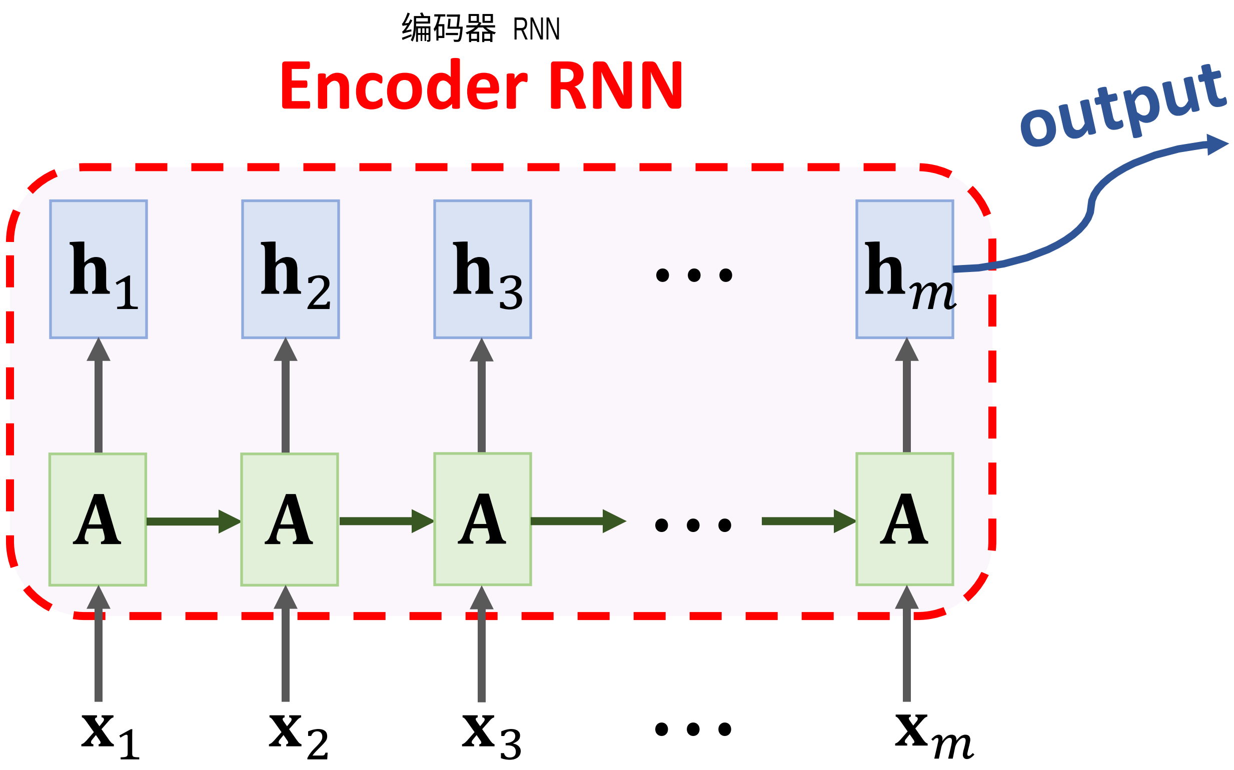
Attention

2015年提出
为了解决RNN的遗忘问题

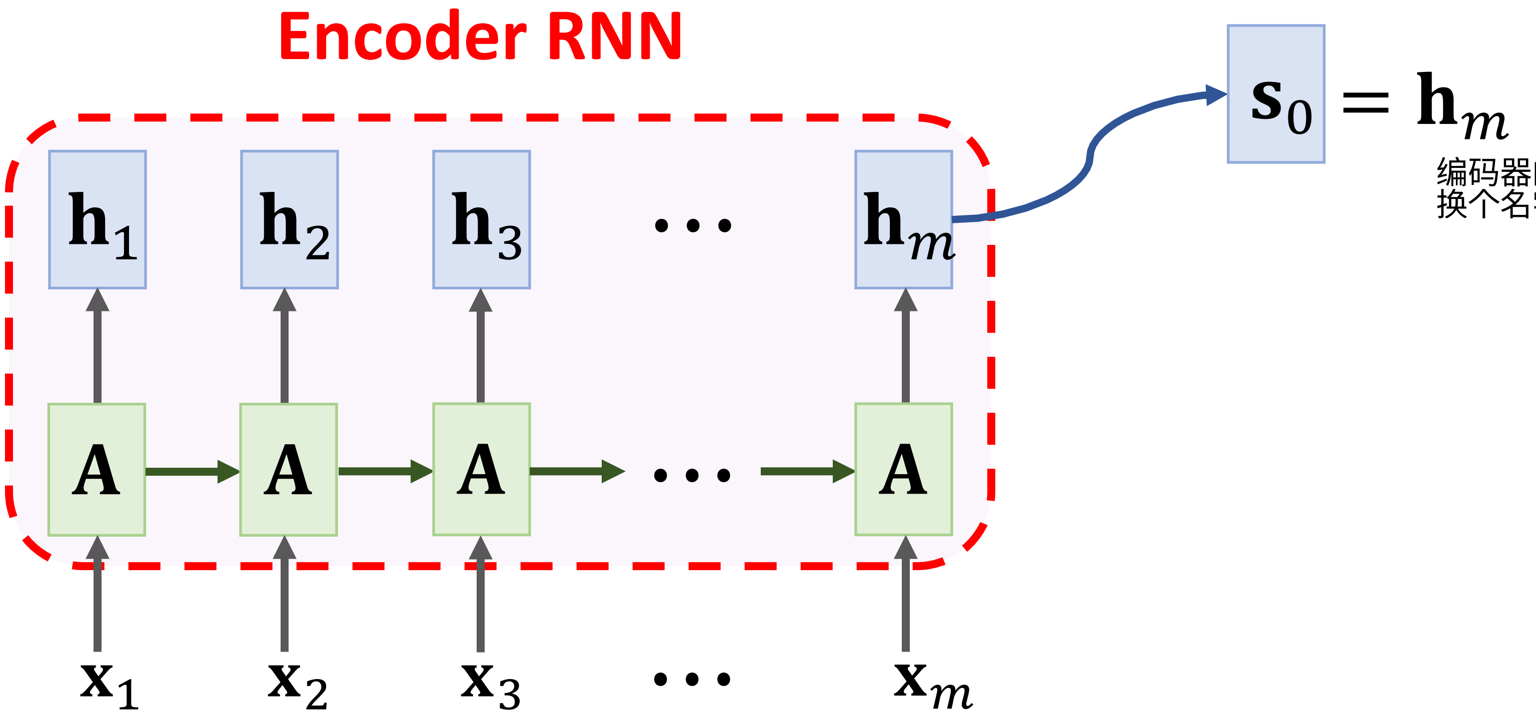
Shusen Wang

Revisiting Seq2Seq Model

Seq2Seq Model

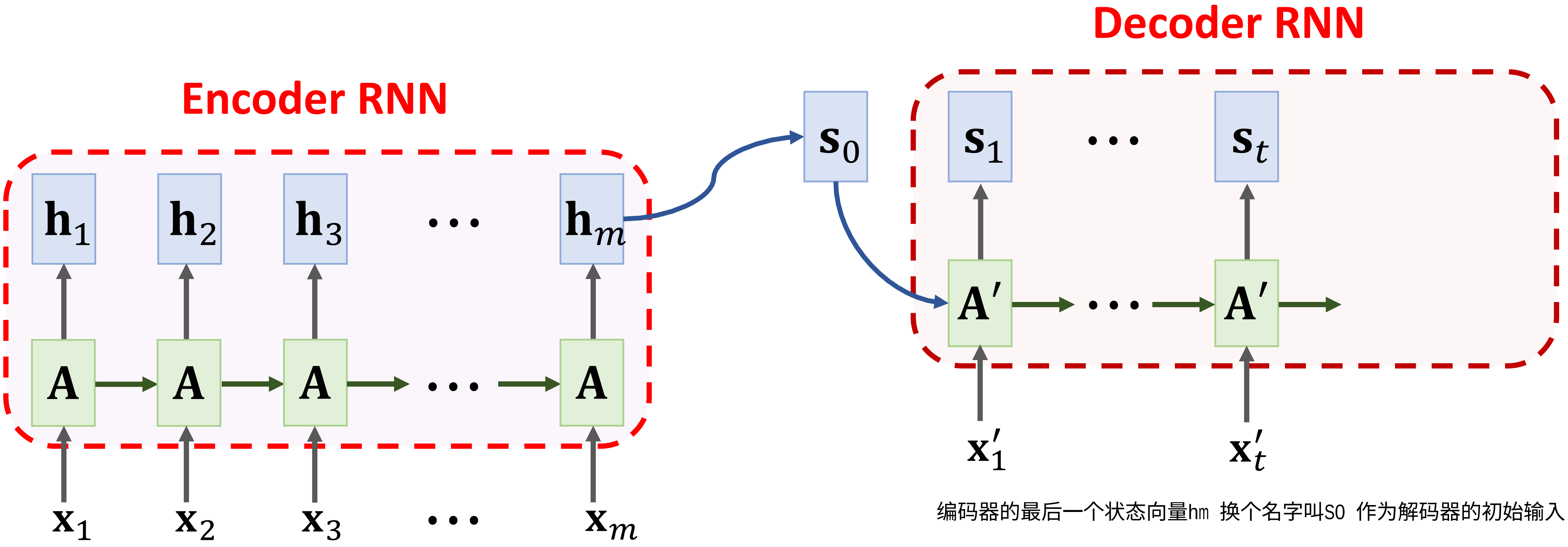


Seq2Seq Model



编码器的最后一个状态向量hm
换个名字叫S0 作为解码器的初始输入

Seq2Seq Model



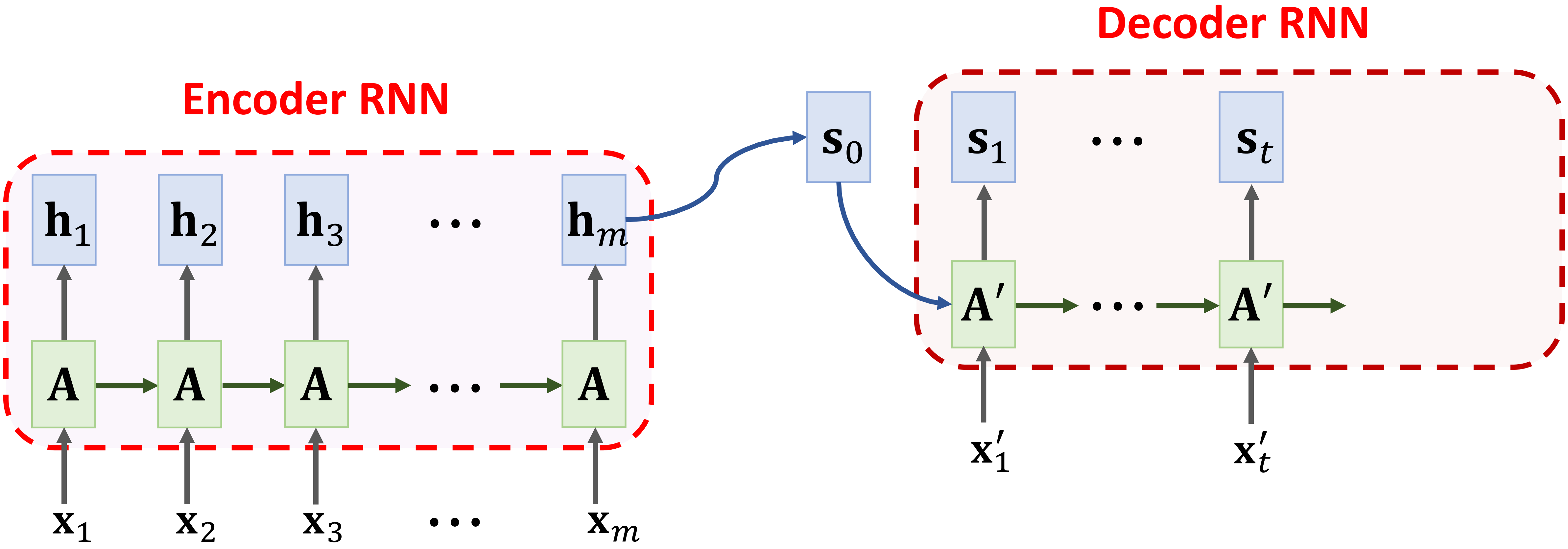
编码器的最后一个状态向量hm 换个名字叫S0 作为解码器的初始输入

Seq2Seq Model

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

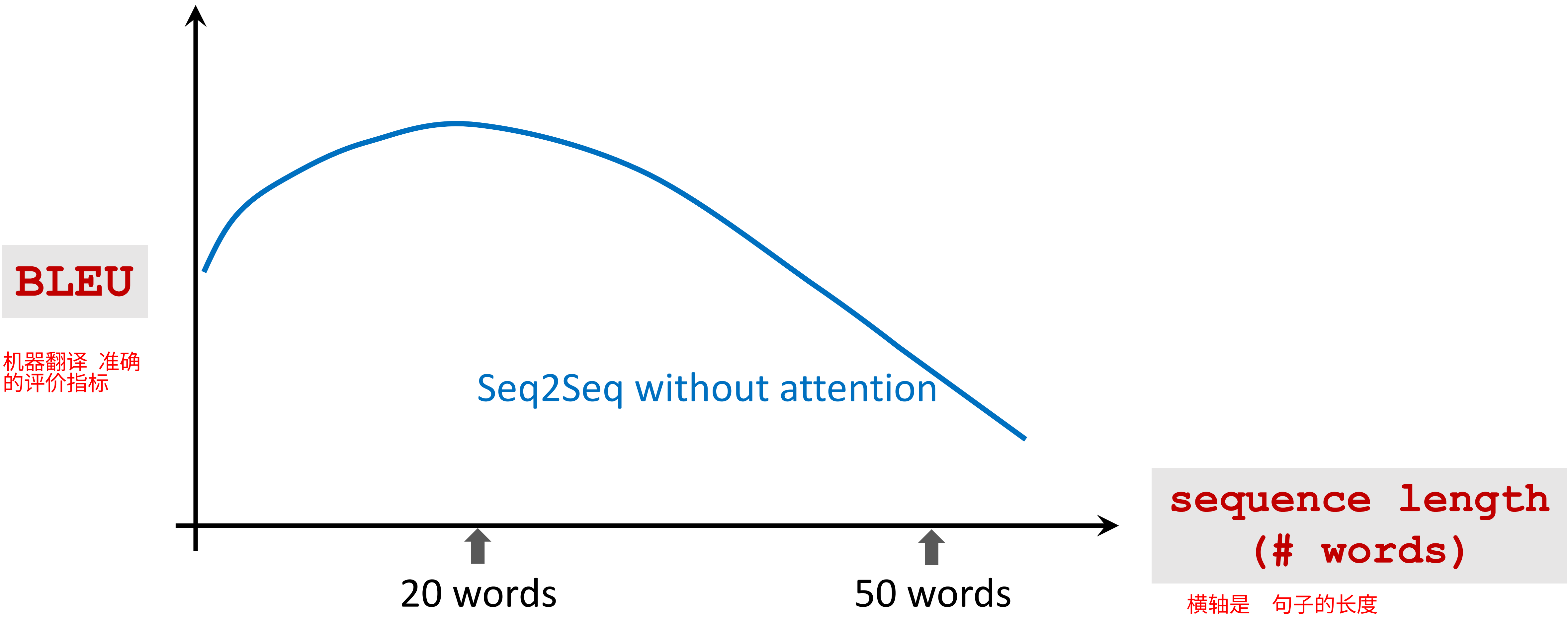
RNN缺点：最终状态无法记住一个长序列

最后的句子 翻译结果有丢失



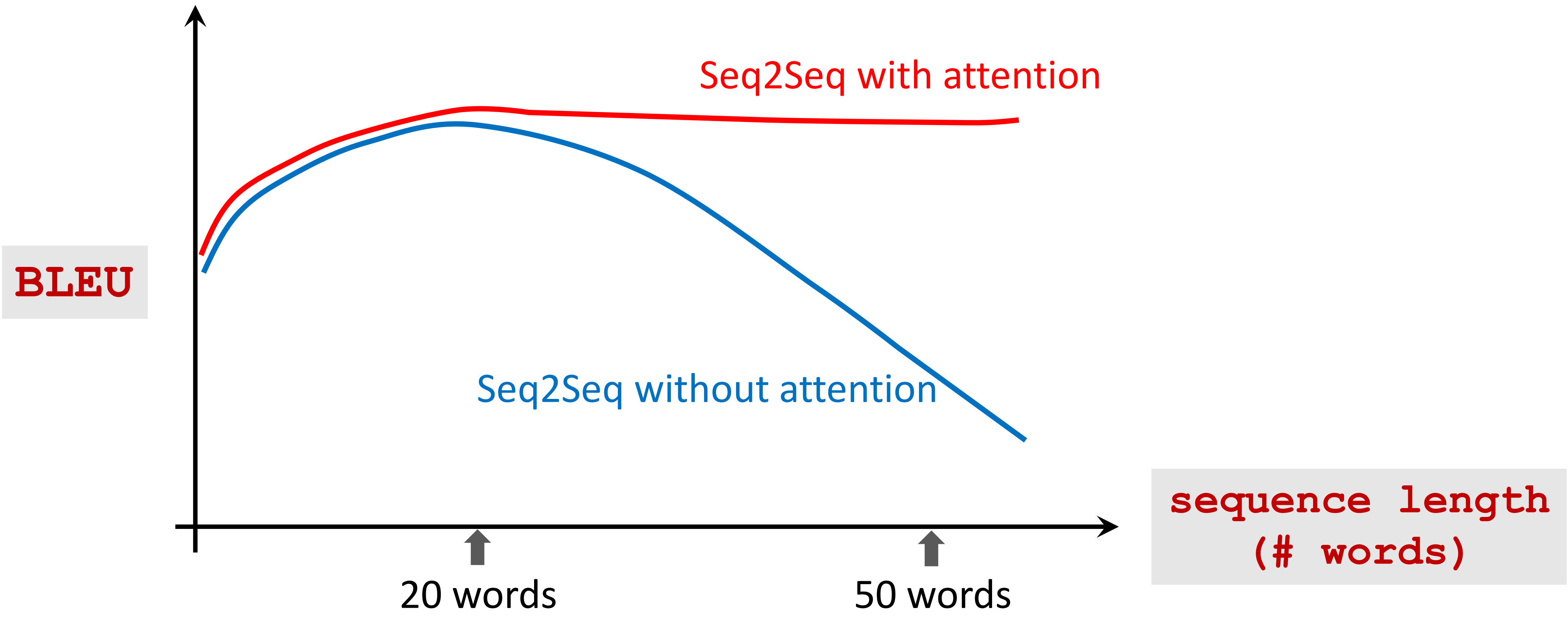
Seq2Seq Model

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



