Transformer Model (1/2): Attention without RNN

Shusen Wang

Transformer Model

- **Original paper:** Vaswani et al. Attention Is All You Need. In *NIPS*, 2017.

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

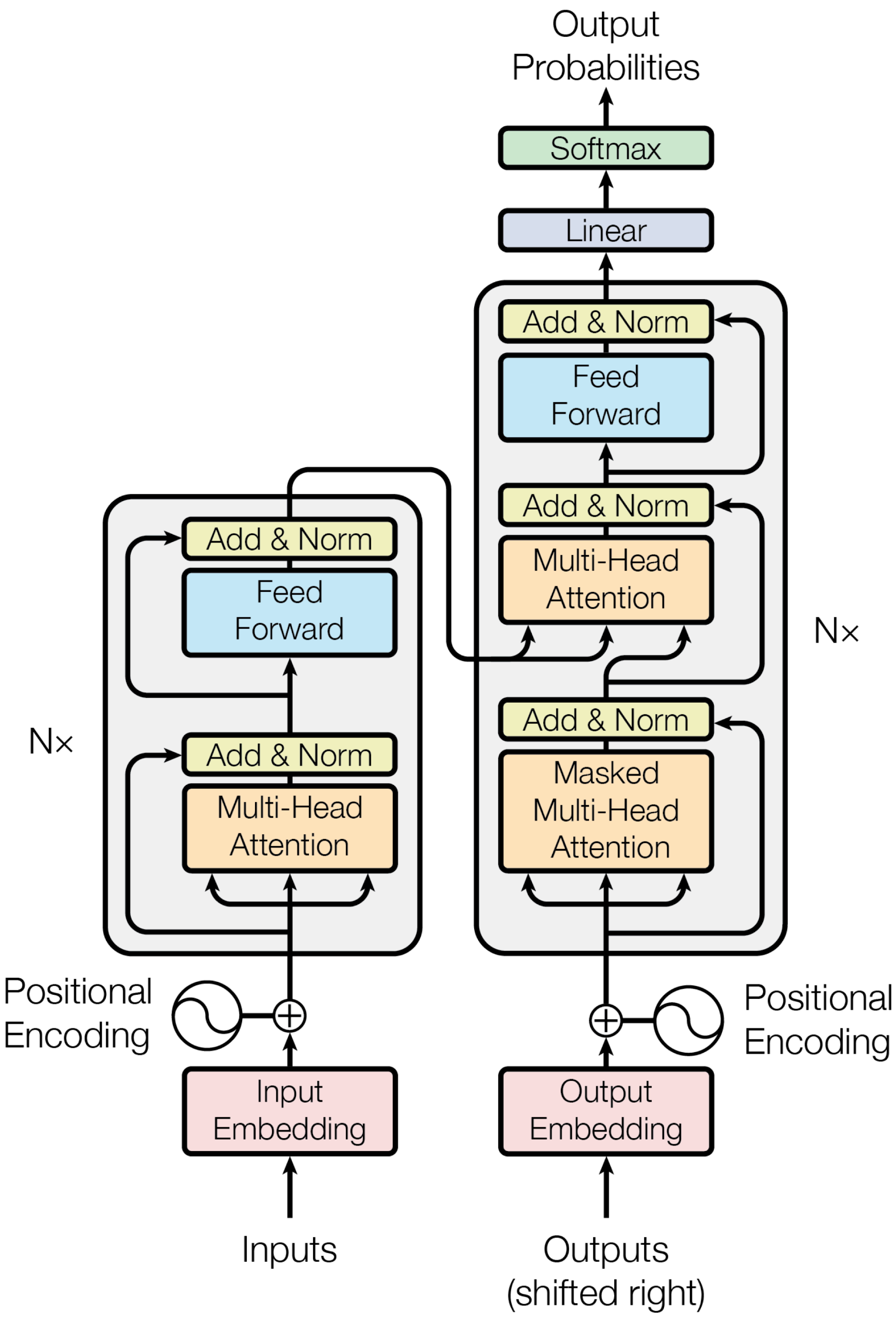
Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

