ECE 884 Deep Learning

Lecture 19: RNN Applications and Attention

03/30/2021

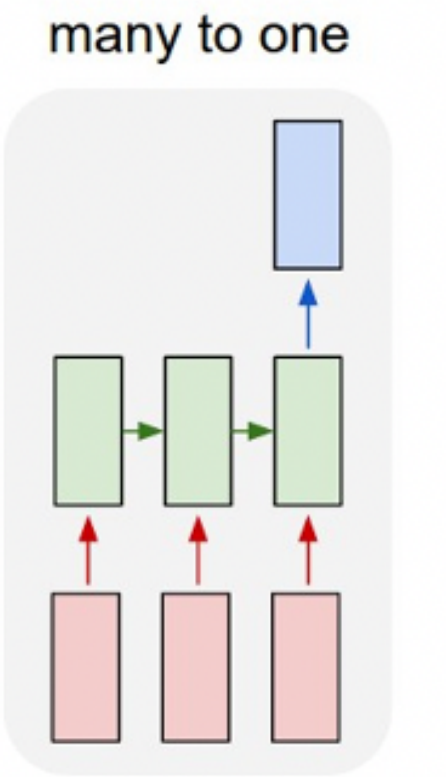
Review of last lecture

- Recurrent Neural Network (RNN)
  - Word Embeddings
  - Simple / Vanilla RNN
  - Long Short Term Memory (LSTM)
  - Making RNNs More Effective

Today’s lecture

- RNN Applications
  - Text generation (many to one)
  - Image Captioning (one to many)
  - Neural Machine Translation (many to many - Seq2Seq)
- Attention with RNN
- Self-attention with RNN

Text Generation

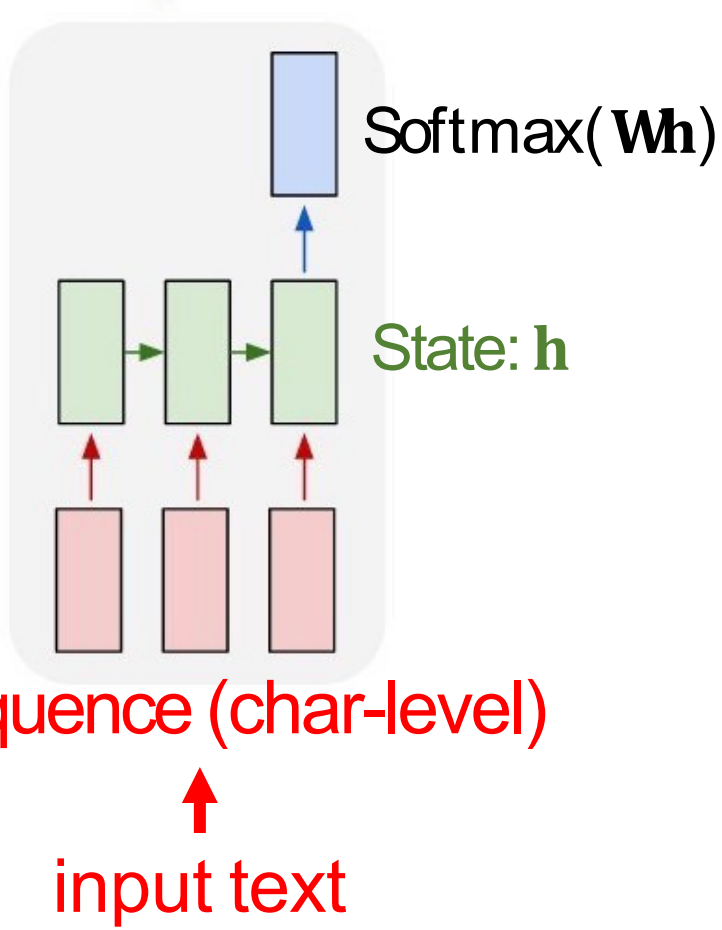


RNN for Text Prediction

Example

- Input text: "`the cat sat on the ma`"
- Question: what is the next char?

RNN for Text Prediction

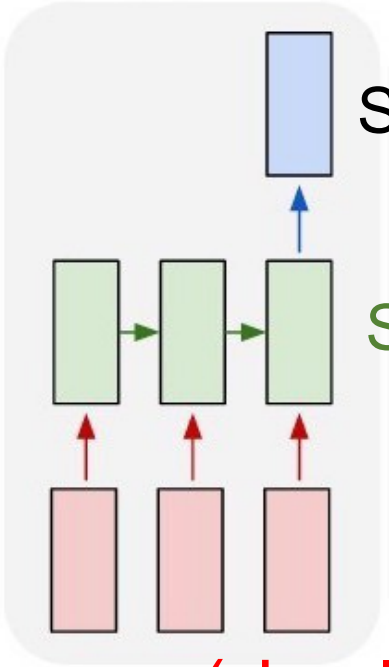


Example

- Input text: "`the cat sat on the ma`"
- Question: what is the next char?

RNN for Text Prediction

predict the next char



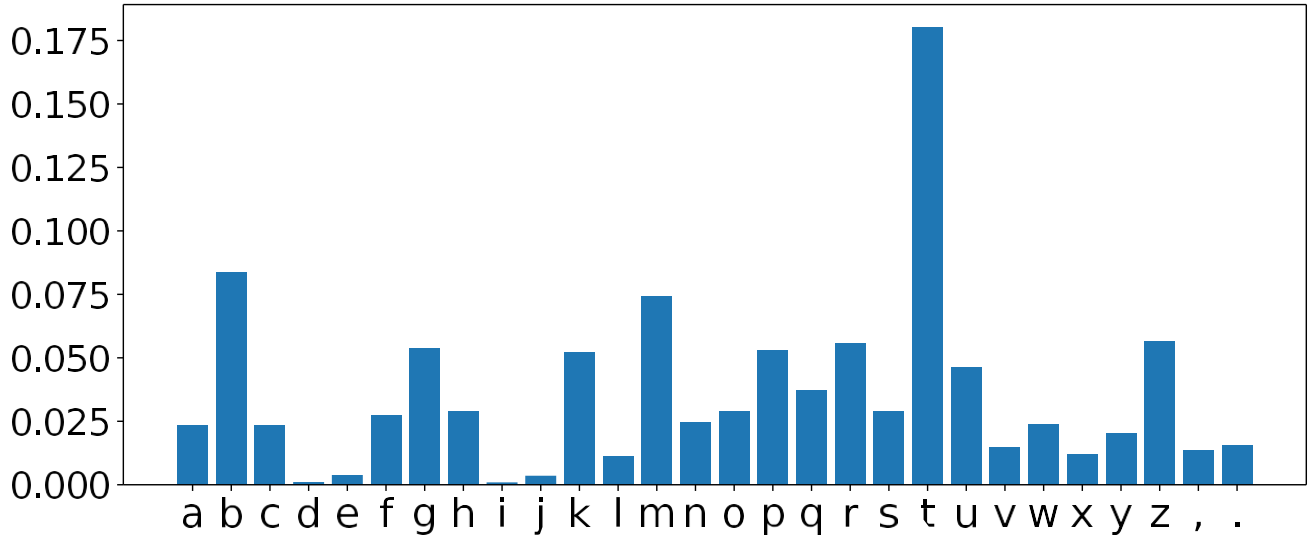
sequence (char-level)

input text

Example

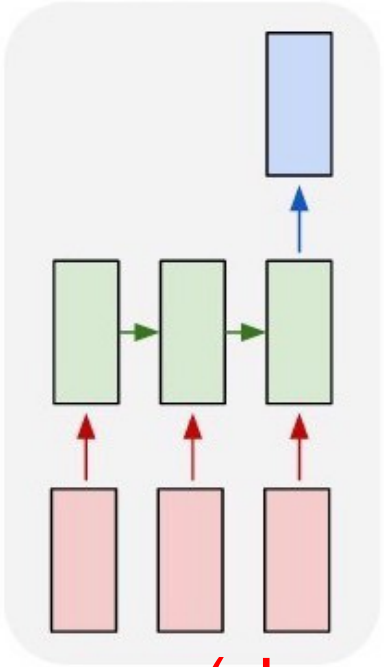
- Input text: "`the cat sat on the ma`"
- Question: what is the next char?

- RNN outputs a distribution over the chars.



RNN for Text Prediction

predict the next char



sequence (char-level)

input text

Example

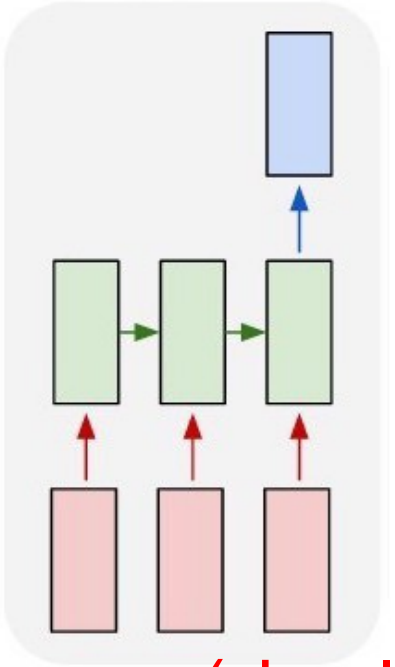
- Input text: "`the cat sat on the ma`"
- Question: what is the next char?

- RNN outputs a distribution over the chars.

- Sample a char from it; we may get '`t`'.
- Take "`the cat sat on the mat`" as input.
- Maybe the next char is period '`.`'.

Train an RNN for Text Prediction

predict the next char



sequence (char-level)

input text

How do we train such an RNN?

- Cut text to segments (with overlap).
  - E.g., `seg_len=40` and `stride=3`.

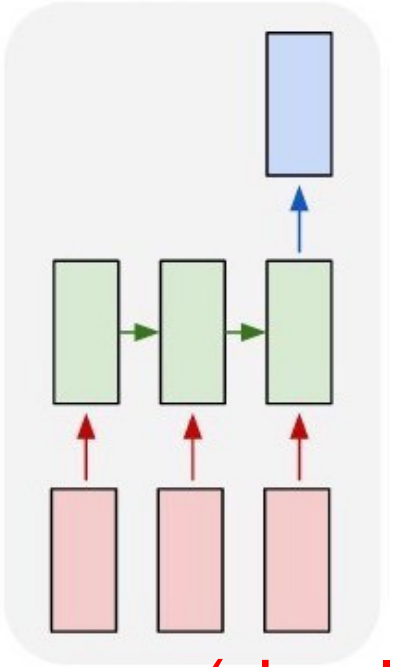
Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. The name machine learning was coined in 1959 by Arthur Samuel.

…

…

Train an RNN for Text Prediction

predict the next char



sequence (char-level)

input text

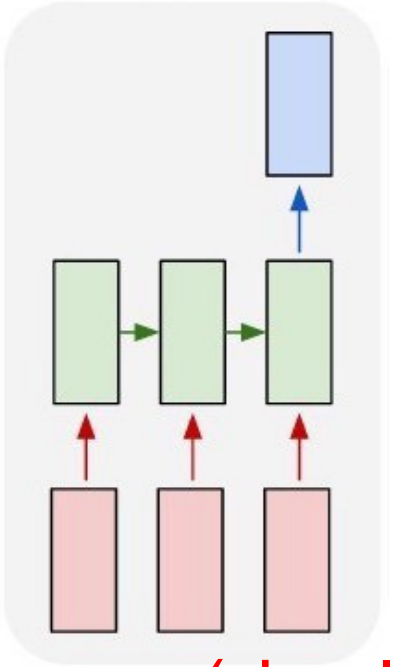
How do we train such an RNN?

- Cut text to segments (with overlap).
  - E.g., `seg_len=40` and `stride=3`.

```
Machine learning is a subset of artificial intelligence in
the field of computer science that often uses statistical
techniques to give computers the ability to "learn" (i.e.,
progressively improve performance on a specific task) with
data, without being explicitly programmed. The name machine
learning was coined in 1959 by Arthur Samuel.
…
…
```

Train an RNN for Text Prediction

predict the next char



sequence (char-level)

input text

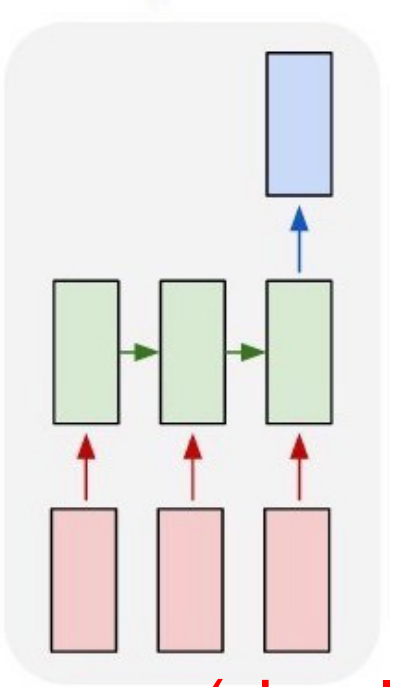
How do we train such an RNN?

- Cut text to segments (with overlap).
  - E.g., `seg_len=40` and `stride=3`.

```
Machine learning is a subset of artificial intelligence in
the field of computer science that often uses statistical
techniques to give computers the ability to "learn" (i.e.,
progressively improve performance on a specific task) with
data, without being explicitly programmed. The name machine
learning was coined in 1959 by Arthur Samuel.
…
…
```

Train an RNN for Text Prediction

predict the next char



sequence (char-level)

input text

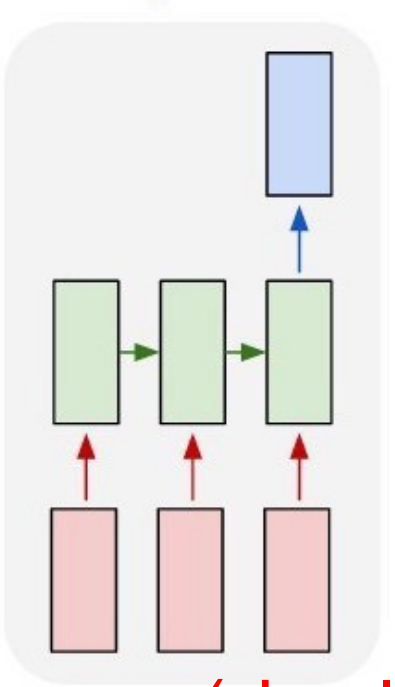
How do we train such an RNN?

- Cut text to segments (with overlap).
  - E.g., `seg_len=40` and `stride=3`.

```
Machine learning is a subset of artificial intelligence in
the field of computer science that often uses statistical
techniques to give computers the ability to "learn" (i.e.,
progressively improve performance on a specific task) with
data, without being explicitly programmed. The name machine
learning was coined in 1959 by Arthur Samuel.
…
…
```

Train an RNN for Text Prediction

predict the next char



sequence (char-level)

input text

How do we train such an RNN?

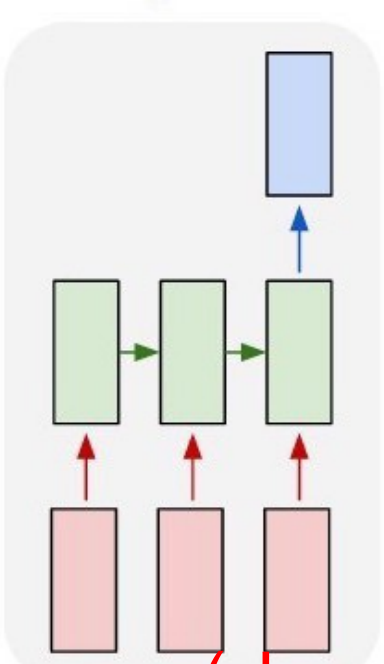
- Cut text to segments (with overlap).
- A segment is used as input text.
- Its next char is used as label.
- Training data: `(segment, next_char)` pairs

```
Machine learning is a subset of artificial intelligence in
the field of computer science that often uses statistical
techniques to give computers the ability to "learn" (i.e.,
progressively improve performance on a specific task) with
data, without being explicitly programmed. The name machine
learning was coined in 1959 by Arthur Samuel.
…
…
```

Train an RNN for Text Prediction

How do we train such an RNN?

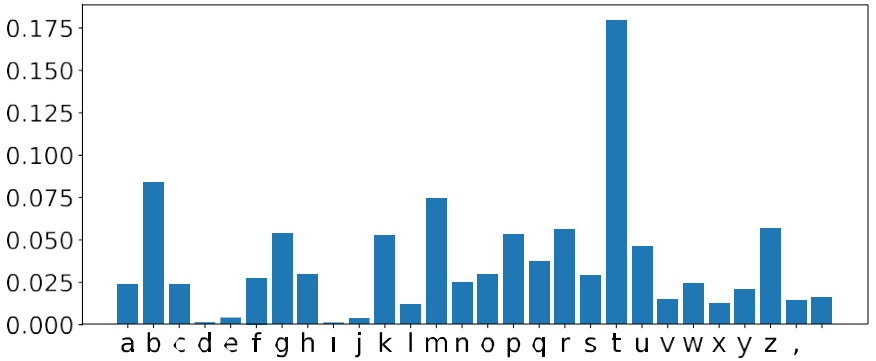
predict the next char



sequence (char-level)

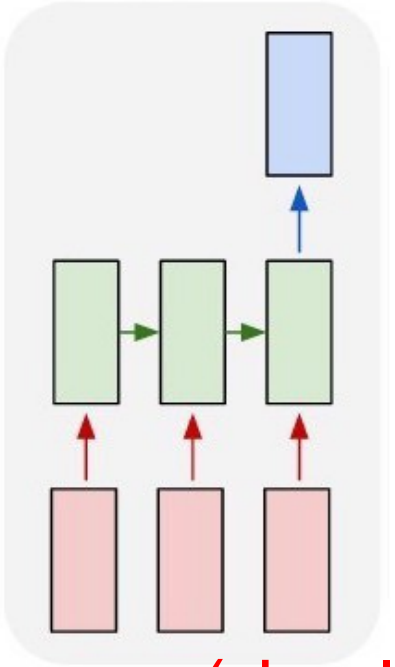
input text

- Cut text to segments (with overlap).
- A segment is used as input text.
- Its next char is used as label.
- Training data: `(segment, next_char)` pairs
  - It is a multi-class classification problem.
    - #class = #unique chars.



Train an RNN for Text Prediction

predict the next char



sequence (char-level)

input text

How do we train such an RNN?

- Cut text to segments (with overlap).
- A segment is used as input text.
- Its next char is used as label.
- Training data: `(segment, next_char)` pairs
  - It is a multi-class classification problem.
    - #class = #unique chars.

If the RNN is trained on Shakespeare's books, then the generated text is Shakespeare's style.

Text Generation#1

Generate baby names (trained on 8000 baby names).

Rudi Levette Berice Lussa Hany Mareanne Chrestina Carissy Marylen Hammine Janye Marlise Jacacrie Hendred Romand Charienna Nenotto Ette Dorane Wallen Marly Darine Salina Elvyn Ersia Maralena Minoria Ellia Charmin Antley Nerille Chelon Walmor Evena Jeryly Stachon Charisa Allisa Anatha Cathanie Geetra Alexie Jerin Cassen Herbett Cossie Velen Daurenge Robester Shermond Terisa Licia Roselen Ferine Jayn Lusine Charyanne Sales Sanny Resa Wallon Martine Merus Jelen Candica Wallin Tel Rachene Tarine Ozila Ketia Shanne Arnande Karella Roselina Alessia Chasty Deland Berther Geamar Jackein Mellisand Sagdy Nenc Lessie Rasemy Guen Gavi Milea Anneda Margoris Janin Rodelin Zeanna Elyne Janah Ferzina Susta Pey Castina

Text Generation#2

Generate Ccode (trained on Linux source code).

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

Text Generation#3

Generate academic articles (LaTeX source files).

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the compariciely in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, **??** and the fact that any U affine, see Morphisms, Lemma **??**. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma **??** we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example **??**. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma **??**. Namely, by Lemma **??** we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ *(by the formal open covering* X *and a single map* $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ *over* U *compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ *is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition* **??** *(without element is when the closed subschemes are catenary. If* T *is surjective we may assume that* T *is connected with residue fields of* S*. Moreover there exists a closed subspace* $Z \subset X$ *of* X *where* U *in* X' *is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) f *is locally of finite type. Since* $S = \mathrm{Spec}(R)$ *and* $Y = \mathrm{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let* X *be a locally Noetherian scheme over* S, $E = \mathcal{F}_{X/S}$. *Set* $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. *Since* $\mathcal{I}^n \subset \mathcal{I}^n$ *are nonzero over* $i_0 \leq \mathfrak{p}$ *is a subset of* $\mathcal{J}_{n,0} \circ \overline{A}_2$ *works.*

Lemma 0.3. *In Situation* **??**. *Hence we may assume* $\mathfrak{q}' = 0$.

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (**??**). On the other hand, by Lemma **??** we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

Image Captioning

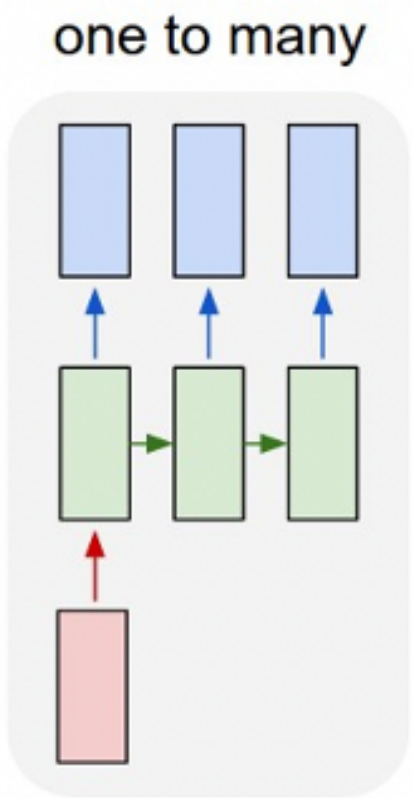


Image Captioning

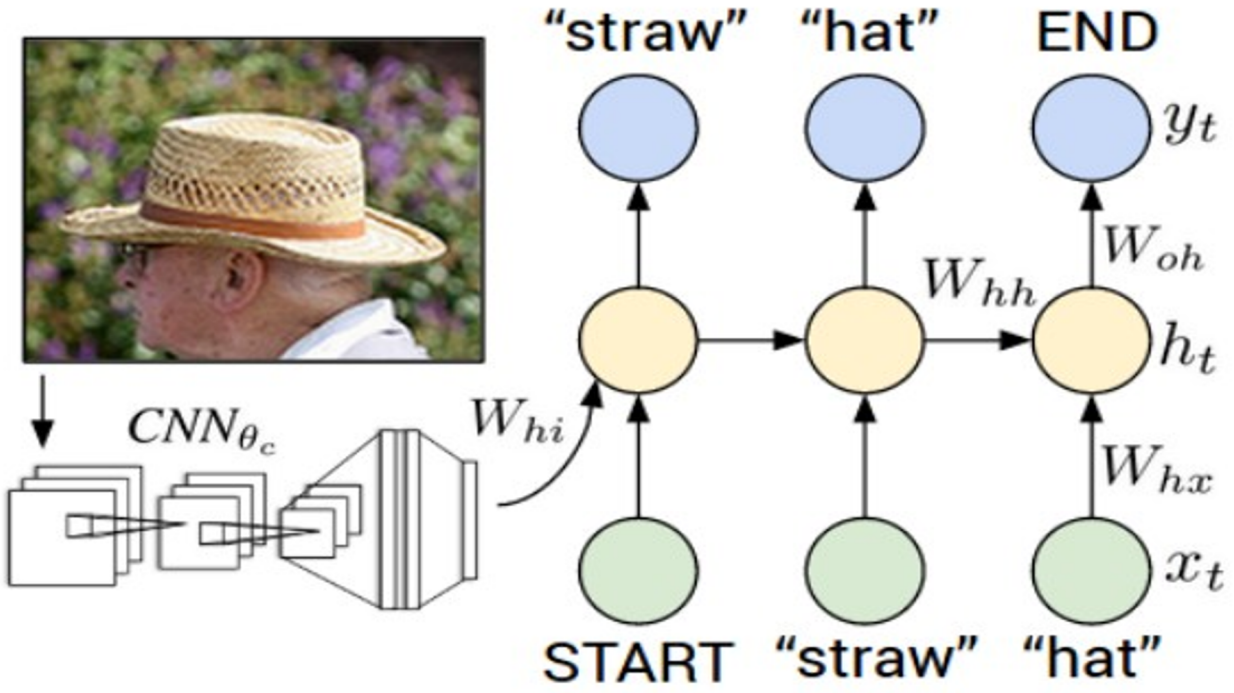
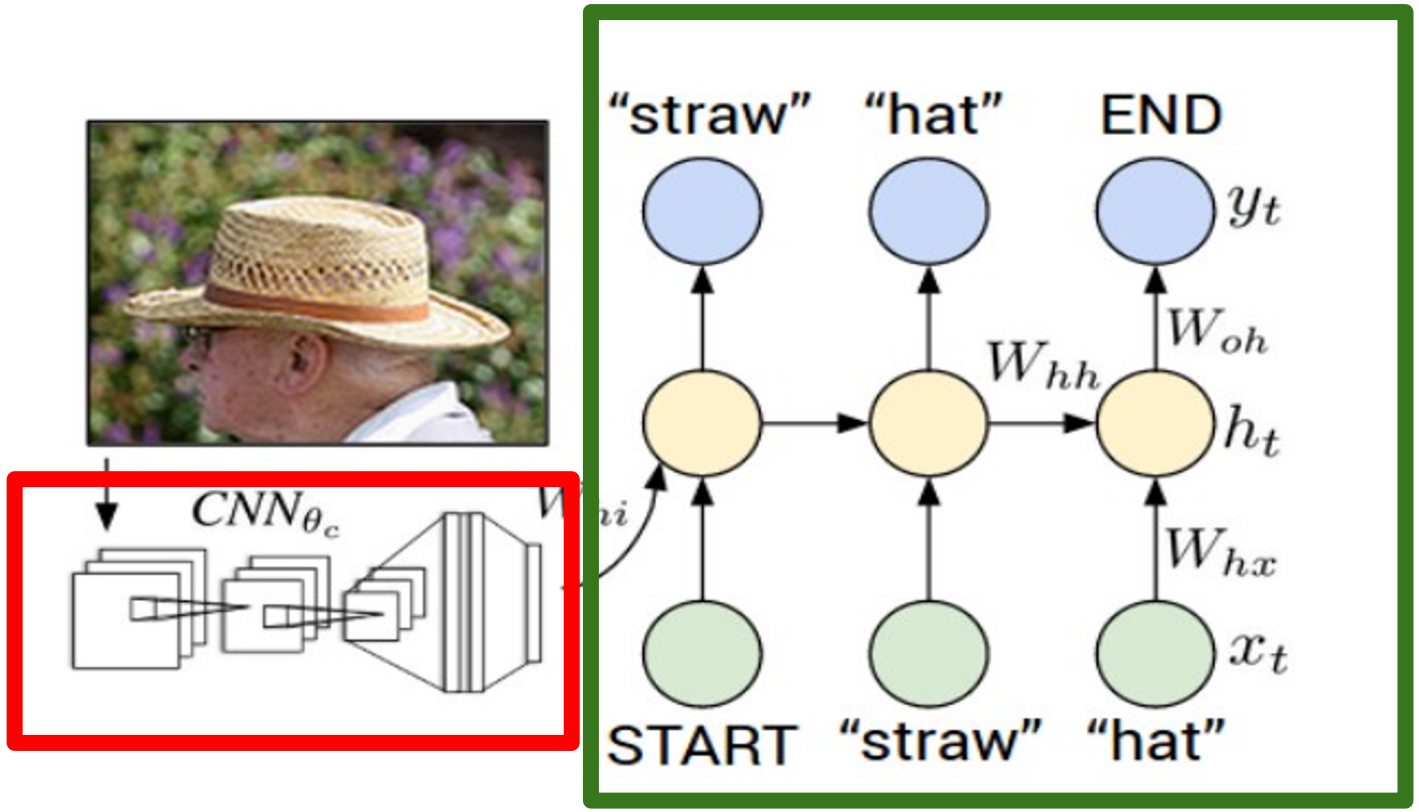


Image Captioning



Recurrent Neural Network

Convolutional Neural Network

Transfer learning: Take CNN trained on ImageNet, chop off last layer

This image is CC0 public domain

image

conv-64
conv-64
maxpool

conv-128
conv-128
maxpool

conv-256
conv-256
maxpool

conv-512
conv-512
maxpool

conv-512
conv-512
maxpool

FC-4096
FC-4096

x0

<START>

image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096

Before:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Now:

$$\tanh(W_{hh}h_{t-1} + W_{xh}x_t + W_{ih}v + b_h)$$

image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096

Before:

$$h_t = \tanh(\boldsymbol{W_{hh}h_{t-1}} + \boldsymbol{W_{xh}x_t} + b_h)$$

Now:

$$\tanh(\boldsymbol{W_{hh}h_{t-1}} + \boldsymbol{W_{xh}x_t} + \boldsymbol{W_{ih}v} + b_h)$$

W_{ih}

y0
h0
x0

<START>
man
man

Sample word and copy to input

image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096

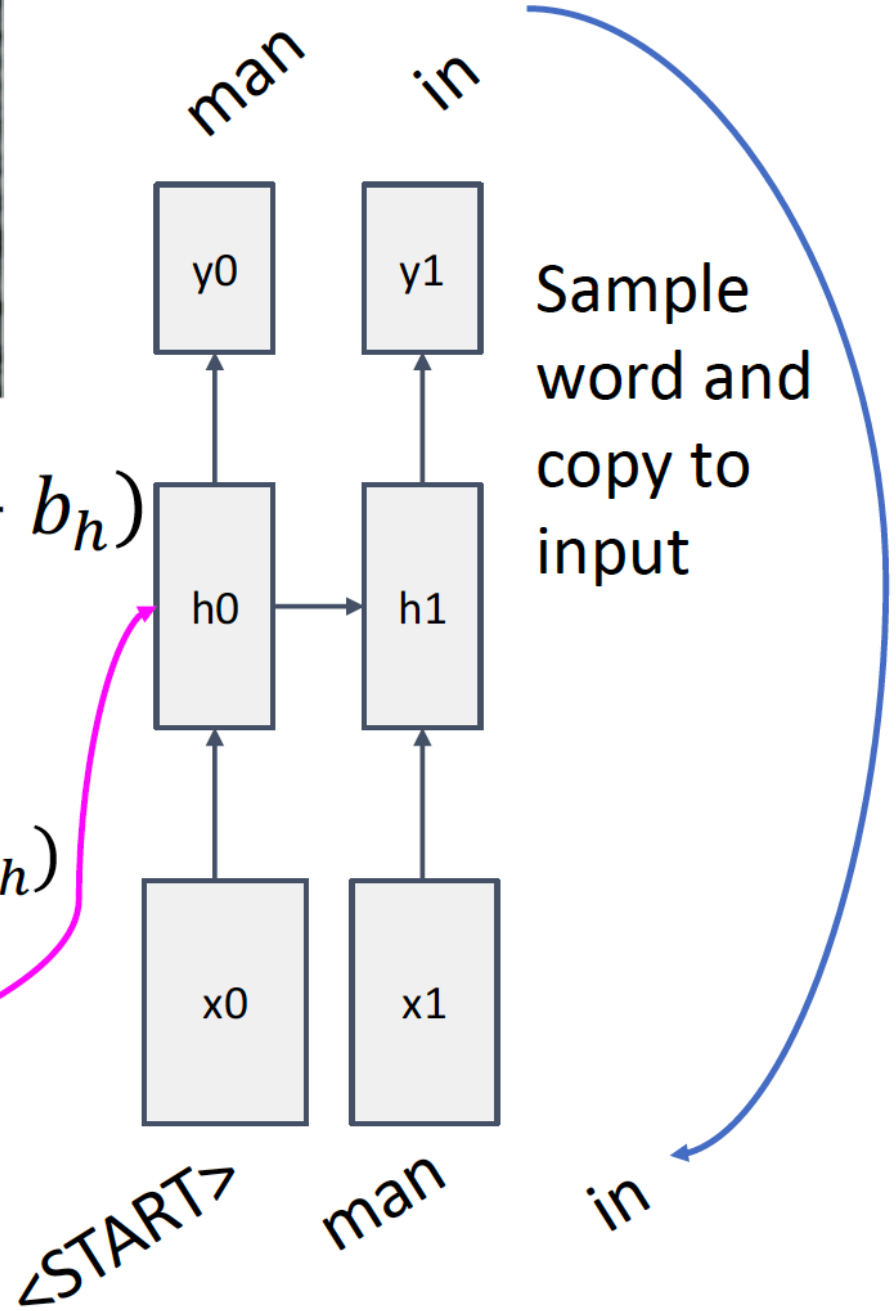
Before:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Now:

$$\tanh(W_{hh}h_{t-1} + W_{xh}x_t + W_{ih}v + b_h)$$

W_{ih}



image

conv-64
conv-64
maxpool

conv-128
conv-128
maxpool

conv-256
conv-256
maxpool

conv-512
conv-512
maxpool

conv-512
conv-512
maxpool

FC-4096
FC-4096

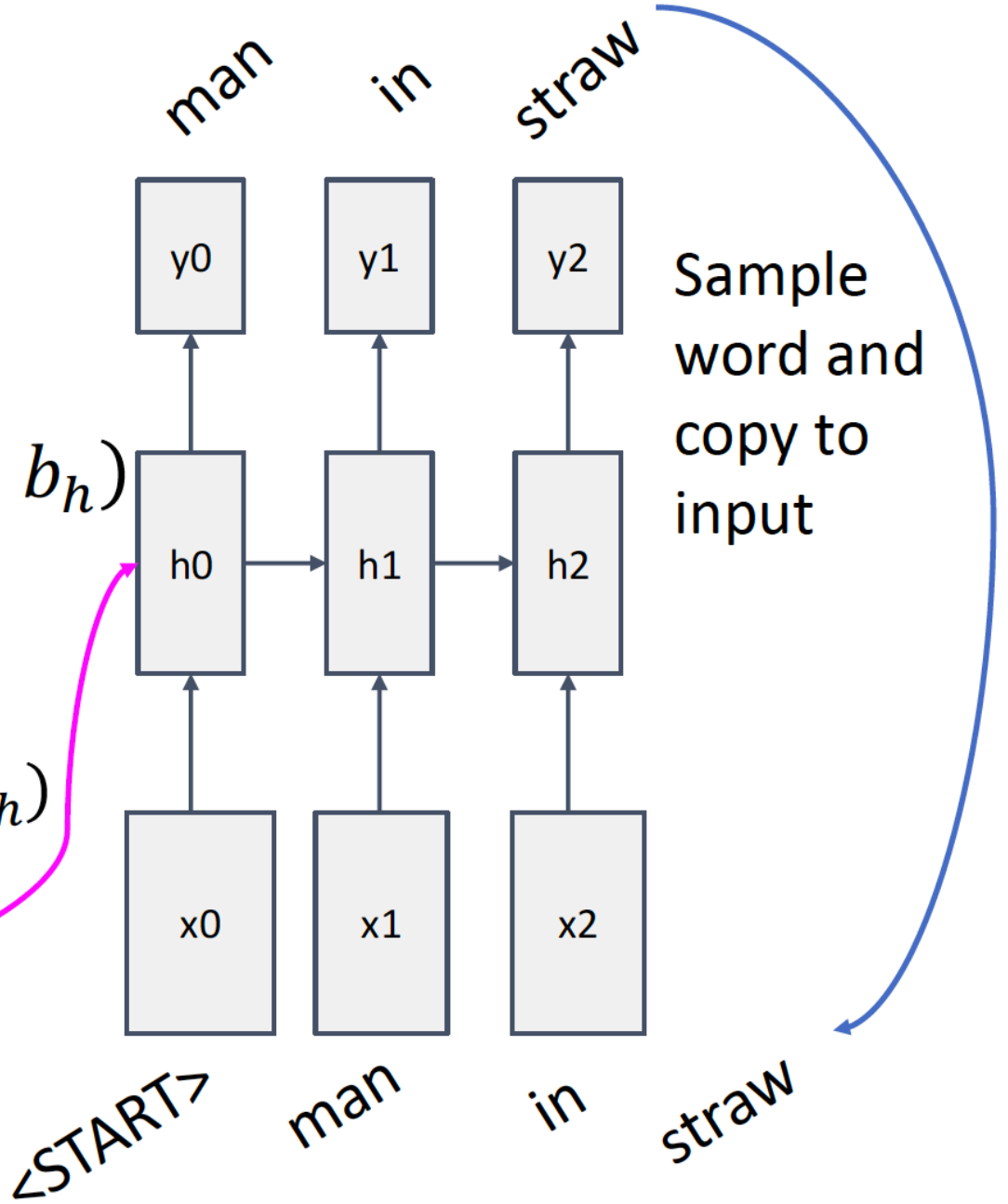
Before:

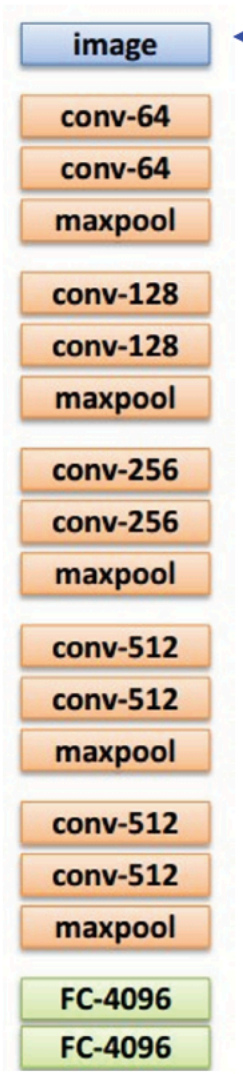
$$h_t = \tanh(\boldsymbol{W_{hh}h_{t-1}} + \boldsymbol{W_{xh}x_t} + b_h)$$

Now:

$$\tanh(\boldsymbol{W_{hh}h_{t-1}} + \boldsymbol{W_{xh}x_t} + \boldsymbol{W_{ih}v} + b_h)$$

W_{ih}





Before:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Now:

$$\tanh(W_{hh}h_{t-1} + W_{xh}x_t + W_{ih}v + b_h)$$

Before:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Now:

$$\tanh(W_{hh}h_{t-1} + W_{xh}x_t + W_{ih}v + b_h)$$

W_{ih}

Stop after sampling <END> token

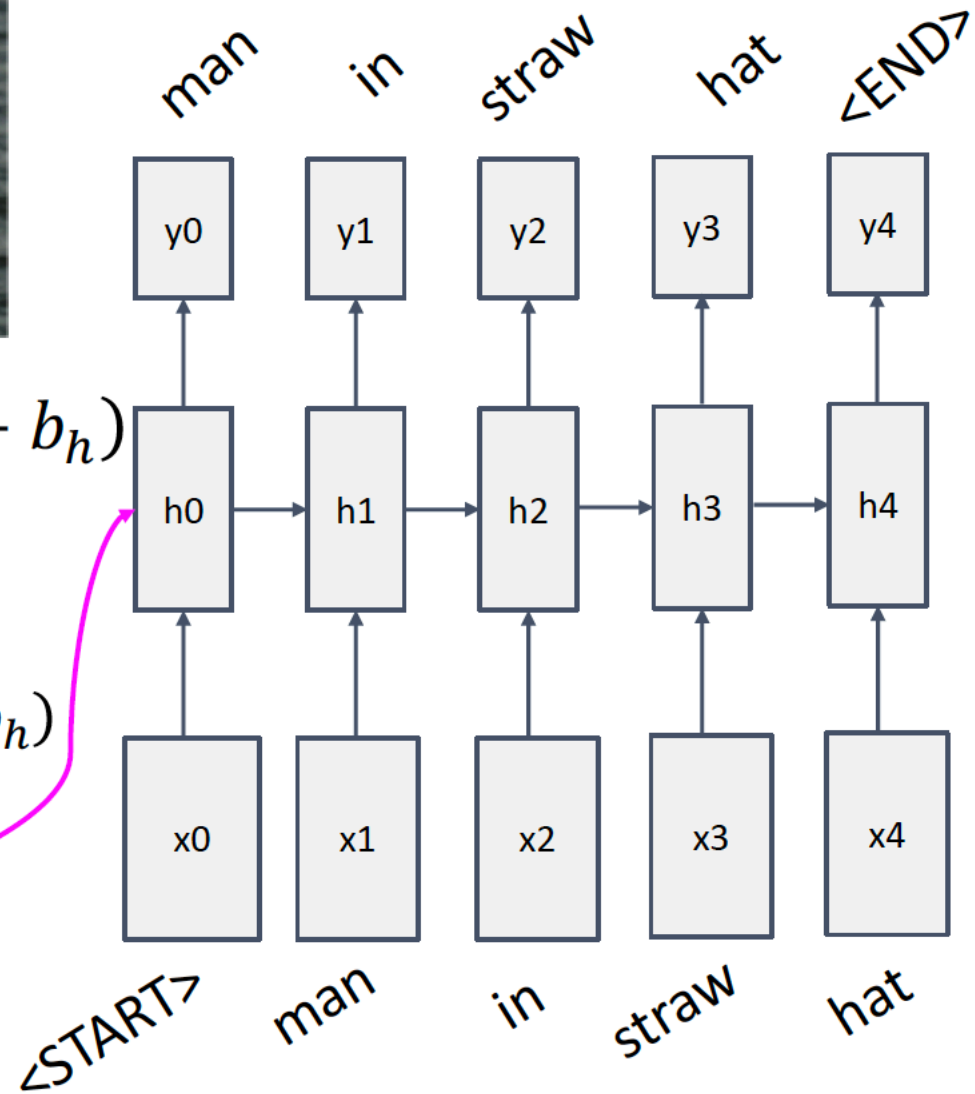


Image Captioning: Example Results

Captions generated using neuraltalk2
All images are CC0Public domain: cat suitcase, cat tree, dog, bear, surfers, tennis, giraffe, motorcycle

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

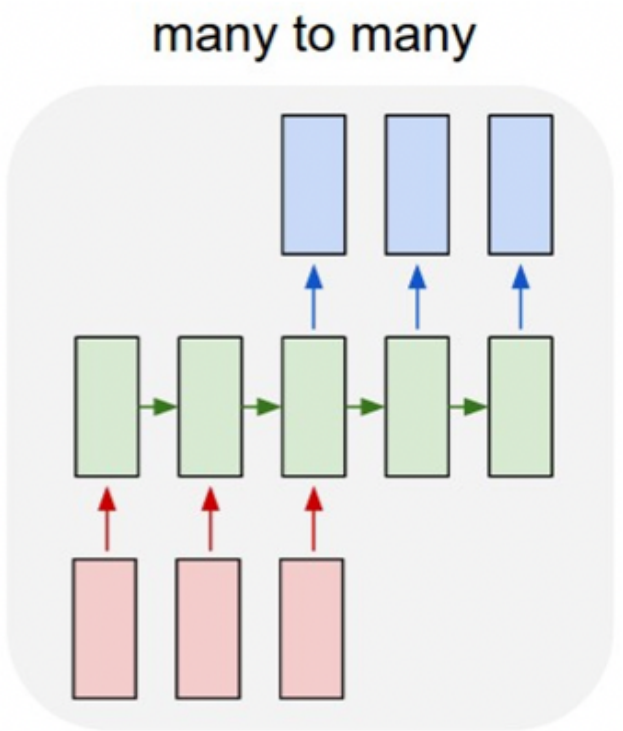
Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

A man riding a dirt bike on a dirt track

Neural Machine Translation



Neural Machine Translation

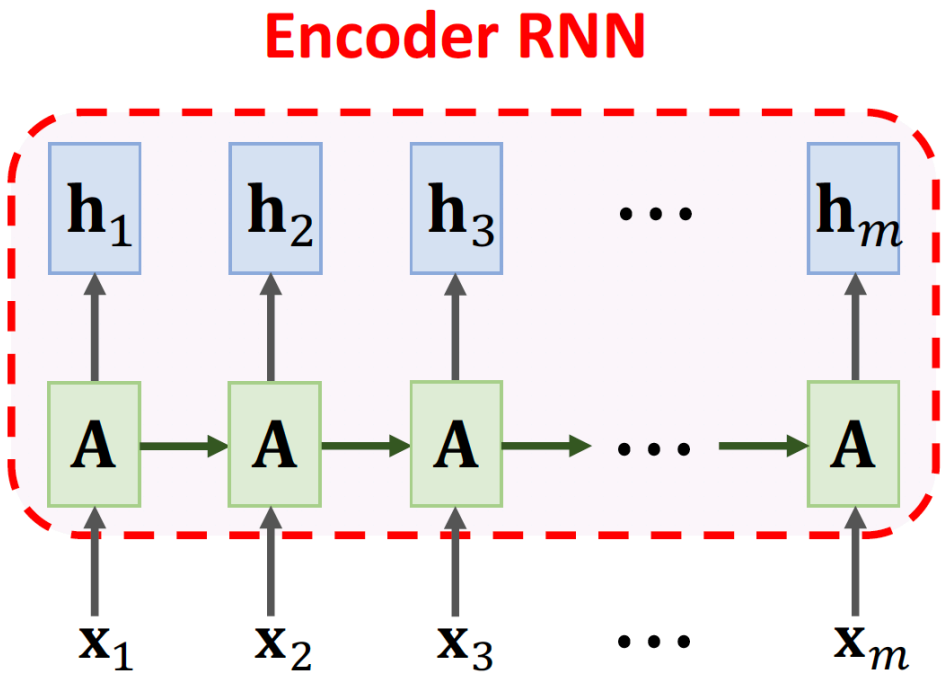
English		German
“do you agree”	=> **[Seq2Seq]** =>	“bist du einverstanden”
“go to sleep”	=> **[Seq2Seq]** =>	“gehen Sie schlafen”
“We will fight”	=> **[Seq2Seq]** =>	“Wir werden kämpfen”

Sequence-to-Sequence Model (Seq2Seq)

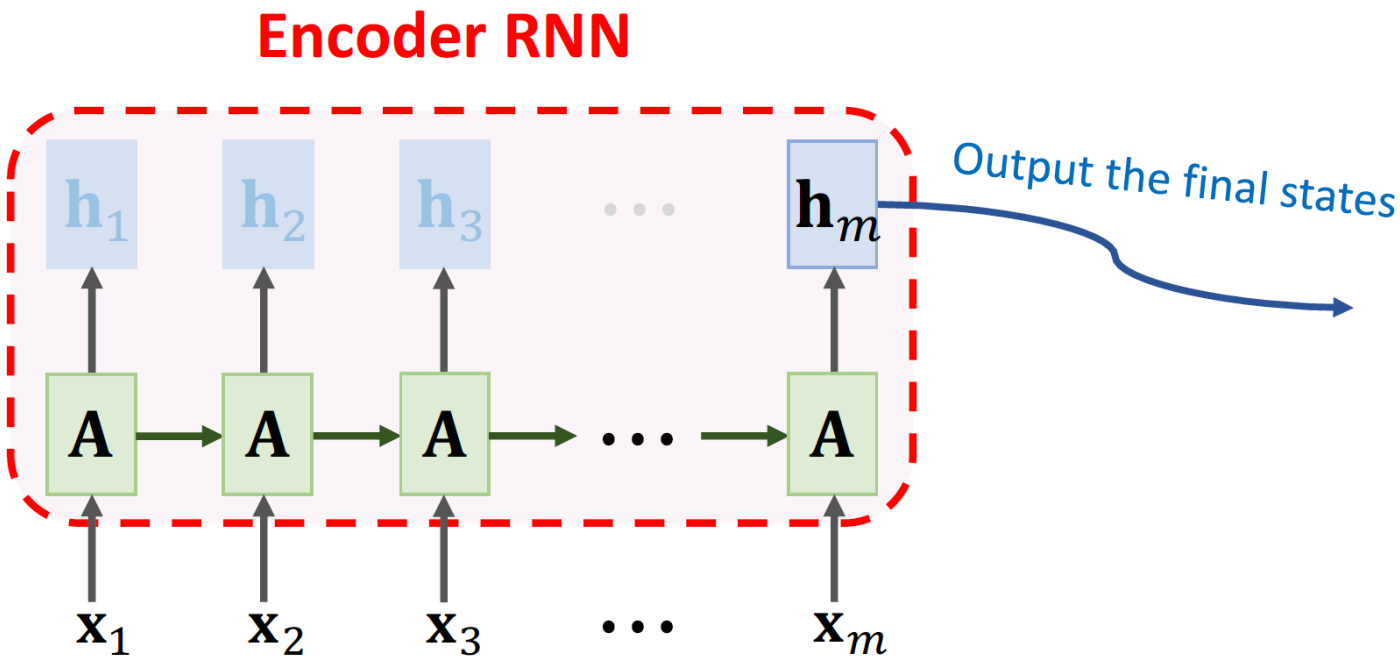
Seq2Seq Model

- Seq2Seq adopts an encoder-decoder architecture
  - Encoder is an RNN
  - Decoder is an RNN

