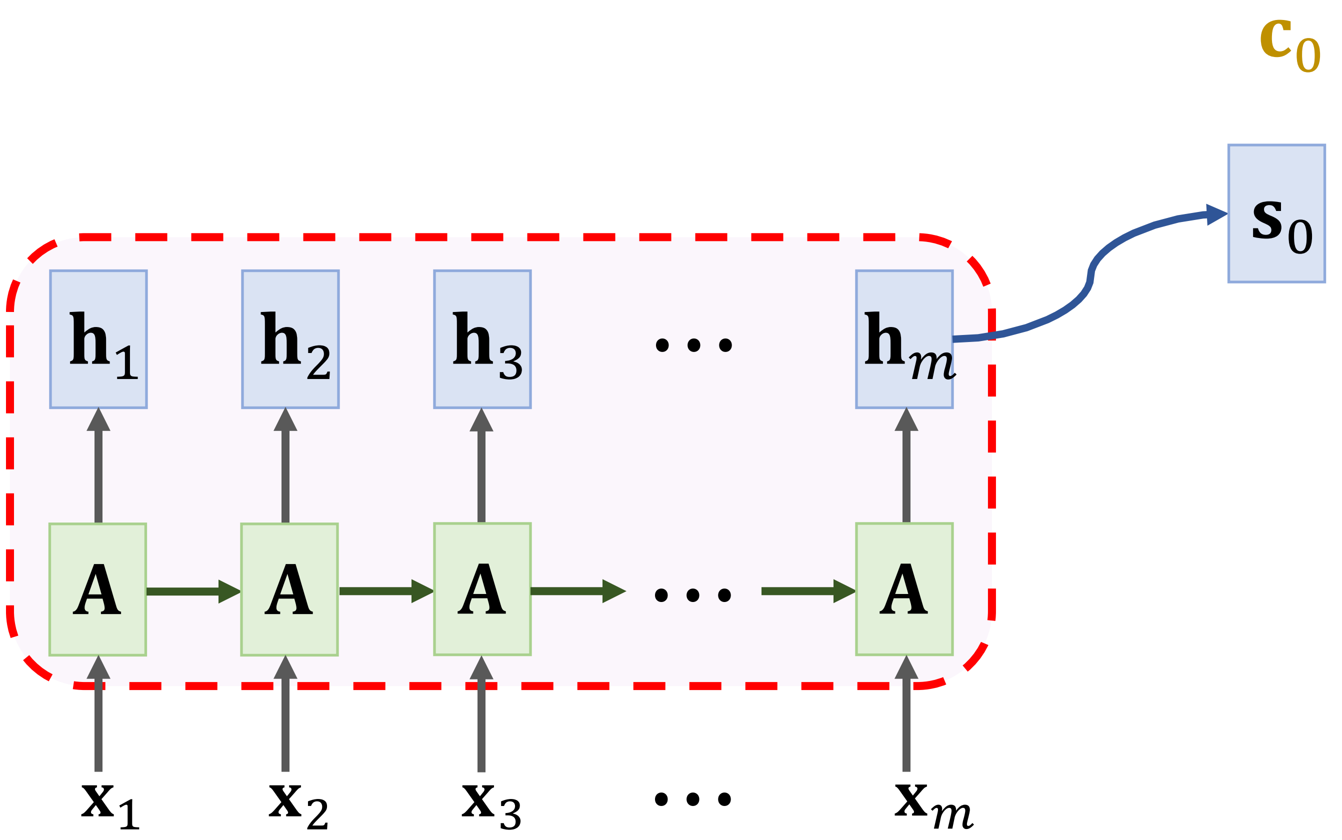
Context vector: $\mathbf{c}_0 = \alpha_1 \mathbf{h}_1 + \cdots + \alpha_m \mathbf{h}_m.$



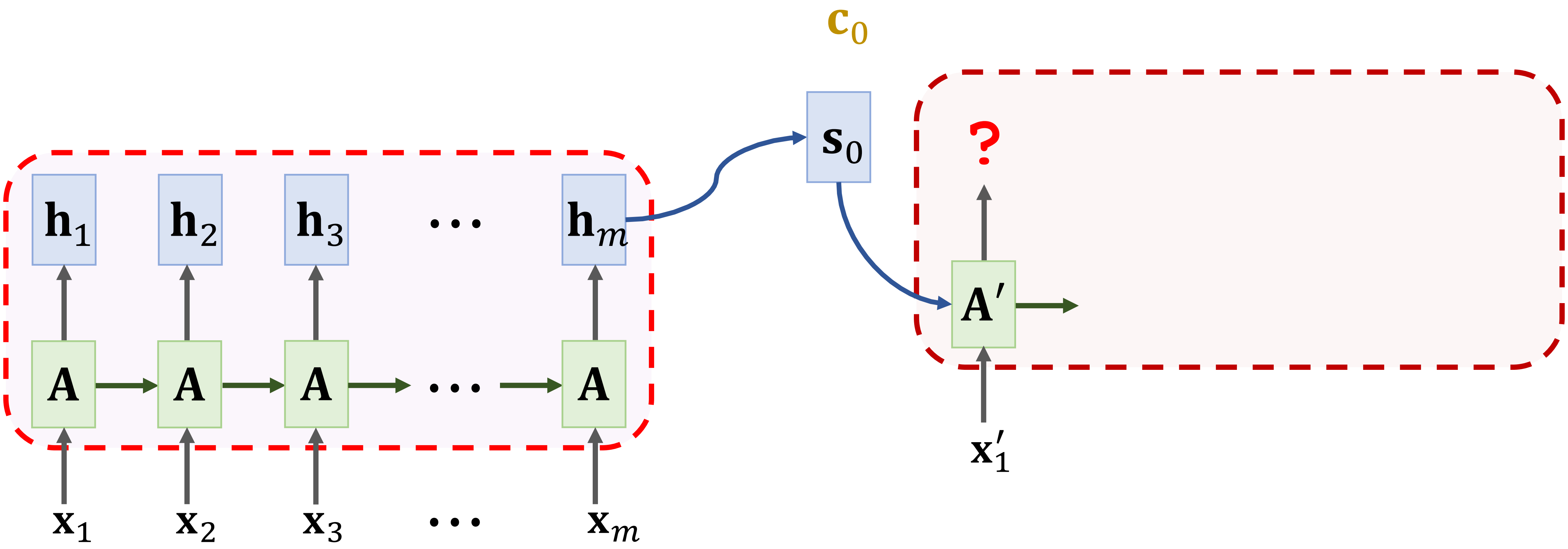
SimpleRNN + Attention

Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_0)$

Context vector: $\mathbf{c}_0 = \alpha_1 \mathbf{h}_1 + \cdots + \alpha_m \mathbf{h}_m$.



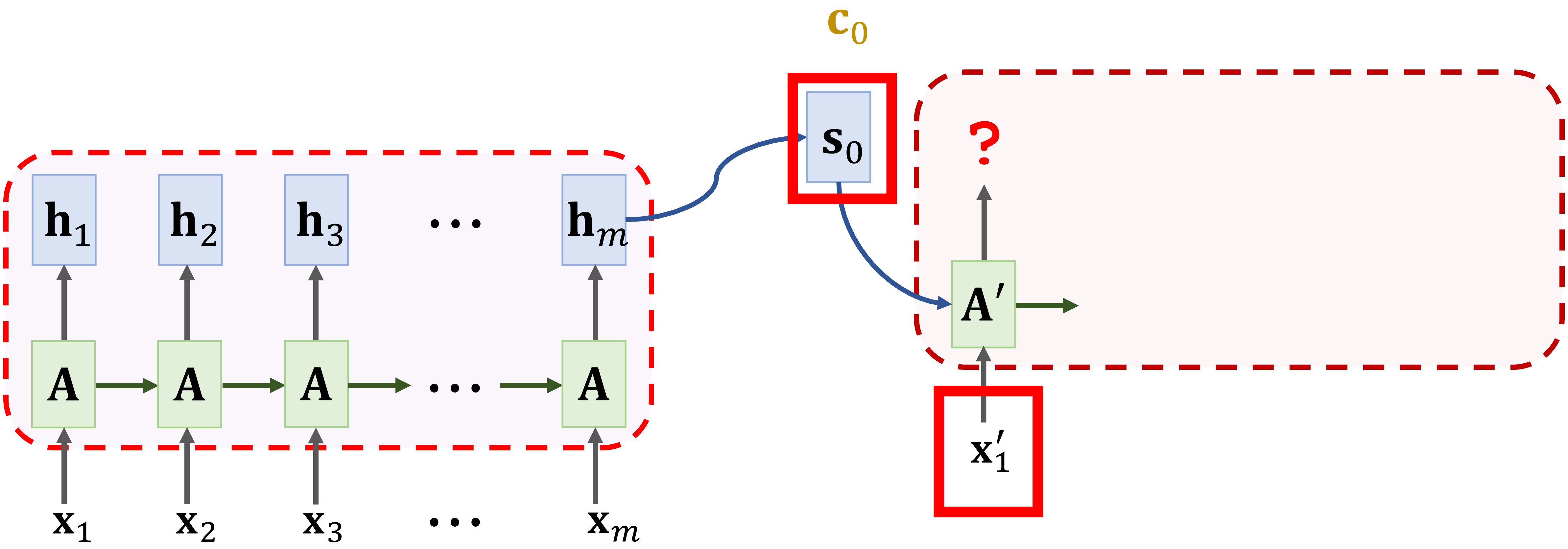
SimpleRNN + Attention



SimpleRNN + Attention

SimpleRNN:

$$\mathbf{s}_1 = \tanh\left(\mathbf{A}' \cdot \begin{bmatrix} \mathbf{x}'_1 \\ \mathbf{s}_0 \end{bmatrix} + \mathbf{b}\right)$$



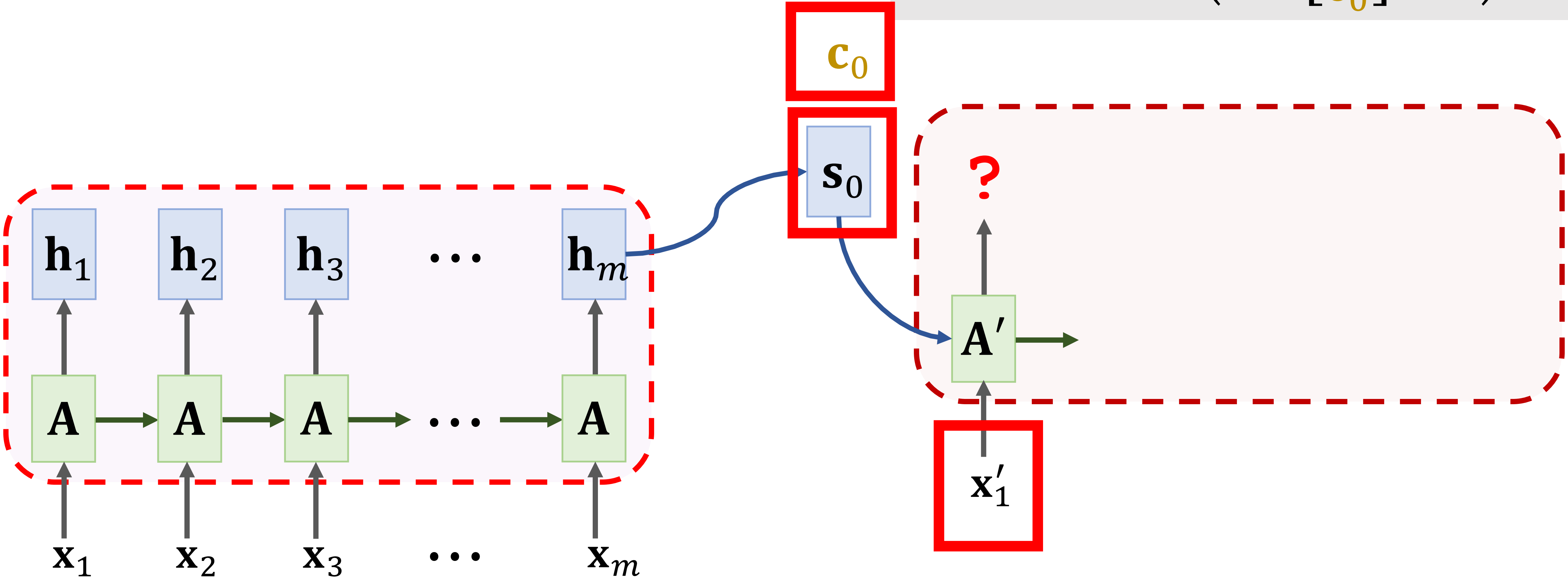
SimpleRNN + Attention

SimpleRNN:

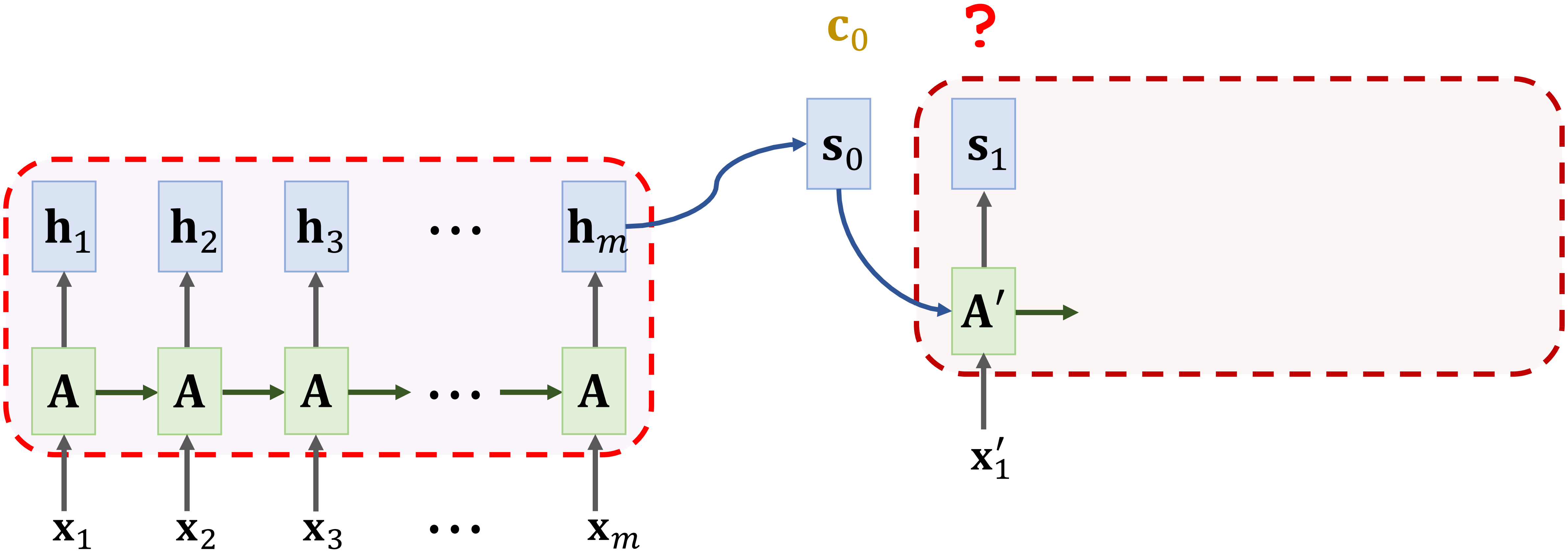
$$\mathbf{s}_1 = \tanh\left(\mathbf{A}' \cdot \begin{bmatrix} \mathbf{x}'_1 \\ \mathbf{s}_0 \end{bmatrix} + \mathbf{b}\right)$$

SimpleRNN + Attention:

$$\mathbf{s}_1 = \tanh\left(\mathbf{A}' \cdot \begin{bmatrix} \mathbf{x}'_1 \\ \mathbf{s}_0 \\ \mathbf{c}_0 \end{bmatrix} + \mathbf{b}\right)$$

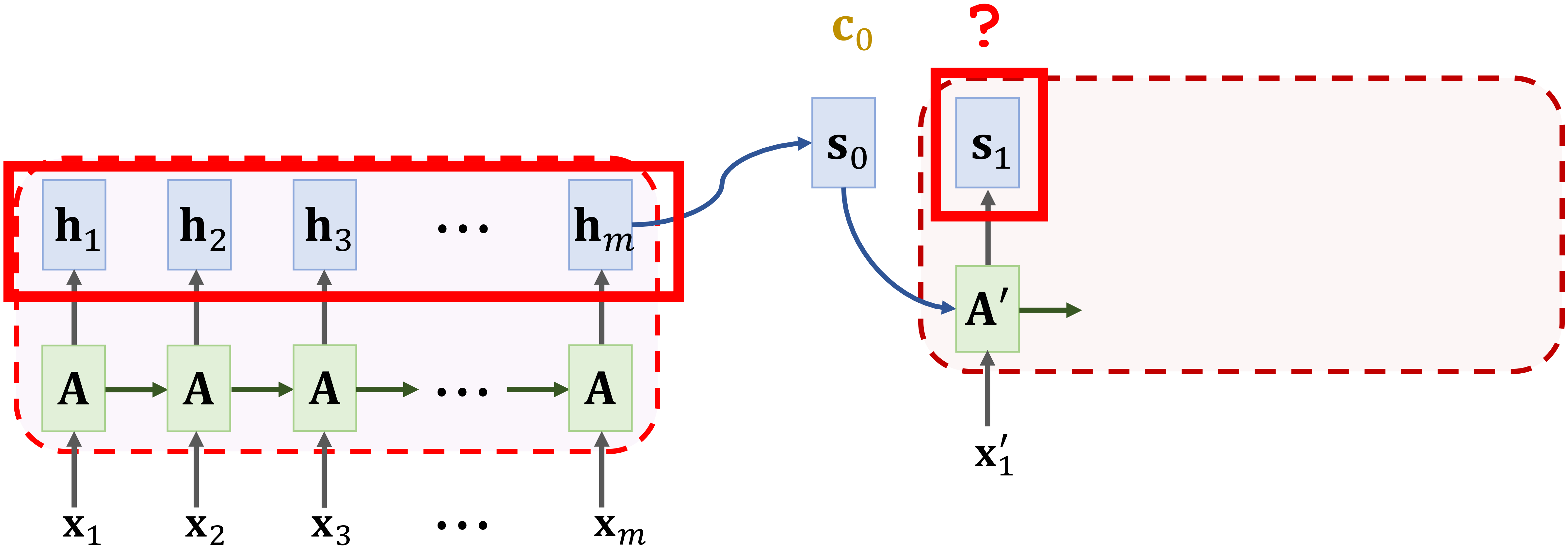


SimpleRNN + Attention



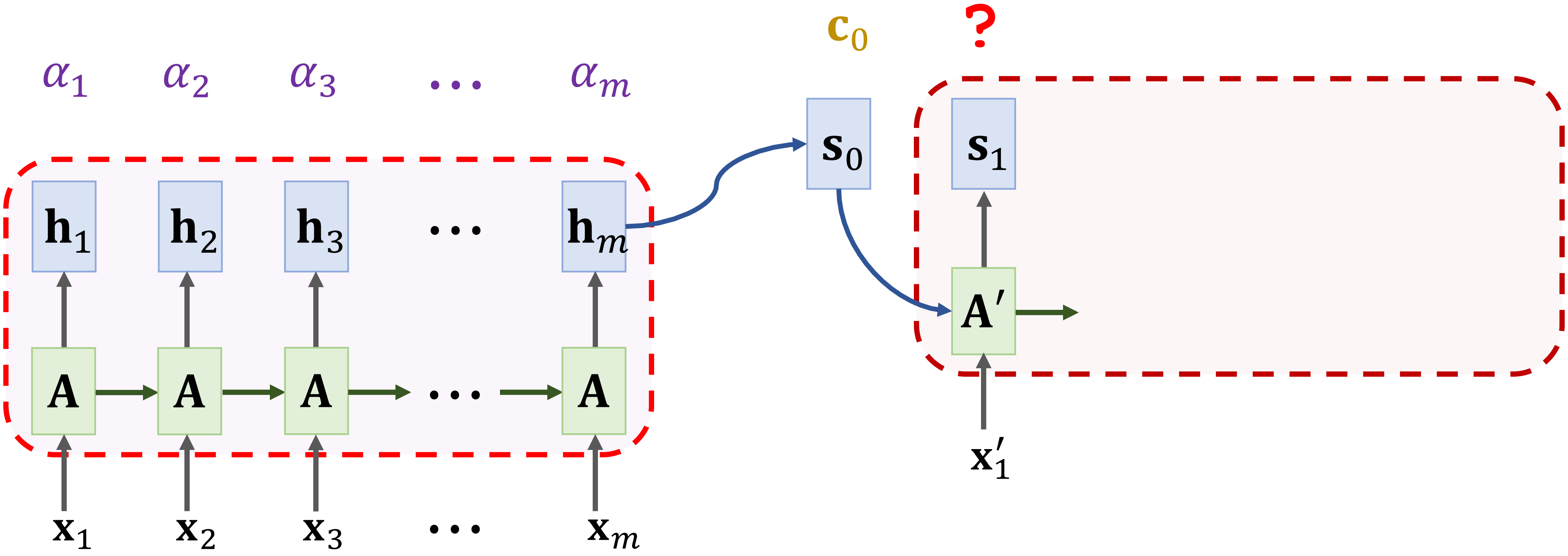
SimpleRNN + Attention

Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_1)$



SimpleRNN + Attention

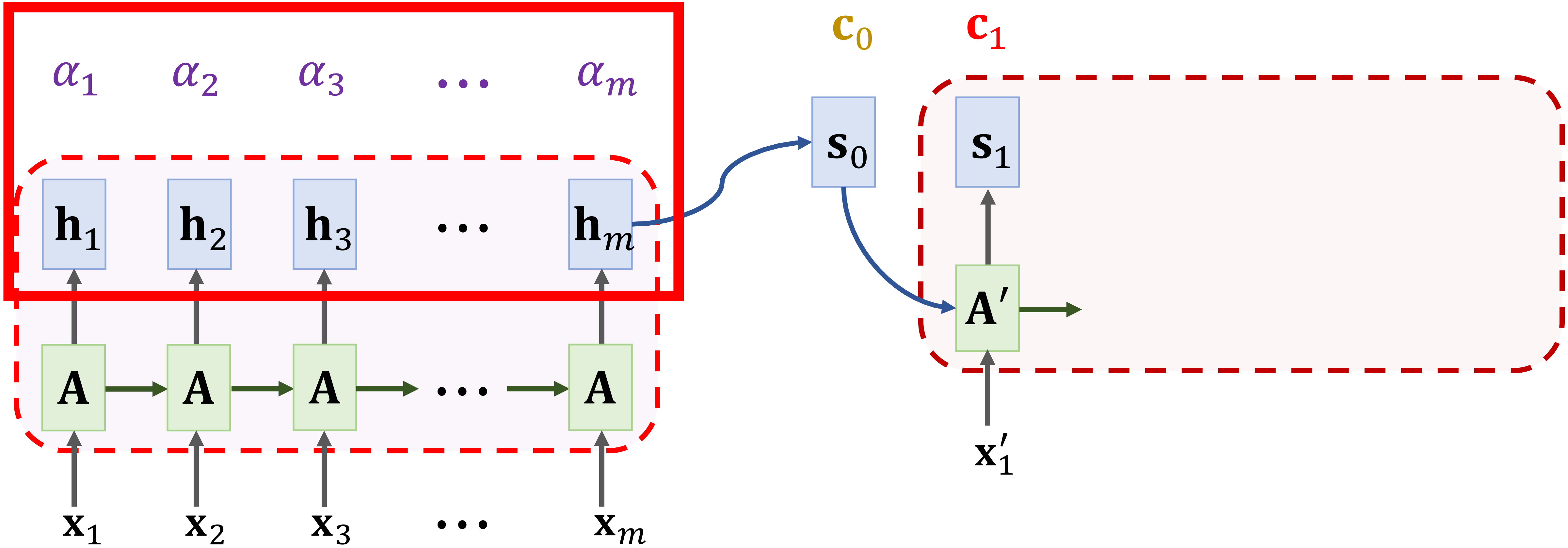
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_1)$