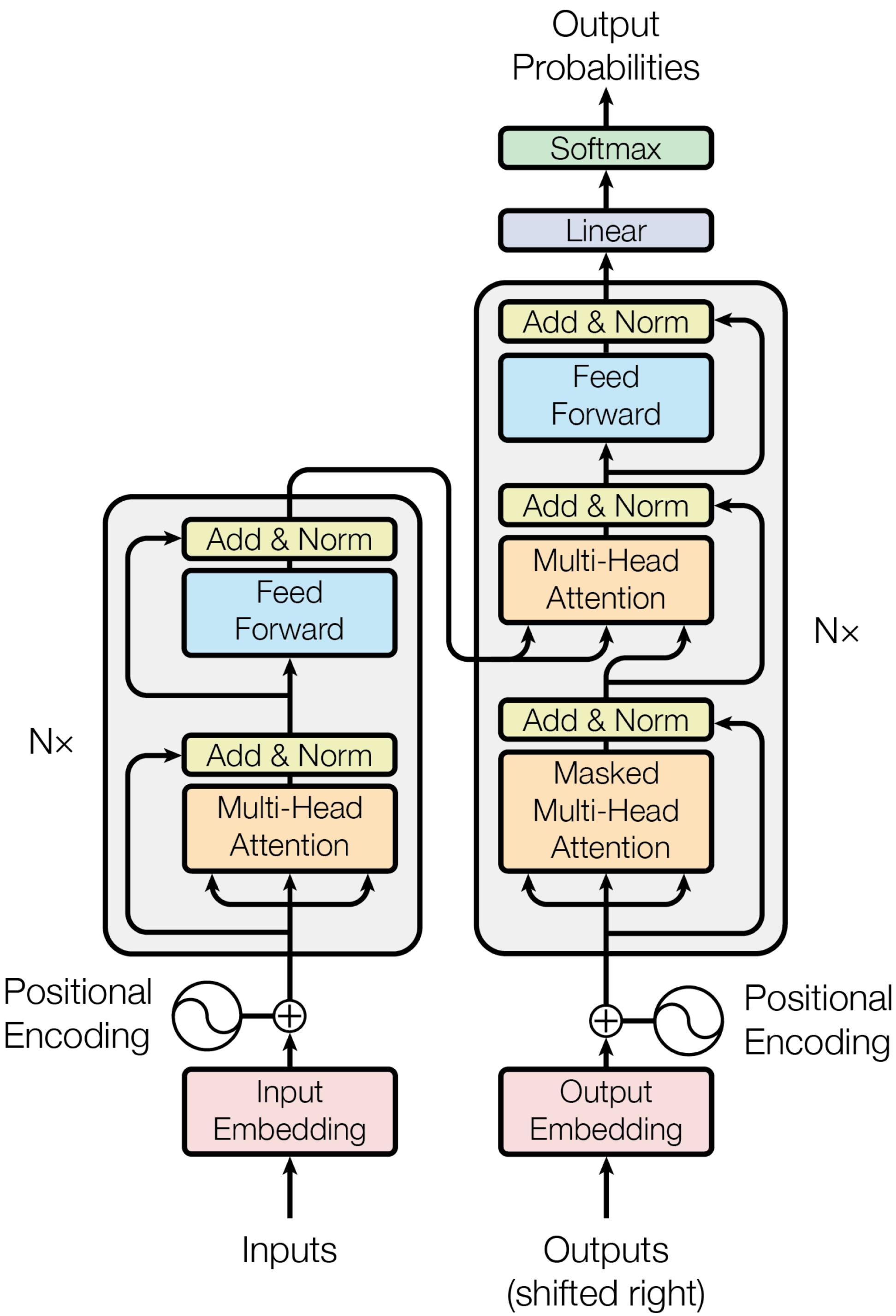


Transformer Model

- Transformer is a Seq2Seq model.
- Transformer is not RNN.
- Purely based attention and dense layers.

- Higher accuracy than RNNs on large datasets.