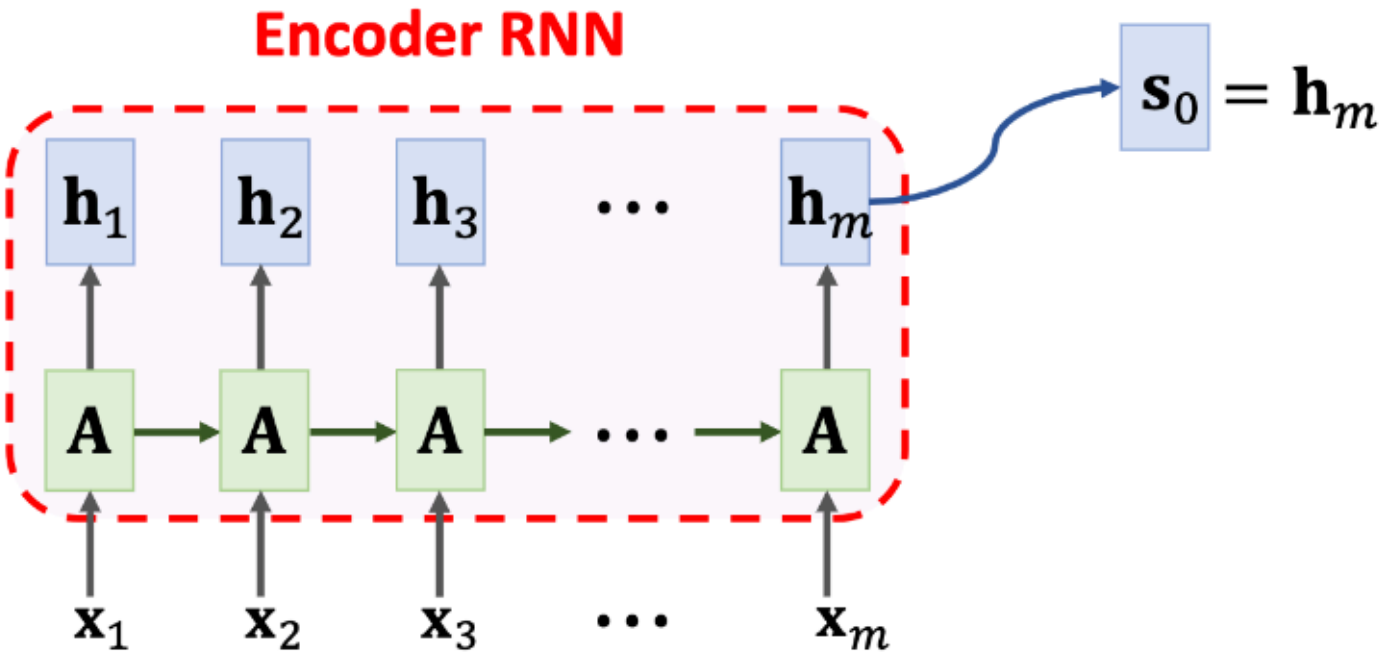
Seq2Seq Model



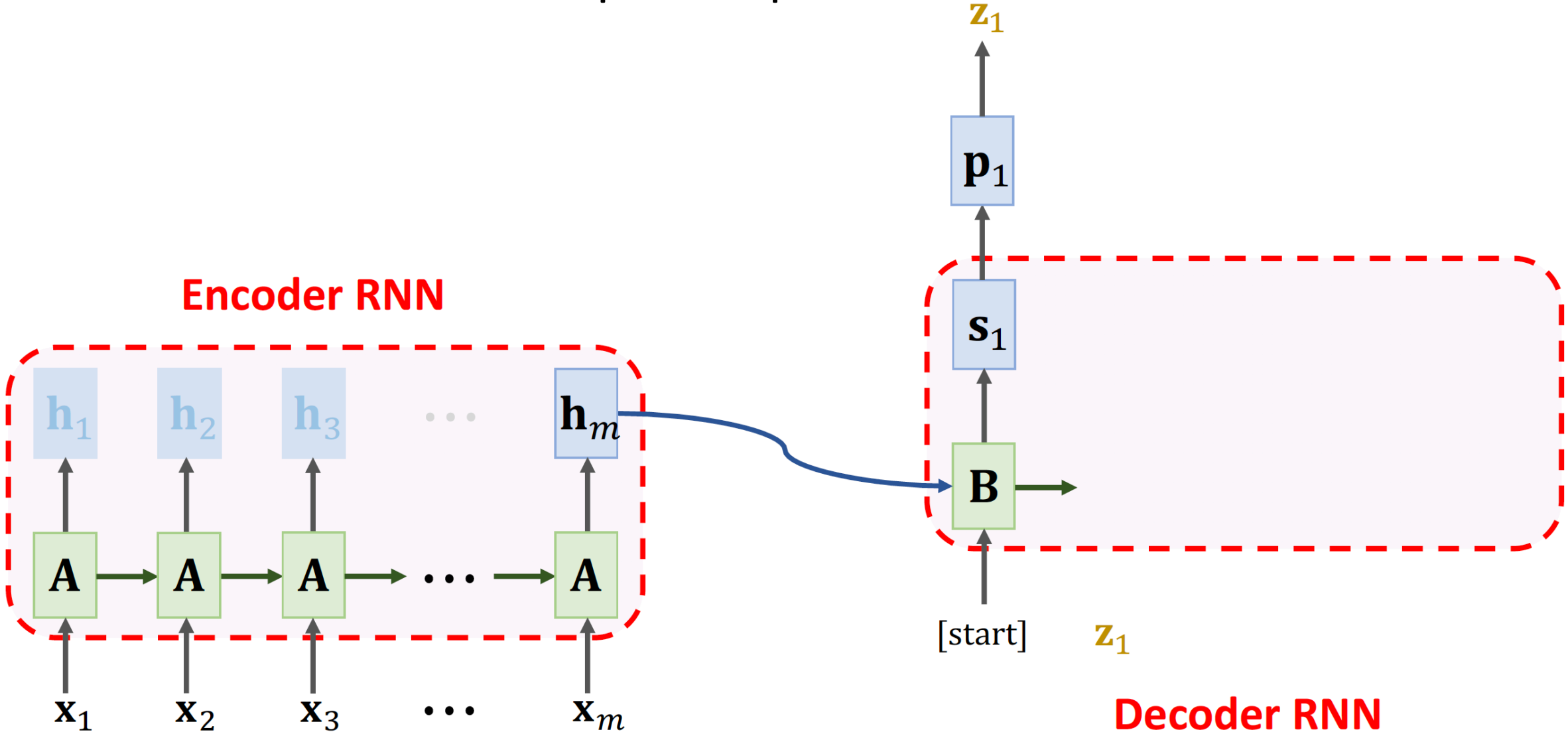
Seq2Seq Model



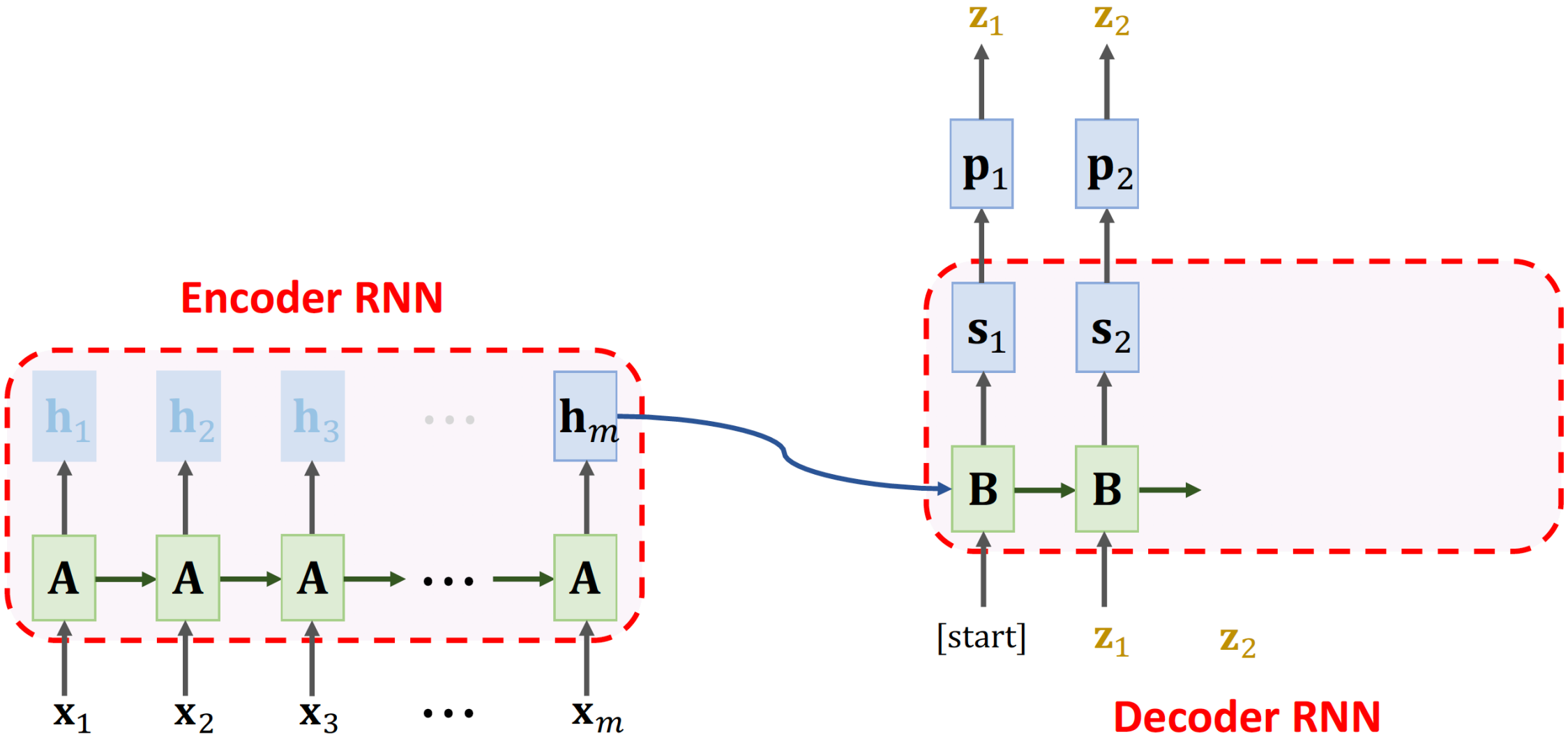
Seq2Seq Model



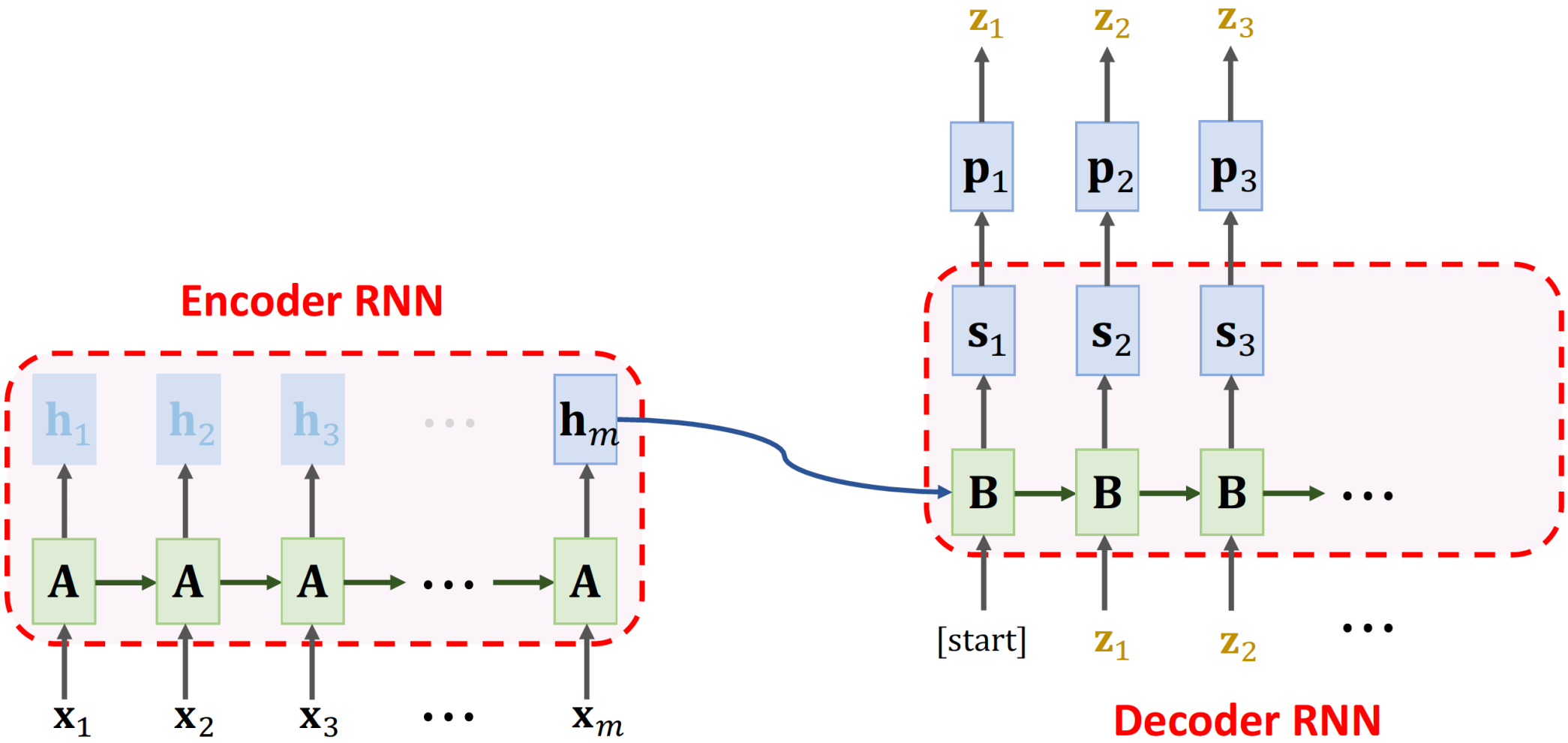
Seq2Seq Model



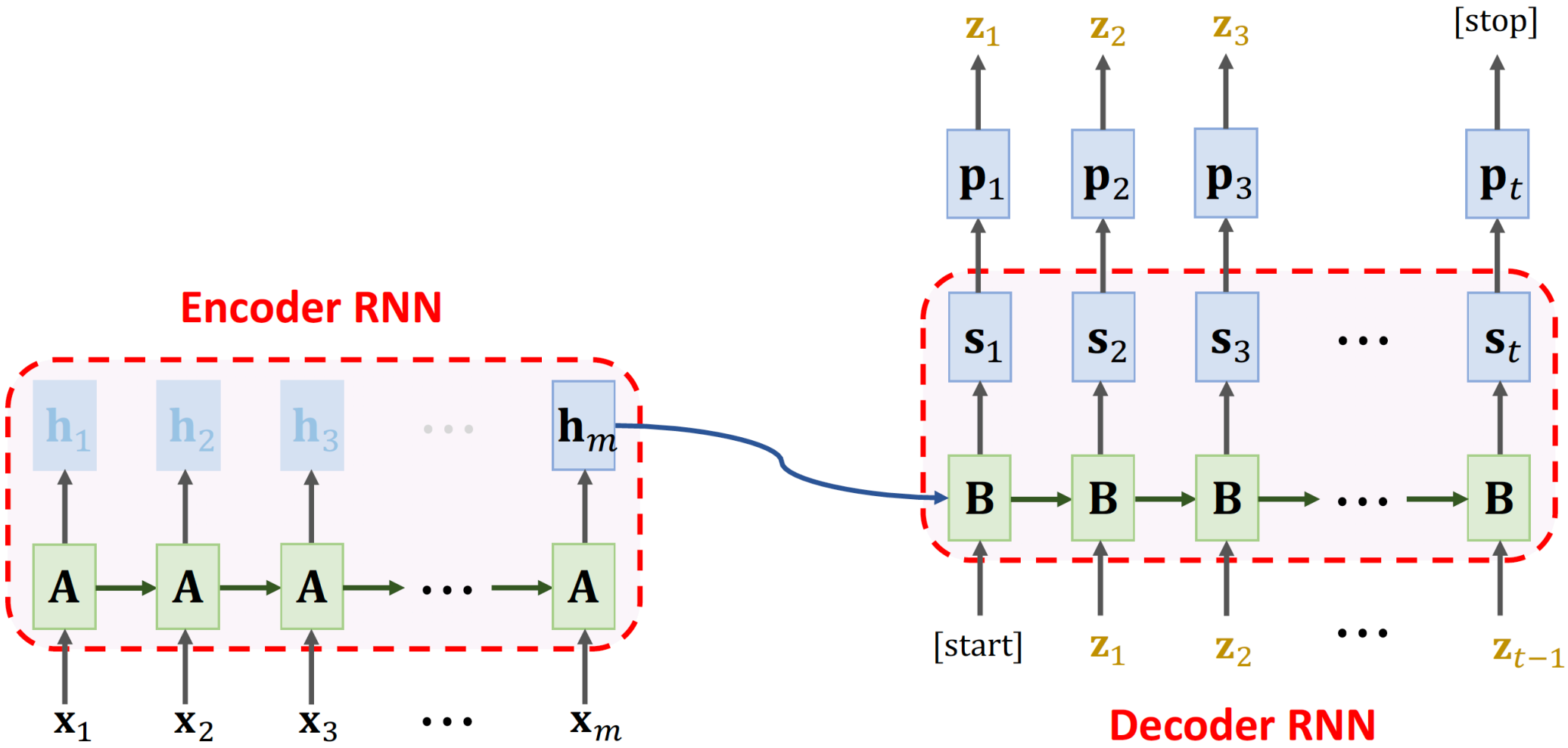
Seq2Seq Model



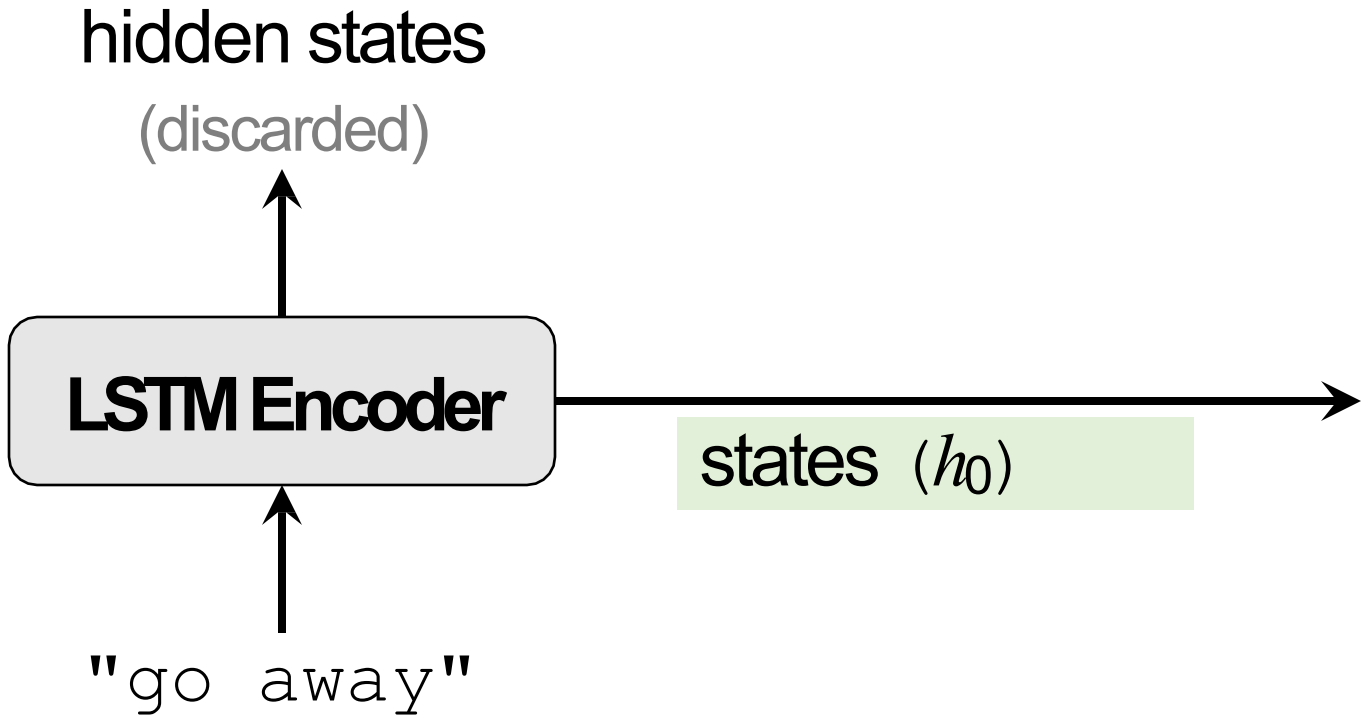
Seq2Seq Model



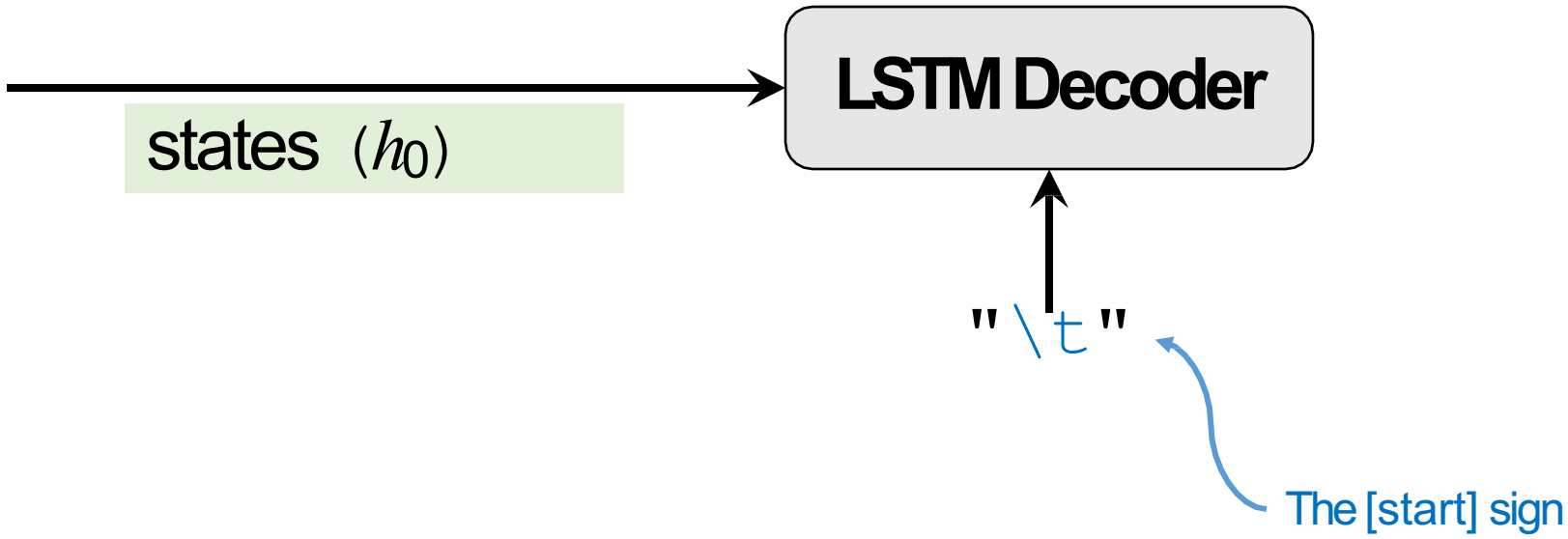
Seq2Seq Model



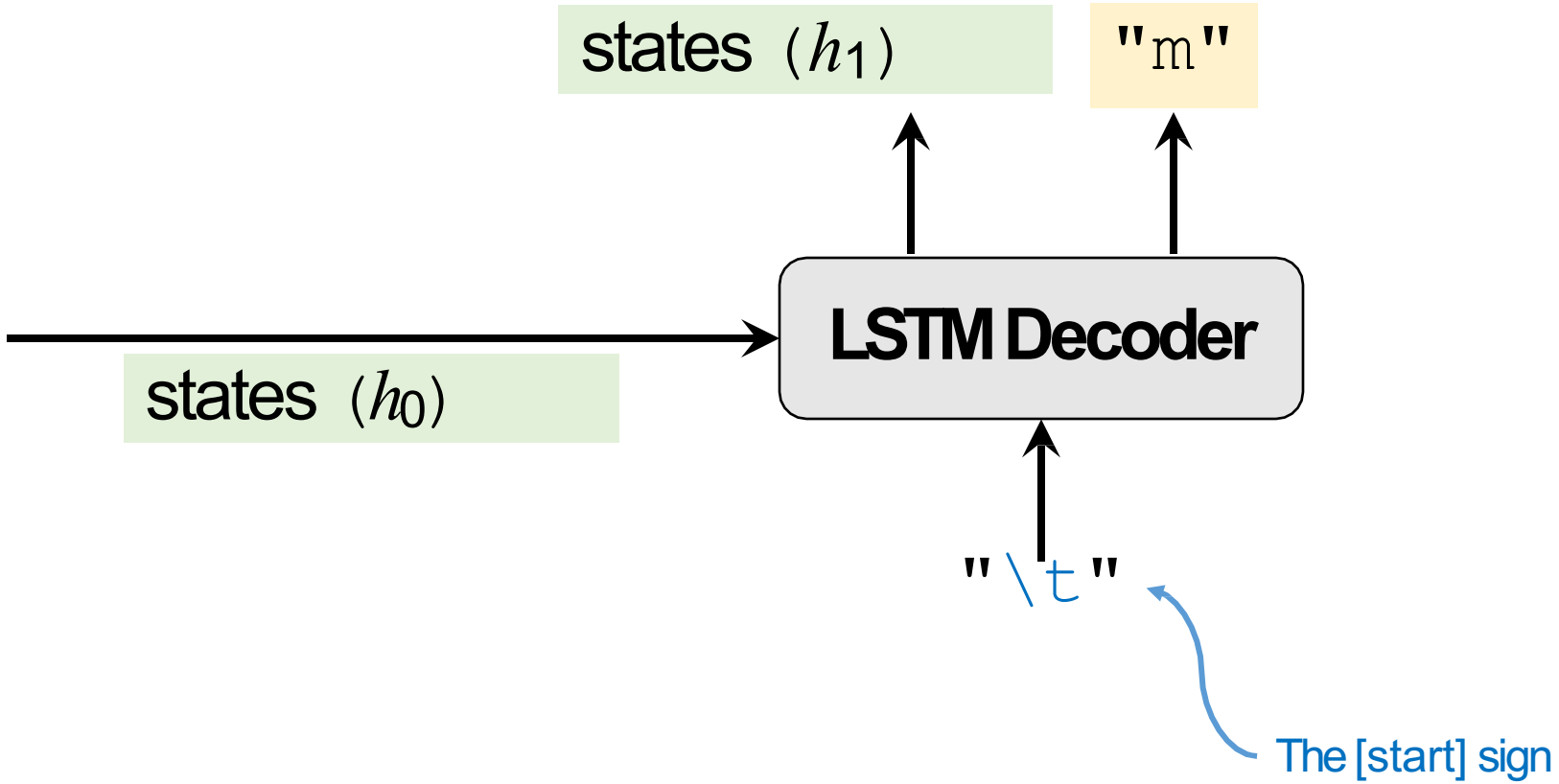
An example



An example

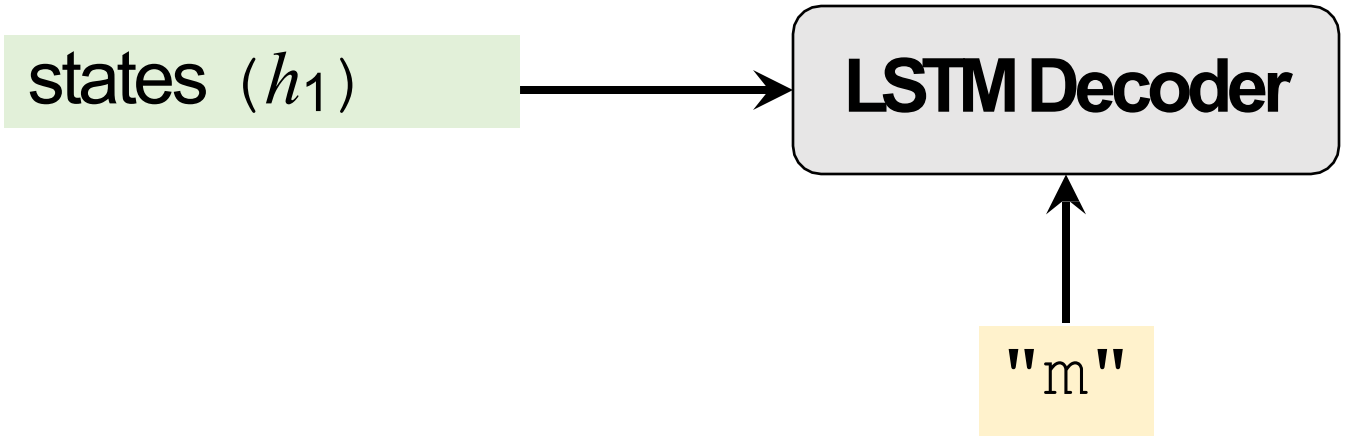


An example



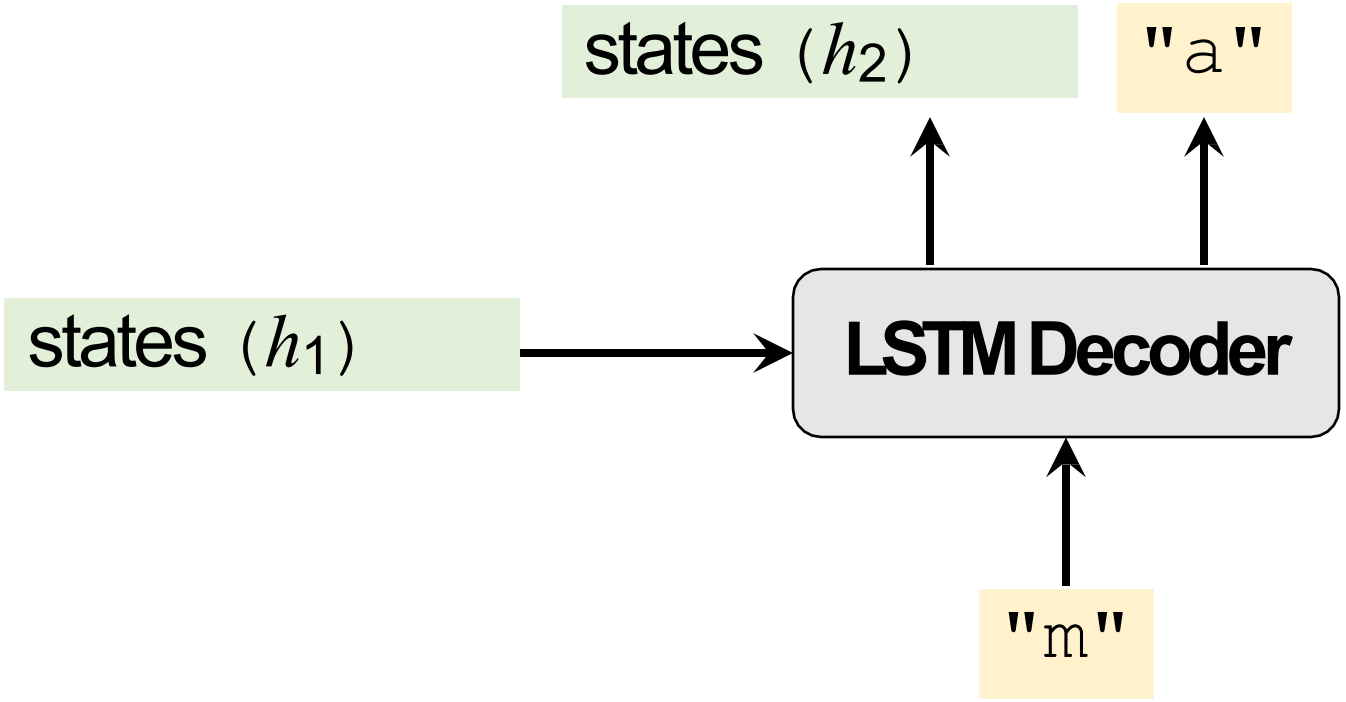
Record: "m"

An example



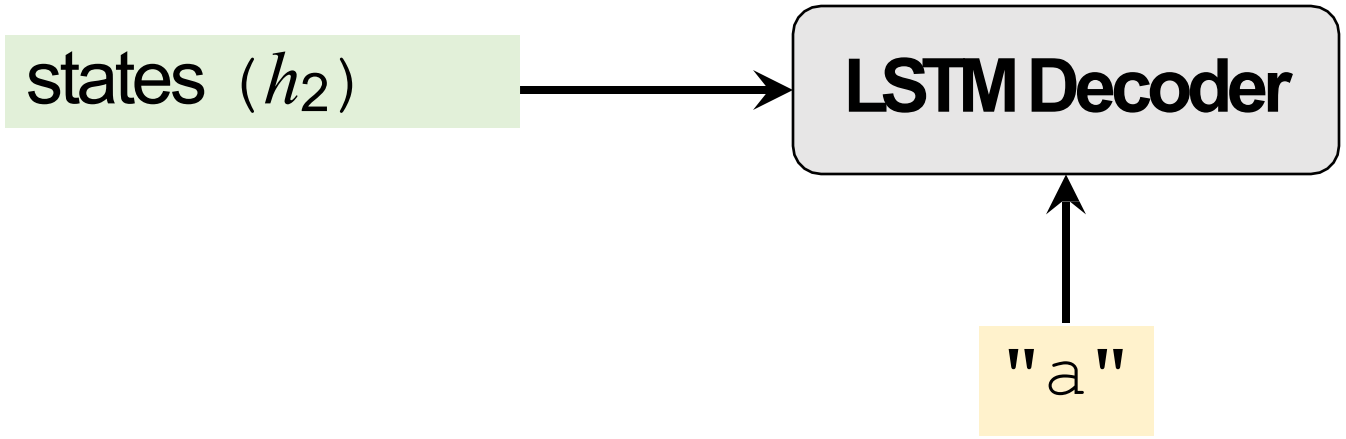
Record: "m"

An example



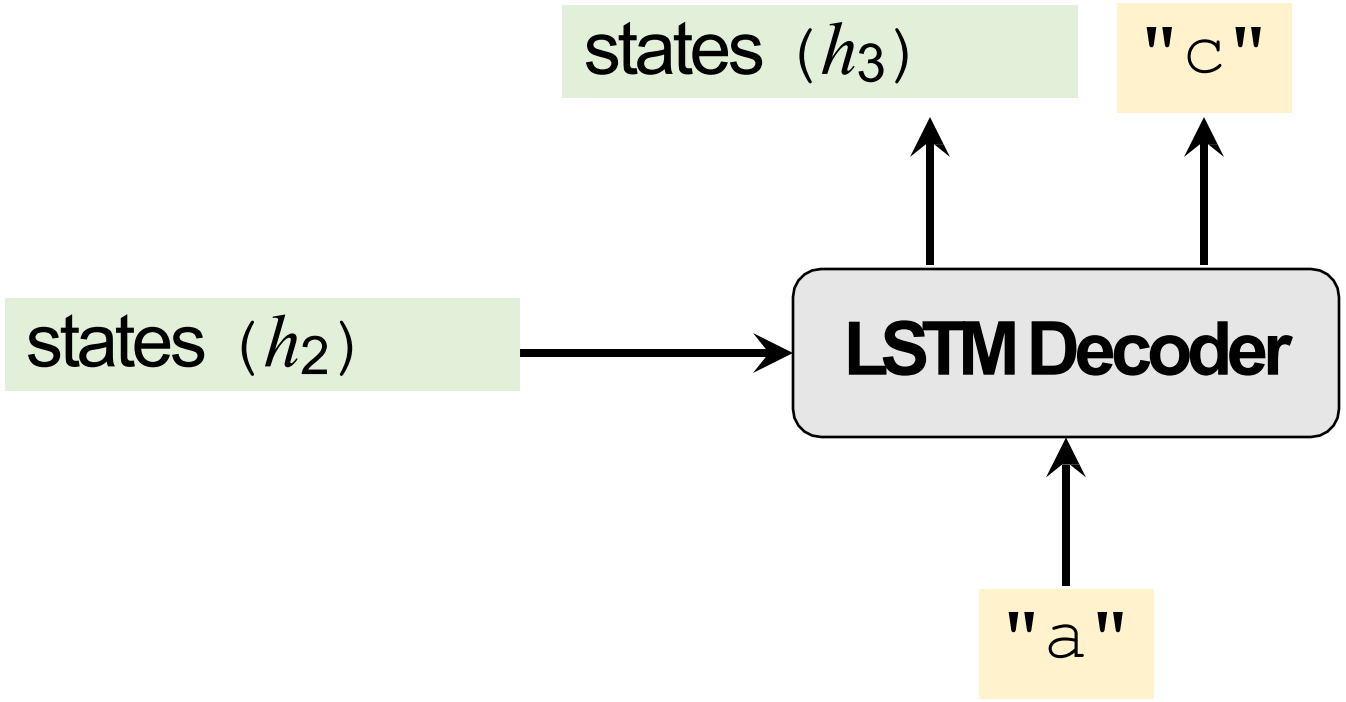
Record: "ma"

An example



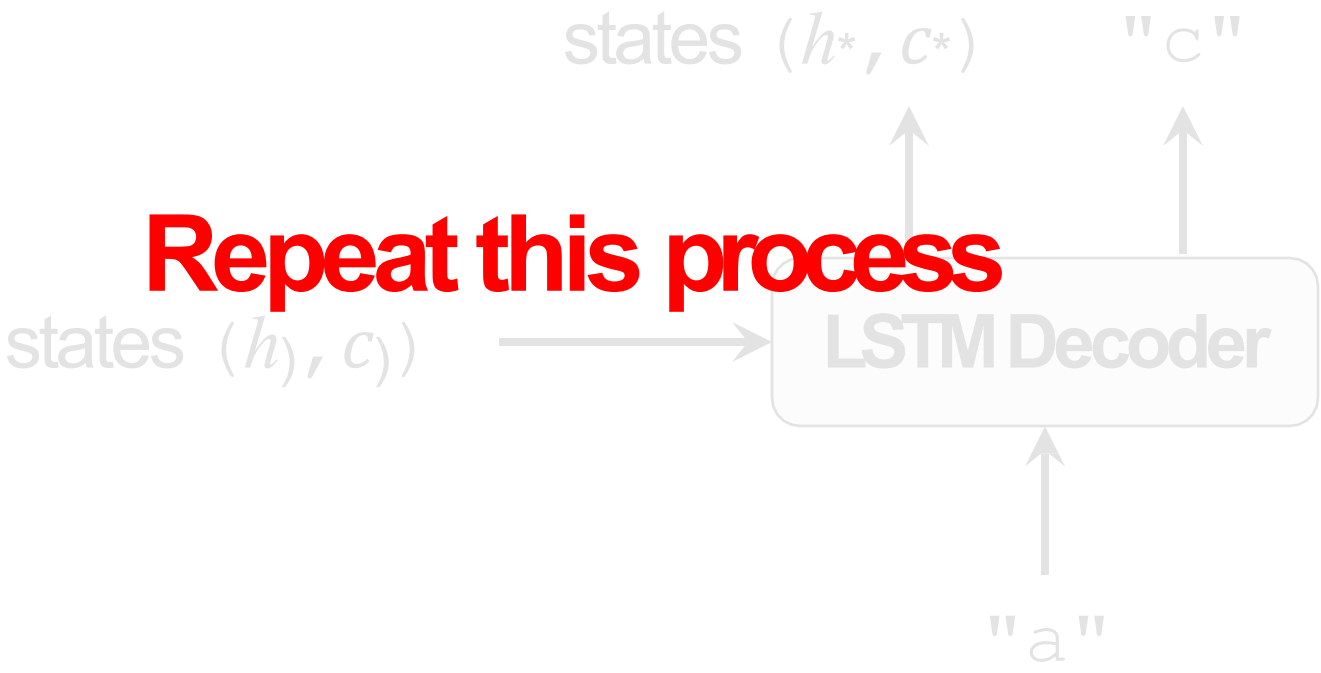
Record: "ma"

An example



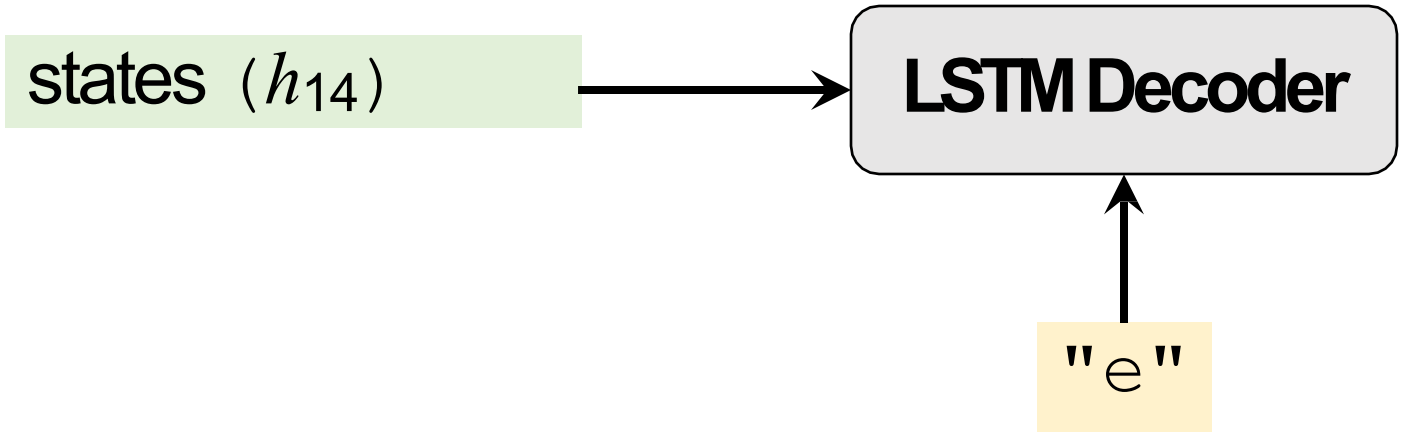
Record: "mac"

An example



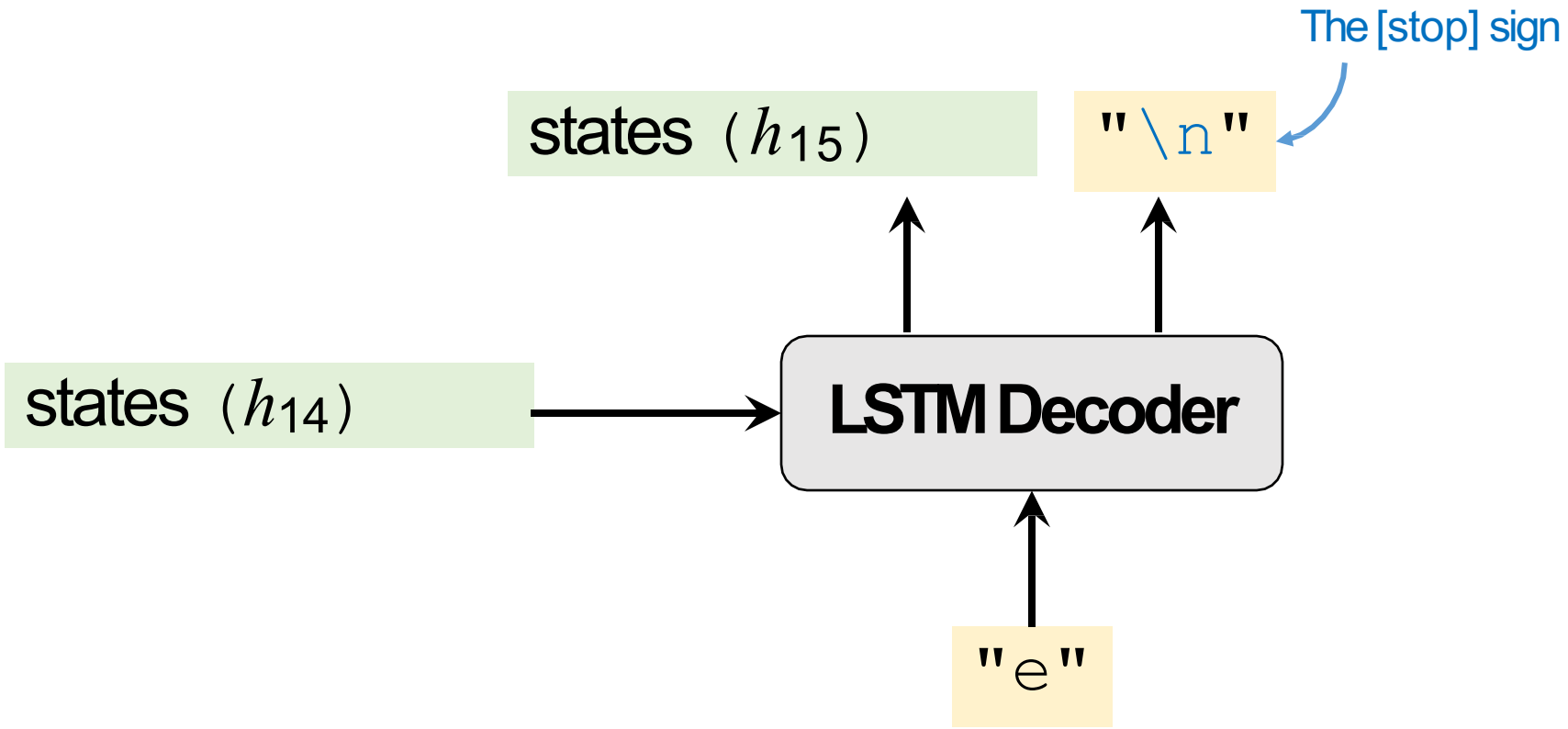
Record: "mac"

An example



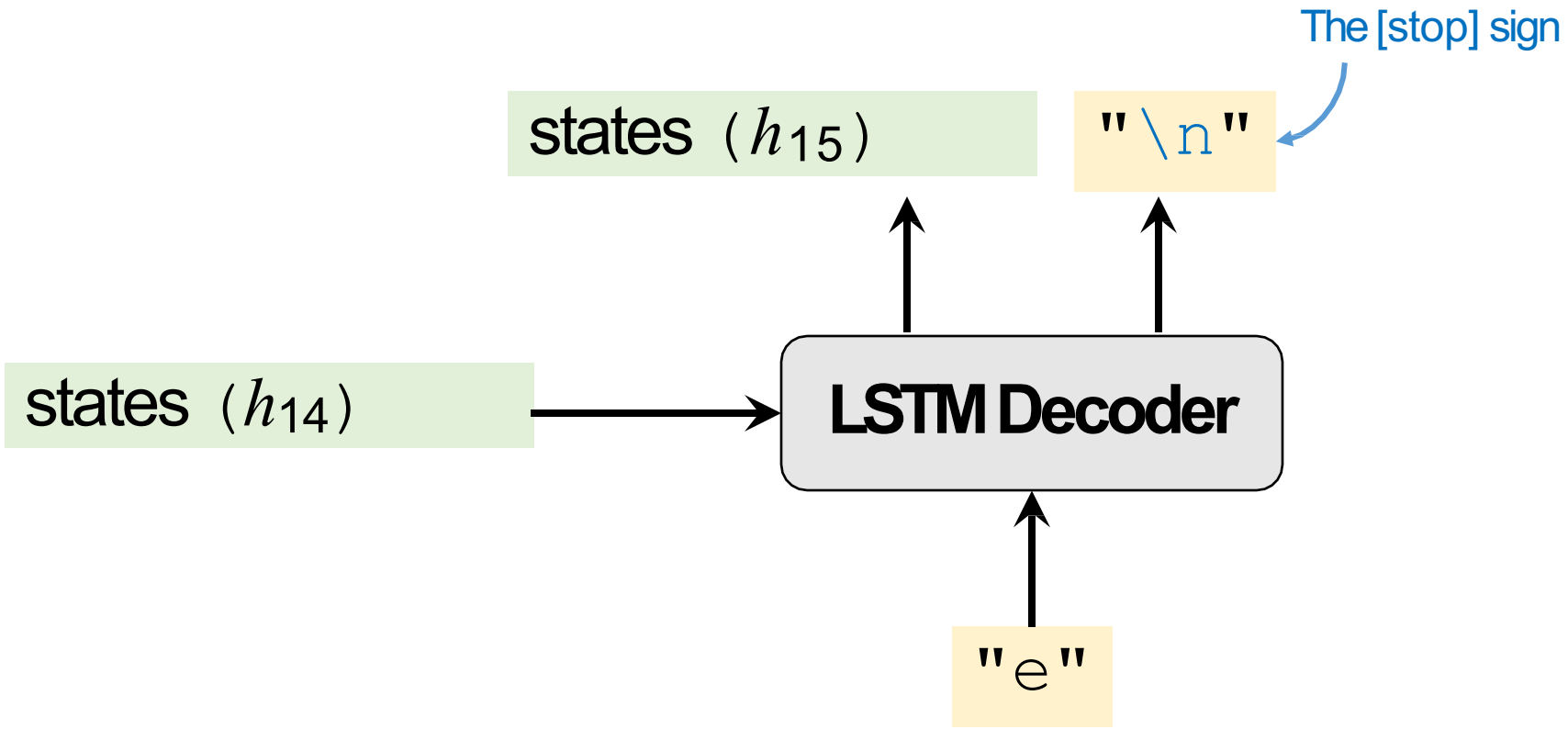
Record: "mach ne fliege"

An example



Record: "mach ne fliege"

An example



Return: "mach ne fliege"

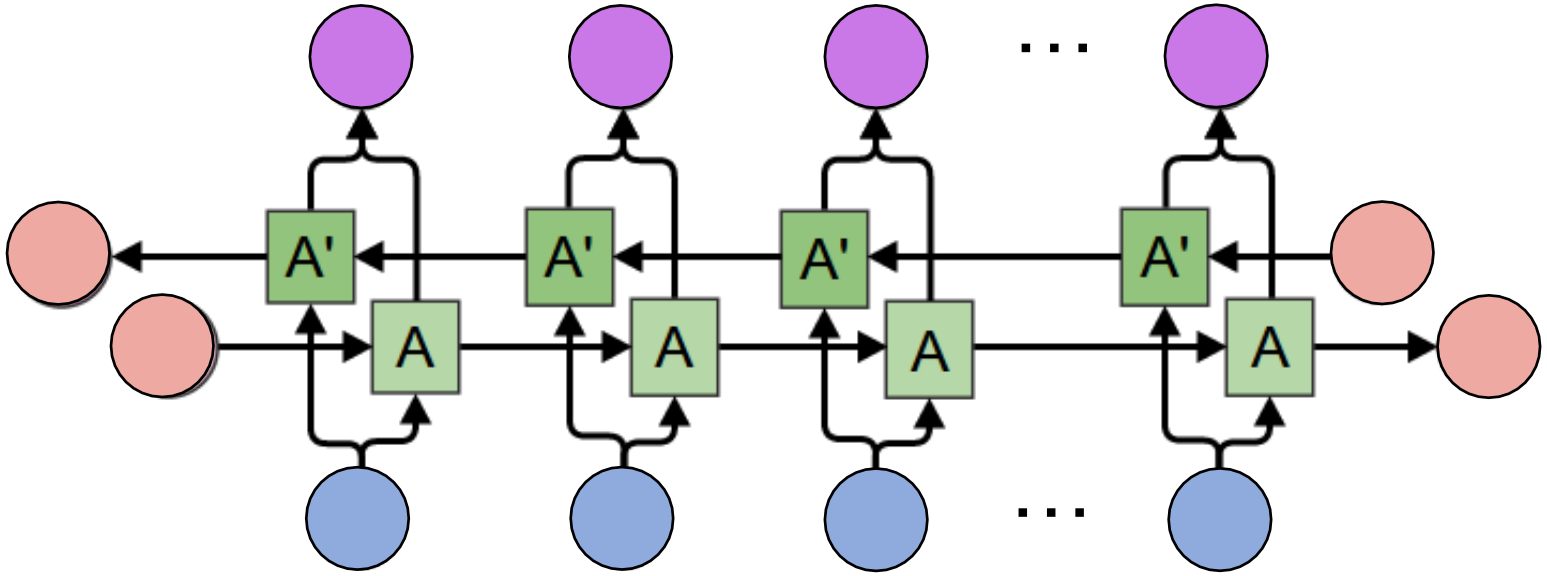
How to Improve?

