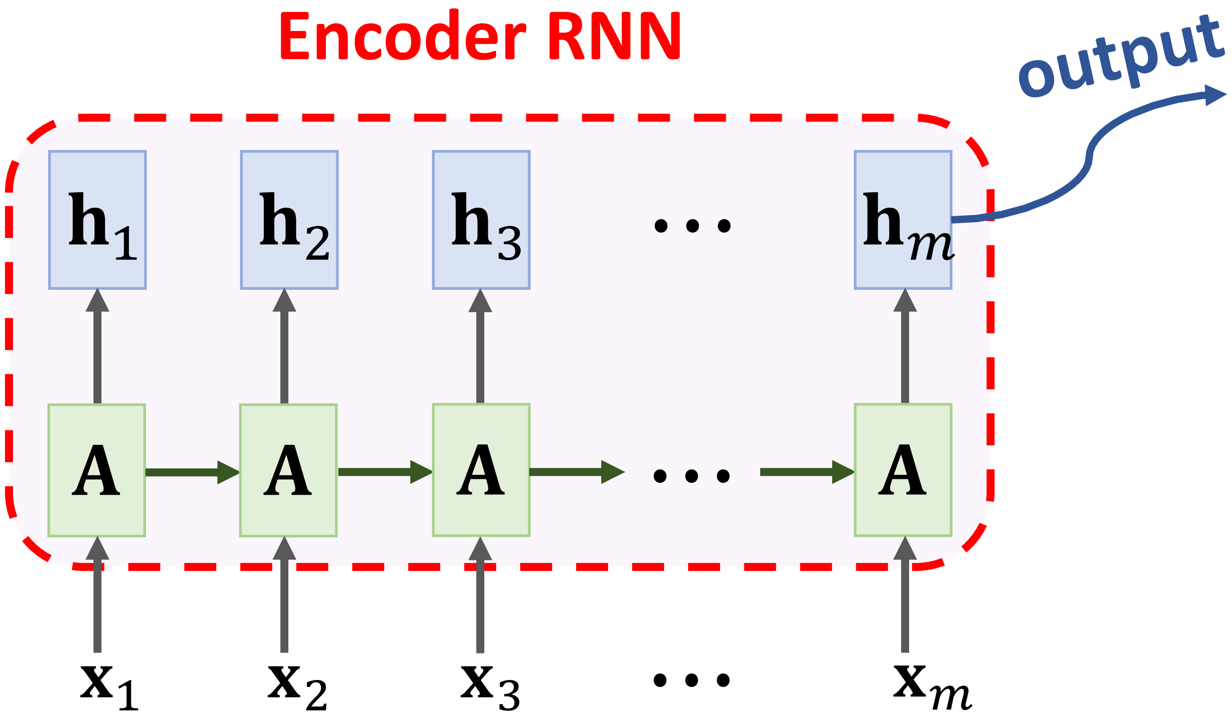
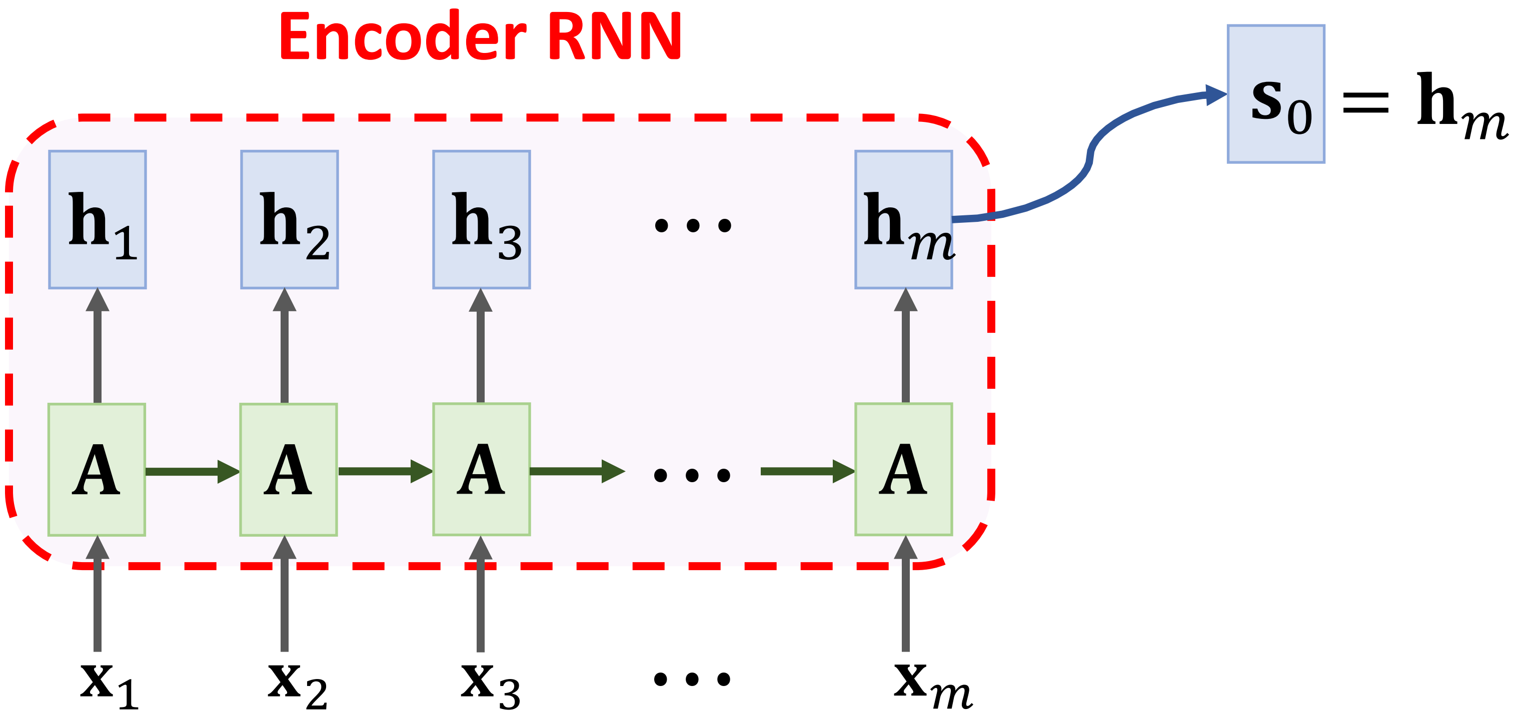
Attention