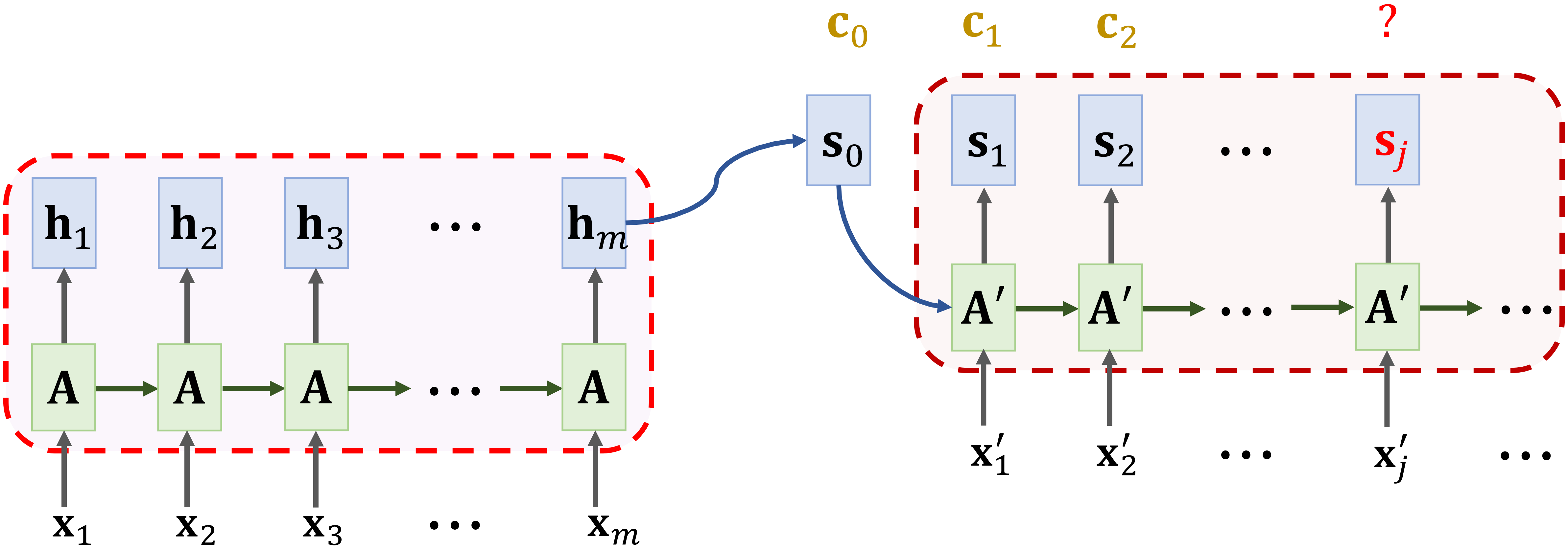
在大数据集上比rnn具有更高的精度



Revisiting Attention for RNN

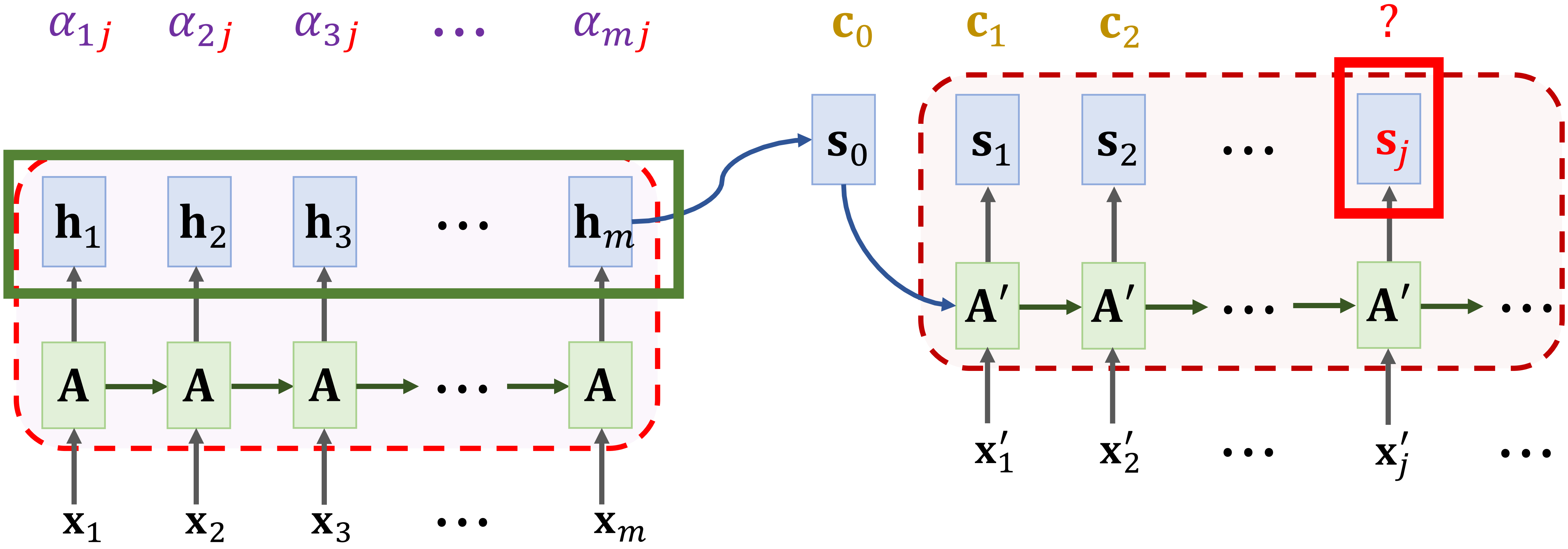
Attention for Seq2Seq Model

如何去掉RNN 只用Attention,怎么实现（本节内容）



Attention for Seq2Seq Model

Weights: $\alpha_{ij} = \mathrm{align}\left(\mathbf{h}_i, \mathbf{s}_j\right).$

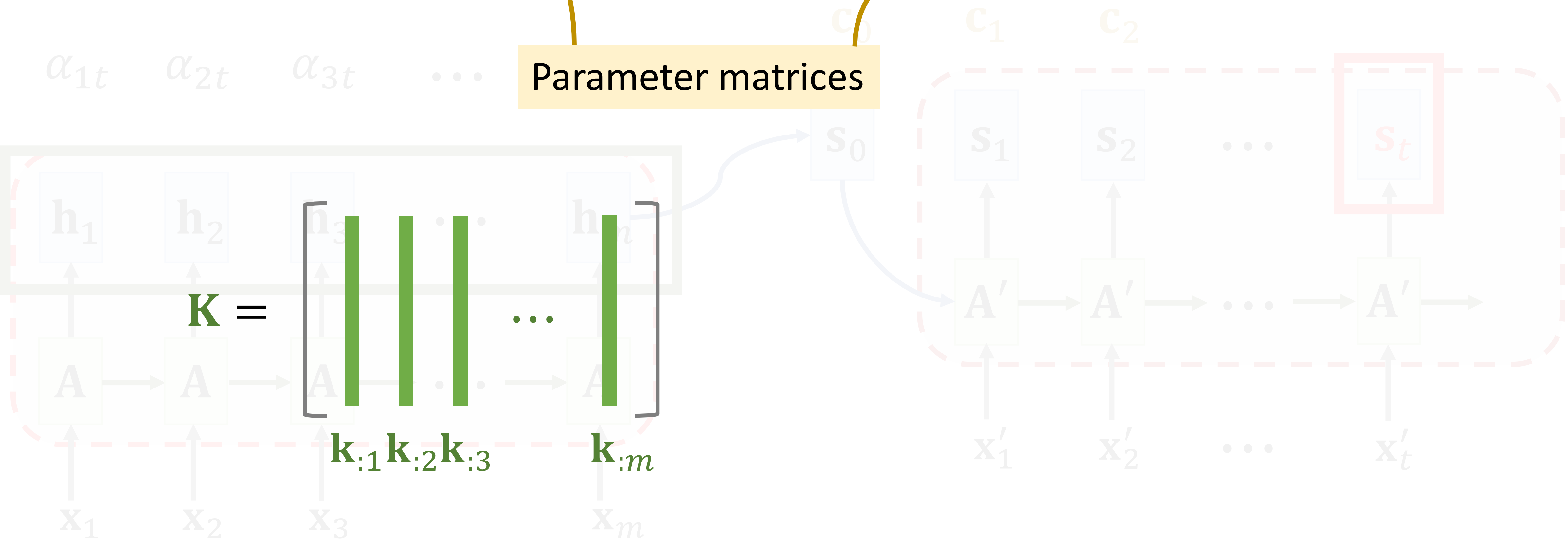


Attention for Seq2Seq Model

Weights: $\alpha_{ij} = \mathrm{align}\left(\mathbf{h}_i, \mathbf{s}_j\right)$.

- Compute $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{h}_i$ and $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{s}_j$.

