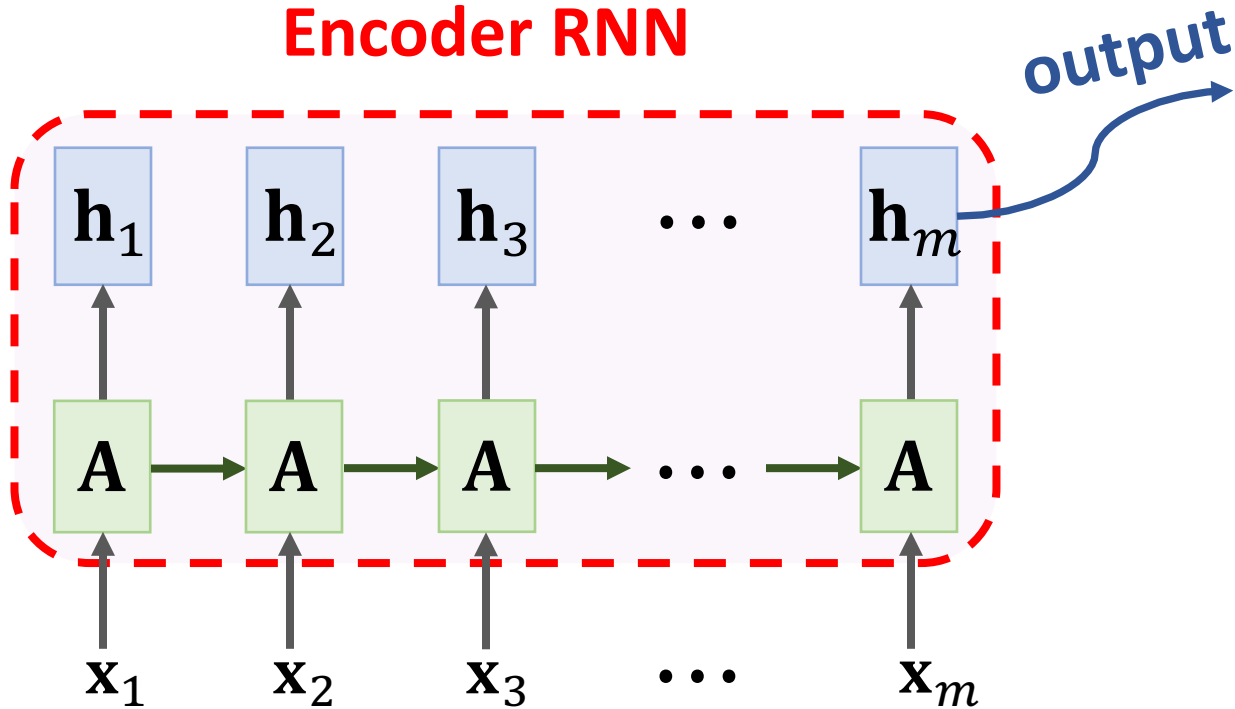


Attention

Shusen Wang

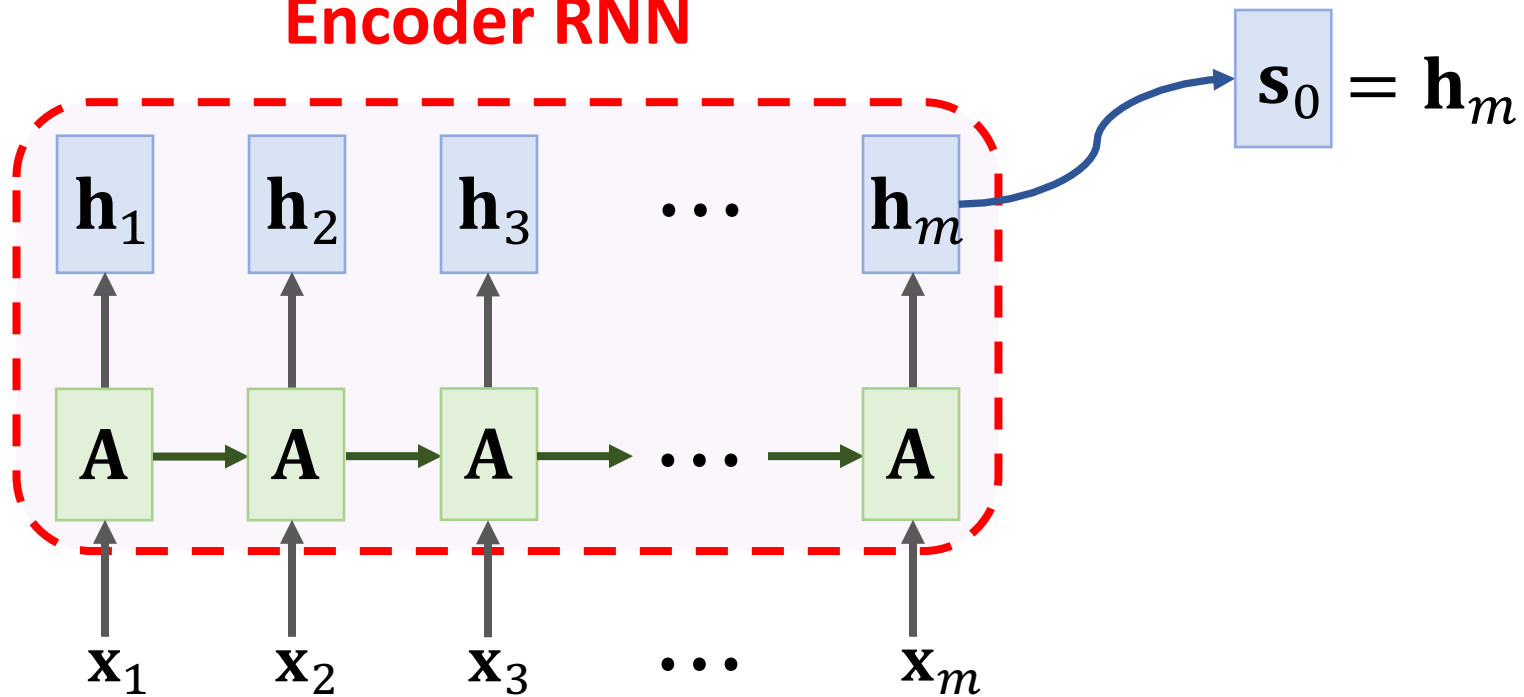
Seq2Seq Model

Encoder RNN



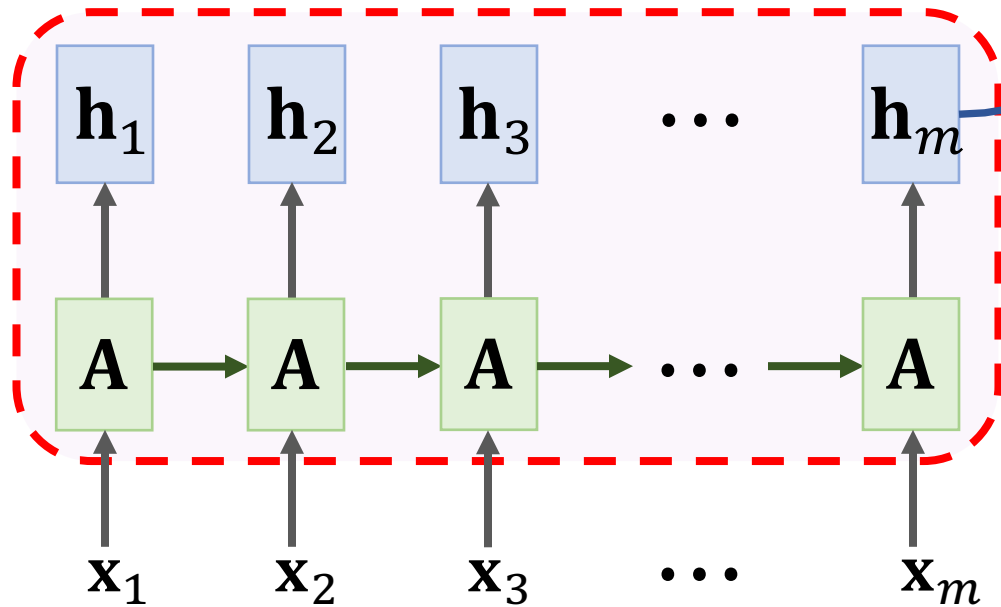
Seq2Seq Model

Encoder RNN

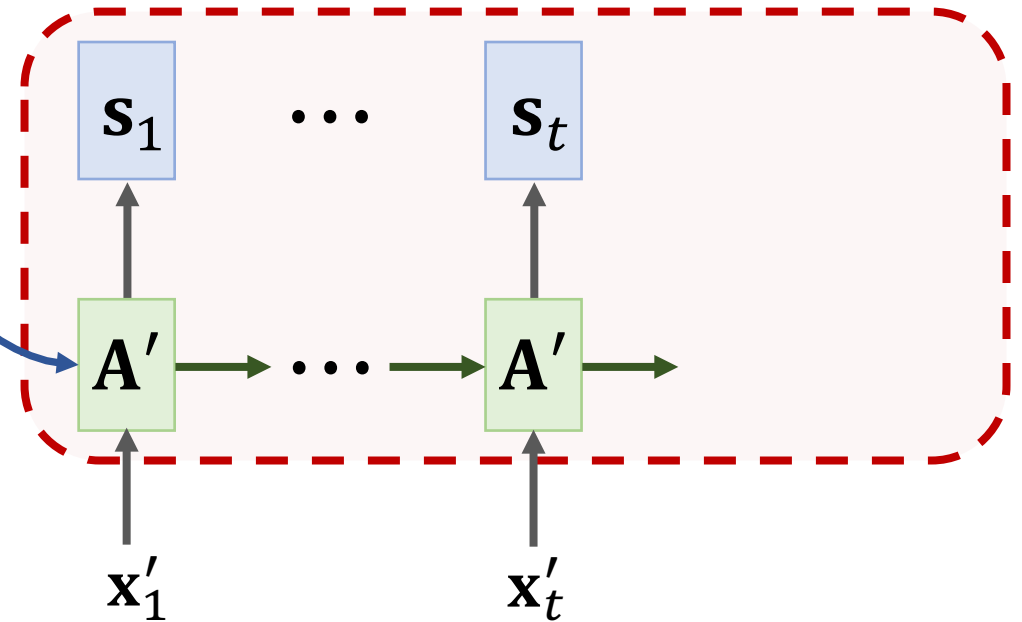


Seq2Seq Model

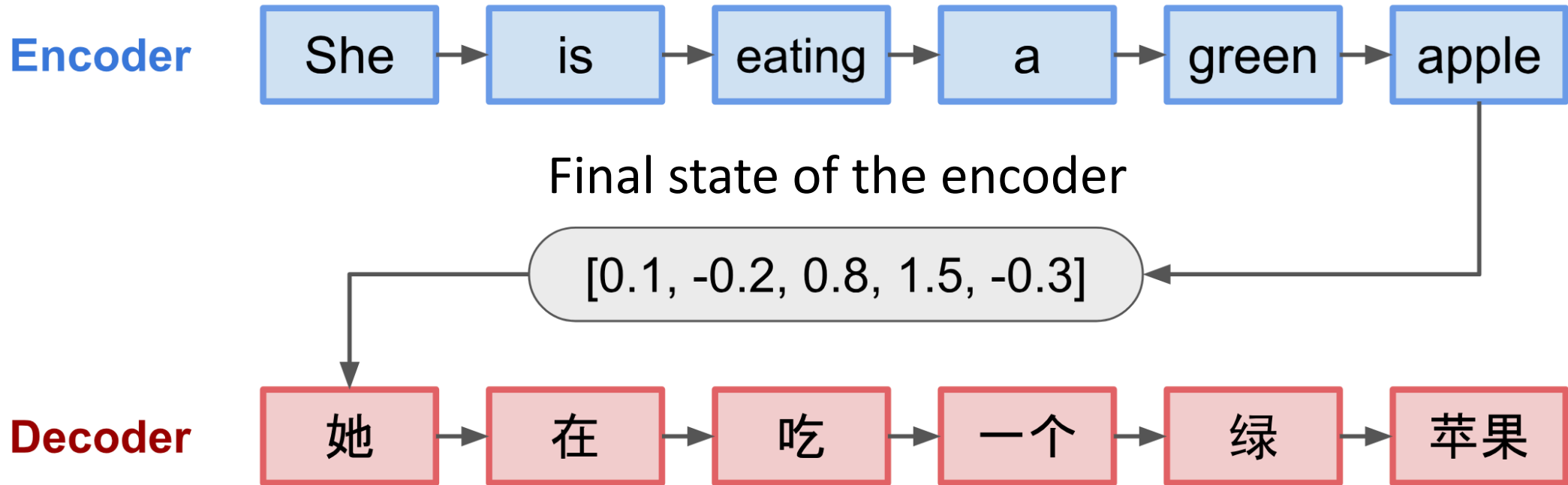
Encoder RNN



Decoder RNN



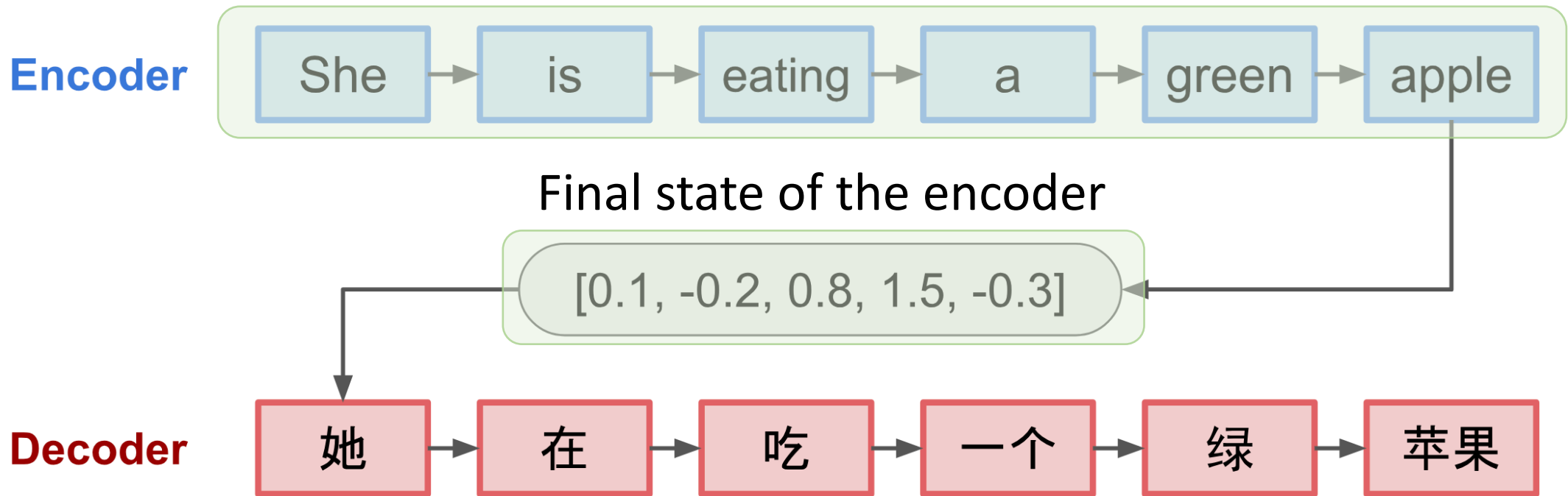
Seq2Seq Model



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

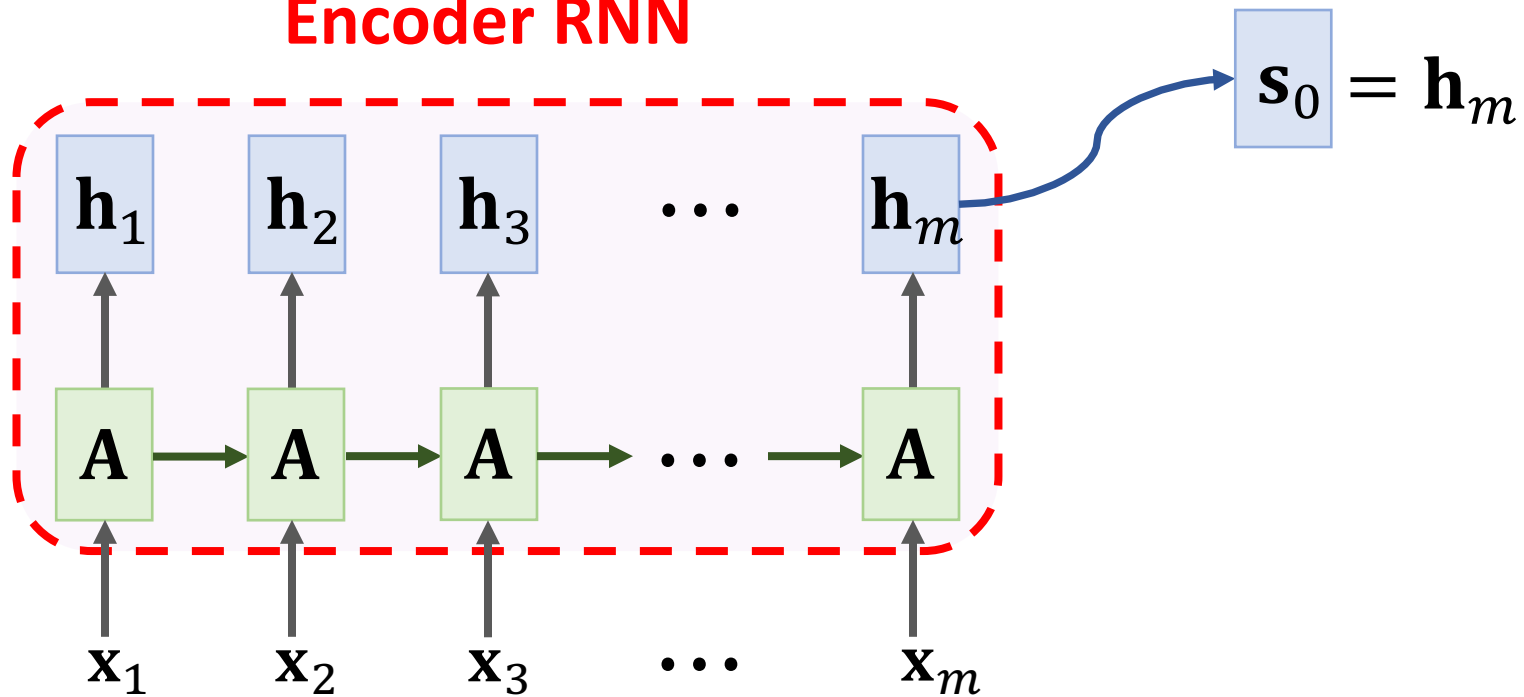
Seq2Seq Model with Attention

Original paper:

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

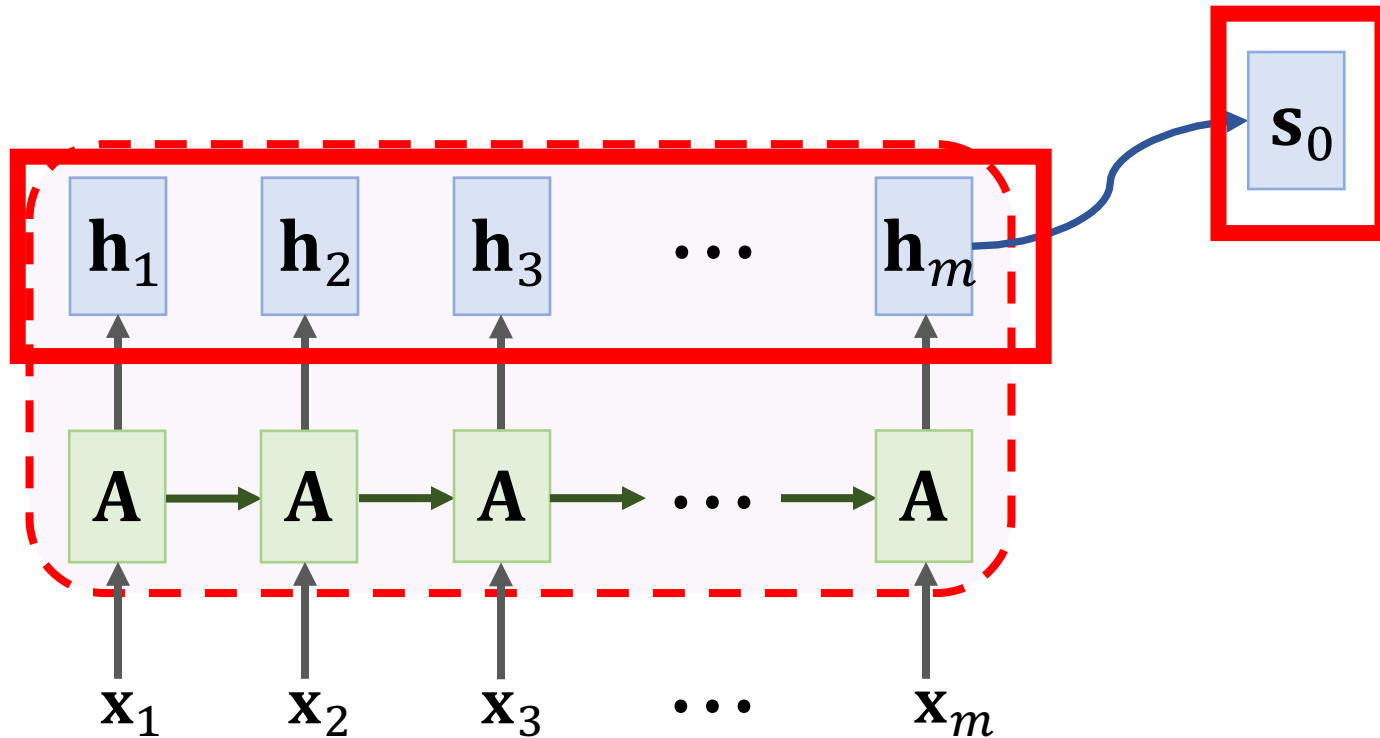
SimpleRNN + Attention

Encoder RNN



SimpleRNN + Attention

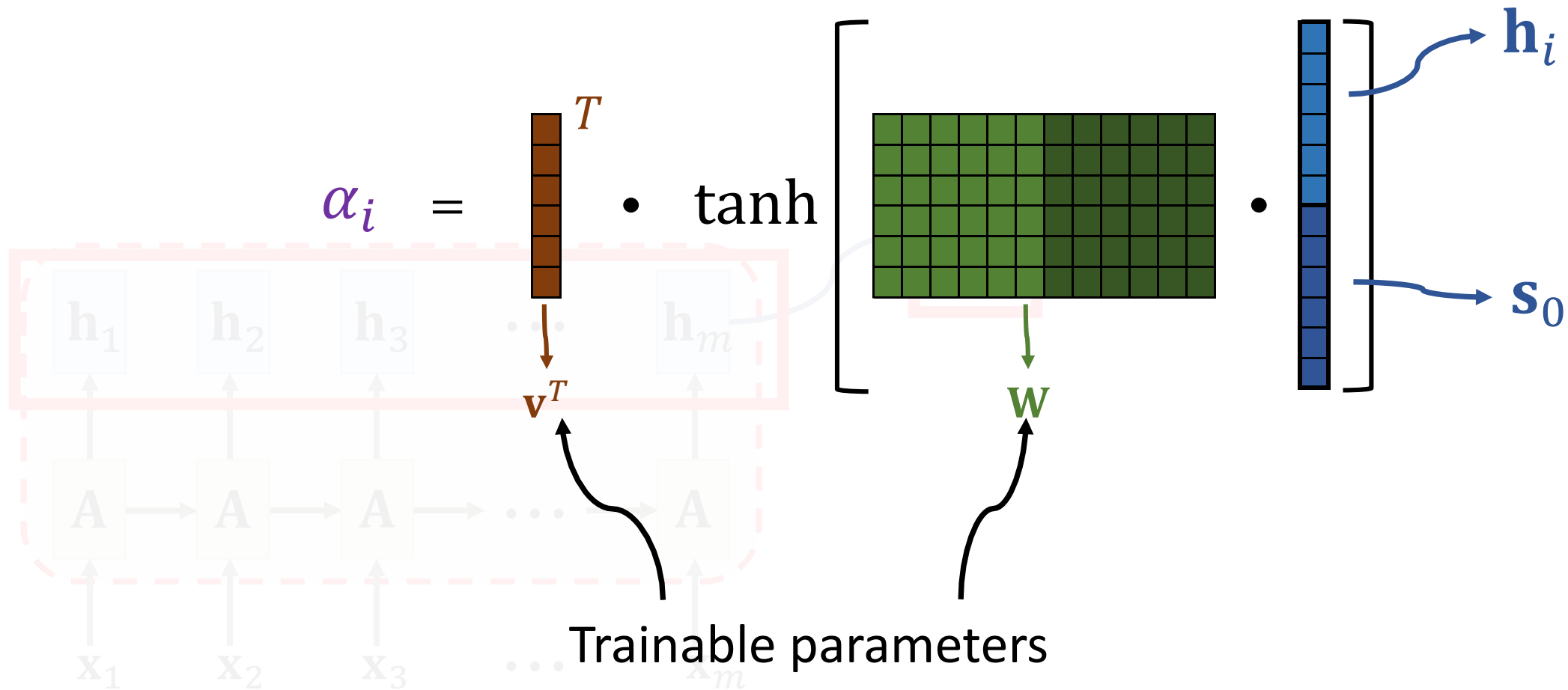
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_0)$