Shusen Wang

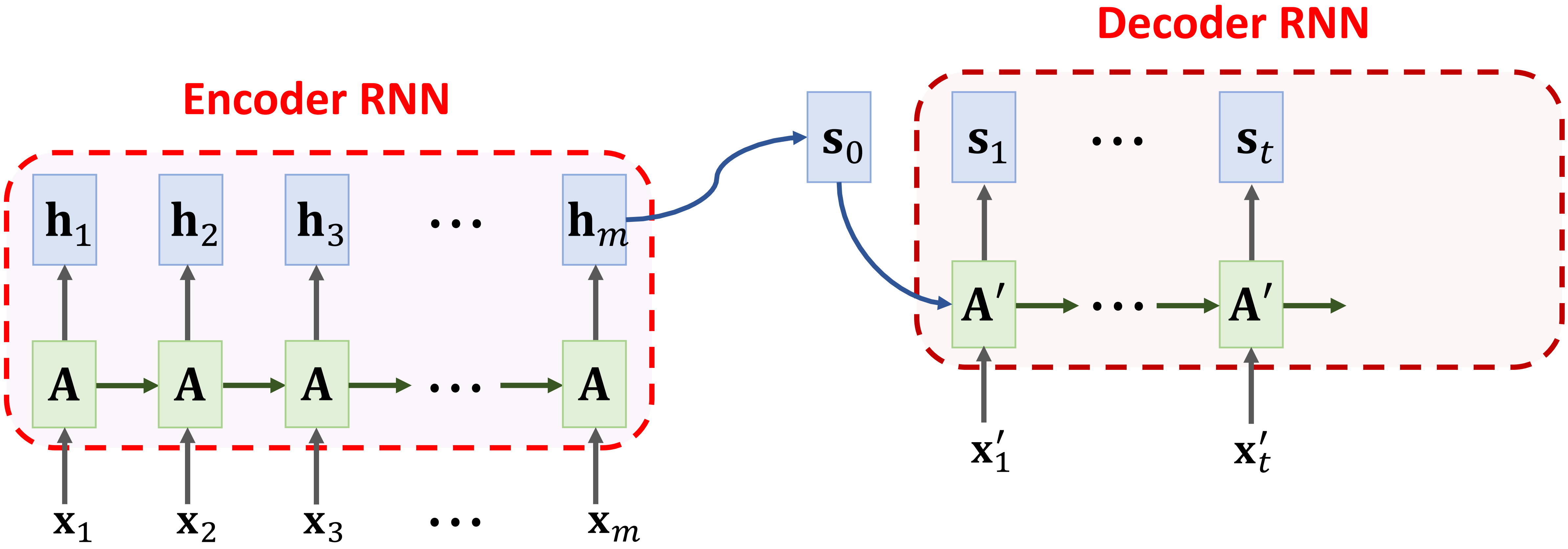
Seq2Seq Model



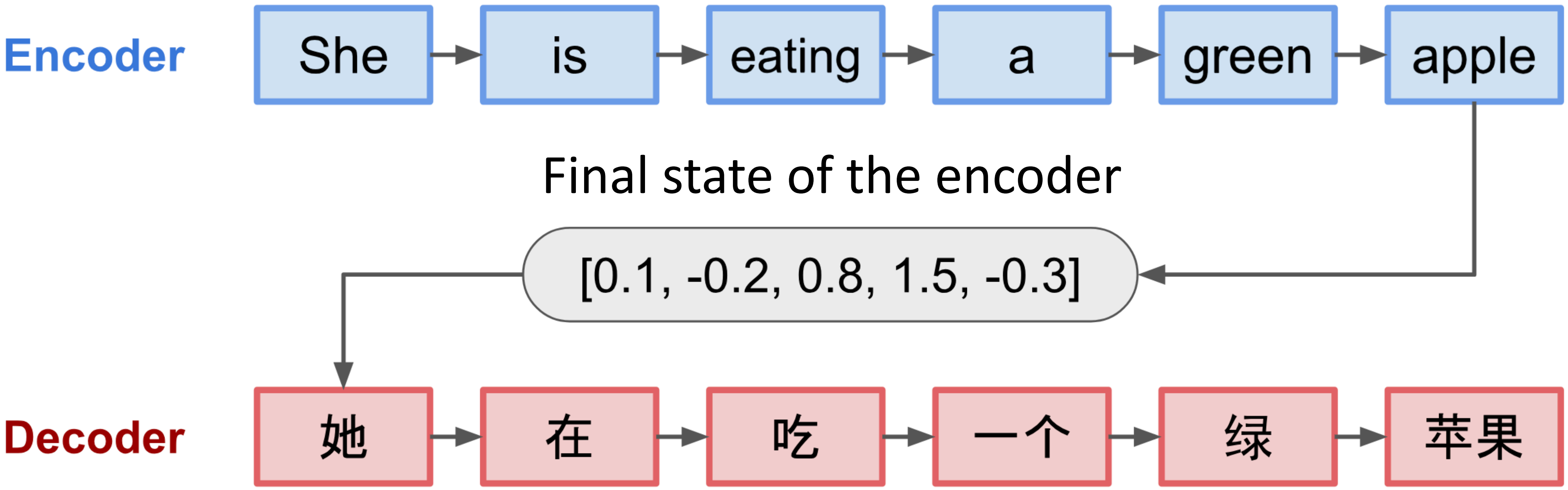
Seq2Seq Model



Seq2Seq Model



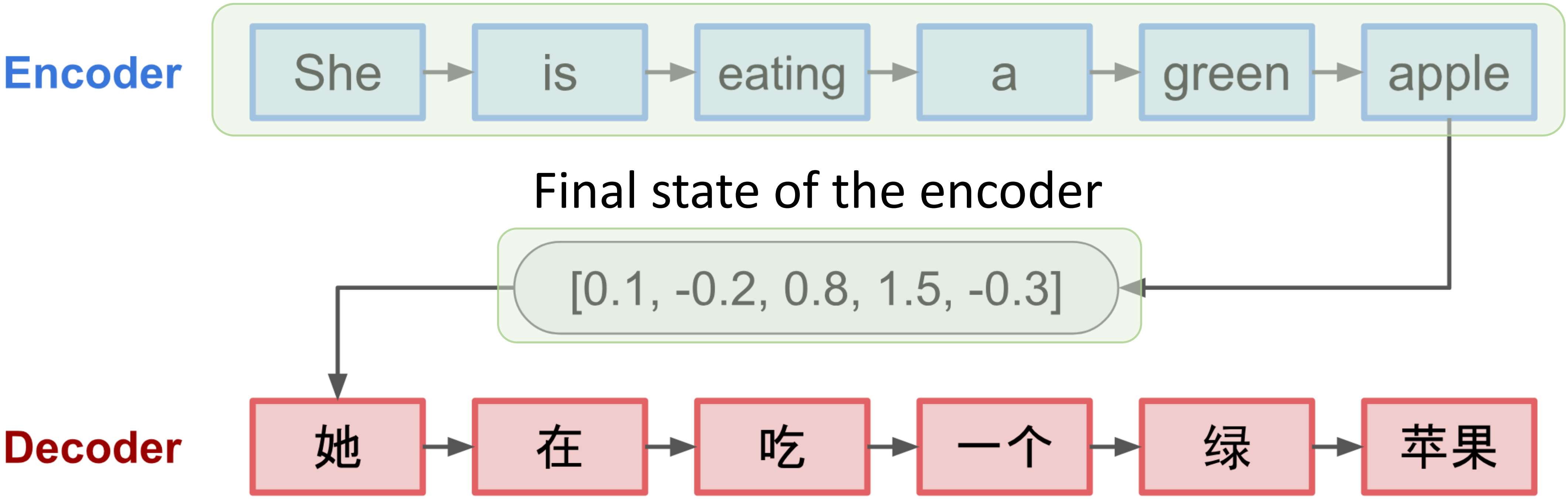
Seq2Seq Model



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

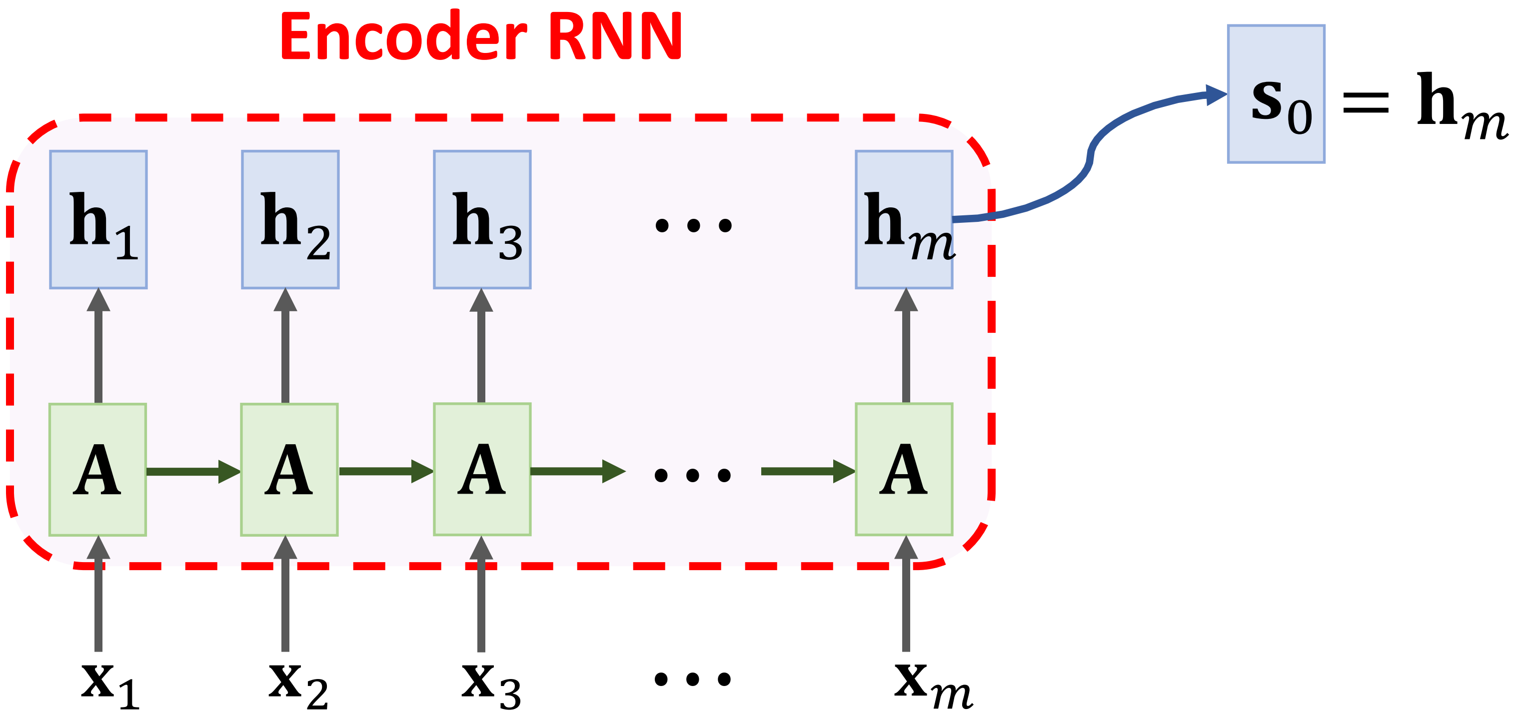
Seq2Seq Model with Attention

- Attention tremendously improves Seq2Seq model.
- With attention, Seq2Seq model does not forget source input.
- With attention, the decoder knows where to focus.
- Downside: much more computation.

Original paper:

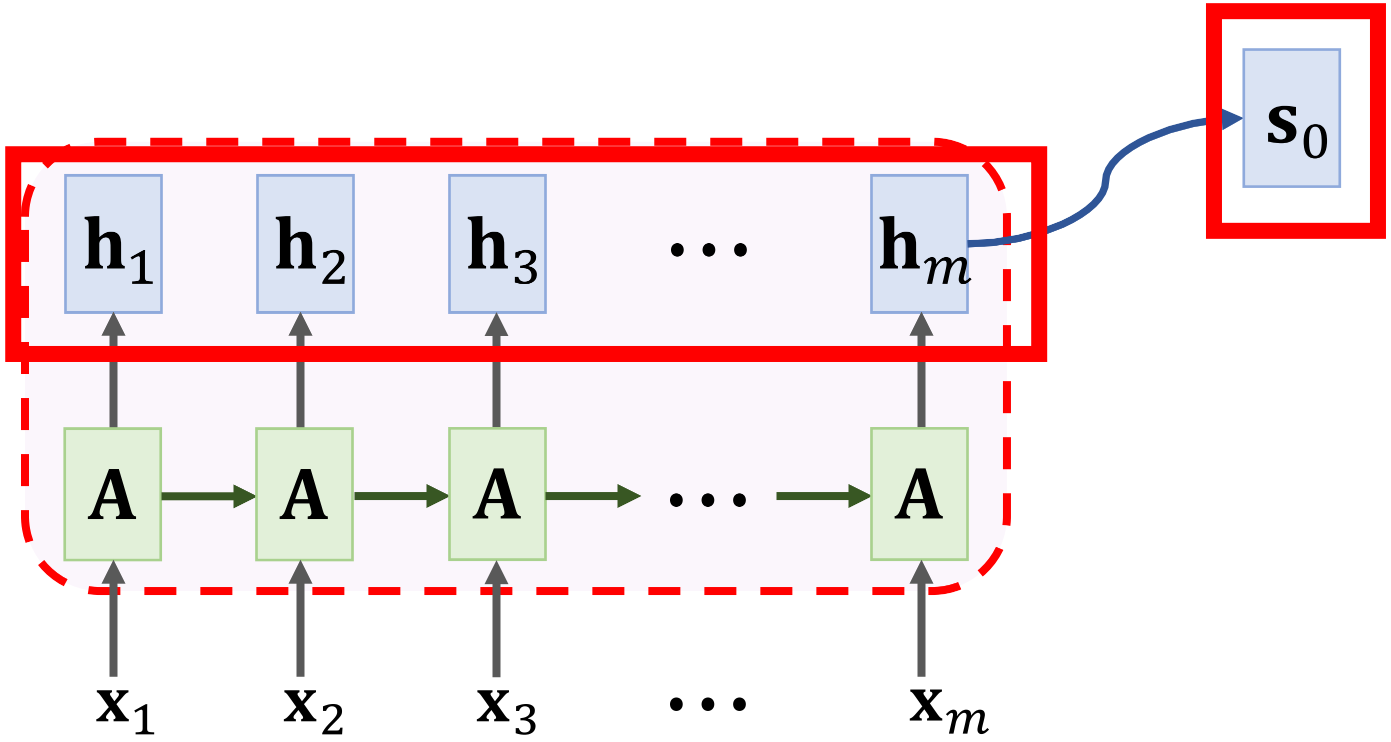
- Bahdanau, Cho, & Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

SimpleRNN + Attention



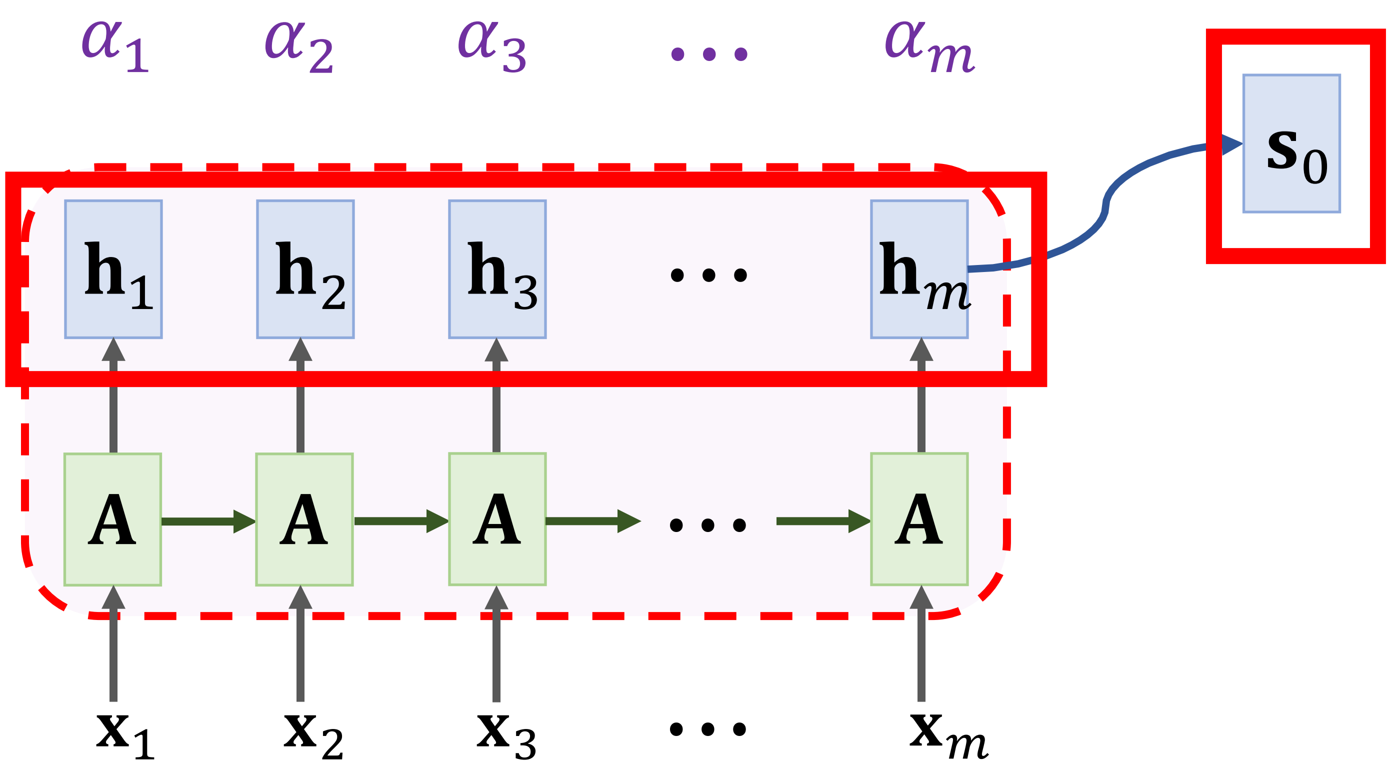
SimpleRNN + Attention

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



SimpleRNN + Attention

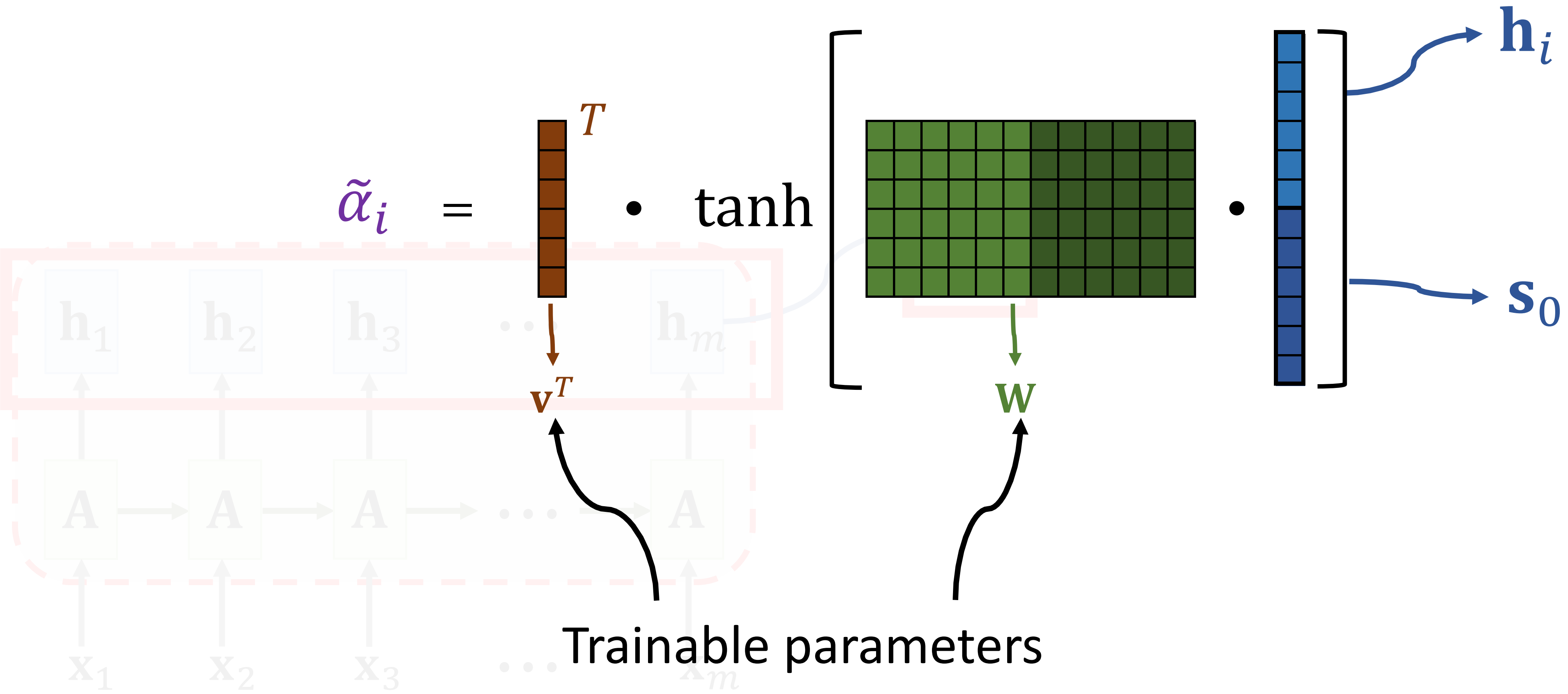
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_0)$