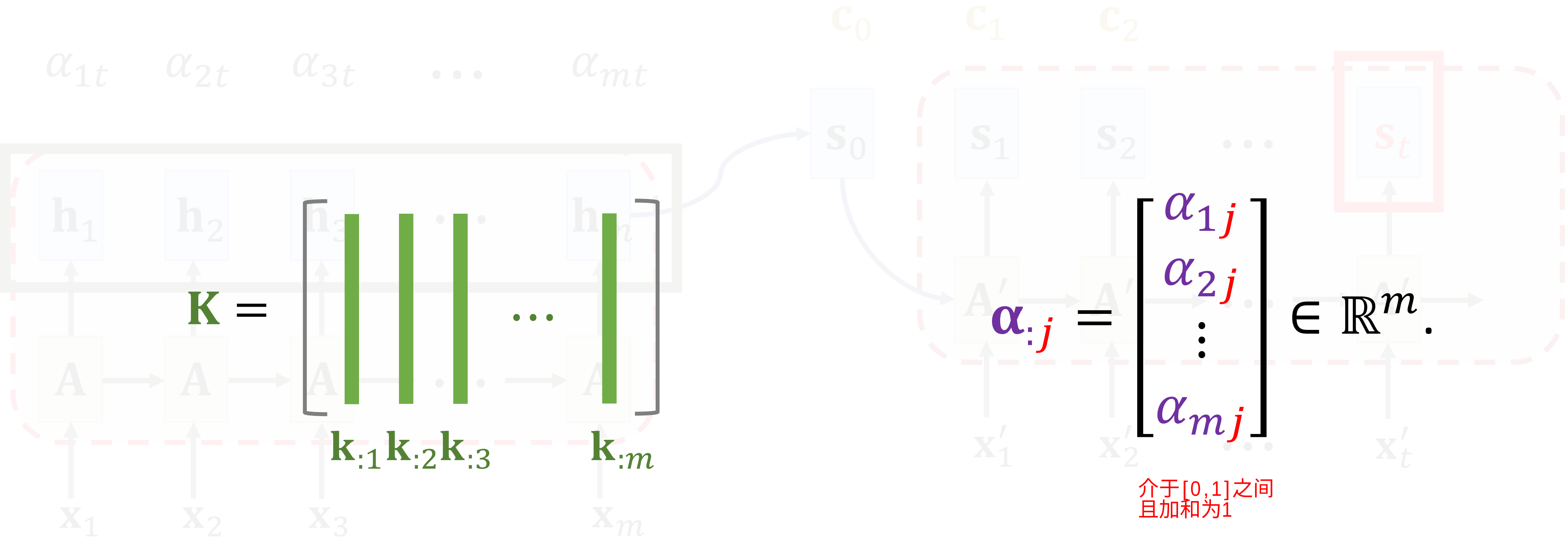
Parameter matrices



Attention for Seq2Seq Model

Weights: $\alpha_{ij} = \text{align}(\mathbf{h}_i, \mathbf{s}_j)$.

- Compute $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{h}_i$ and $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{s}_j$.
- Compute weights: $\boldsymbol{\alpha}_{:j} = \text{Softmax}(\mathbf{K}^T \mathbf{q}_{:j}) \in \mathbb{R}^m$.



$$\mathbf{K} = \begin{bmatrix} \mathbf{k}_{:1} & \mathbf{k}_{:2} & \mathbf{k}_{:3} & \cdots & \mathbf{k}_{:m} \end{bmatrix}$$

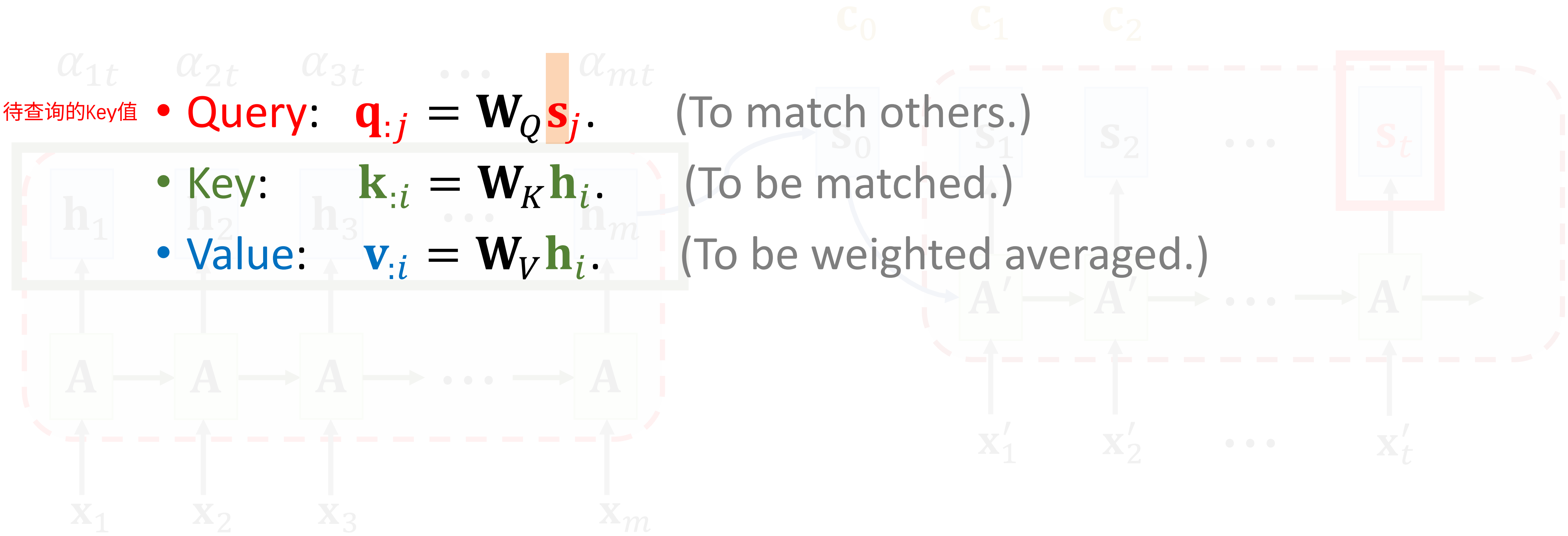
$$\boldsymbol{\alpha}_{:j} = \begin{bmatrix} \alpha_{1j} \\ \alpha_{2j} \\ \vdots \\ \alpha_{mj} \end{bmatrix} \in \mathbb{R}^m.$$

介于[0,1]之间
且加和为1

Attention for Seq2Seq Model

Weights: $\alpha_{ij} = \text{align}\left(\mathbf{h}_i, \mathbf{s}_j\right).$

- Compute $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{h}_i$ and $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{s}_j$.
- Compute weights: $\boldsymbol{\alpha}_{:j} = \text{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m$.