Bi-LSTM instead of LSTM (Encoder only!)

- Encoder's final states have all the information of the English sentence.
- If the sentence is long, the final states have forgotten early inputs.

Bi-LSTM instead of LSTM (Encoder only!)

- Encoder's final states have all the information of the English sentence.
- If the sentence is long, the final states have forgotten early inputs.
- Bi-LSTM (left-to-right and right-to-left) has longer memory.



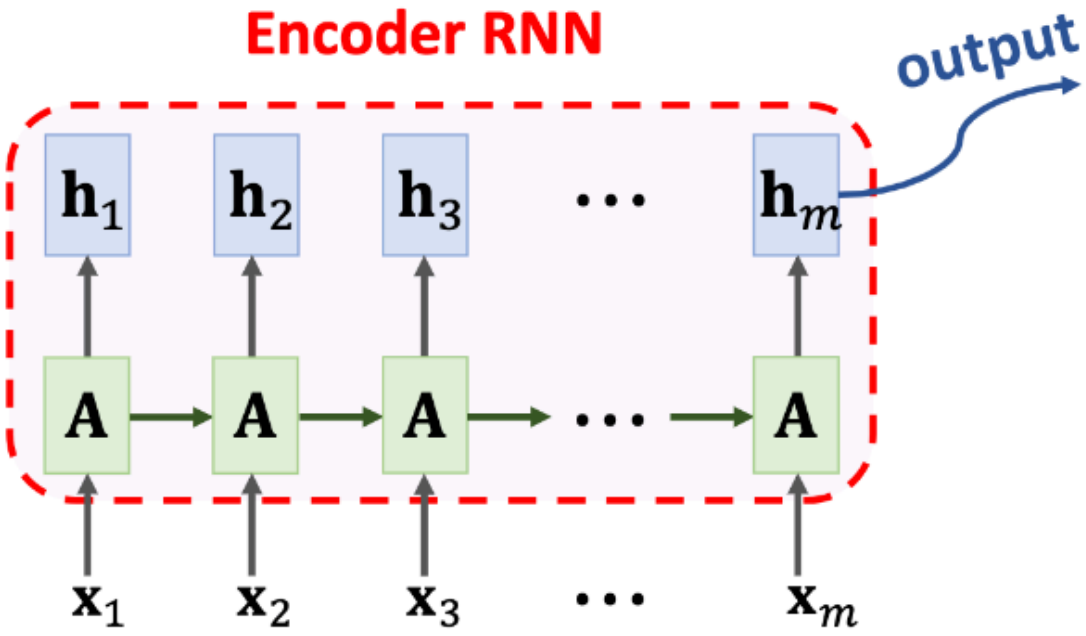
Bi-LSTM instead of LSTM (Encoder only!)

- Encoder's final states have all the information of the English sentence.
- If the sentence is long, the final states have forgotten early inputs.
- Bi-LSTM (left-to-right and right-to-left) has longer memory.
- Use Bi-LSTM in the encoder; use unidirectional LSTM in the decoder.

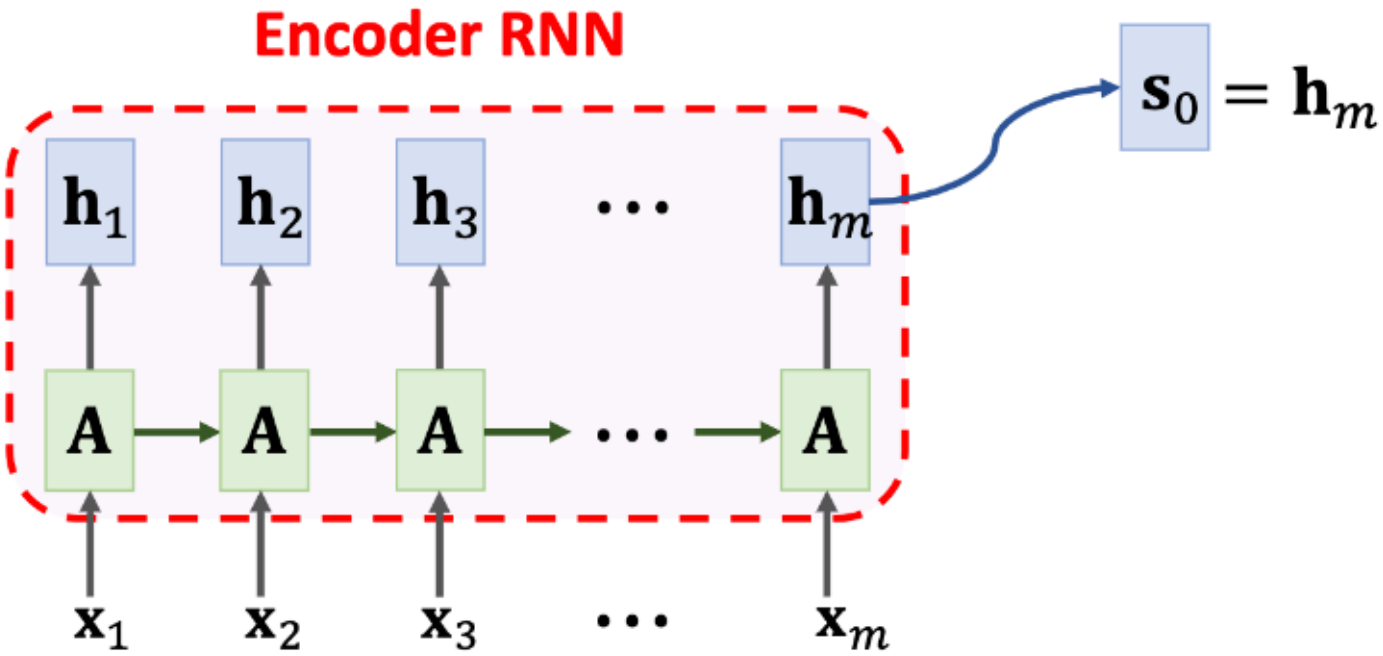
How to Improve?

1. Bi-LSTM instead of LSTM. (Encoder only!)
2. Attention

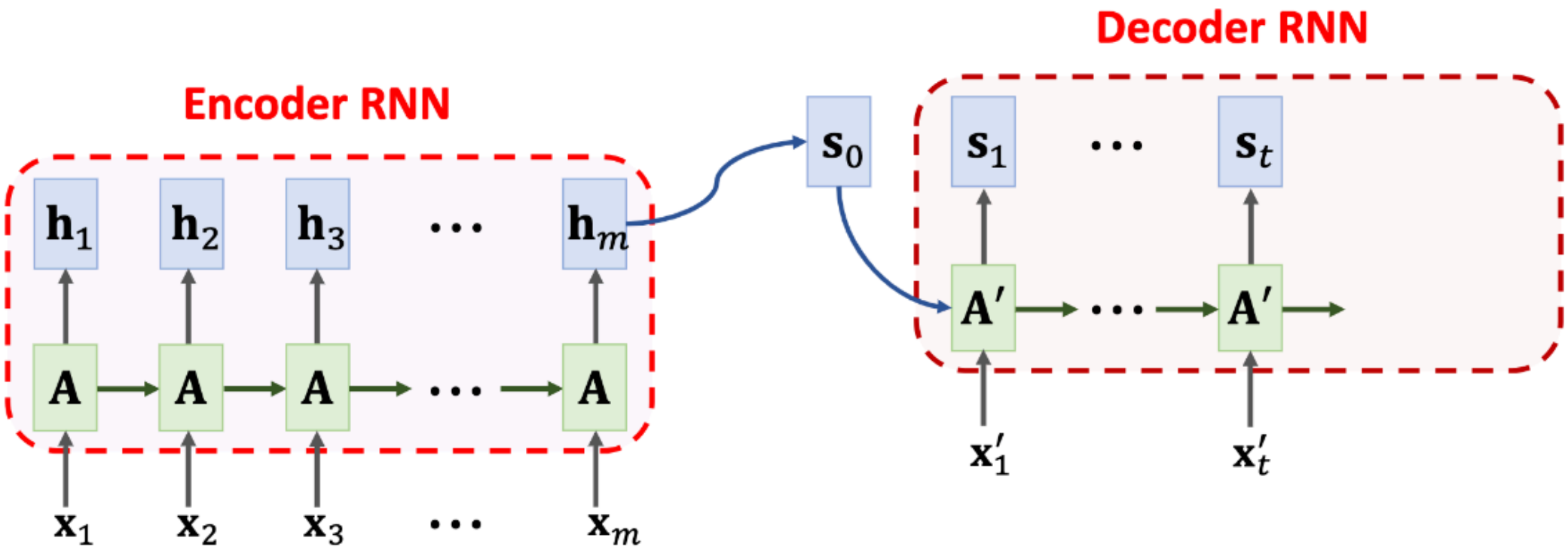
Seq2Seq Model



Seq2Seq Model

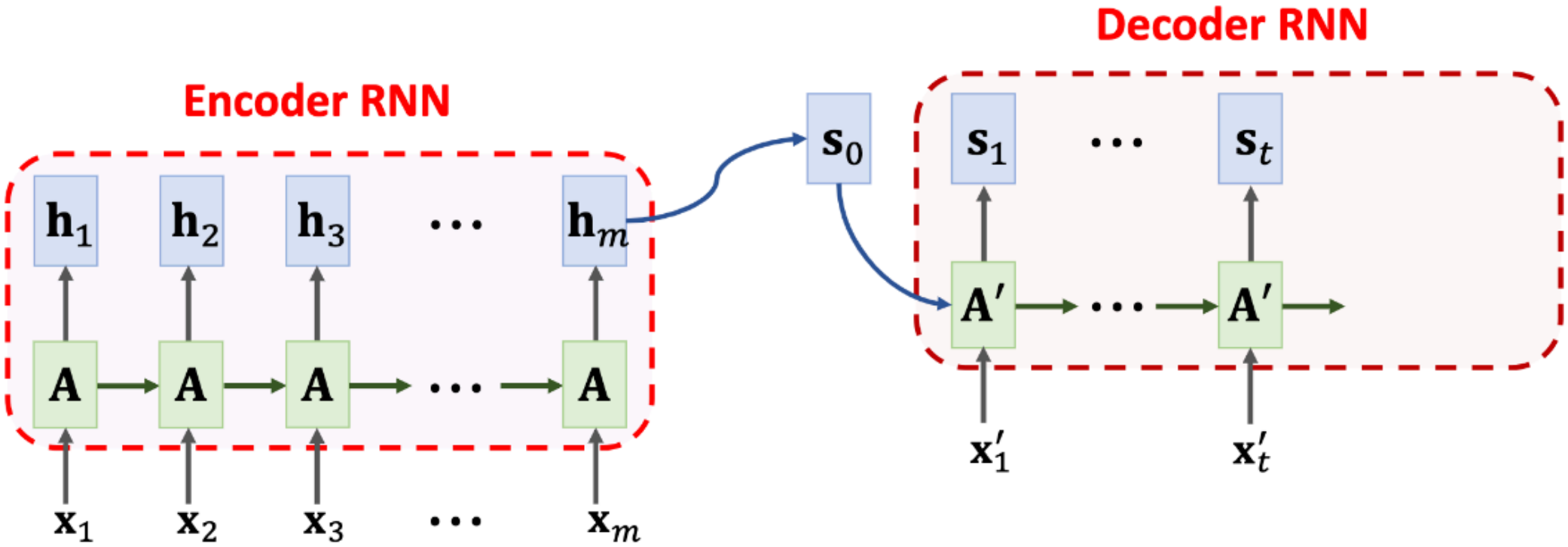


Seq2Seq Model



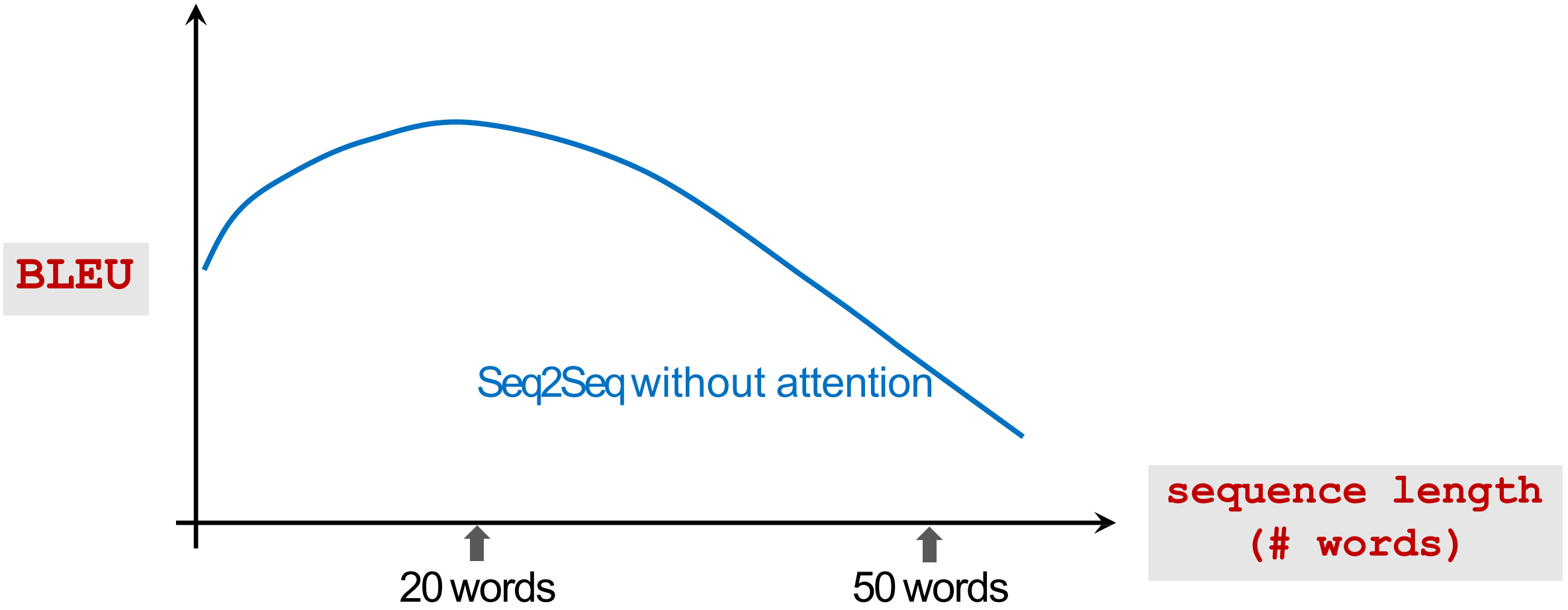
Seq2Seq Model

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



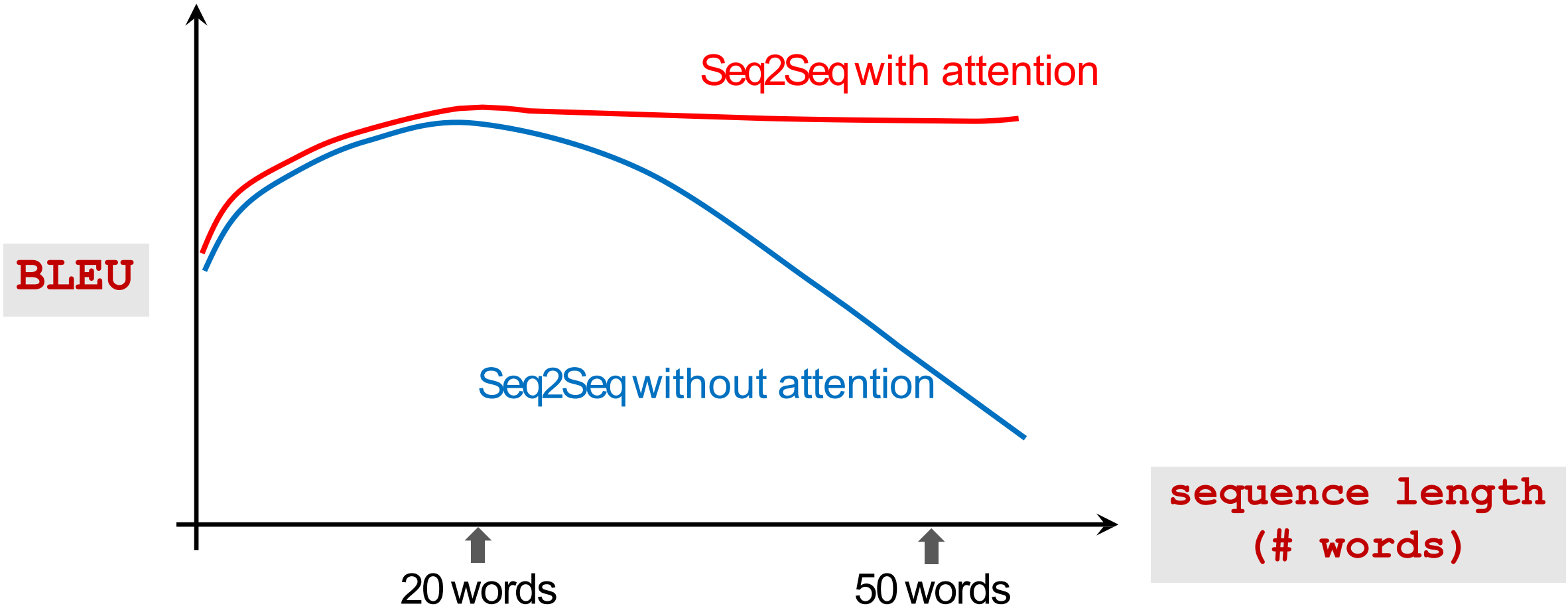
Seq2Seq Model

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



Seq2Seq Model

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



Attention for Seq2Seq Model

Seq2Seq Model with Attention

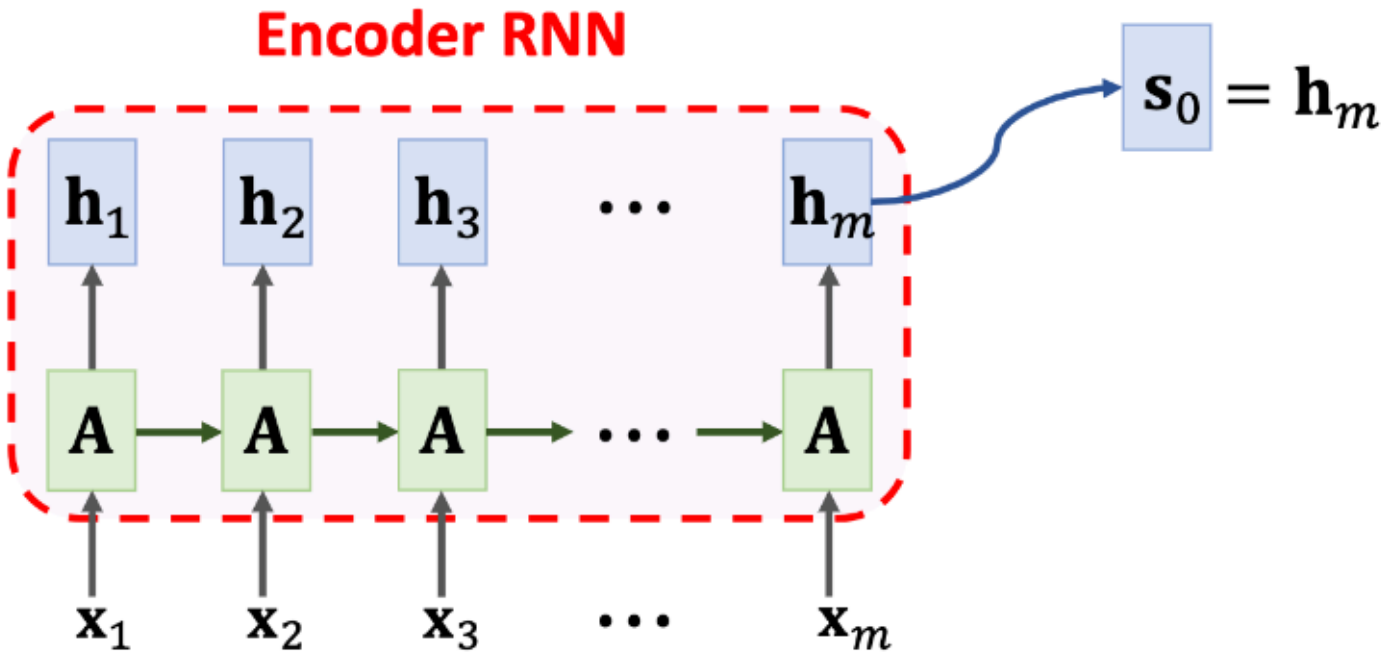
- Attention tremendously improves Seq2Seq model.
- With attention, Seq2Seq model does not forget source input.
- With attention, the decoder knows where to focus.
- Downside: much more computation.

Original paper:

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

SimpleRNN + Attention

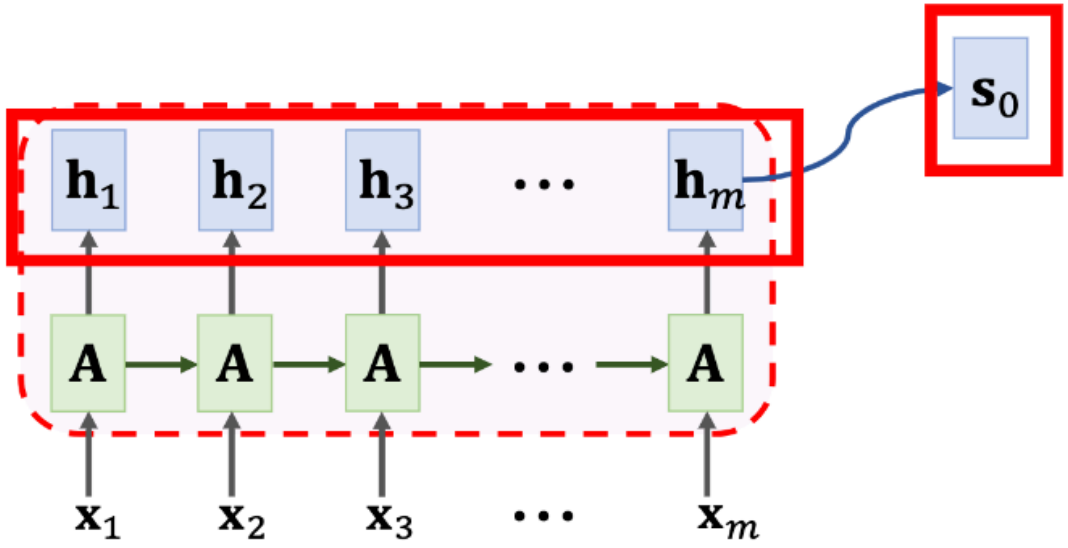
Root problem: S0 is assigned with hm ONLY



SimpleRNN + Attention

Key idea:

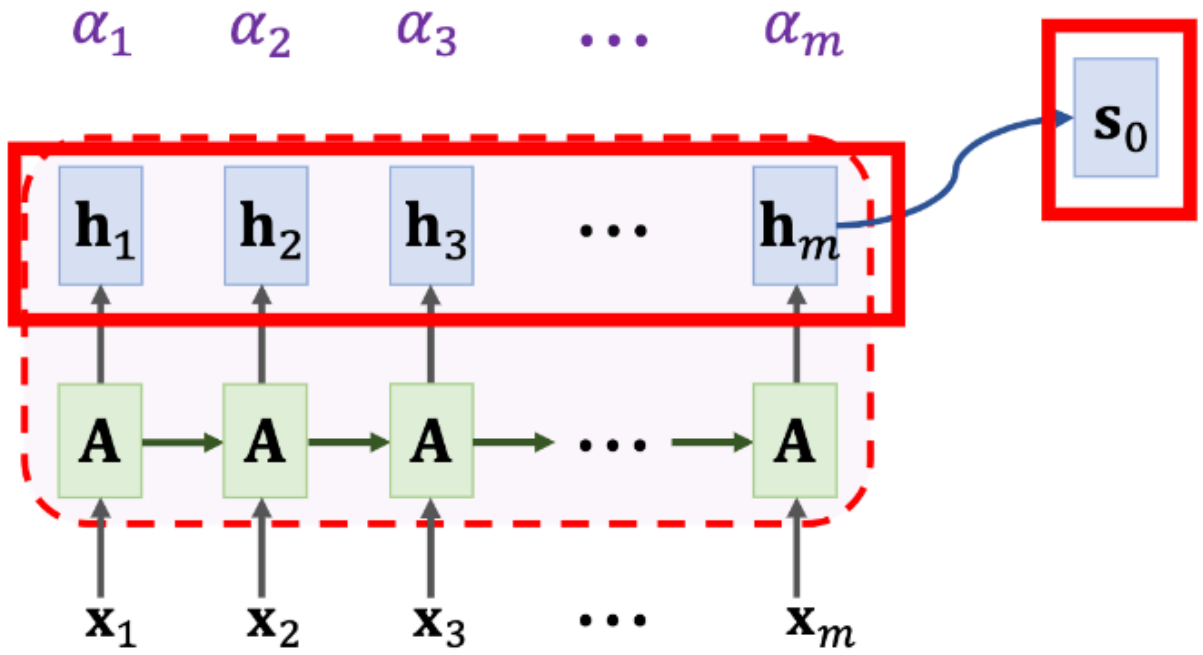
s_0 should be explicitly dependent on all the hi



SimpleRNN + Attention

Key idea:

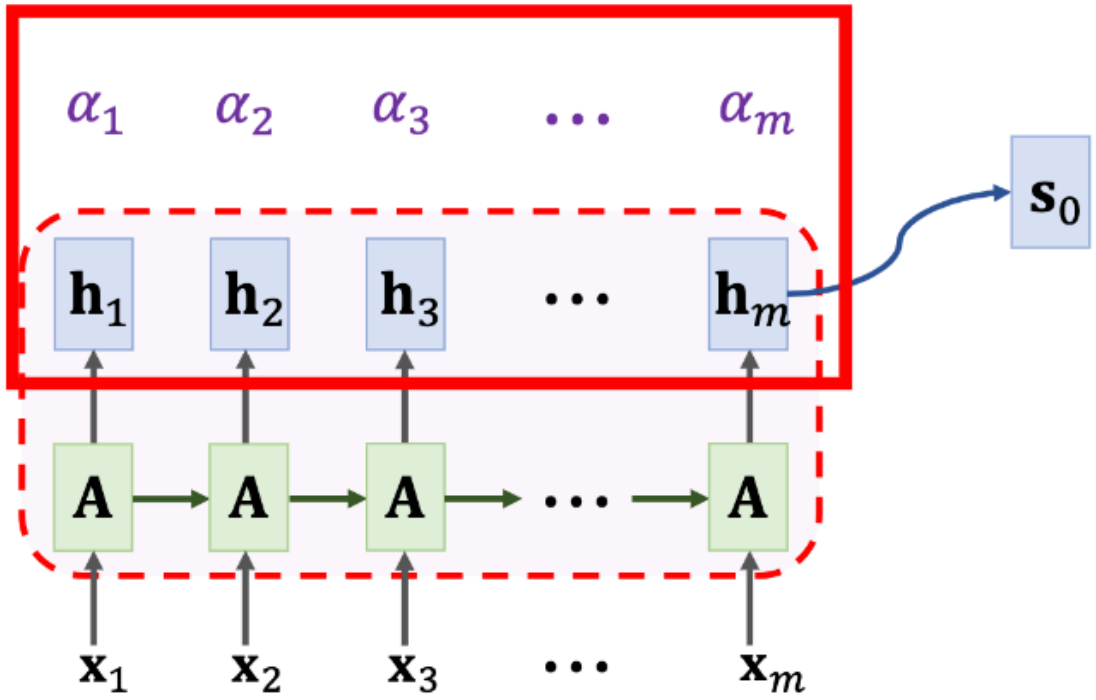
S_0 should be explicitly dependent on all the hi



SimpleRNN + Attention

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

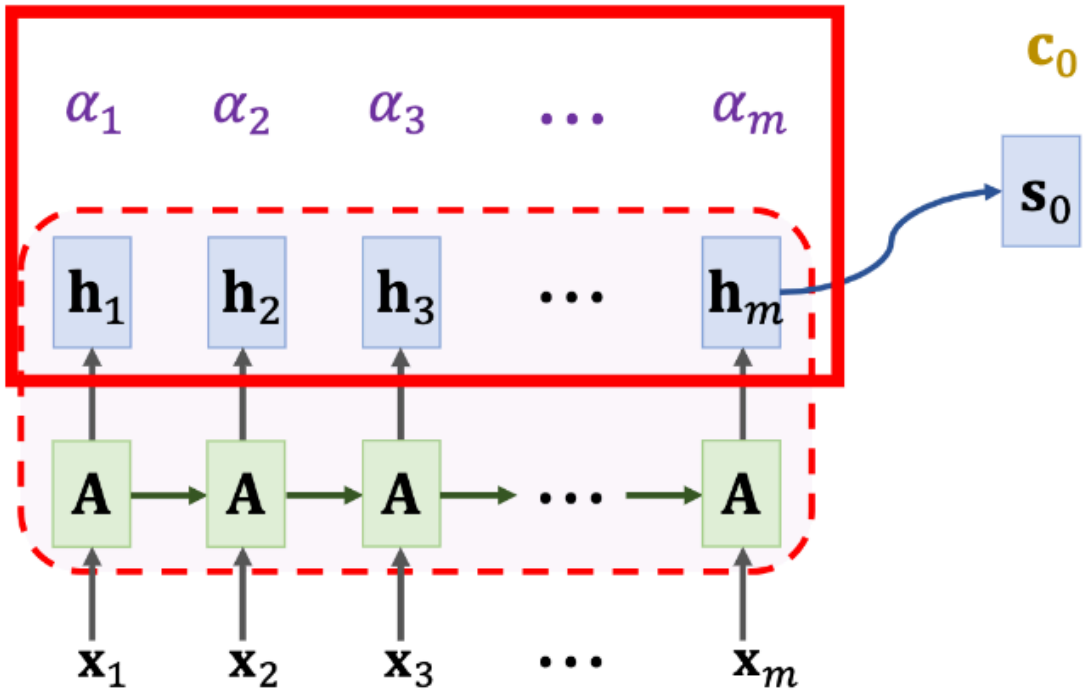
Context vector: $\mathbf{c}_0 = \alpha_1 \mathbf{h}_1 + \cdots + \alpha_m \mathbf{h}_m$.



SimpleRNN + Attention

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

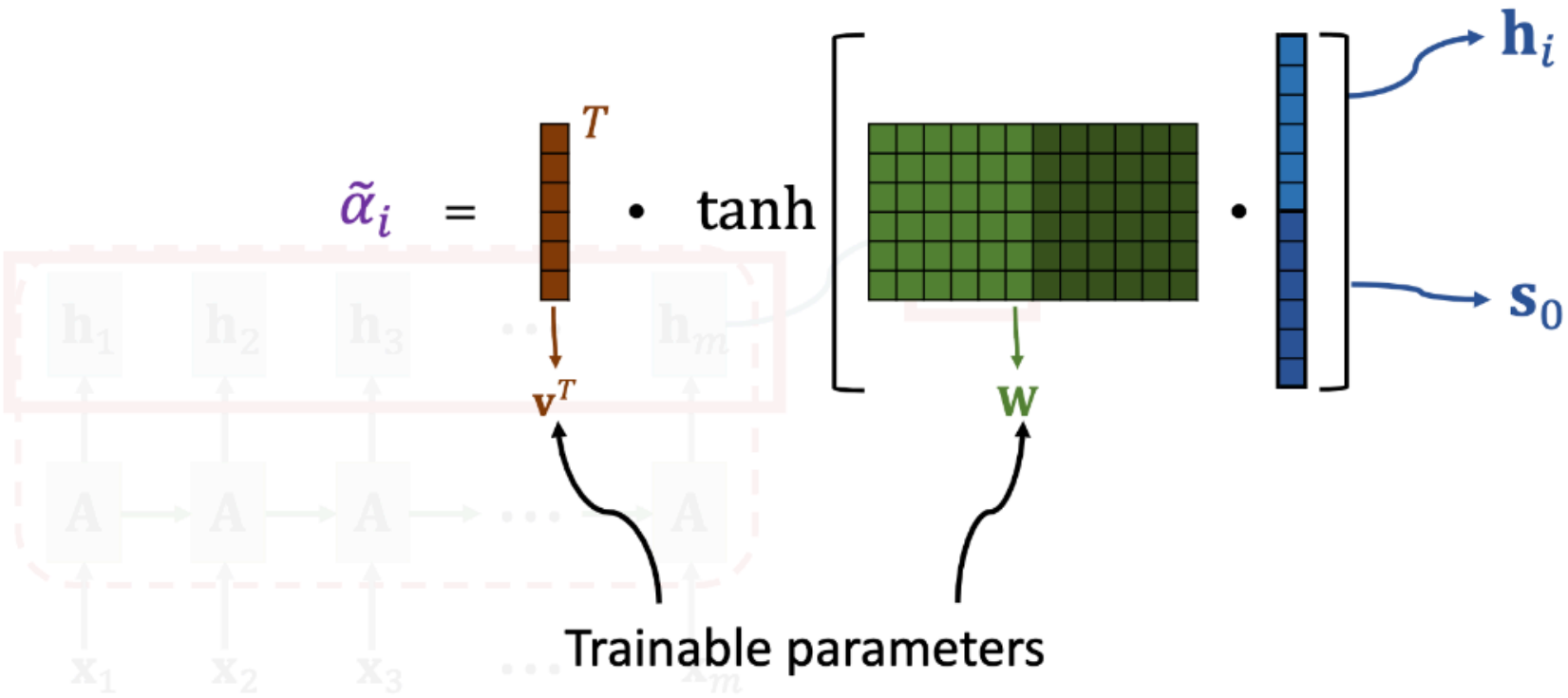
Context vector: $\mathbf{c}_0 = \alpha_1 \mathbf{h}_1 + \cdots + \alpha_m \mathbf{h}_m$.



