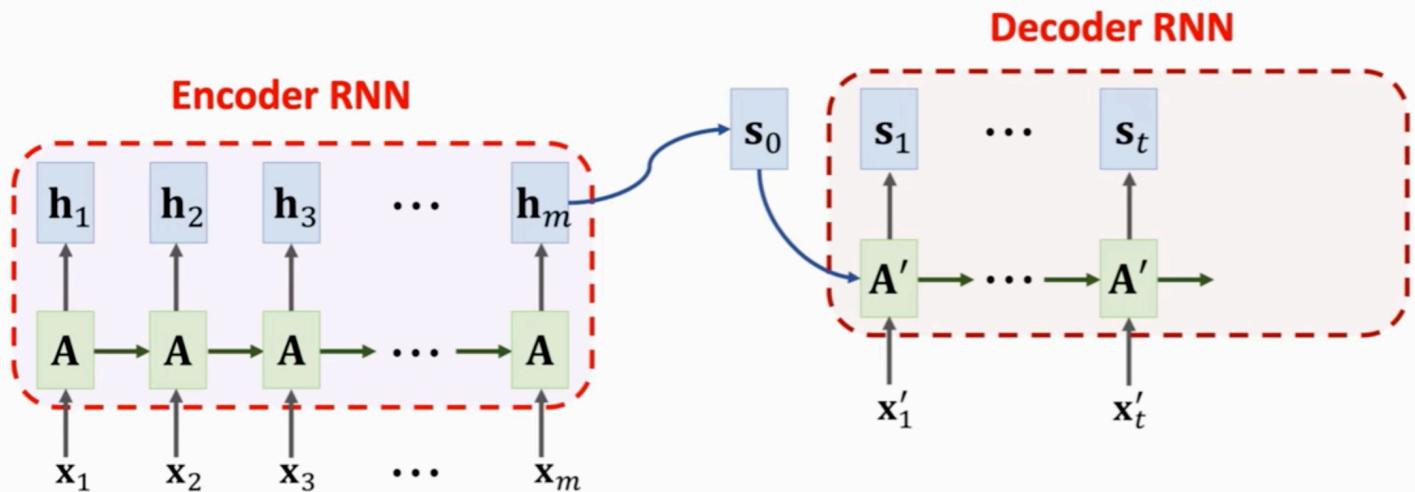


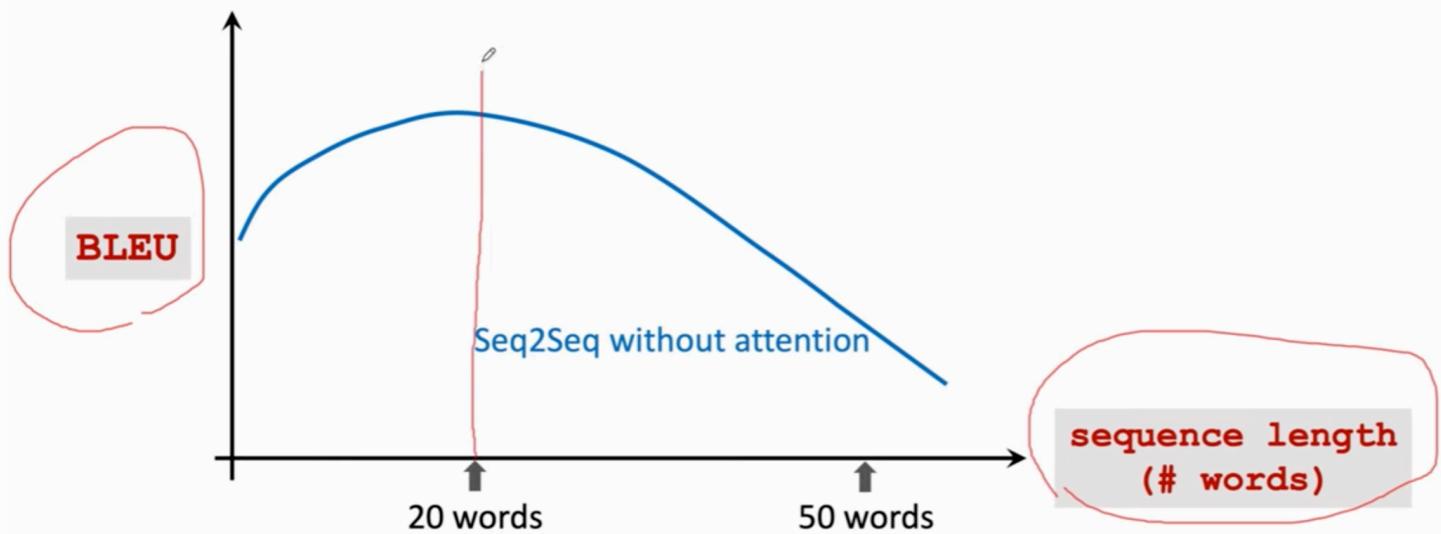
Seq2Seq Model

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



Seq2Seq Model

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



Seq2Seq Model with Attention

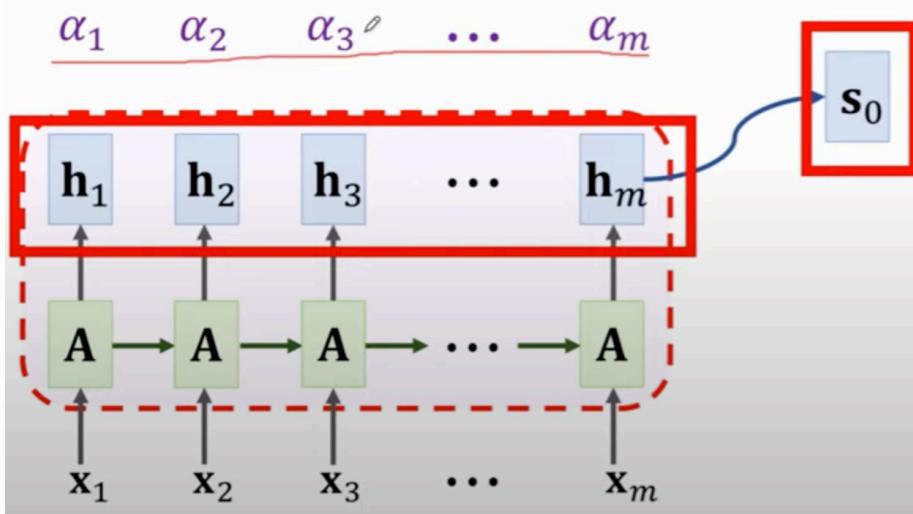
- Attention tremendously improves Seq2Seq model.