Seq2Seq Model

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



Attention for Seq2Seq Model

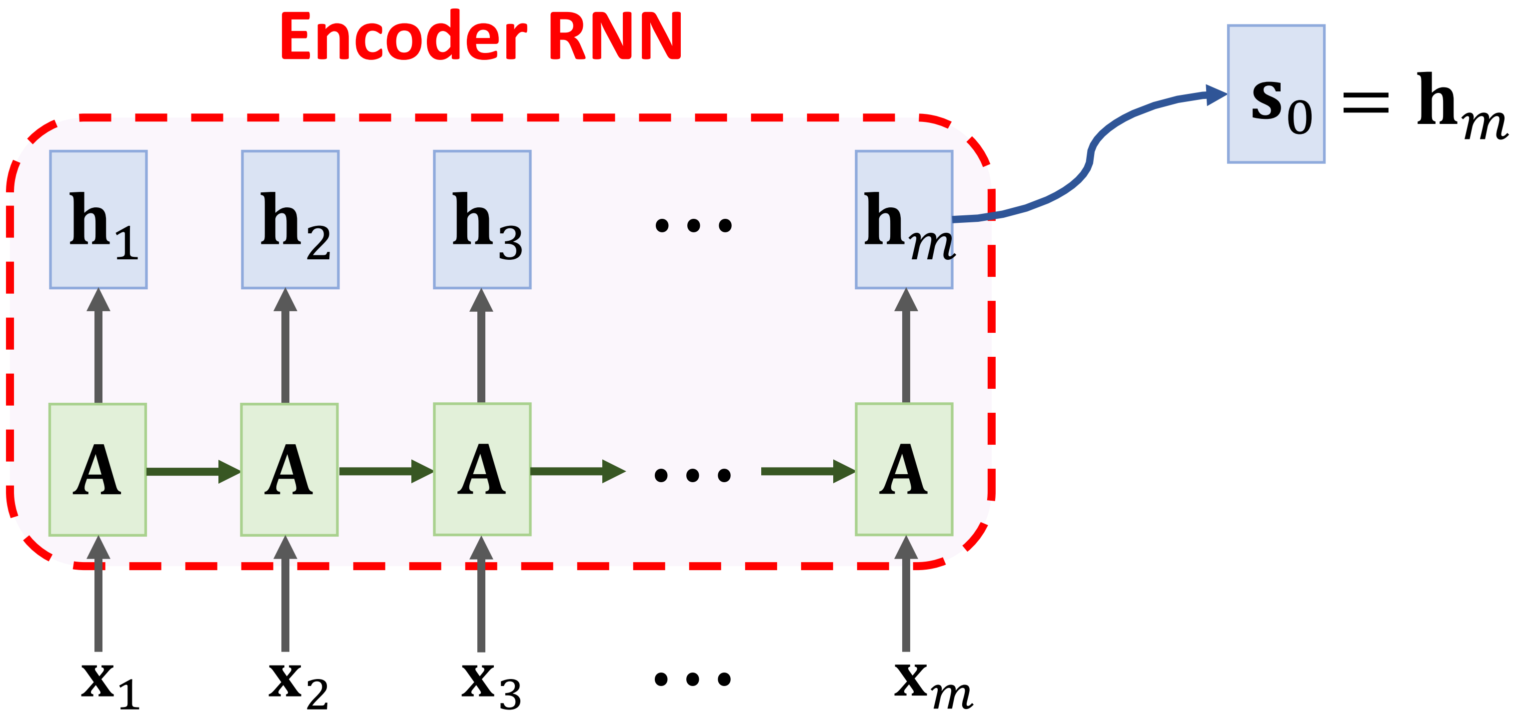
Seq2Seq Model with Attention

- Attention tremendously improves Seq2Seq model. Attention 极大地改进了Seq2Seq模型。
- With attention, Seq2Seq model does not forget source input. Attention使得Seq2Seq模型不会忘记源输入。
- With attention, the decoder knows where to focus. 注意力Attention 使得 解码器知道 重点在哪。
- Downside: much more computation. 缺点：更多的计算。

Original paper:

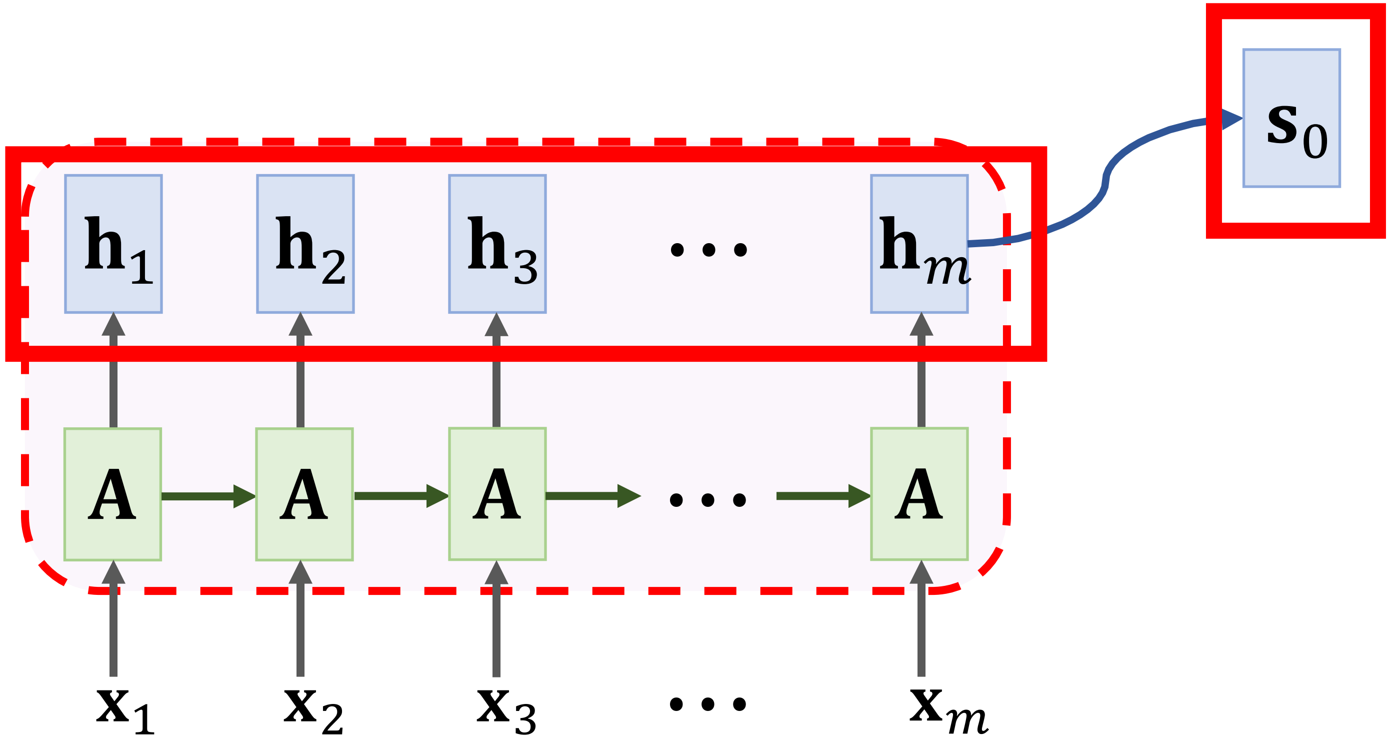
- Bahdanau, Cho, & Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