SimpleRNN + Attention

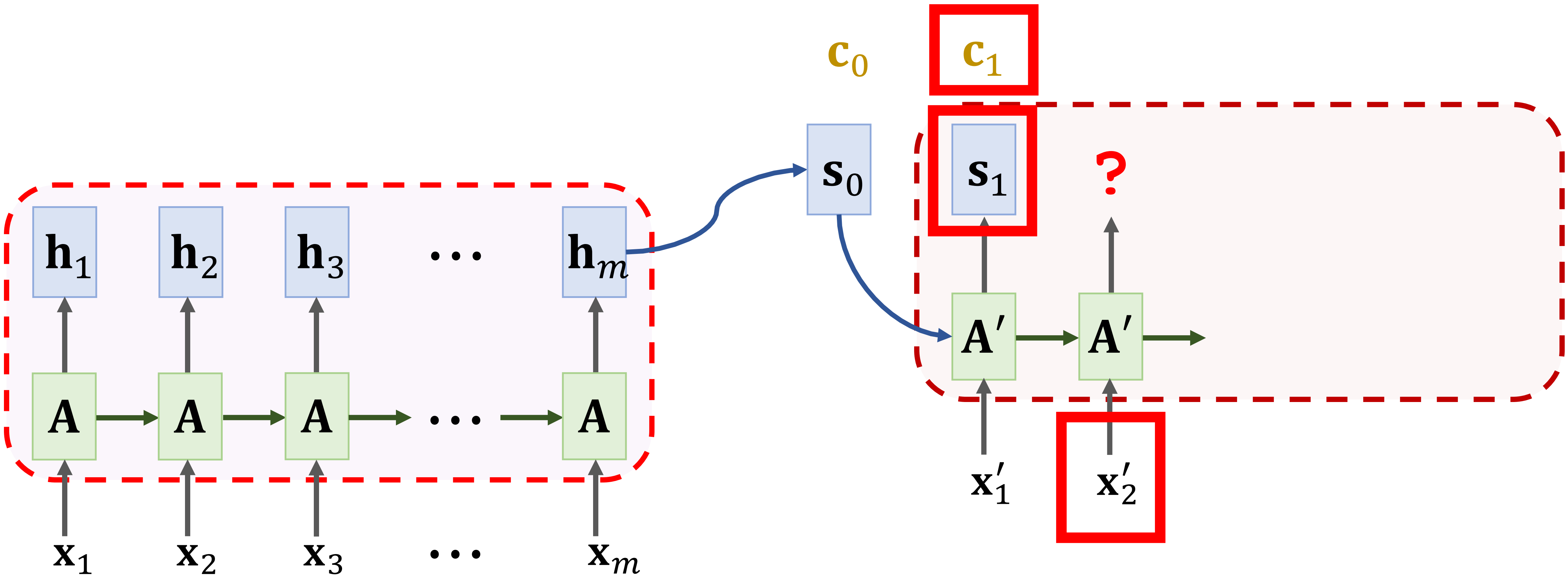
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_1)$

Context vector: $\mathbf{c}_1 = \alpha_1 \mathbf{h}_1 + \cdots + \alpha_m \mathbf{h}_m.$

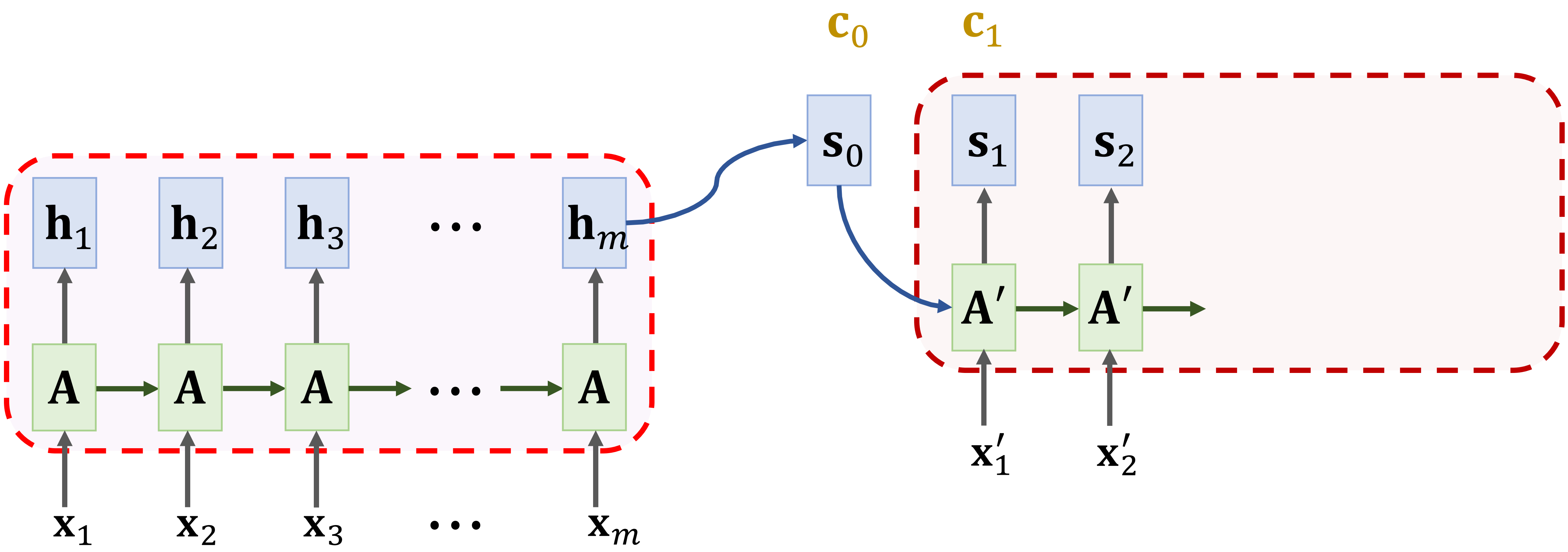


SimpleRNN + Attention

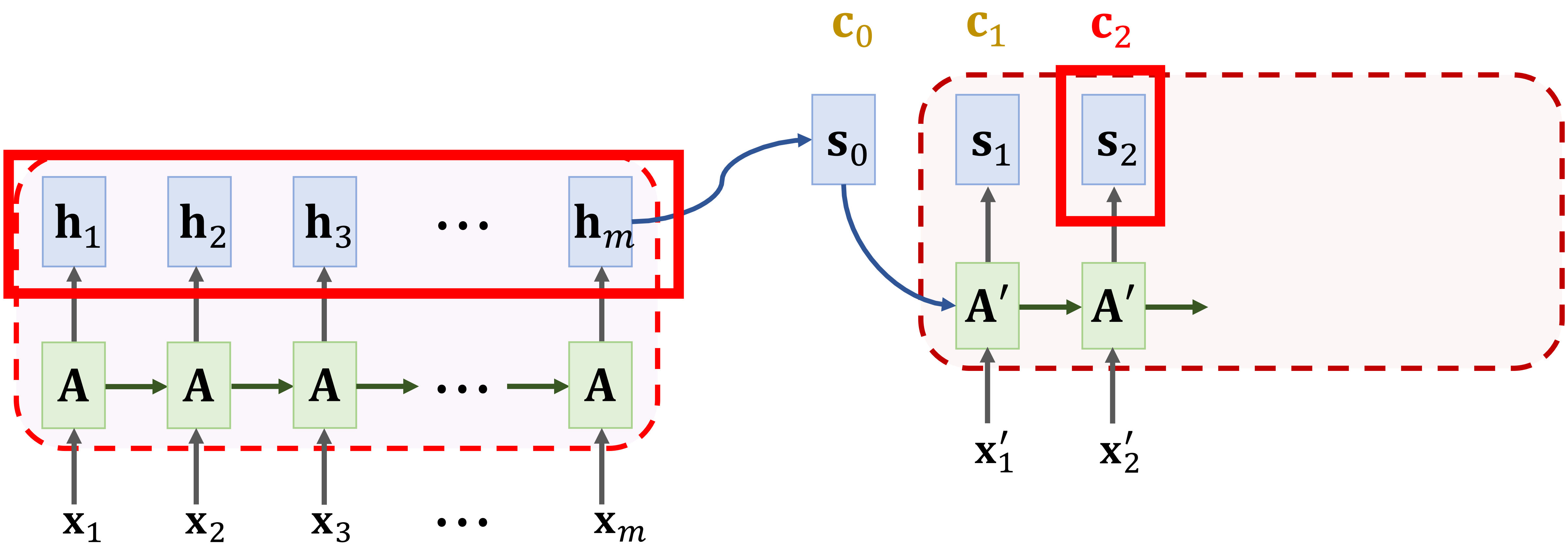
$$\mathbf{s}_2 = \tanh\left(\mathbf{A}' \cdot \begin{bmatrix} \mathbf{x}'_2 \\ \mathbf{s}_1 \\ \mathbf{c}_1 \end{bmatrix} + \mathbf{b} \right)$$