SimpleRNN + Attention

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

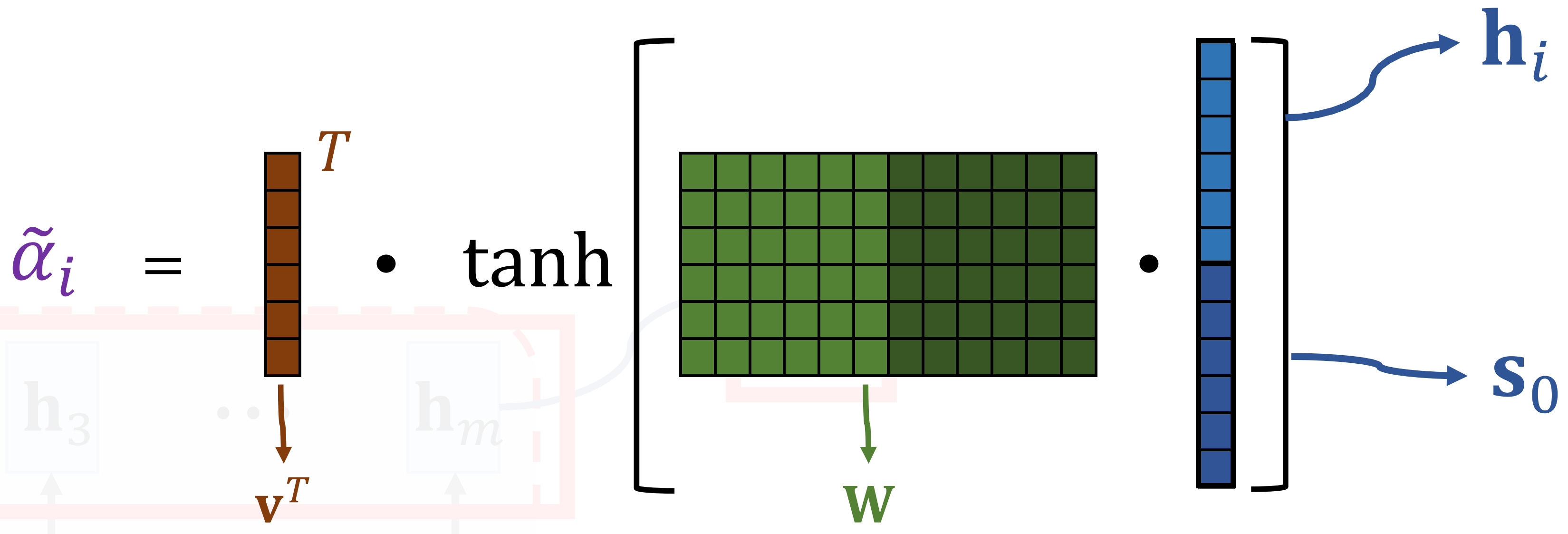
Option 1 (used in the original paper):



SimpleRNN + Attention

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

Option 1 (used in the original paper):



Then **normalize** $\tilde{\alpha}_1, \cdots, \tilde{\alpha}_m$ (so that they sum to 1):

$$[\alpha_1, \cdots, \alpha_m] = \text{Softmax}\left([\tilde{\alpha}_1, \cdots, \tilde{\alpha}_m]\right)$$

SimpleRNN + Attention

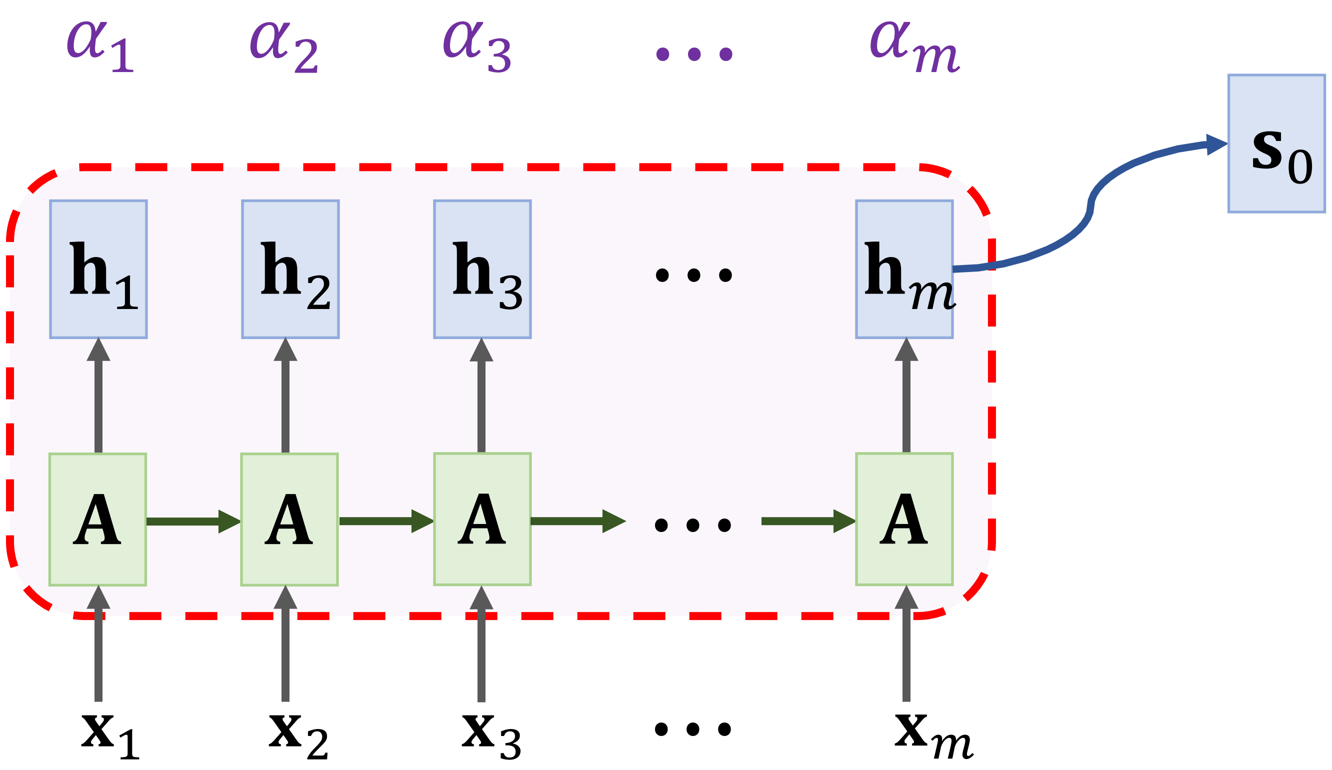
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_0)$

Option 2 (more popular; similar 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 product:
   - $\tilde{\alpha}_i = \tilde{\mathbf{h}}_i^T \cdot \tilde{\mathbf{s}}_0$.
3. Normalization:
   - $[\alpha_1, \cdots, \alpha_m] = \text{Softmax}\left([\tilde{\alpha}_1, \cdots, \tilde{\alpha}_m]\right)$

SimpleRNN + Attention

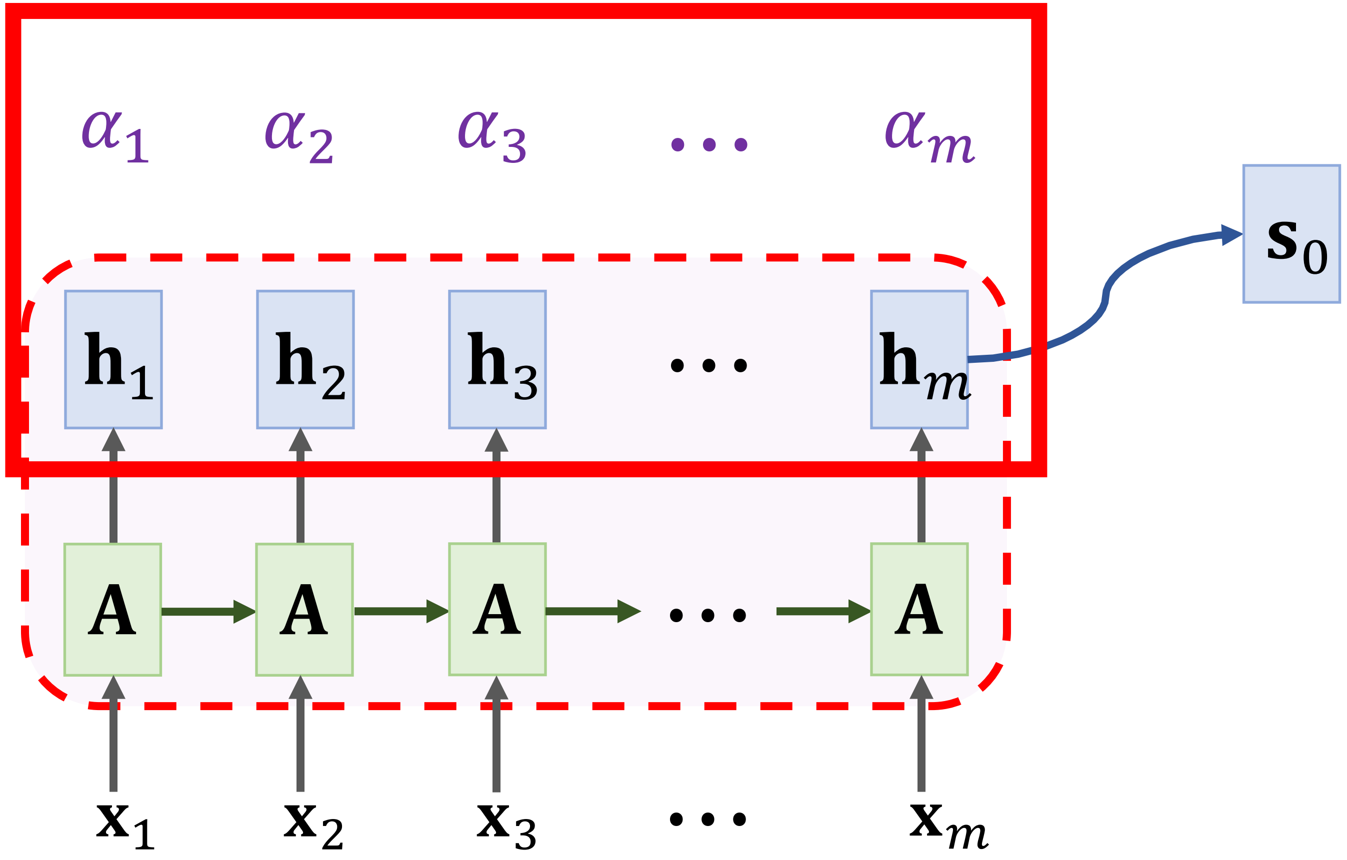
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_0)$



SimpleRNN + Attention

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

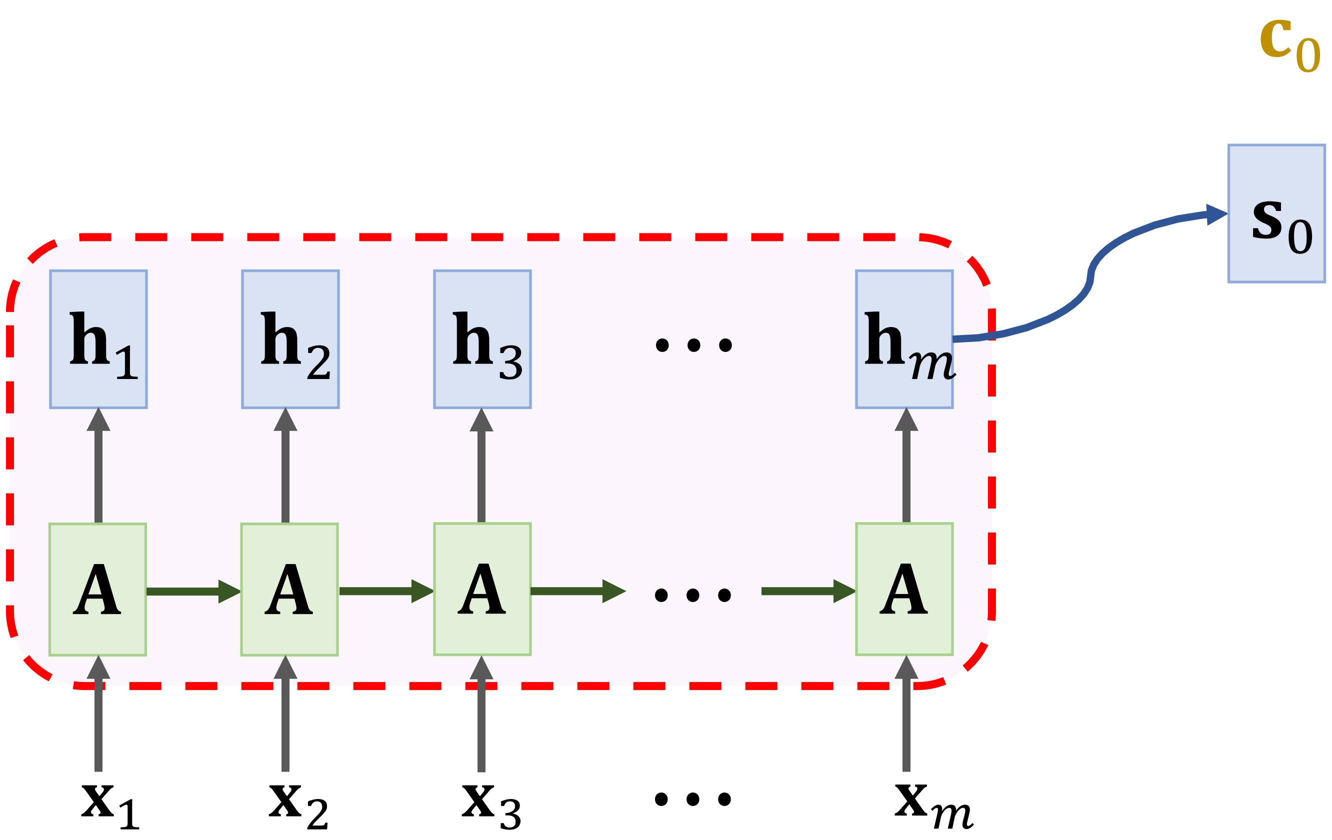
Context vector: $\mathbf{c}_0 = \alpha_1 \mathbf{h}_1 + \cdots + \alpha_m \mathbf{h}_m.$