SimpleRNN + Attention



SimpleRNN + Attention

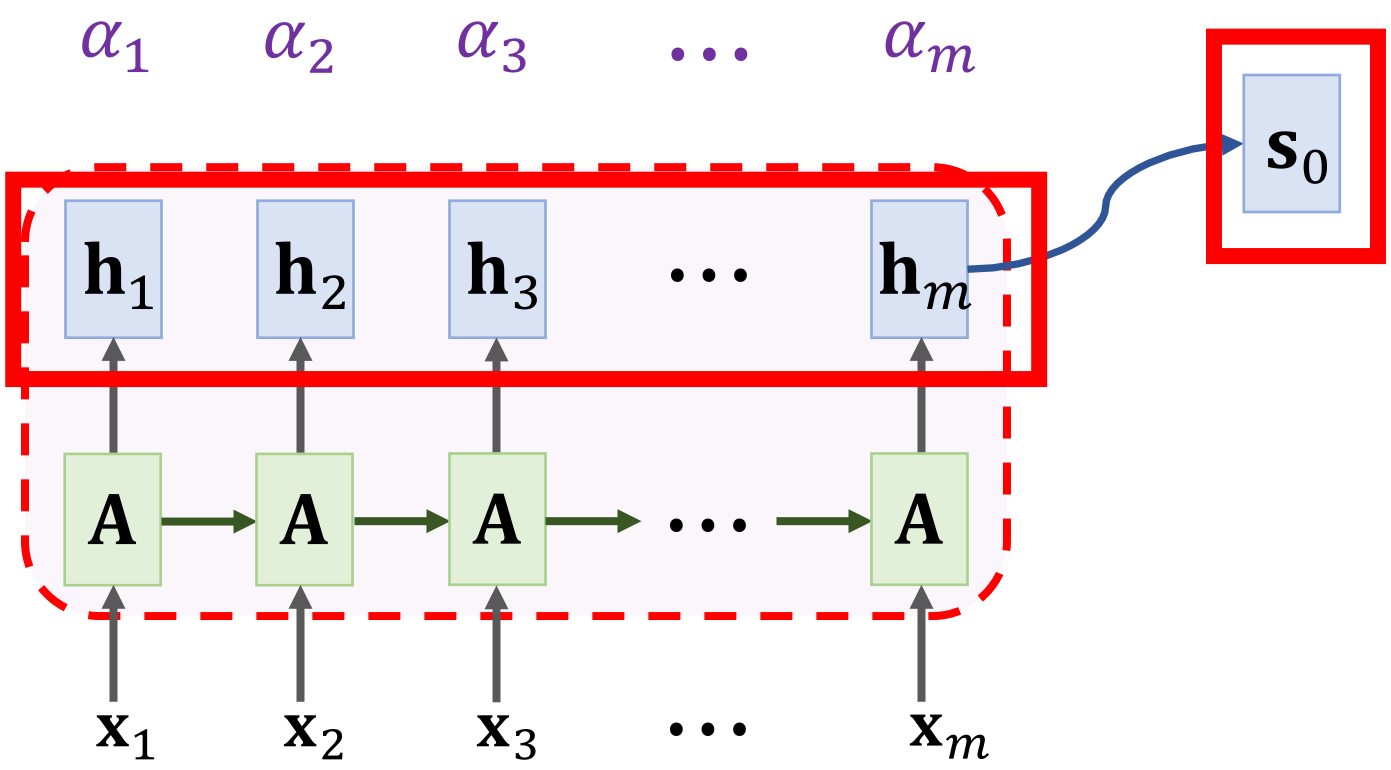
Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.



SimpleRNN + Attention

Weight: $\alpha_i = \mathrm{align}(\mathbf{h}_i, \mathbf{s}_0).$

计算权重α，α1 ... αm 的加和 为1



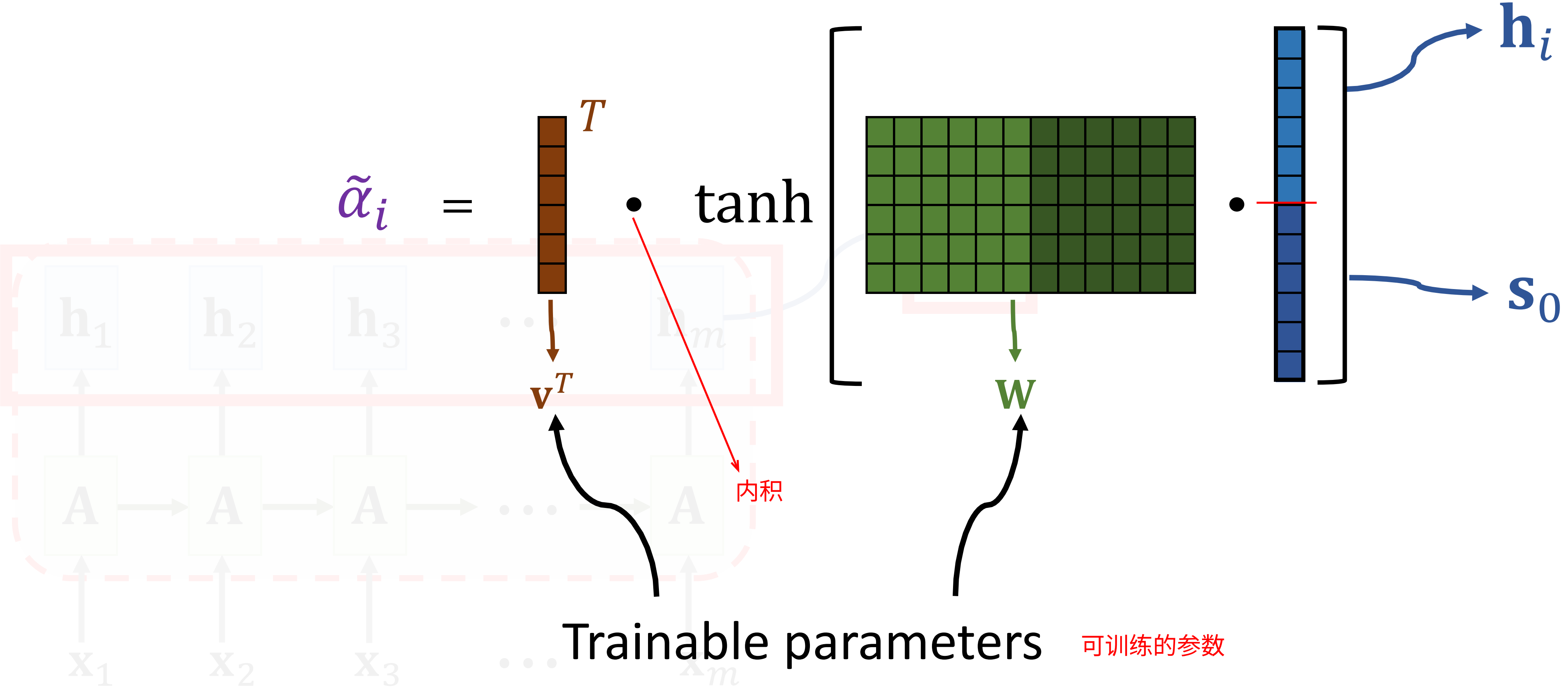
重点

SimpleRNN + Attention

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$. 权重的计算方法 有很多种

Option 1 (used in the original paper): 方法一：论文中最初始的计算方法

参数是固定的



Trainable parameters 可训练的参数

SimpleRNN + Attention

Weight: $\alpha_i = \mathrm{align}(\mathbf{h}_i, \mathbf{s}_0)$.

Option 1 (used in the original paper):

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

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

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

满足加和为1

SimpleRNN + Attention

重点：

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

Option 2 (more popular; the same to Transformer):

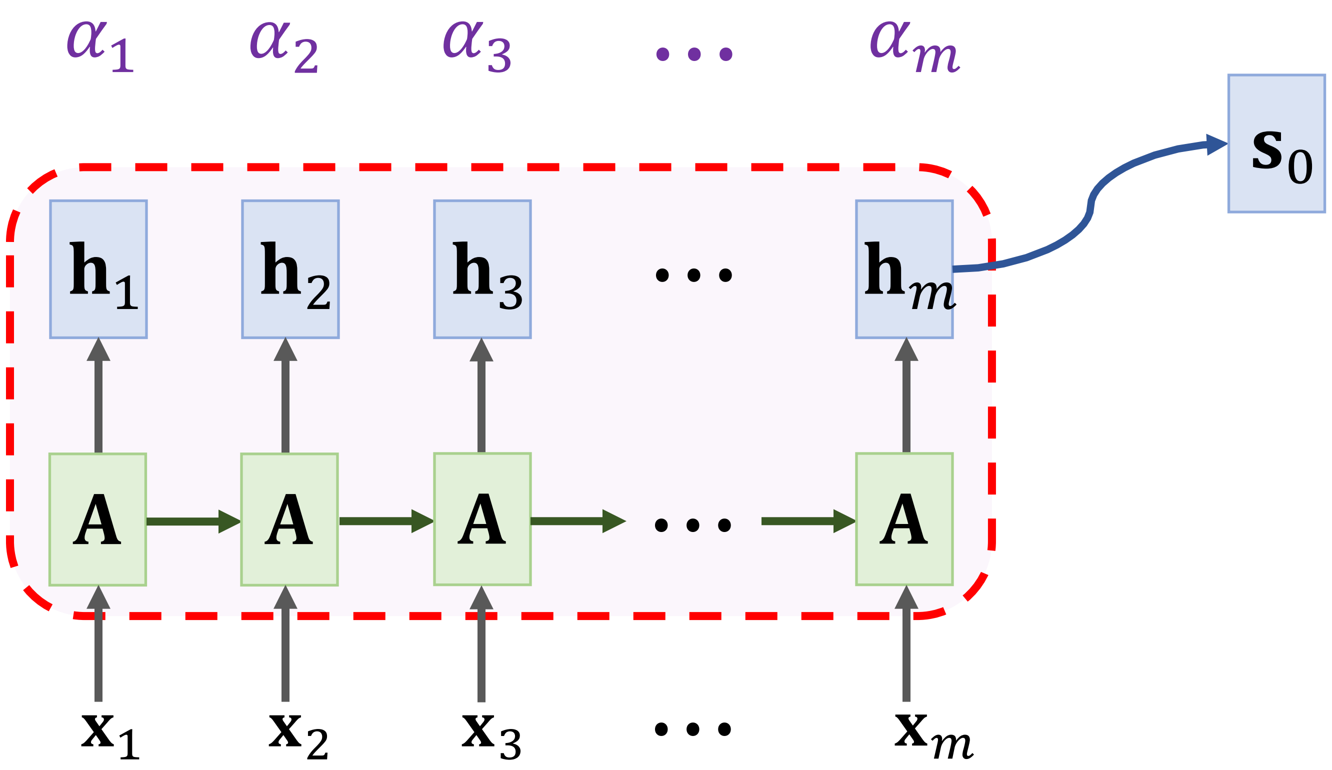
方法2 ：目前更受欢迎
和Transformer类似

参数可学习

1. Linear maps: 用两个矩阵 对输入做线性变换
   输入
   - $\mathbf{k}_i = \mathbf{W}_K \cdot \mathbf{h}_i$, for $i = 1$ to m.
   - $\mathbf{q}_0 = \mathbf{W}_Q \cdot \mathbf{s}_0$.
2. Inner product: 计算内积
   - $\tilde{\alpha}_i = \mathbf{k}_i^T \mathbf{q}_0$, for $i = 1$ to m.
3. Normalization: 标准化
   - $[\alpha_1, \cdots, \alpha_m] = \text{Softmax}\left([\tilde{\alpha}_1, \cdots, \tilde{\alpha}_m]\right)$.

SimpleRNN + Attention

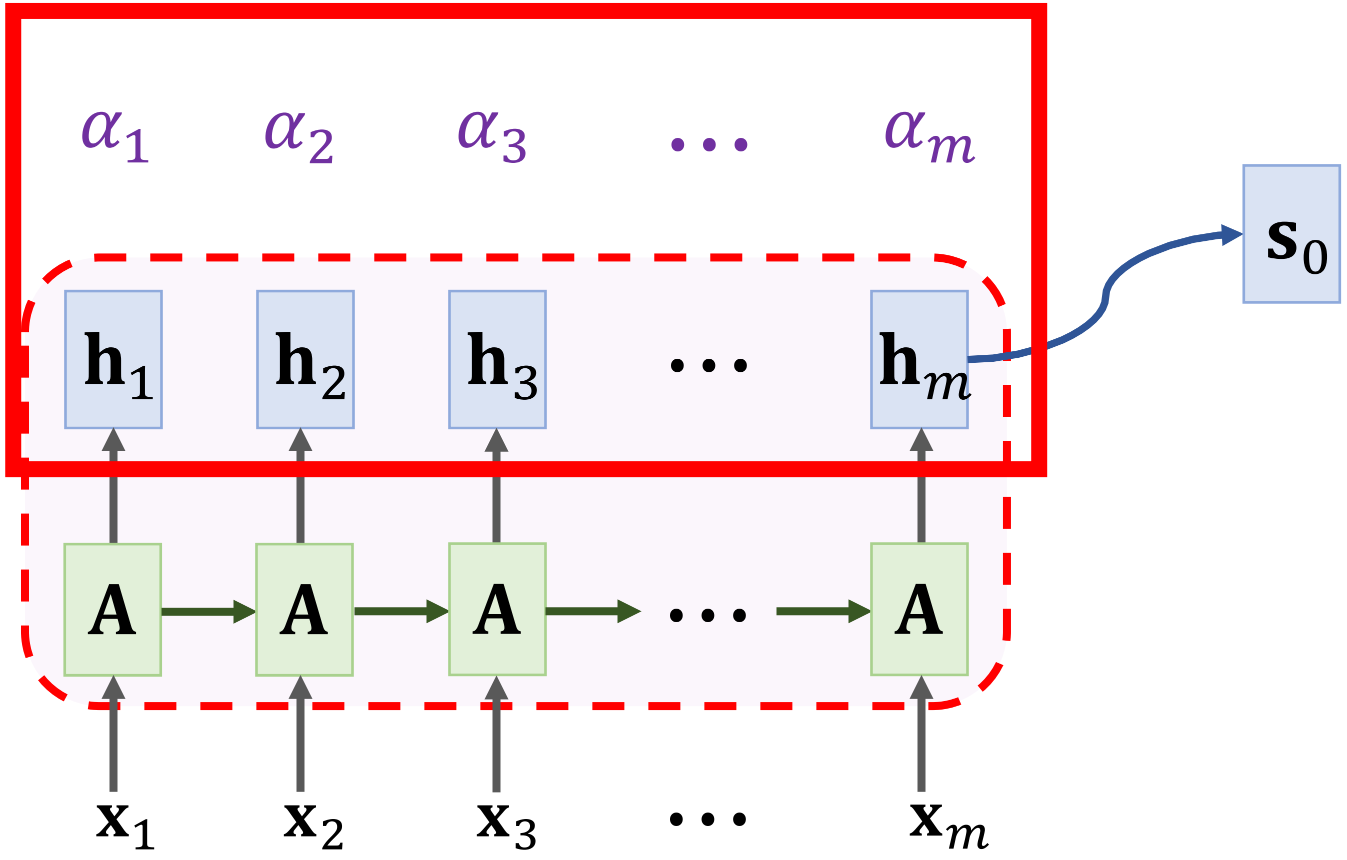
Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0).$