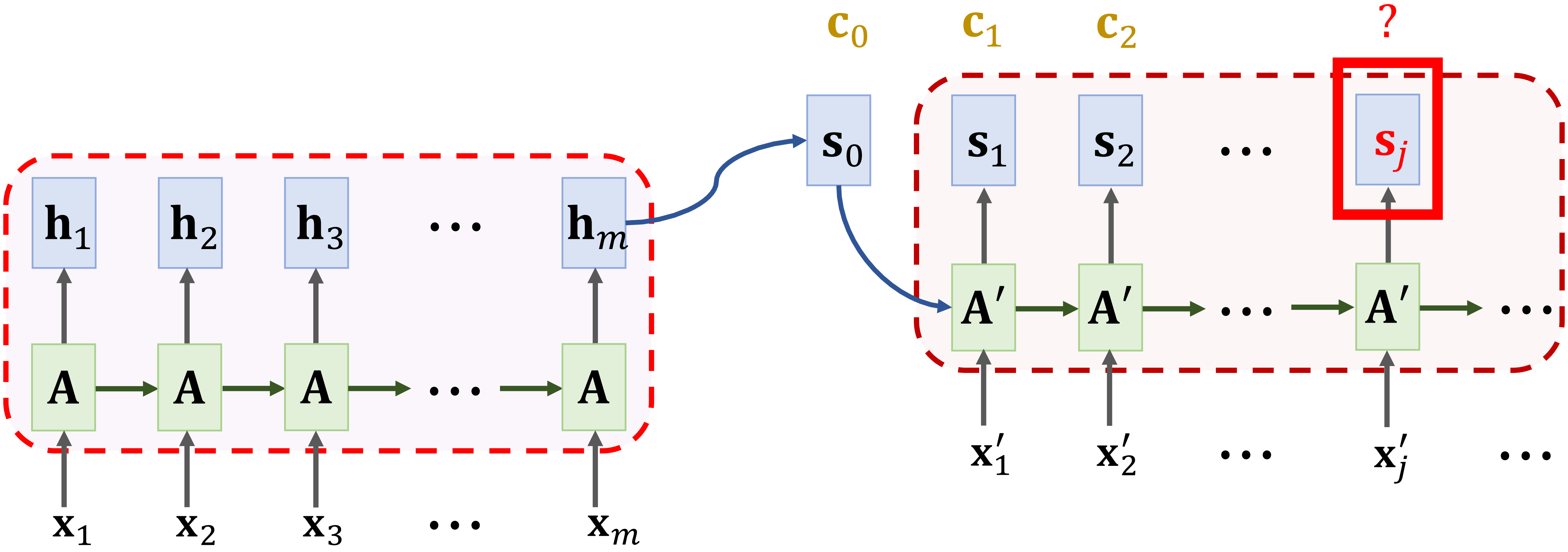


待查询的Key值

- Query: $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{s}_j$. (To match others.)
- Key: $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{h}_i$. (To be matched.)
- Value: $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{h}_i$. (To be weighted averaged.)

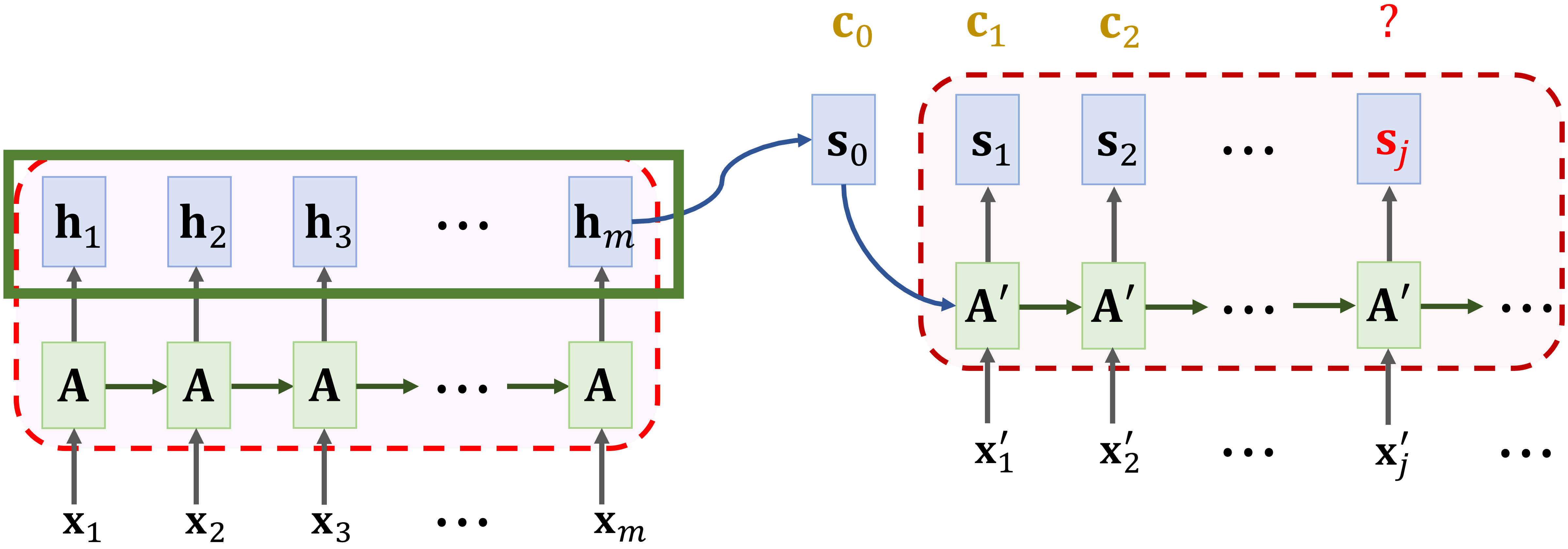
Attention for Seq2Seq Model

Query: $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{s}_j$, Key: $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{h}_i$, Value: $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{h}_i$.