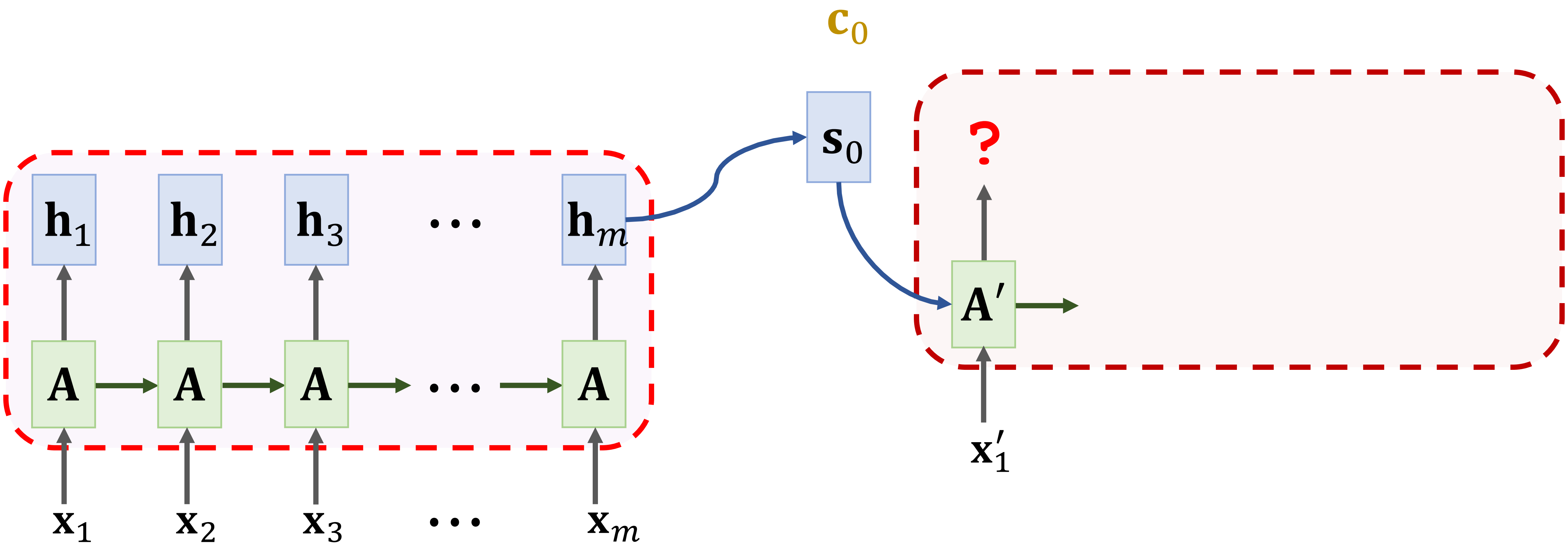
SimpleRNN + Attention

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

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



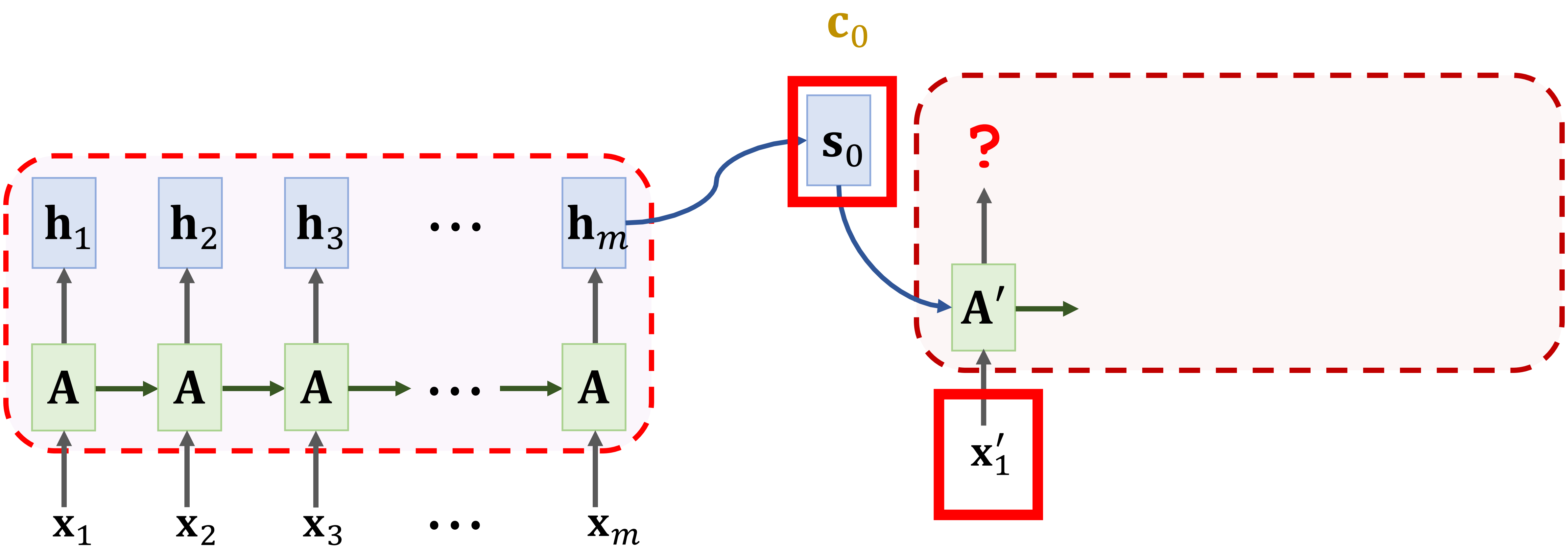
SimpleRNN + Attention



SimpleRNN

SimpleRNN:

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



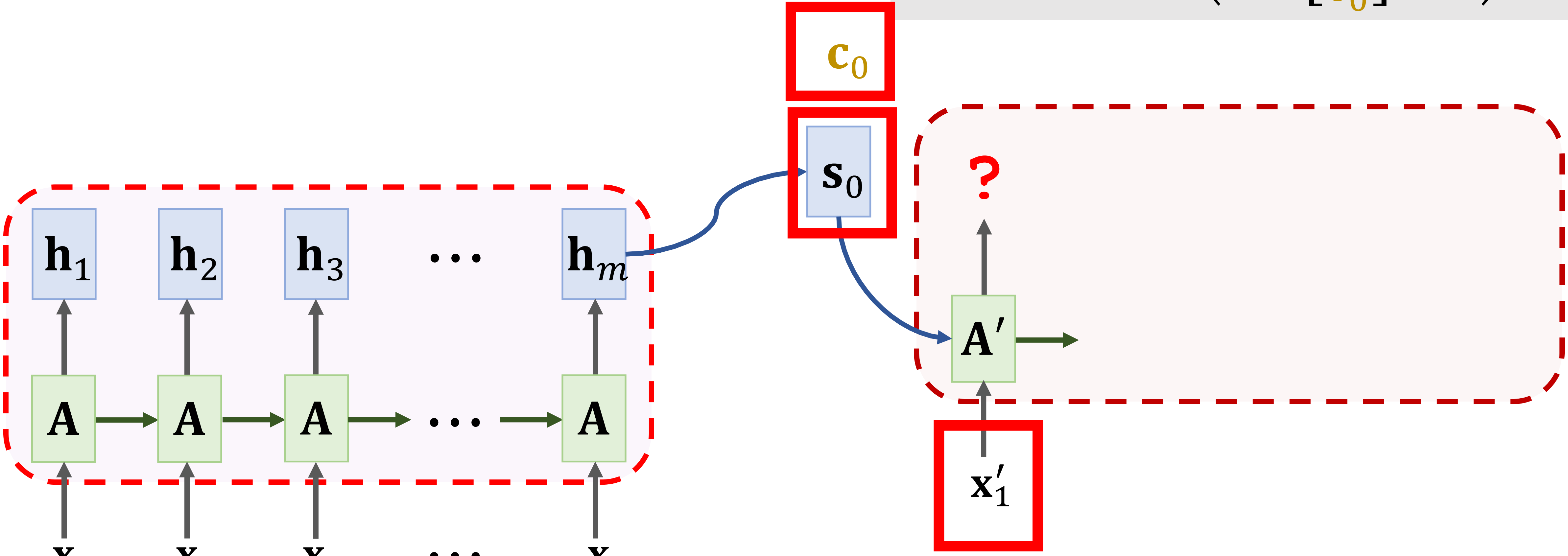
SimpleRNN + Attention

SimpleRNN:

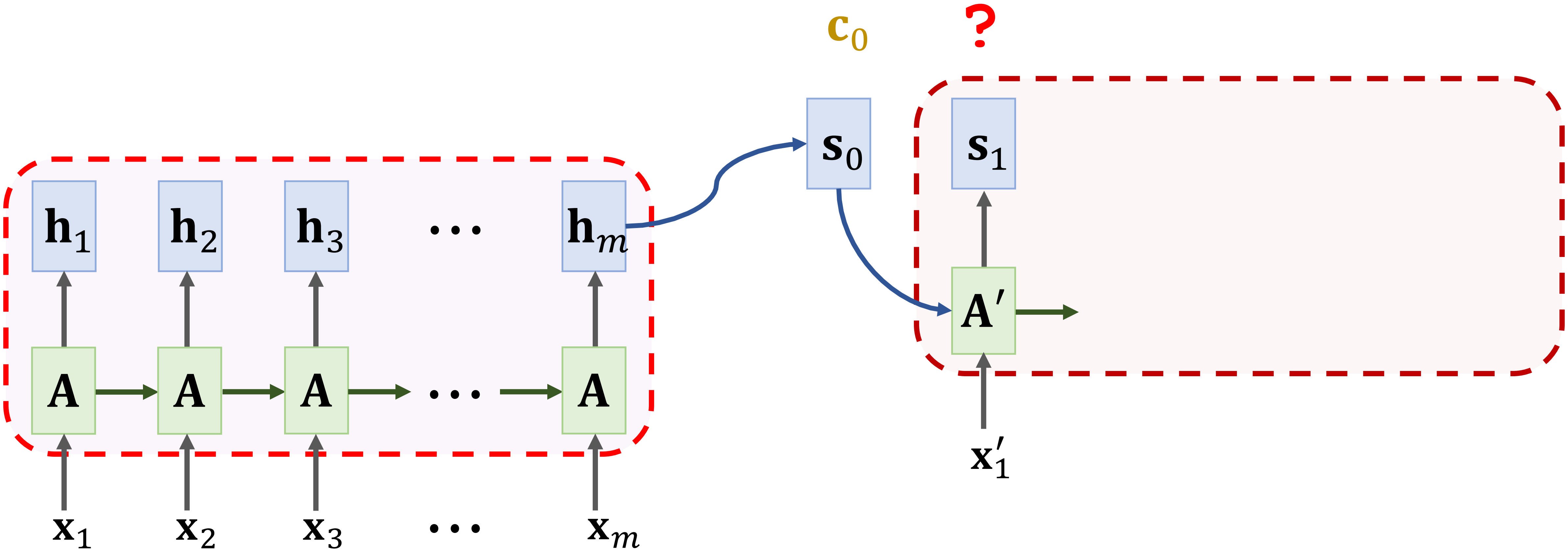
$$\mathbf{s}_1 = \tanh\left(\mathbf{A}' \cdot \begin{bmatrix} \mathbf{x}'_1 \\ \mathbf{s}_0 \end{bmatrix} + \mathbf{b}\right)$$

SimpleRNN + Attention:

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

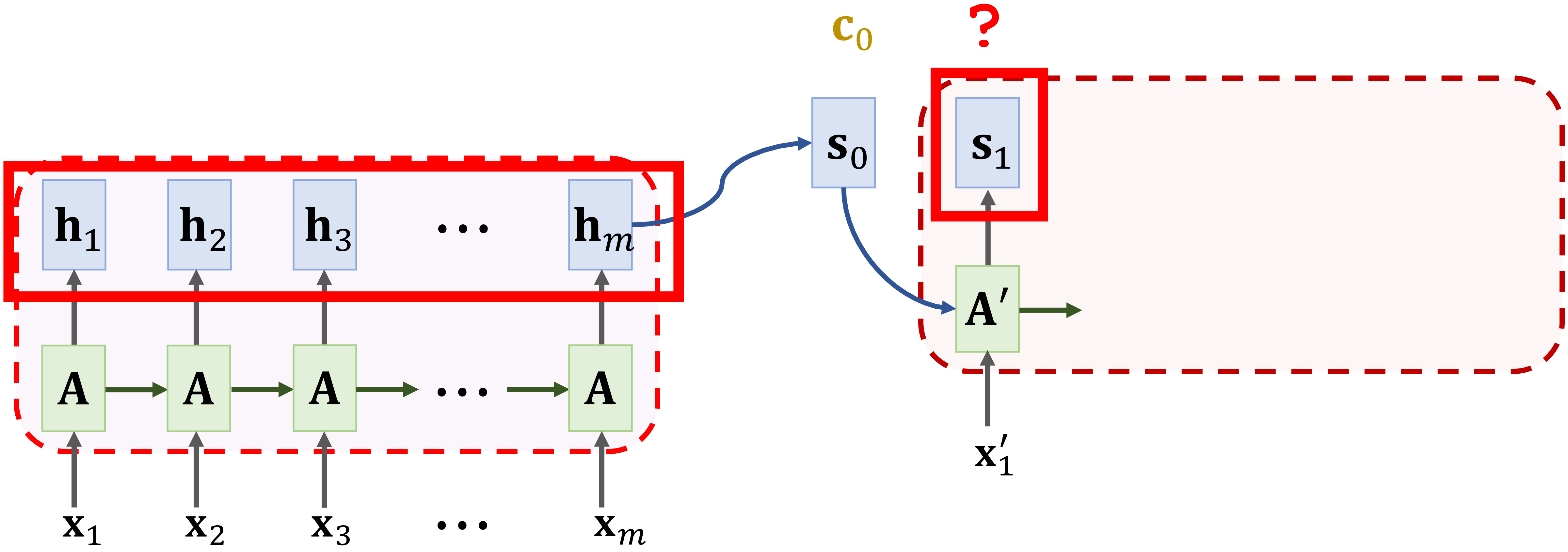


SimpleRNN + Attention



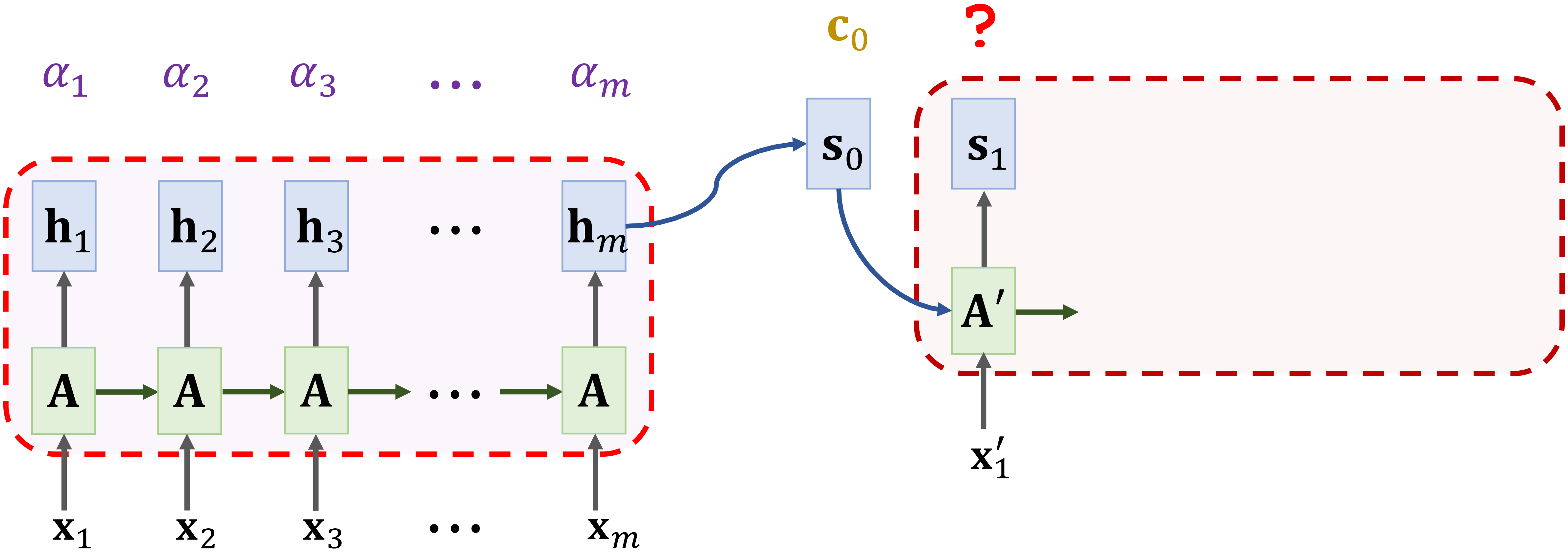
SimpleRNN + Attention

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



SimpleRNN + Attention

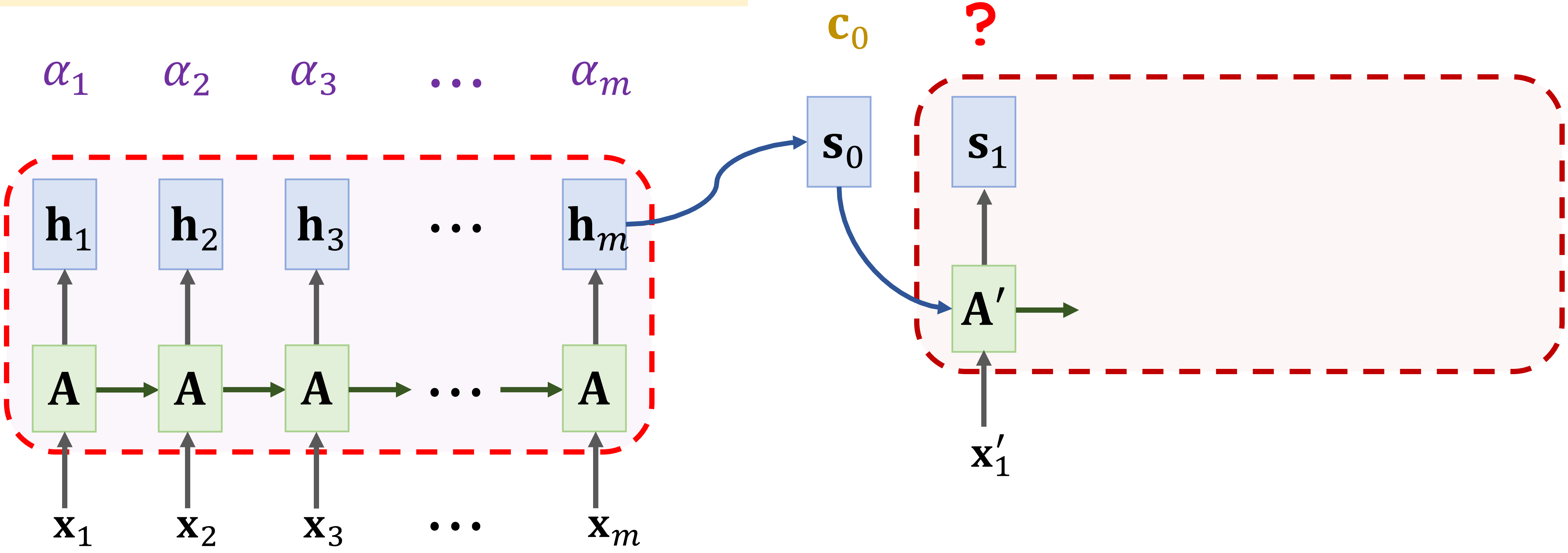
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_1)$



SimpleRNN + Attention

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

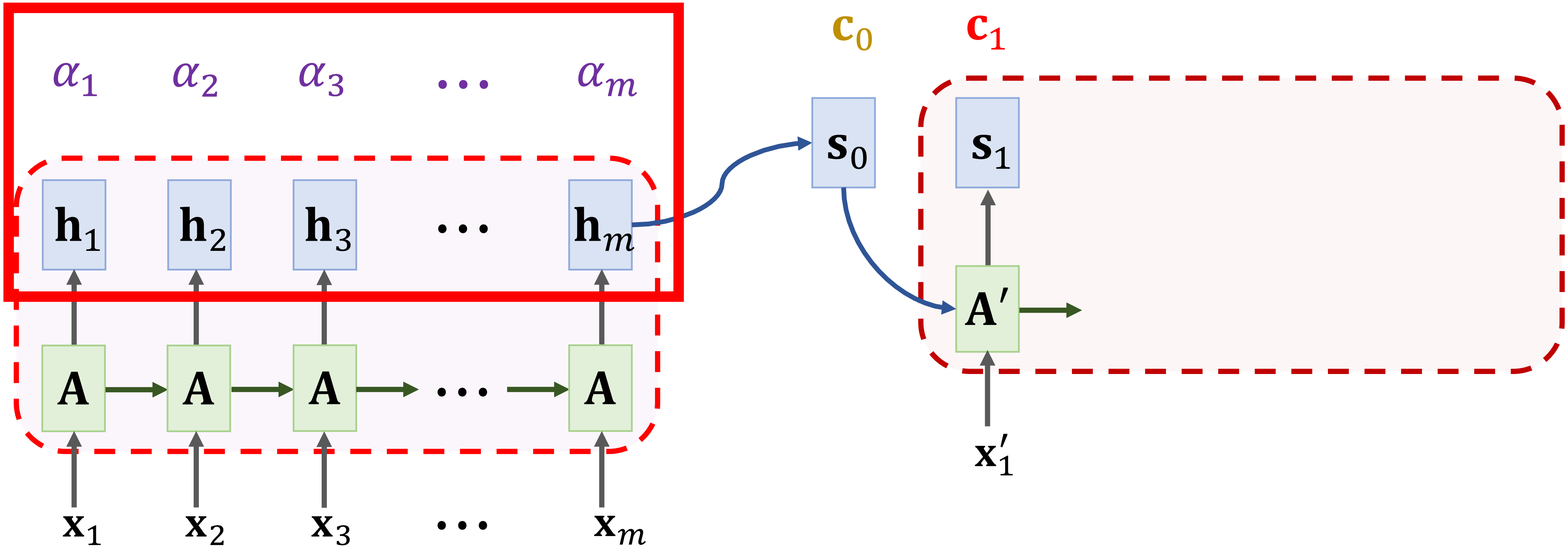
Do not re-use the α's computed previously.



SimpleRNN + Attention

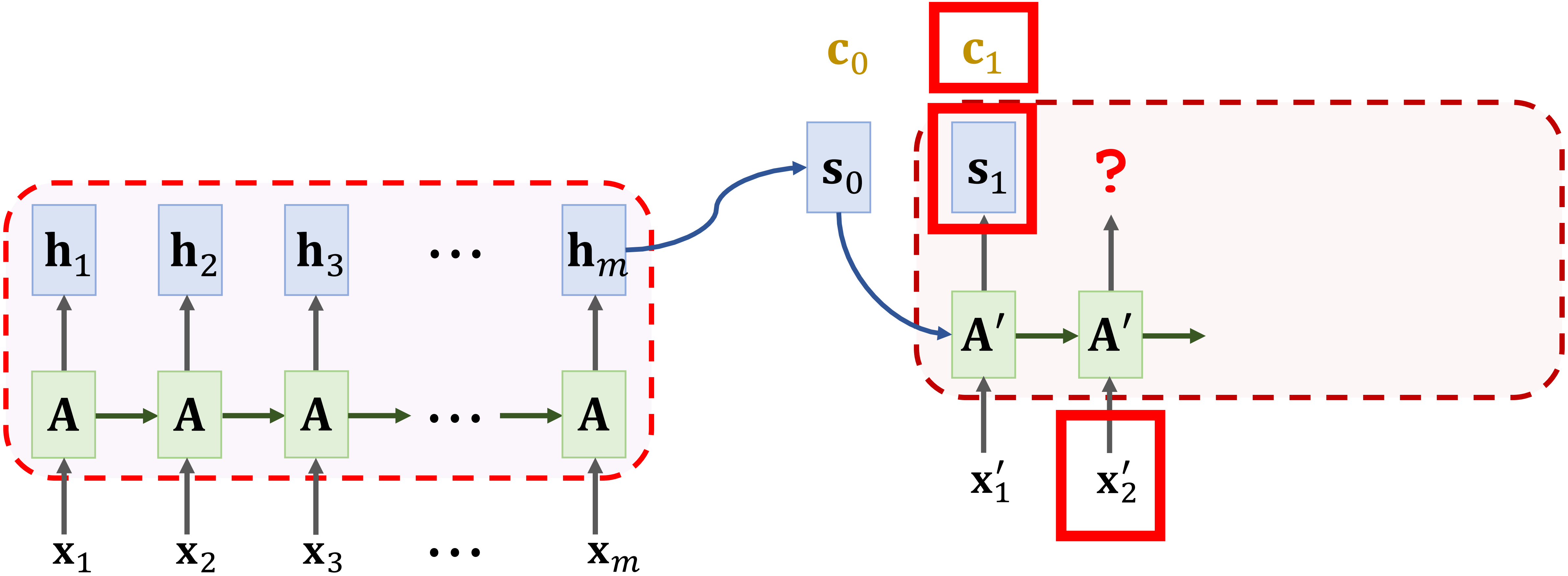
Weights: $\alpha_i = \text{similarity}(\mathbf{h}_i, \mathbf{s}_1)$