SimpleRNN + Attention

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0).$

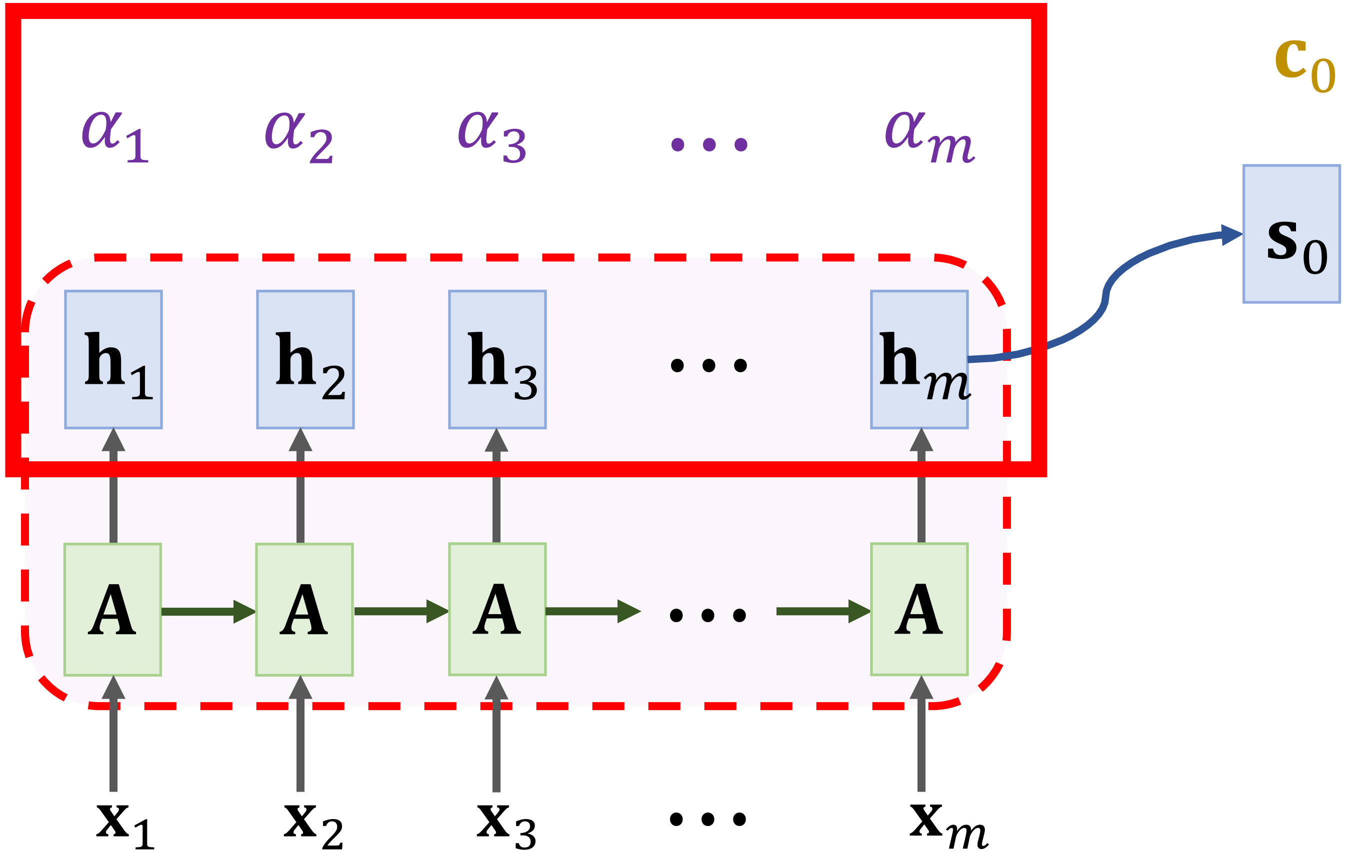
上下文向量 **Context vector:** $\mathbf{c}_0 = \alpha_1 \mathbf{h}_1 + \cdots + \alpha_m \mathbf{h}_m.$



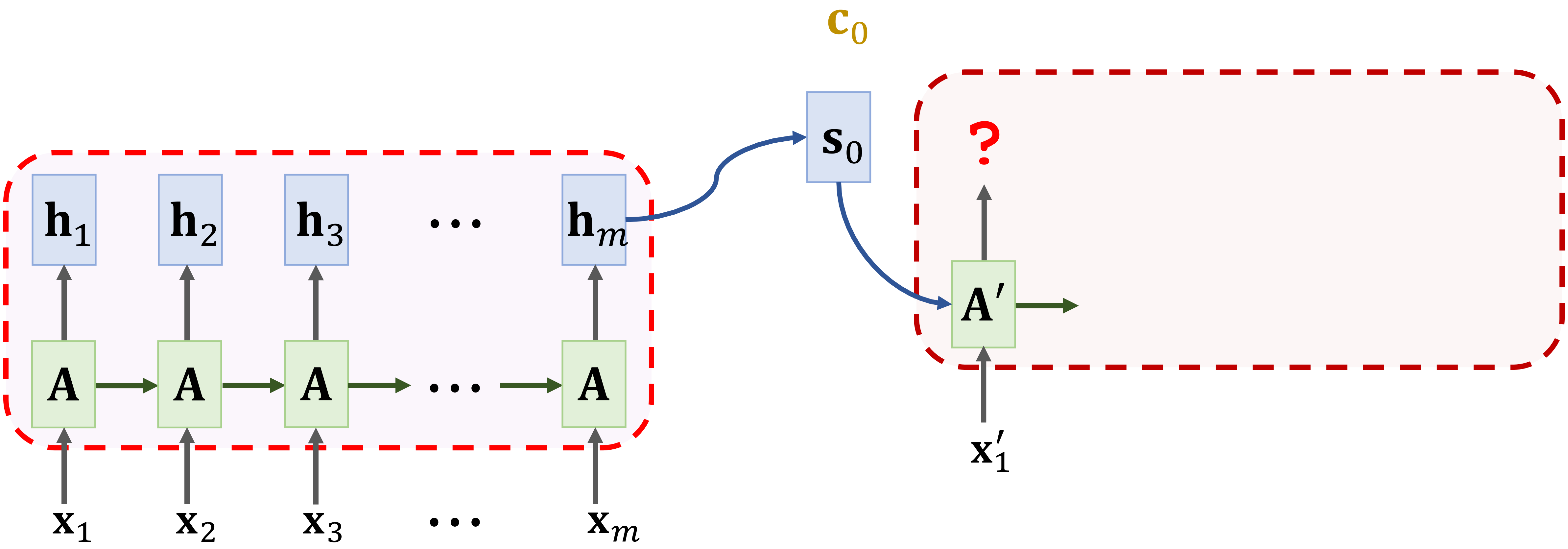
SimpleRNN + Attention

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0).$

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



SimpleRNN + Attention

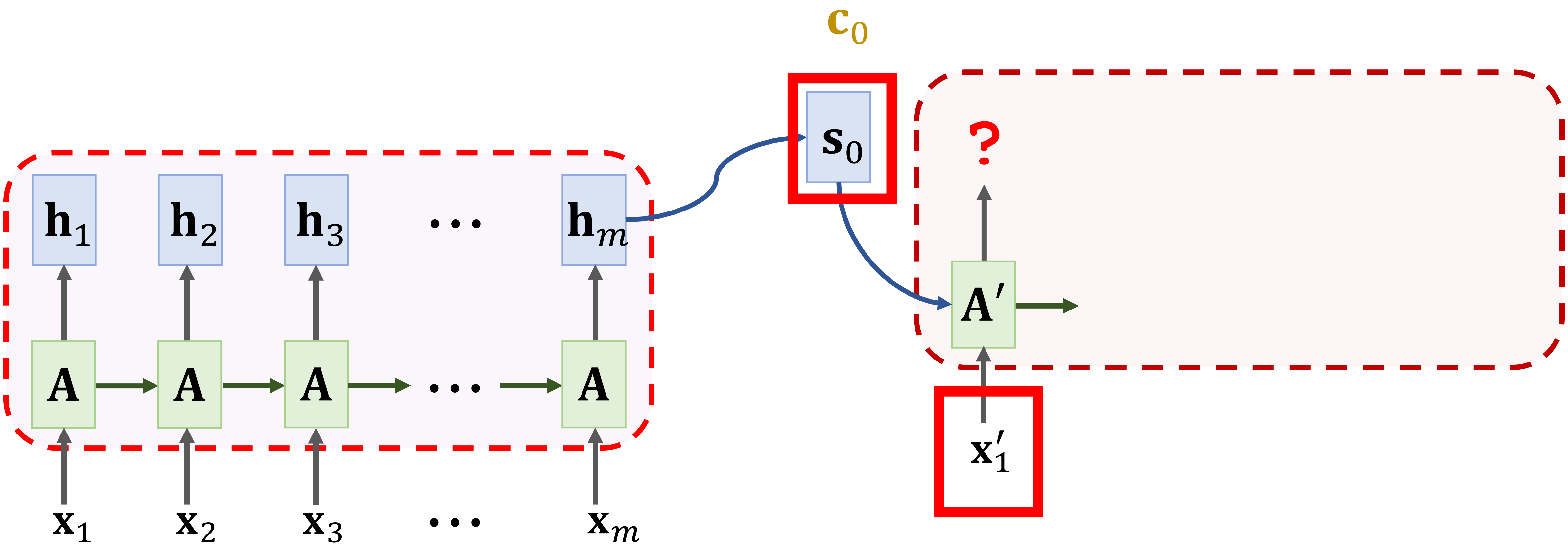


SimpleRNN

SimpleRNN: 的 计算输出的方法

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

concat



SimpleRNN + Attention

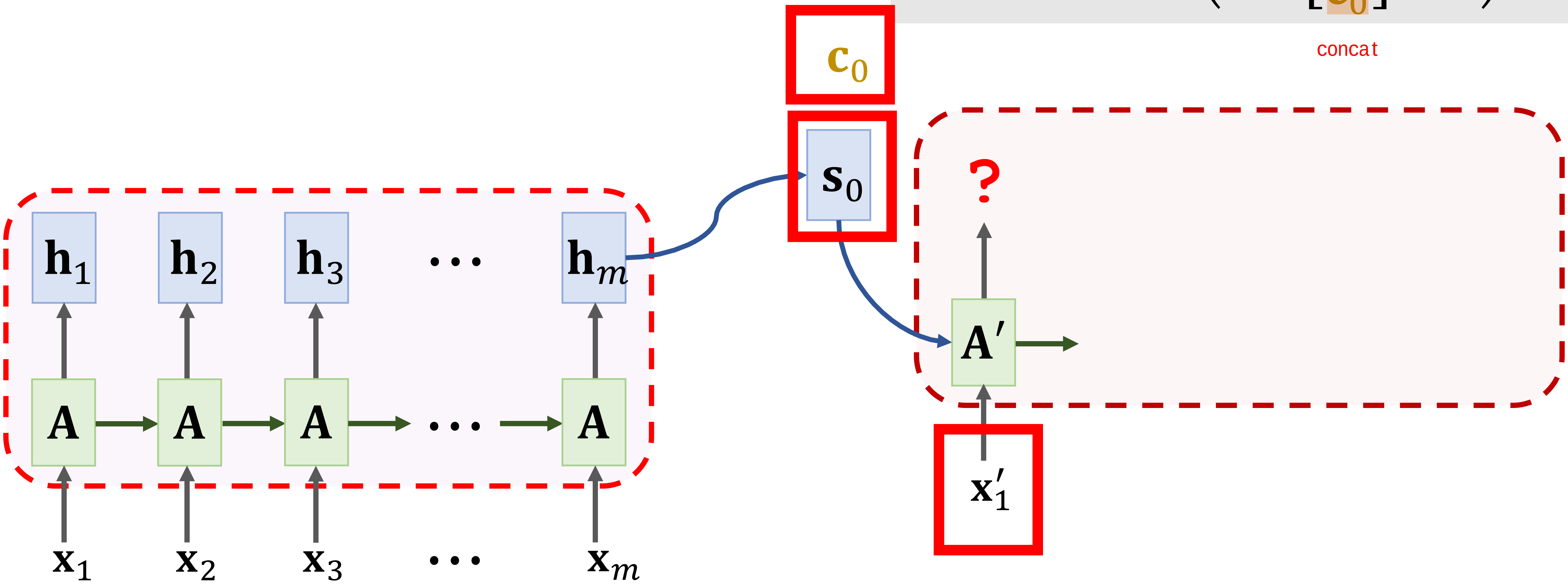
SimpleRNN:

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

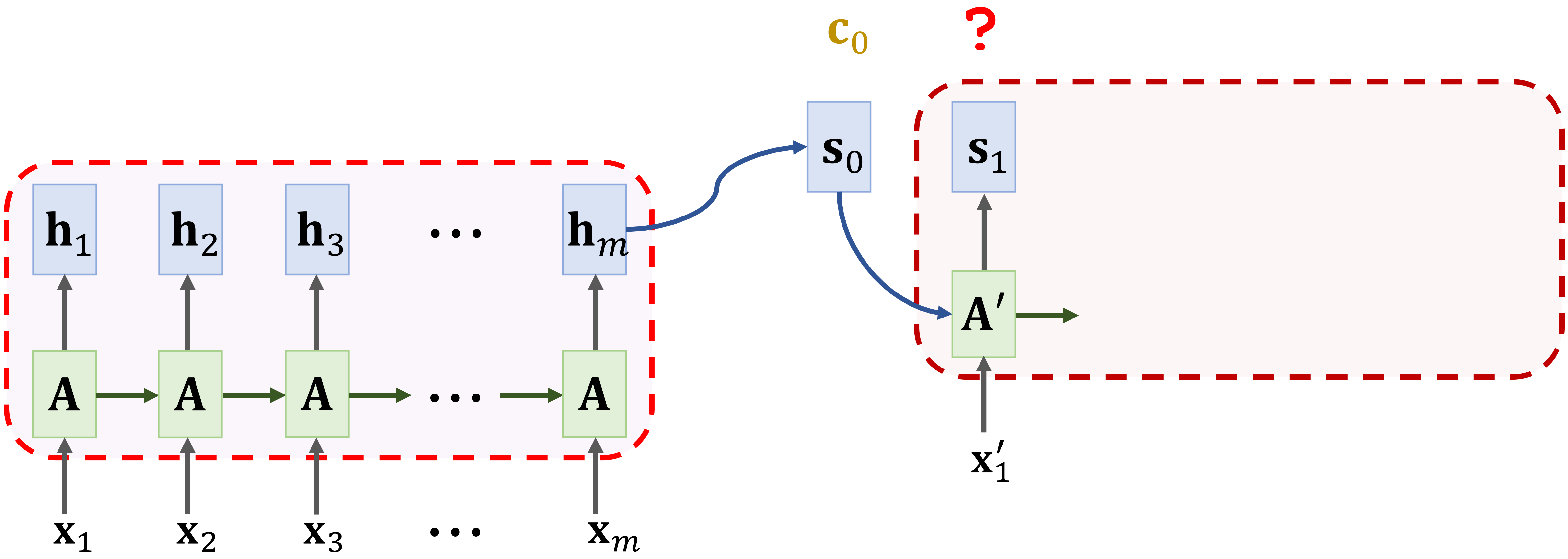
SimpleRNN + Attention:

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

concat

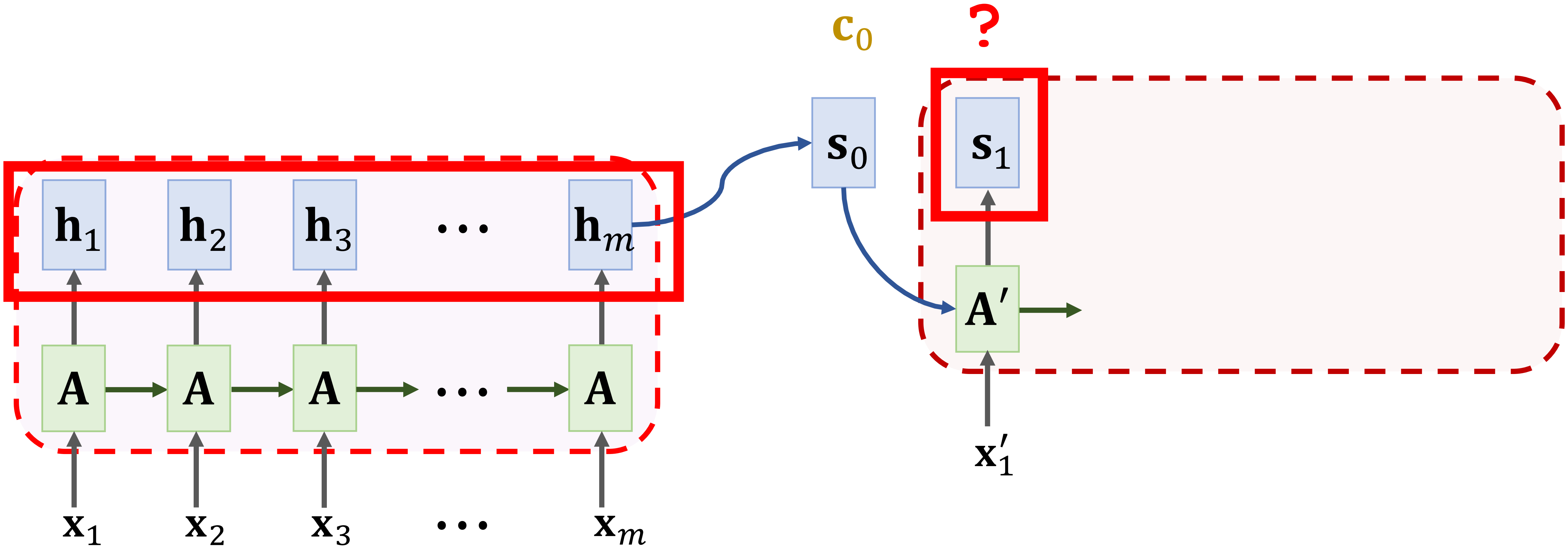


SimpleRNN + Attention



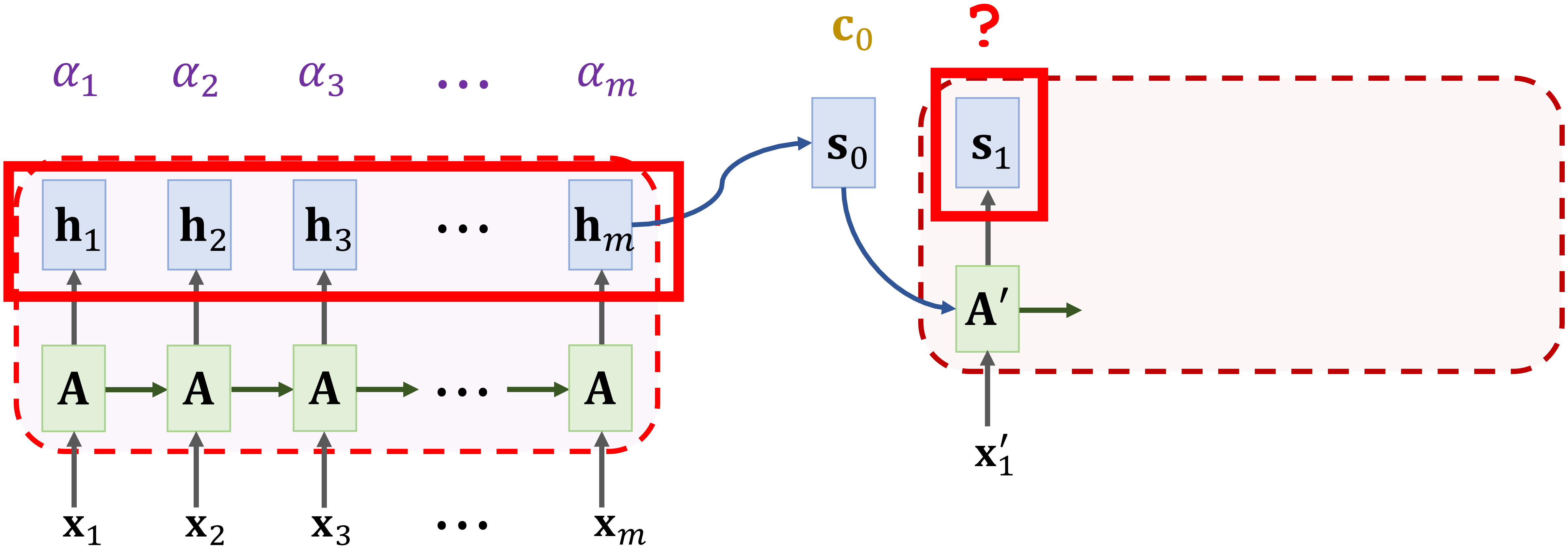
SimpleRNN + Attention

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_1)$.



SimpleRNN + Attention

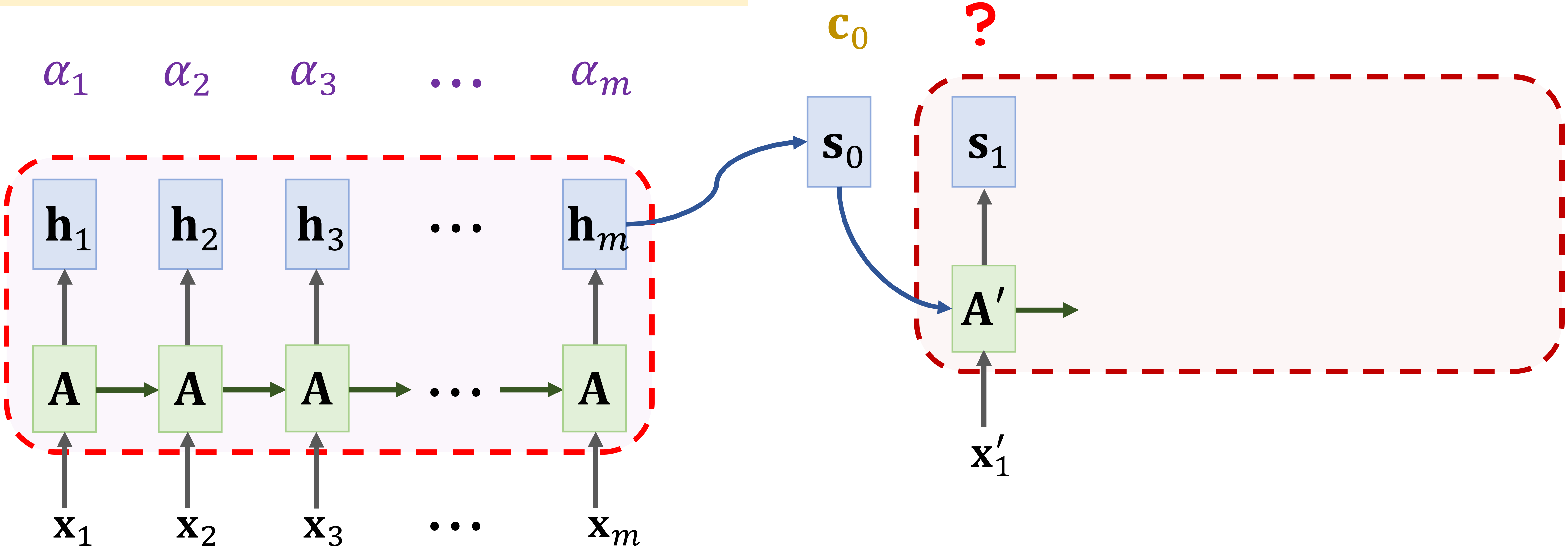
Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_1)$.