Attention for Seq2Seq Model

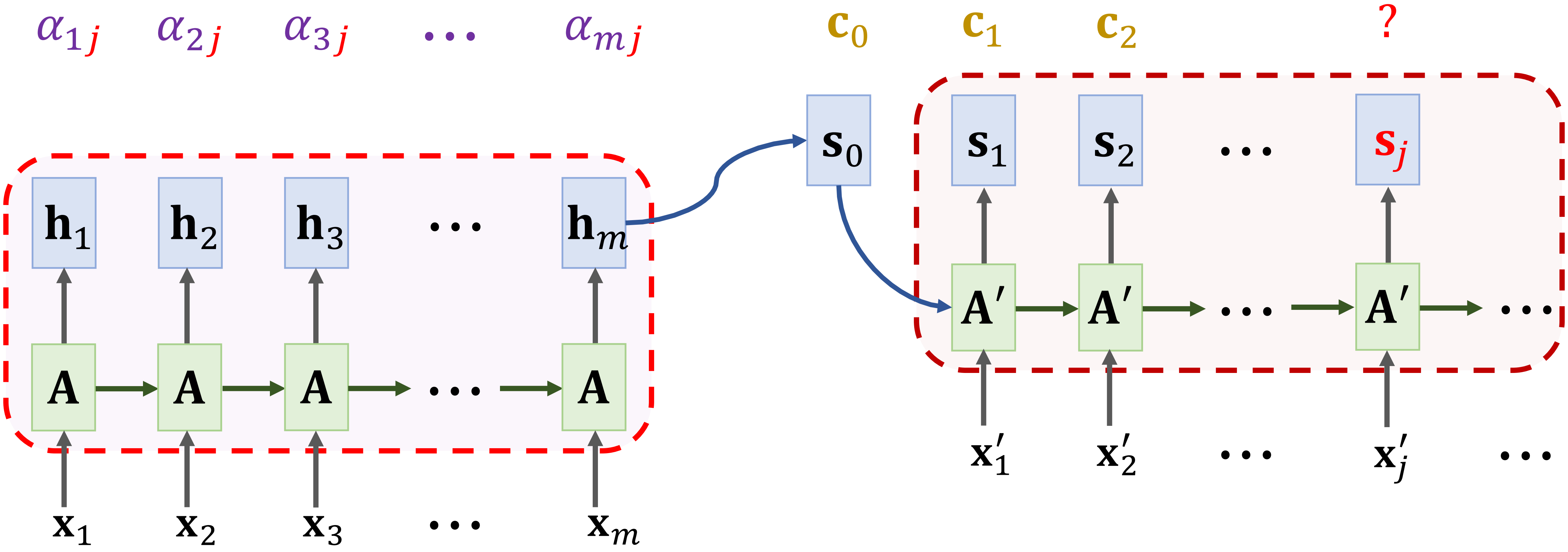
Query: $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{s}_j$, Key: $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{h}_i$, Value: $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{h}_i$.



Attention for Seq2Seq Model

Query: $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{s}_j$, Key: $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{h}_i$, Value: $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{h}_i$.

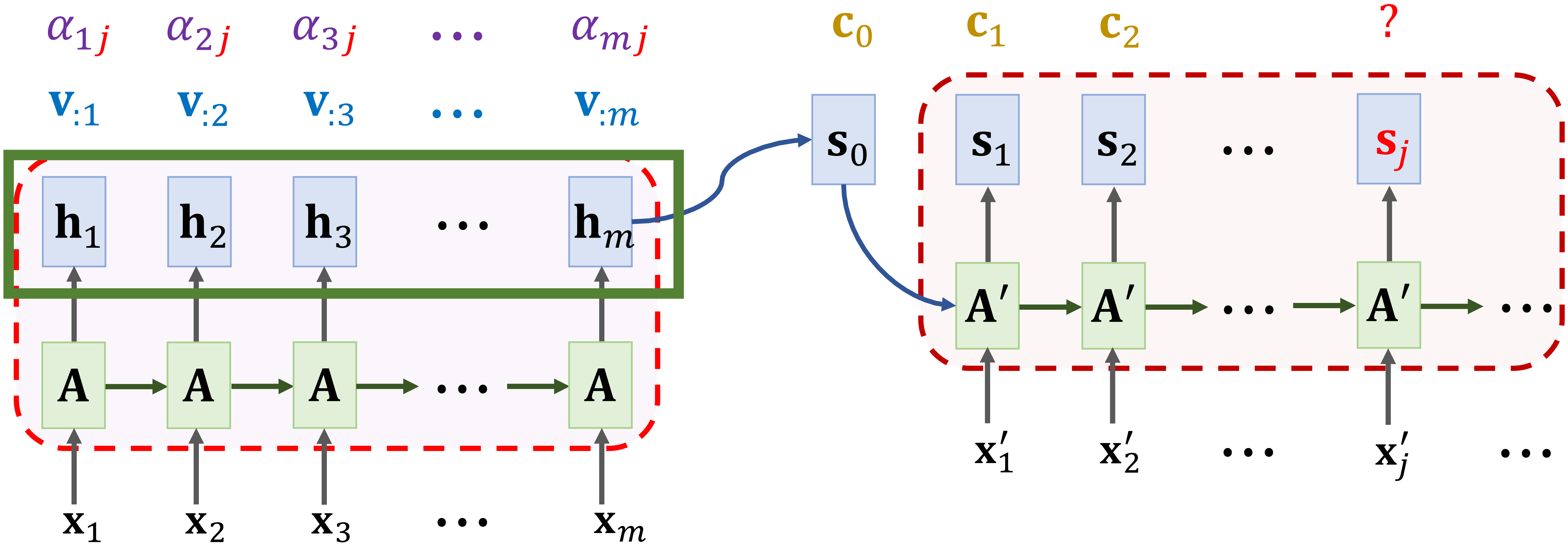
Weights: $\boldsymbol{\alpha}_{:j} = \mathrm{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m$.



Attention for Seq2Seq Model

Query: $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{s}_j$, Key: $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{h}_i$, Value: $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{h}_i$.