SimpleRNN + Attention

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

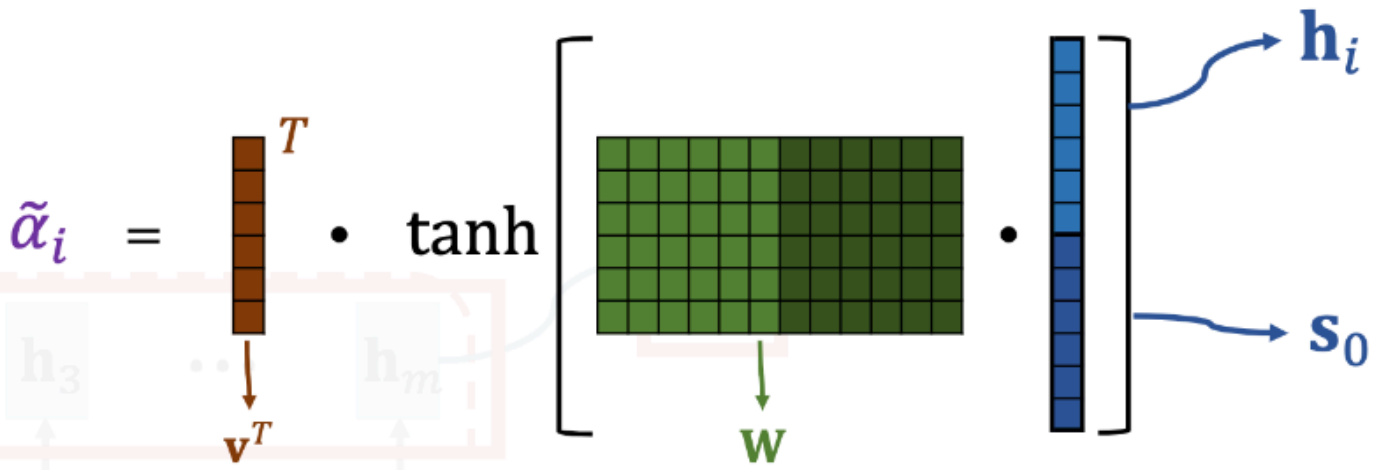
Option 1 (used in the original paper):



SimpleRNN + Attention

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

Option 1 (used in the original paper):



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

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

SimpleRNN + Attention

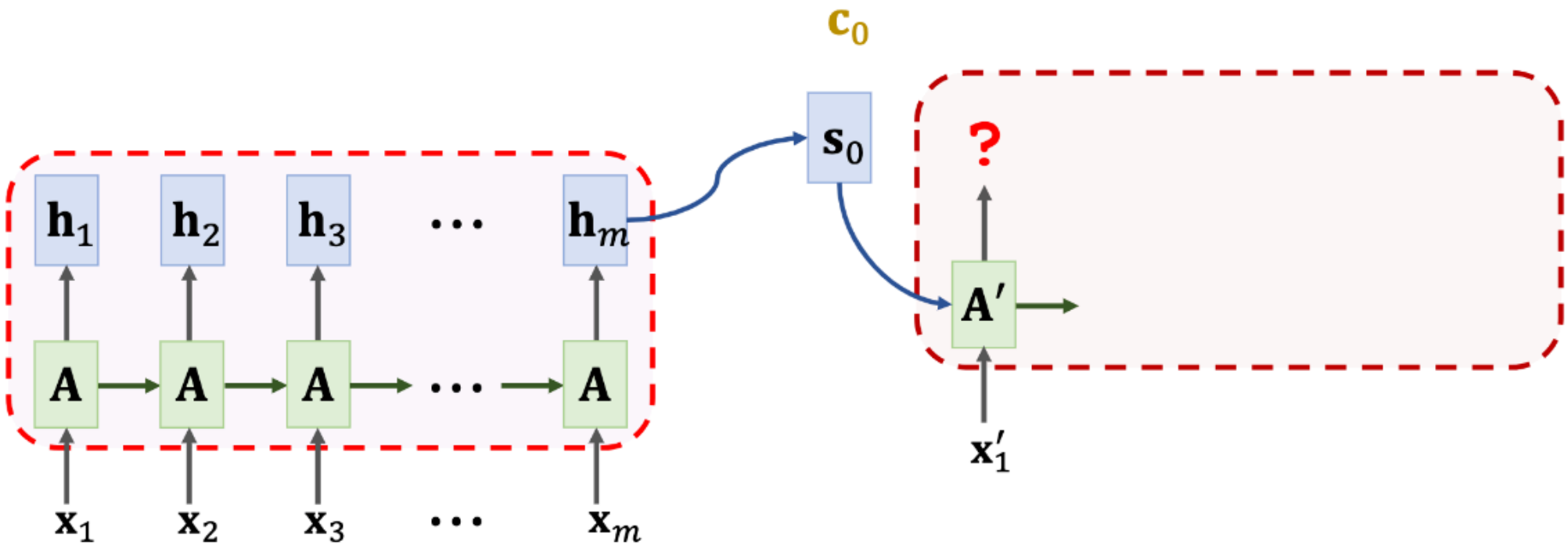
Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

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:
   - $\tilde{\alpha}_i = \mathbf{k}_i^T \mathbf{q}_0$, for $i = 1$ to m.
3. Normalization:
   - $[\alpha_1, \cdots, \alpha_m] = \text{Softmax}\,([\tilde{\alpha}_1, \cdots, \tilde{\alpha}_m])$.

Why do we need W_K and W_Q?

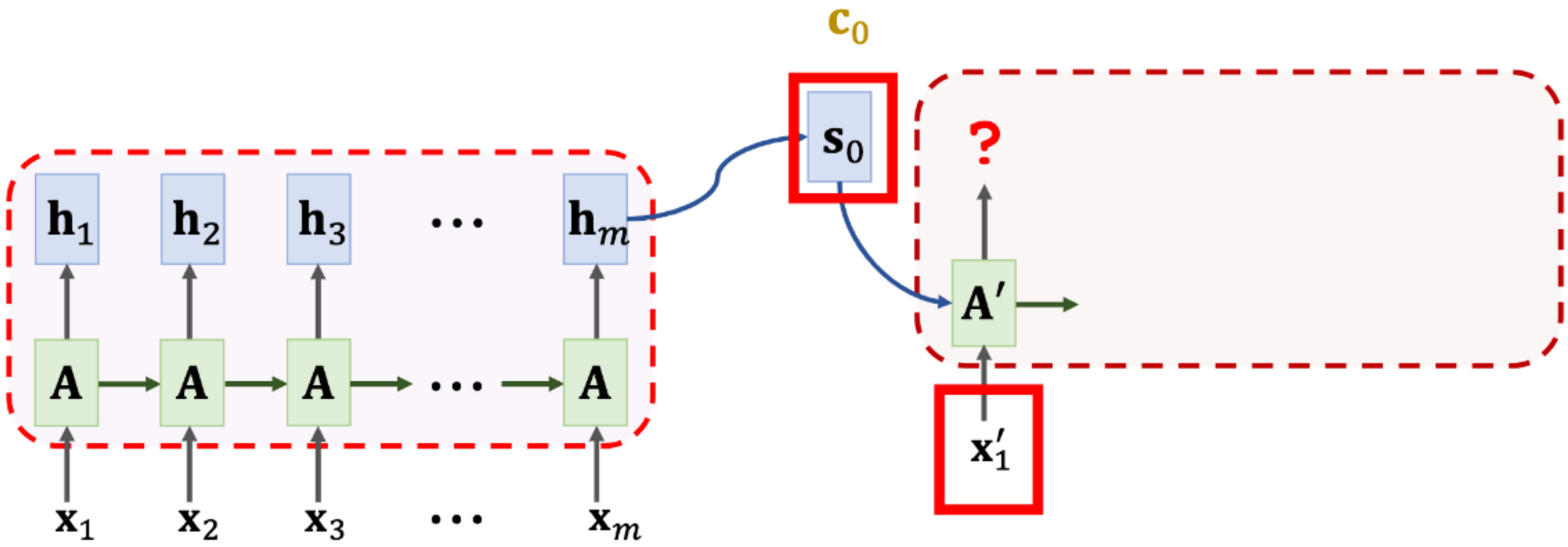
SimpleRNN + Attention



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

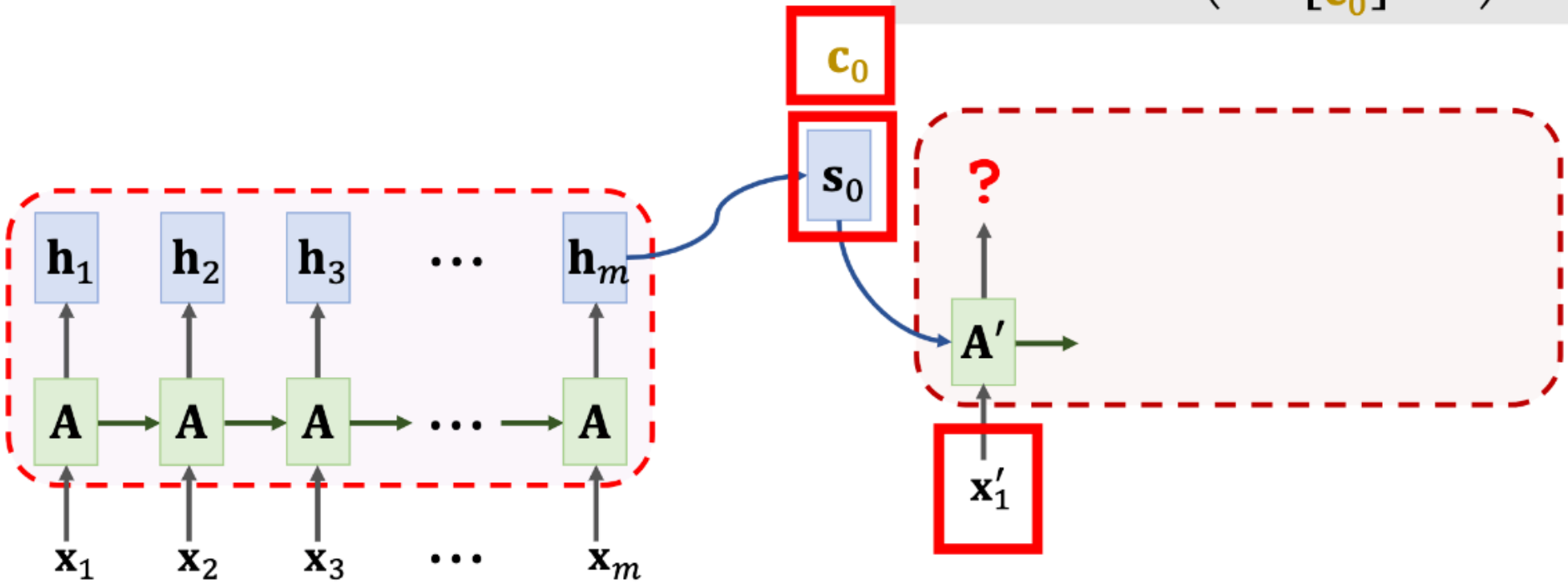
Should we incorporate S_0 here?

SimpleRNN:

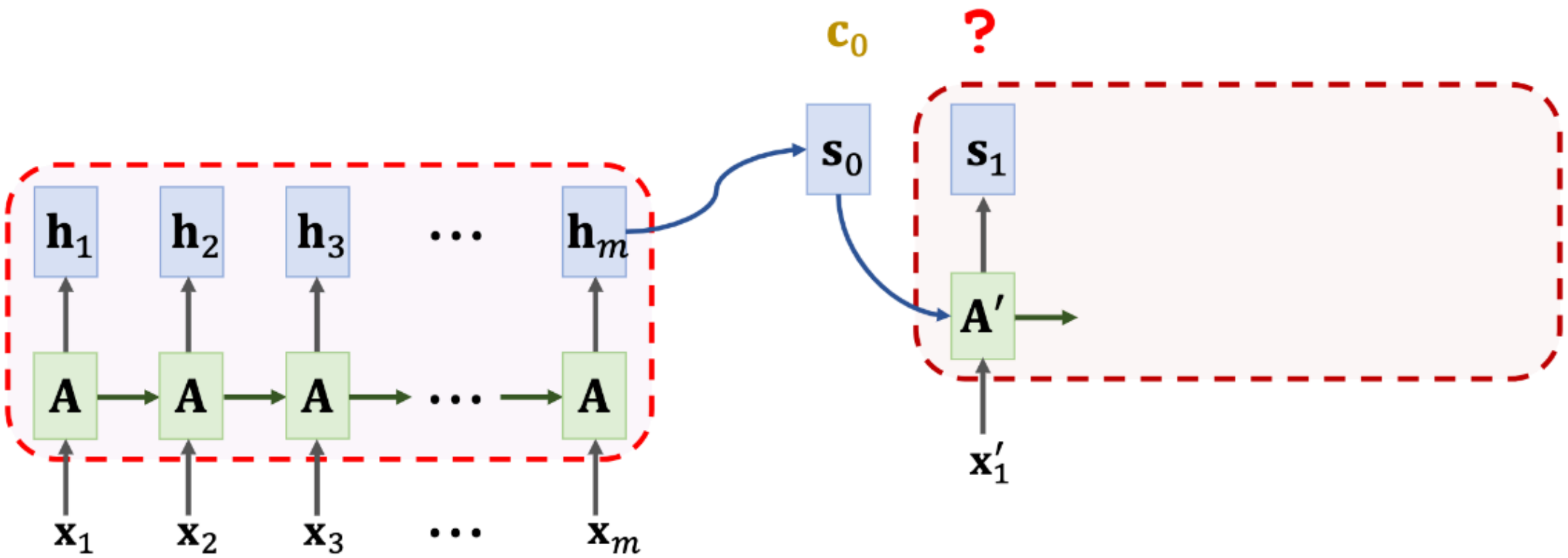
$$s_1 = \tanh\left(A' \cdot \begin{bmatrix} x'_1 \\ s_0 \end{bmatrix} + b\right)$$

SimpleRNN + Attention:

$$s_1 = \tanh\left(A' \cdot \begin{bmatrix} x'_1 \\ s_0 \\ c_0 \end{bmatrix} + b\right)$$

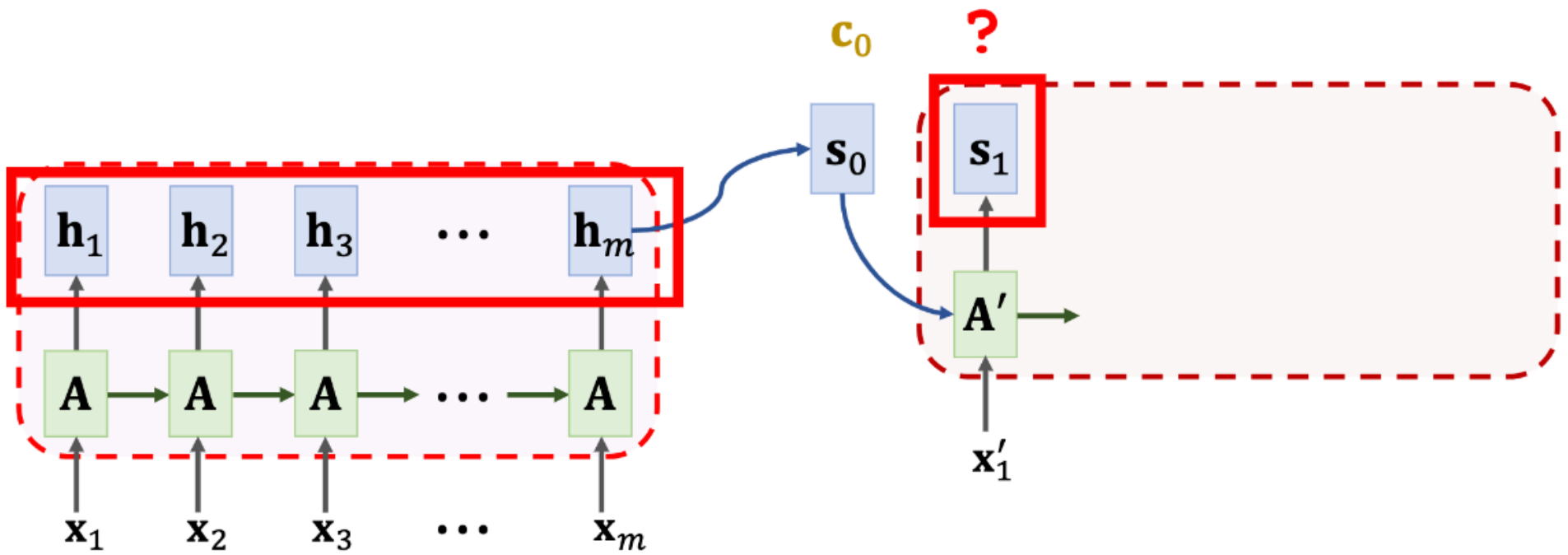


SimpleRNN + Attention



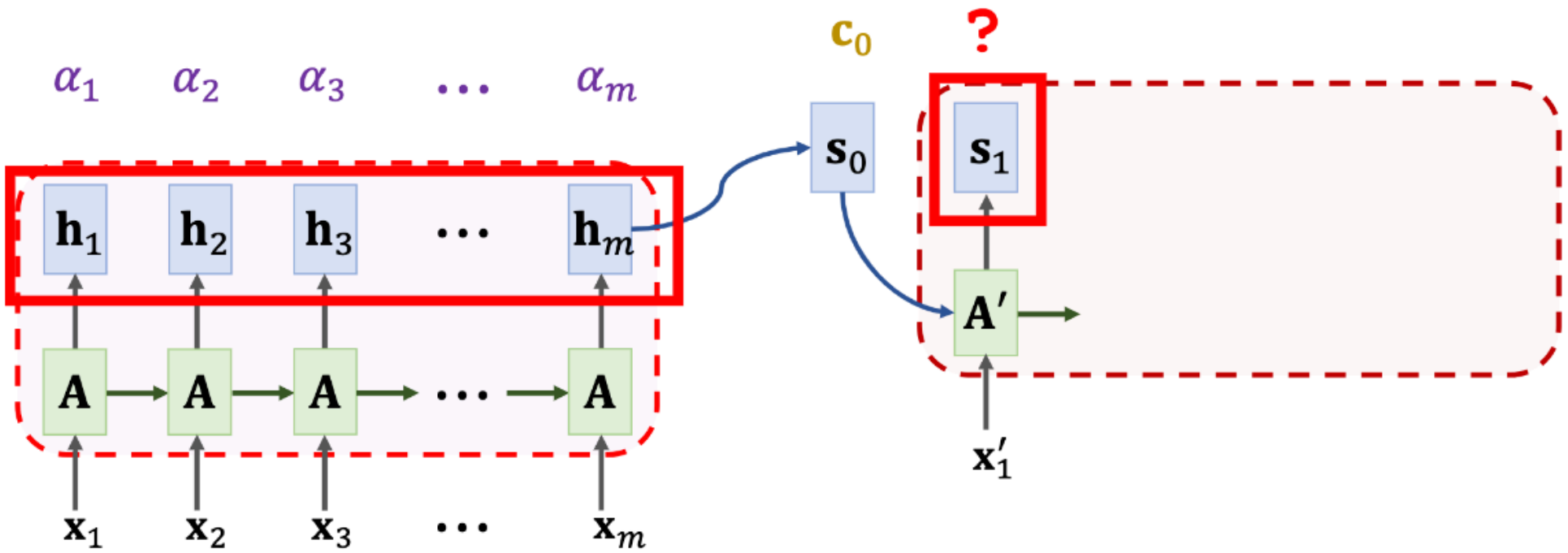
SimpleRNN + Attention

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



SimpleRNN + Attention

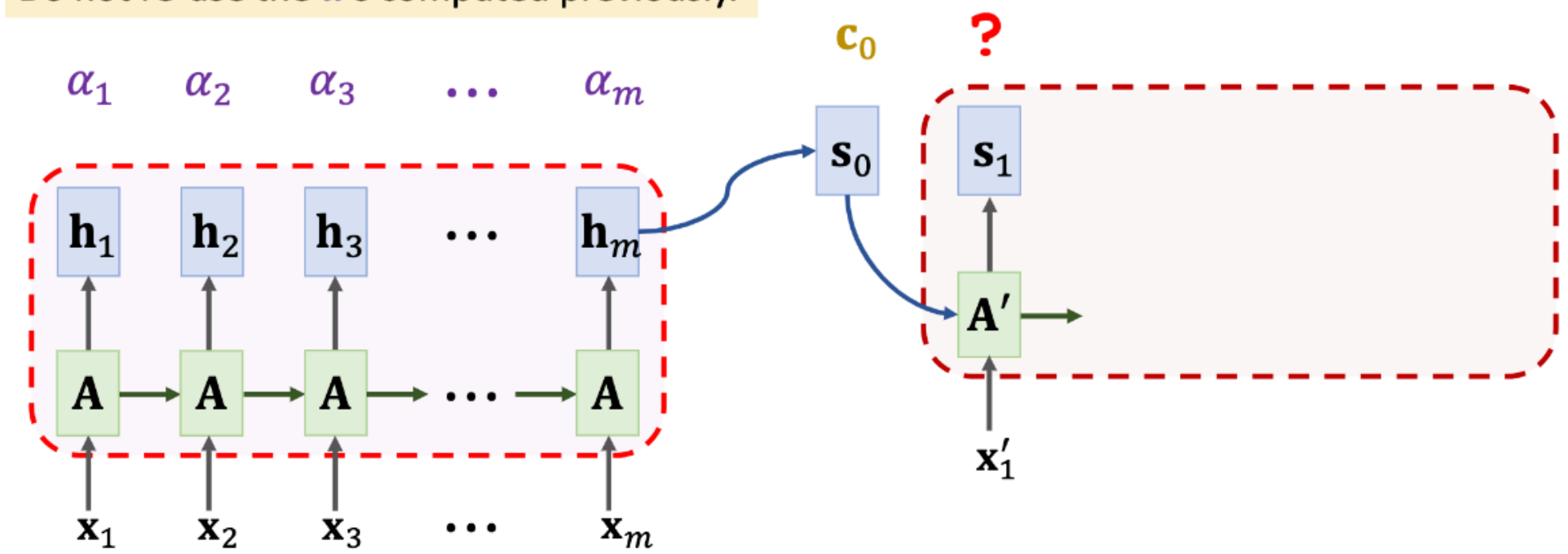
Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_1)$.



SimpleRNN + Attention

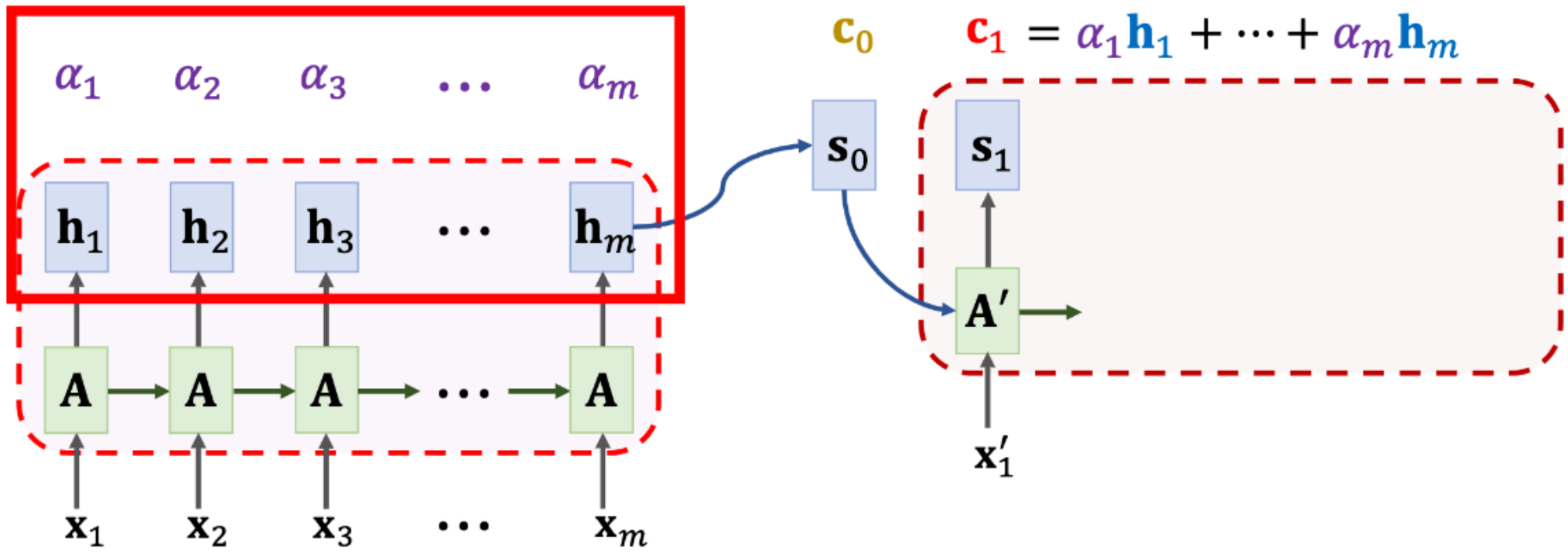
Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_1)$.

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



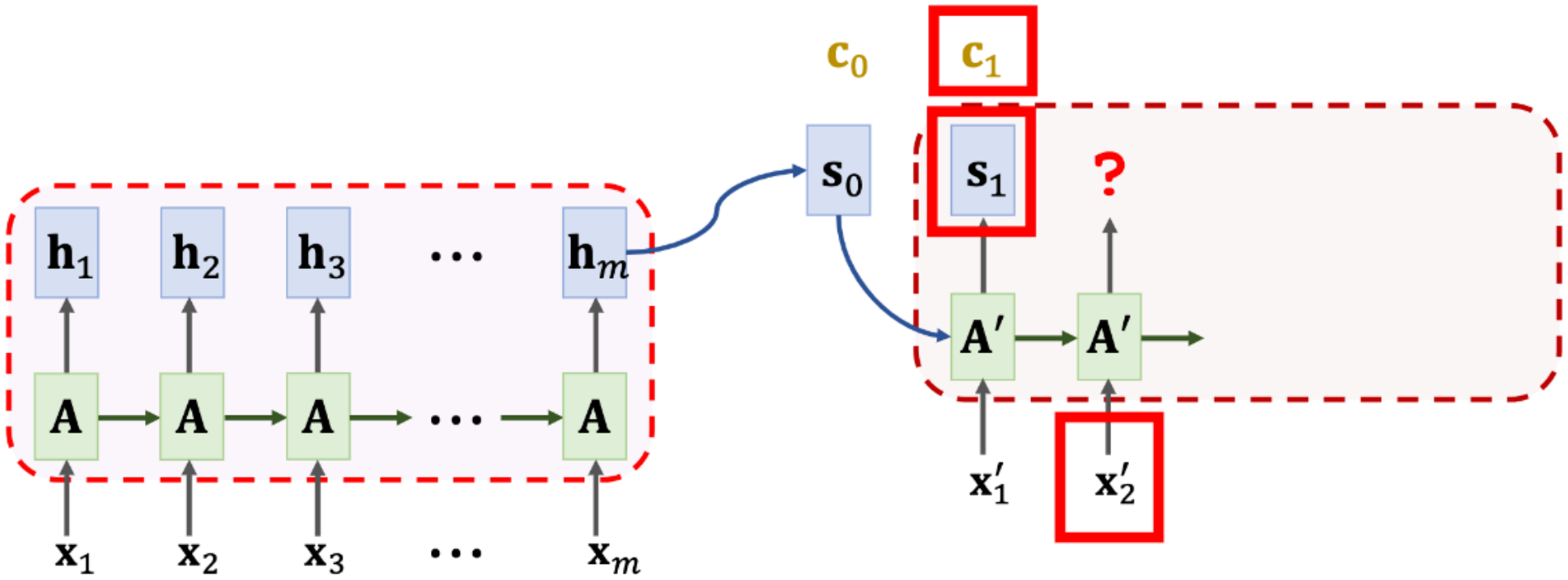
SimpleRNN + Attention

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

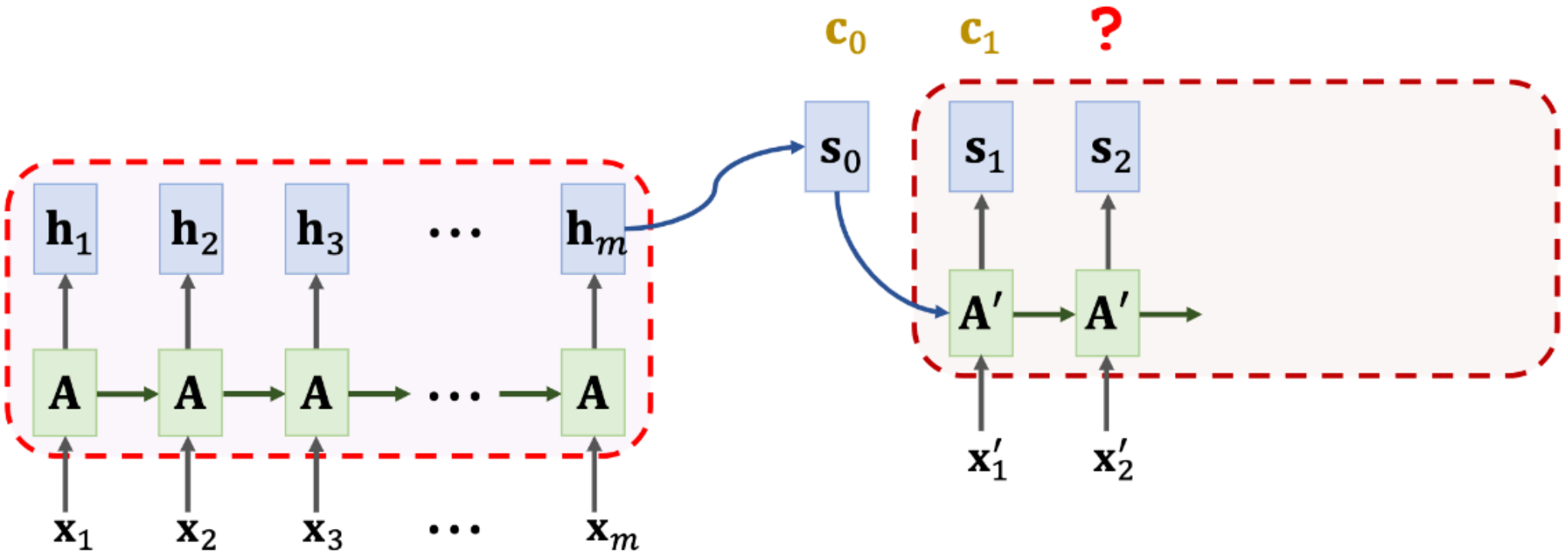


SimpleRNN + Attention

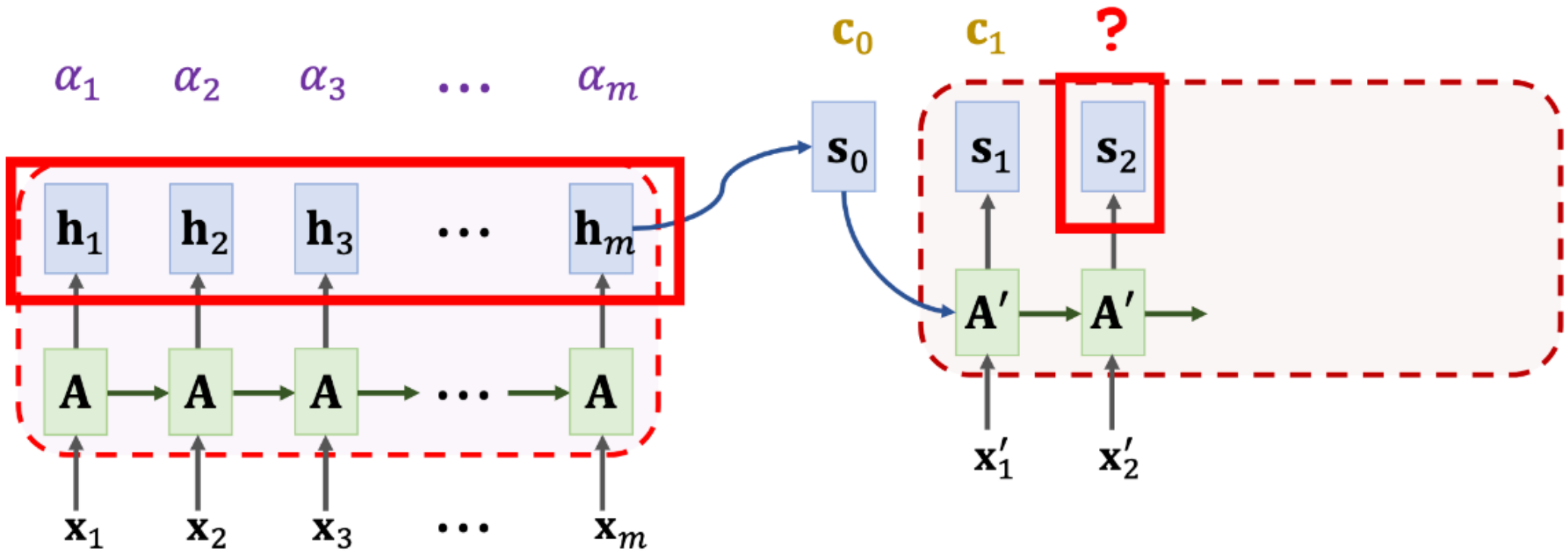
$$\mathbf{s}_2 = \tanh\left(\mathbf{A}' \cdot \begin{bmatrix} \mathbf{x}'_2 \\ \mathbf{s}_1 \\ \mathbf{c}_1 \end{bmatrix} + \mathbf{b}\right)$$