SimpleRNN + Attention

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_1).$

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

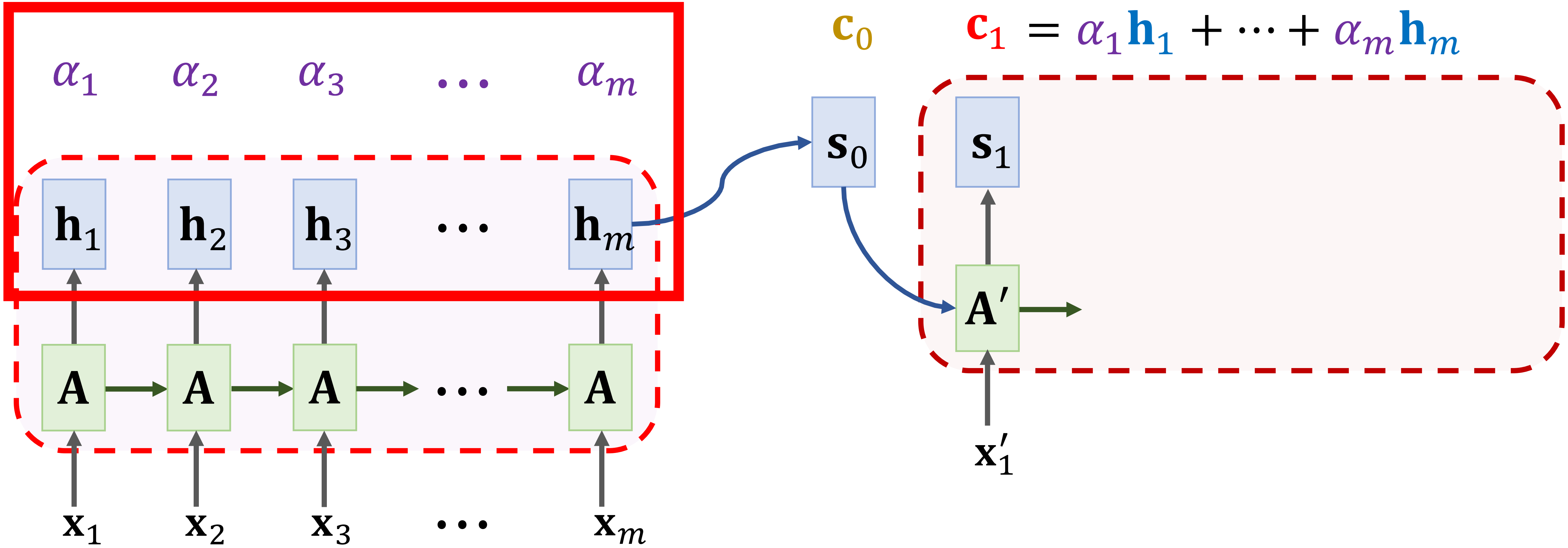


SimpleRNN + Attention

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_1)$. 与 s1的相关性

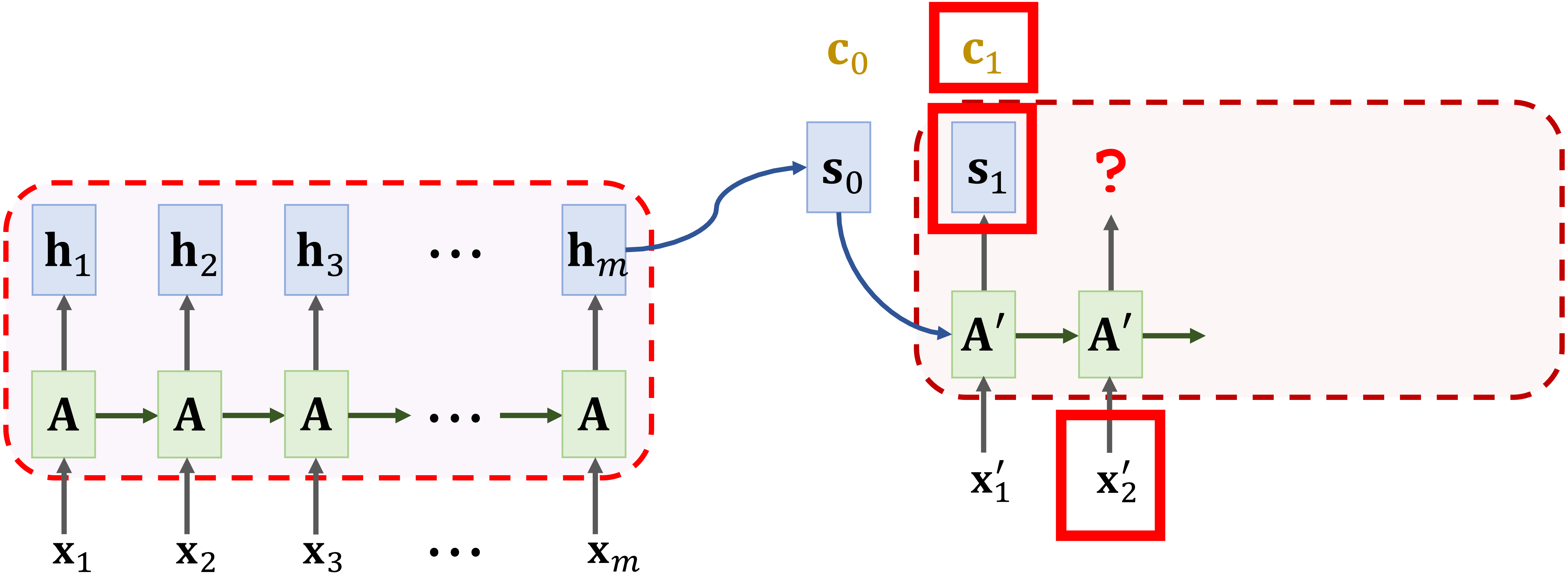
重新计算权值

$\mathbf{c}_1 = \alpha_1 \mathbf{h}_1 + \cdots + \alpha_m \mathbf{h}_m$

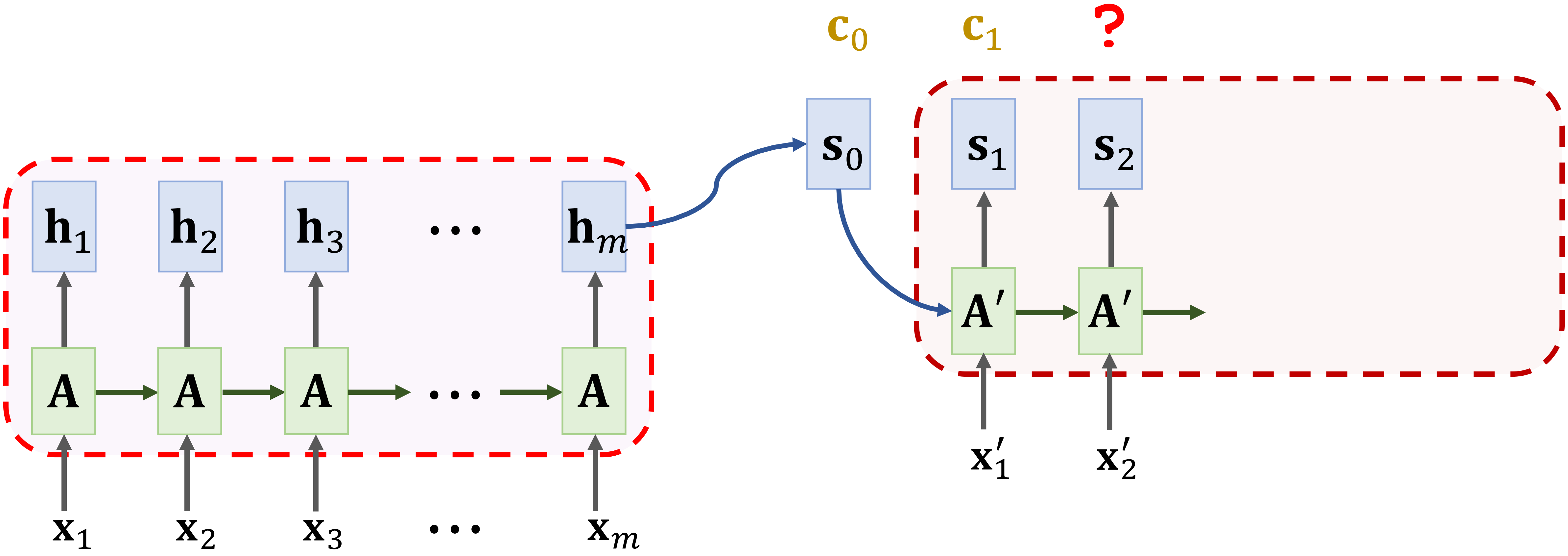


SimpleRNN + Attention

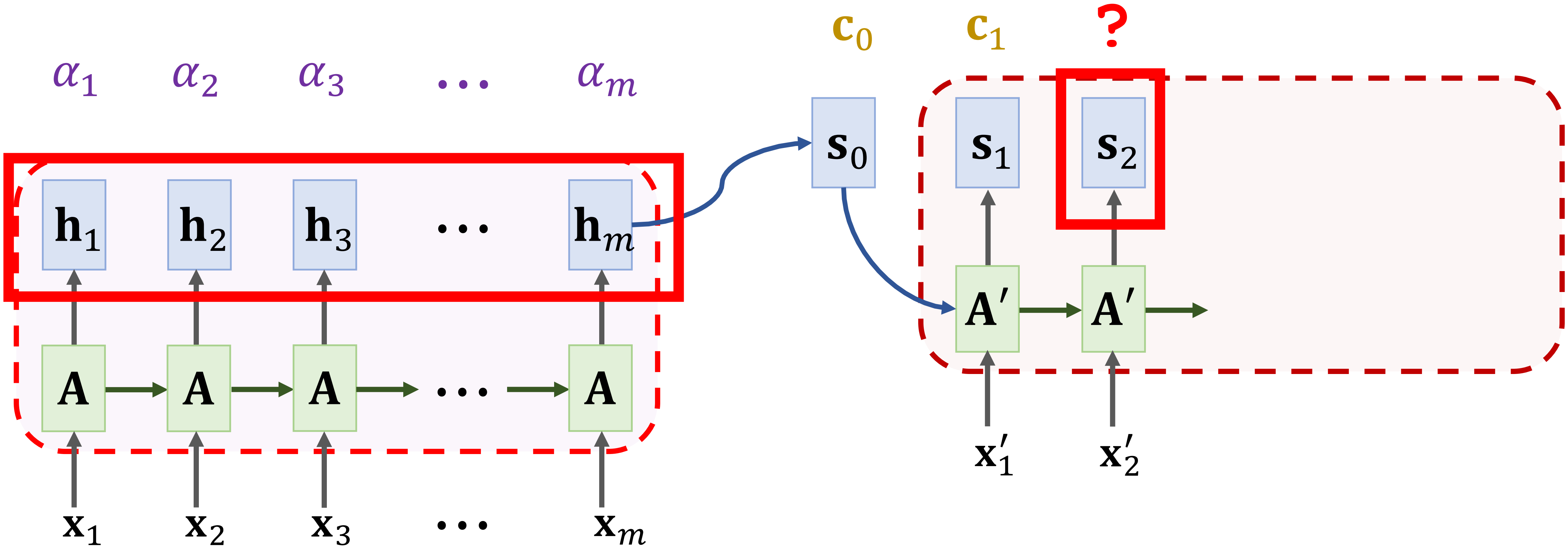
$$\mathbf{s}_2 = \tanh\left(\mathbf{A}' \cdot \begin{bmatrix}\mathbf{x}'_2 \\ \mathbf{s}_1 \\ \mathbf{c}_1\end{bmatrix} + \mathbf{b}\right)$$