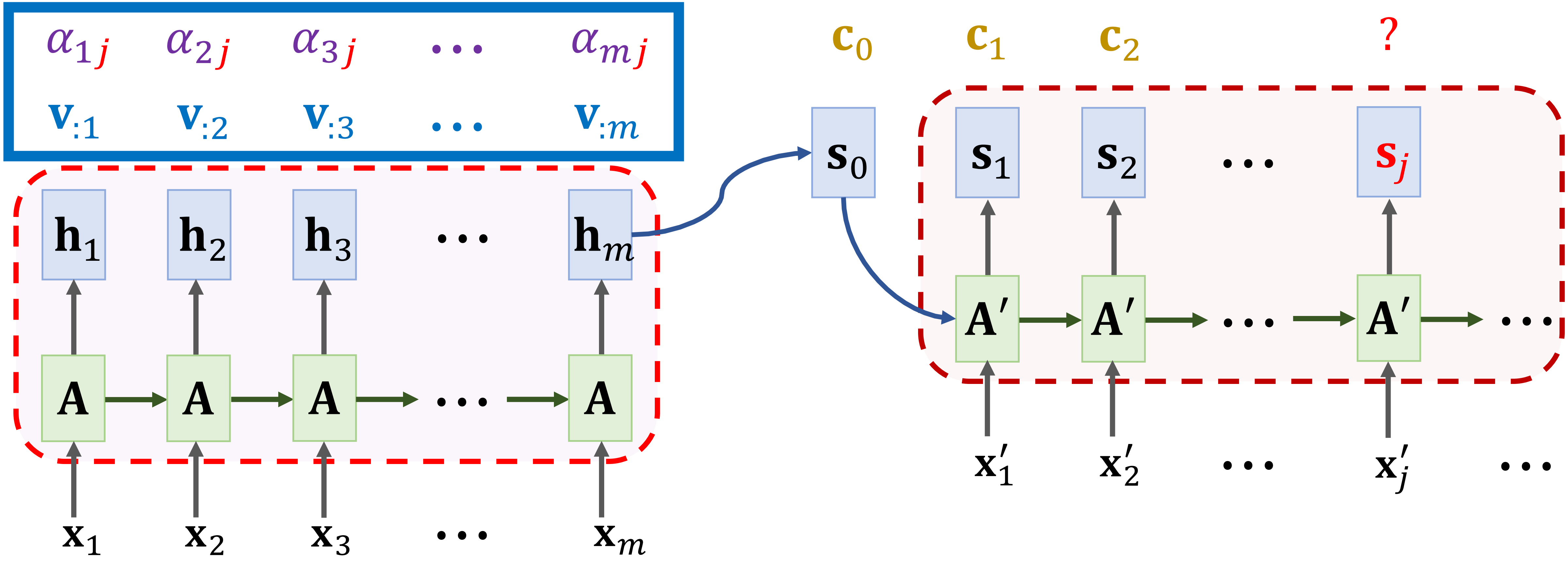
Weights: $\boldsymbol{\alpha}_{:j} = \mathrm{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m$.



Attention for Seq2Seq Model

Query: $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{s}_j$, Key: $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{h}_i$, Value: $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{h}_i$.

Weights: $\boldsymbol{\alpha}_{:j} = \mathrm{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m$.

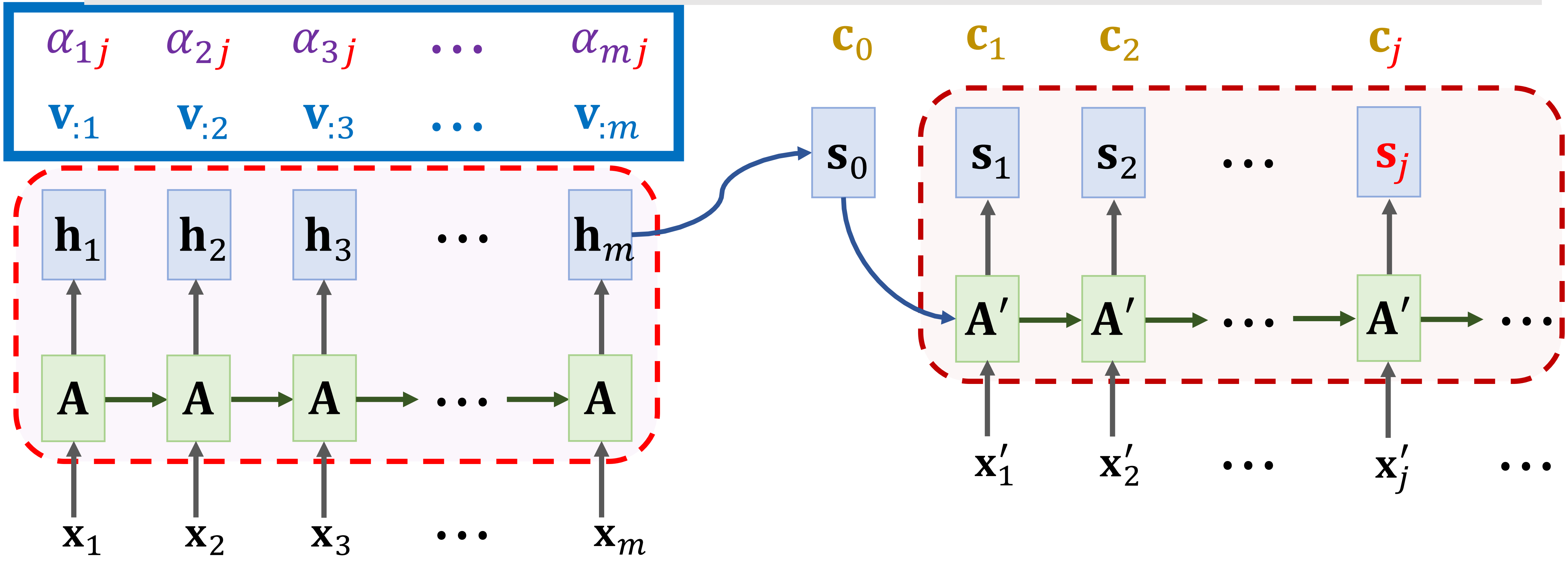


Attention for Seq2Seq Model

Query: $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{s}_j$, Key: $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{h}_i$, Value: $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{h}_i$.

Weights: $\boldsymbol{\alpha}_{:j} = \mathrm{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m$.

Context vector: $\mathbf{c}_j = \alpha_{1j} \mathbf{v}_{:1} + \cdots + \alpha_{mj} \mathbf{v}_{:m}$.



Attention for Seq2Seq Model

Query: $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{s}_j$, Key: $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{h}_i$, Value: $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{h}_i$.

Weights: $\boldsymbol{\alpha}_{:j} = \mathrm{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m$.

Context vector: $\mathbf{c}_j = \alpha_{1j} \mathbf{v}_{:1} + \cdots + \alpha_{mj} \mathbf{v}_{:m}$.

Question: How to remove RNN while keeping attention?

如何在保持注意力的同时删除RNN

Attention without RNN

Attention Layer

- We study Seq2Seq model (encoder + decoder).
- Encoder's inputs are vectors $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$.
- Decoder's inputs are vectors $\mathbf{x}'_1, \mathbf{x}'_2, \cdots, \mathbf{x}'_t$.