SimpleRNN + Attention

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

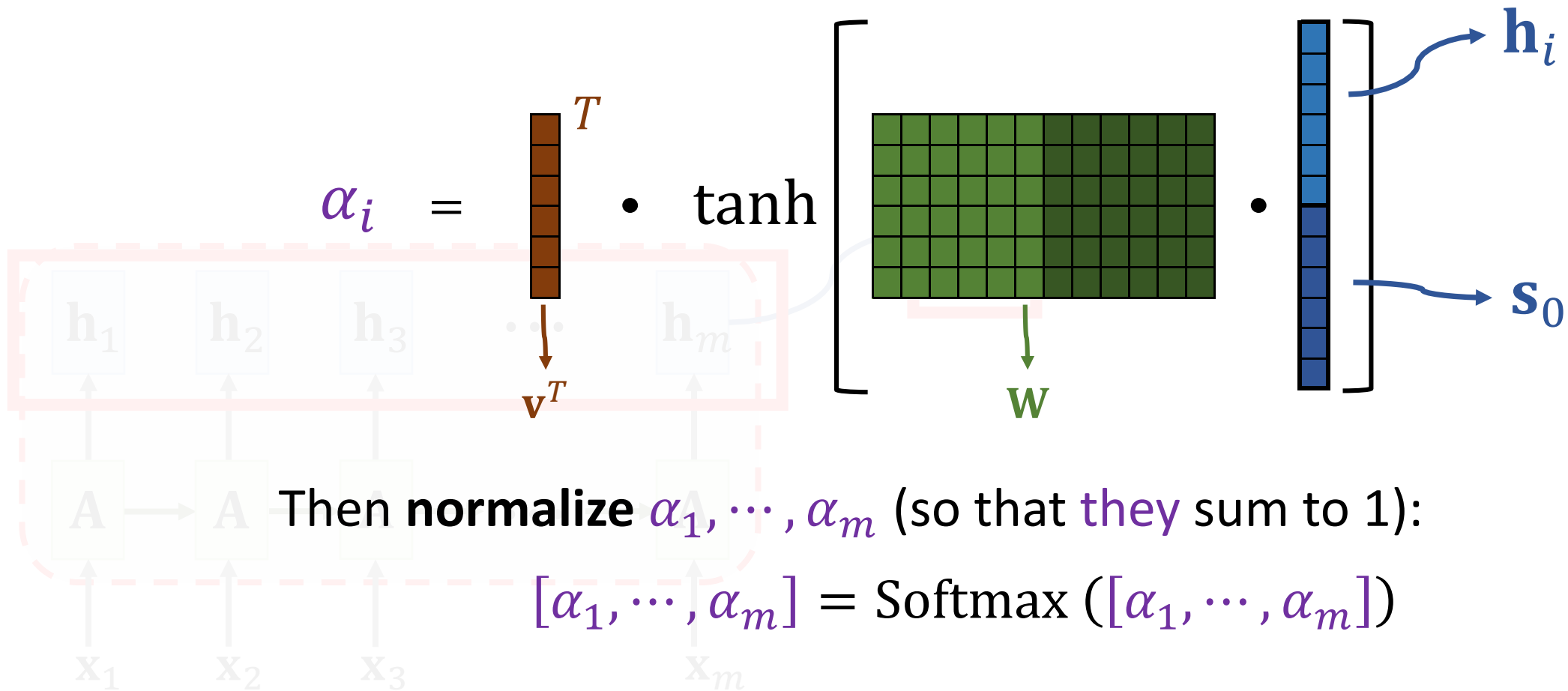
One option (used in the original paper):



SimpleRNN + Attention

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

One option (used in the original paper):



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:

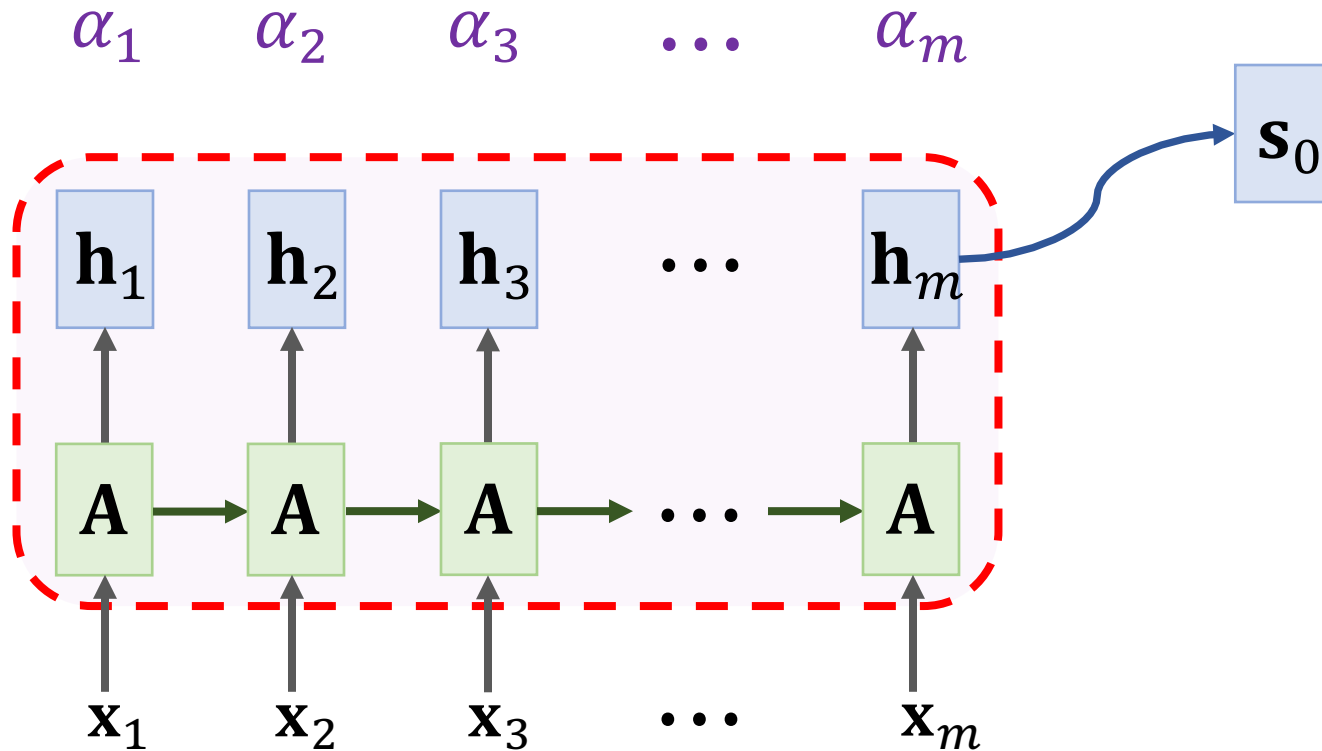
- $\alpha_i = \tilde{\mathbf{h}}_i^T \cdot \tilde{\mathbf{s}}_0$.

3. Normalization:

- $[\alpha_1, \dots, \alpha_m] = \text{Softmax}([\alpha_1, \dots, \alpha_m])$

SimpleRNN + Attention

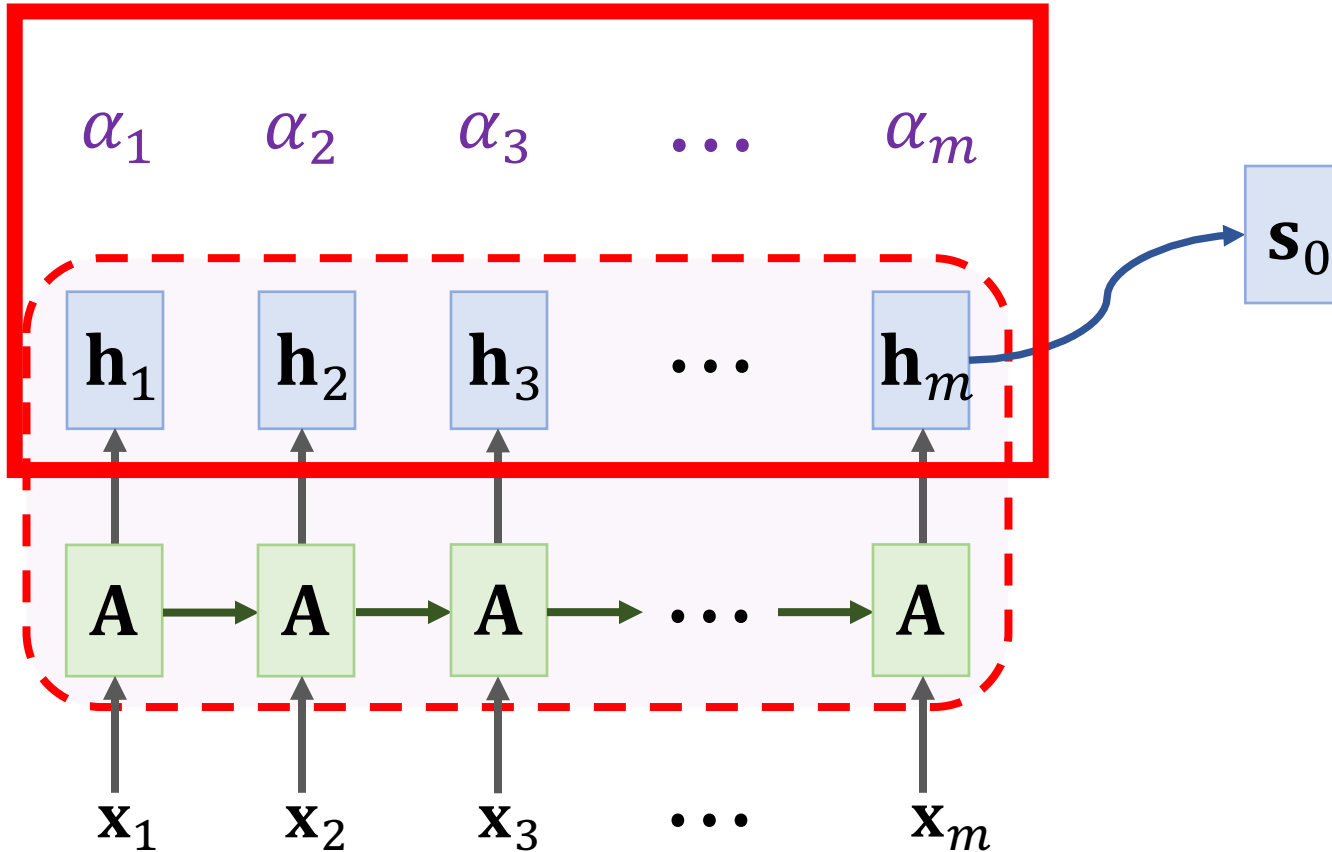
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_0)$



SimpleRNN + Attention

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

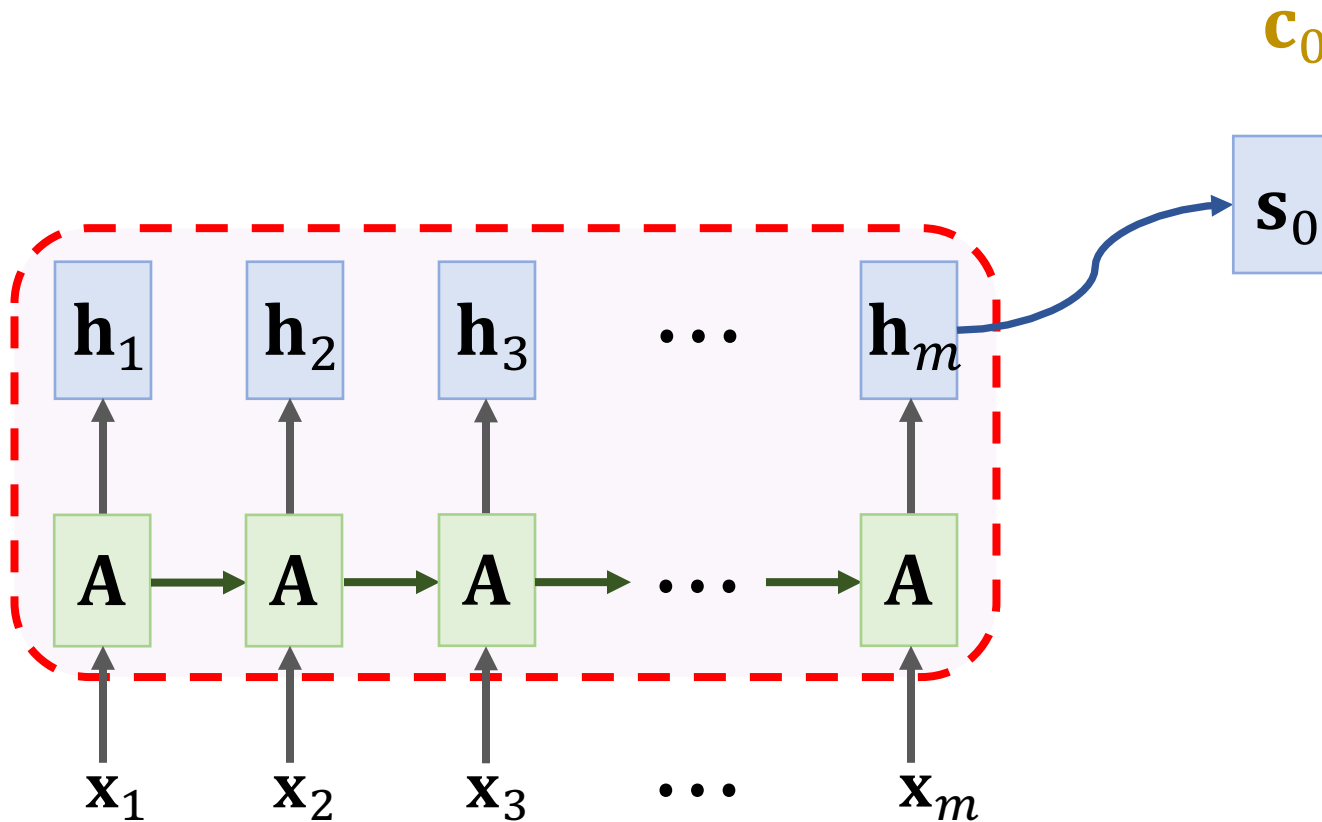
Context vector: $\mathbf{c}_0 = \alpha_1 \mathbf{h}_1 + \dots + \alpha_m \mathbf{h}_m.$



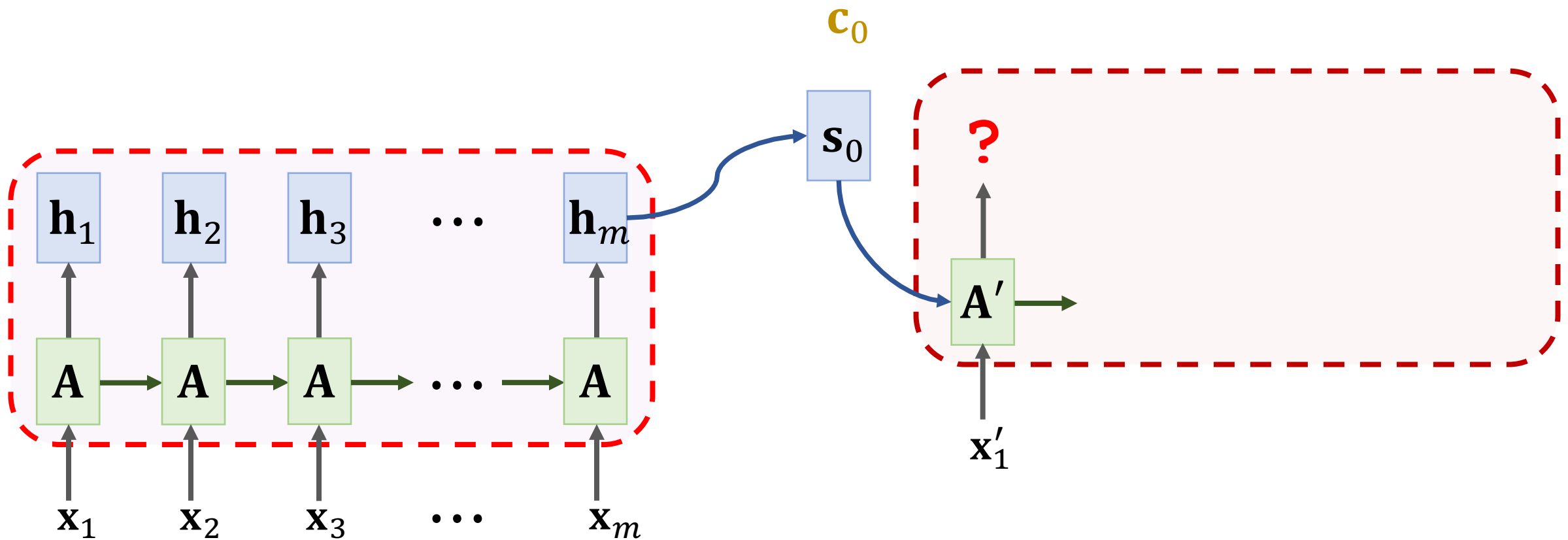
SimpleRNN + Attention

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

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



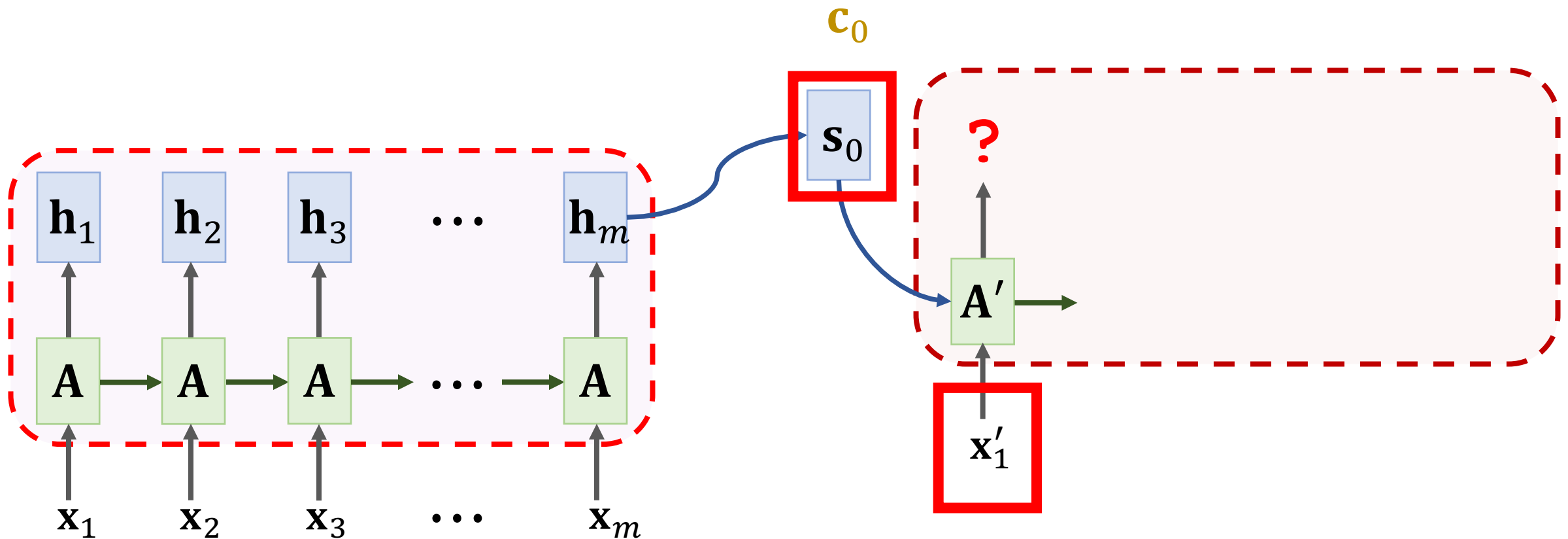
SimpleRNN + Attention



SimpleRNN + Attention

SimpleRNN:

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



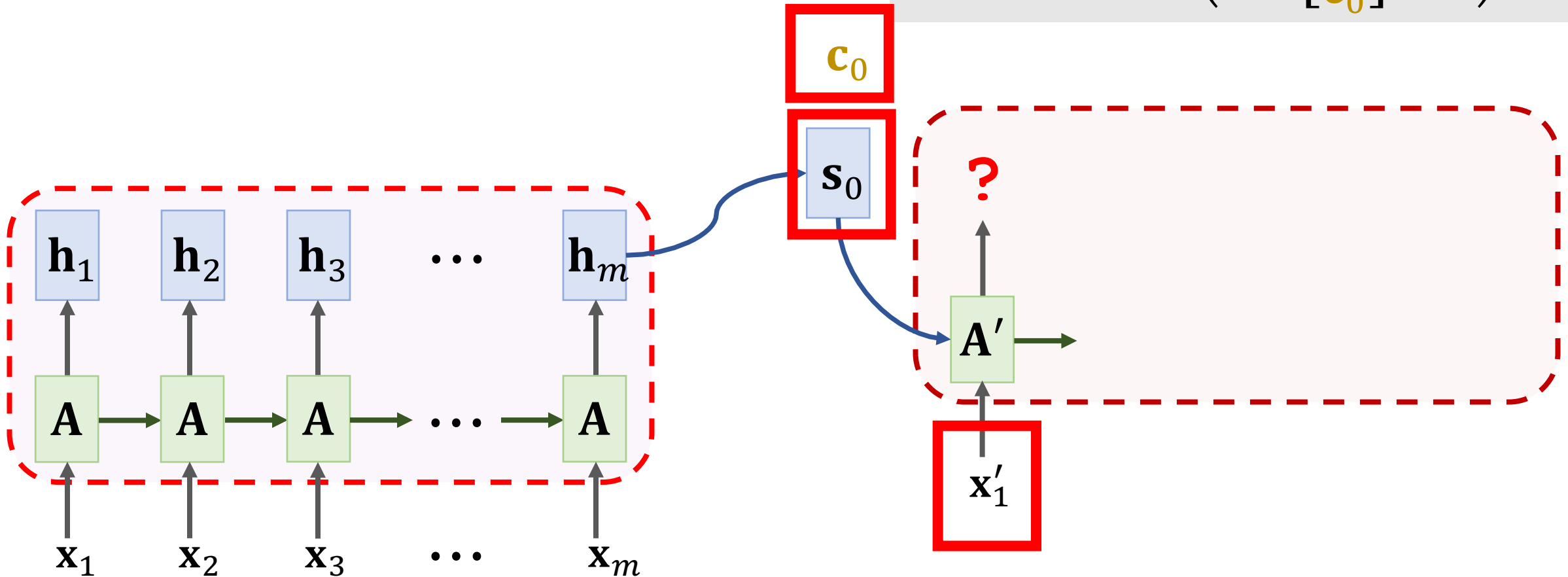
SimpleRNN + Attention

SimpleRNN:

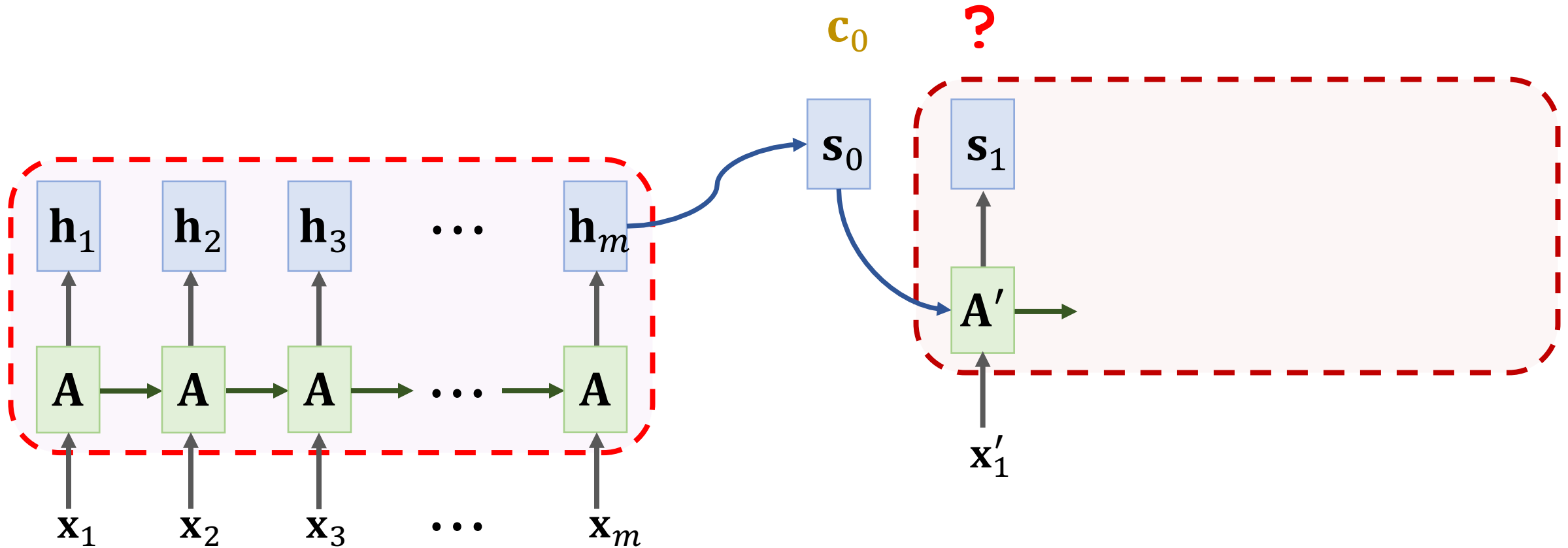
$$\mathbf{s}_1 = \tanh \left(\mathbf{A}' \cdot \begin{bmatrix} \mathbf{x}'_1 \\ \mathbf{s}_0 \end{bmatrix} + \mathbf{b} \right)$$

SimpleRNN + Attention:

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

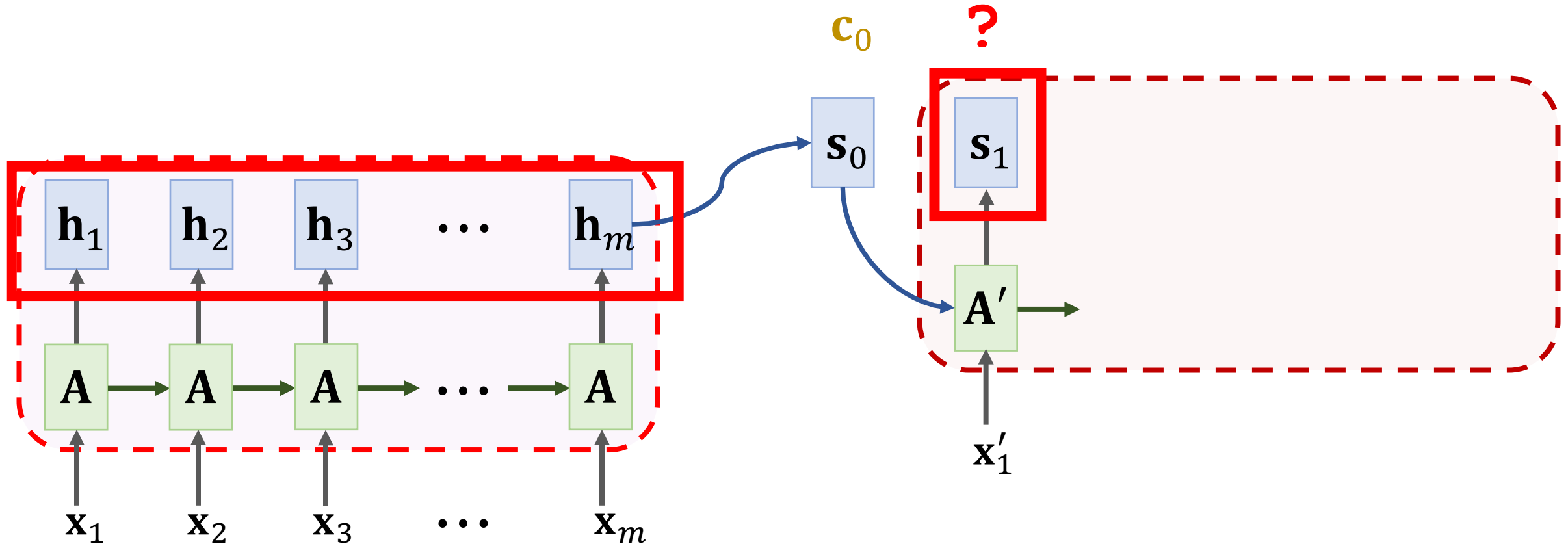


SimpleRNN + Attention



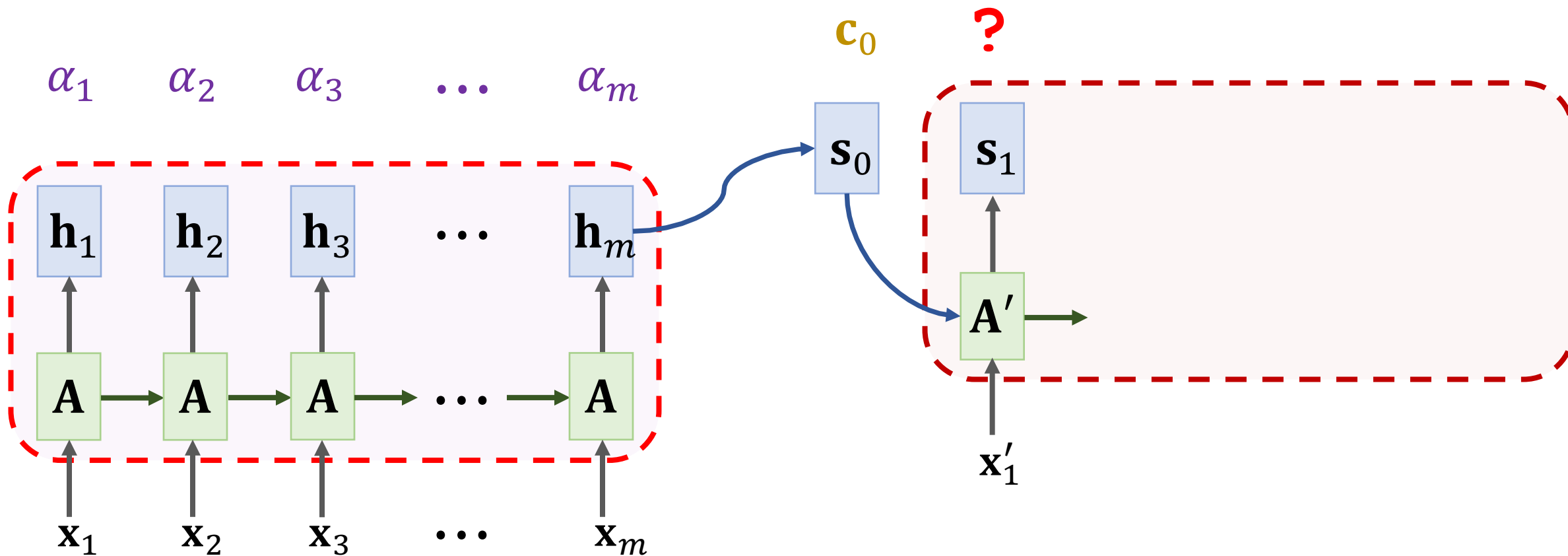
SimpleRNN + Attention

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



SimpleRNN + Attention

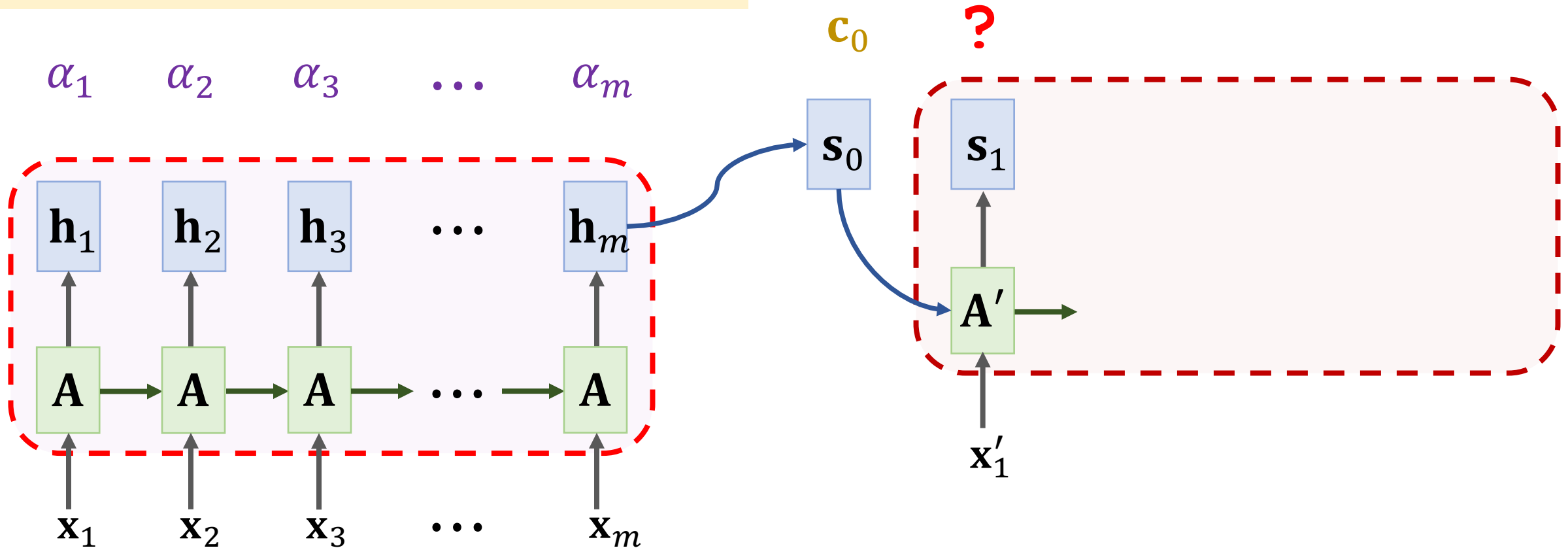
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_1)$



SimpleRNN + Attention

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

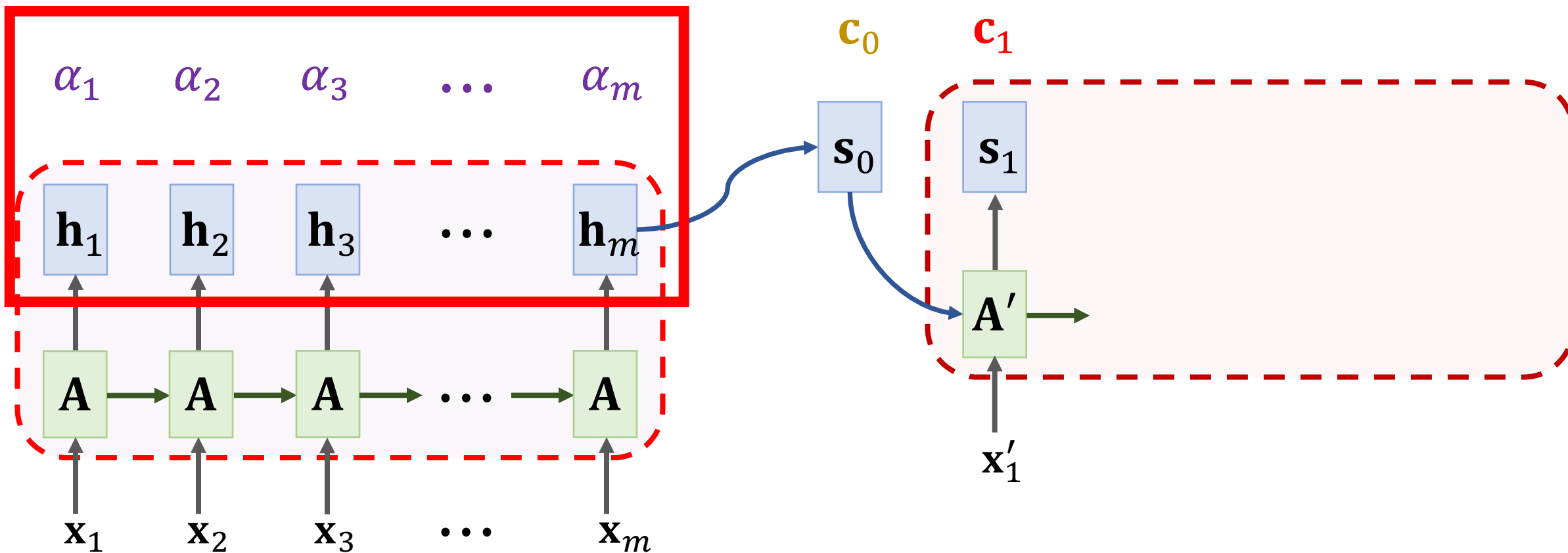
Do not re-use the α 's computed previously.



SimpleRNN + Attention

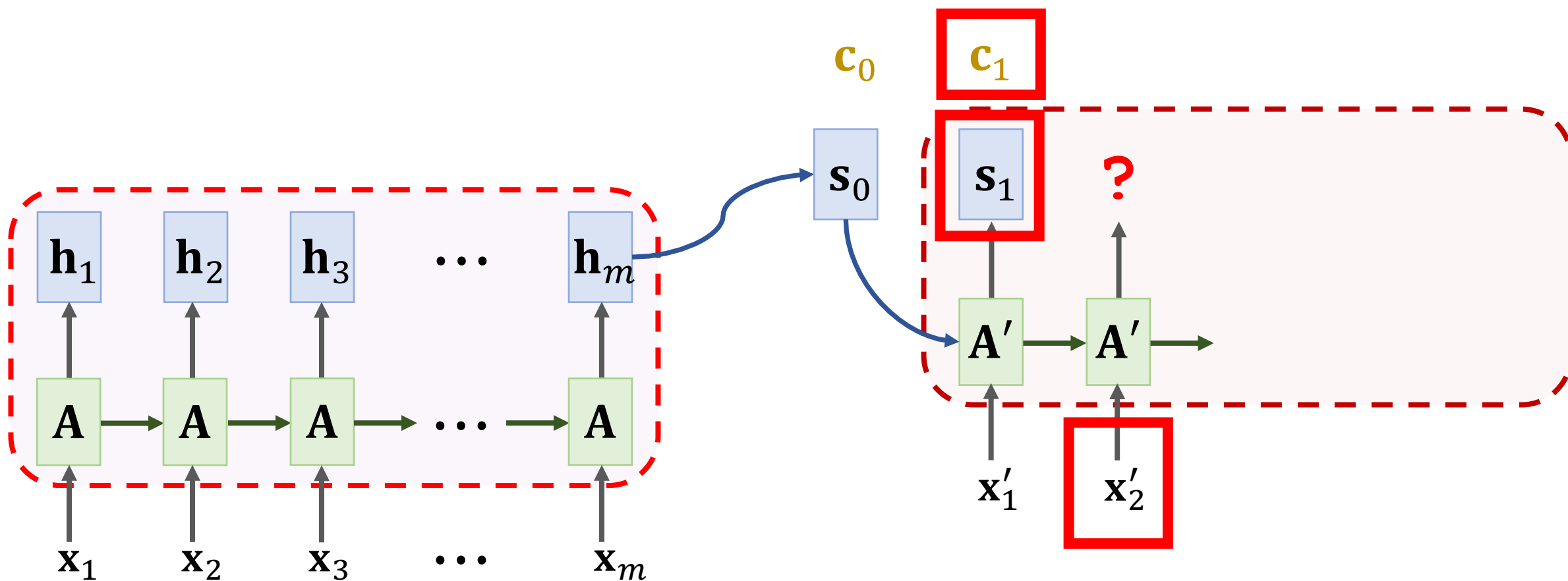
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_1)$

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

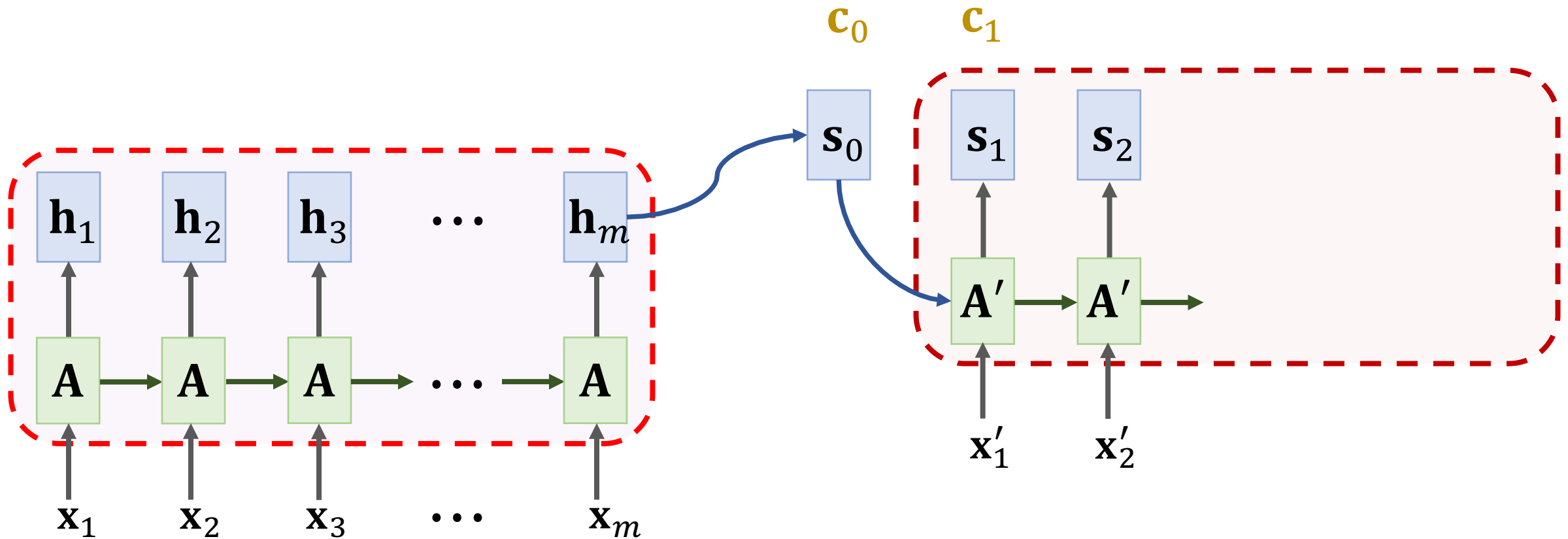


SimpleRNN + Attention

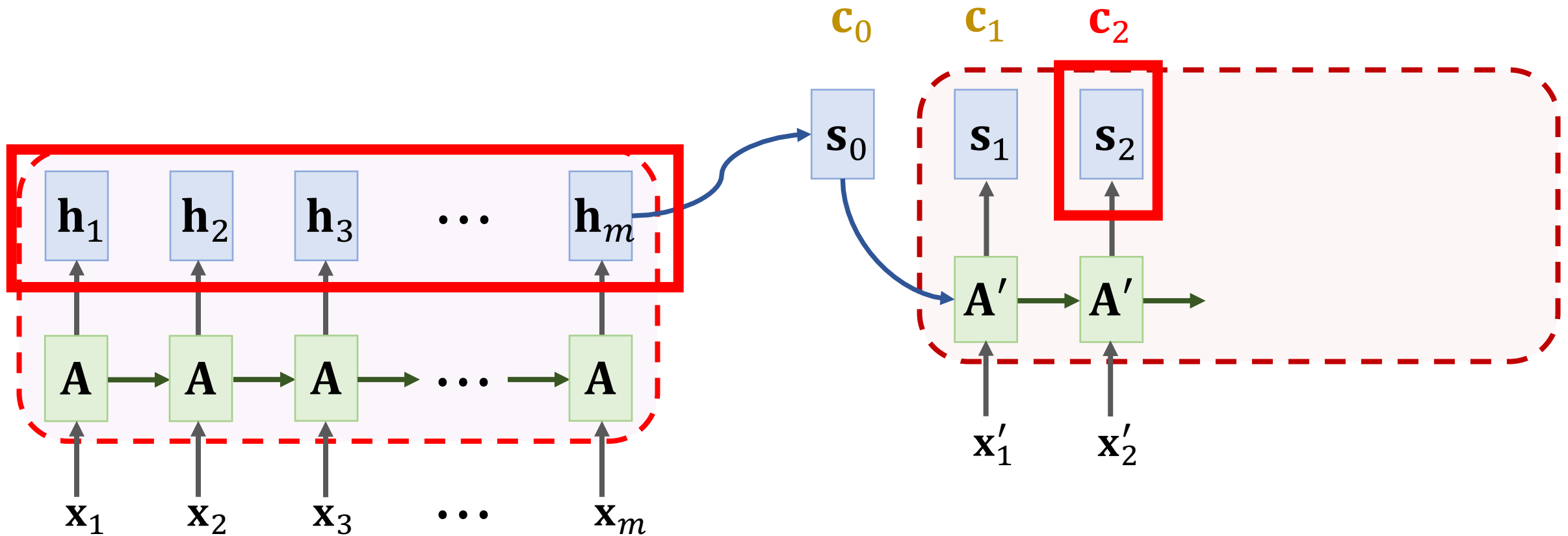
$$\mathbf{s}_2 = \tanh \left(\mathbf{A}' \cdot \begin{bmatrix} \mathbf{x}'_2 \\ \mathbf{s}_1 \\ \mathbf{c}_1 \end{bmatrix} + \mathbf{b} \right)$$