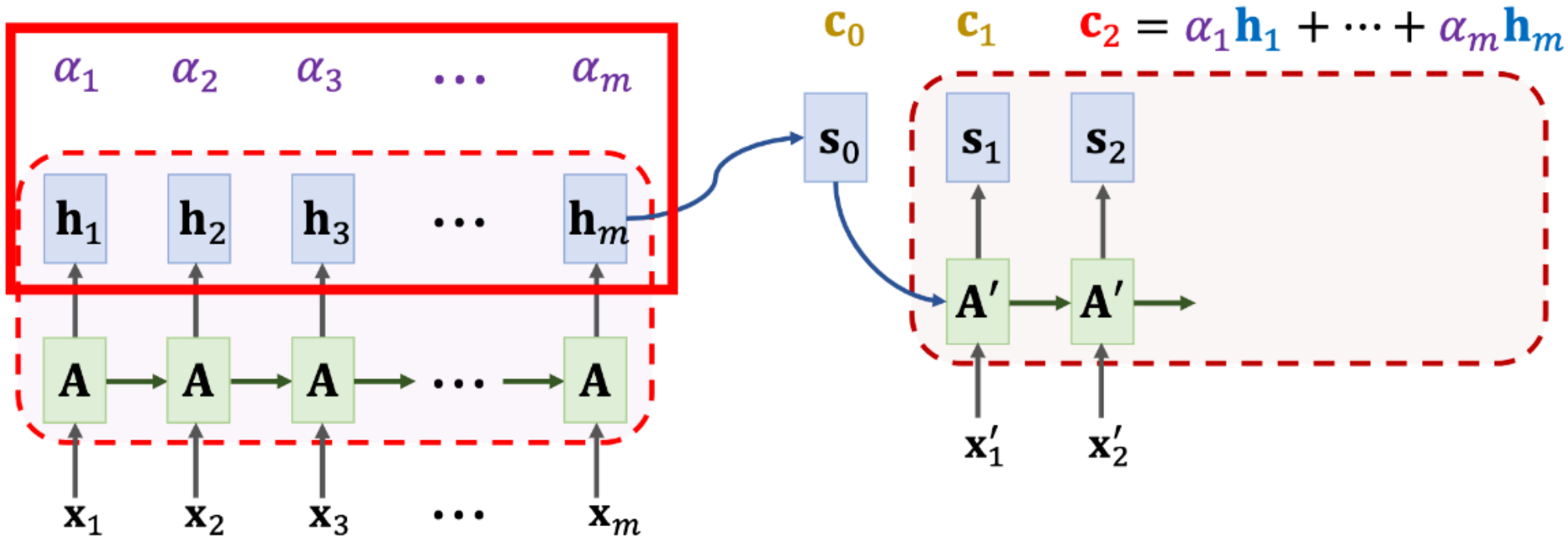
SimpleRNN + Attention



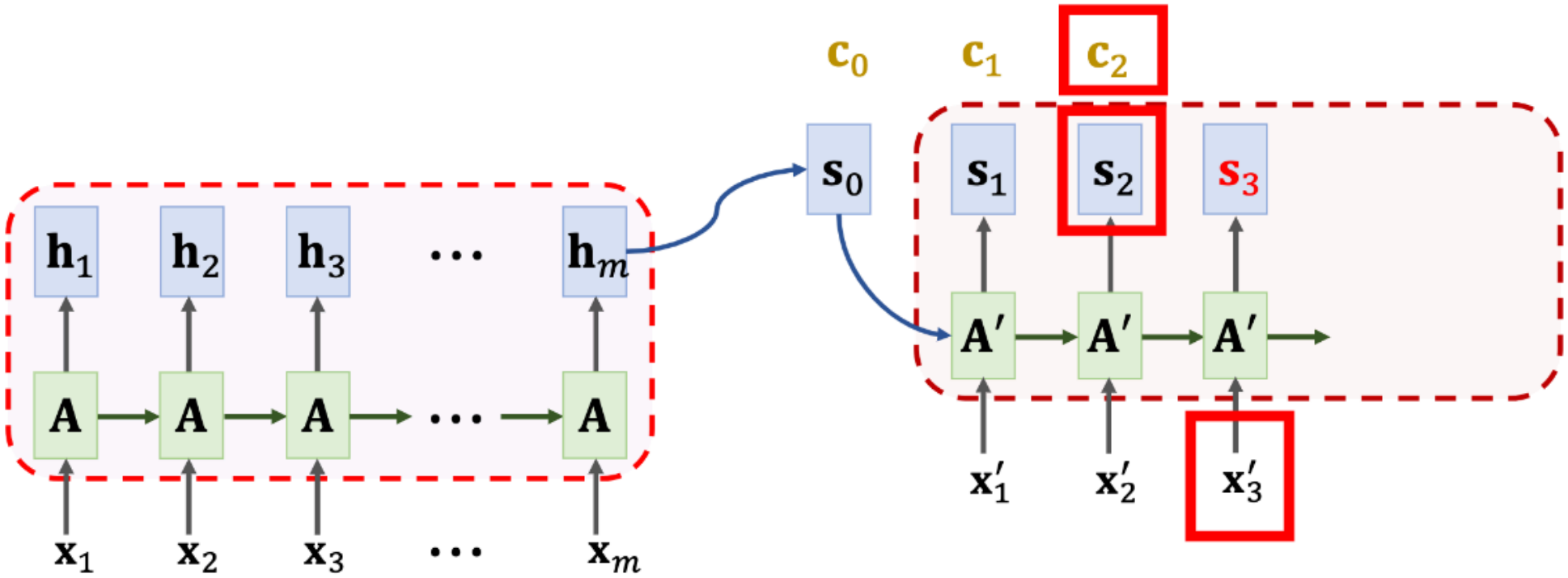
SimpleRNN + Attention



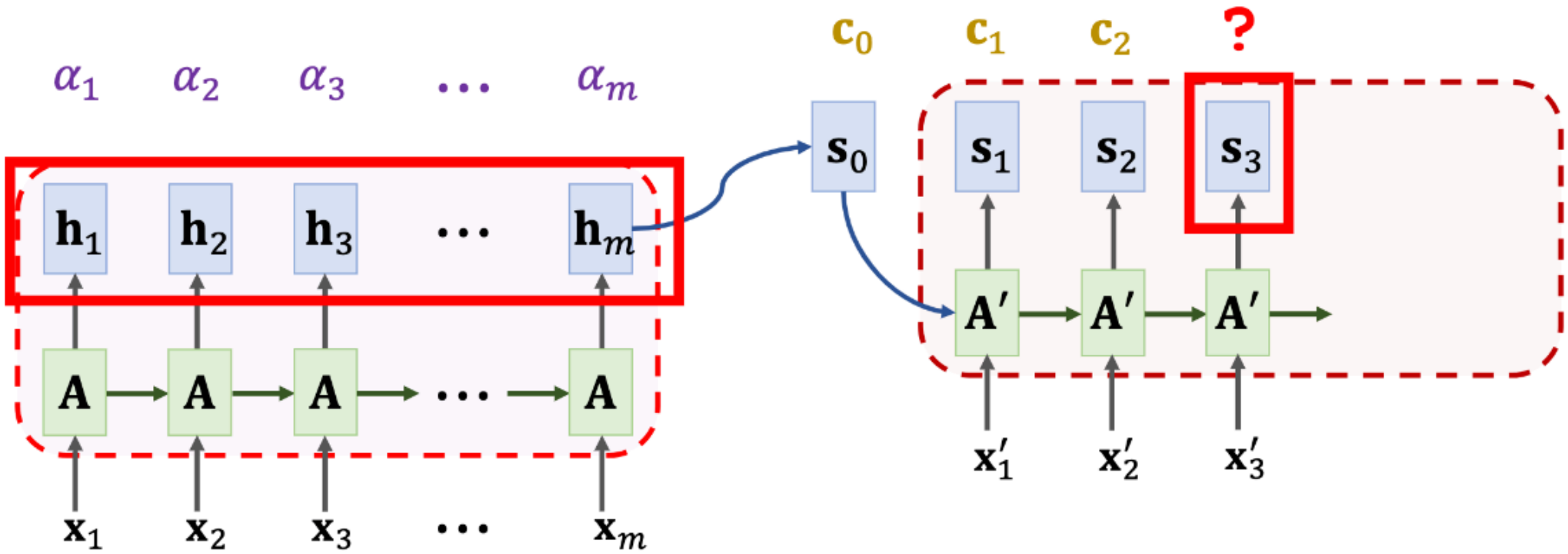
SimpleRNN + Attention



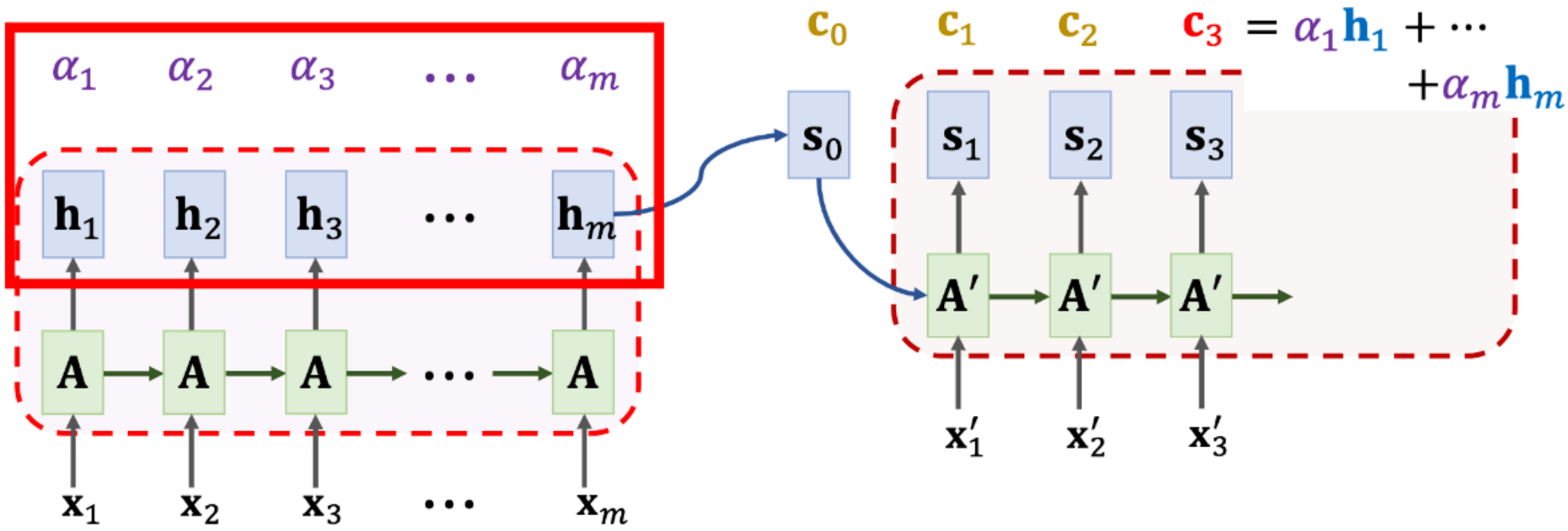
SimpleRNN + Attention



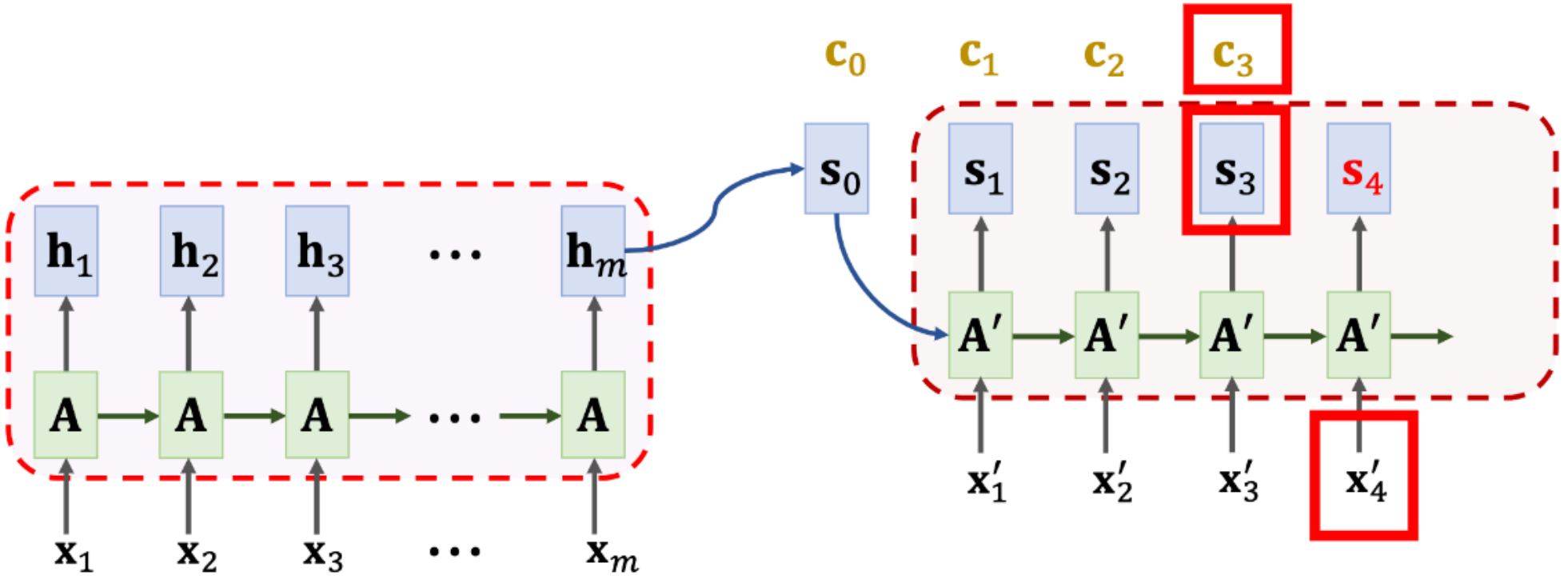
SimpleRNN + Attention



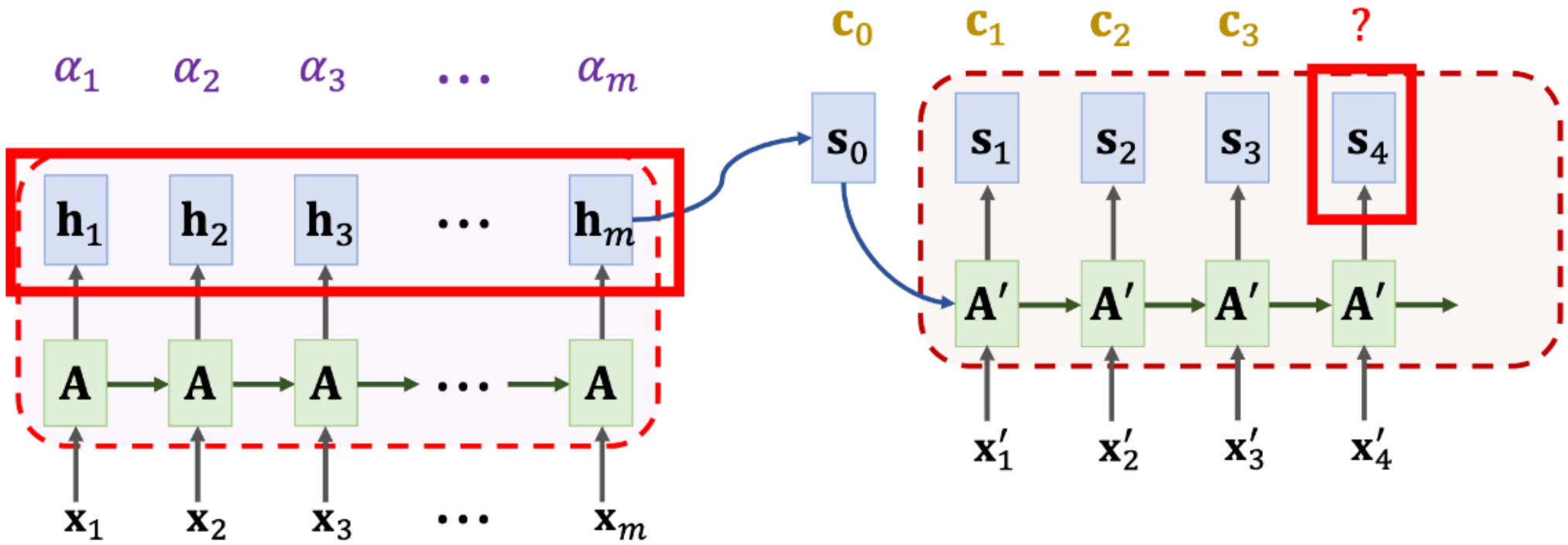
SimpleRNN + Attention



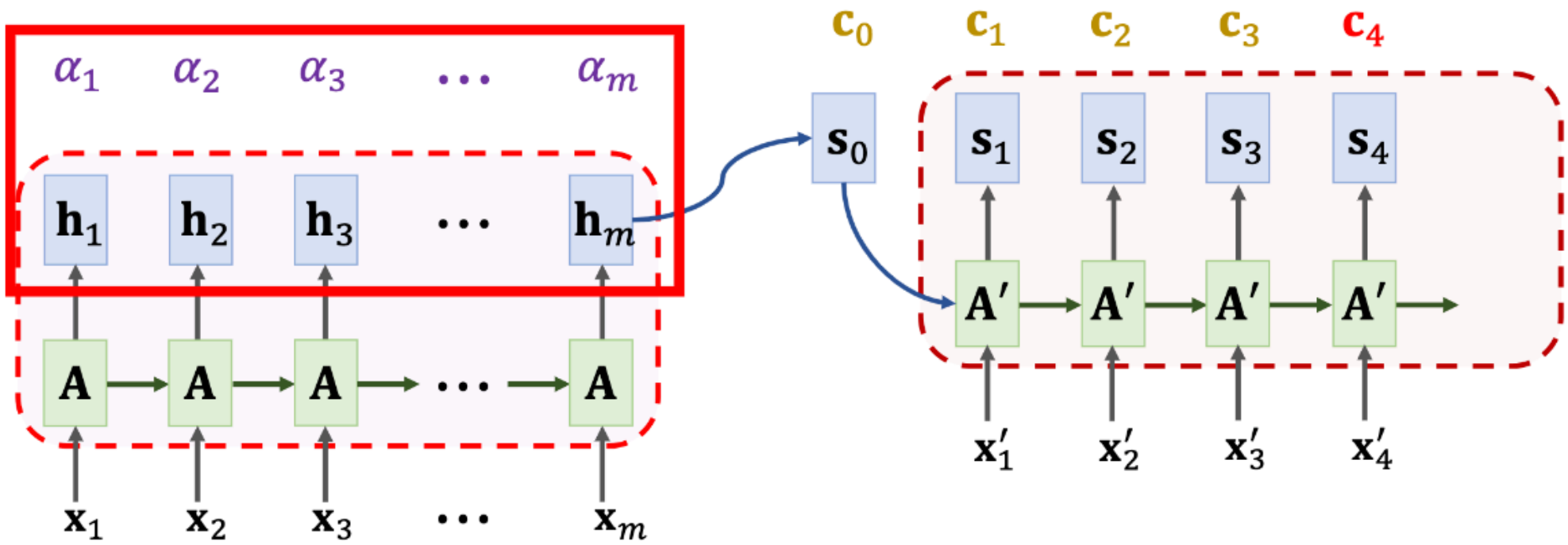
SimpleRNN + Attention



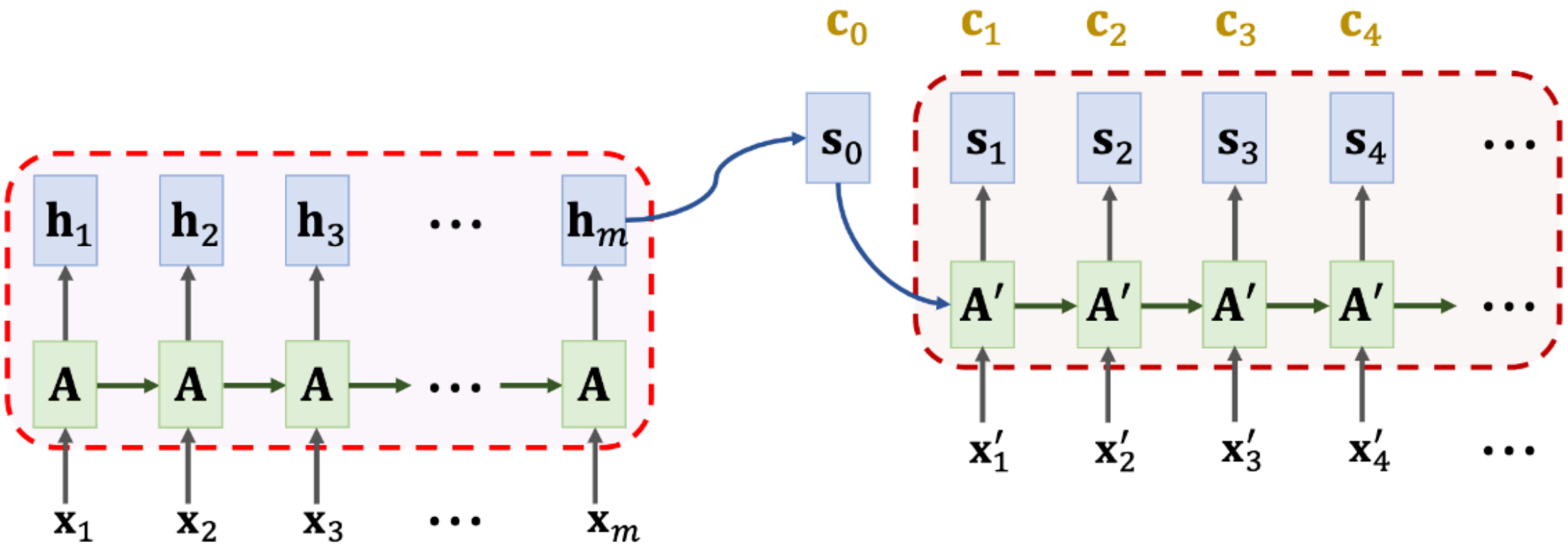
SimpleRNN + Attention



SimpleRNN + Attention

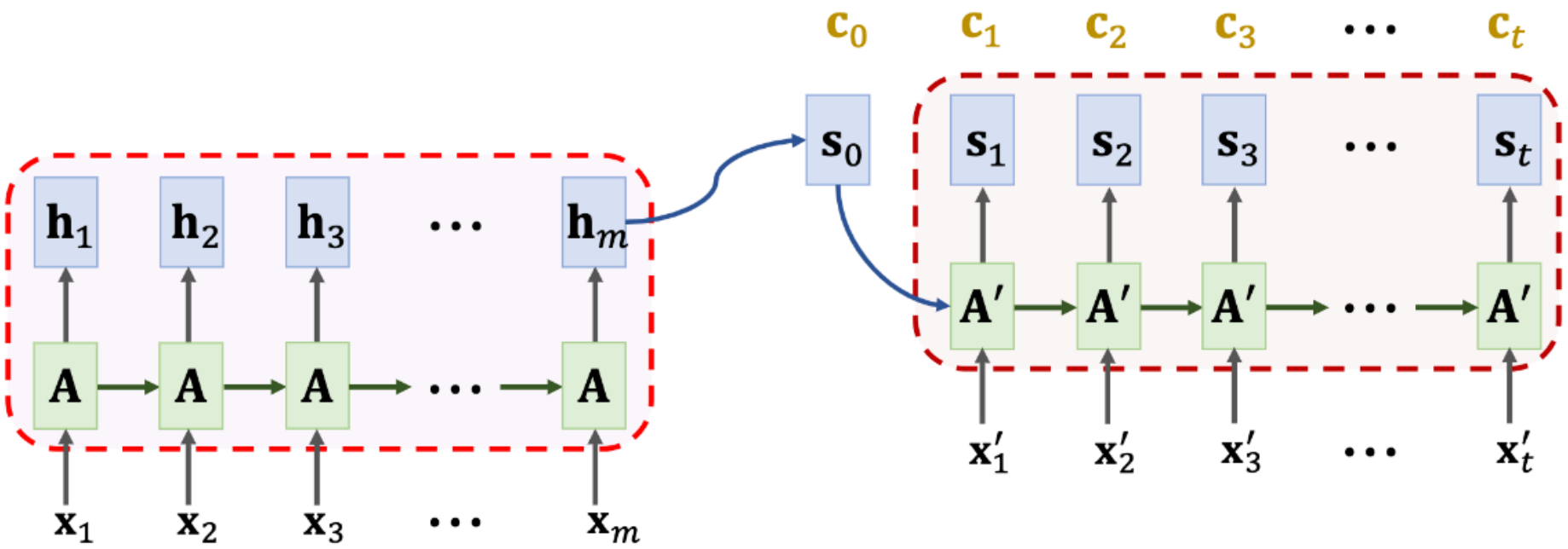


SimpleRNN + Attention



Time Complexity

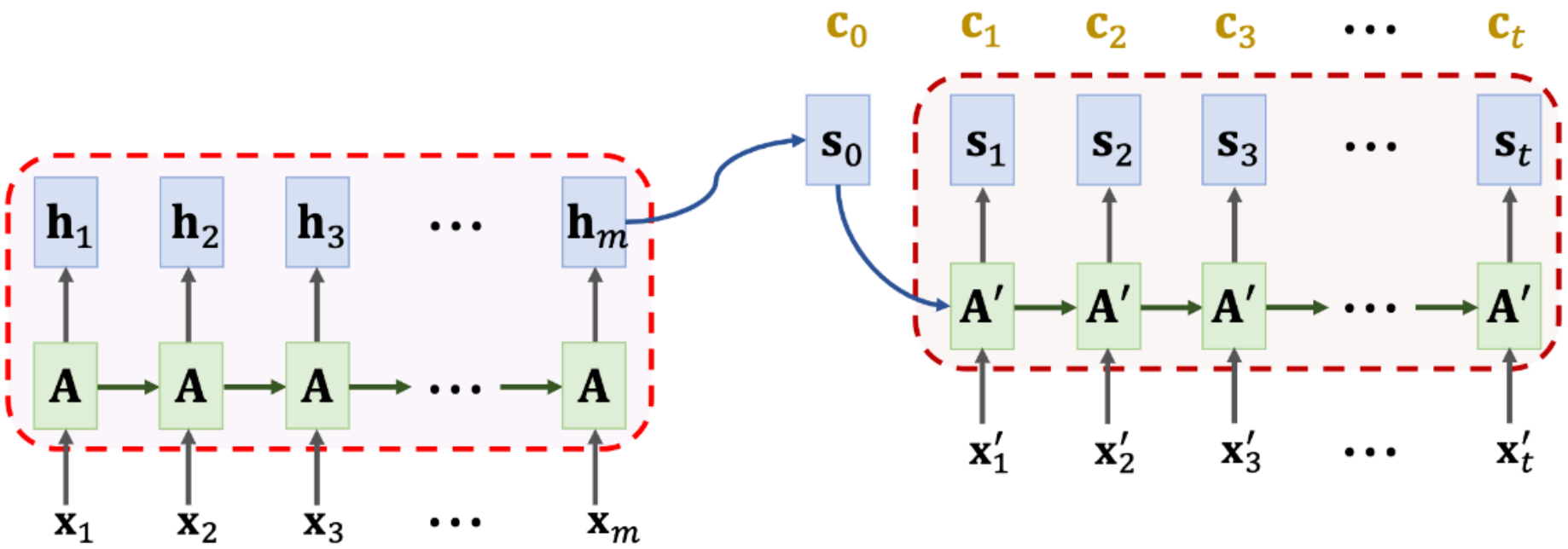
Question: How many weights α_{ij} have been computed?



Time Complexity

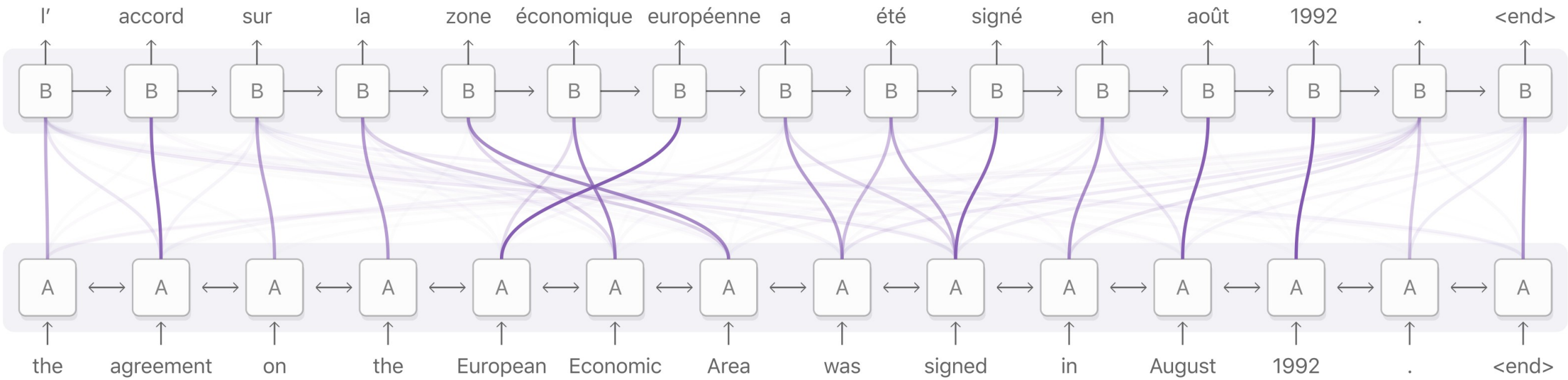
Question: How many weights α_i have been computed?

- To compute one vector **cj** we compute m weights: $\alpha_1, \cdots, \alpha_m$.
- The decoder has t states, so there are totally mt weights.



Attention: Weights Visualization

Decoder RNN (target language: French)

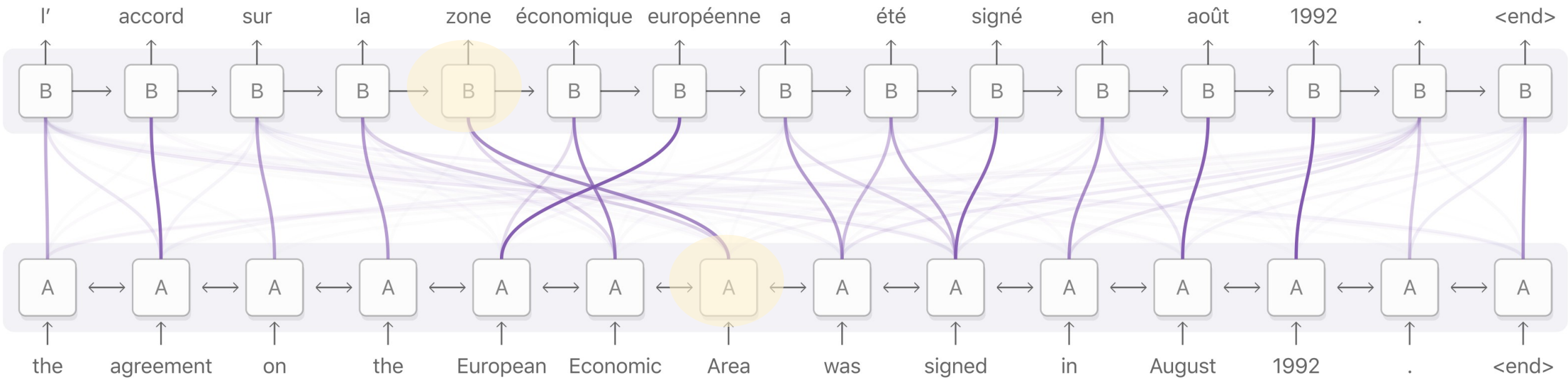


Encoder RNN (source language: English)

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.

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.

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

Self-Attention

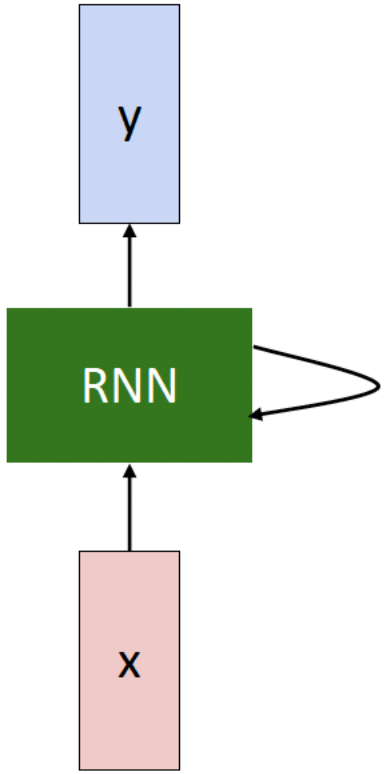
Self-Attention

- Attention [1] is designed for Seq2Seq which includes 2 RNNs: RNN Encoder + RNN Decoder.
- Self-Attention [2]: improvement over a single RNN.
- Key idea: calculate state vector using context vector.

Original paper:

1. Bahdanau, Cho, & Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.
2. Cheng, Dong, & Lapata. Long Short-Term Memory-Networks for Machine Reading. In *EMNLP*, 2016.