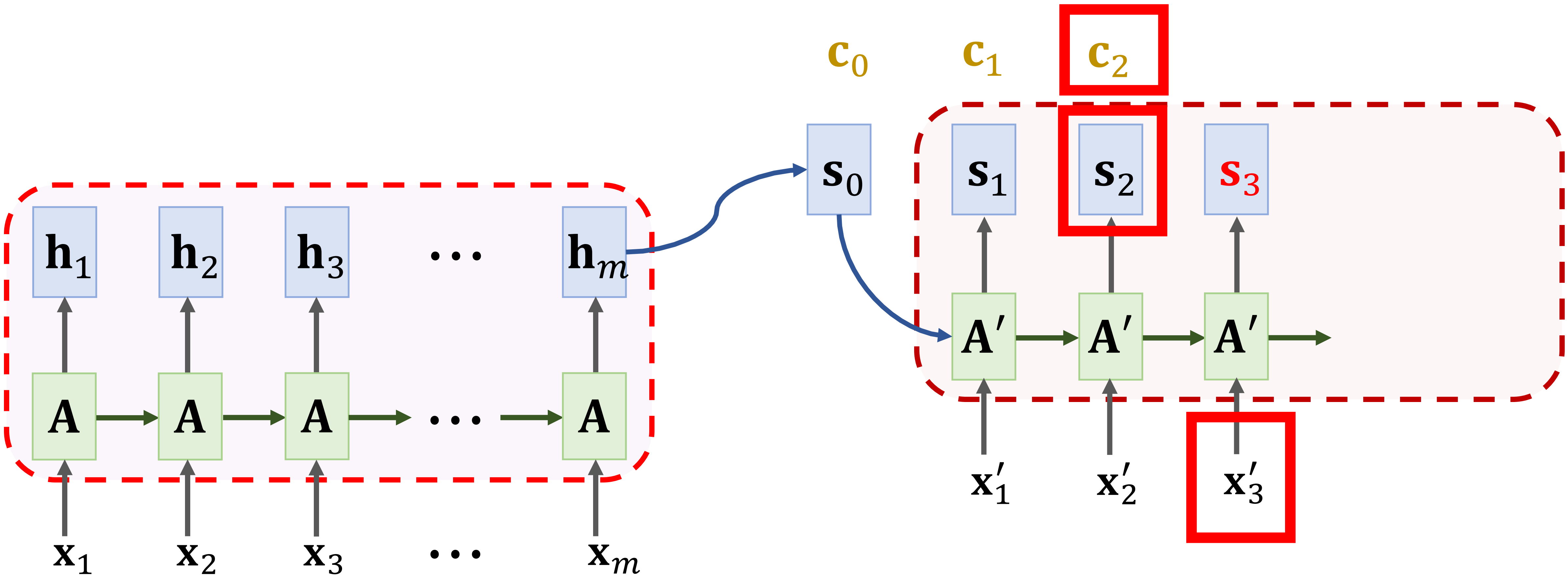
SimpleRNN + Attention



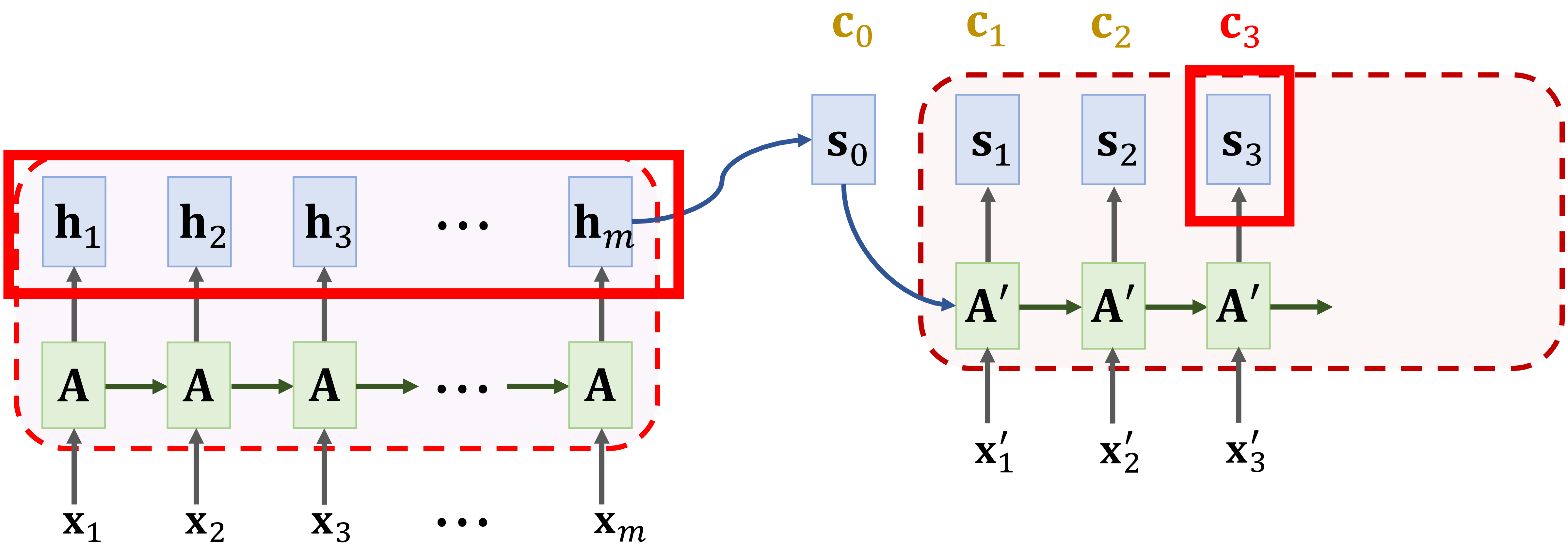
SimpleRNN + Attention



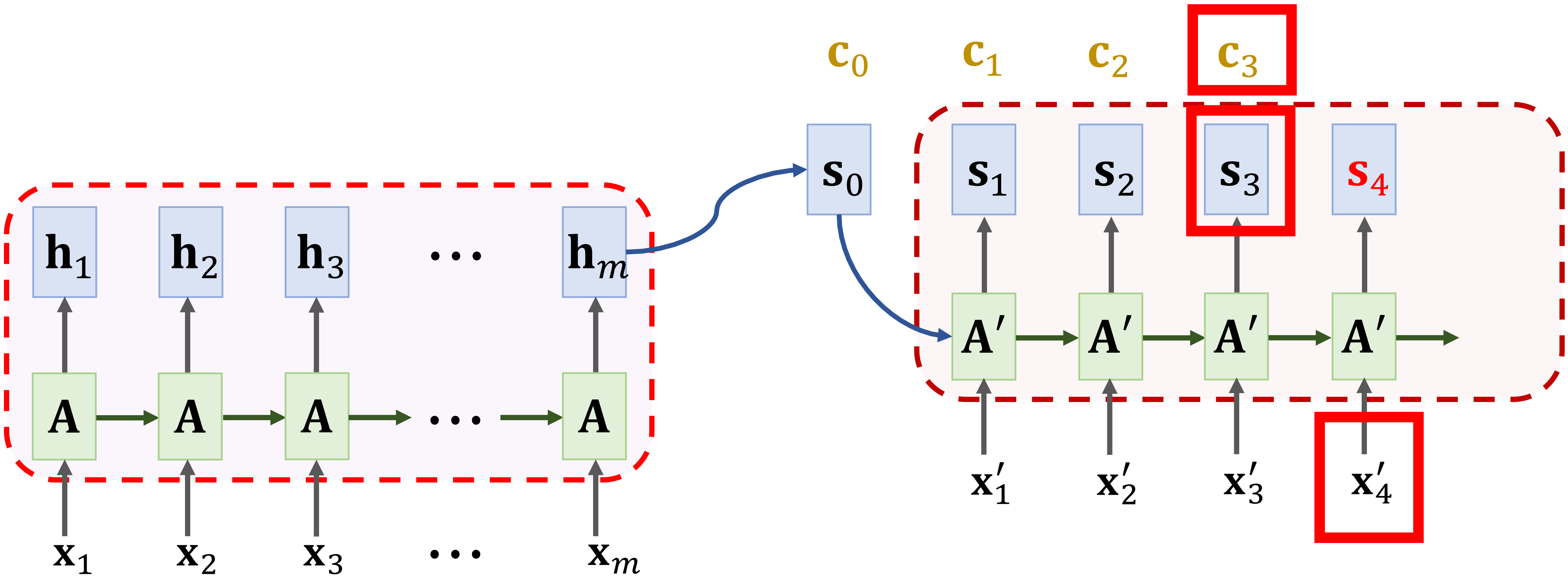
SimpleRNN + Attention



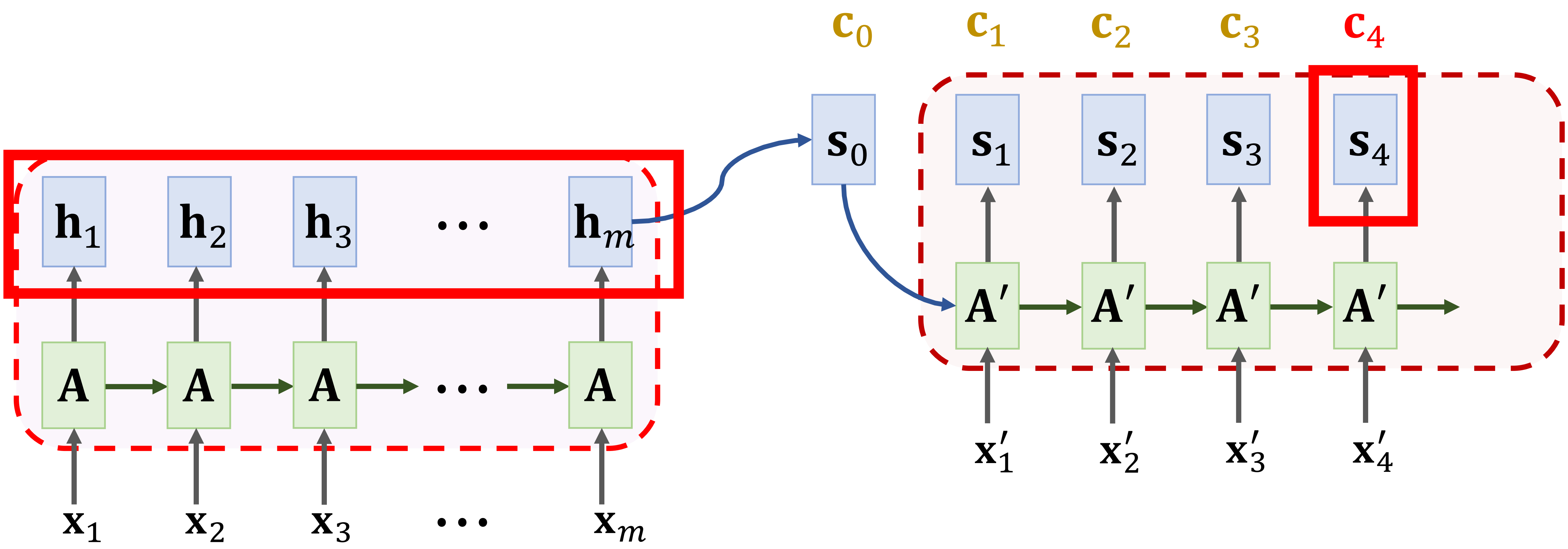
SimpleRNN + Attention



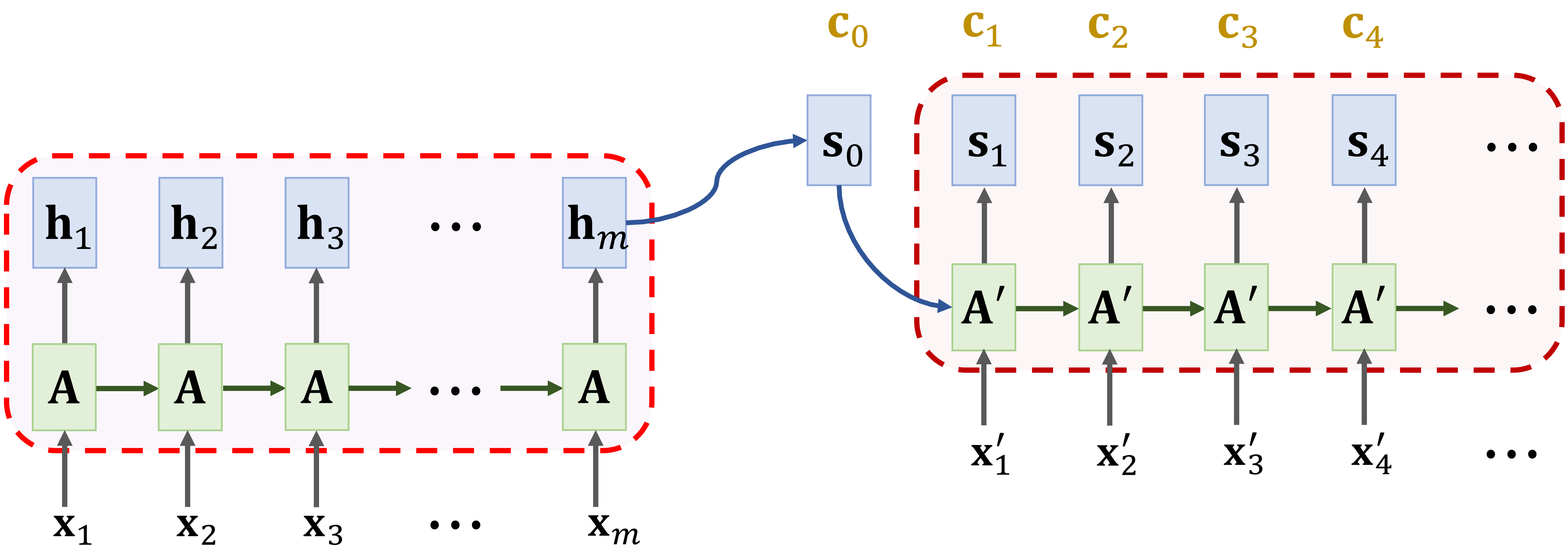
SimpleRNN + Attention



SimpleRNN + Attention

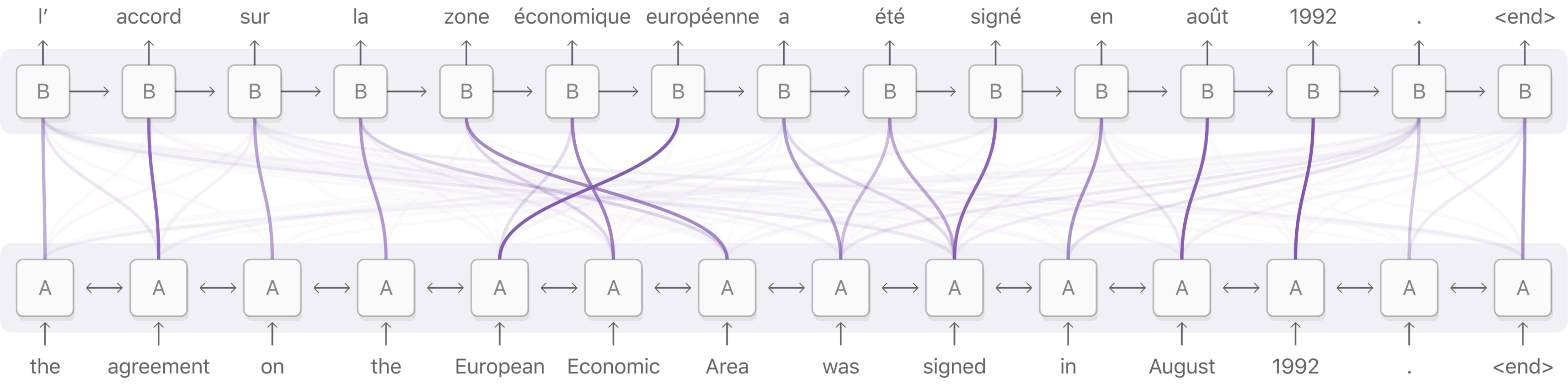


SimpleRNN + Attention



Attention: Weights Visualization

Decoder RNN



Encoder RNN

Figure is from https://distill.pub/2016/augmented-rnns/

Attention: Weights Visualization

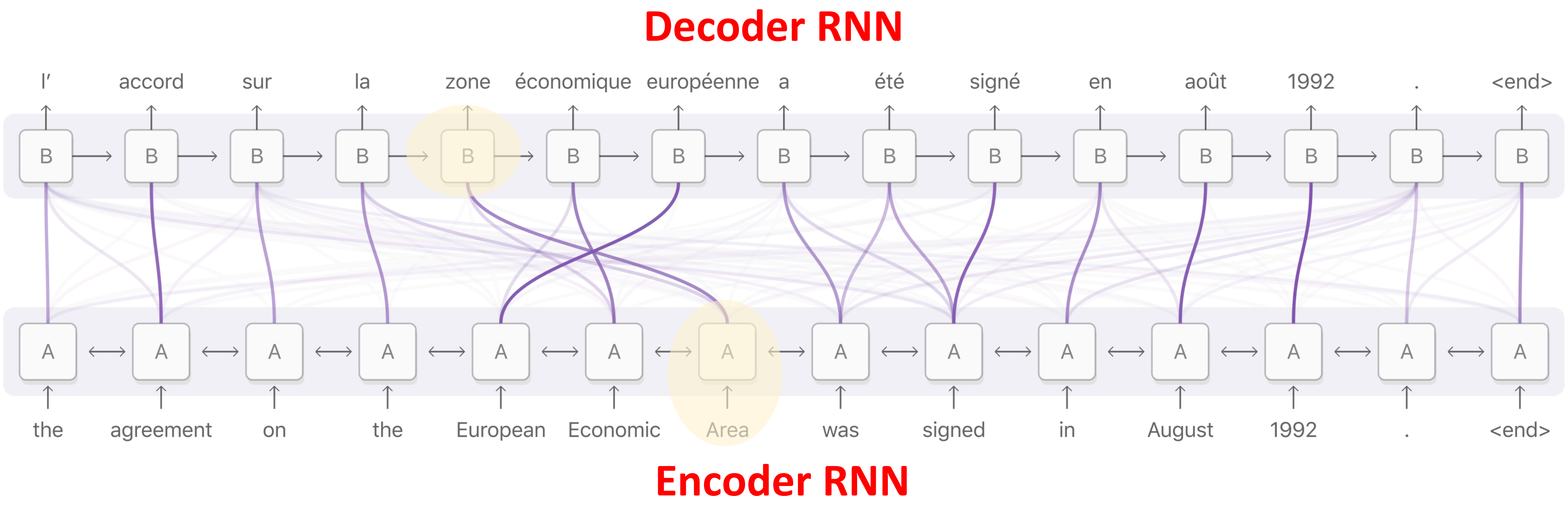
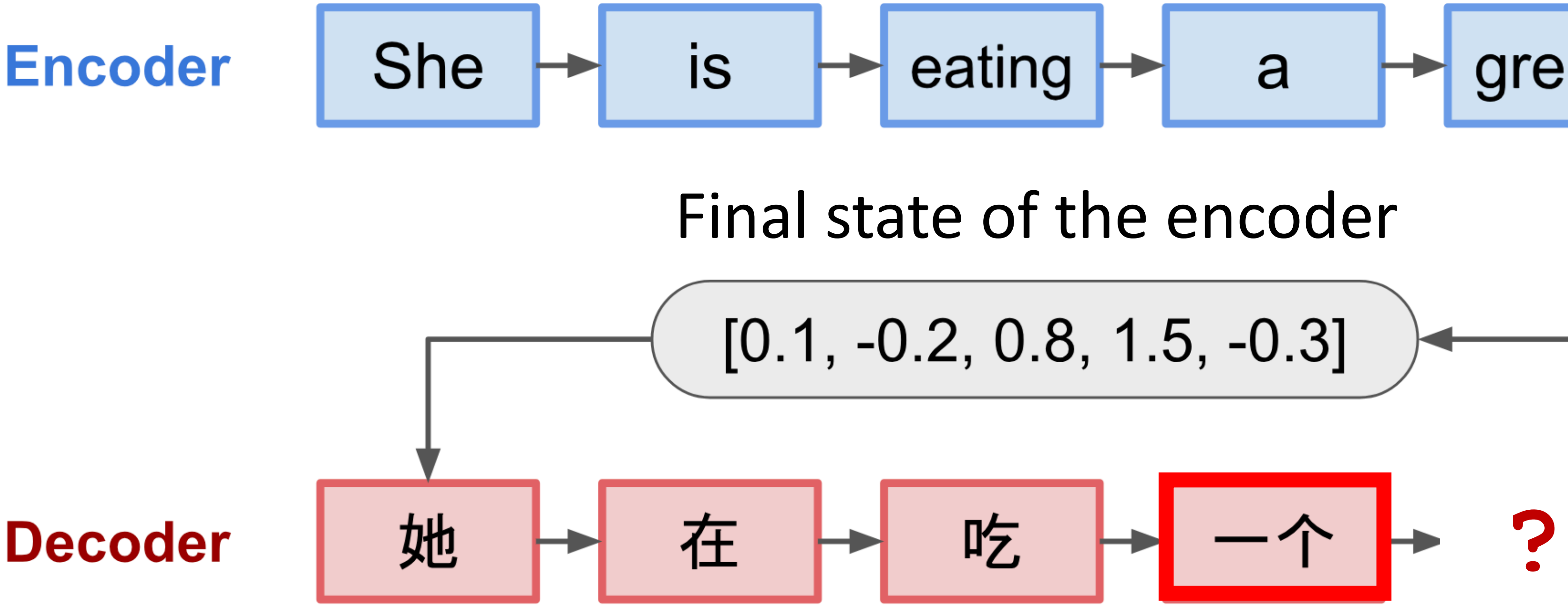


Figure is from https://distill.pub/2016/augmented-rnns/

Summary

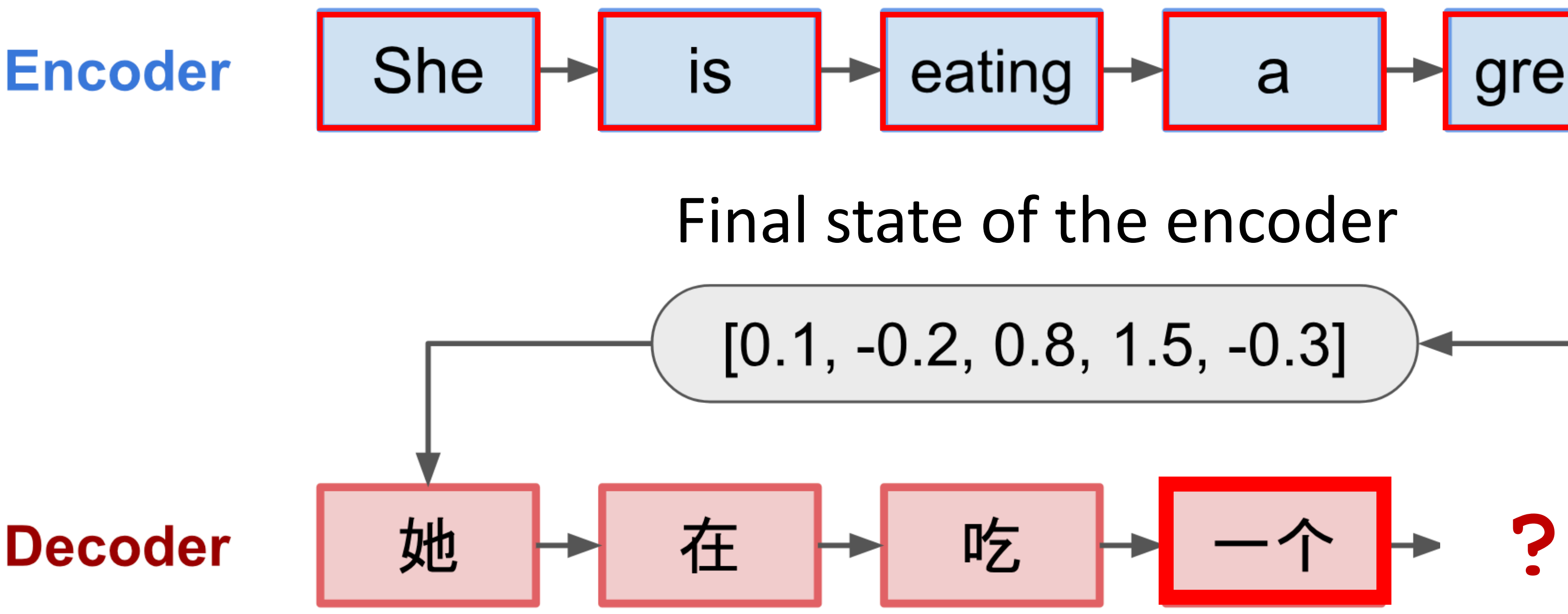
- Standard Seq2Seq model: the decoder looks at only its current state.



The figure is from blog lilianweng.github.io

Summary

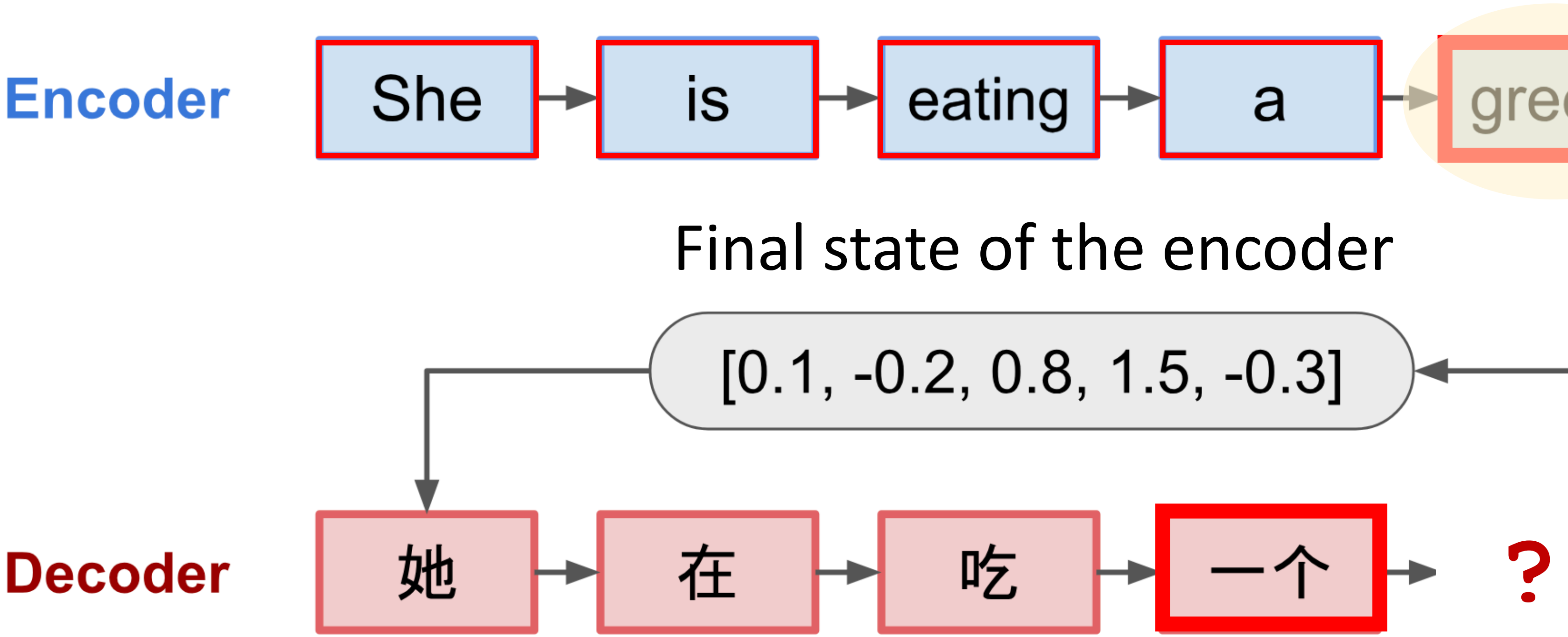
- Standard Seq2Seq model: the decoder looks at only its current state.
- Attention: decoder additionally looks at all the states of the encoder.



The figure is from blog lilianweng.github.io

Summary

- Standard Seq2Seq model: the decoder looks at only its current state.
- Attention: decoder additionally looks at all the states of the encoder.



The figure is from blog lilianweng.github.io

Summary

- Standard Seq2Seq model: the decoder looks at only its current state.
- Attention: decoder additionally looks at all the states of the encoder.

- Downside: higher time complexity.
  - l_1: source sequence length
  - l_2: target sequence length
  - Standard Seq2Seq: $O(l_1 + l_2)$ time complexity
  - Seq2Seq + attention: $O(l_1 \, l_2)$ time complexity

Self-Attention: Attention beyond Seq2Seq Models

Original paper:

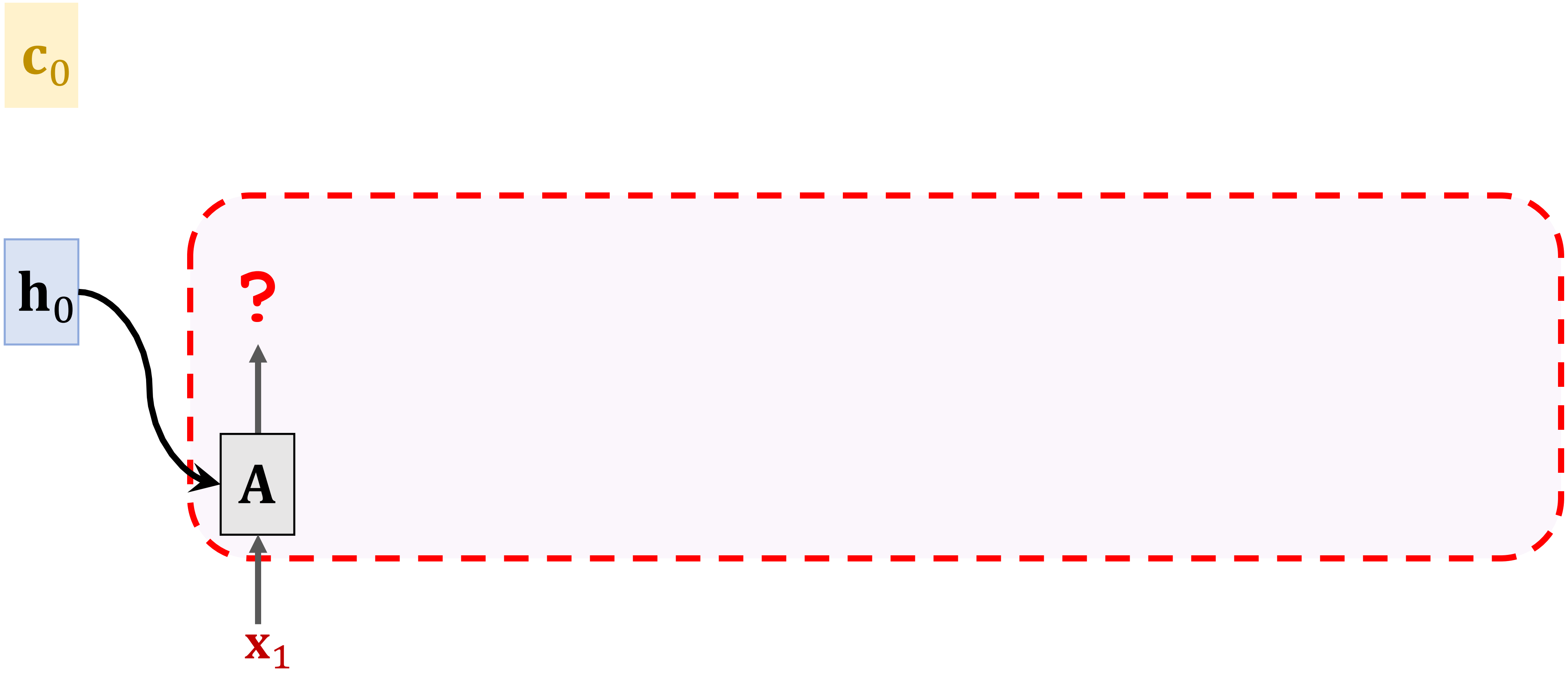
- Cheng, Dong, & Lapata. Long Short-Term Memory-Networks for Machine Reading. In *EMNLP*, 2016.