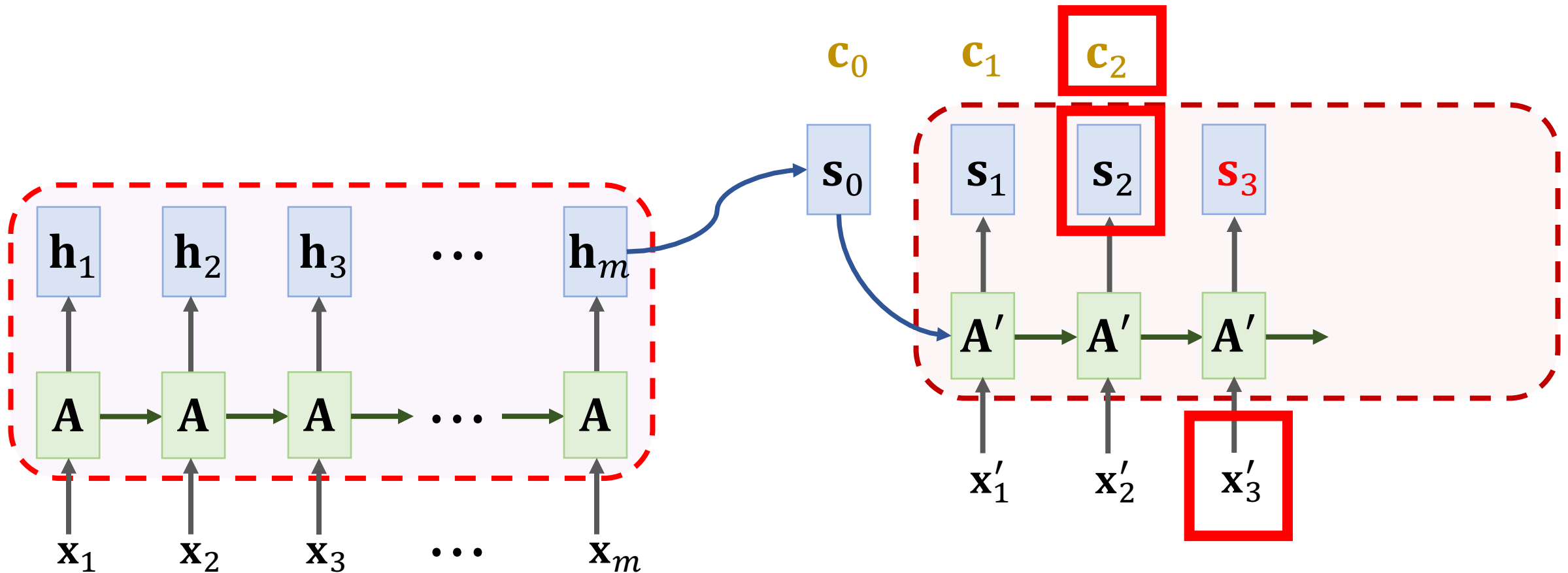
SimpleRNN + Attention



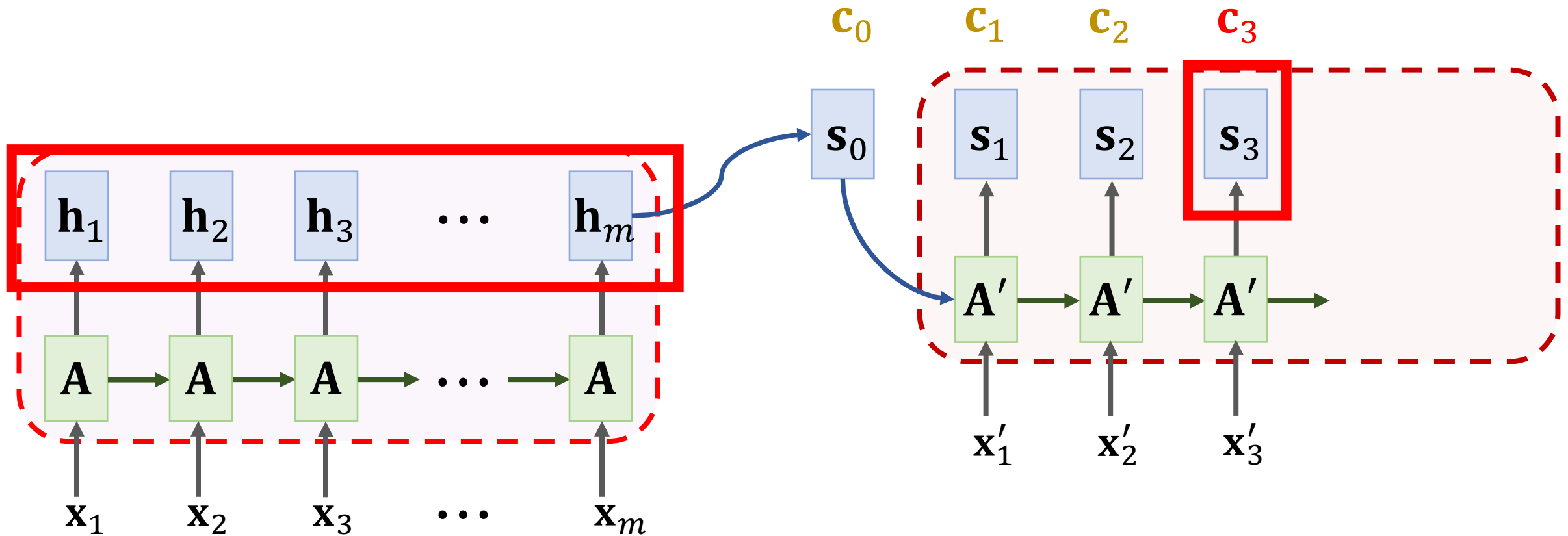
SimpleRNN + Attention



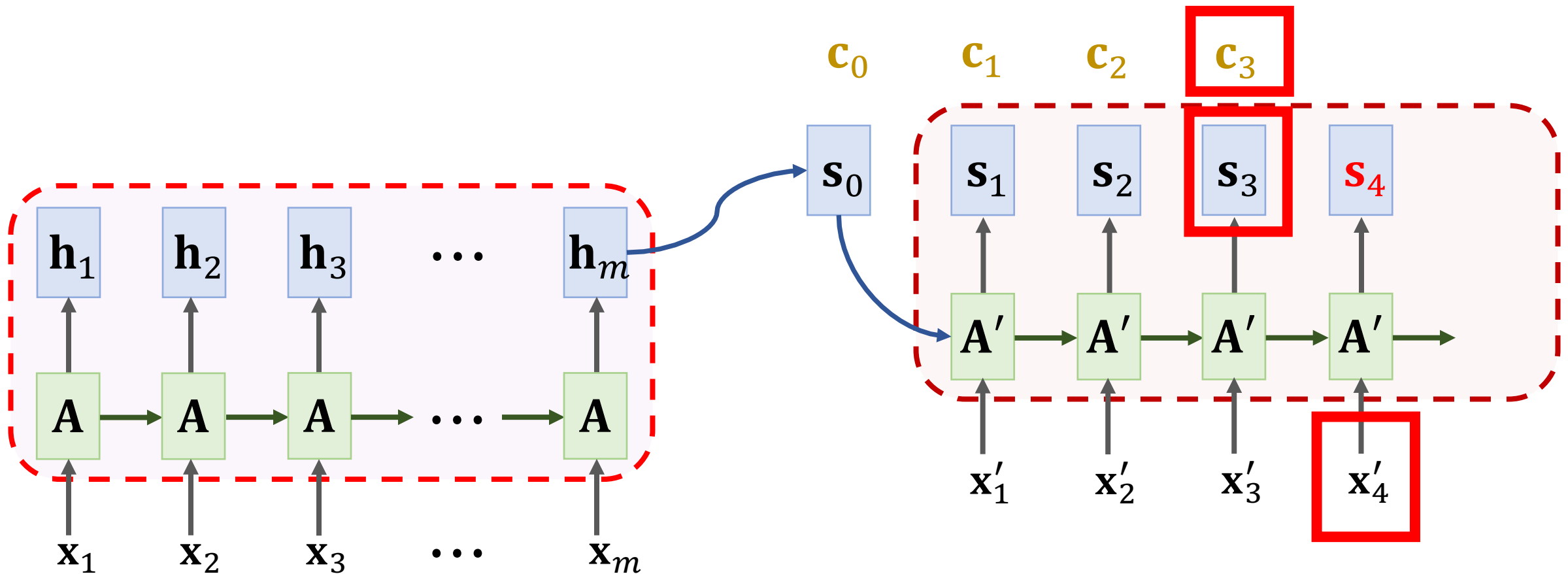
SimpleRNN + Attention



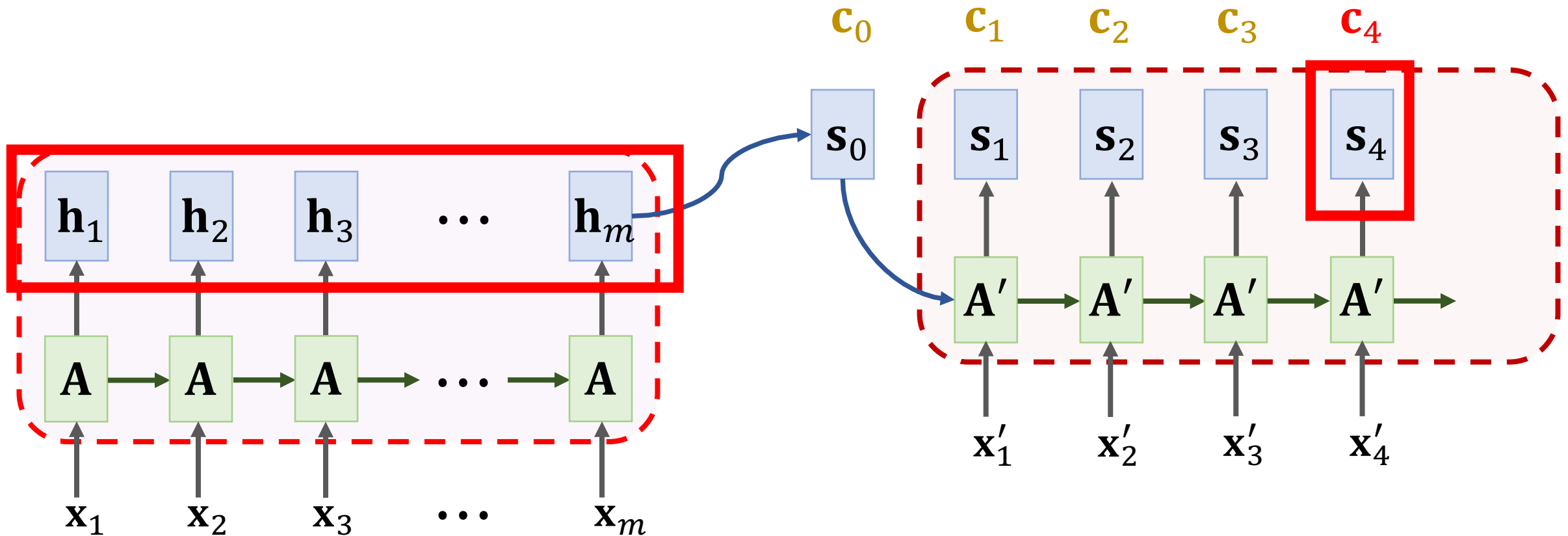
SimpleRNN + Attention



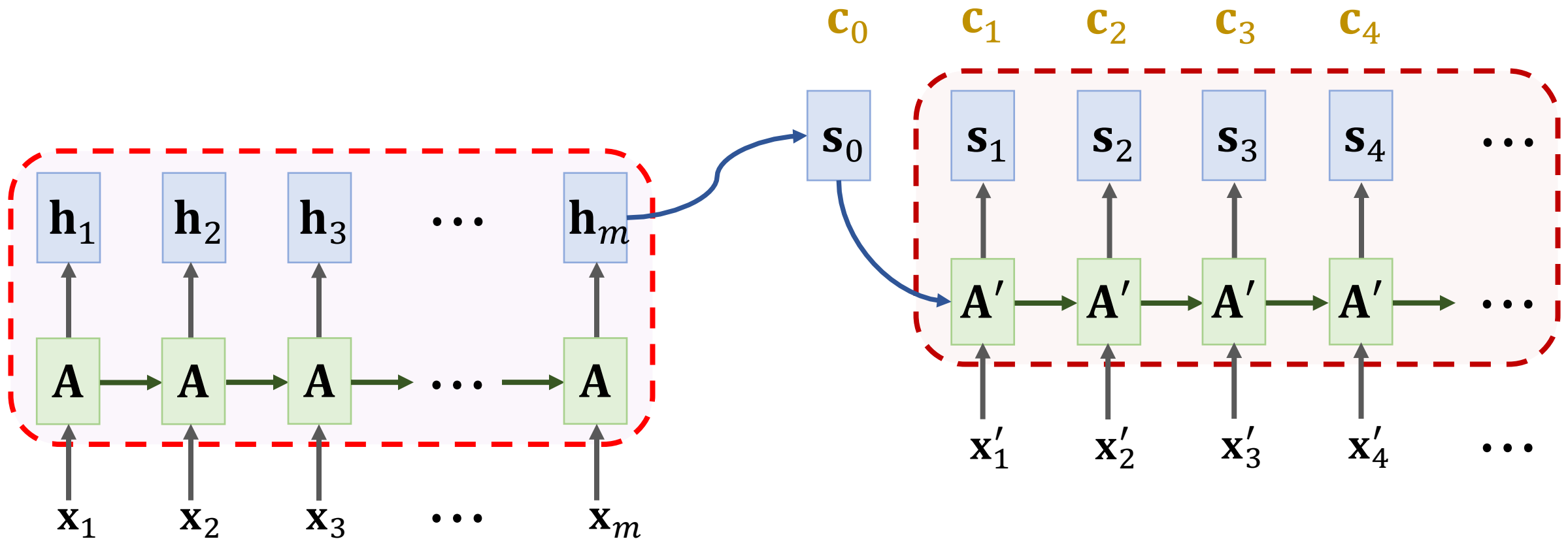
SimpleRNN + Attention



SimpleRNN + Attention



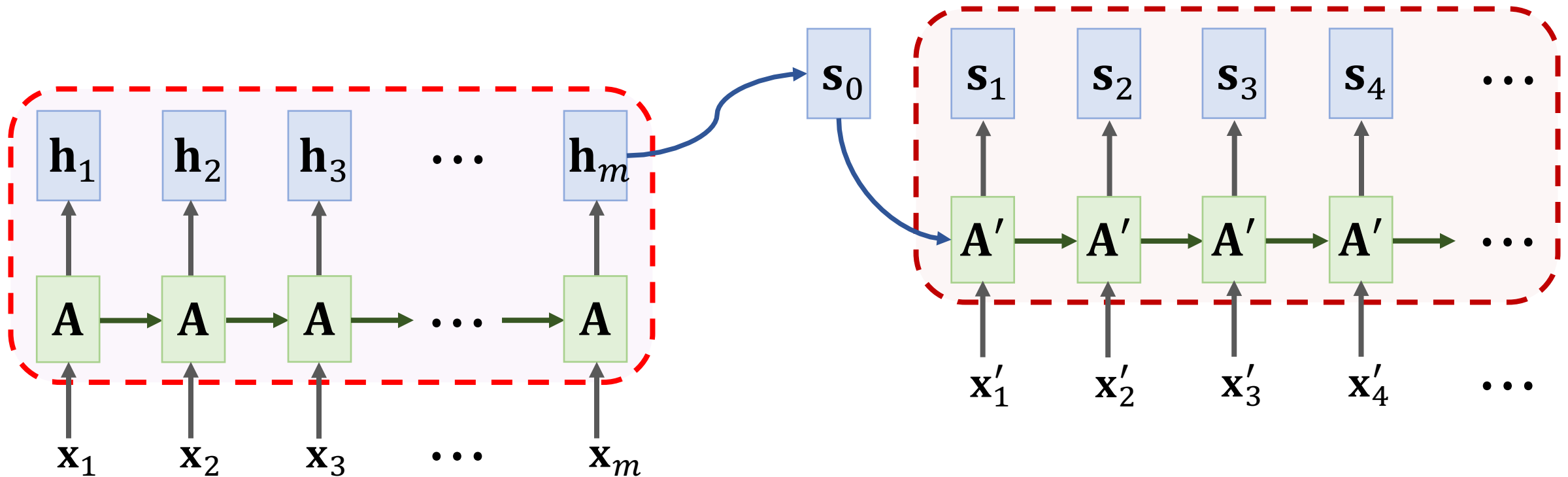
SimpleRNN + Attention



SimpleRNN + Attention

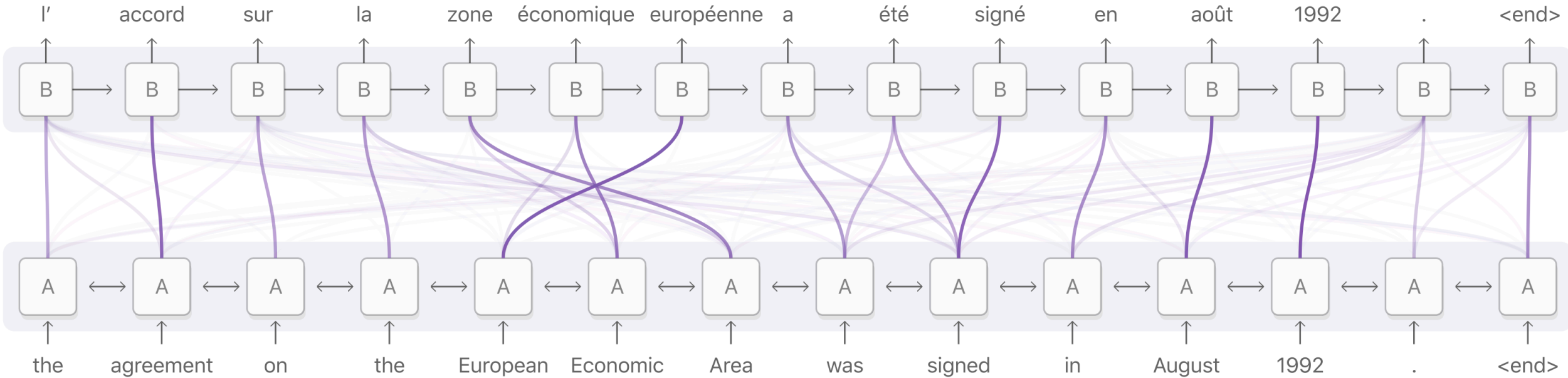
Question: How many weights have been computed?

- For every decoder state \mathbf{s}_t , there are m weights: $\alpha_1, \dots, \alpha_m$.
- If the decode has T states, then there are **totally mT states**.