- With attention, Seq2Seq model does not forget source input.
- With attention, the decoder knows where to focus.
- Downside: much more computation.

Original paper:

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

SimpleRNN + Attention

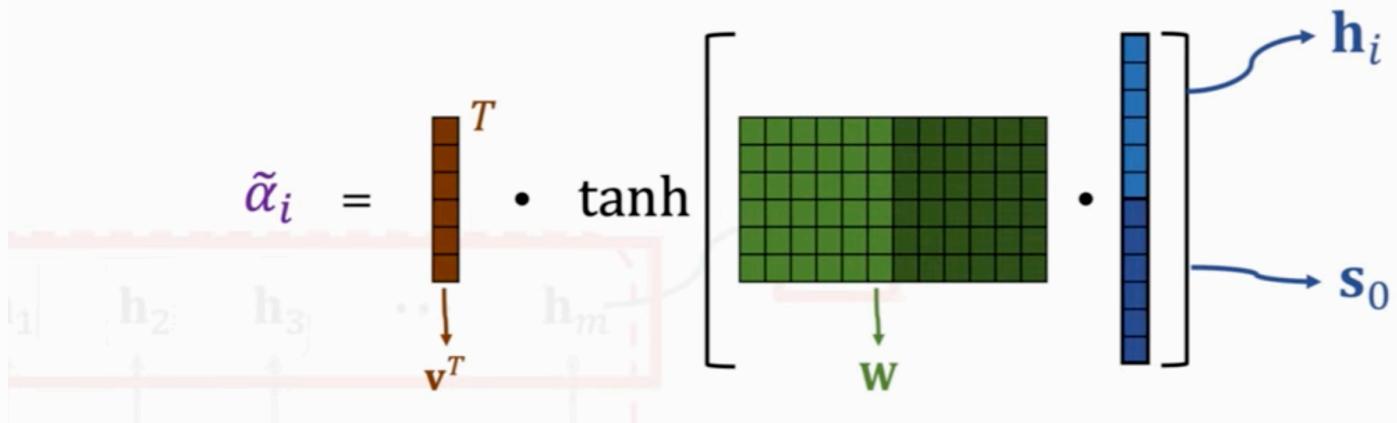
Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.