SimpleRNN



We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

new state

some function with parameters W

old state

input vector at some time step

SimpleRNN + Self-Attention

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:

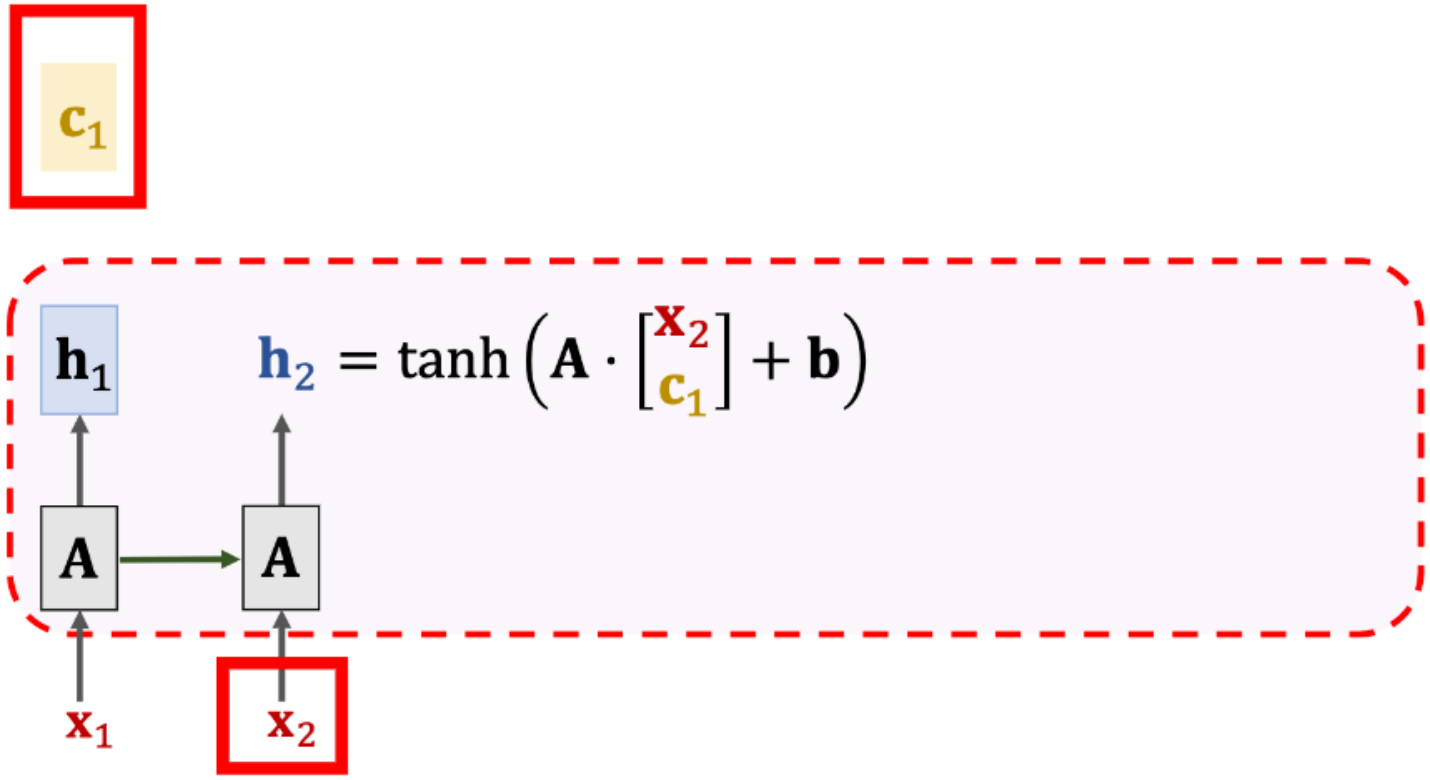
$$\mathbf{h}_1 = \tanh\left(\mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{c}_0 \end{bmatrix} + \mathbf{b}\right)$$

SimpleRNN + Self-Attention

SimpleRNN + Self-Attention

SimpleRNN + Self-Attention

SimpleRNN + Self-Attention



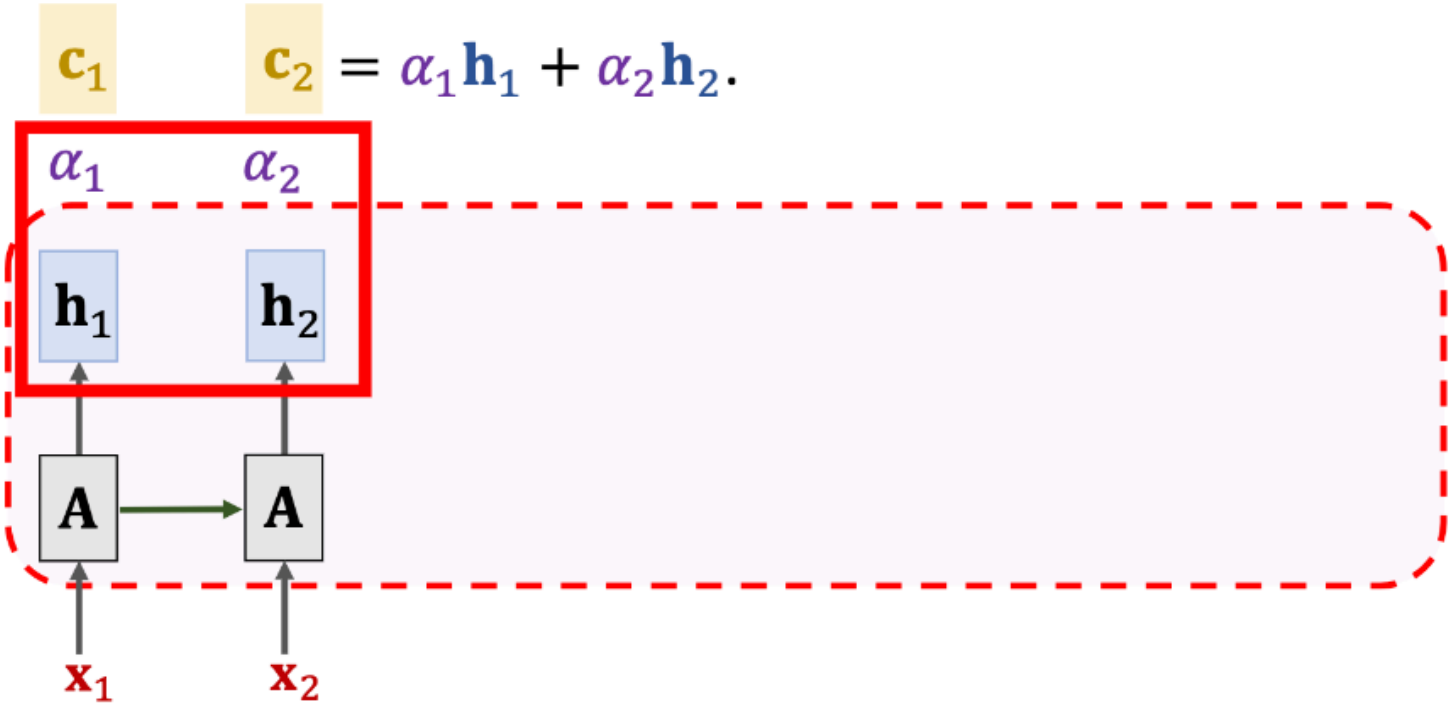
SimpleRNN + Self-Attention

SimpleRNN + Self-Attention

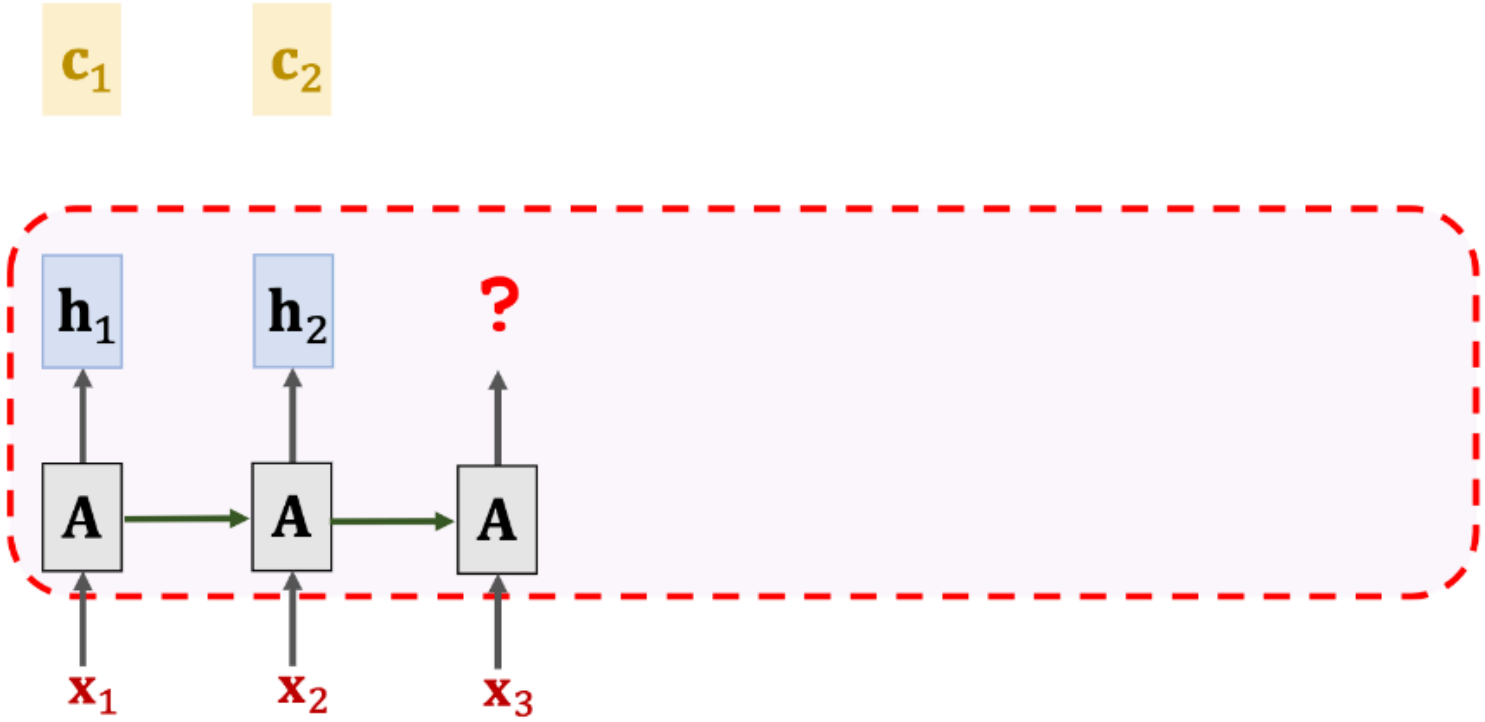
Weights: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{h}_2)$.

SimpleRNN + Self-Attention

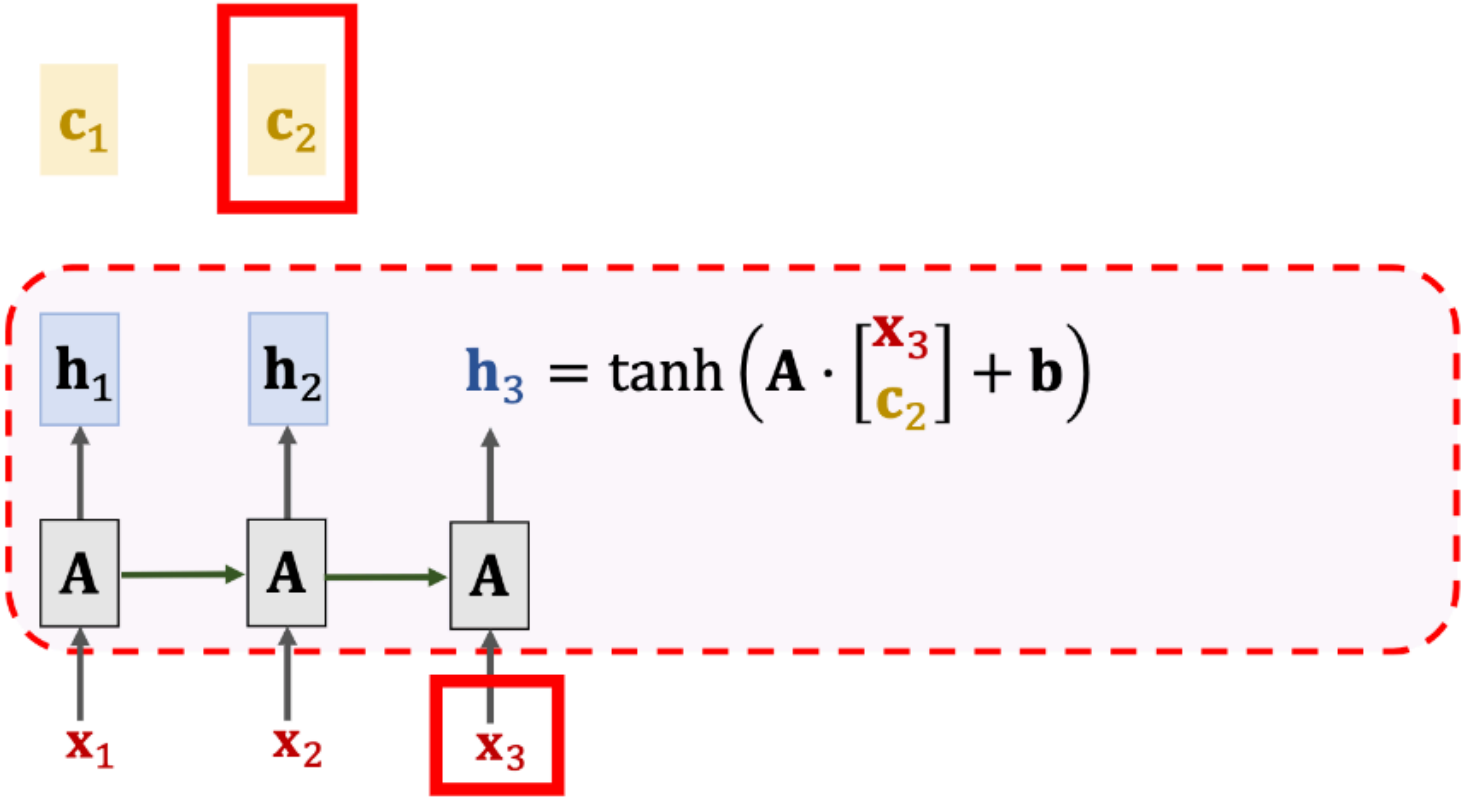
SimpleRNN + Self-Attention



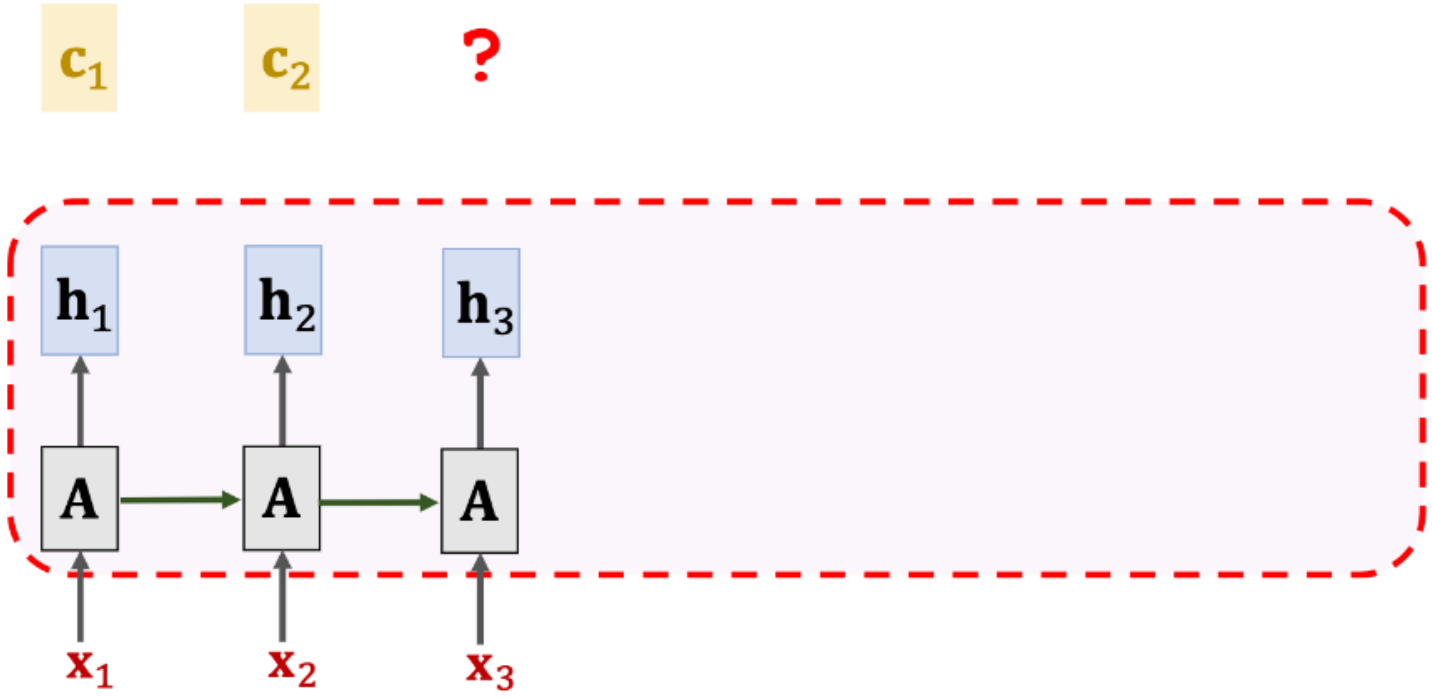
SimpleRNN + Self-Attention



SimpleRNN + Self-Attention

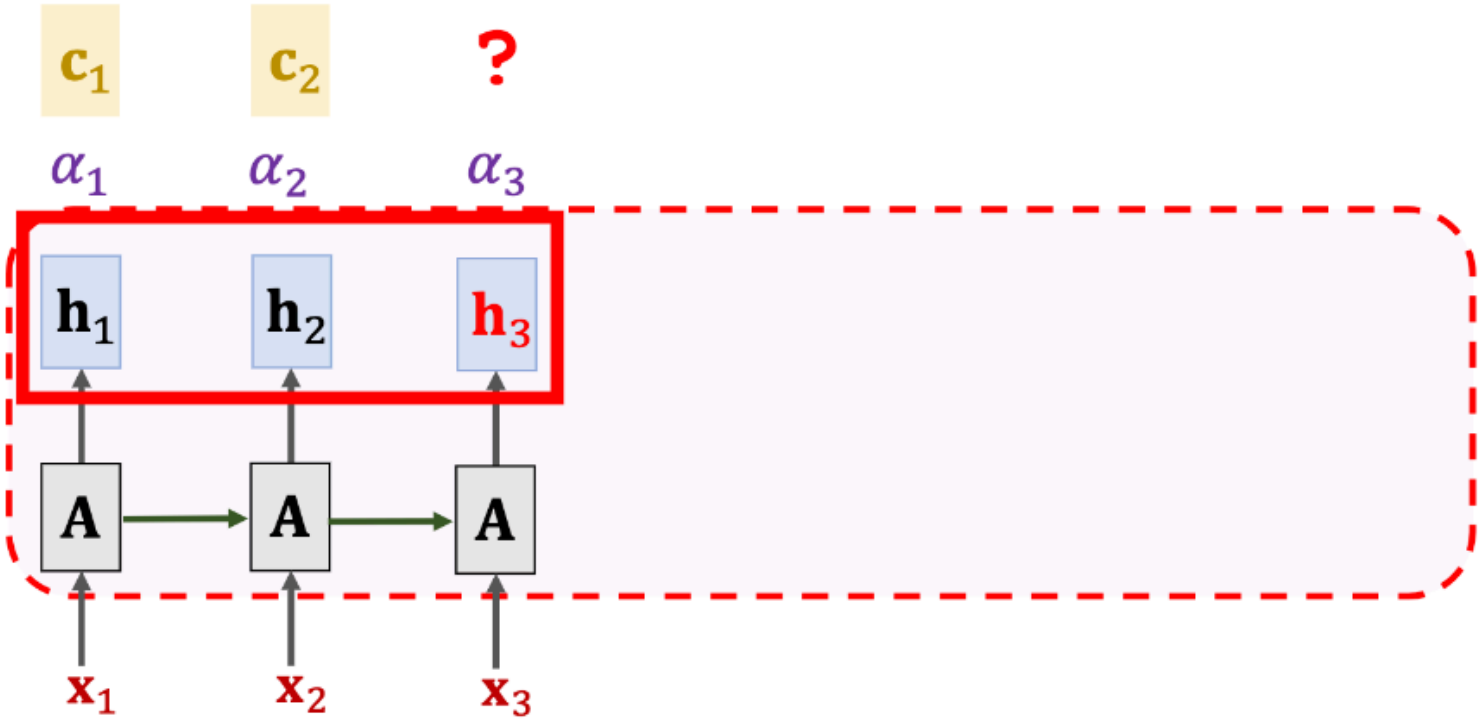


SimpleRNN + Self-Attention

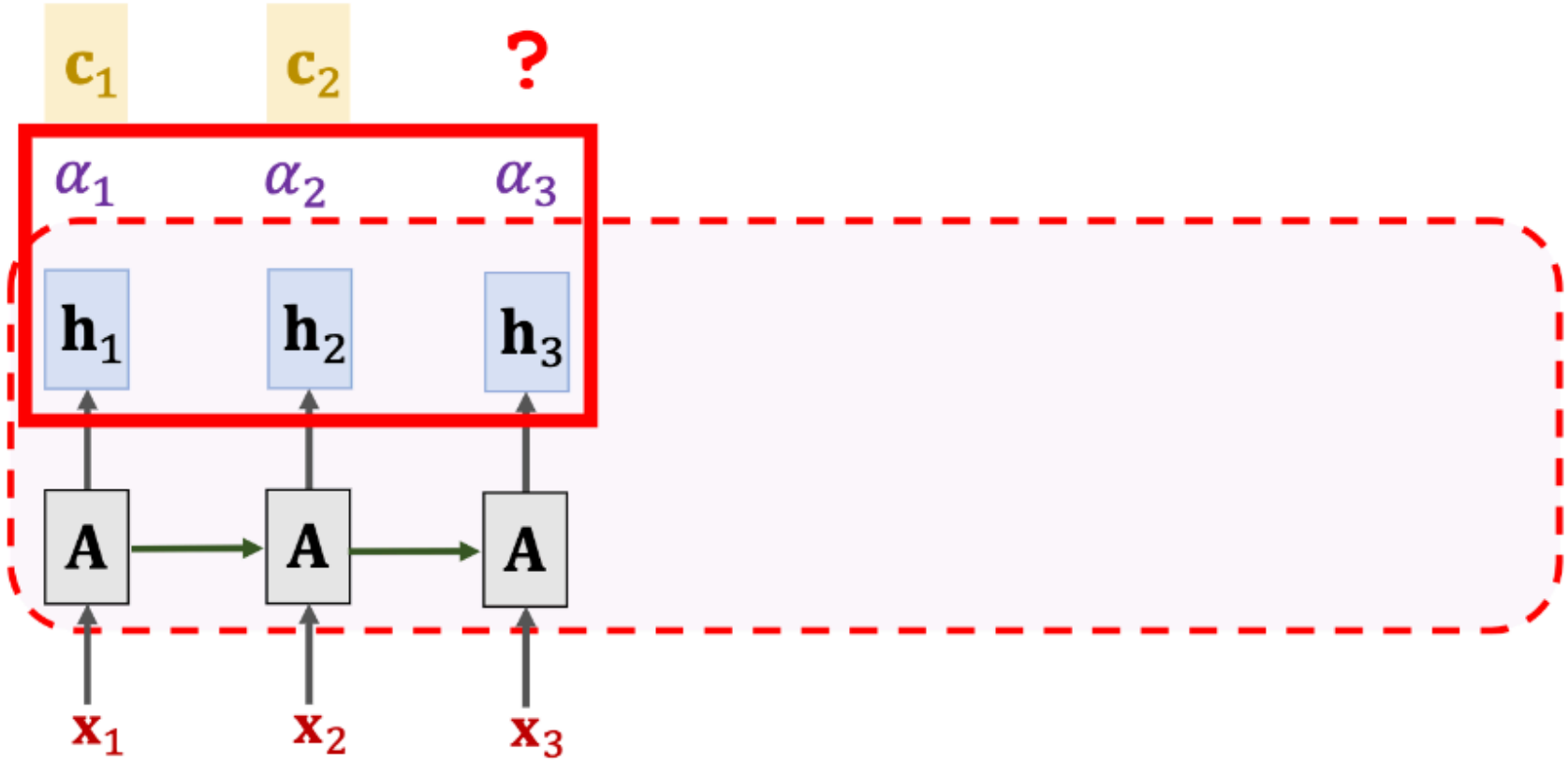


SimpleRNN + Self-Attention

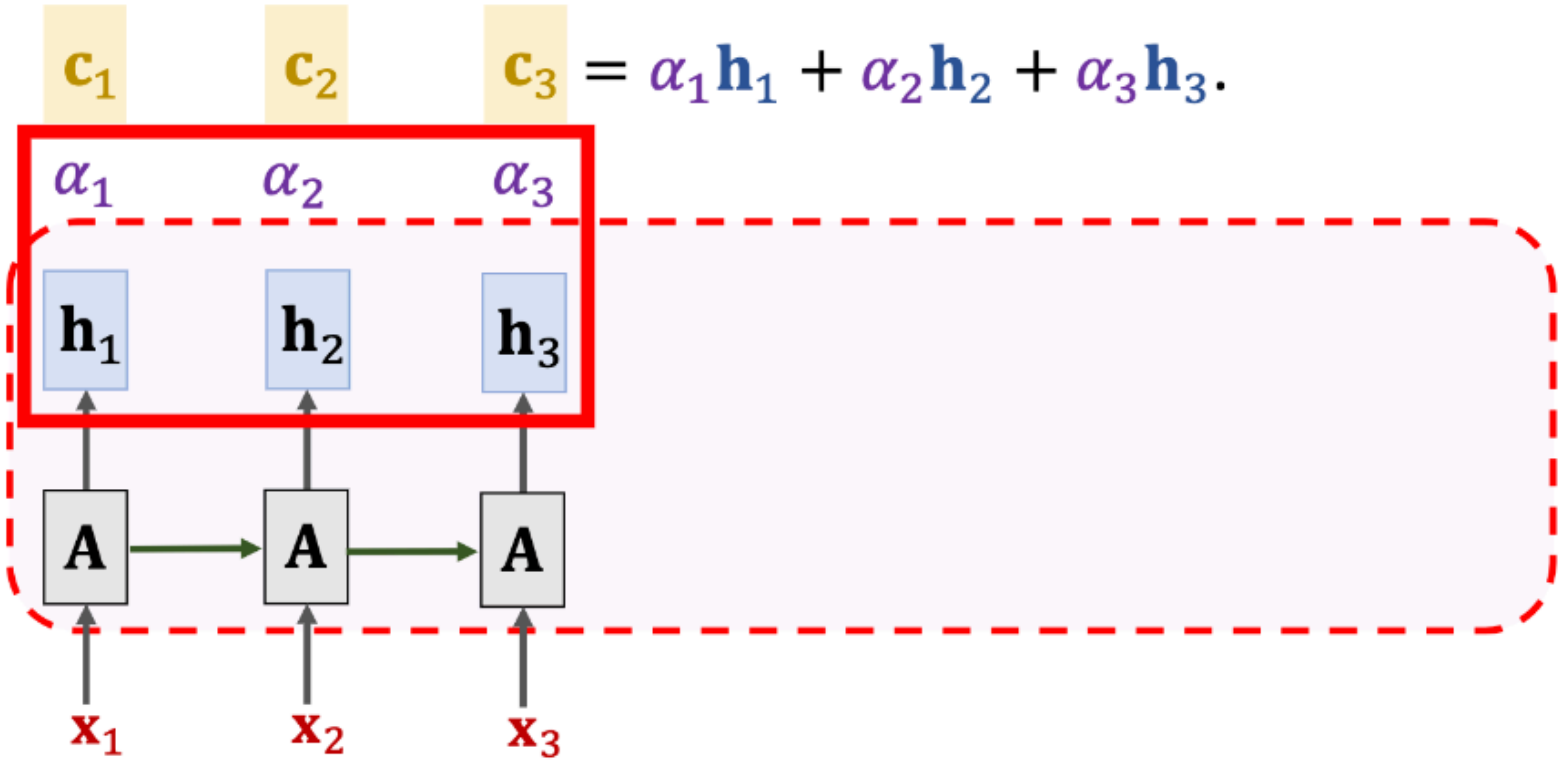
Weights: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{h}_3)$.