Attention: Weights Visualization

Decoder RNN

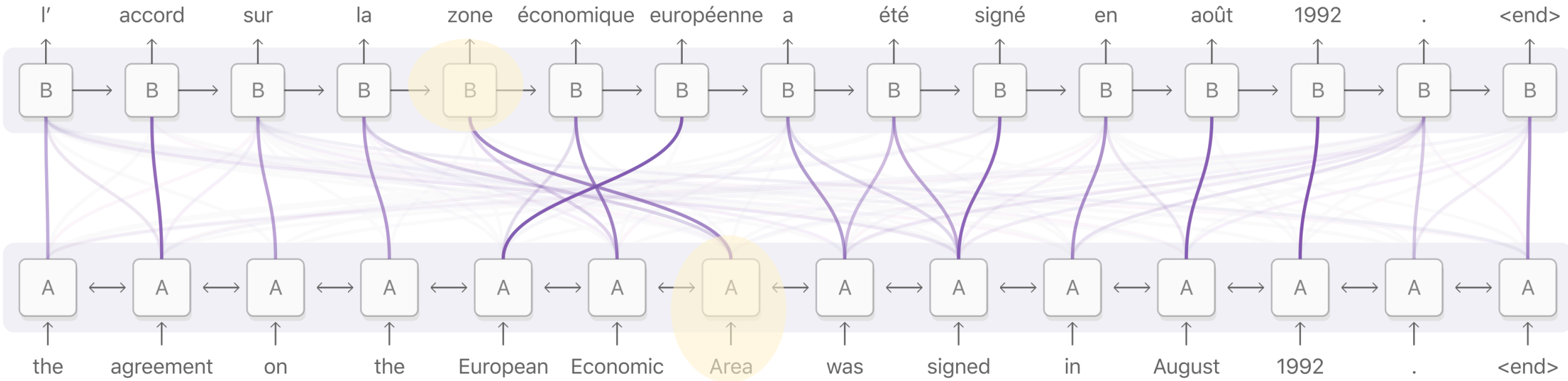


Encoder RNN

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

Attention: Weights Visualization

Decoder RNN

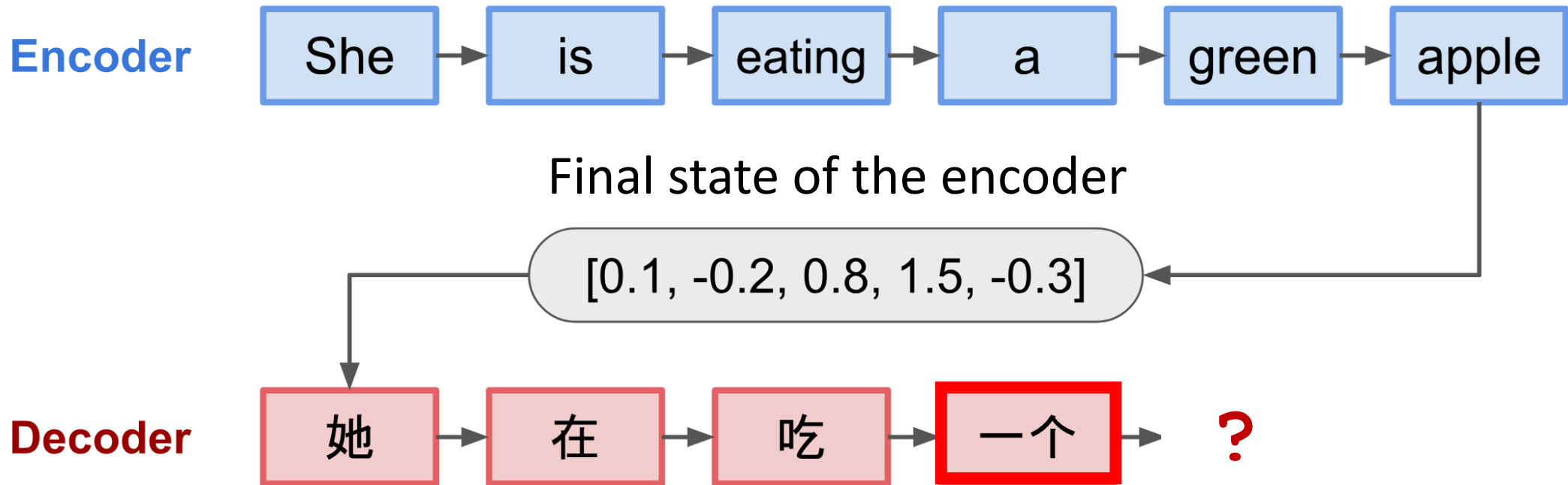


Encoder RNN

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

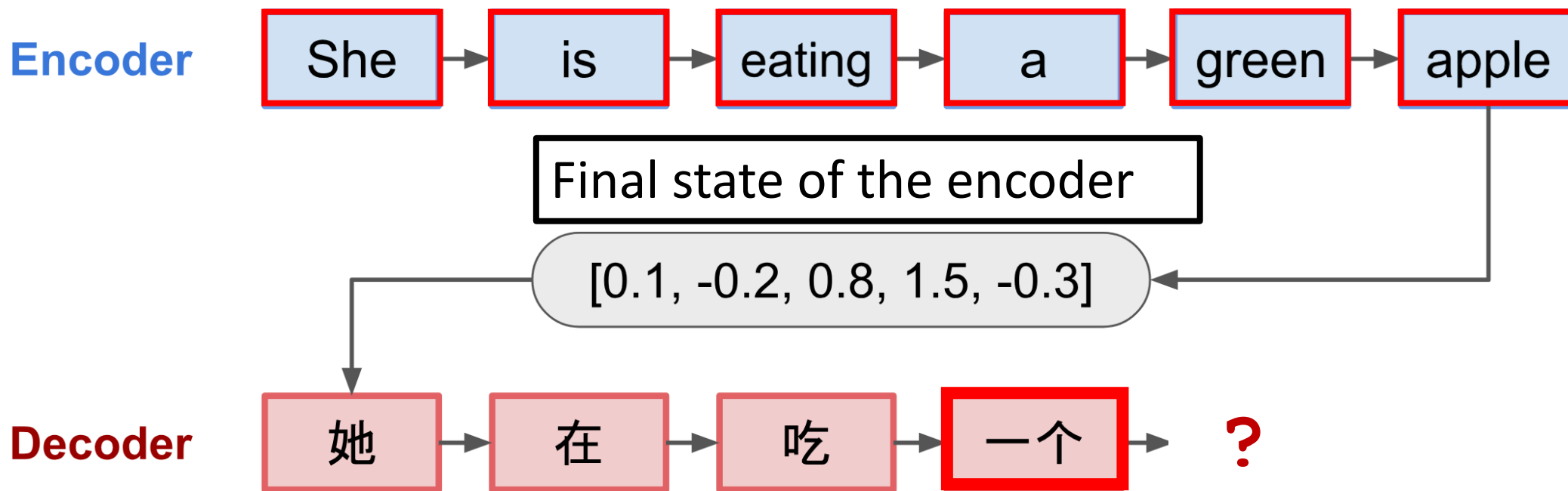
Summary

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



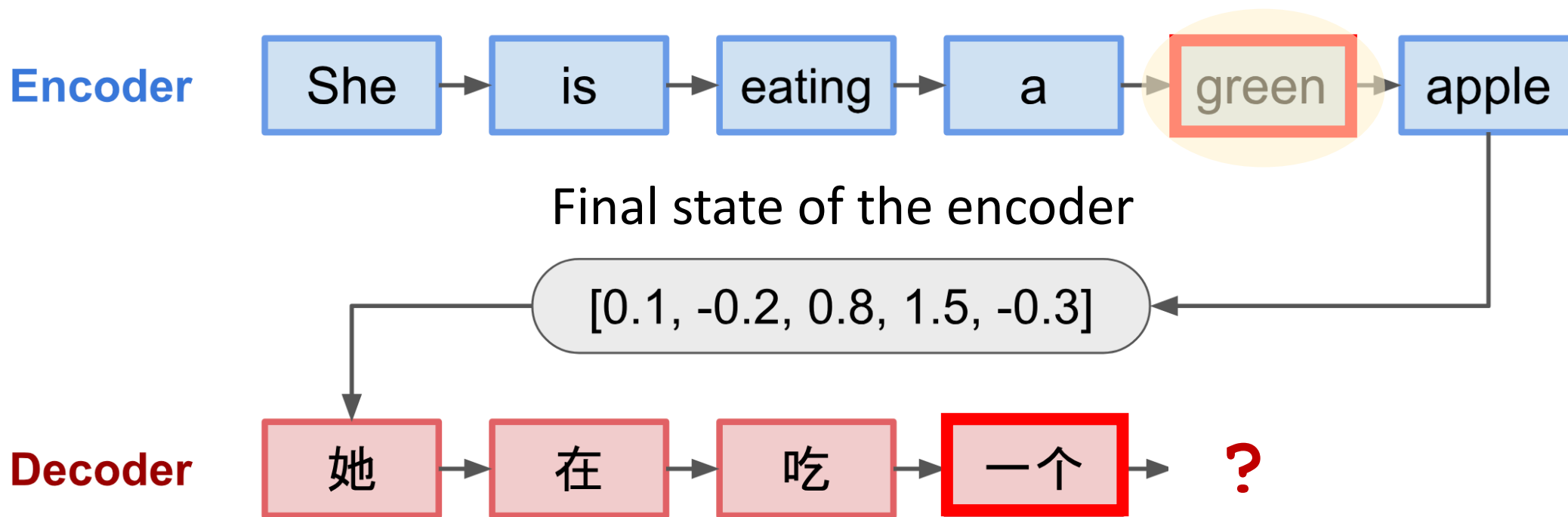
Summary

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



Summary

- Standard Seq2Seq model: the decoder looks at only its current state.
- Attention: decoder additionally looks at all the states of the encoder.
- Attention: decoder knows where to **focus** on.



Summary

- Standard Seq2Seq model: the decoder looks at only its current state.
- Attention: decoder additionally looks at all the states of the encoder.
- Attention: decoder knows where to focus on.
- **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

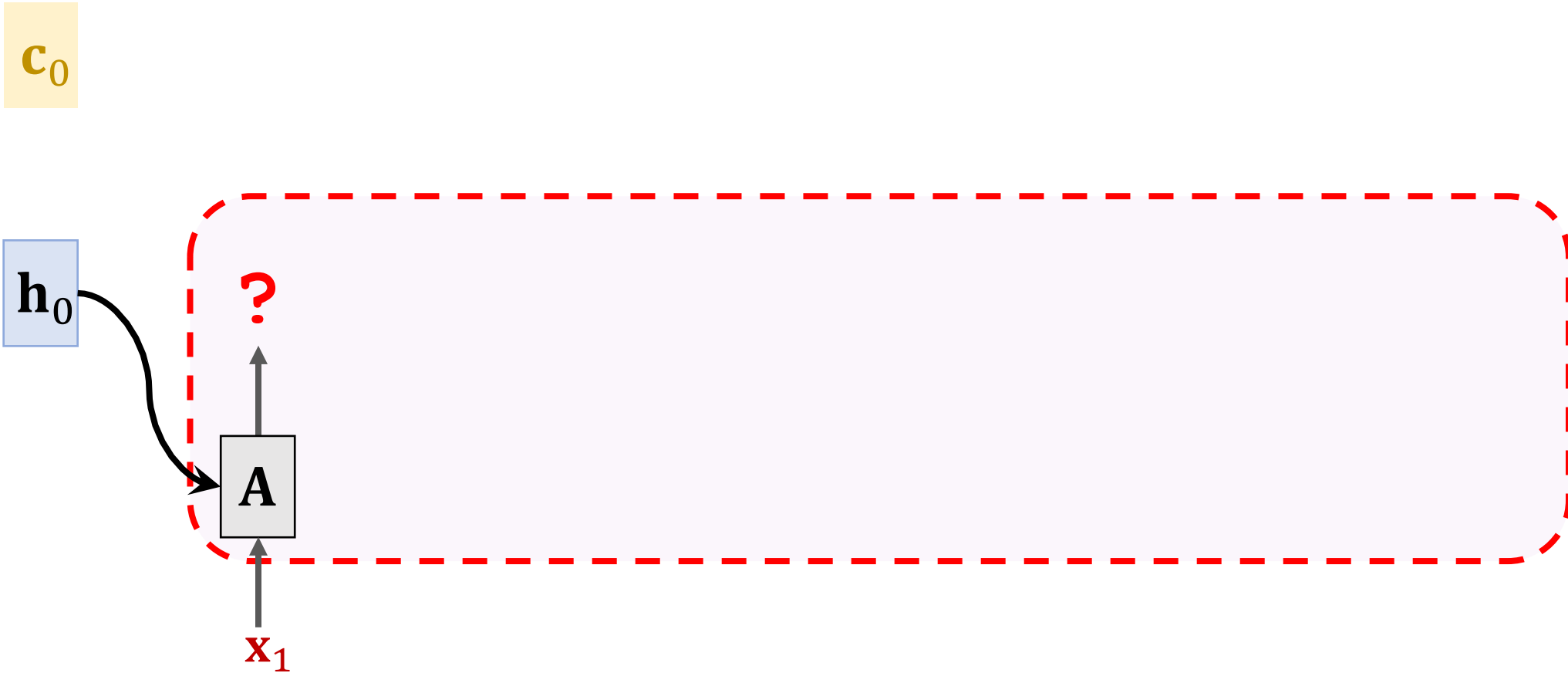
- The original paper uses LSTM.
- To make teaching easy, I replace LSTM by SimpleRNN.

SimpleRNN + Self-Attention

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

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


SimpleRNN + Self-Attention



SimpleRNN + Self-Attention

SimpleRNN:

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

\mathbf{c}_0



SimpleRNN + Self-Attention

SimpleRNN:

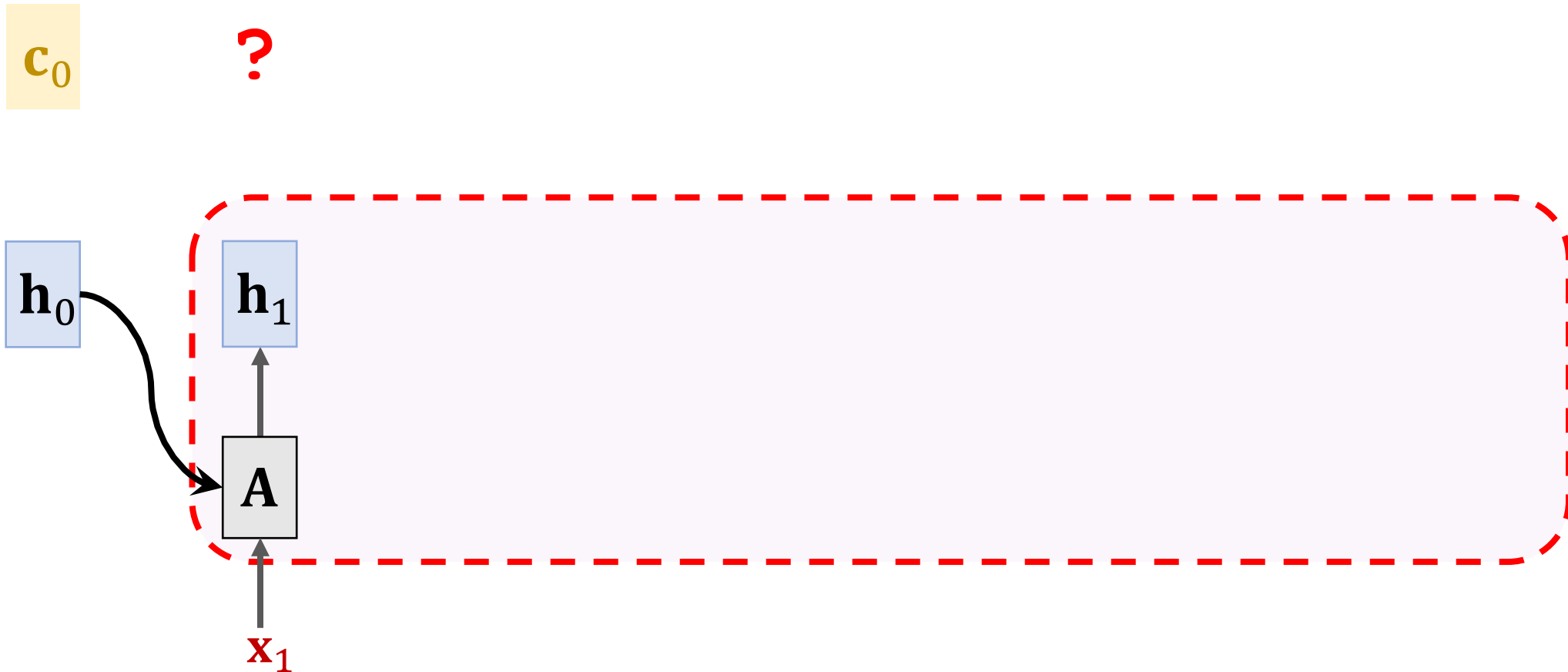
$$\mathbf{h}_1 = \tanh \left(\mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{h}_0 \end{bmatrix} + \mathbf{b} \right)$$

SimpleRNN + Self-Attention:

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

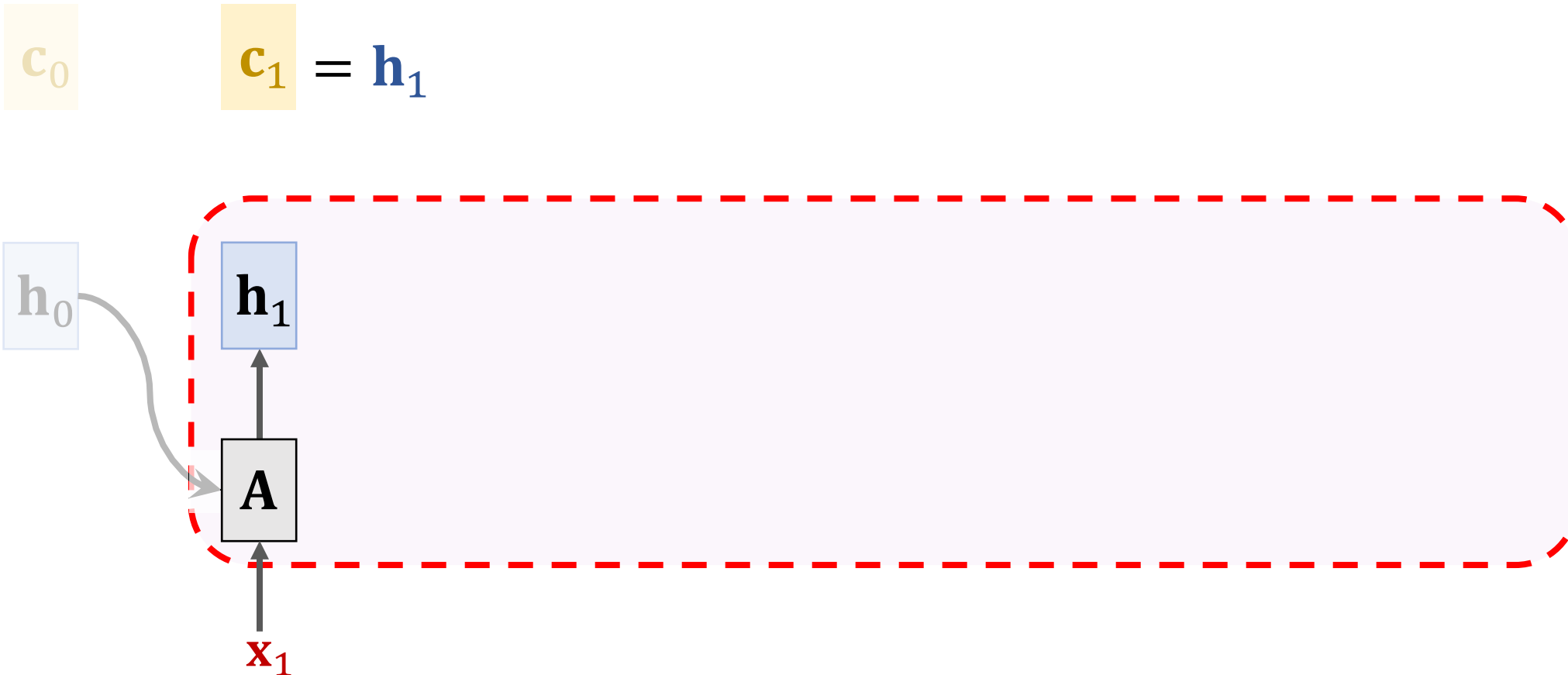


SimpleRNN + Self-Attention



SimpleRNN + Self-Attention

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

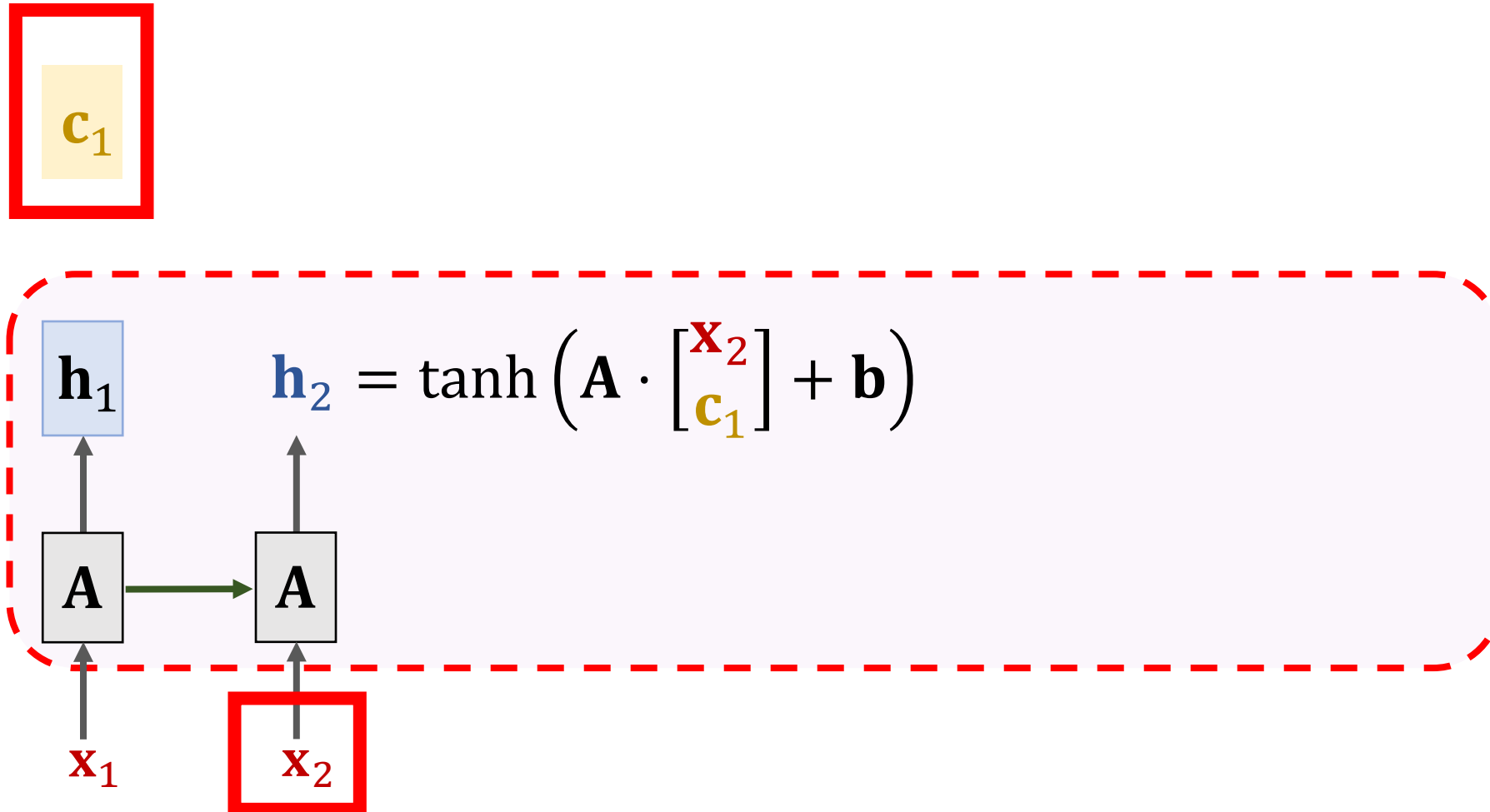


SimpleRNN + Self-Attention

c_1



SimpleRNN + Self-Attention



SimpleRNN + Self-Attention

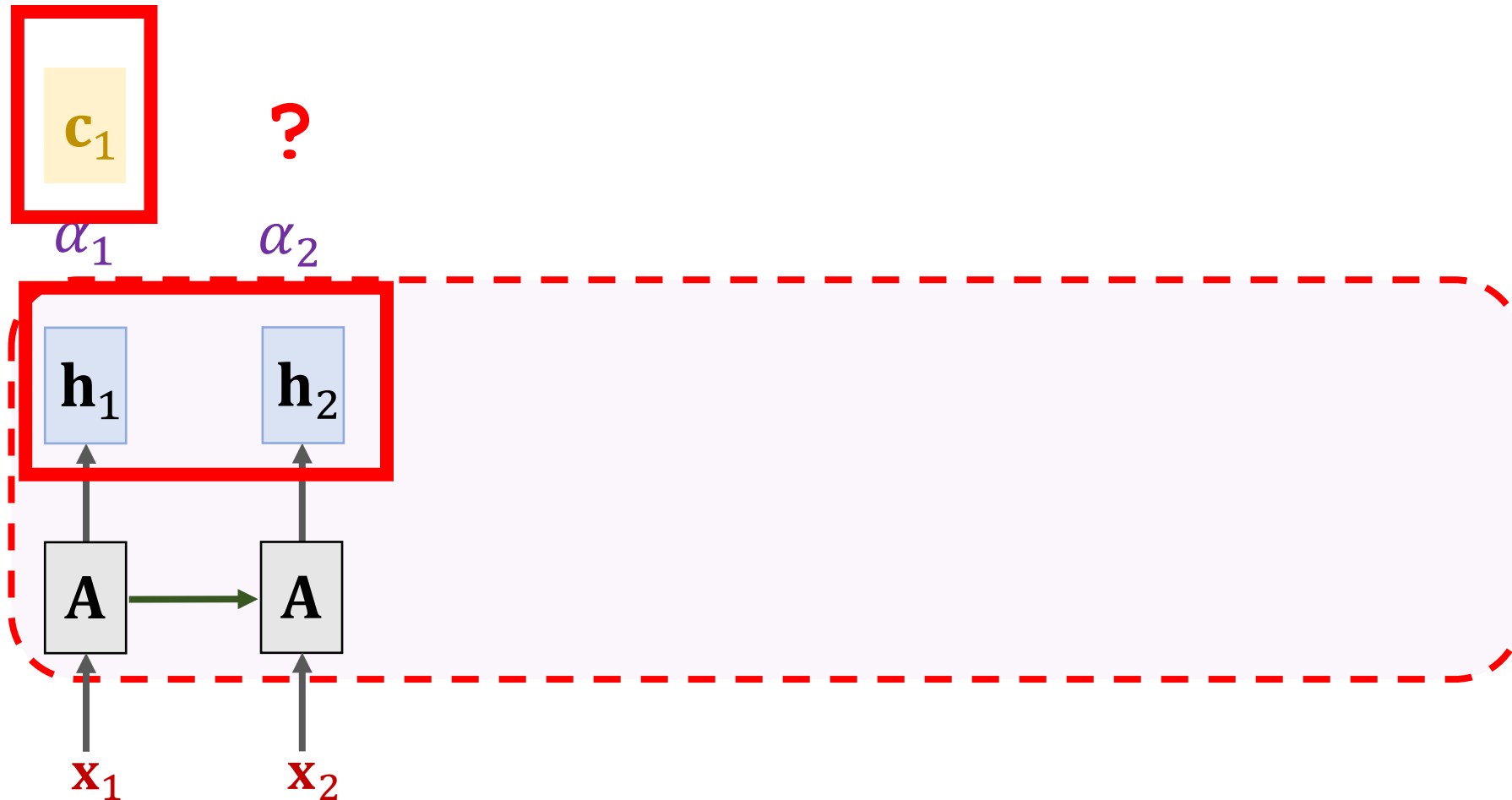
c_1

?