SimpleRNN + Attention

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

Option 1 (used in the original paper):



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

$$[\alpha_1, \dots, \alpha_m] = \text{Softmax}([\tilde{\alpha}_1, \dots, \tilde{\alpha}_m]).$$

SimpleRNN + Attention

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

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

1. Linear maps:

- $\mathbf{k}_i = \mathbf{W}_K \cdot \mathbf{h}_i$, for $i = 1$ to m .
- $\mathbf{q}_0 = \mathbf{W}_Q \cdot \mathbf{s}_0$.

2. Inner product:

- $\tilde{\alpha}_i = \underline{\mathbf{k}_i^T} \underline{\mathbf{q}_0}$, for $i = 1$ to m .

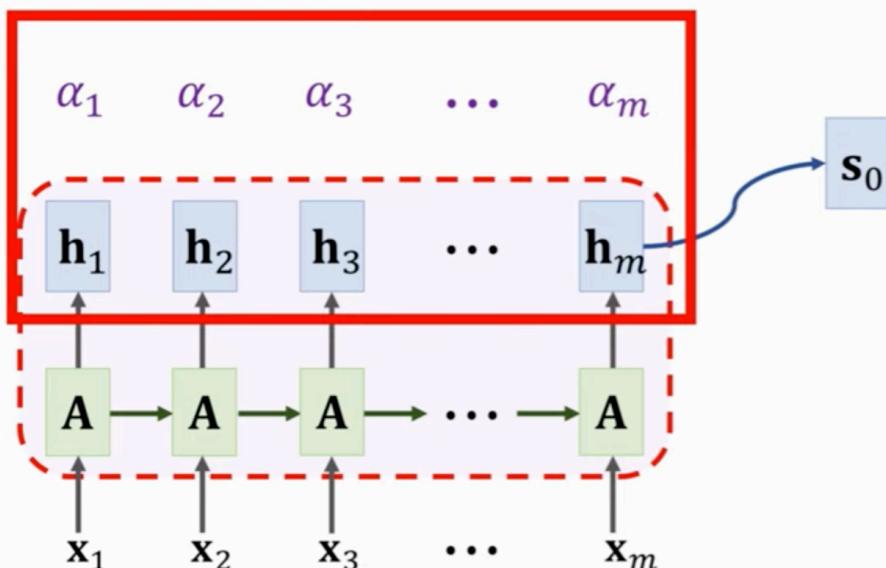
3. Normalization:

- $[\alpha_1, \dots, \alpha_m] = \text{Softmax}([\tilde{\alpha}_1, \dots, \tilde{\alpha}_m])$.