SimpleRNN + Self-Attention

$\mathbf{c}_0 = \mathbf{0}$

$\mathbf{h}_0 = \mathbf{0}$

SimpleRNN + Self-Attention



SimpleRNN + Self-Attention

SimpleRNN:

$$\mathbf{h}_1 = \tanh\left(\mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{h}_0 \end{bmatrix} + \mathbf{b}\right)$$

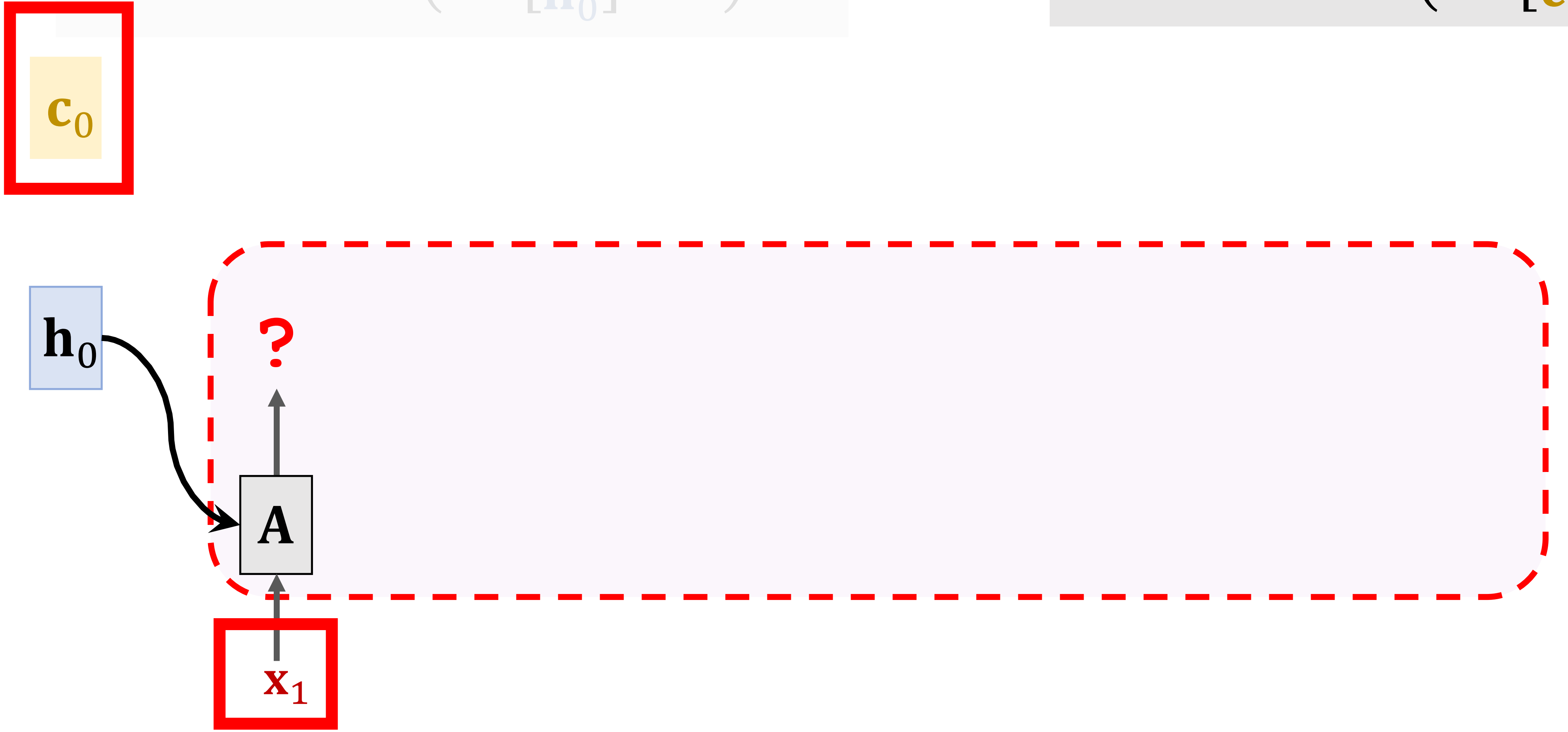
SimpleRNN + Self-Attention

SimpleRNN:

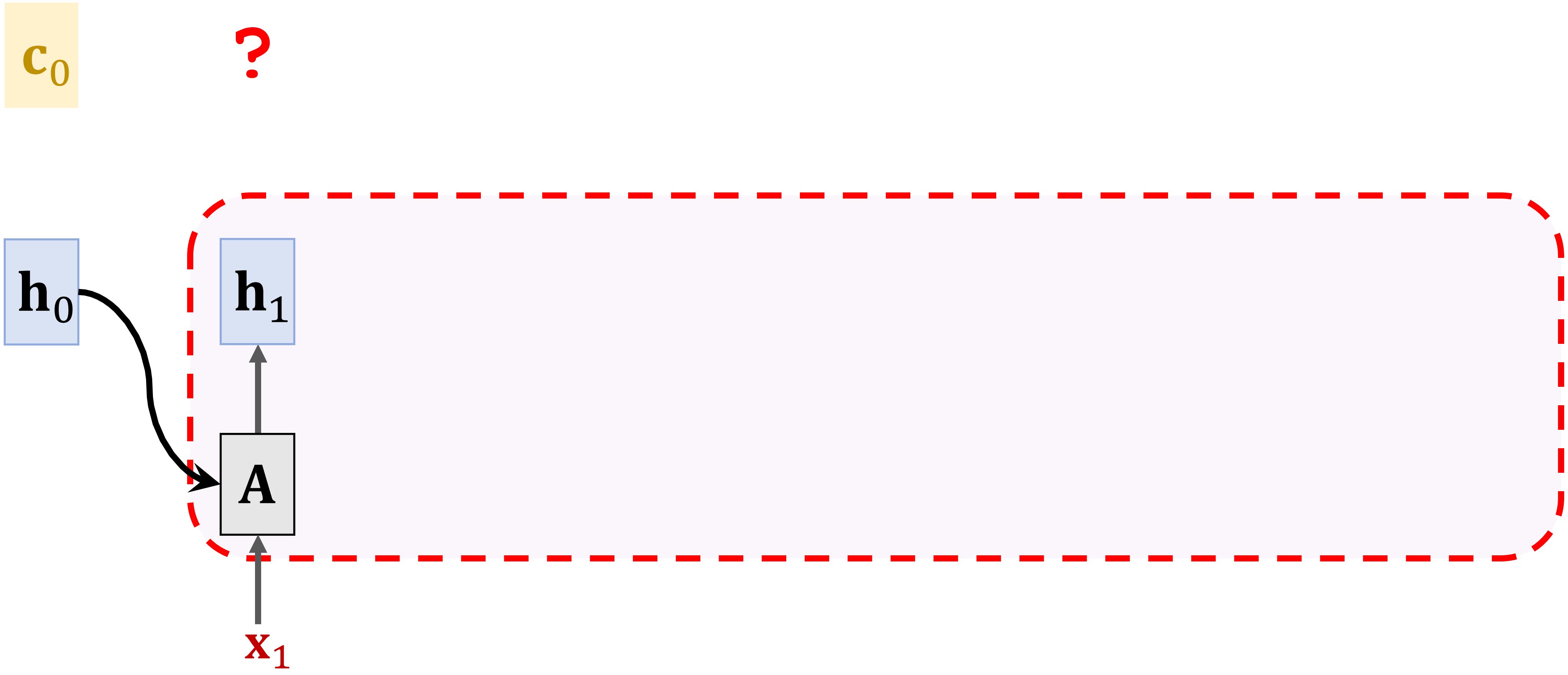
$$\mathbf{h}_1 = \tanh\left(\mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{h}_0 \end{bmatrix} + \mathbf{b}\right)$$

SimpleRNN + Self-Attention:

$$\mathbf{h}_1 = \tanh\left(\mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{c}_0 \end{bmatrix} + \mathbf{b}\right)$$

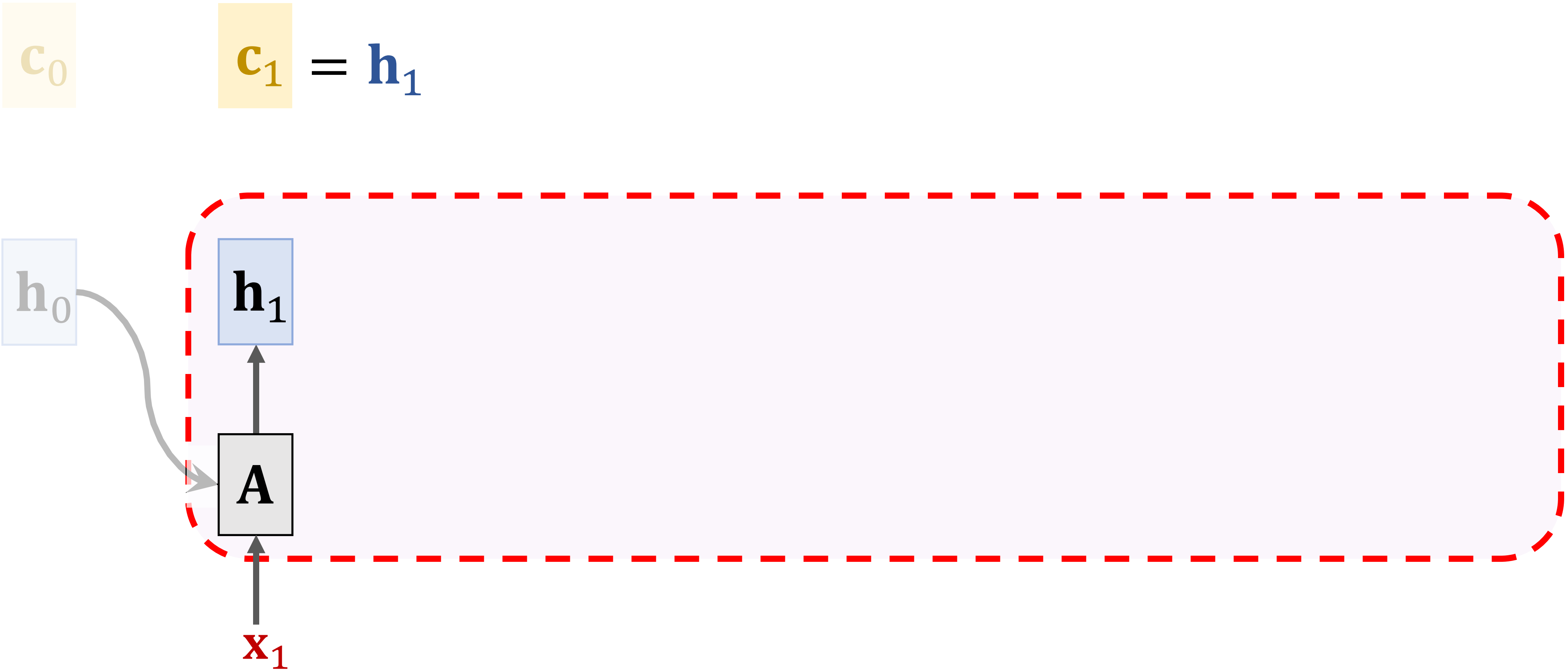


SimpleRNN + Self-Attention

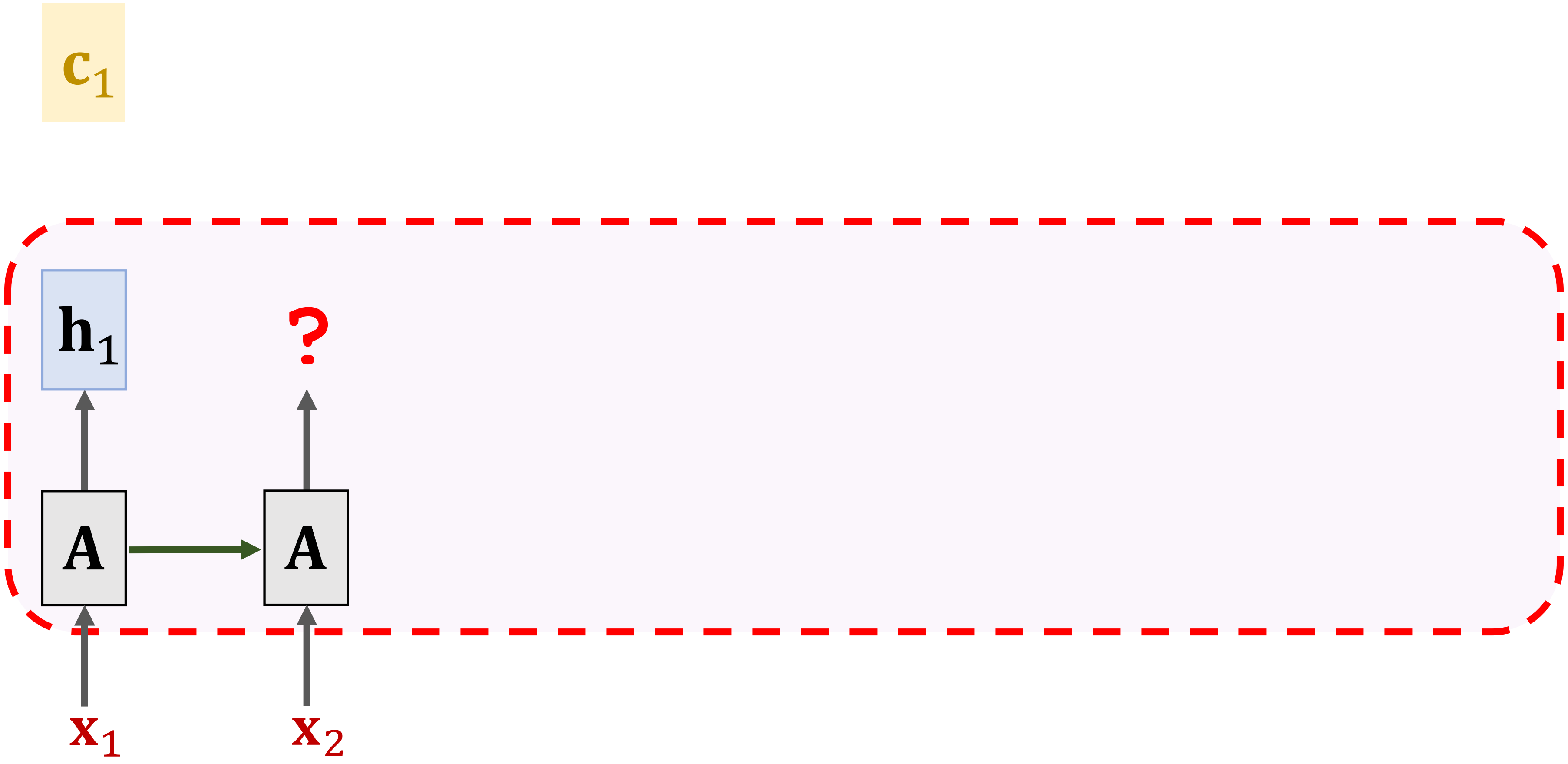


SimpleRNN + Self-Attention

First context vector: $\mathbf{c}_1 = \mathbf{h}_1.$

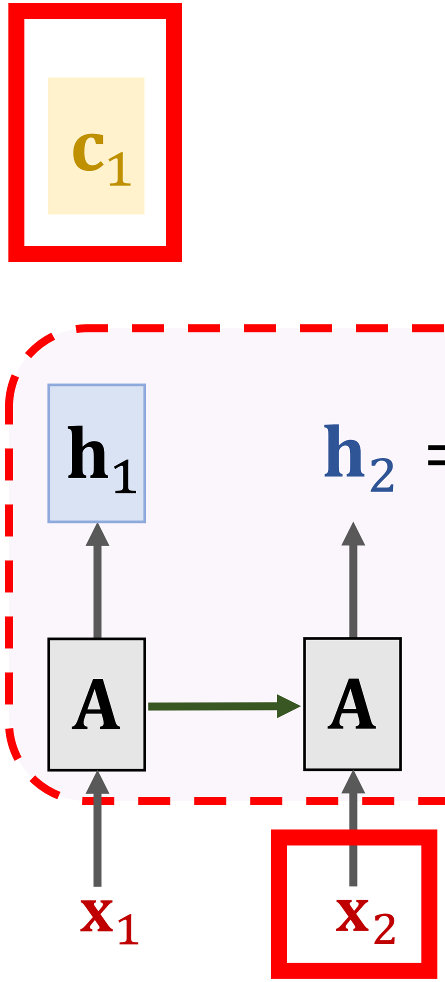


SimpleRNN + Self-Attention



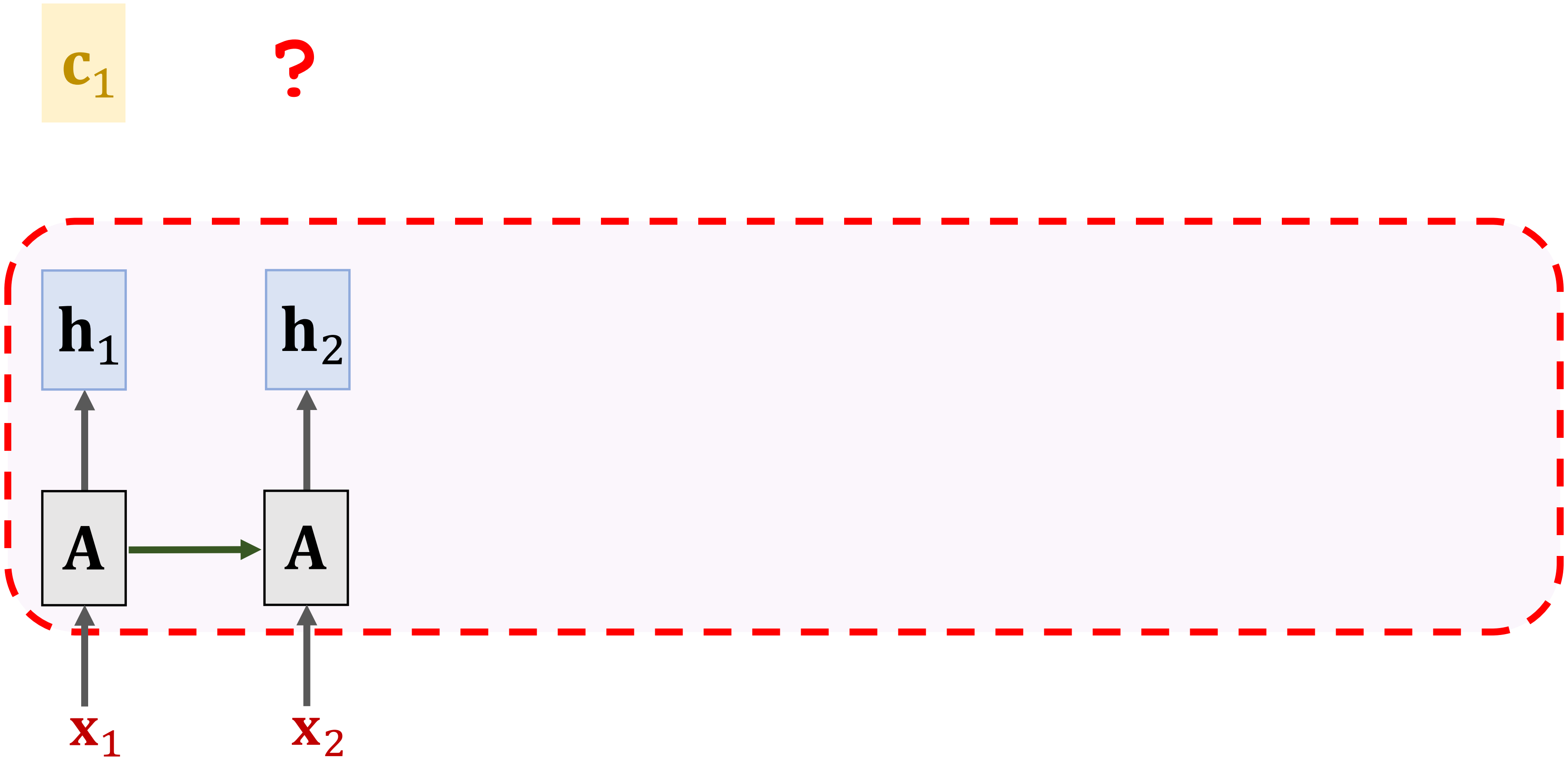
SimpleRNN + Self-Attention

$$\mathbf{h}_2 = \tanh\left(\mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{c}_1 \end{bmatrix} + \mathbf{b}\right)$$