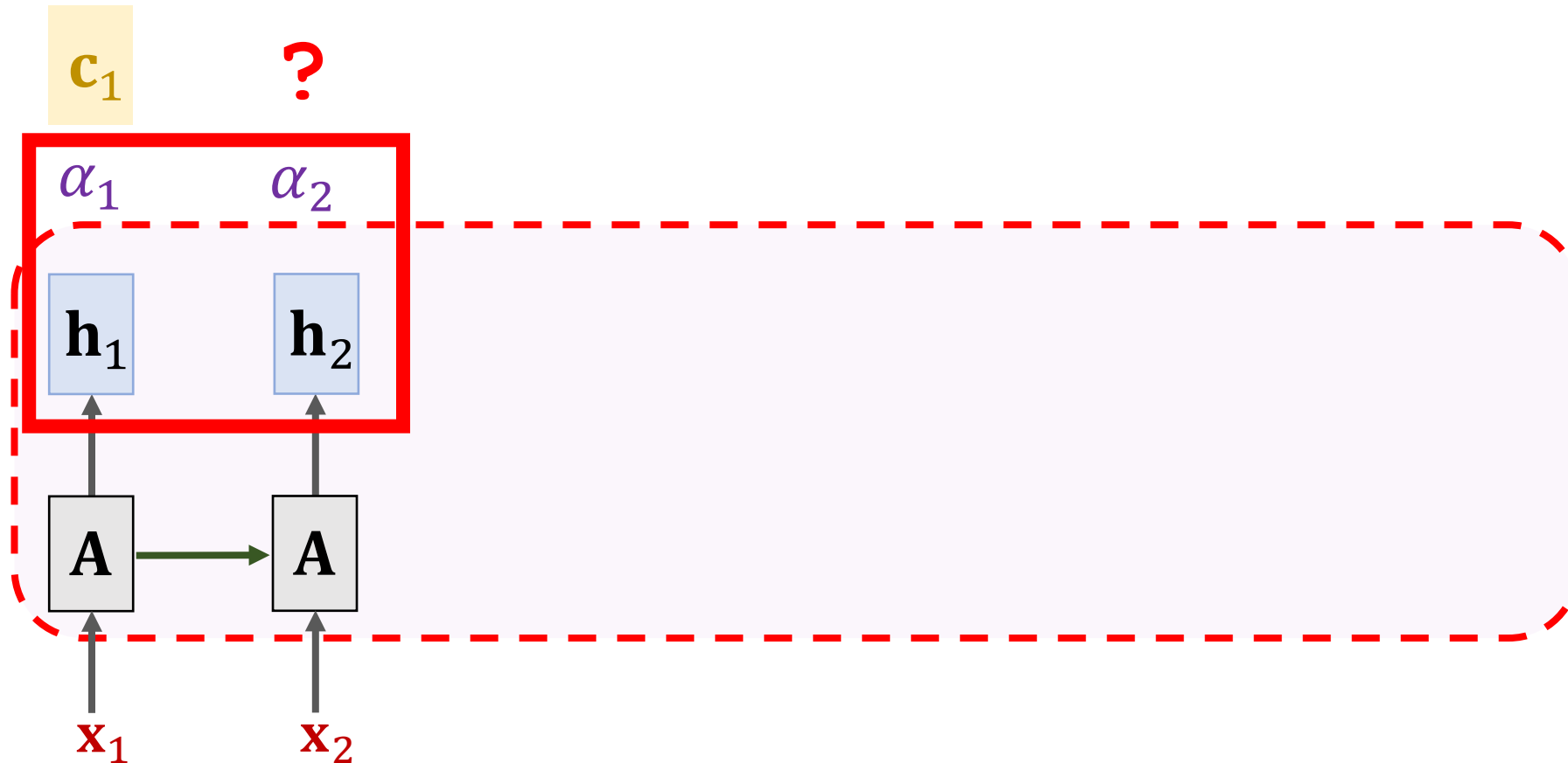
SimpleRNN + Self-Attention

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



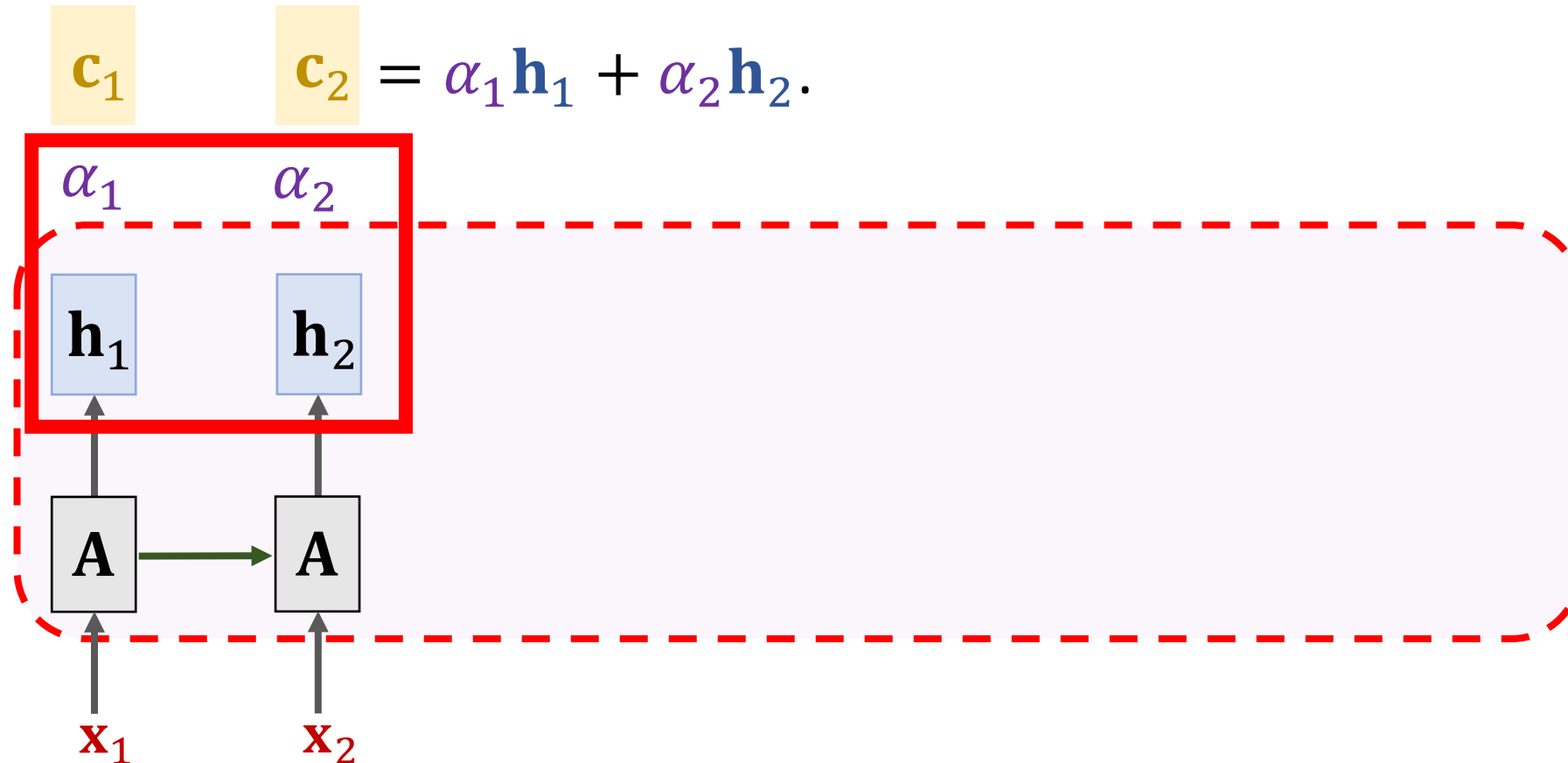
SimpleRNN + Self-Attention

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

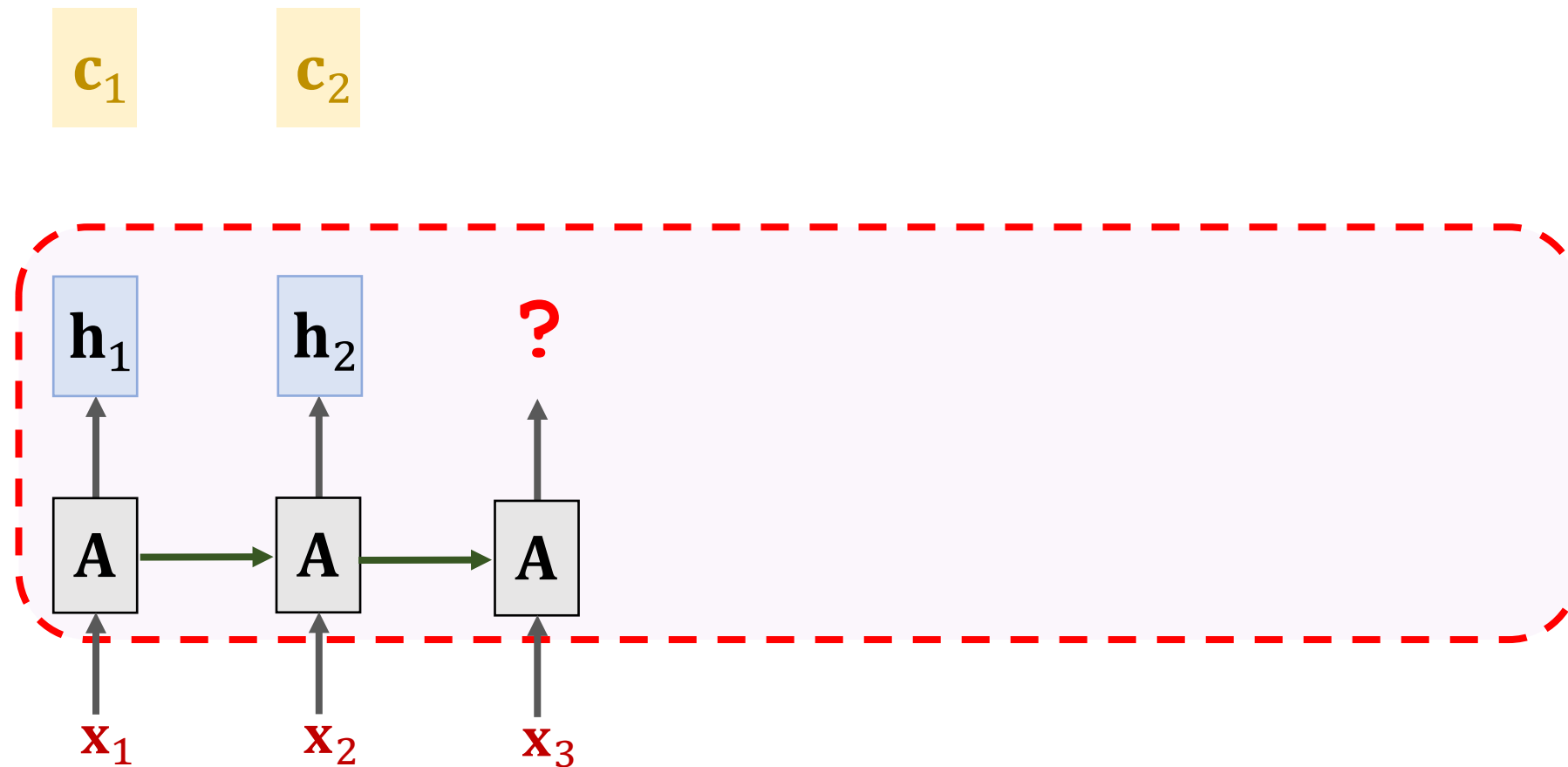


SimpleRNN + Self-Attention

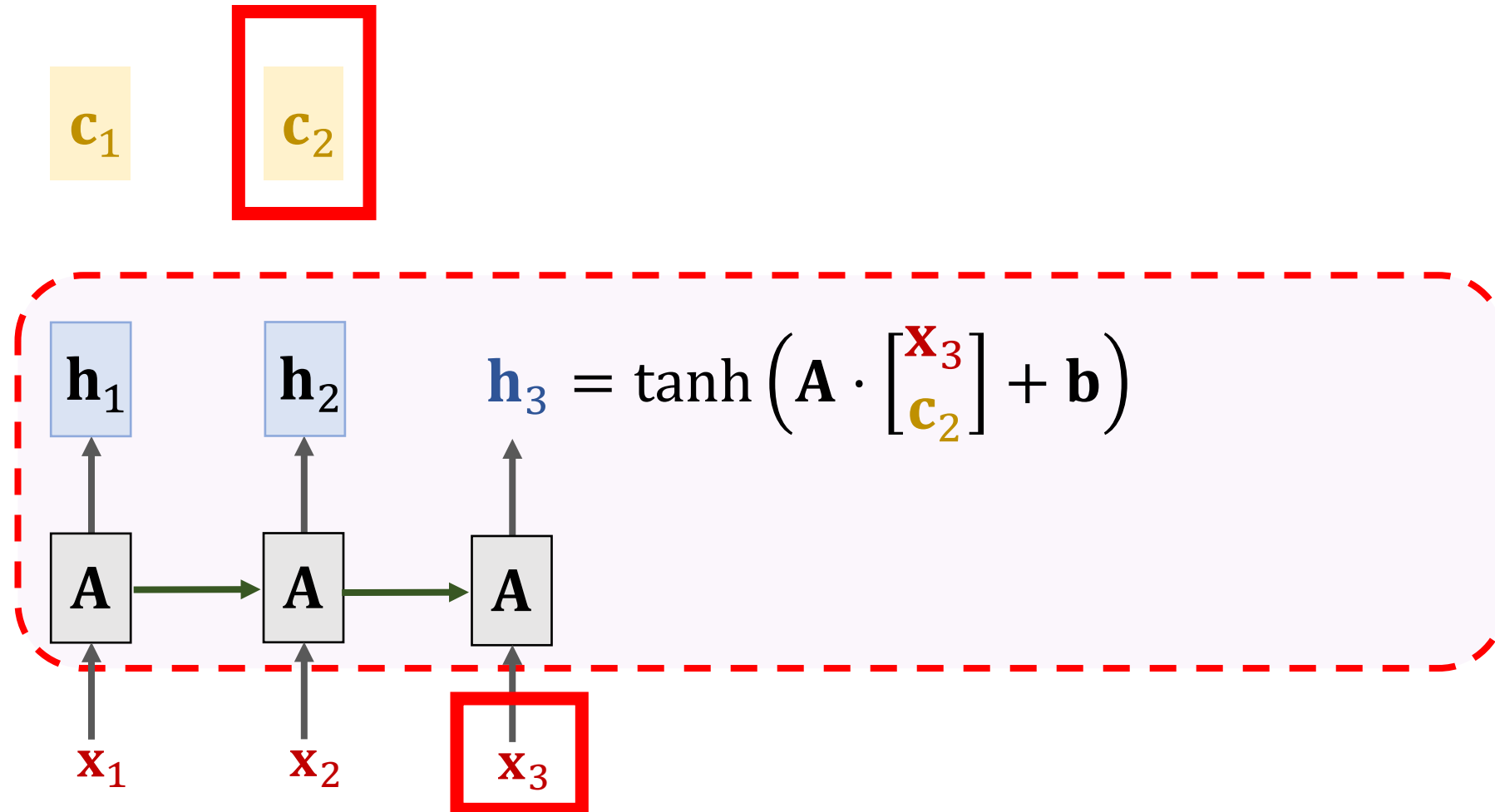
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{c}_1)$



SimpleRNN + Self-Attention



SimpleRNN + Self-Attention

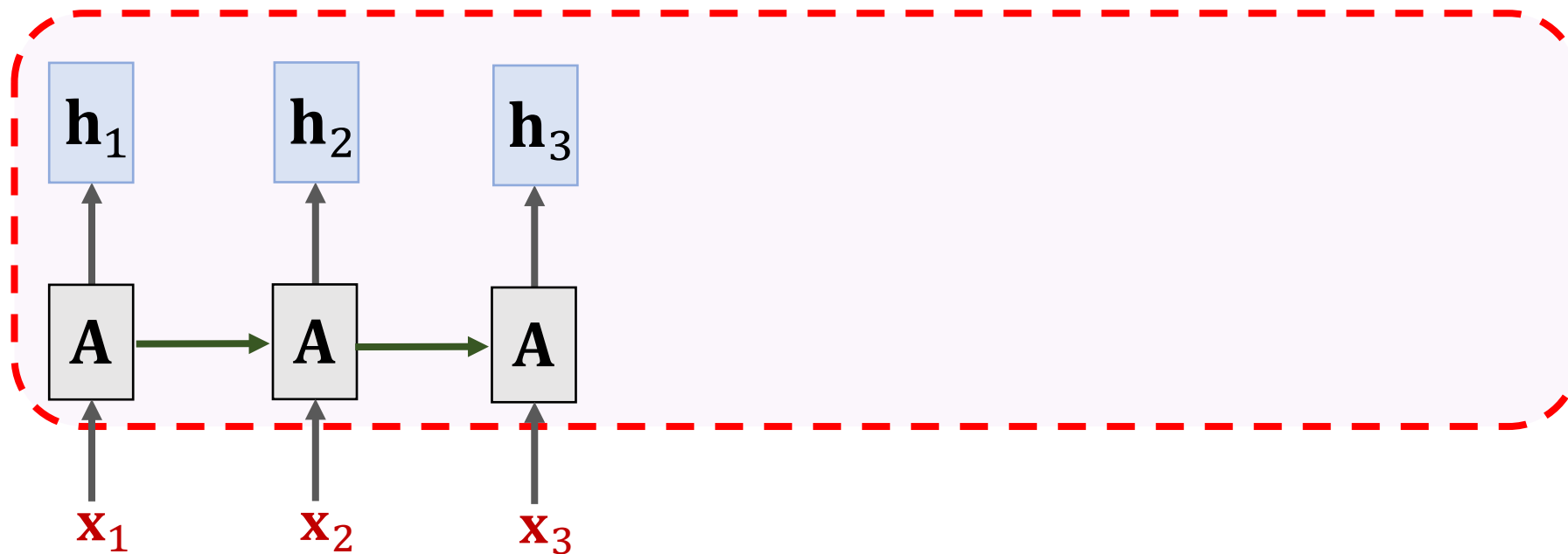


SimpleRNN + Self-Attention

\mathbf{c}_1

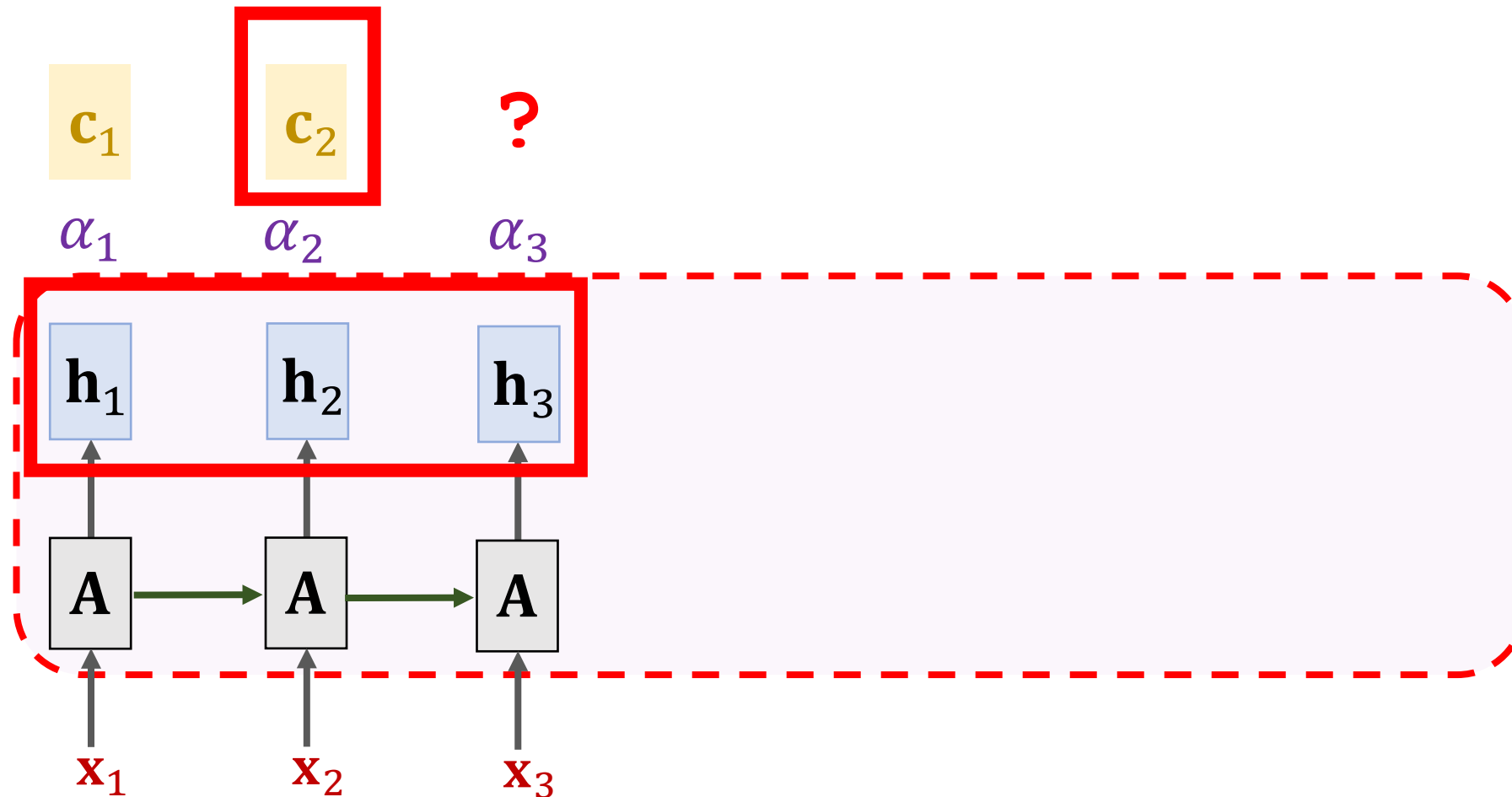
\mathbf{c}_2

?