SimpleRNN + Attention

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

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



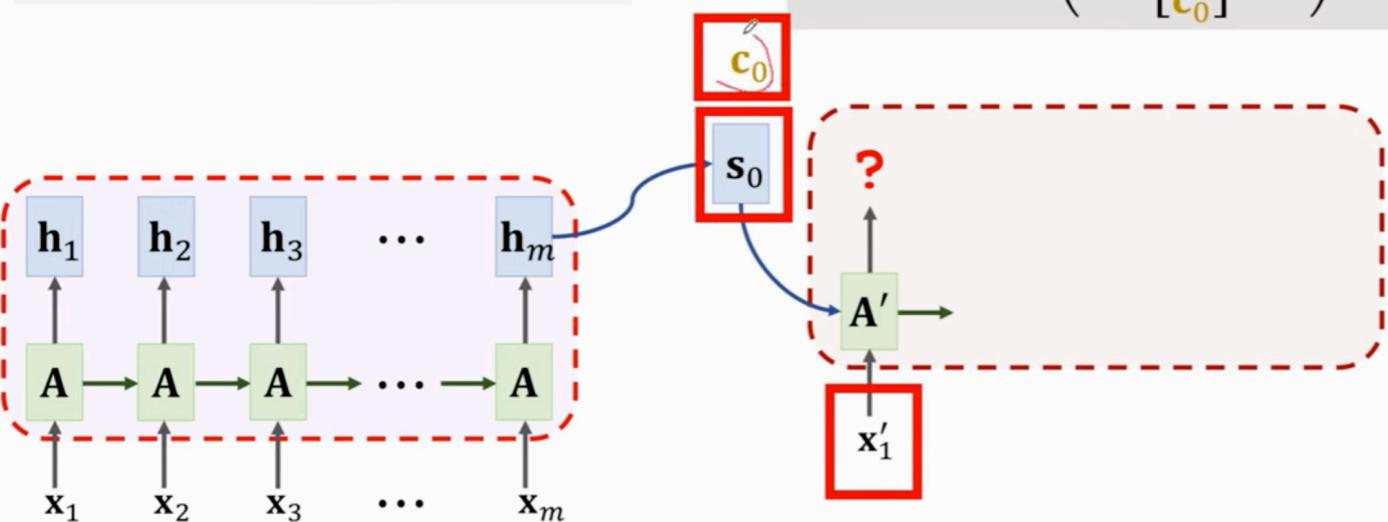
SimpleRNN + Attention

SimpleRNN:

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

SimpleRNN + Attention:

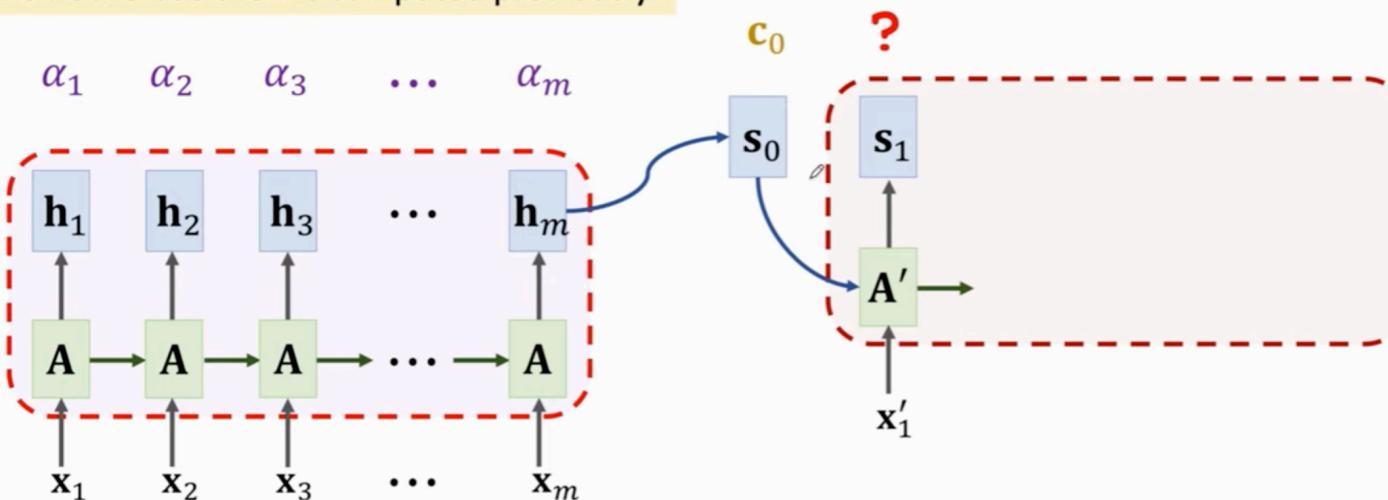
$$\mathbf{s}_1 = \tanh\left(\mathbf{A}' \cdot \begin{bmatrix} \mathbf{x}'_1 \\ \mathbf{s}_0 \\ \mathbf{c}_0 \end{bmatrix} + \mathbf{b}\right)$$