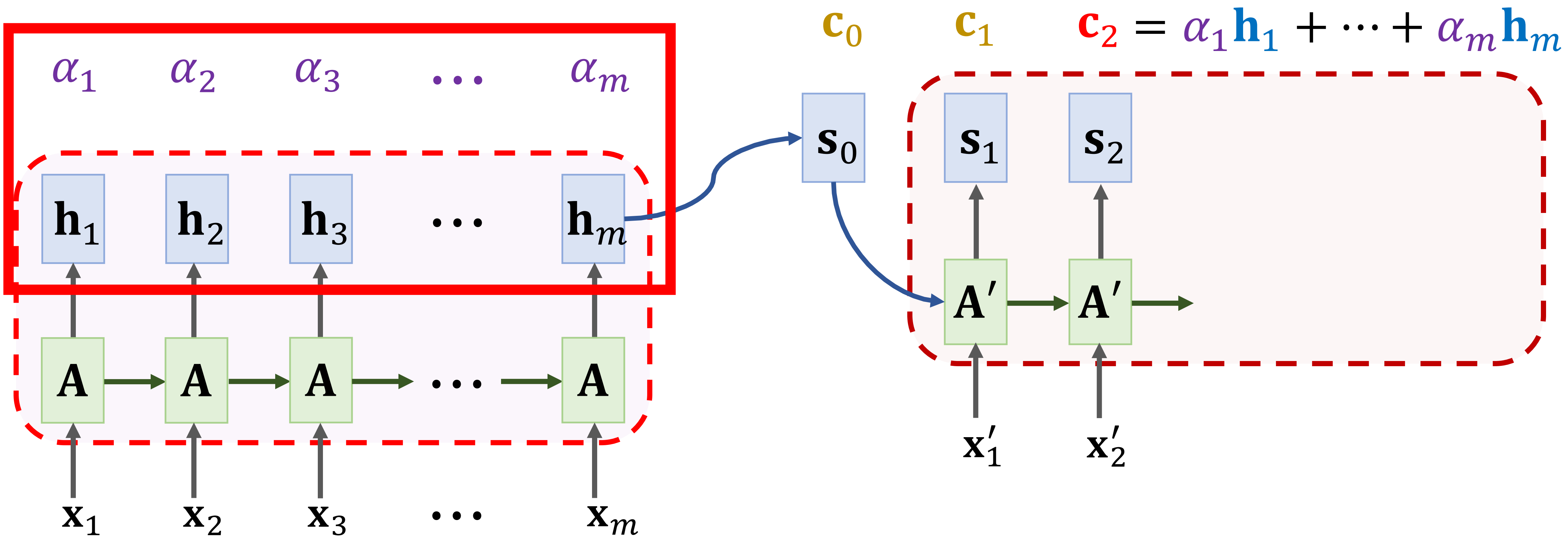
SimpleRNN + Attention



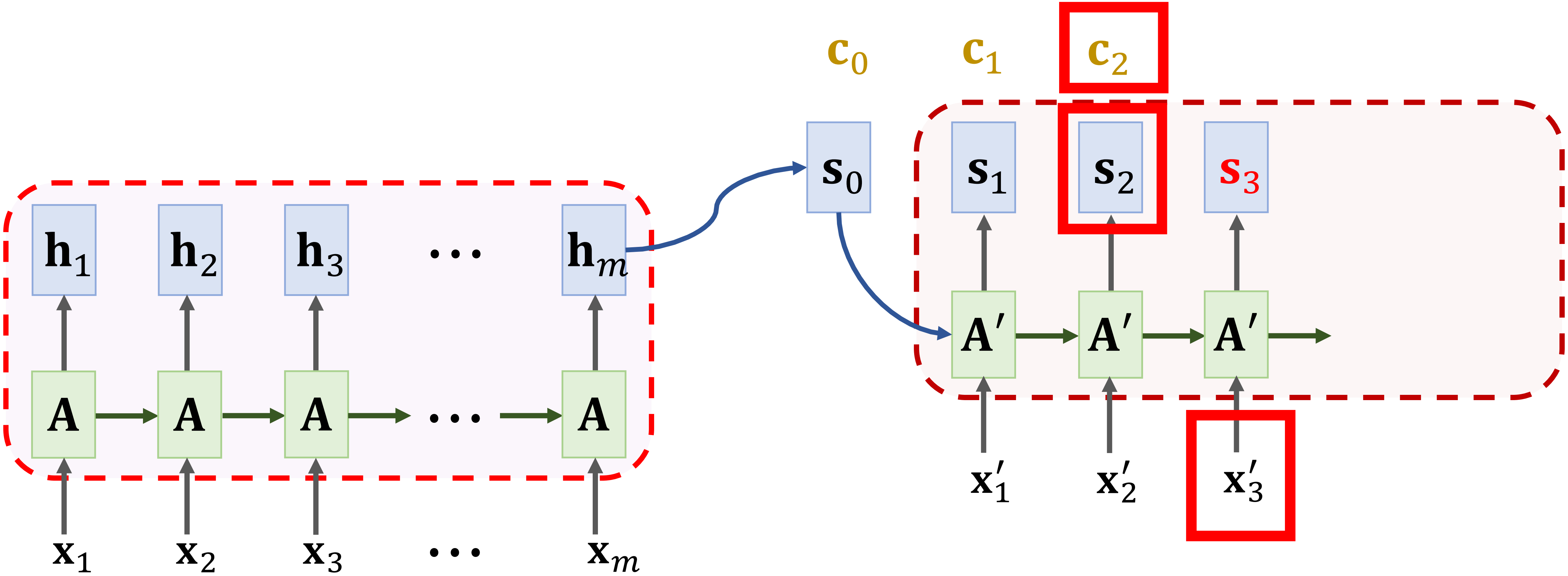
SimpleRNN + Attention



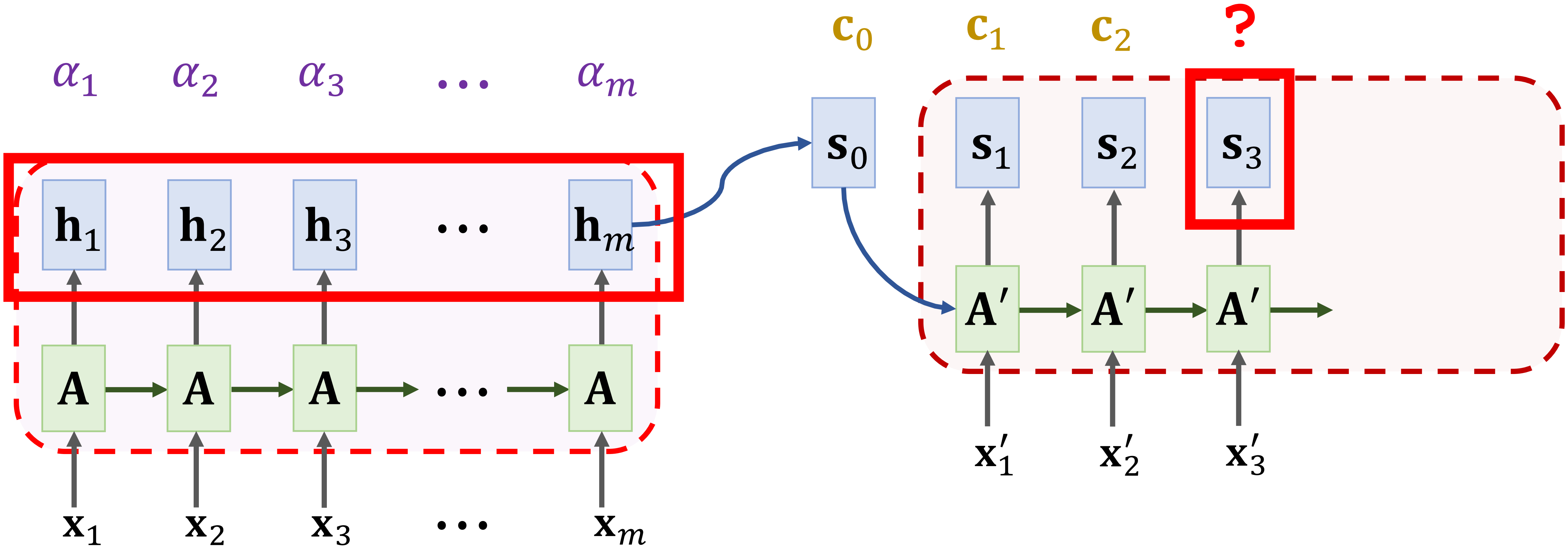
SimpleRNN + Attention



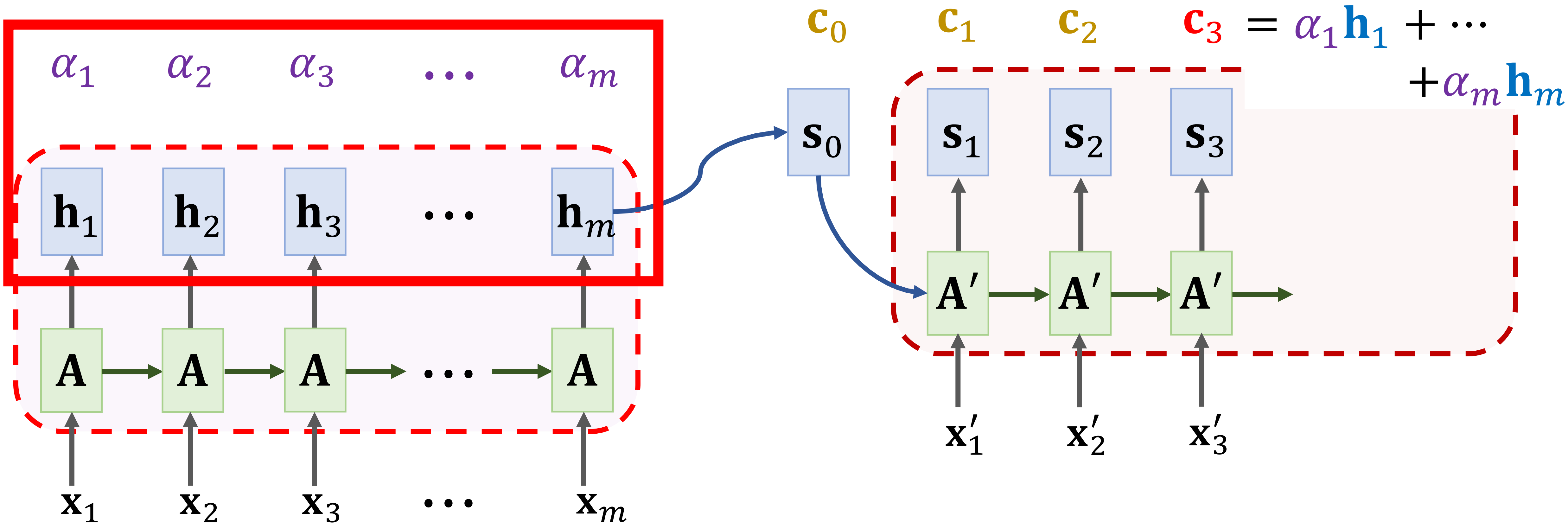
SimpleRNN + Attention



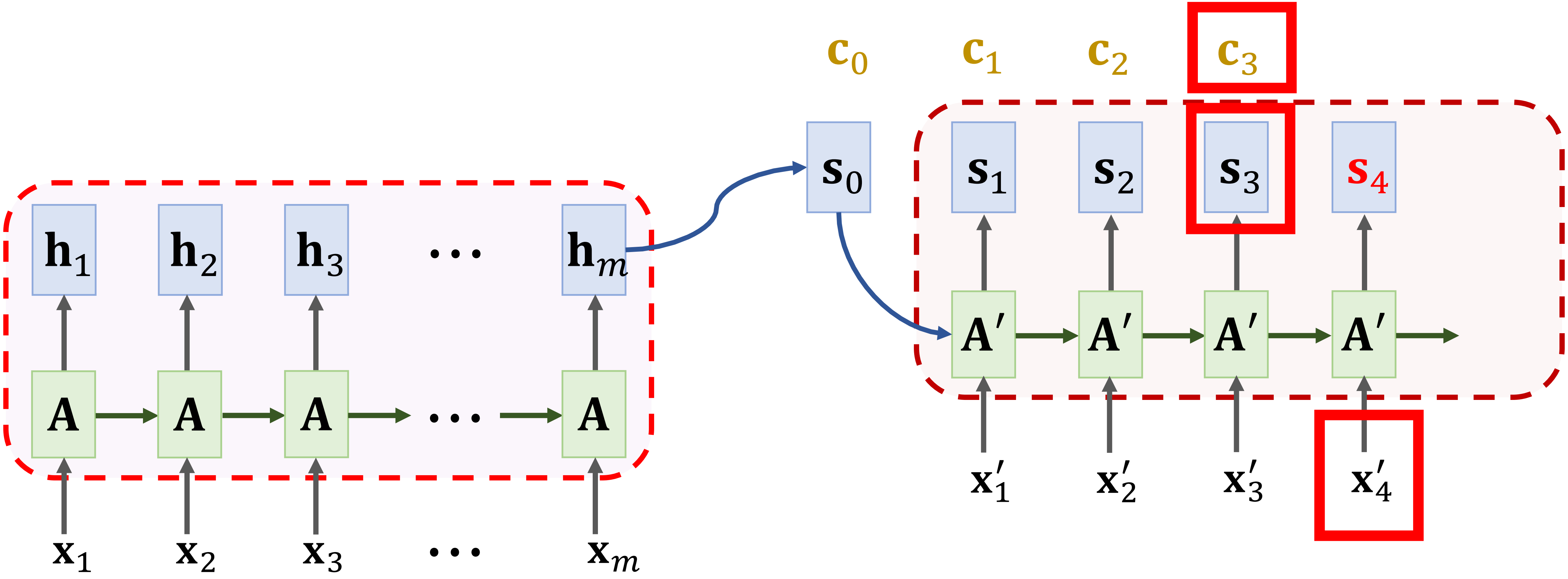
SimpleRNN + Attention



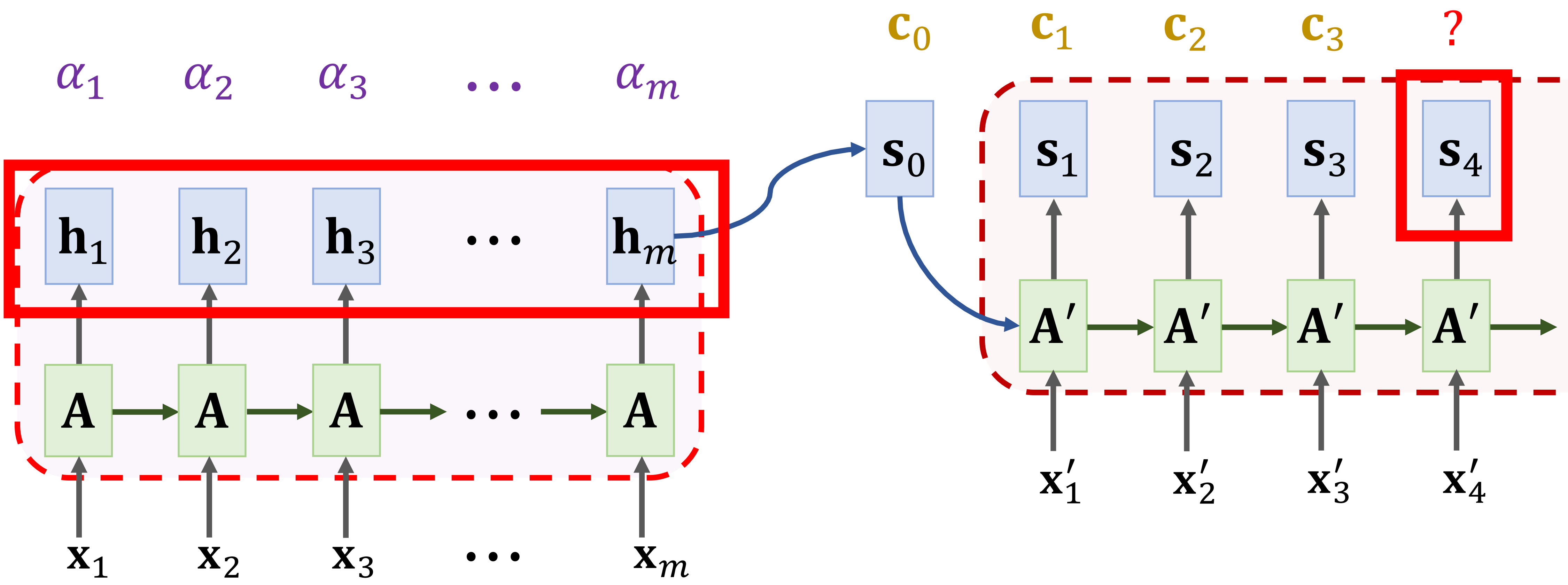
SimpleRNN + Attention



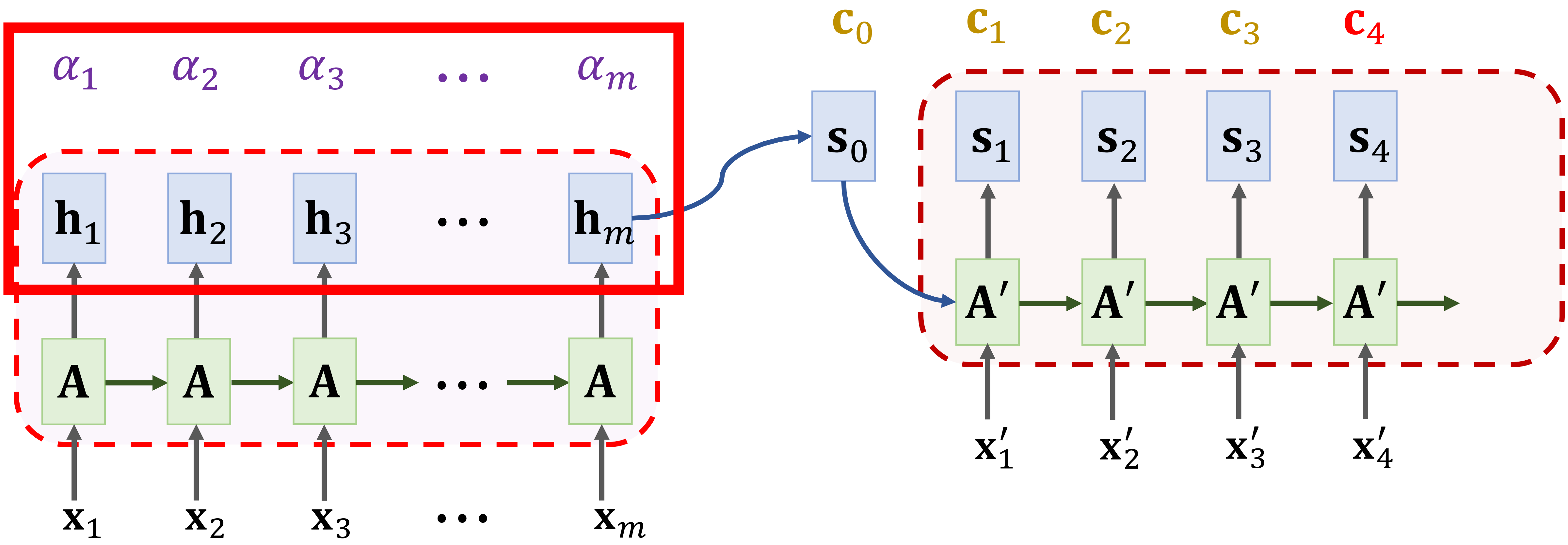
SimpleRNN + Attention



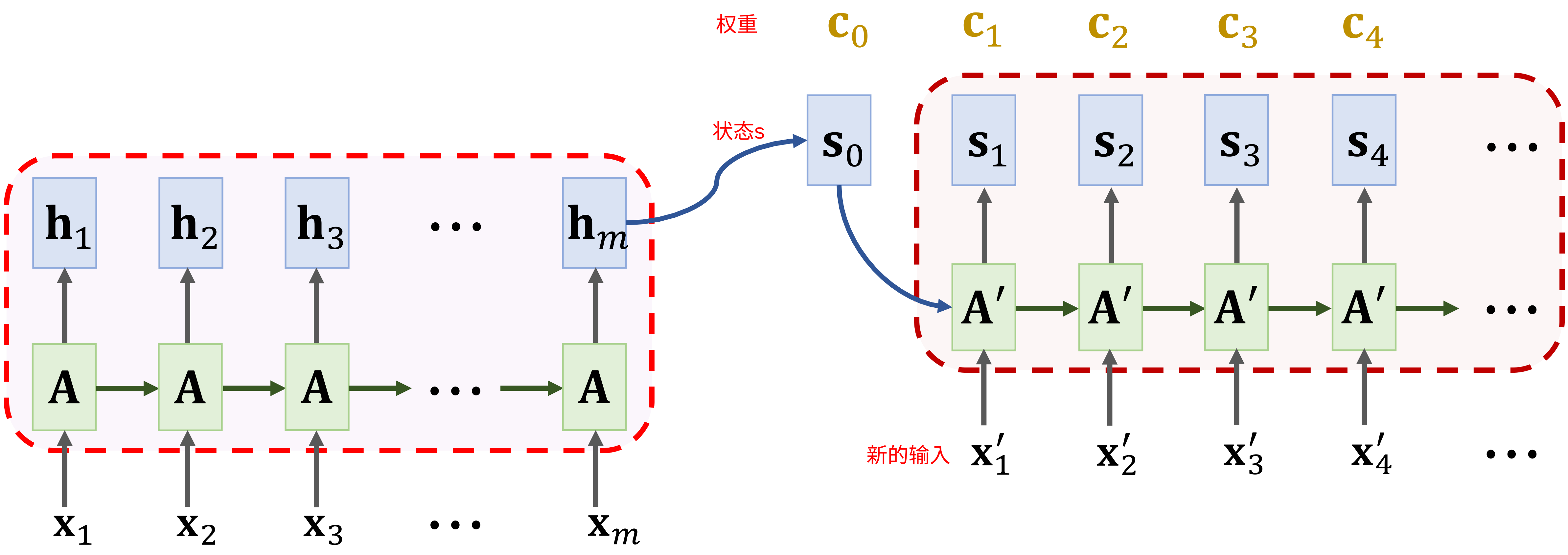
SimpleRNN + Attention



SimpleRNN + Attention

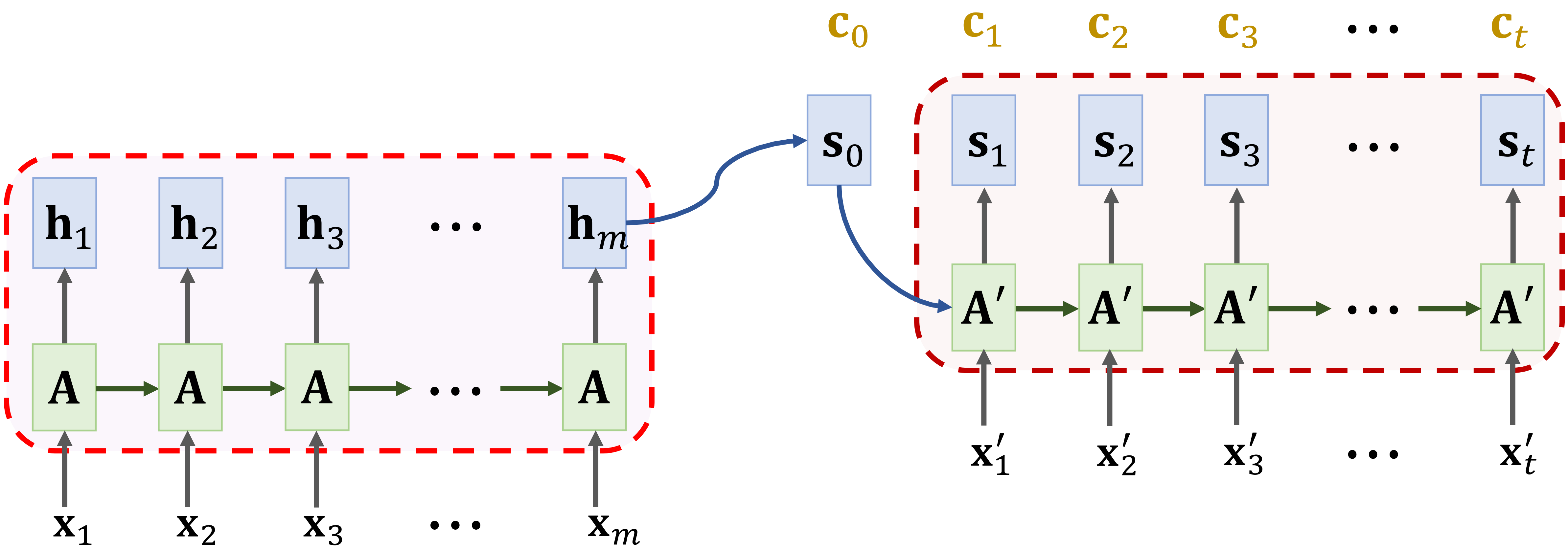


SimpleRNN + Attention



Time Complexity

Question: How many weights α_i have been computed?



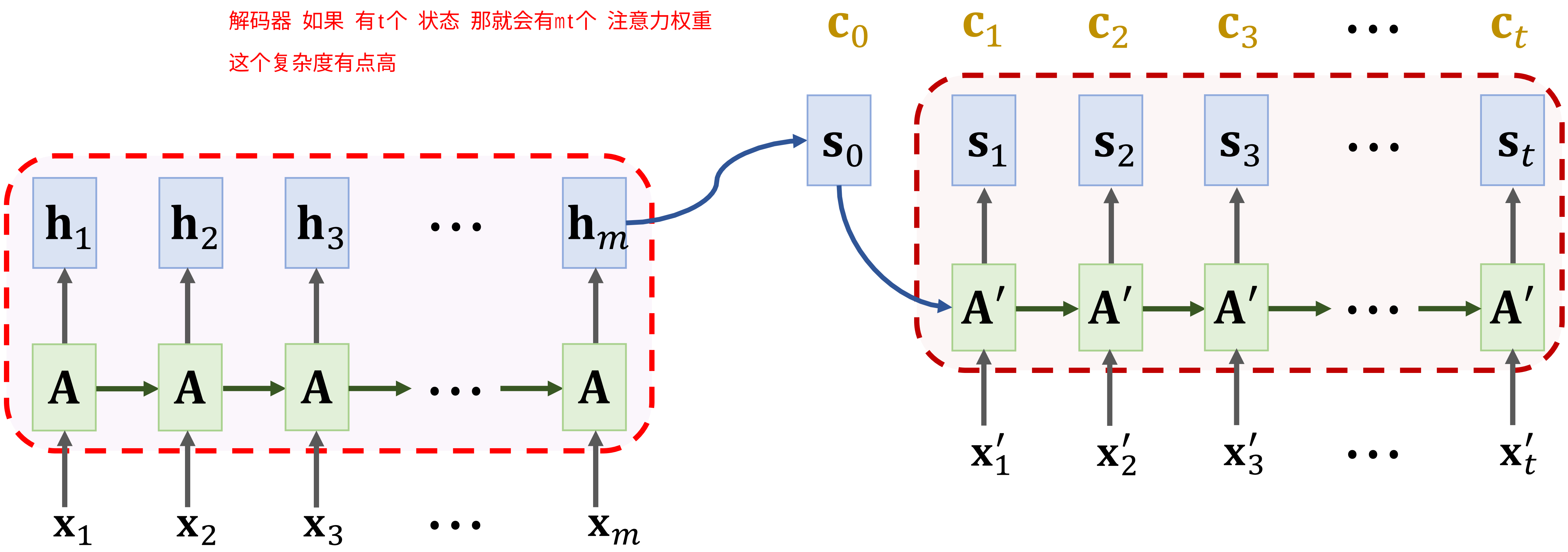
Time Complexity

Question: How many weights α_i have been computed?