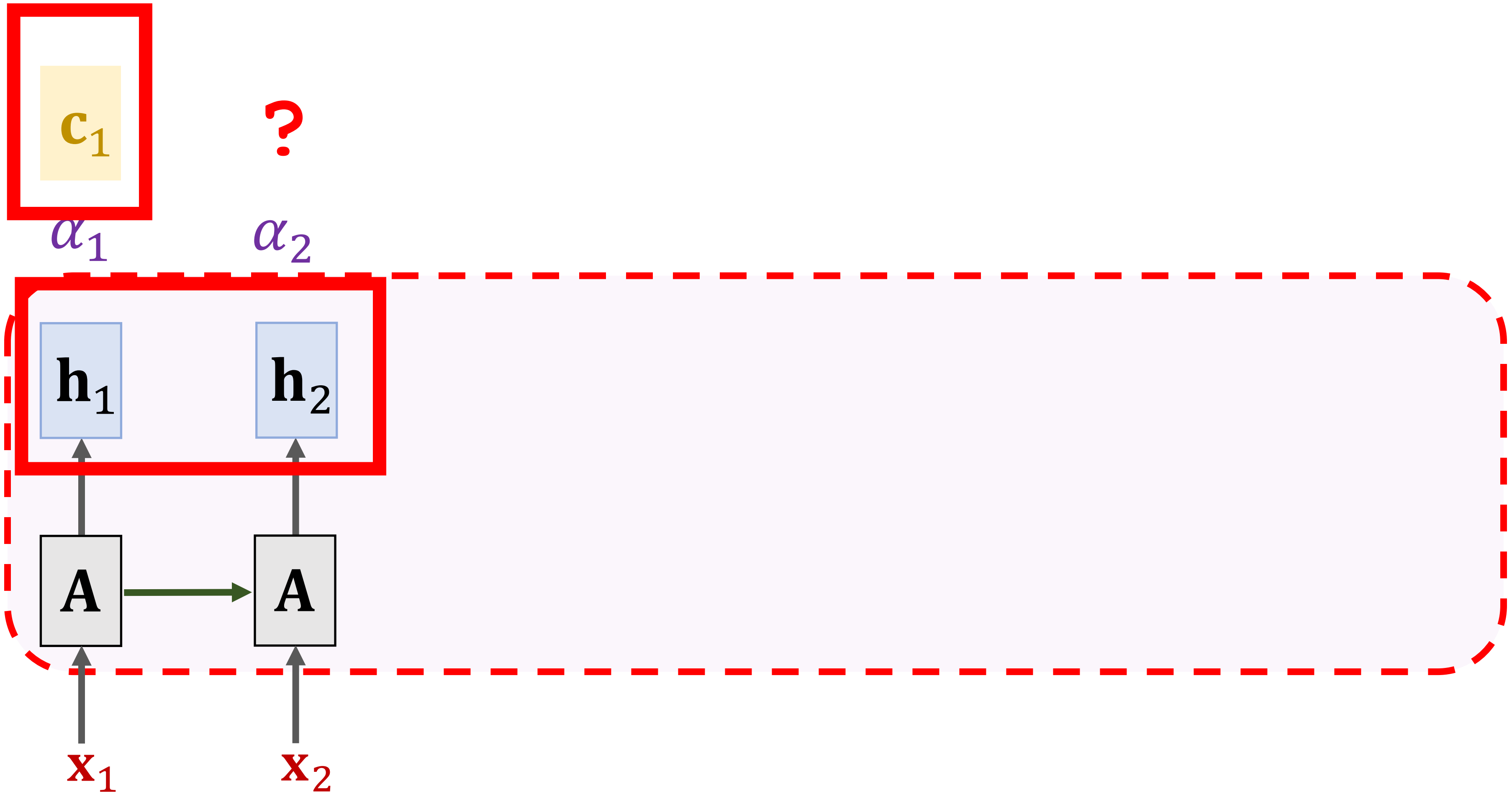
SimpleRNN + Self-Attention

Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \ \mathbf{c}_1)$



SimpleRNN + Self-Attention

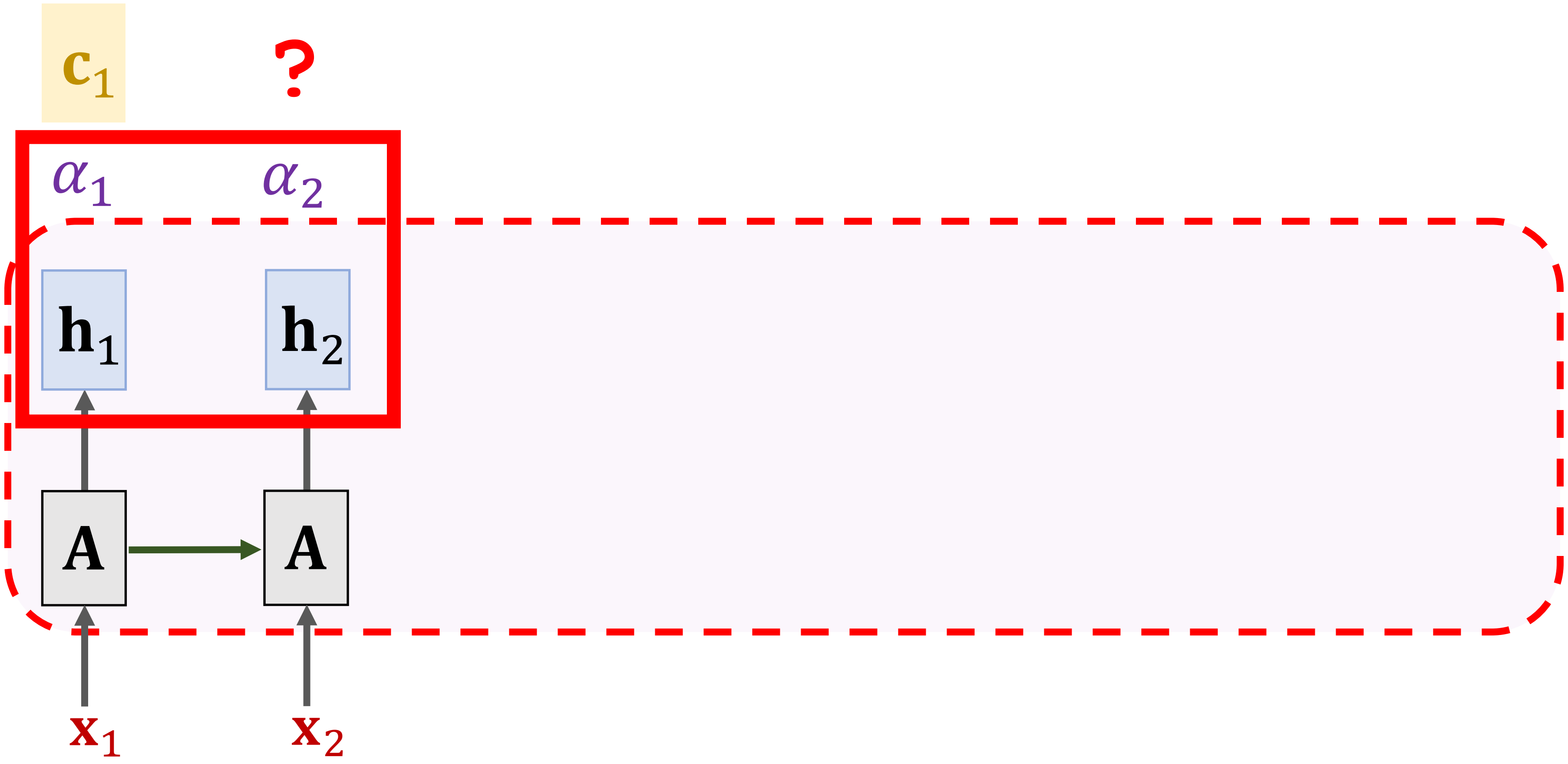
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{c}_1)$



SimpleRNN + Self-Attention

Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i,\ \mathbf{c}_1)$

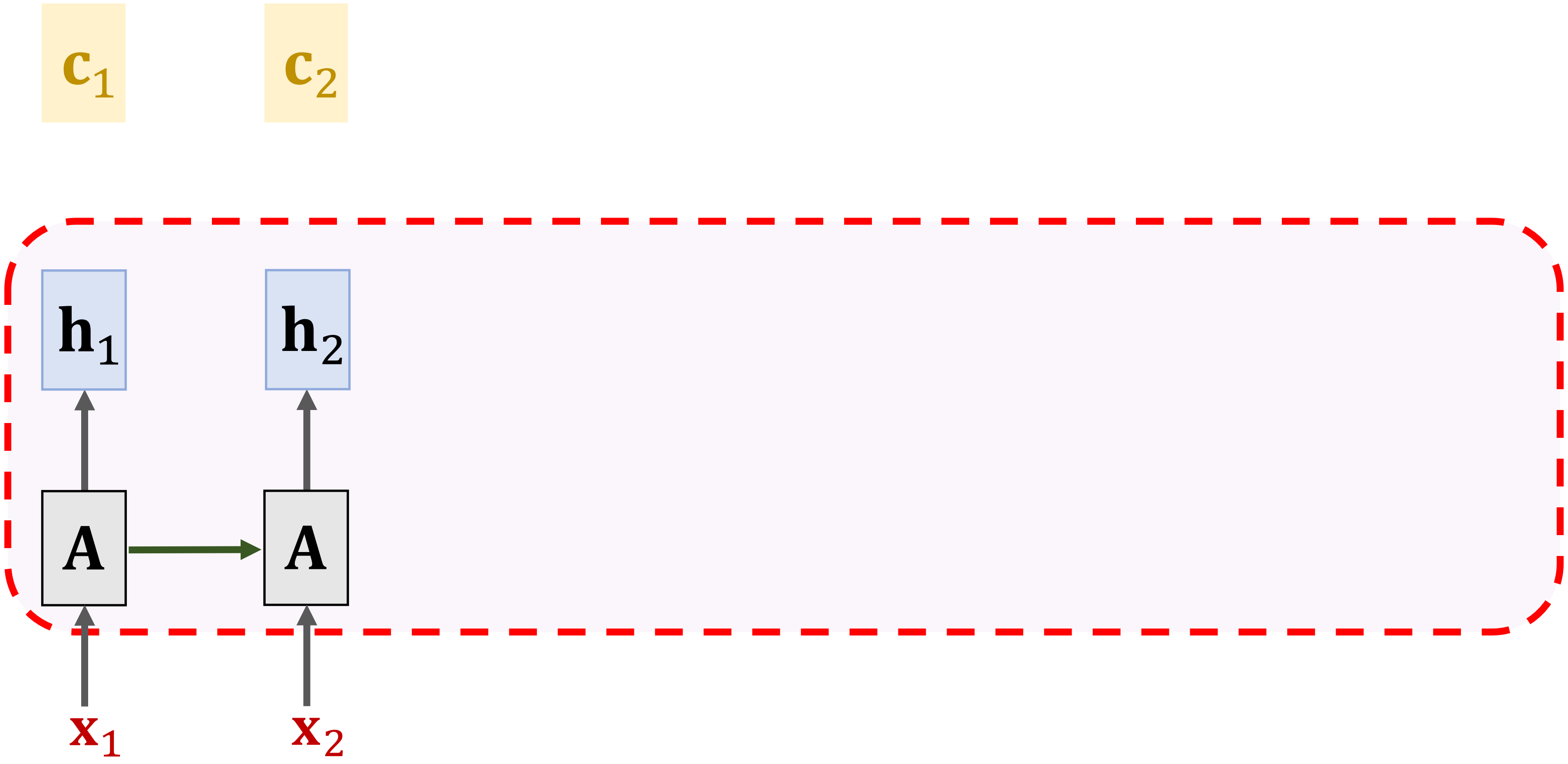
Context vector: $\mathbf{c}_2 = \alpha_1 \mathbf{h}_1 + \alpha_2 \mathbf{h}_2.$



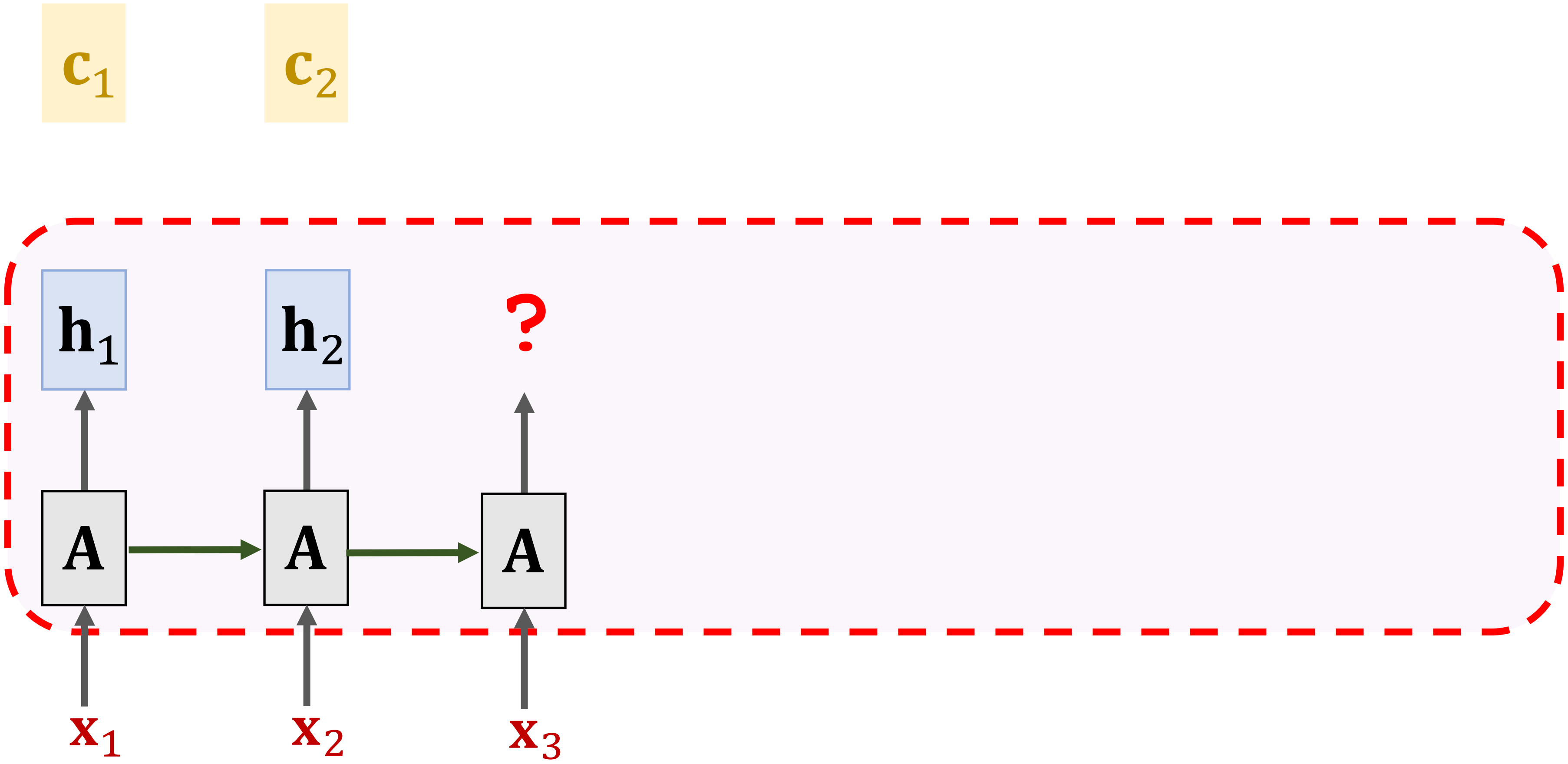
SimpleRNN + Self-Attention

Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i,\ \mathbf{c}_1)$

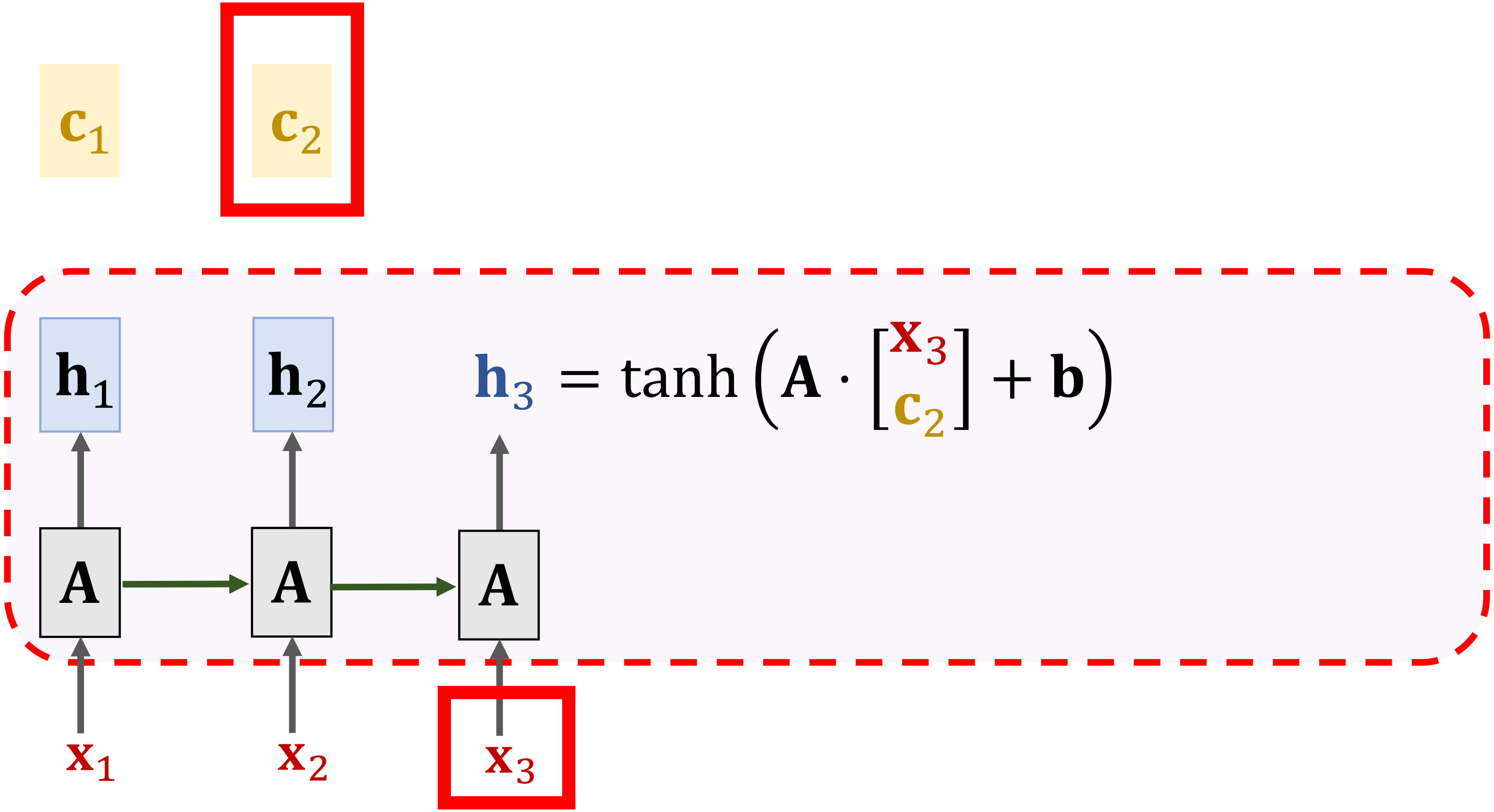
Context vector: $\mathbf{c}_2 = \alpha_1 \mathbf{h}_1 + \alpha_2 \mathbf{h}_2.$