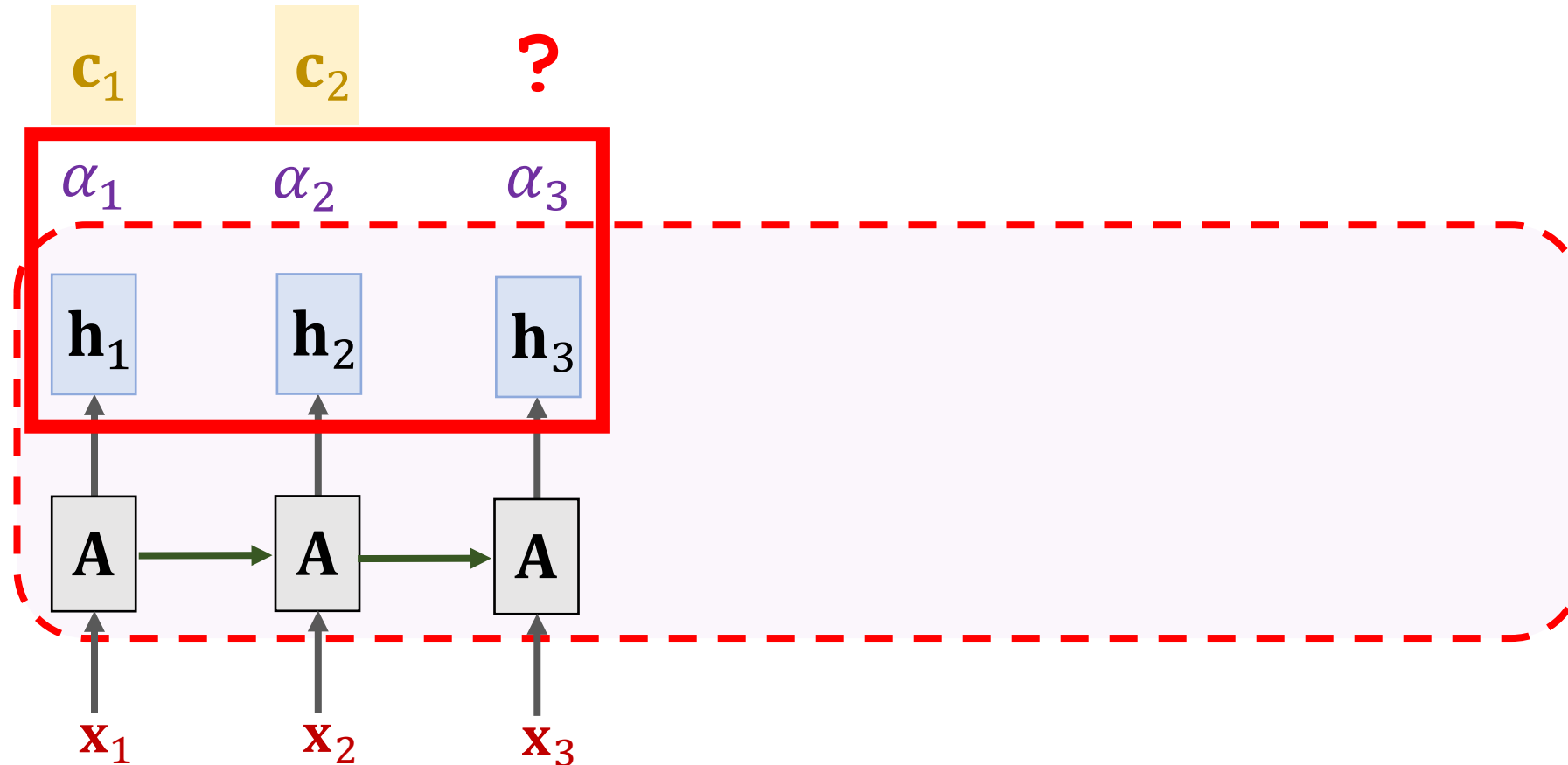
SimpleRNN + Self-Attention

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



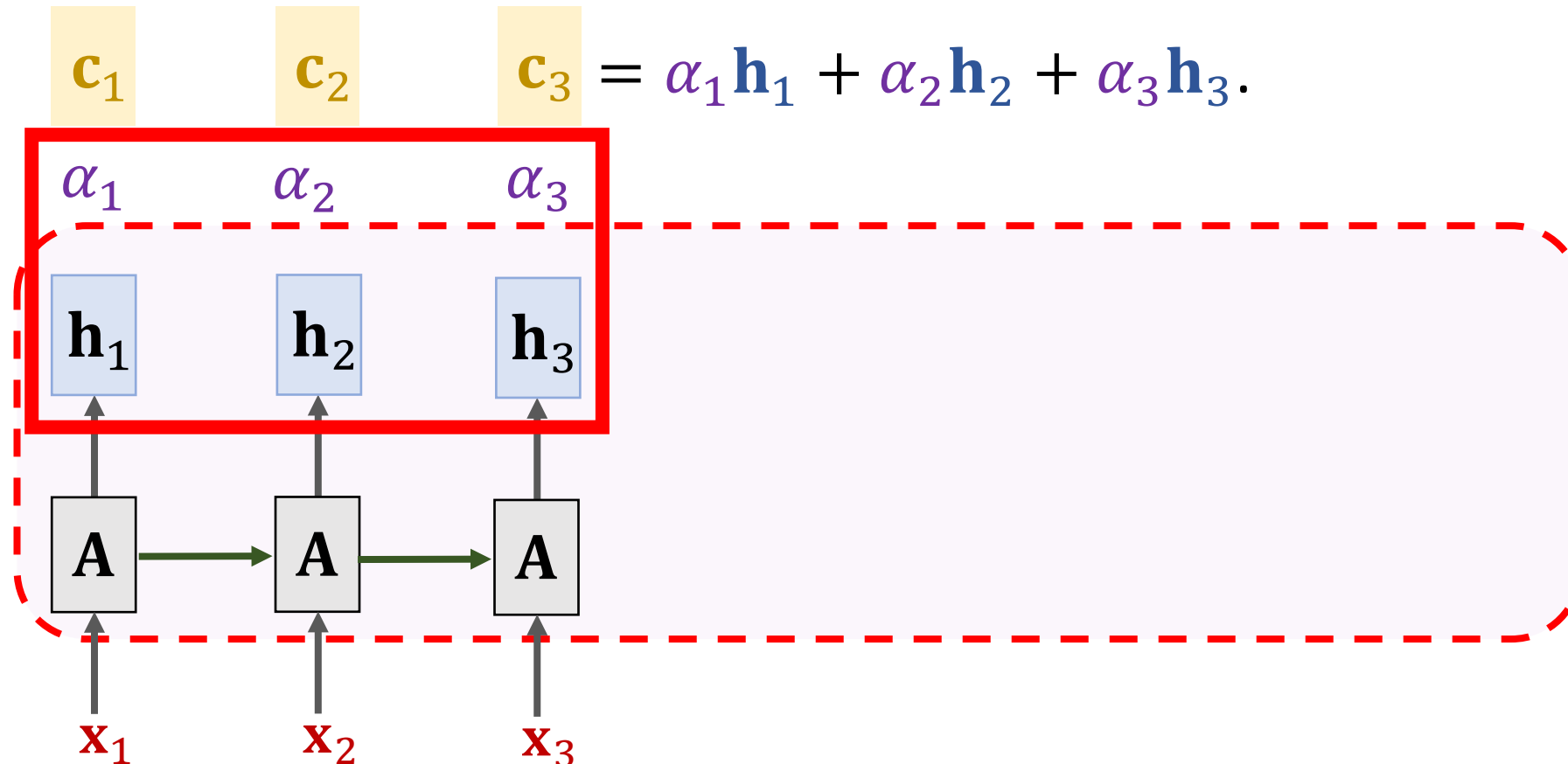
SimpleRNN + Self-Attention

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

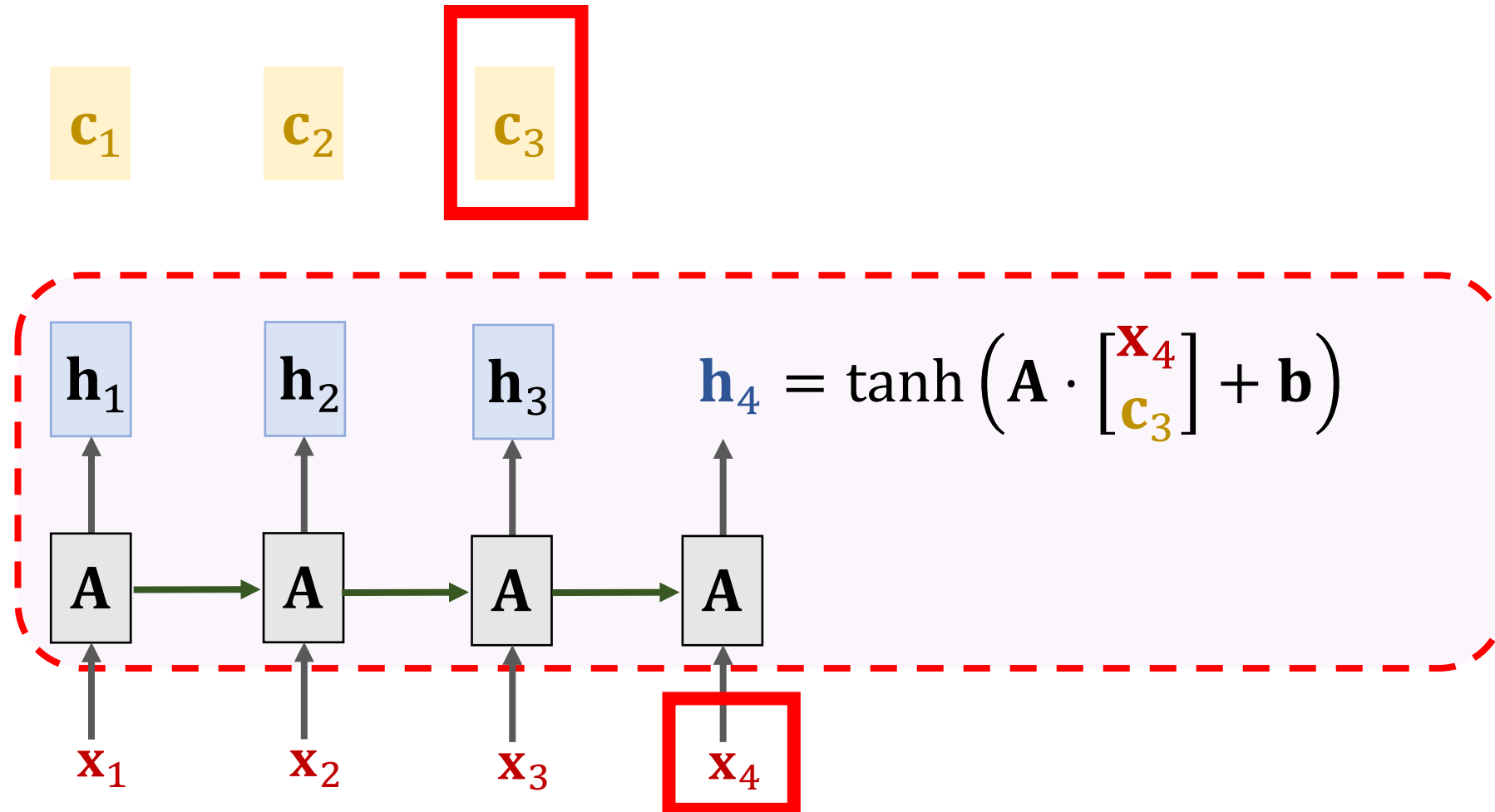


SimpleRNN + Self-Attention

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



SimpleRNN + Self-Attention



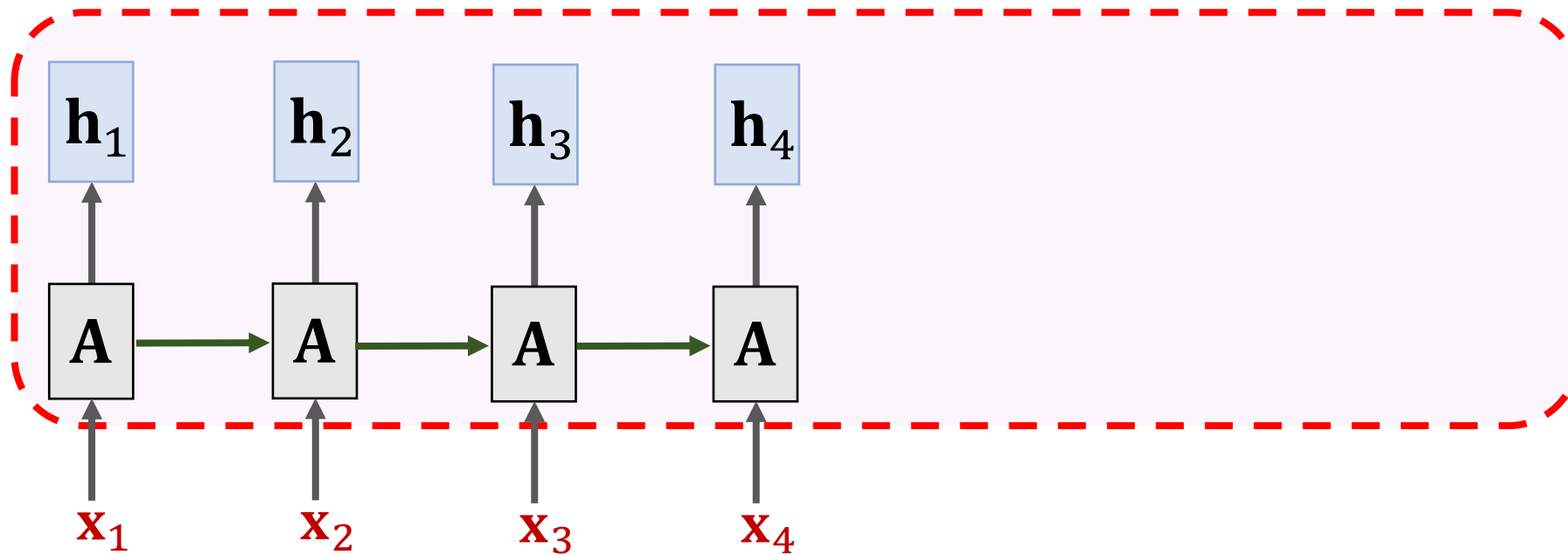
SimpleRNN + Self-Attention

\mathbf{c}_1

\mathbf{c}_2

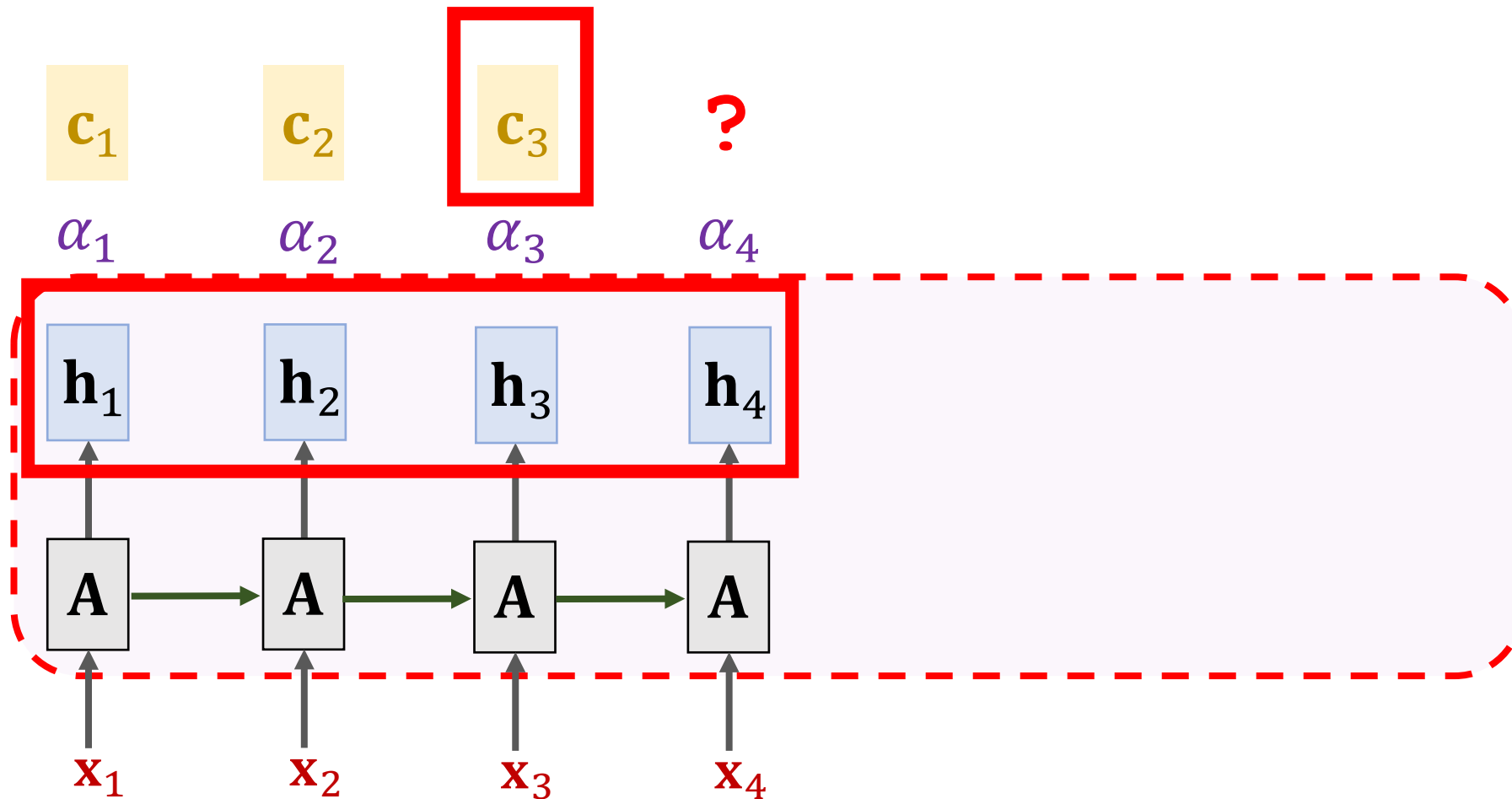
\mathbf{c}_3

?



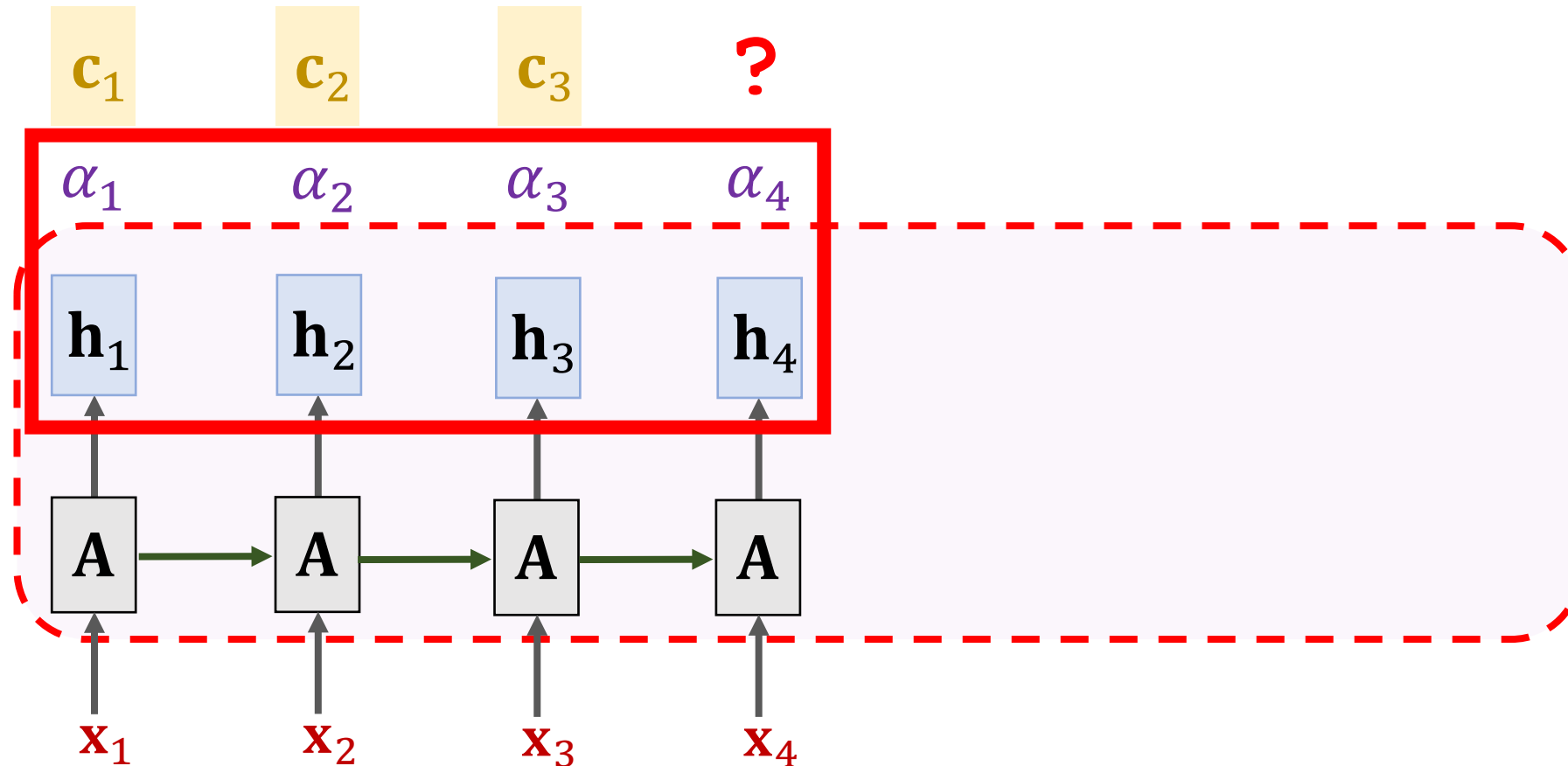
SimpleRNN + Self-Attention

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



SimpleRNN + Self-Attention

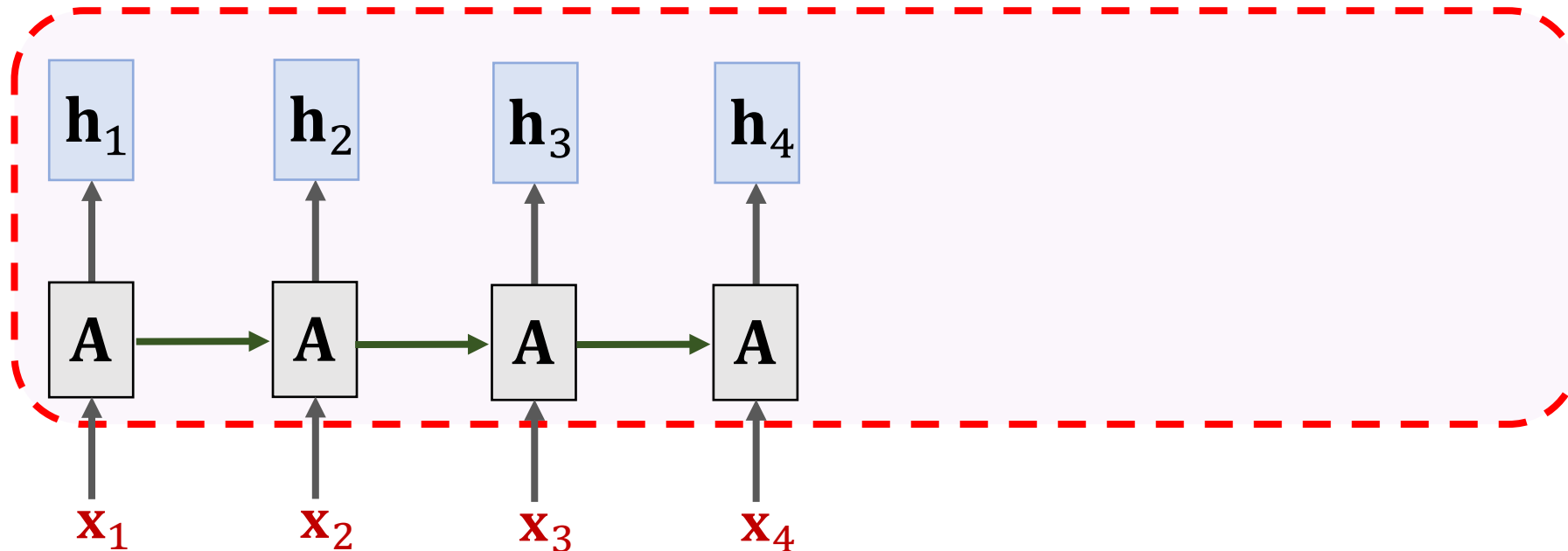
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{c}_3)$