SimpleRNN + Attention

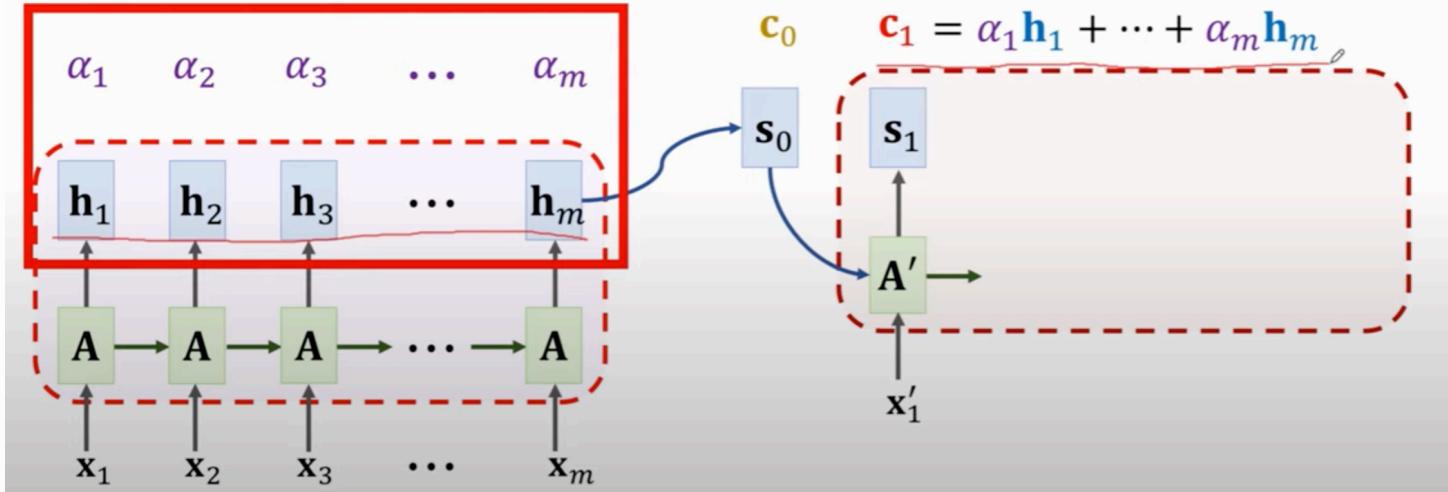
Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_1)$.