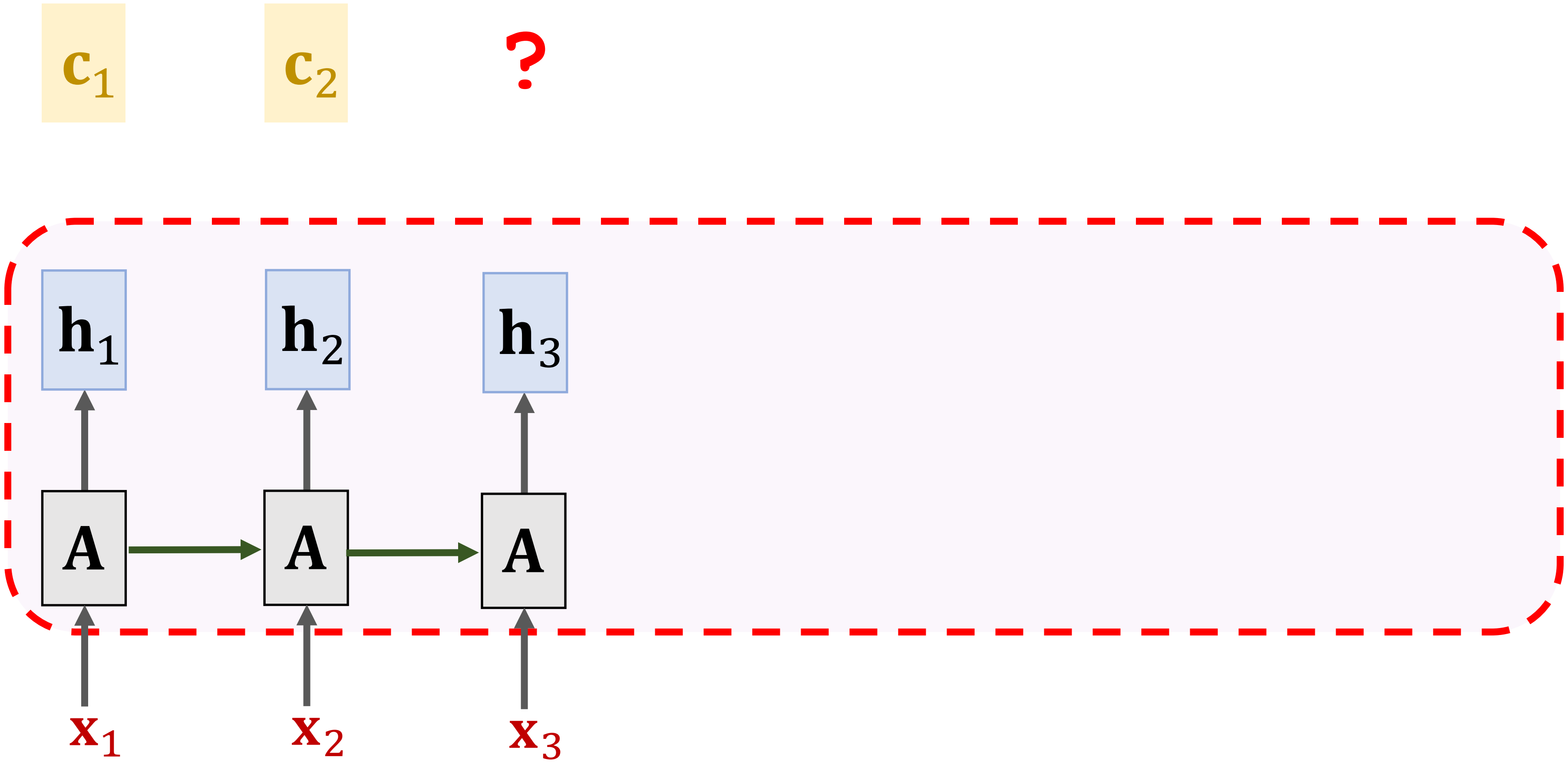
SimpleRNN + Self-Attention



SimpleRNN + Self-Attention

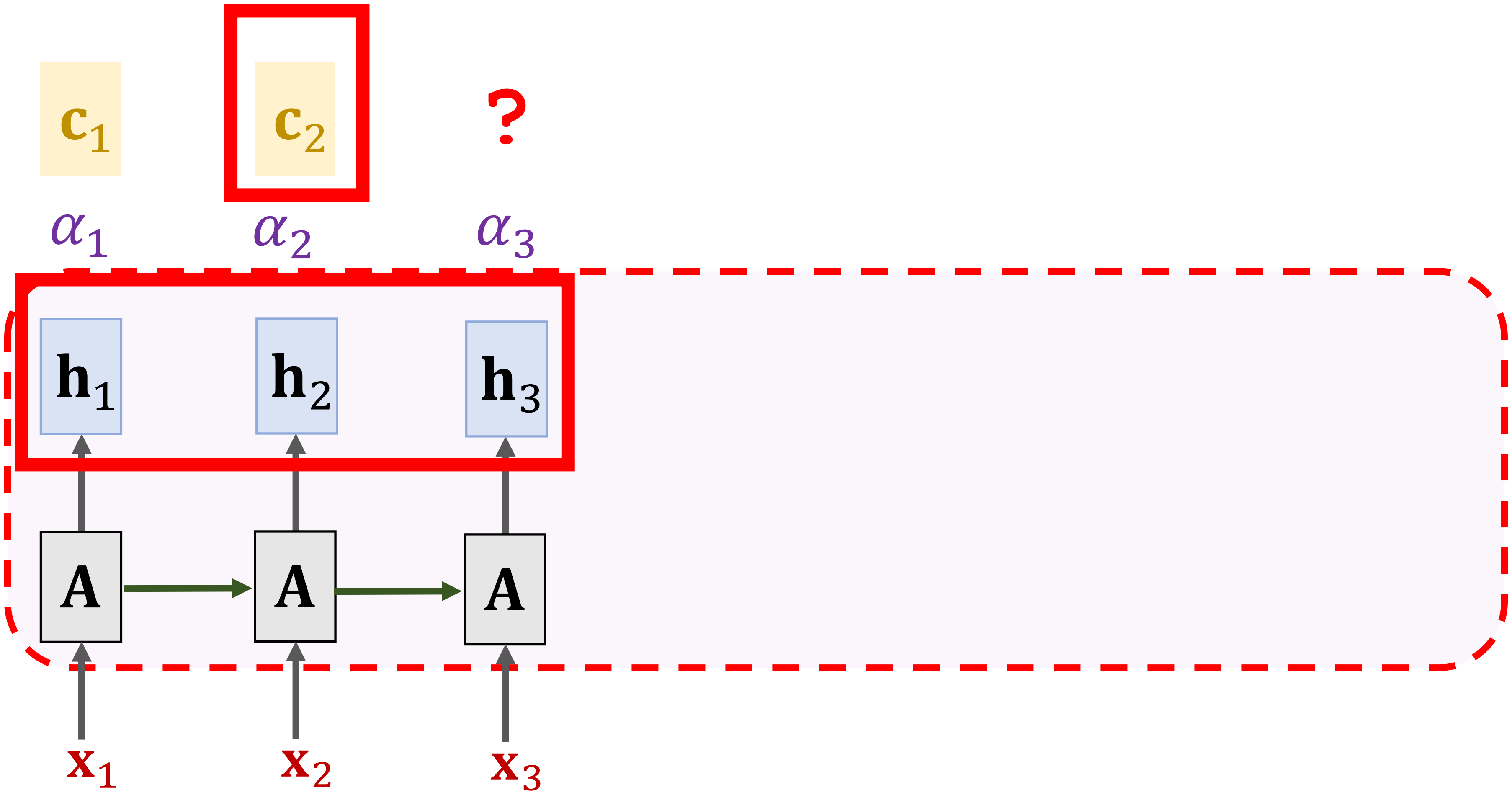


SimpleRNN + Self-Attention



SimpleRNN + Self-Attention

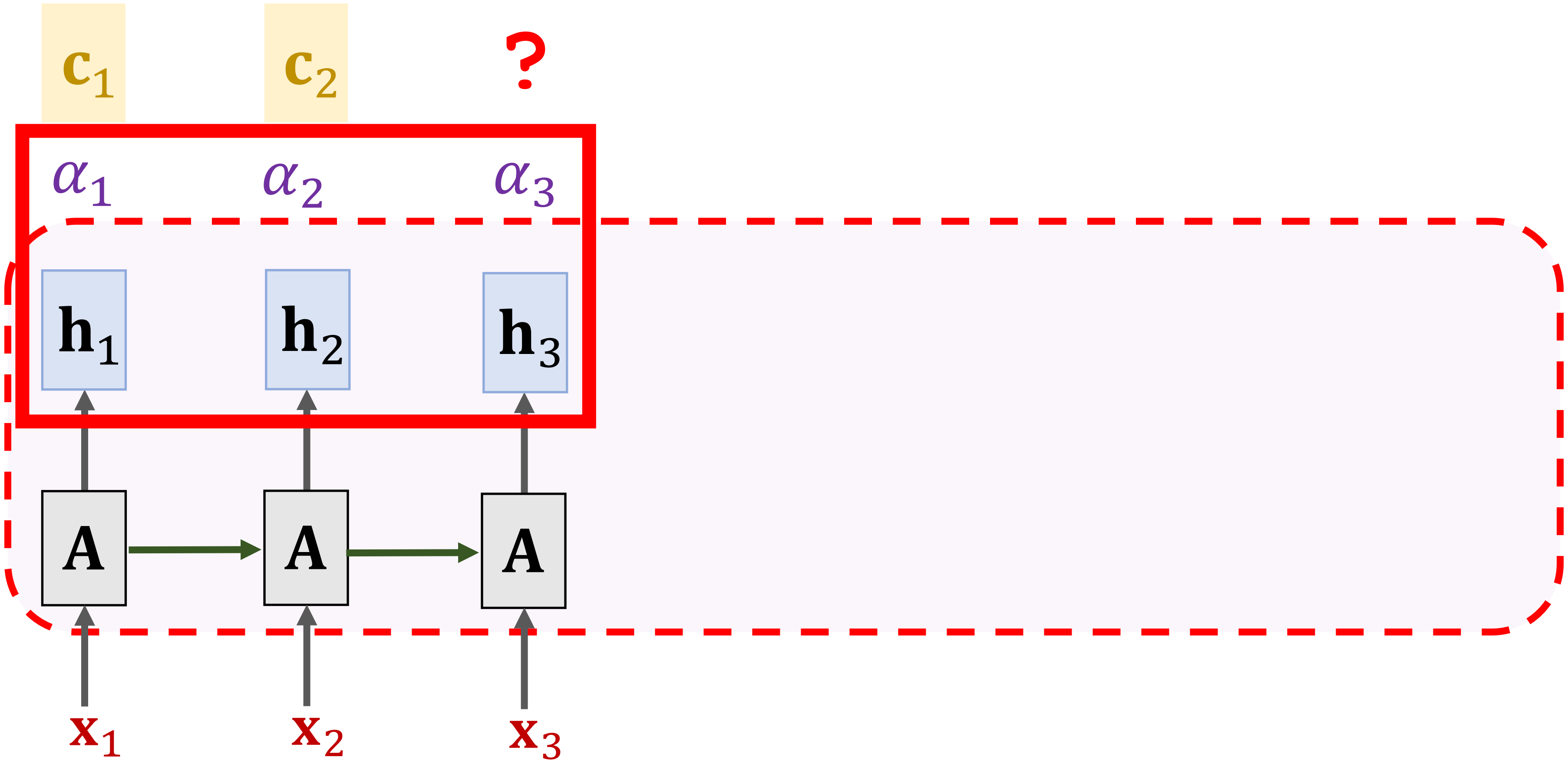
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i,\ \mathbf{c}_2)$



SimpleRNN + Self-Attention

Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \ \mathbf{c}_2)$

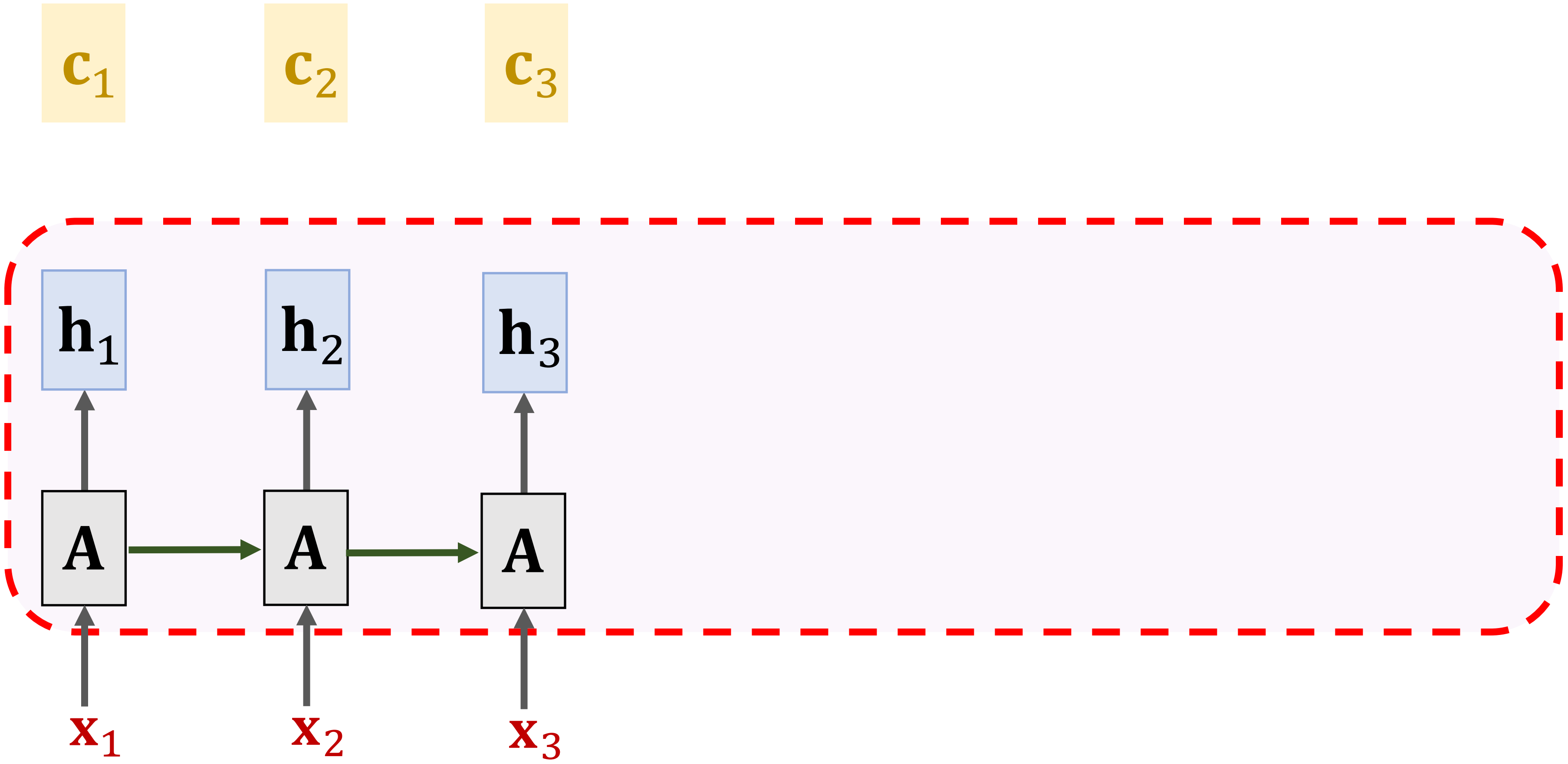
Context vector: $\mathbf{c}_3 = \alpha_1 \mathbf{h}_1 + \alpha_2 \mathbf{h}_2 + \alpha_3 \mathbf{h}_3 .$



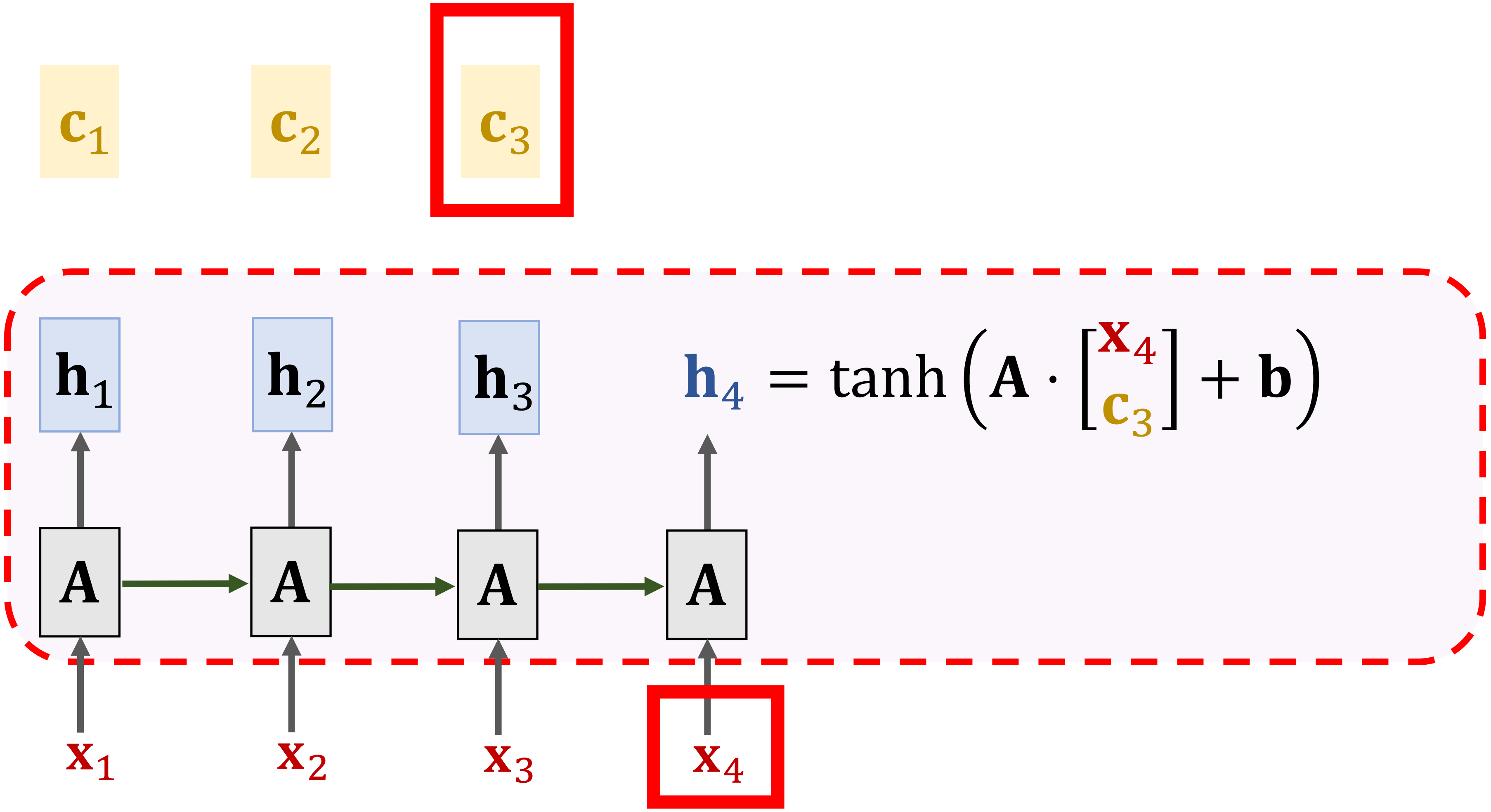
SimpleRNN + Self-Attention

Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i,\ \mathbf{c}_2)$

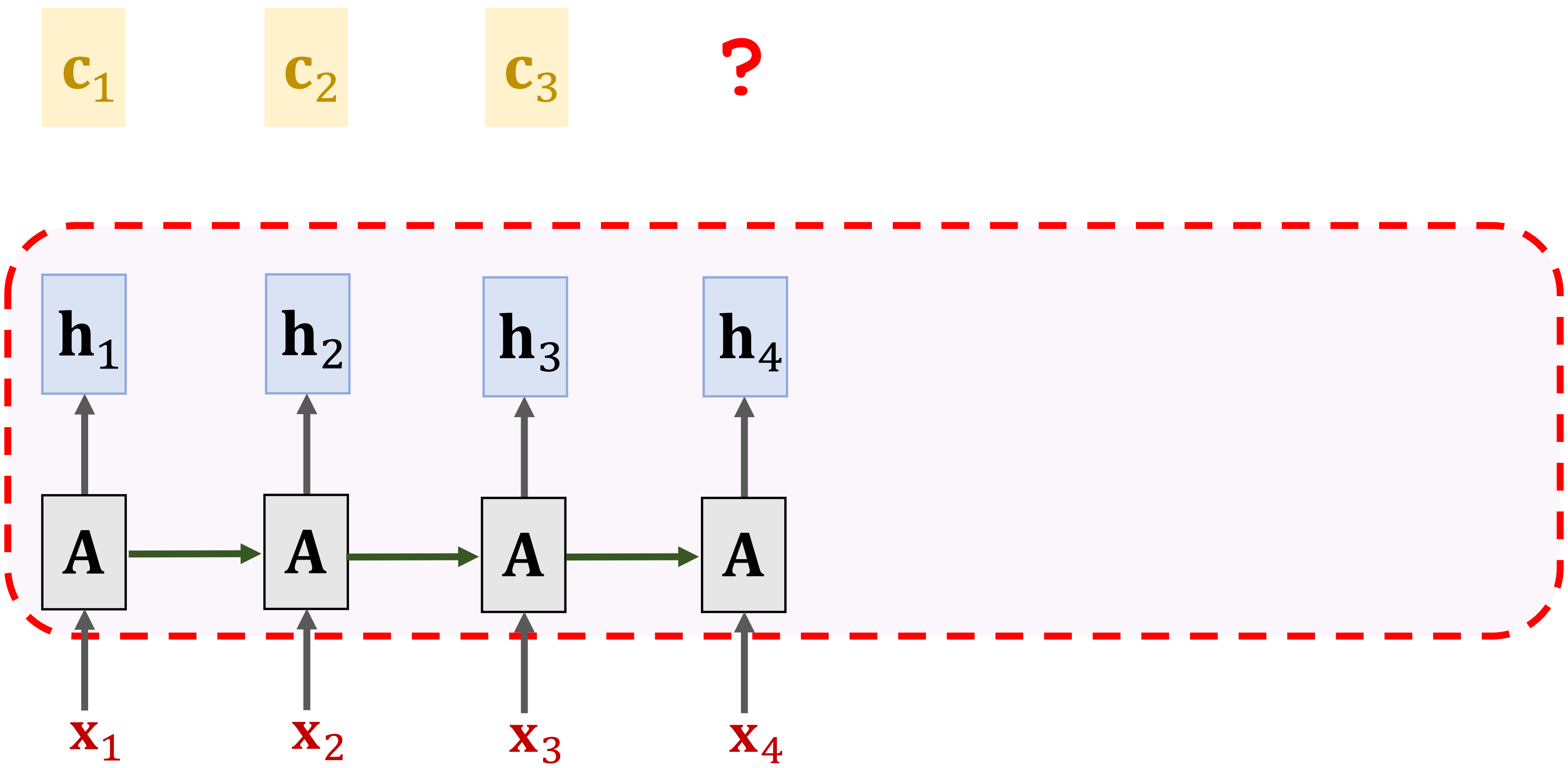
Context vector: $\mathbf{c}_3 = \alpha_1 \mathbf{h}_1 + \alpha_2 \mathbf{h}_2 + \alpha_3 \mathbf{h}_3.$