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

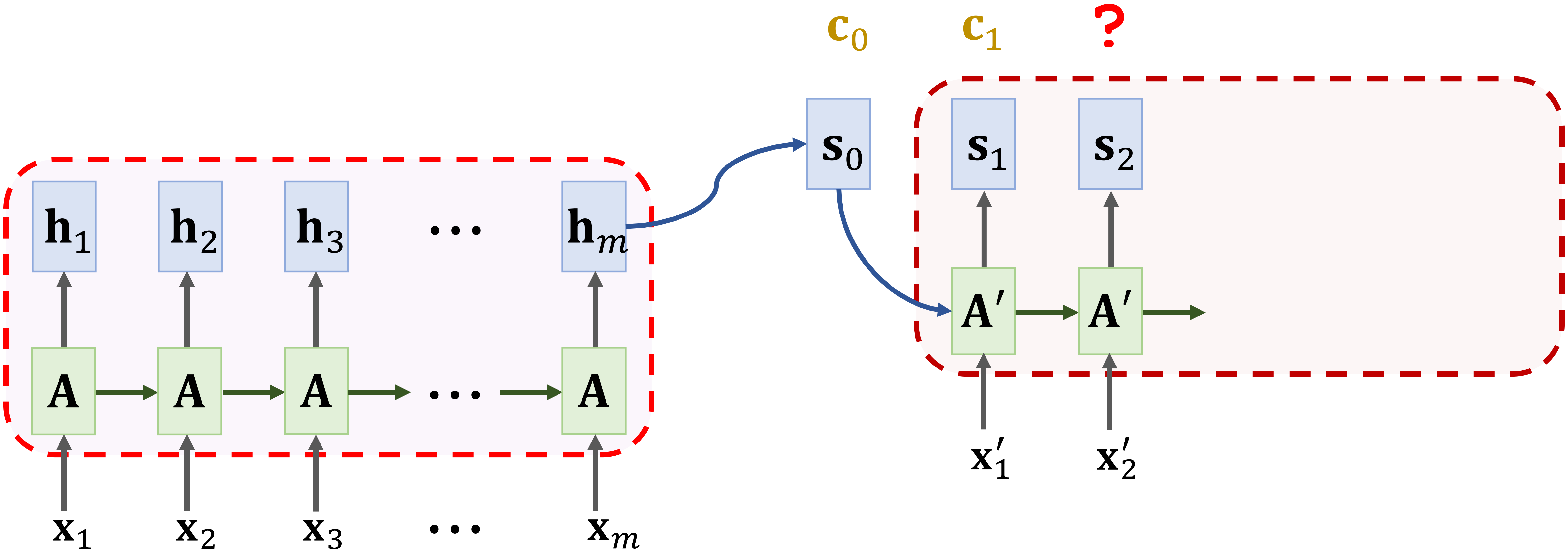


SimpleRNN + Attention

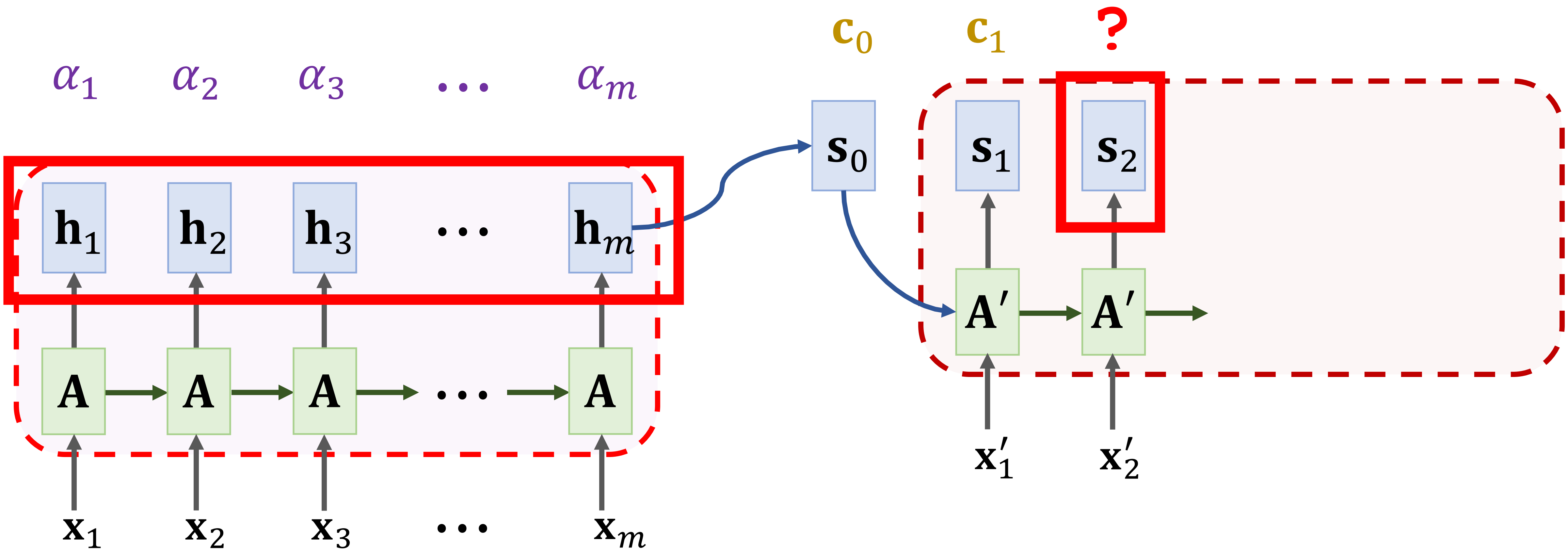
$$\mathbf{s}_2 = \tanh\left(\mathbf{A}' \cdot \begin{bmatrix} \mathbf{x}'_2 \\ \mathbf{s}_1 \\ \mathbf{c}_1 \end{bmatrix} + \mathbf{b}\right)$$