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



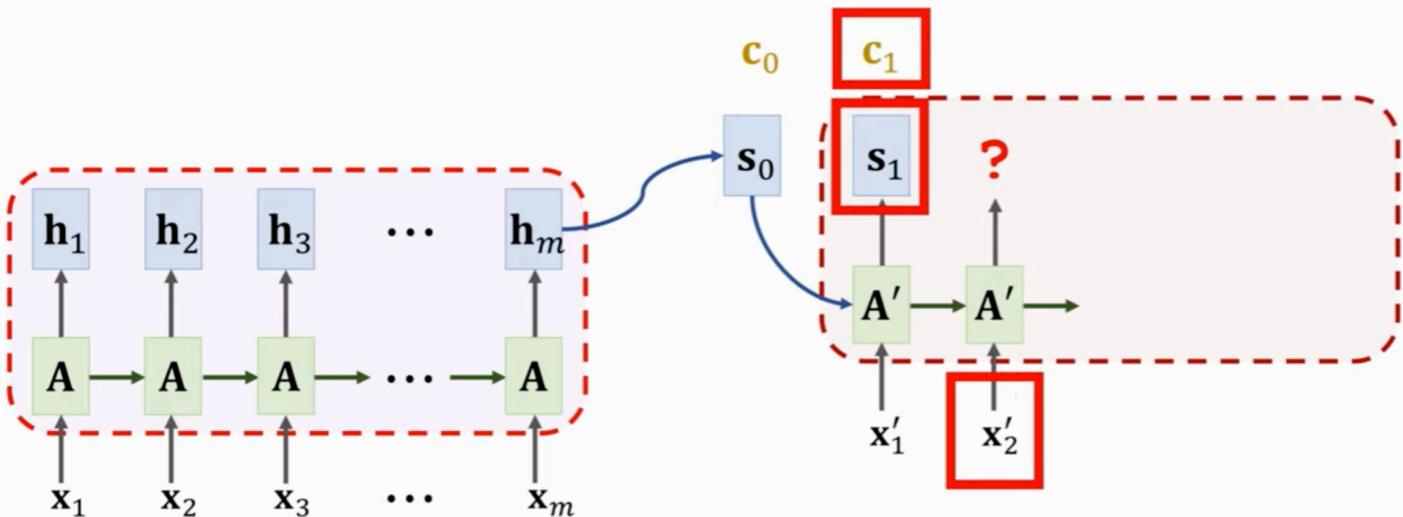
SimpleRNN + Attention

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

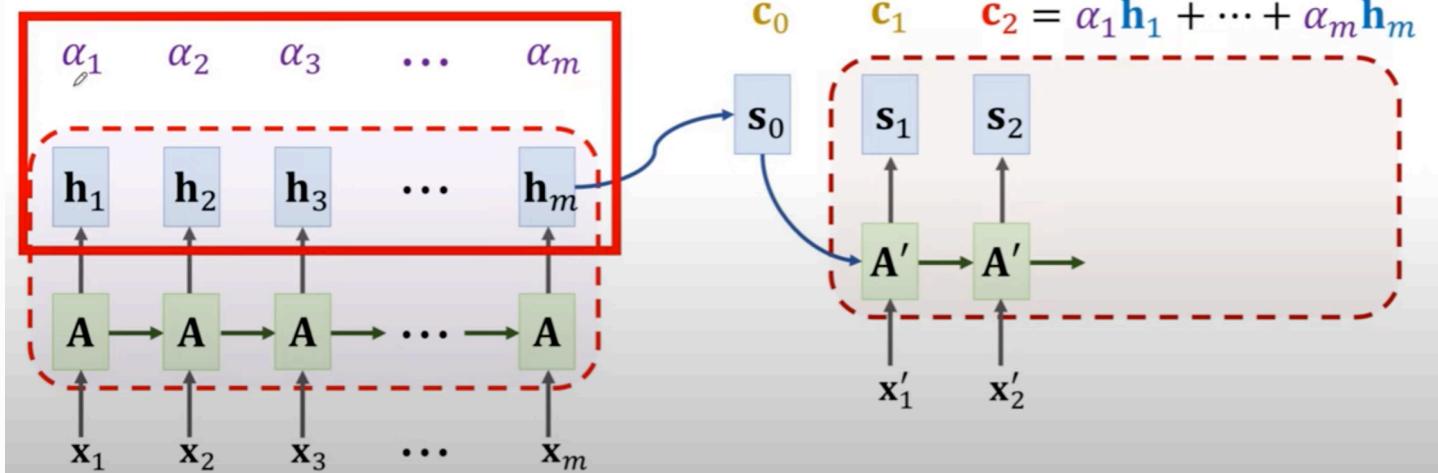


SimpleRNN + Attention

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



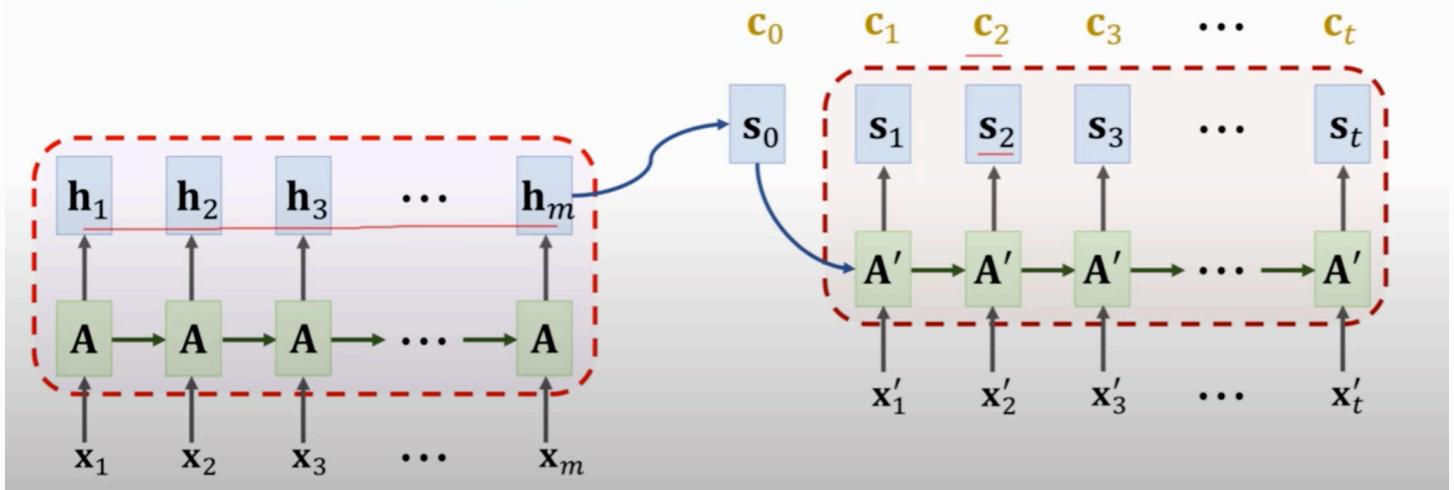
SimpleRNN + Attention



Time Complexity

Question: How many weights α_i have been computed?

- To compute one vector c_j , we compute m weights: $\alpha_1, \dots, \alpha_m$.
- The decode has t states, so there are totally mt weights.



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