SimpleRNN + Self-Attention

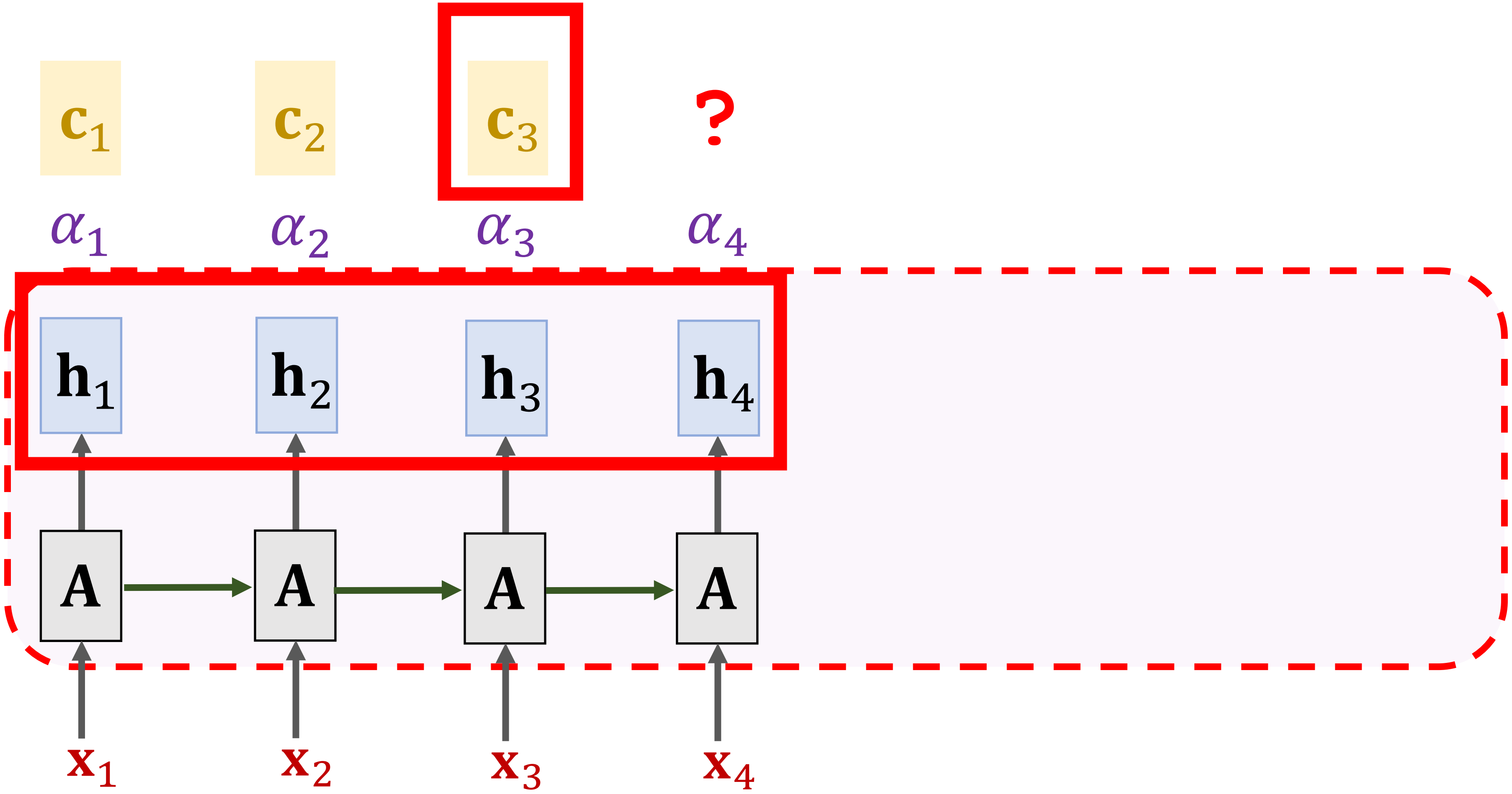


SimpleRNN + Self-Attention



SimpleRNN + Self-Attention

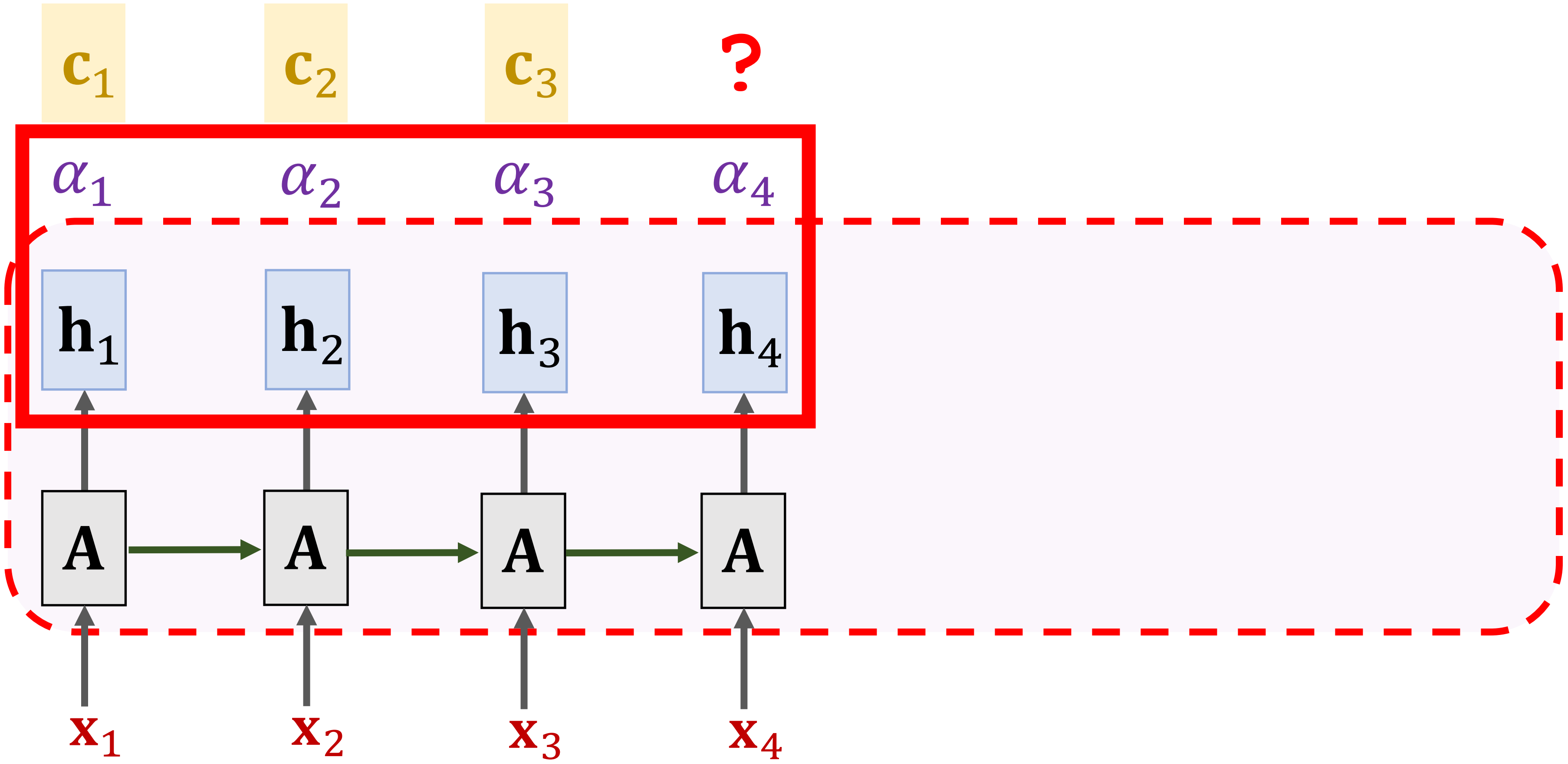
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i,\ \mathbf{c}_3)$



SimpleRNN + Self-Attention

Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i,\ \mathbf{c}_3)$

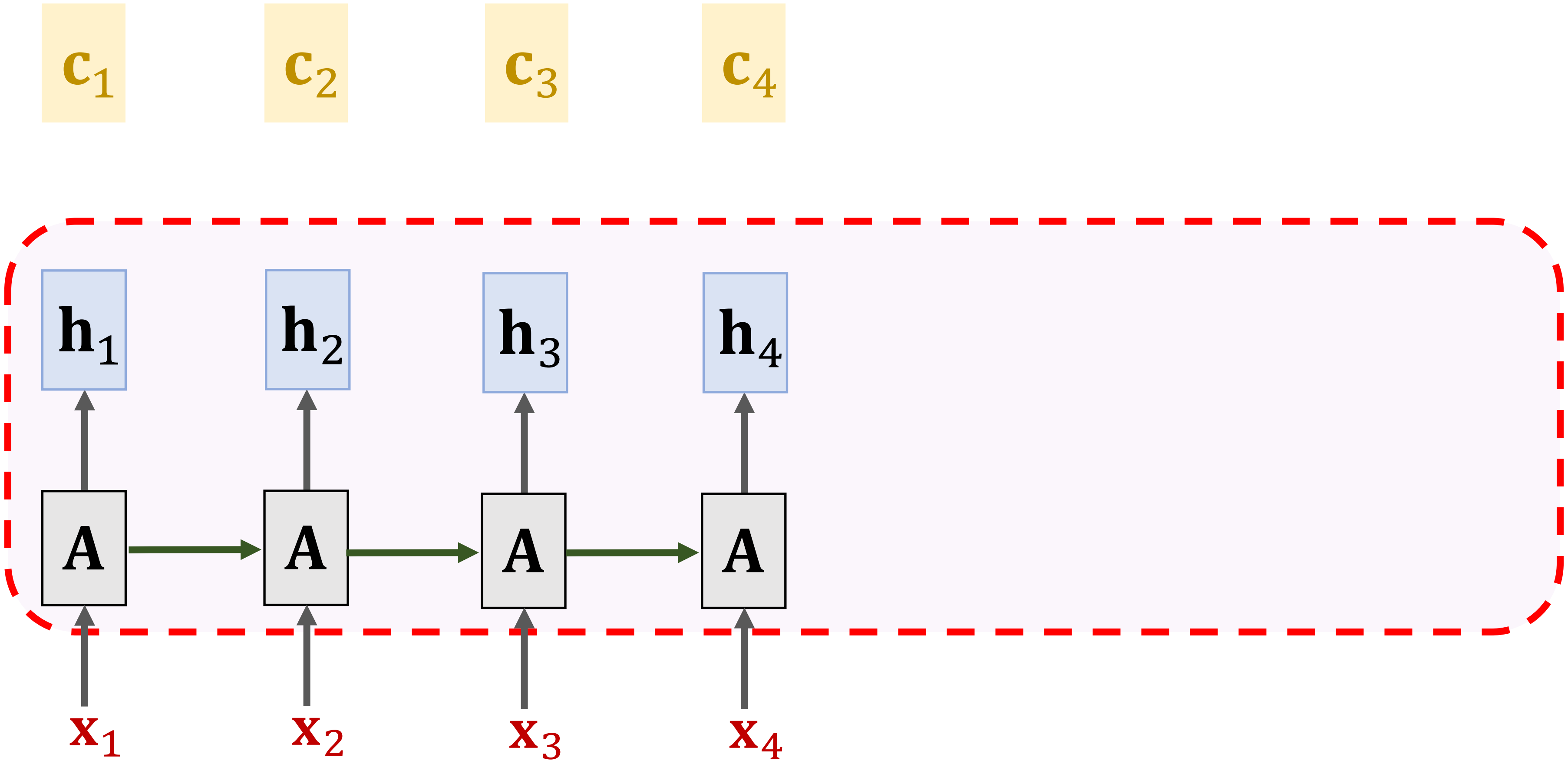
Context vector: $\mathbf{c}_4 = \alpha_1 \mathbf{h}_1 + \alpha_2 \mathbf{h}_2 + \alpha_3 \mathbf{h}_3 + \alpha_4 \mathbf{h}_4.$



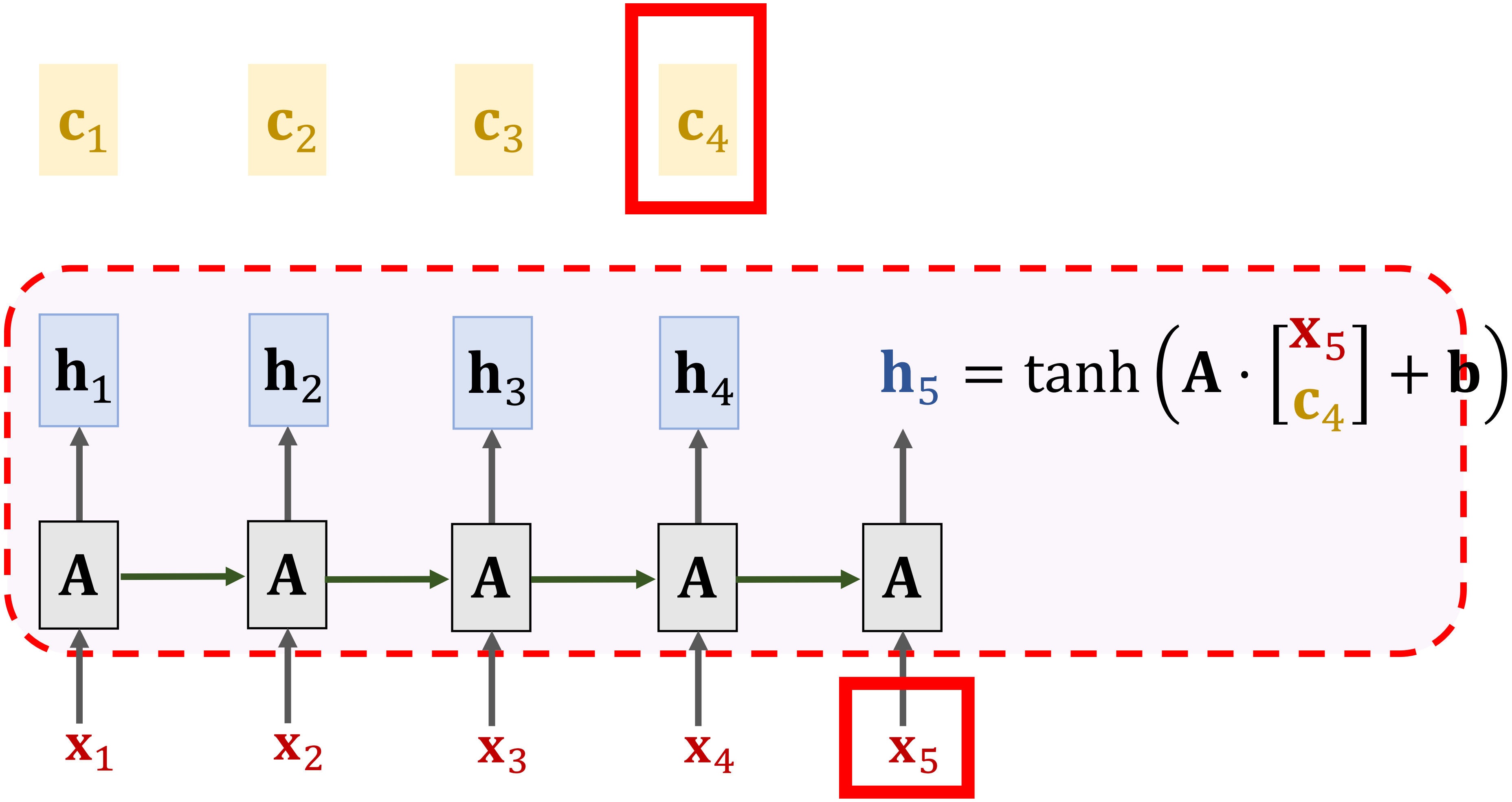
SimpleRNN + Self-Attention

Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{c}_3)$

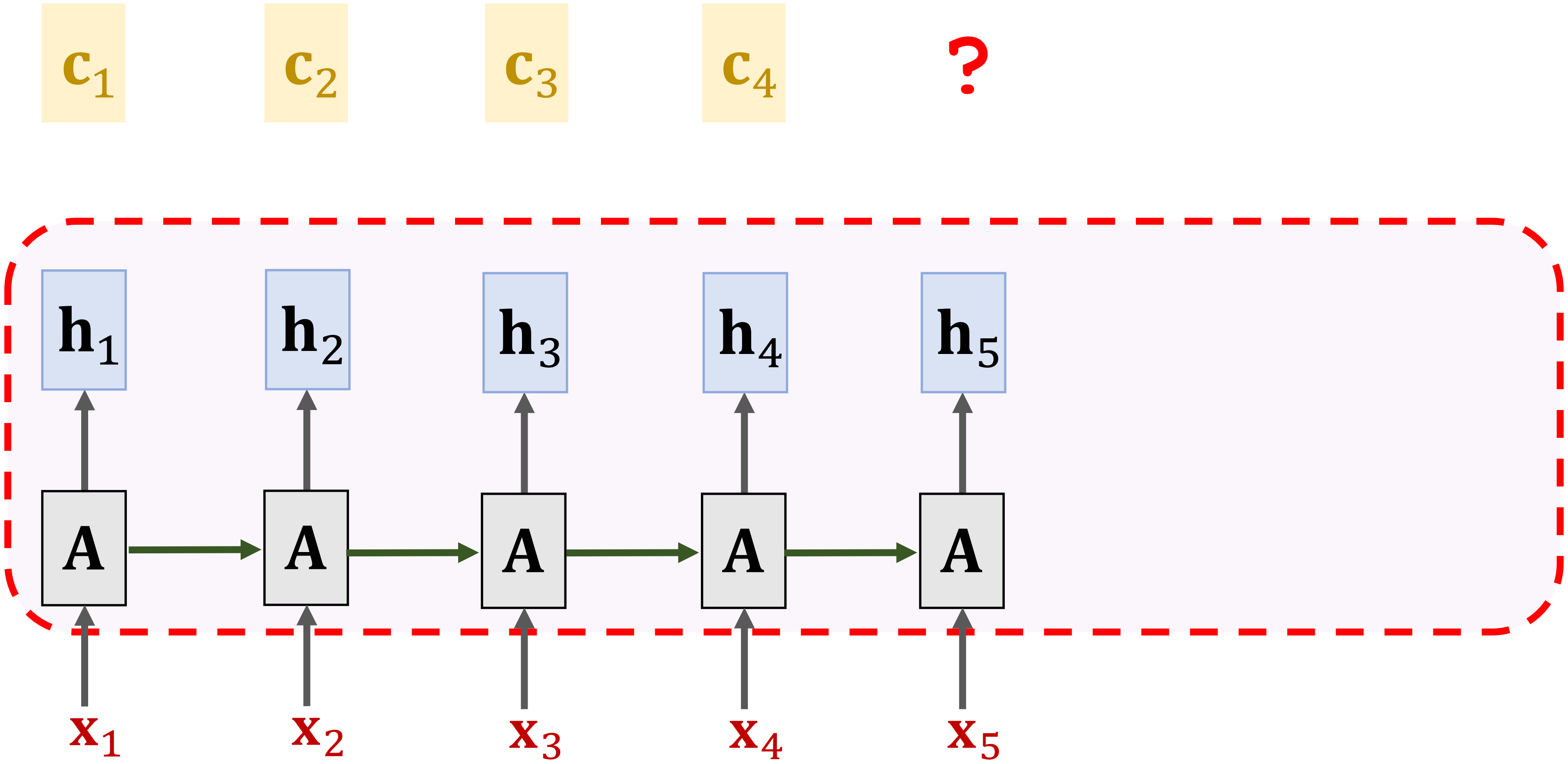
Context vector: $\mathbf{c}_4 = \alpha_1 \mathbf{h}_1 + \alpha_2 \mathbf{h}_2 + \alpha_3 \mathbf{h}_3 + \alpha_4 \mathbf{h}_4.$