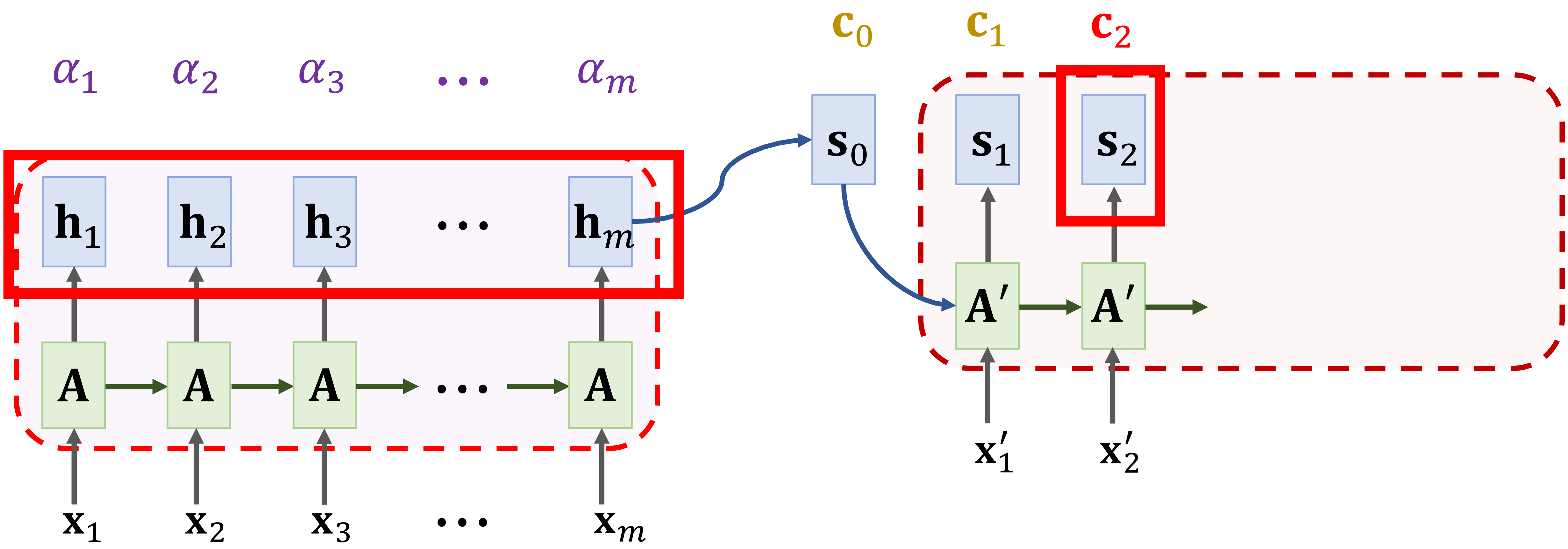
SimpleRNN + Attention



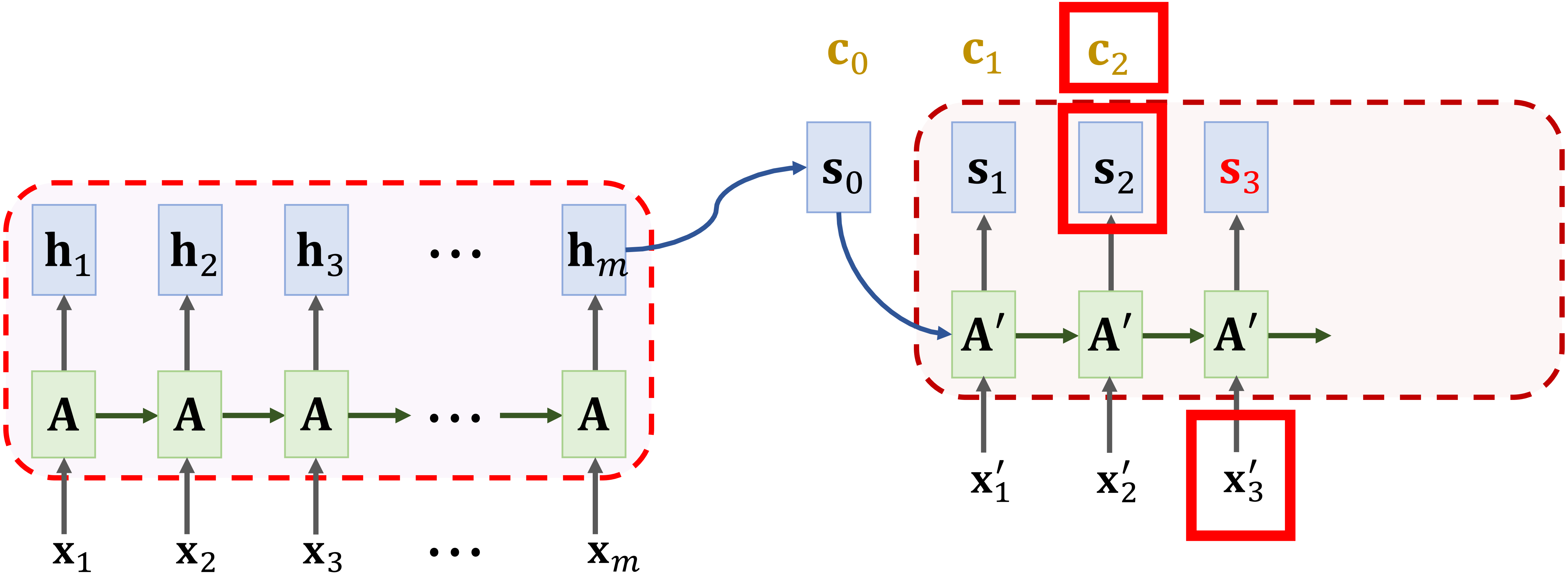
SimpleRNN + Attention



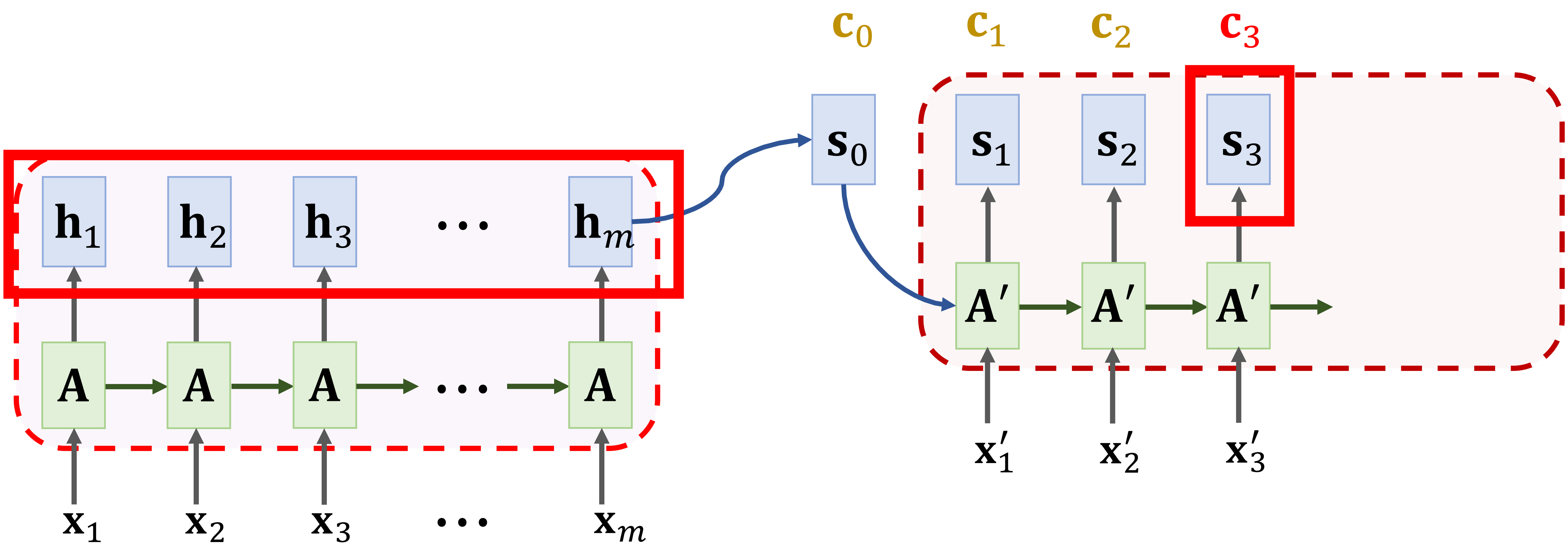
SimpleRNN + Attention



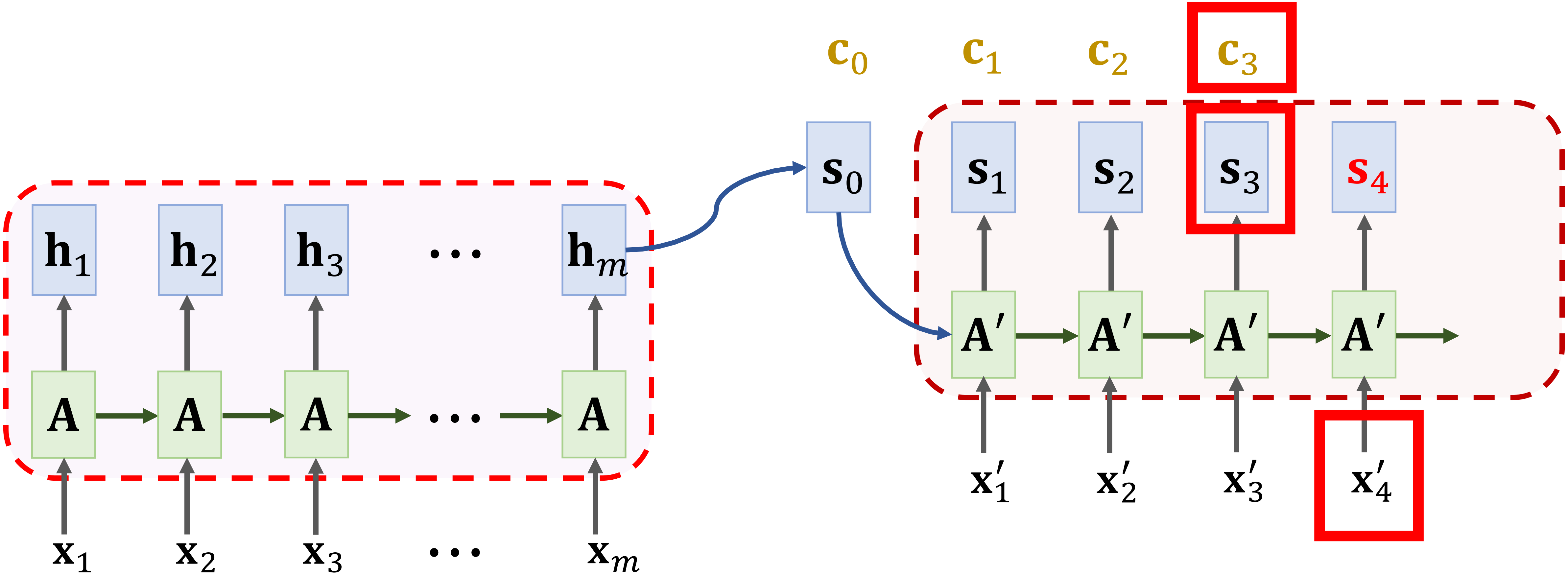
SimpleRNN + Attention



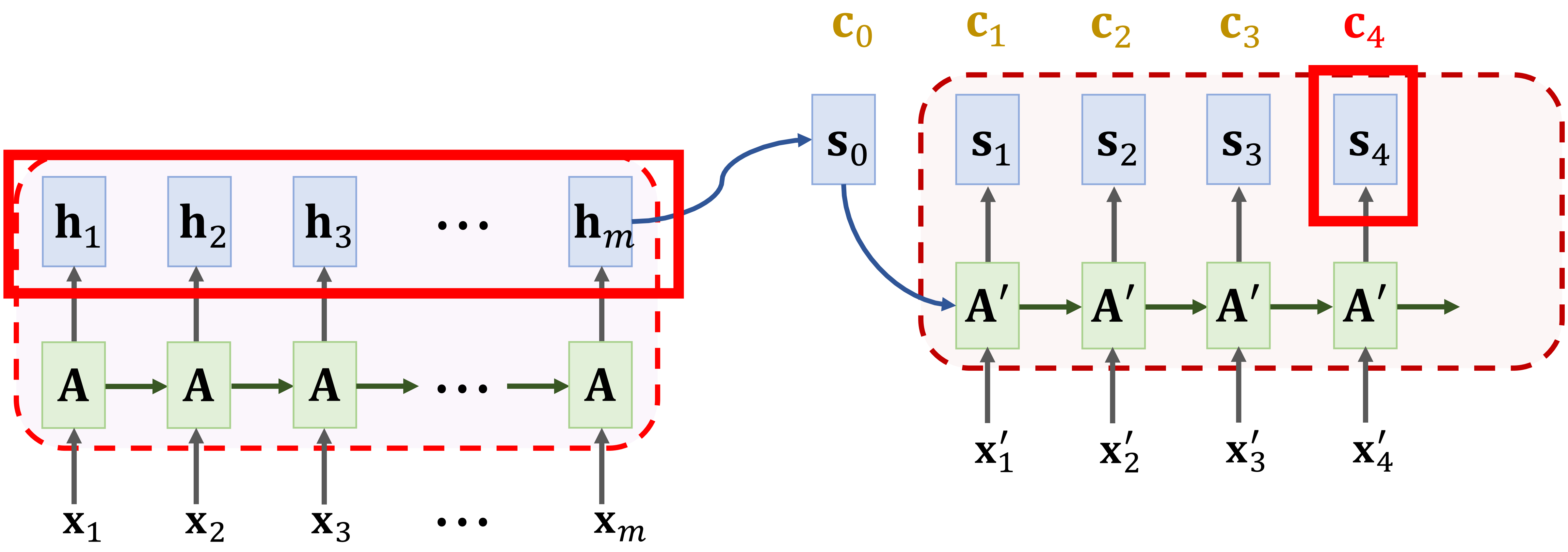
SimpleRNN + Attention



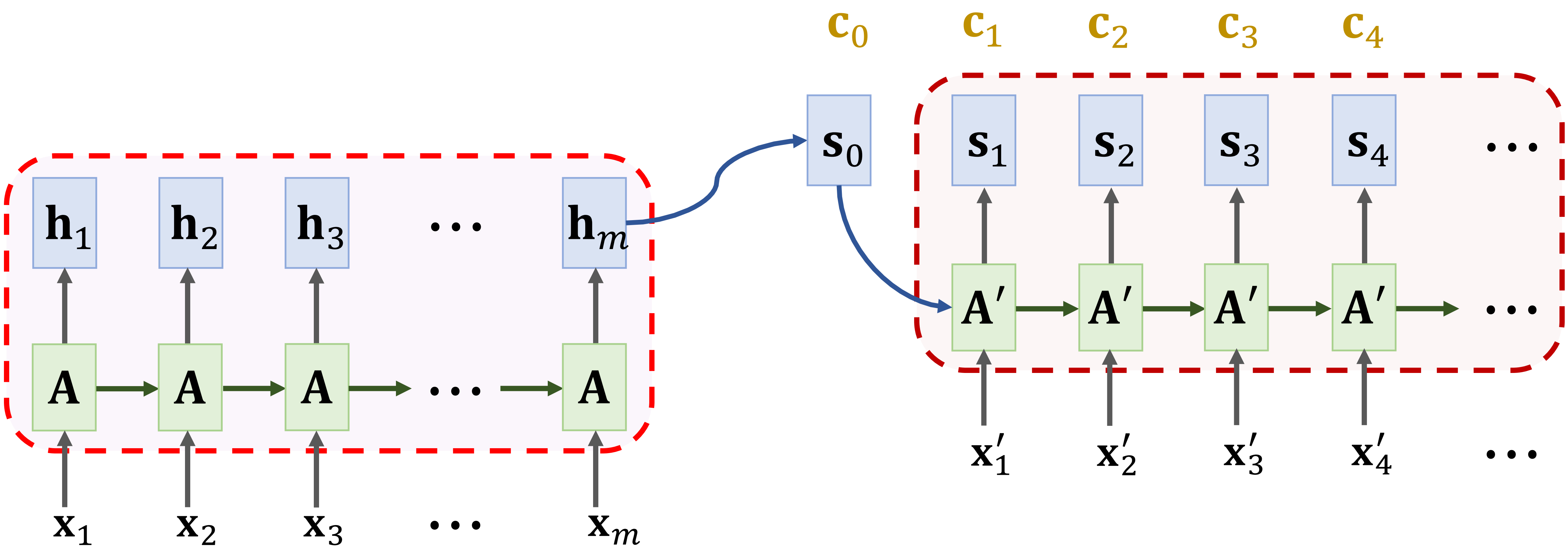
SimpleRNN + Attention



SimpleRNN + Attention

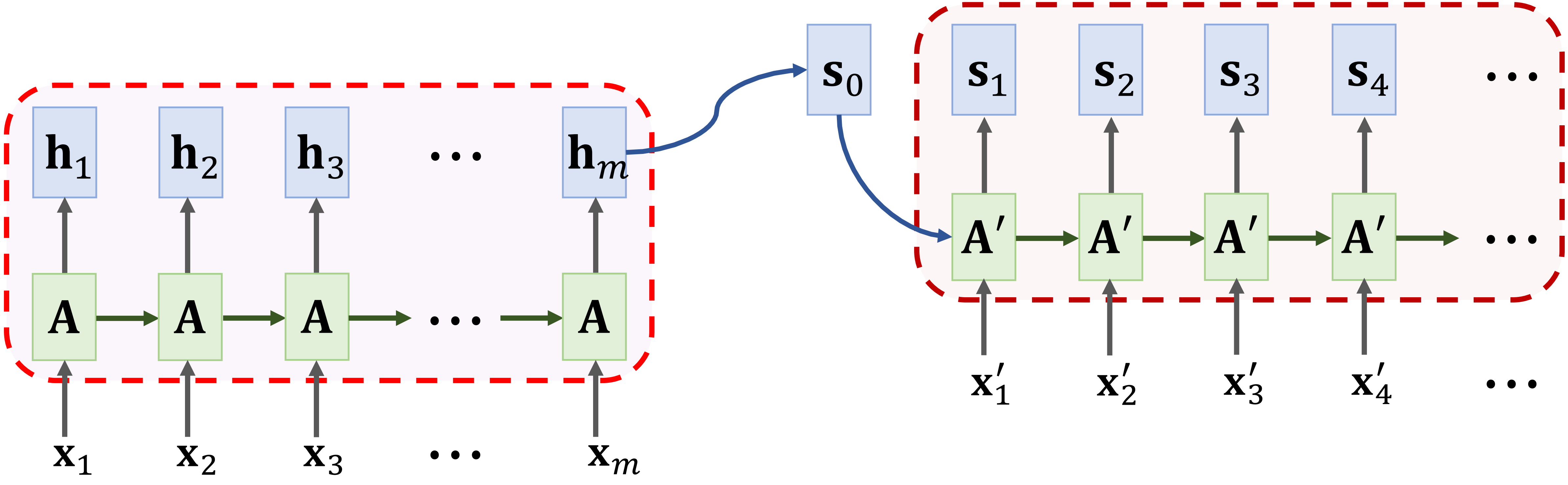


SimpleRNN + Attention



SimpleRNN + Attention

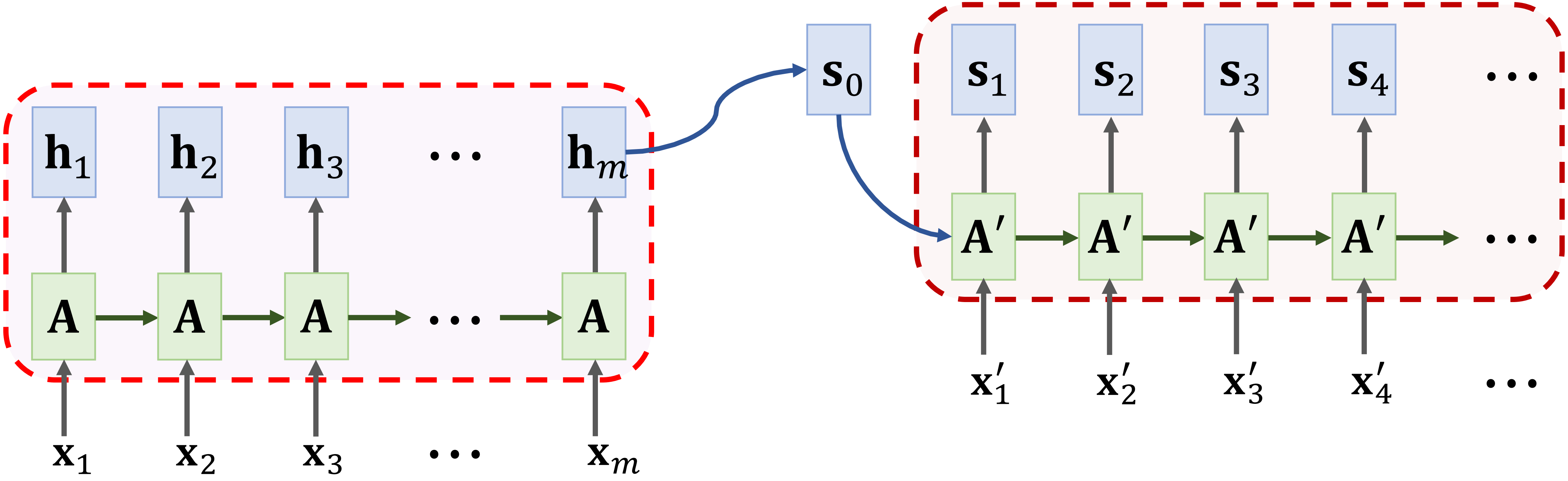
Question: How many weights α_i have been computed?