- To compute one vector $\mathbf{c}_j$, we compute m weights: $\alpha_1, \cdots, \alpha_m$.
- The decode has t states, so there are totally mt weights.

解码器 如果 有t个 状态 那就会有mt个 注意力权重

这个复杂度有点高



Attention: Weights Visualization 注意：权重可视化

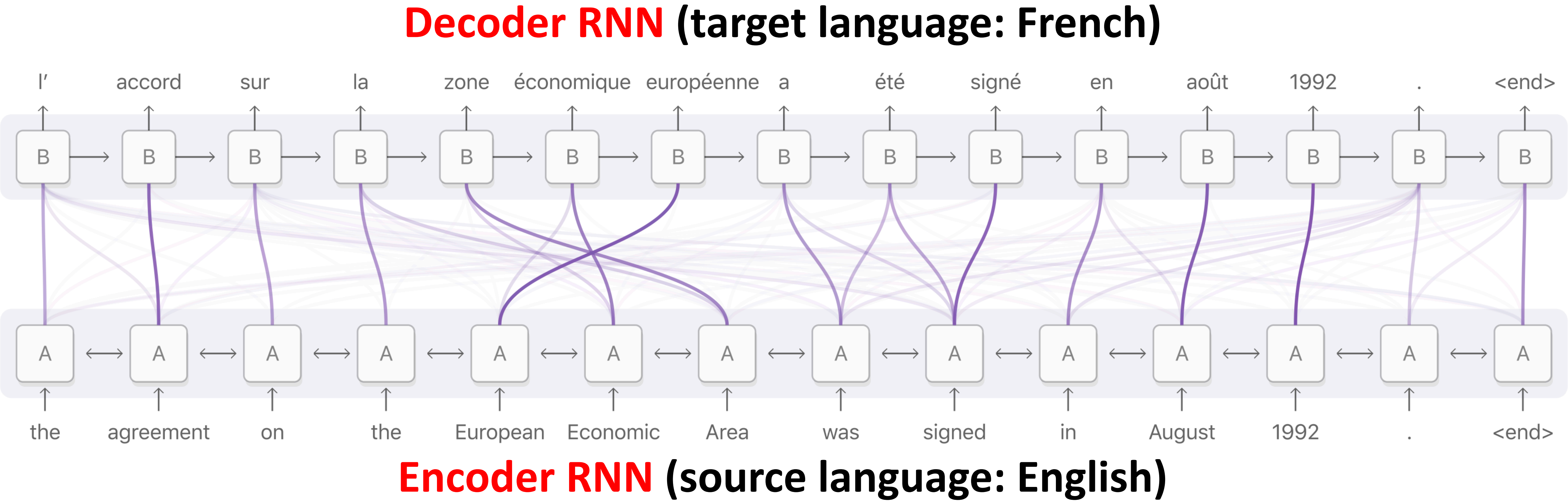
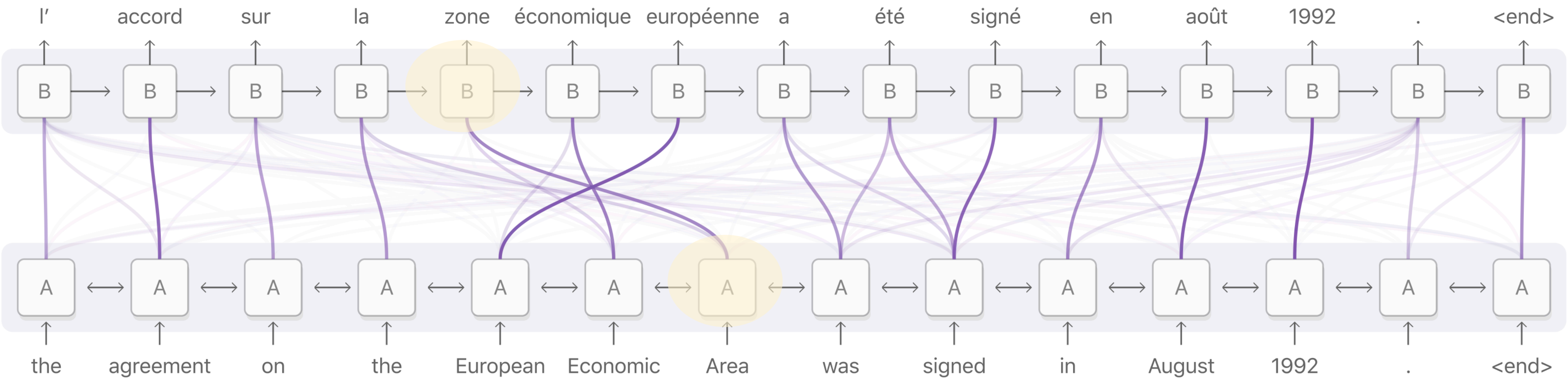


Figure is from https://distill.pub/2016/augmented-rnns/

Attention: Weights Visualization

Decoder RNN (target language: French)



Encoder RNN (source language: English)

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

Summary

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.

注意：解码器还将查看编码器的所有状态

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.

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.

- 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

attention 可以提升准确率
但是会引入大量的计算量

Thank you!