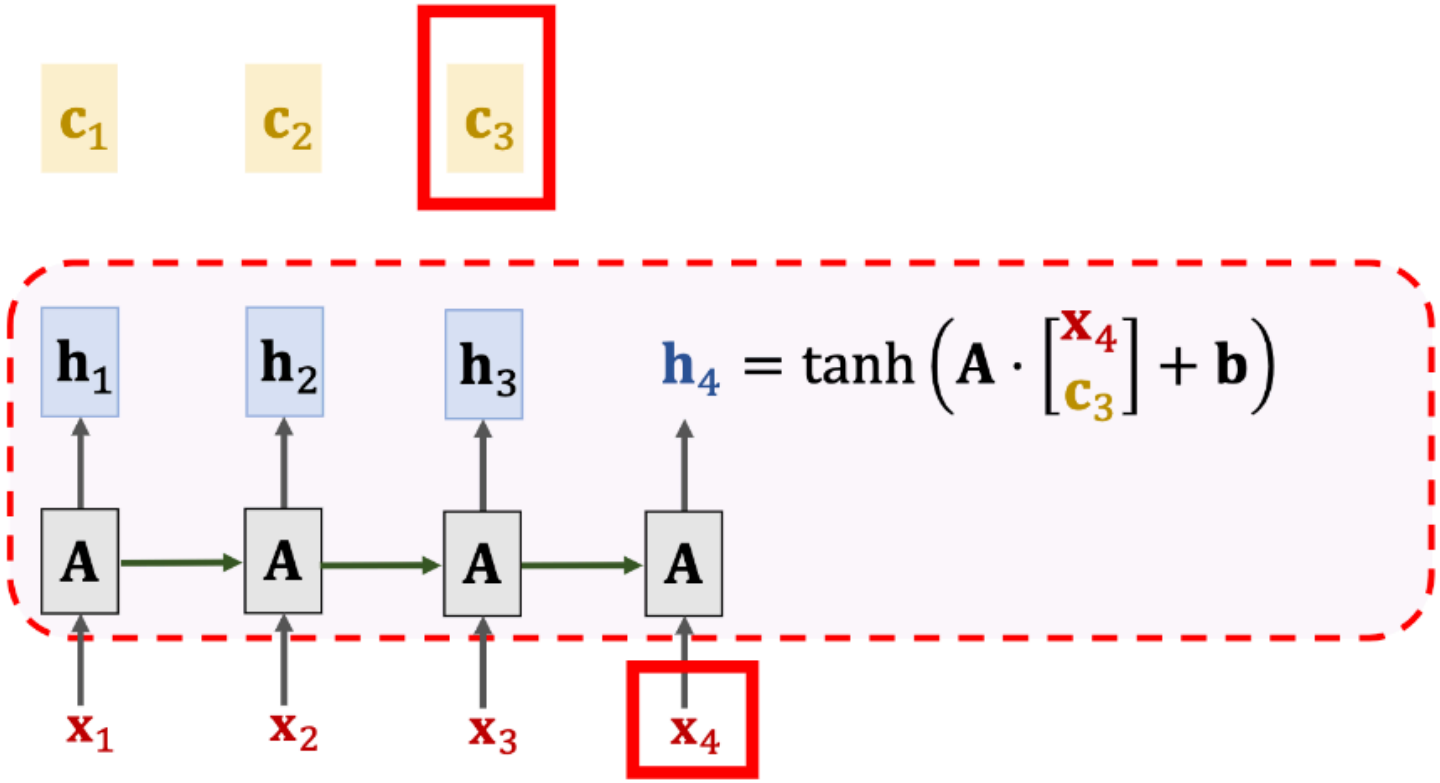
SimpleRNN + Self-Attention



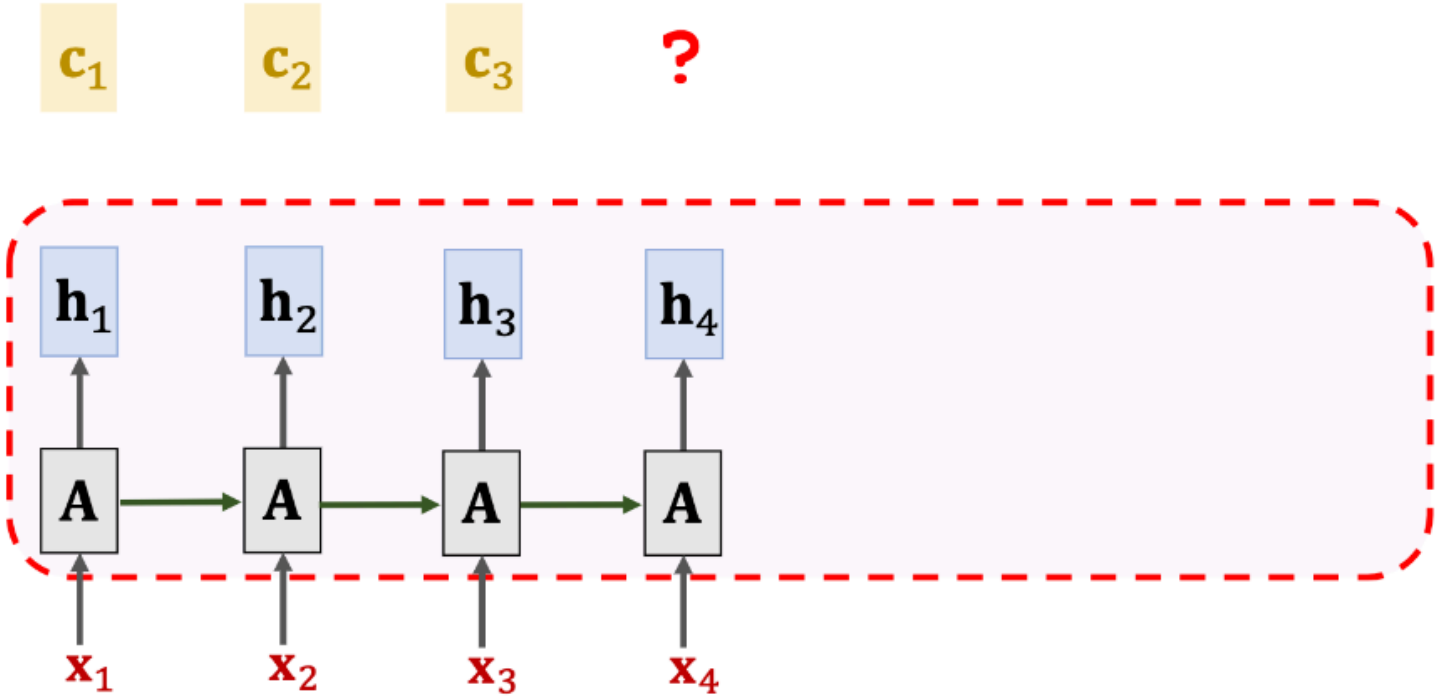
SimpleRNN + Self-Attention



SimpleRNN + Self-Attention

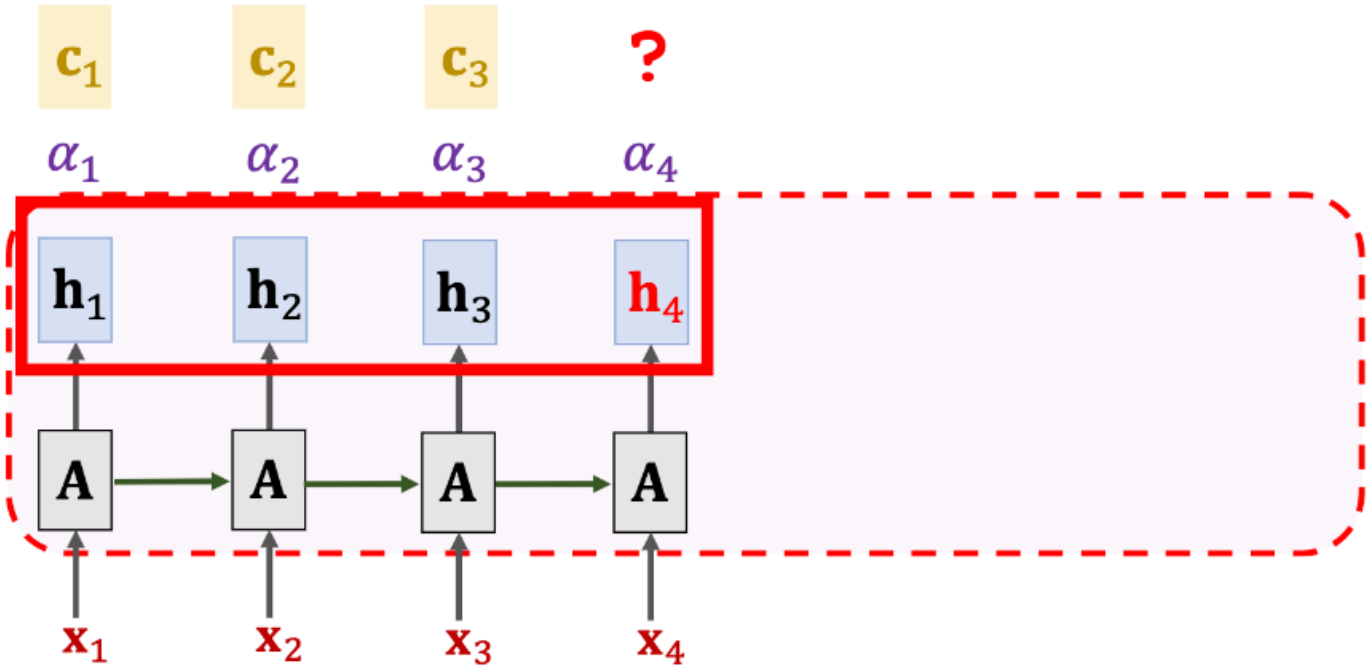


SimpleRNN + Self-Attention

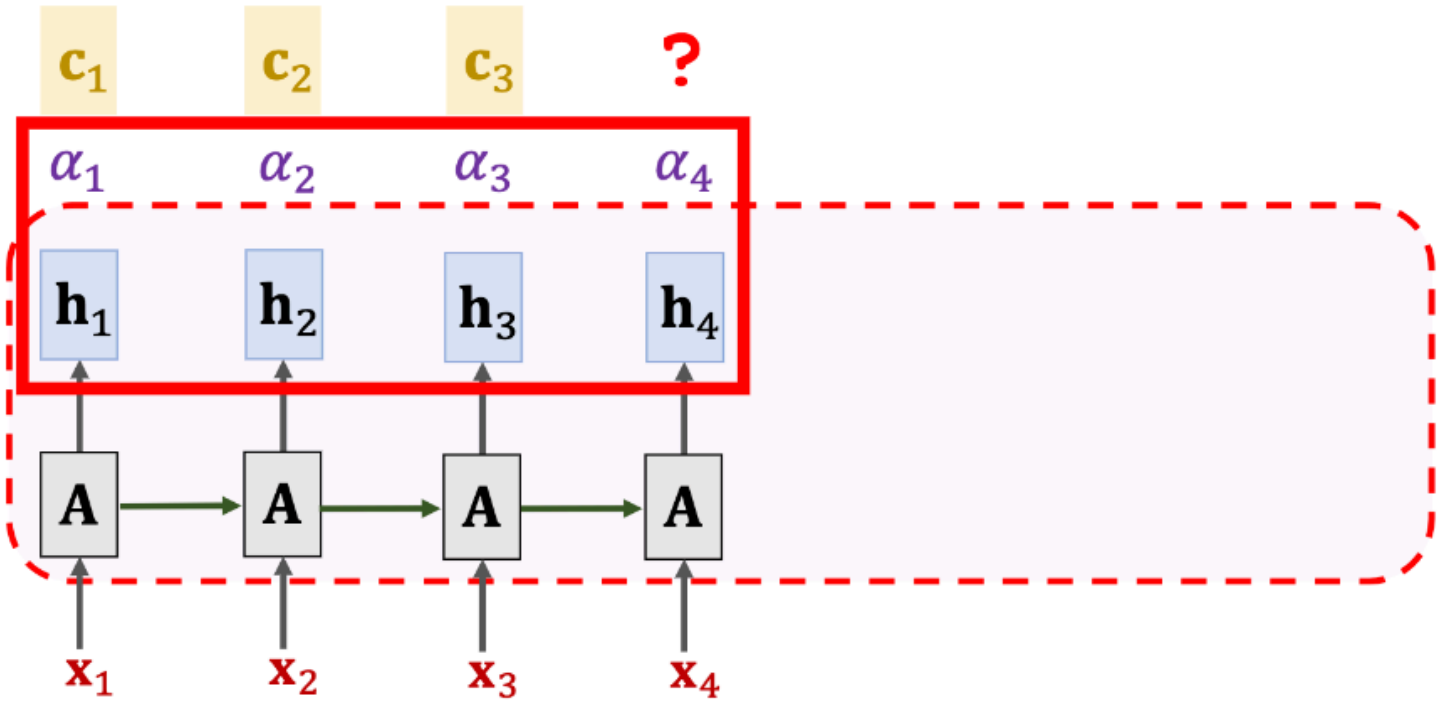


SimpleRNN + Self-Attention

Weights: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{h}_4)$.

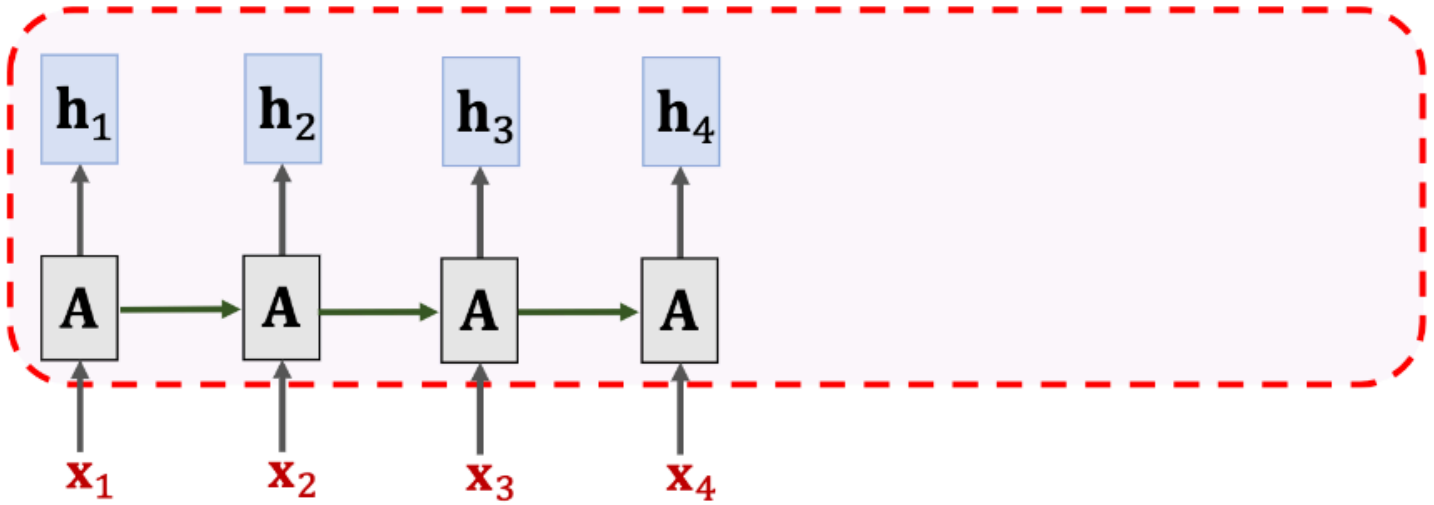


SimpleRNN + Self-Attention

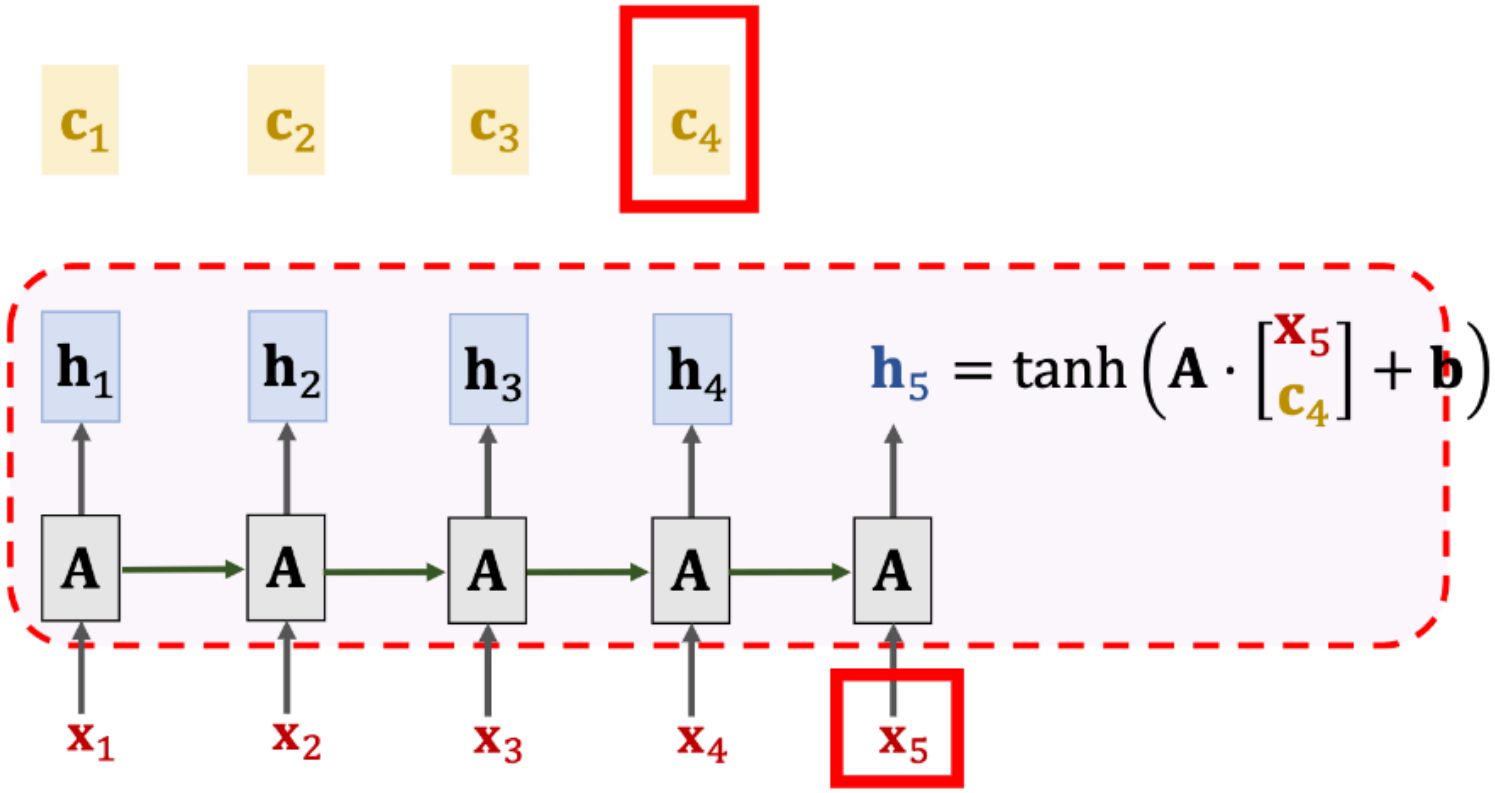


SimpleRNN + Self-Attention

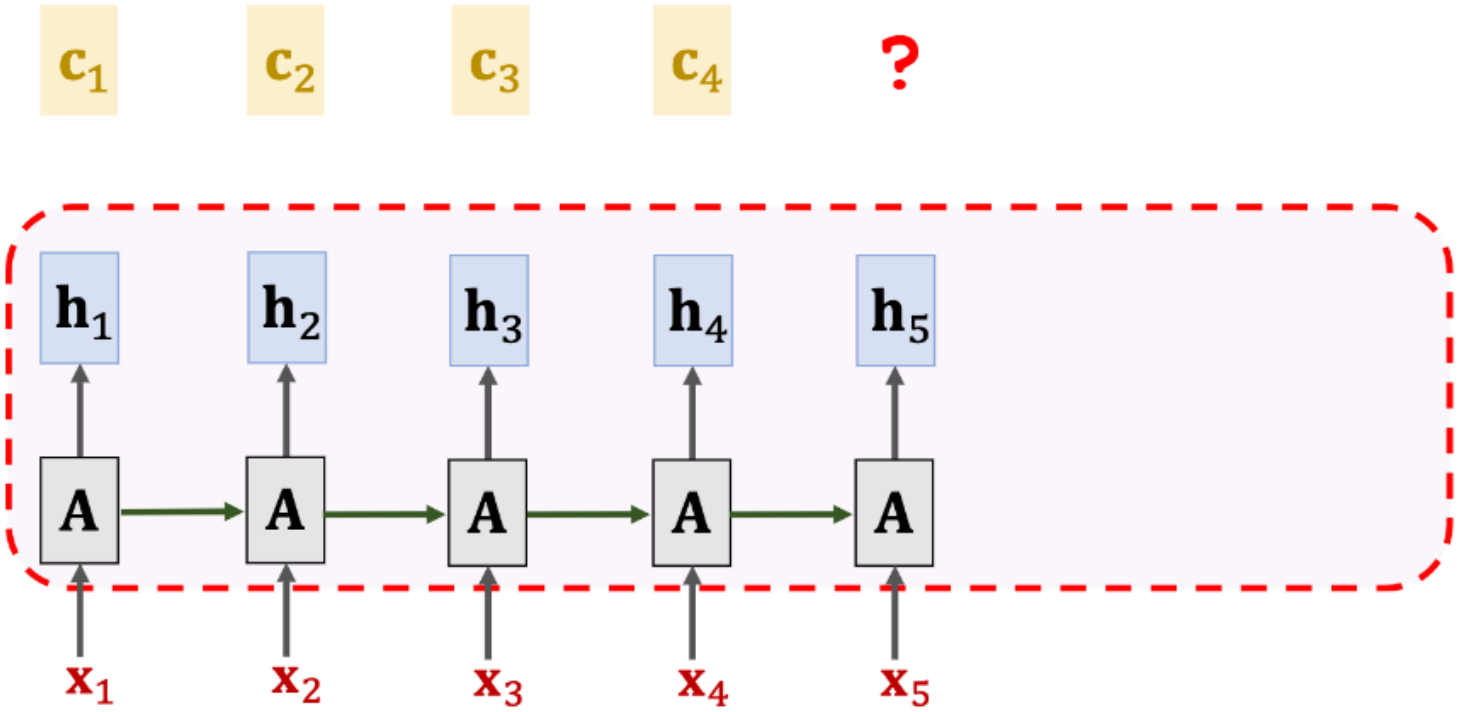
$\mathbf{c}_1$ $\mathbf{c}_2$ $\mathbf{c}_3$ $\mathbf{c}_4 = \alpha_1\mathbf{h}_1 + \alpha_2\mathbf{h}_2 + \alpha_3\mathbf{h}_3 + \alpha_4\mathbf{h}_4.$



SimpleRNN + Self-Attention

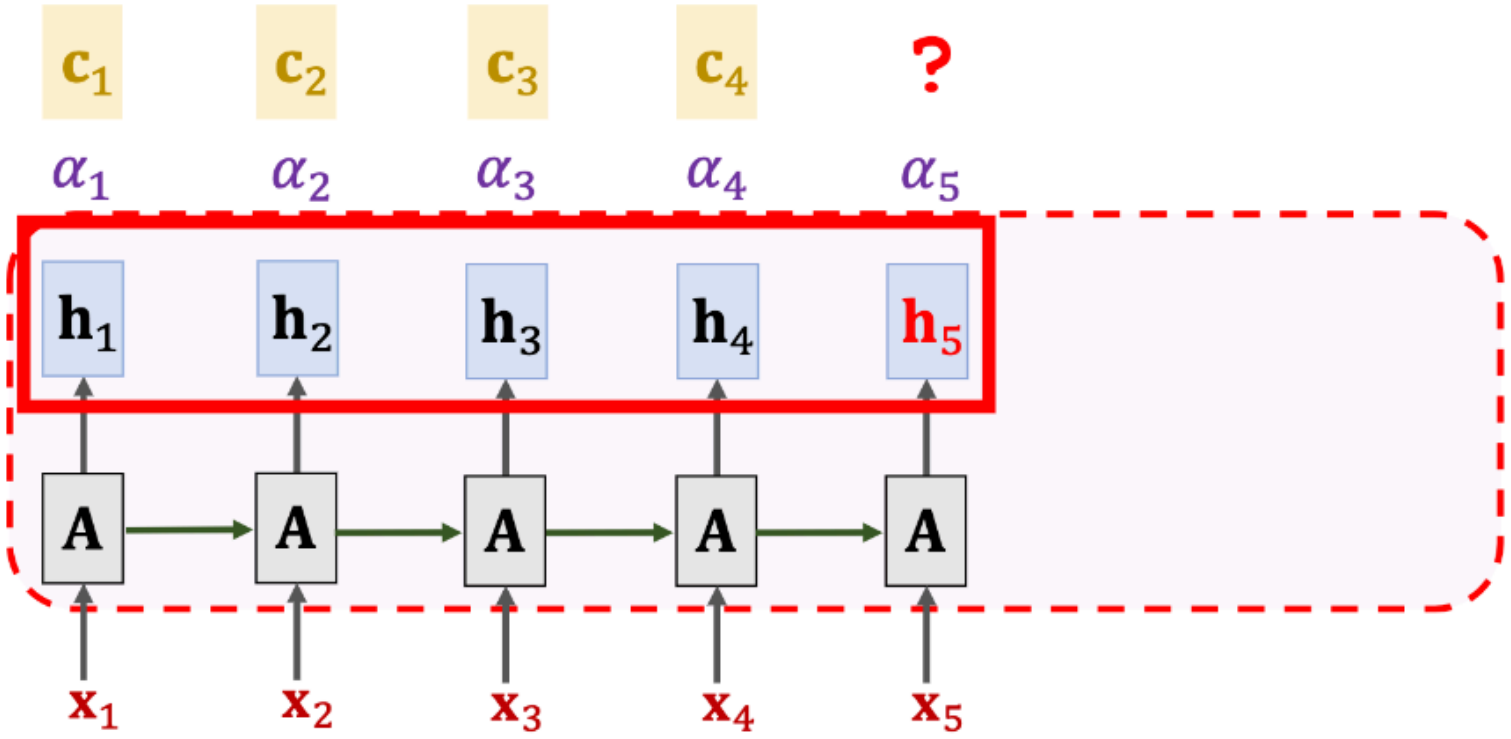


SimpleRNN + Self-Attention

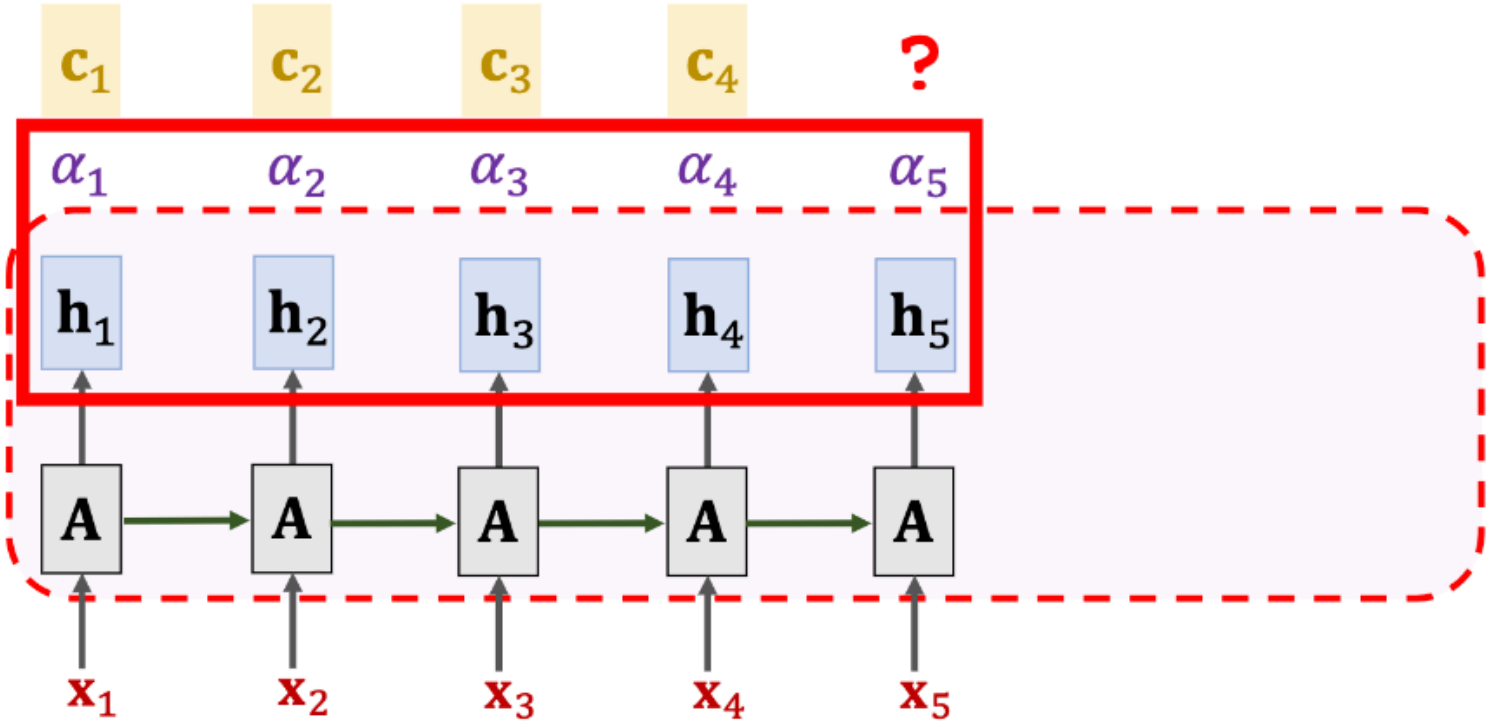


SimpleRNN + Self-Attention

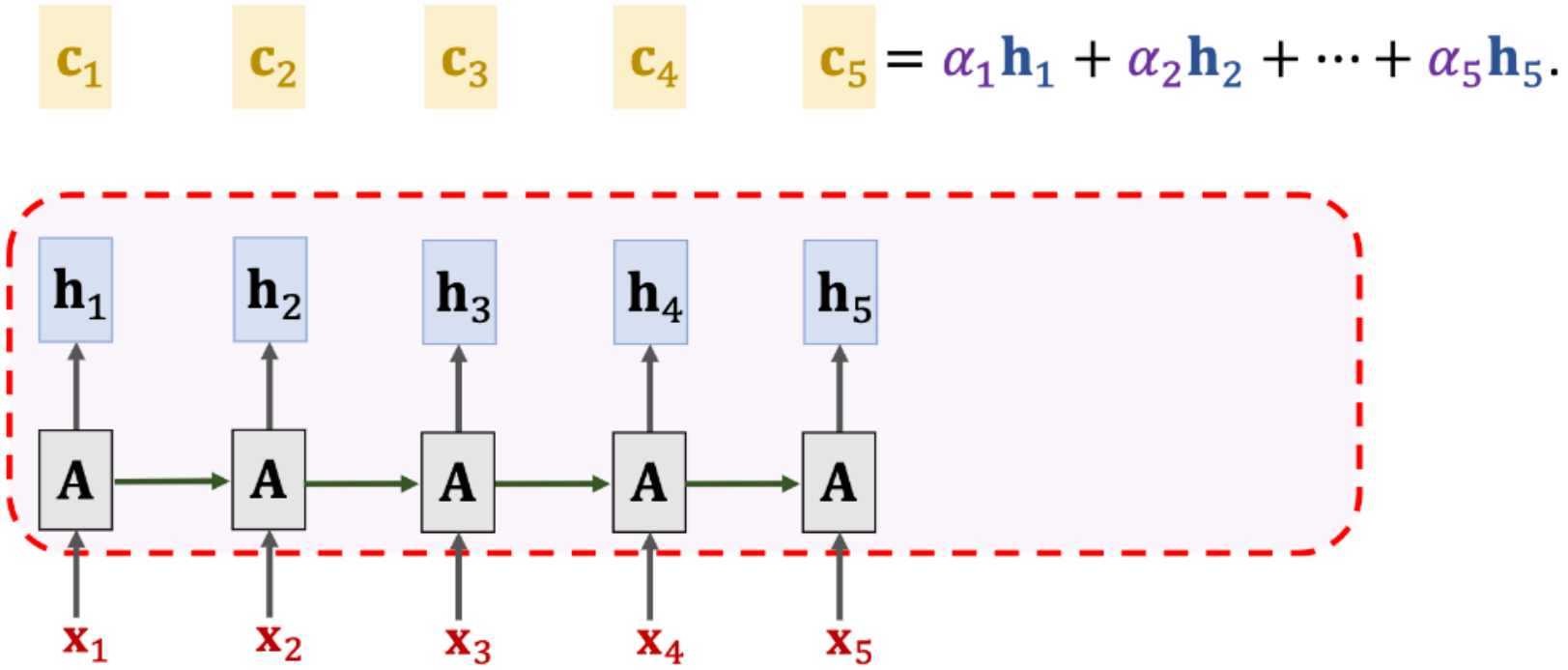
Weights: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{h}_5)$.



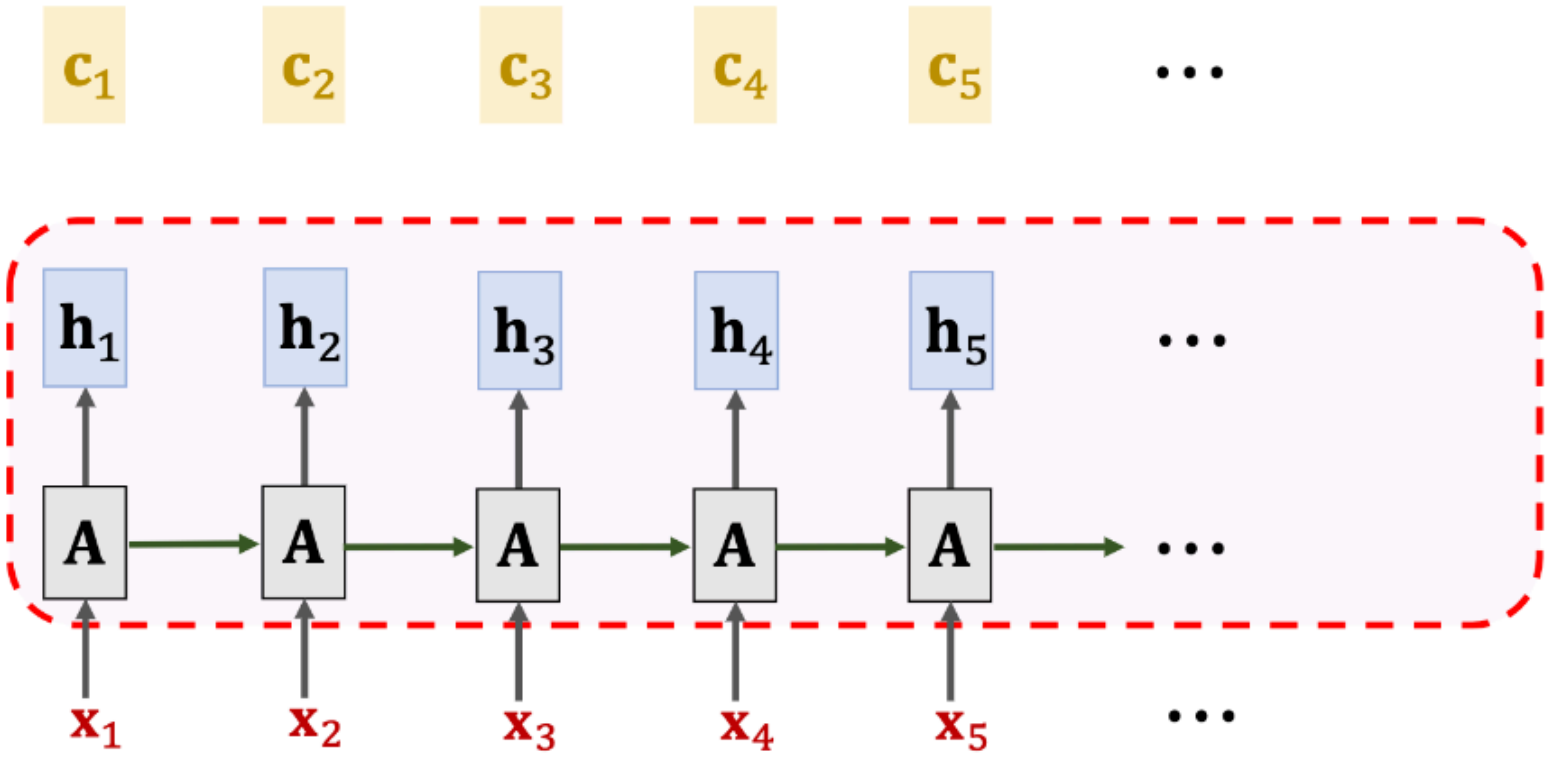
SimpleRNN + Self-Attention



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.

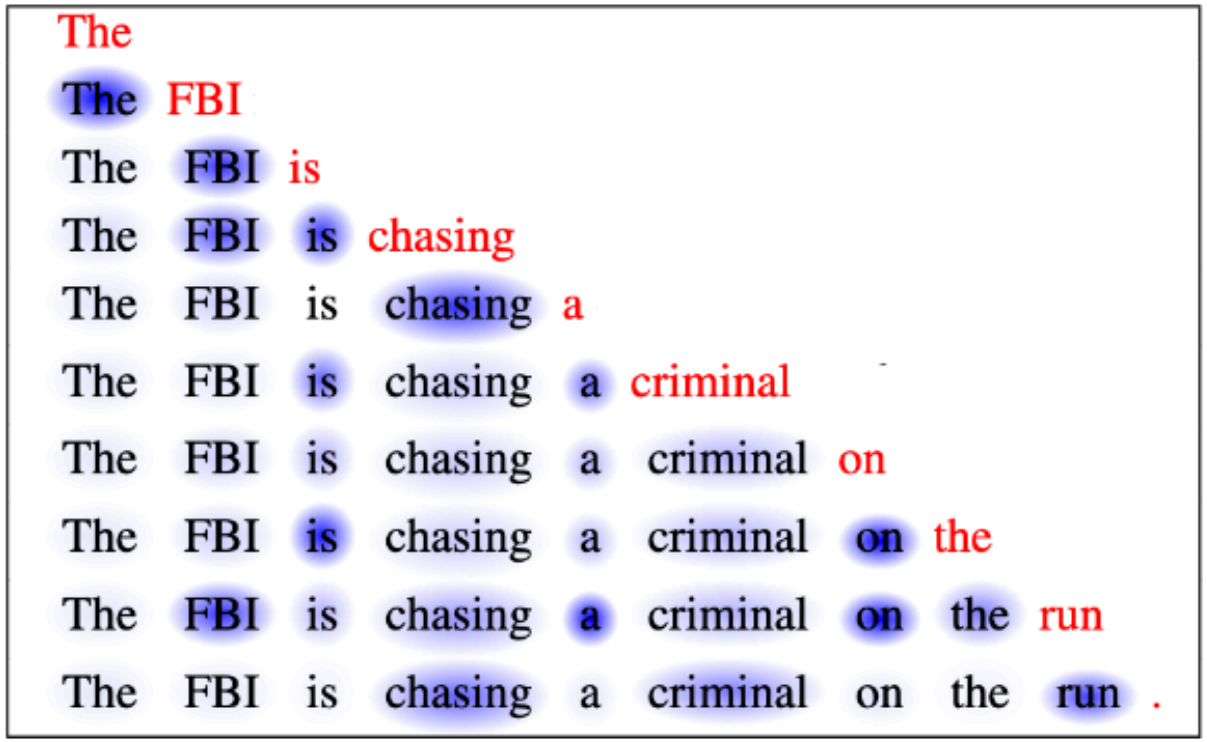


Figure is from the paper " Long Short-Term Memory-Networks for Machine Reading."

Any Question ?