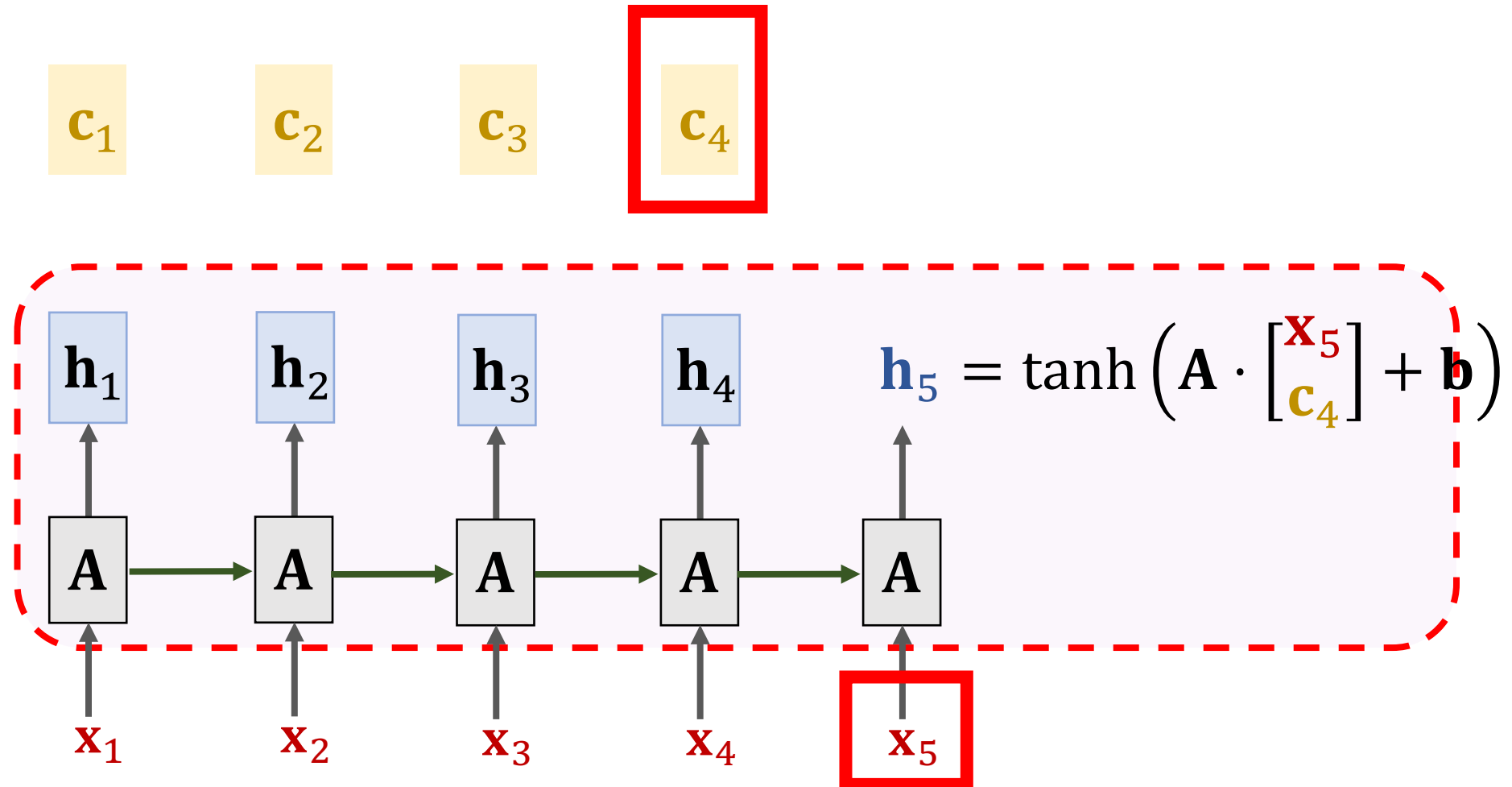
SimpleRNN + Self-Attention

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

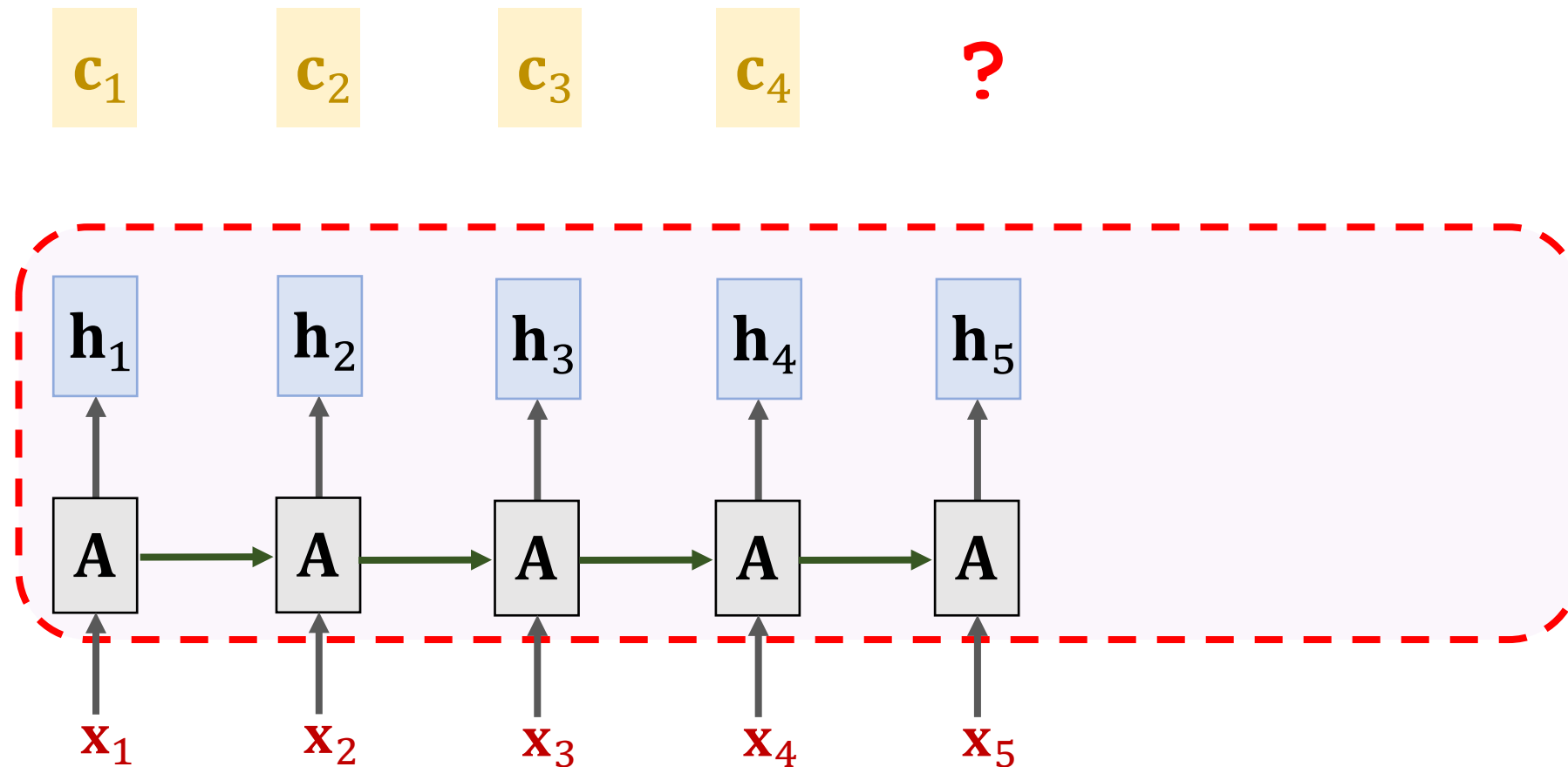
\mathbf{c}_1 \mathbf{c}_2 \mathbf{c}_3 $\mathbf{c}_4 = \alpha_1 \mathbf{h}_1 + \alpha_2 \mathbf{h}_2 + \alpha_3 \mathbf{h}_3 + \alpha_4 \mathbf{h}_4.$



SimpleRNN + Self-Attention

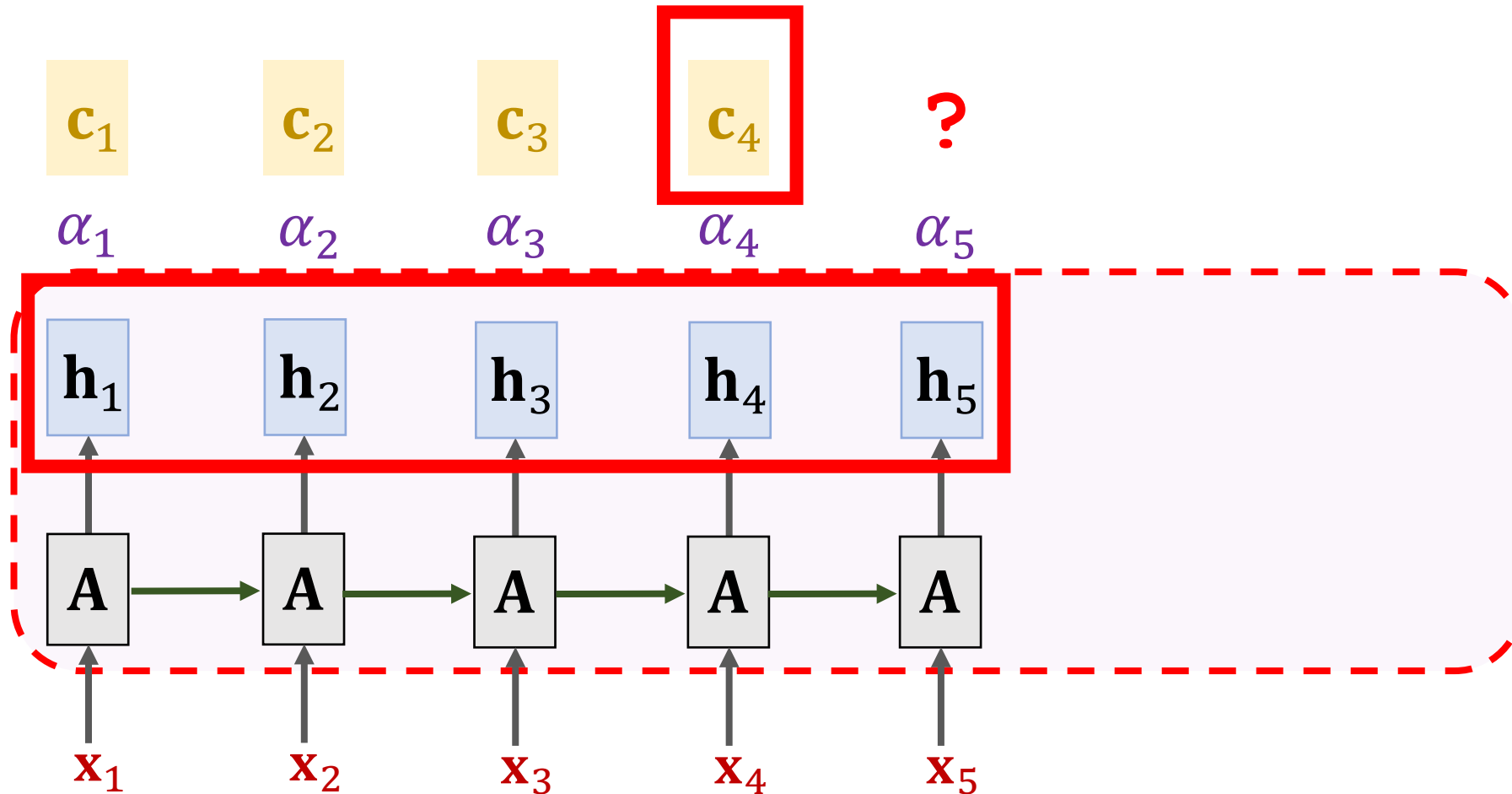


SimpleRNN + Self-Attention



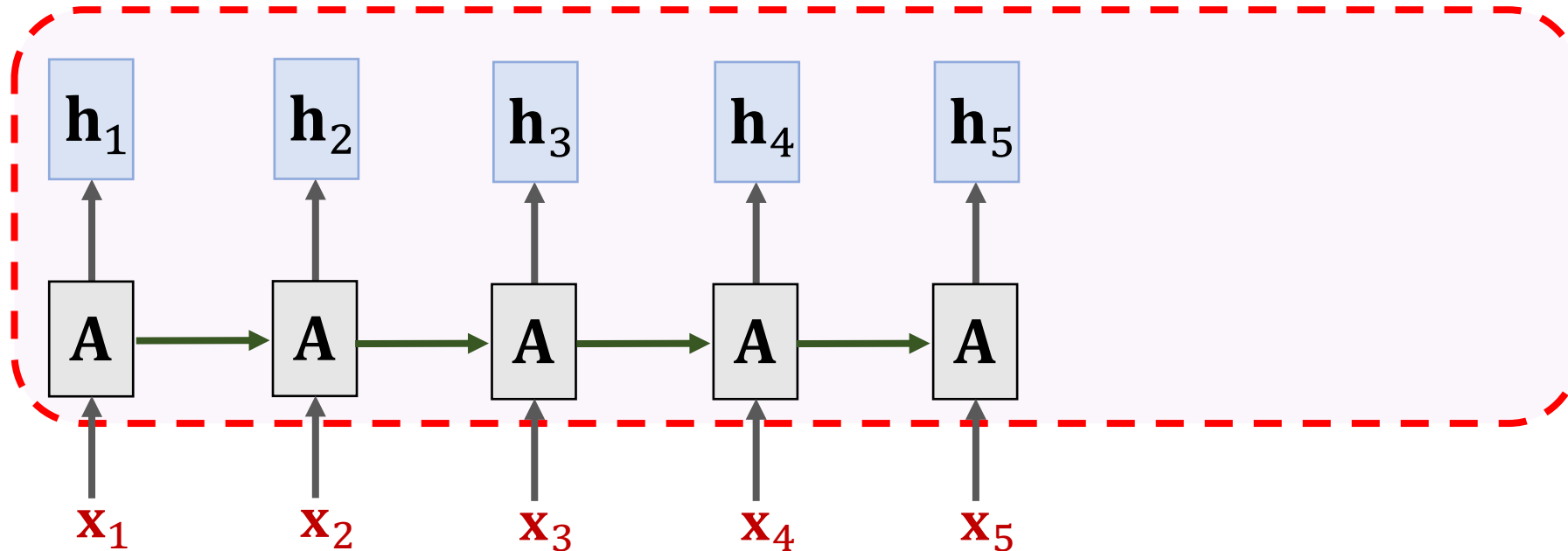
SimpleRNN + Self-Attention

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

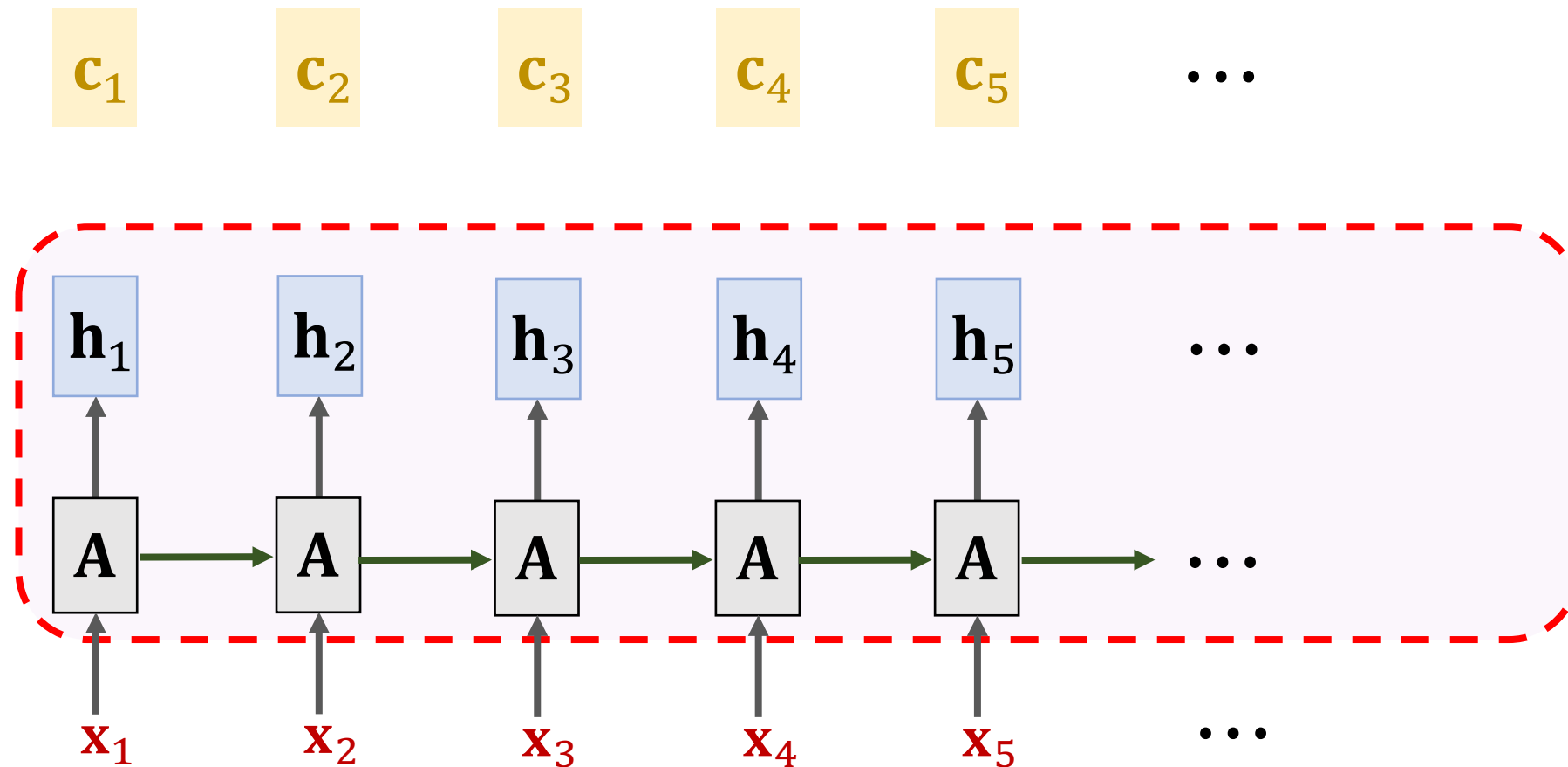


SimpleRNN + Self-Attention

\mathbf{c}_1 \mathbf{c}_2 \mathbf{c}_3 \mathbf{c}_4 $\mathbf{c}_5 = \alpha_1 \mathbf{h}_1 + \alpha_2 \mathbf{h}_2 + \cdots + \alpha_5 \mathbf{h}_5.$



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.

The diagram shows the sentence "The FBI is chasing a criminal on the run." with attention weights. The words are arranged in a grid where each row represents the current word being processed and each column represents a word in the context. Blue highlights indicate high attention weights. The first row shows high attention on "The". The second row shows high attention on "The" and "FBI". The third row shows high attention on "The", "FBI", and "is". The fourth row shows high attention on "The", "FBI", "is", and "chasing". The fifth row shows high attention on "The", "FBI", "is", "chasing", and "a". The sixth row shows high attention on "The", "FBI", "is", "chasing", "a", and "criminal". The seventh row shows high attention on "The", "FBI", "is", "chasing", "a", "criminal", and "on". The eighth row shows high attention on "The", "FBI", "is", "chasing", "a", "criminal", "on", and "the". The ninth row shows high attention on "The", "FBI", "is", "chasing", "a", "criminal", "on", "the", and "run". The tenth row shows high attention on "The", "FBI", "is", "chasing", "a", "criminal", "on", "the", "run", and ".".

The
The FBI
The FBI is
The FBI is chasing
The FBI is chasing a
The FBI is chasing a criminal
The FBI is chasing a criminal on
The FBI is chasing a criminal on the
The FBI is chasing a criminal on the run
The FBI is chasing a criminal on the run .

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