SimpleRNN + Attention

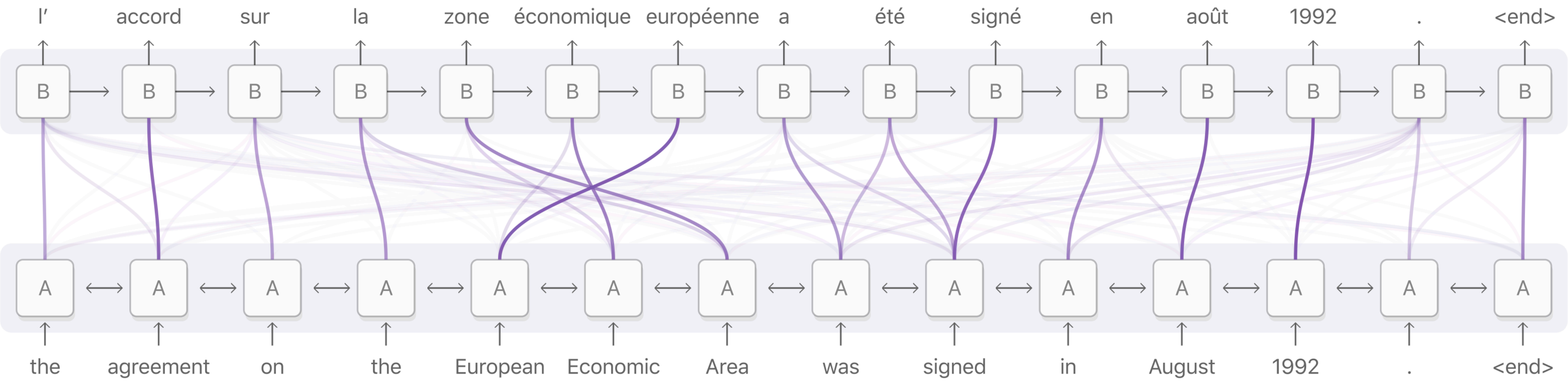
Question: How many weights have been computed?

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



Attention: Weights Visualization

Decoder RNN (target language: French)

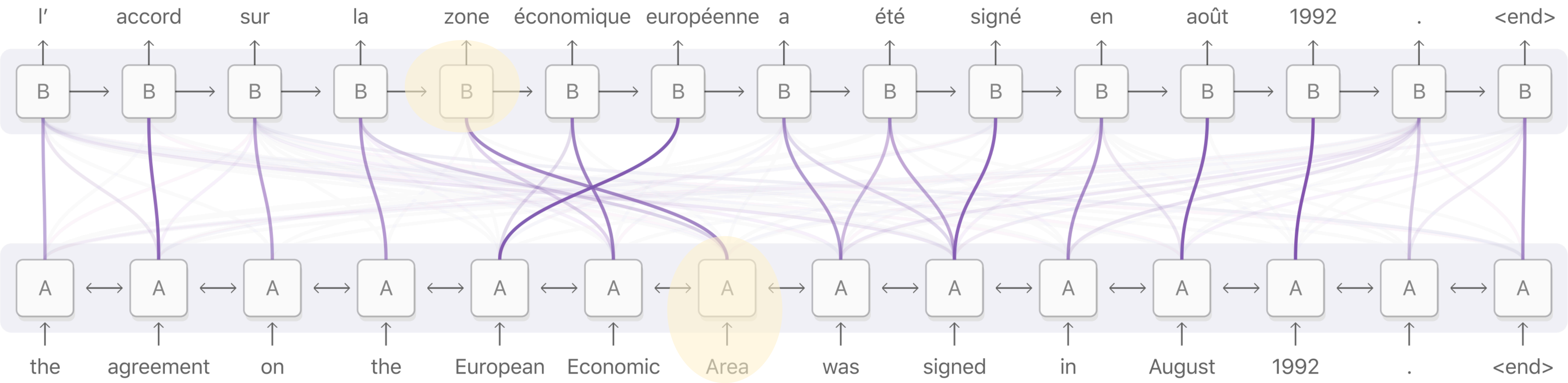


Encoder RNN (source language: English)

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

Attention: Weights Visualization

Decoder RNN (target language: French)

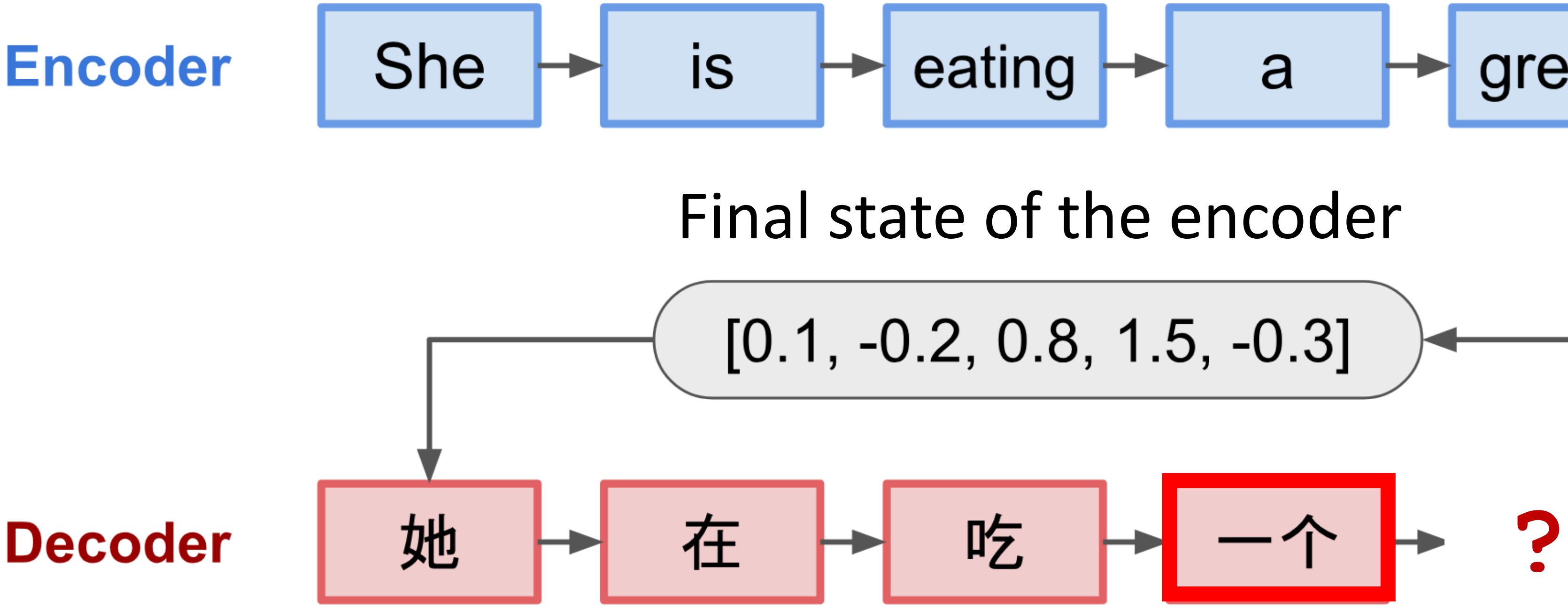


Encoder RNN (source language: English)

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

Summary

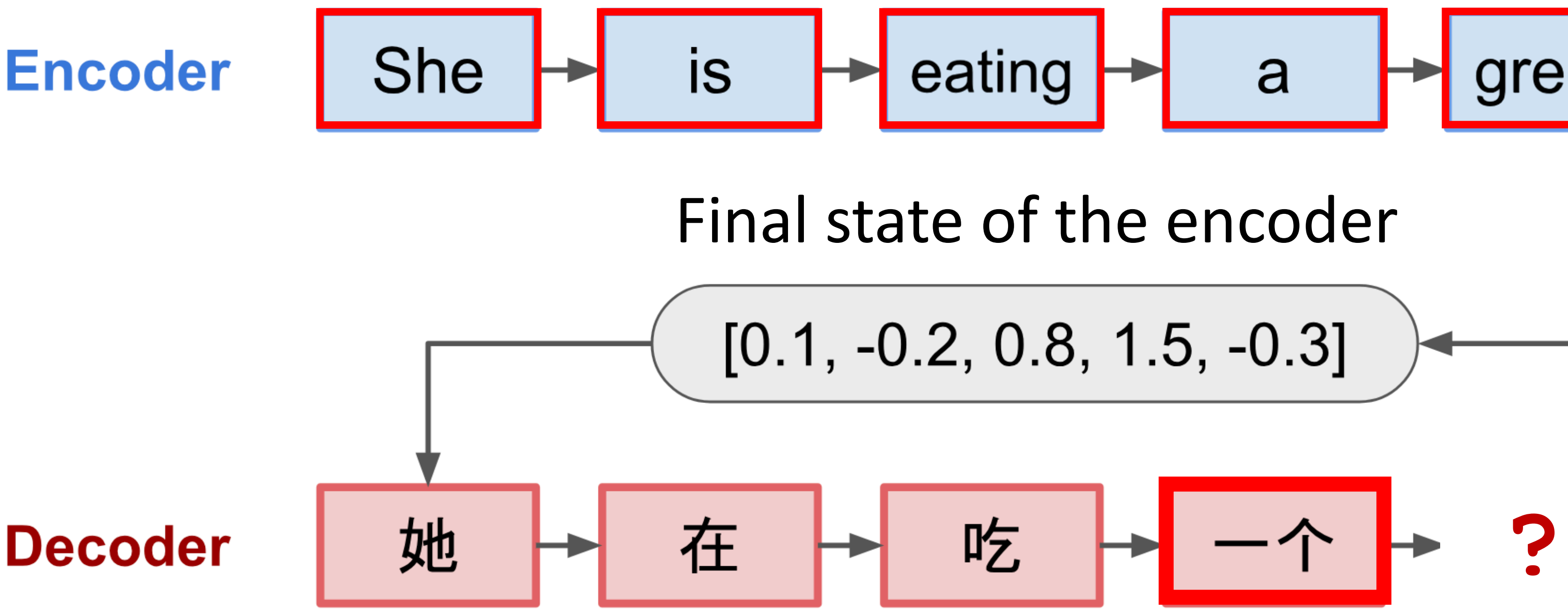
- Standard Seq2Seq model: the decoder looks at only its current state.



The figure is from blog lilianweng.github.io

Summary

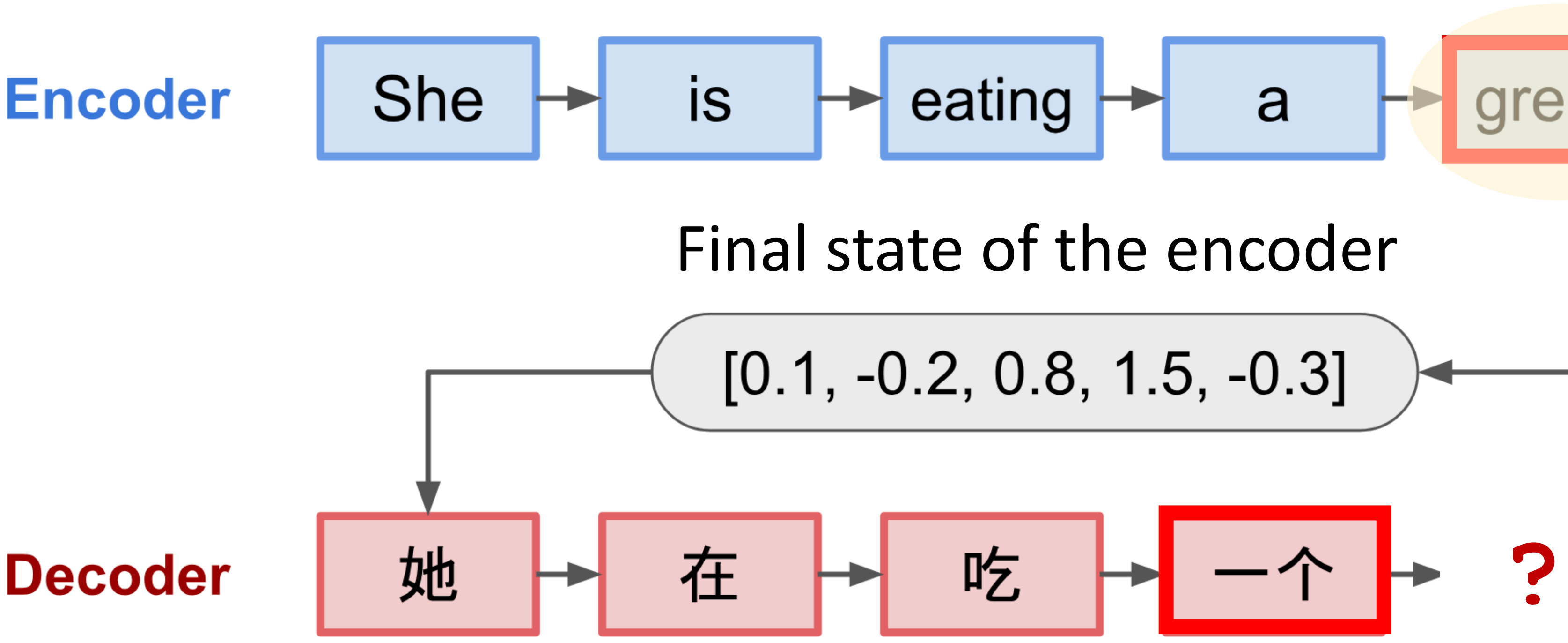
- Standard Seq2Seq model: the decoder looks at only its current state.
- Attention: decoder additionally looks at all the states of the encoder.



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.
- Attention: decoder knows where to focus on.



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.
- Attention: decoder knows where to focus on.

- Downside: higher time complexity.
  - m: source sequence length
  - t: target sequence length
  - Standard Seq2Seq: $O(m+t)$ time complexity
  - Seq2Seq + attention: $O(mt)$ time complexity

Thank you!