SimpleRNN + Self-Attention



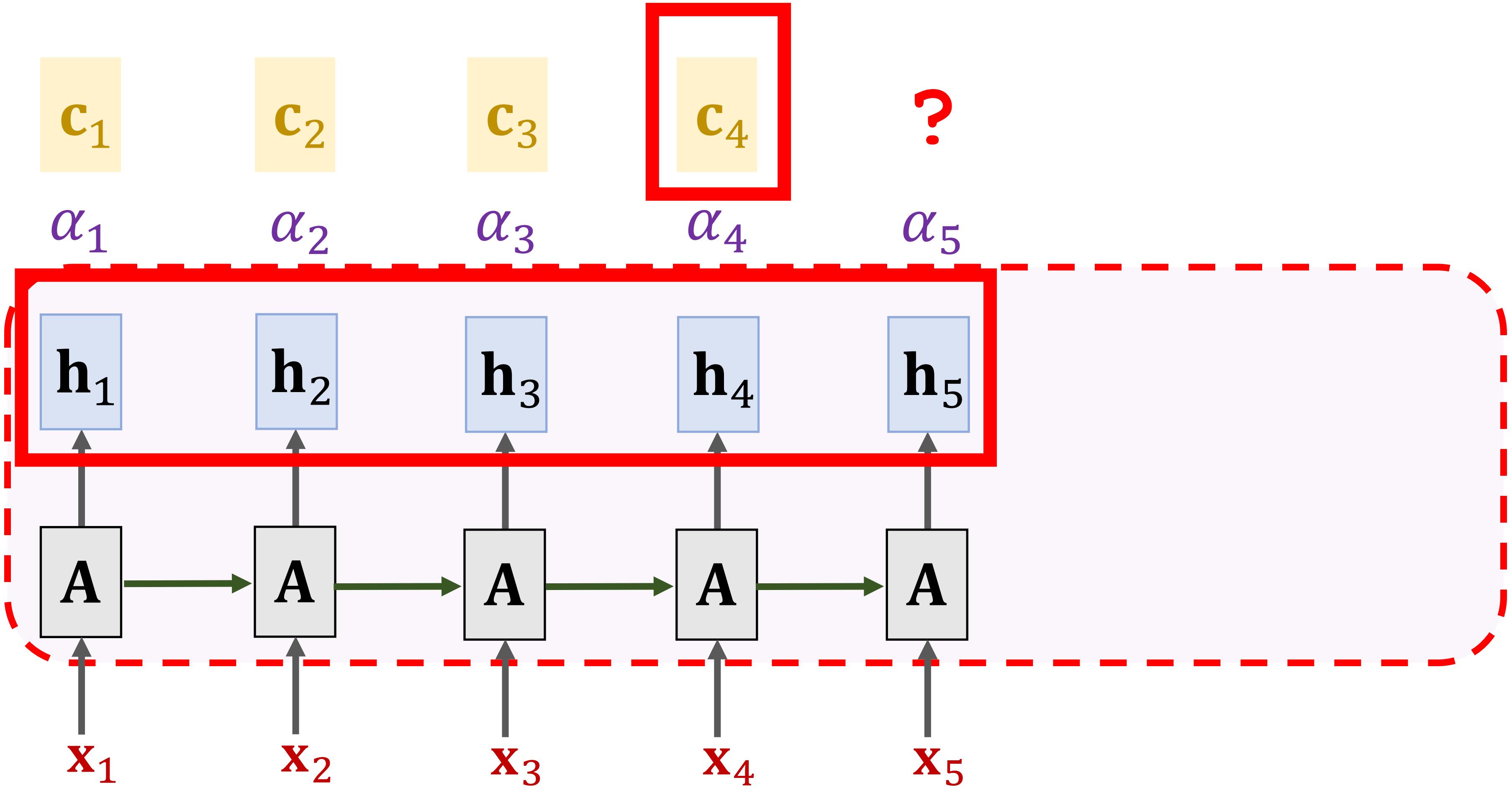
SimpleRNN + Self-Attention



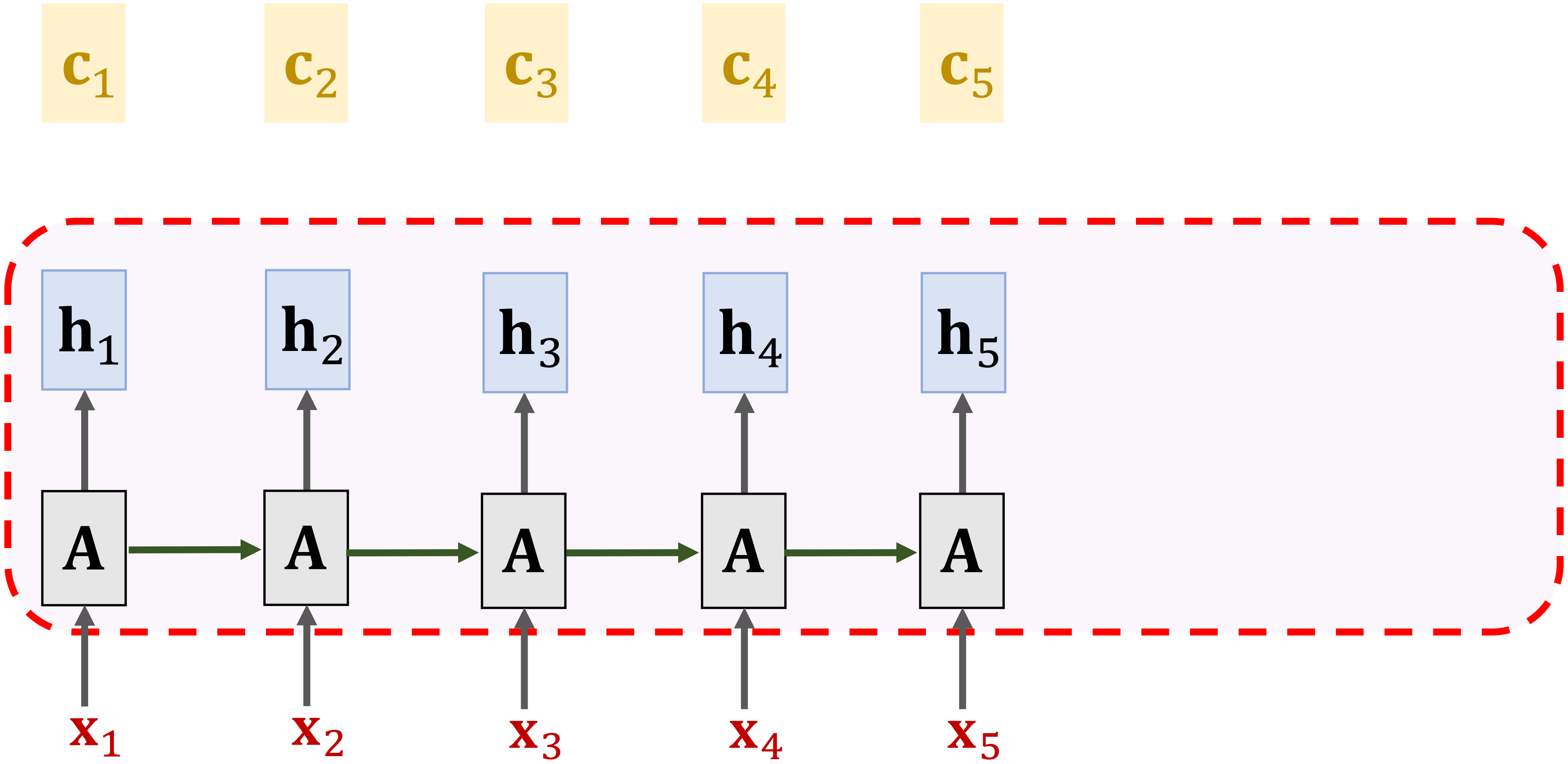
SimpleRNN + Self-Attention

Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i,\ \mathbf{c}_3)$

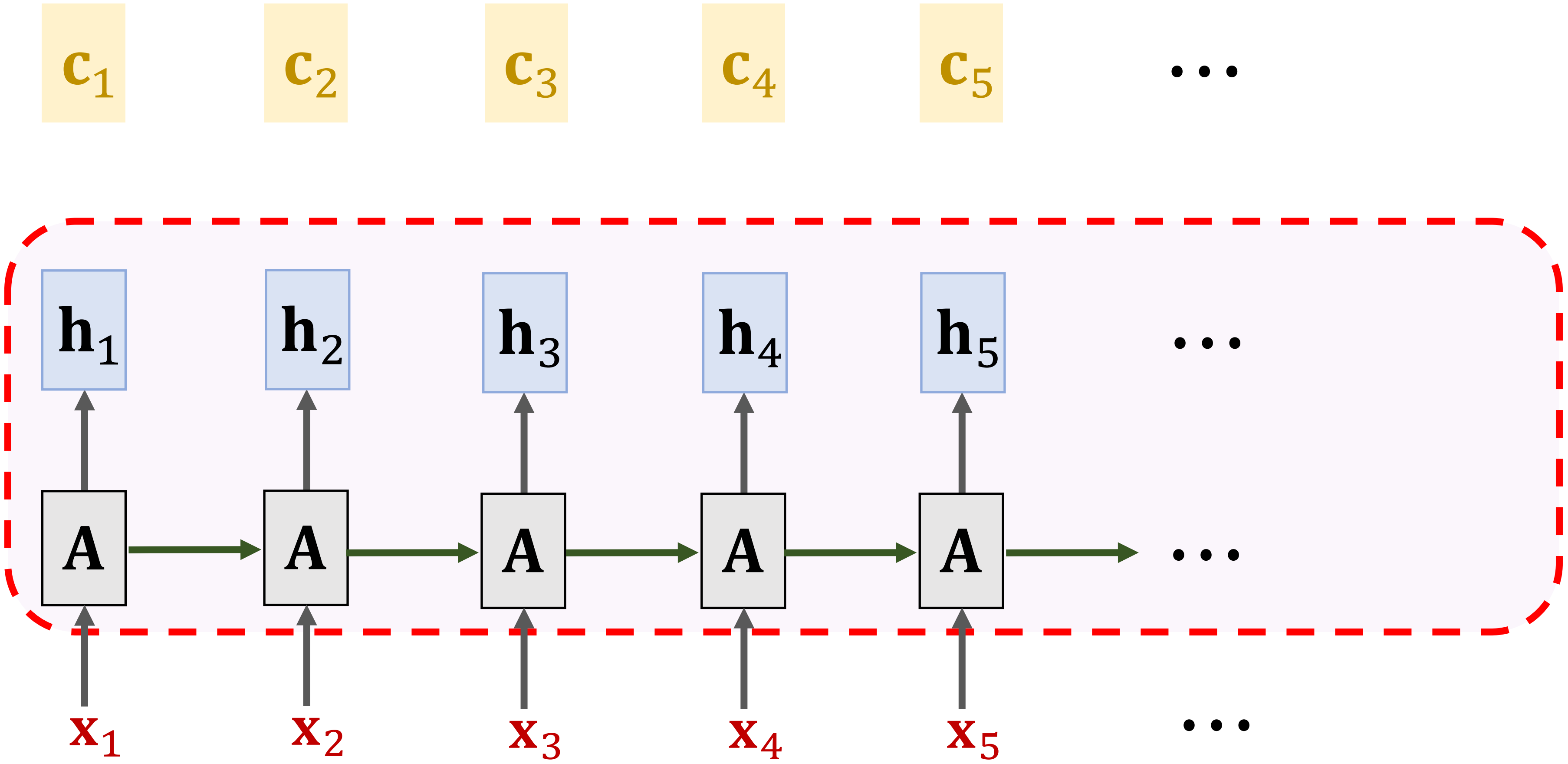
Context vector: $\mathbf{c}_5 = \alpha_1 \mathbf{h}_1 + \alpha_2 \mathbf{h}_2 + \cdots + \alpha_5 \mathbf{h}_5.$



SimpleRNN + Self-Attention



SimpleRNN + Self-Attention



Summary

- With self-attention, RNN is less likely to forget.

Summary

- With self-attention, RNN is less likely to forget.
- Pay attention to the context relevant to the new input.

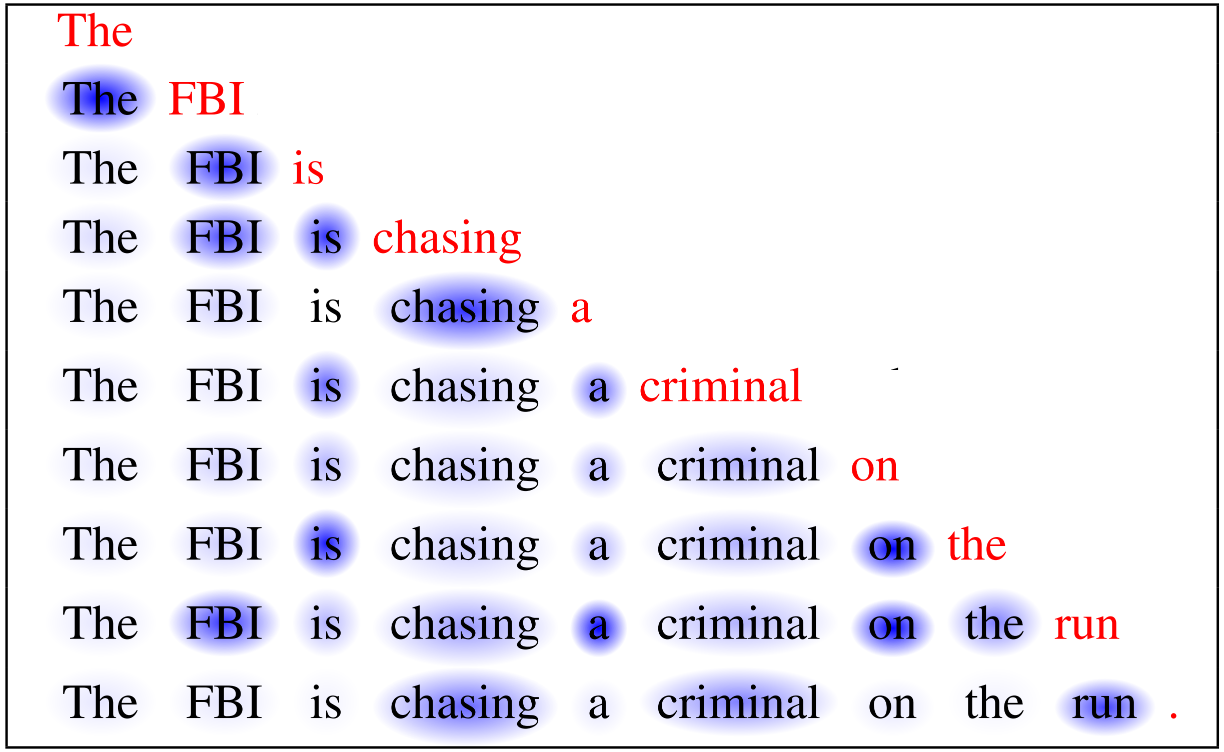


Figure is from the paper " Long Short-Term Memory-Networks for Machine Reading."

SimpleRNN with Self-Attention

- $\mathbf{c}_t = \sum_{i=1}^{t-1} \alpha_i \, \mathbf{h}_i$.
- α_i is computed by a neural network taking $\mathbf{x}_t$, $\mathbf{c}_{t-1}$, and $\mathbf{h}_i$ as inputs.

$$\alpha_i = \mathbf{v}^T \cdot \tanh\left[\mathbf{W}_{\mathrm{x}} \cdot \mathbf{x}_t + \mathbf{W}_{\mathrm{s}} \cdot \mathbf{c}_{t-1} + \mathbf{W}_{\mathrm{h}} \cdot \mathbf{h}_i \right]$$

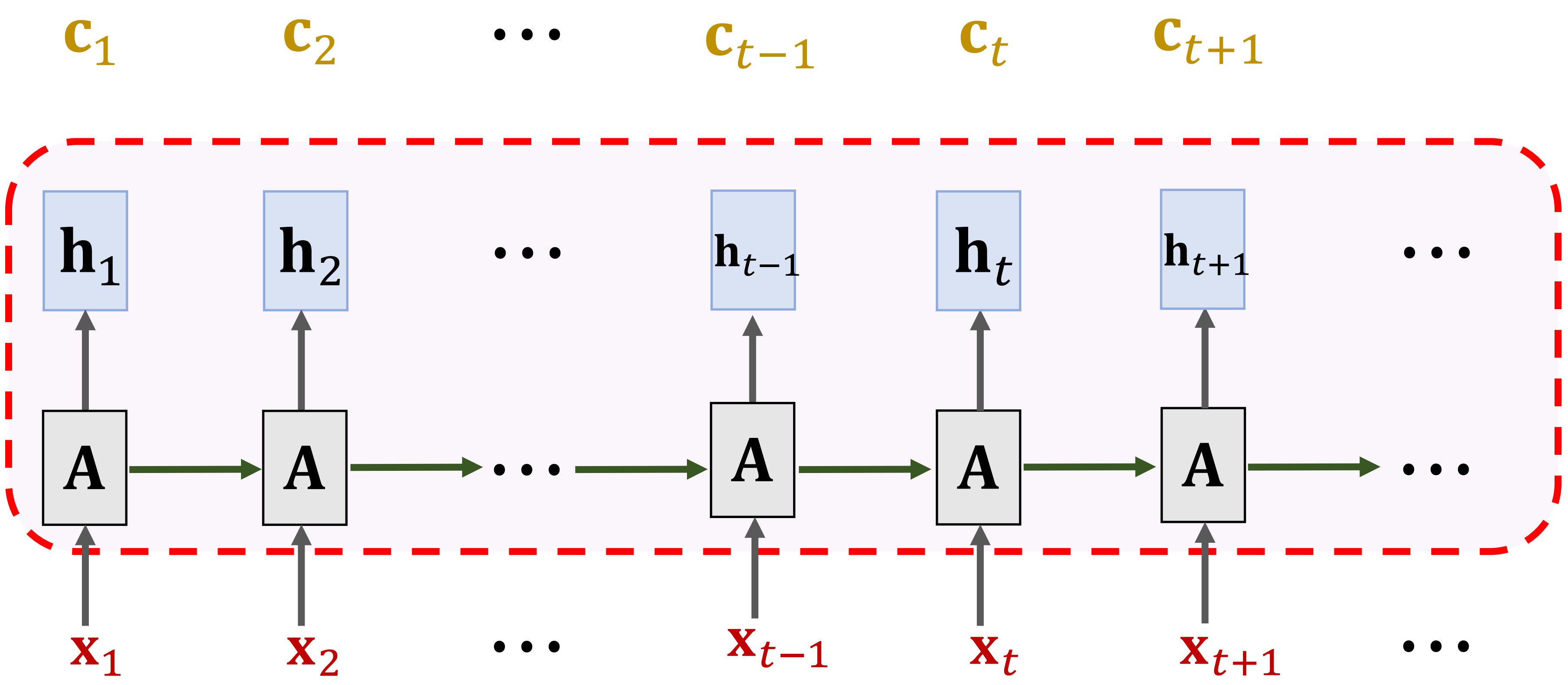
SimpleRNN with Self-Attention

- $\mathbf{c}_t = \sum_{i=1}^{t-1} \alpha_i \, \mathbf{h}_i$.
- α_i is computed by a neural network taking $\mathbf{x}_t$, $\mathbf{c}_{t-1}$, and $\mathbf{h}_i$ as inputs.

$$\alpha_i = \boxed{\mathbf{v}^T} \cdot \tanh\left[\boxed{\mathbf{W}_\mathrm{x}} \cdot \mathbf{x}_t + \boxed{\mathbf{W}_\mathrm{c}} \cdot \mathbf{c}_{t-1} + \boxed{\mathbf{W}_\mathrm{h}} \cdot \mathbf{h}_i \right]$$

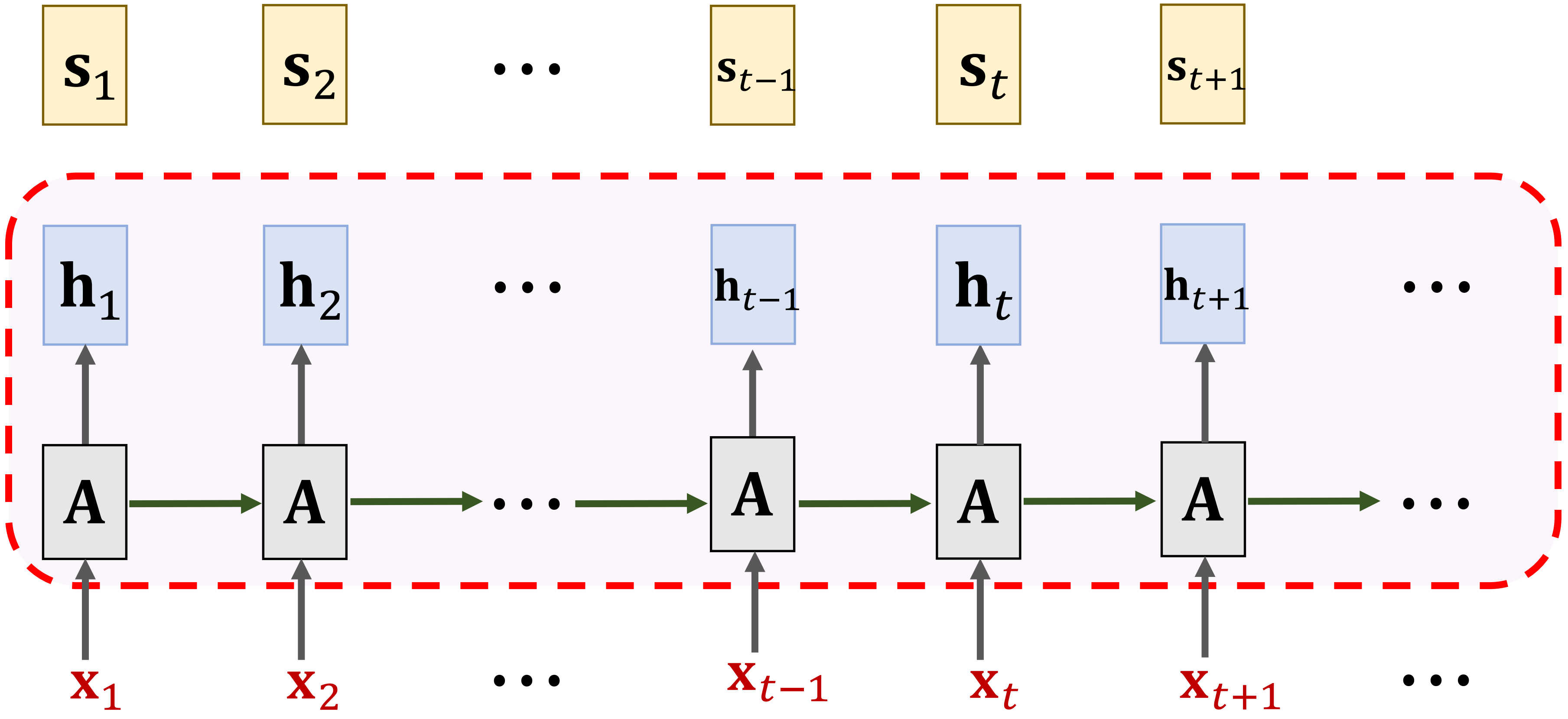
Trainable parameters

SimpleRNN with Self-Attention



LSTM with Self-Attention

- The update of LSTM with self-attention is analogous.



LSTM with Self-Attention

- The update of LSTM with self-attention is analogous.
- $\mathbf{s}_t = \sum_{i=1}^{t-1} \alpha_i \mathbf{h}_i.$

Exactly the same as SimpleRNN

LSTM with Self-Attention

- The update of LSTM with self-attention is analogous.
- $\mathbf{s}_t = \sum_{i=1}^{t-1} \alpha_i \, \mathbf{h}_i.$ Exactly the same as SimpleRNN
- $\mathbf{c}'_t = \sum_{i=1}^{t-1} \alpha_i \, \mathbf{c}_i.$

Conveyor belt

LSTM with Self-Attention

- The update of LSTM with self-attention is analogous.
- $\mathbf{s}_t = \sum_{i=1}^{t-1} \alpha_i \ \mathbf{h}_i.$
- $\mathbf{c}_t' = \sum_{i=1}^{t-1} \alpha_i \ \mathbf{c}_i.$
- $\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_t' + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t.$

Forget gate, computed using $\mathbf{x}_t$ and $\mathbf{s}_t$.

New value, computed using $\mathbf{x}_t$ and $\mathbf{s}_t$.

Input gate, computed using $\mathbf{x}_t$ and $\mathbf{s}_t$.

LSTM with Self-Attention

- The update of LSTM with self-attention is analogous.
- $\mathbf{s}_t = \sum_{i=1}^{t-1} \alpha_i \, \mathbf{h}_i.$
- $\mathbf{c}'_t = \sum_{i=1}^{t-1} \alpha_i \, \mathbf{c}_i.$
- $\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}'_t + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t.$
- $\mathbf{h}_t = \mathbf{o}_t \circ \mathbf{c}_t.$

Output gate, computed using $\mathbf{x}_t$ and $\mathbf{s}_t$.

LSTM with Self-Attention

- The update of LSTM with self-attention is analogous.
- $\mathbf{s}_t = \sum_{i=1}^{t-1} \alpha_i \, \mathbf{h}_i.$
- $\mathbf{c}'_t = \sum_{i=1}^{t-1} \alpha_i \, \mathbf{c}_i.$
- $\mathbf{c}_t = \boxed{\mathbf{f}_t} \circ \mathbf{c}'_t + \boxed{\mathbf{i}_t} \circ \boxed{\tilde{\mathbf{c}}_t.}$
- $\mathbf{h}_t = \boxed{\mathbf{o}_t} \circ \mathbf{c}_t.$

Difference 1:

- In standard LSTM, the gates and new value are computed using $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$.
- With self-attention, the gates and new value are computed using $\mathbf{x}_t$ and $\mathbf{s}_t$.

LSTM with Self-Attention

- The update of LSTM with self-attention is analogous.
- $\mathbf{s}_t = \sum_{i=1}^{t-1} \alpha_i \mathbf{h}_i.$
- $\mathbf{c}'_t = \sum_{i=1}^{t-1} \alpha_i \mathbf{c}_i.$
- $\mathbf{c}_t = \mathbf{f}_t \circ \boxed{\mathbf{c}'_t} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t.$
- $\mathbf{h}_t = \mathbf{o}_t \circ \mathbf{c}_t.$

Difference 2:

- In standard LSTM, apply forget gate to $\mathbf{c}_{t-1}$.
- With self-attention, apply forget gate to $\mathbf{c}'_t = \sum_{i=1}^{t-1} \alpha_i \mathbf{c}_i$.