Encoder's inputs:

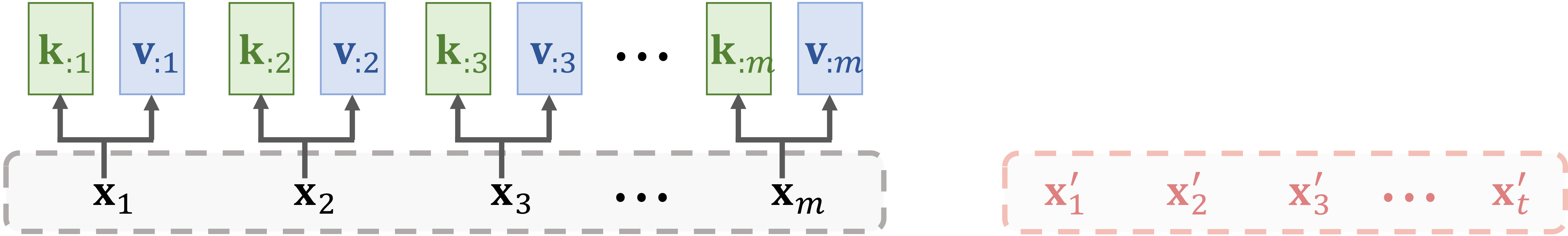
$\mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_3 \quad \cdots \quad \mathbf{x}_m$

Decoder's inputs:

$\mathbf{x}'_1 \quad \mathbf{x}'_2 \quad \mathbf{x}'_3 \quad \cdots \quad \mathbf{x}'_t$

Attention Layer

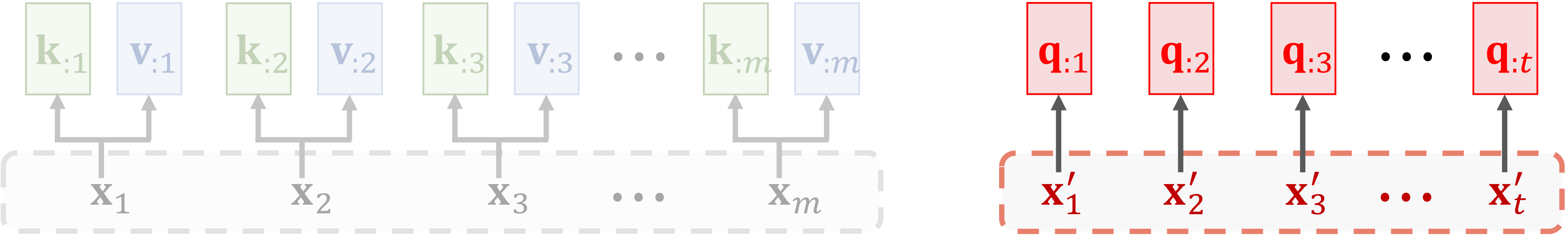
- Keys and values are based on encoder's inputs $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$.
- Key: $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{x}_i$.
- Value: $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{x}_i$.



Attention Layer

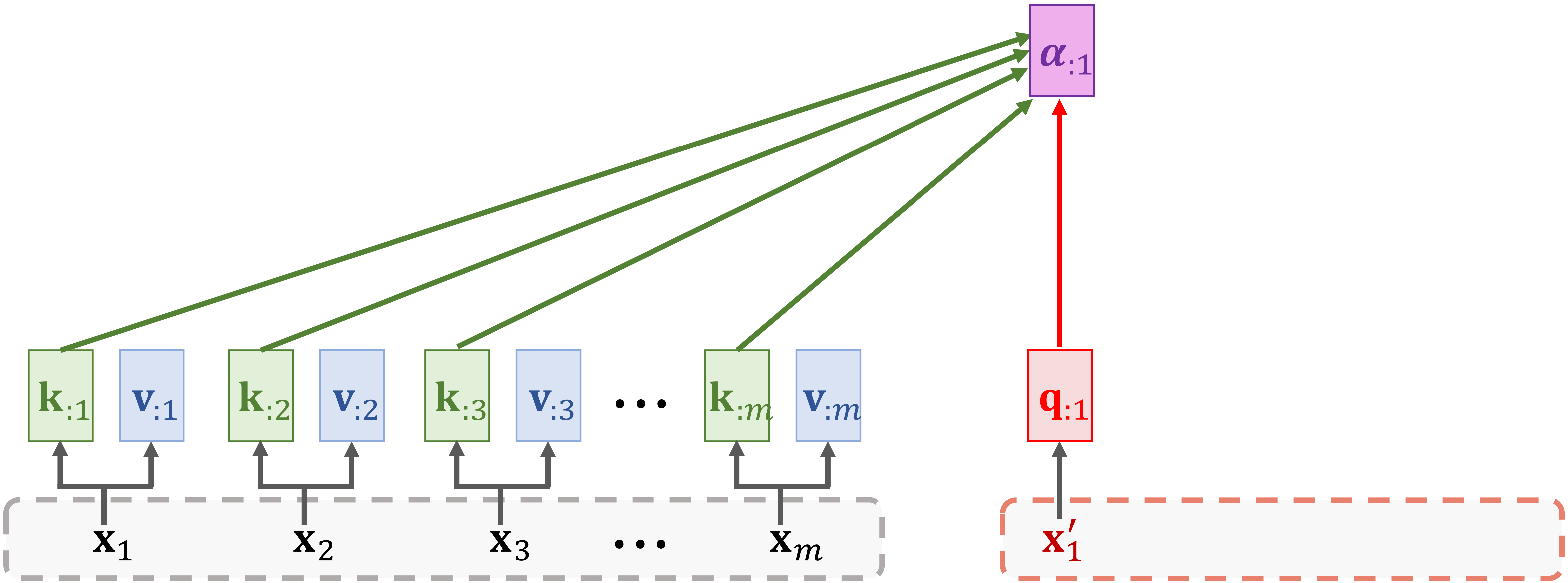
- Keys and values are based on encoder's inputs $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$.
- Key: $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{x}_i$.
- Value: $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{x}_i$.

- Queries are based on decoder's inputs $\mathbf{x}'_1, \mathbf{x}'_2, \cdots, \mathbf{x}'_t$.
- Query: $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{x}'_j$.



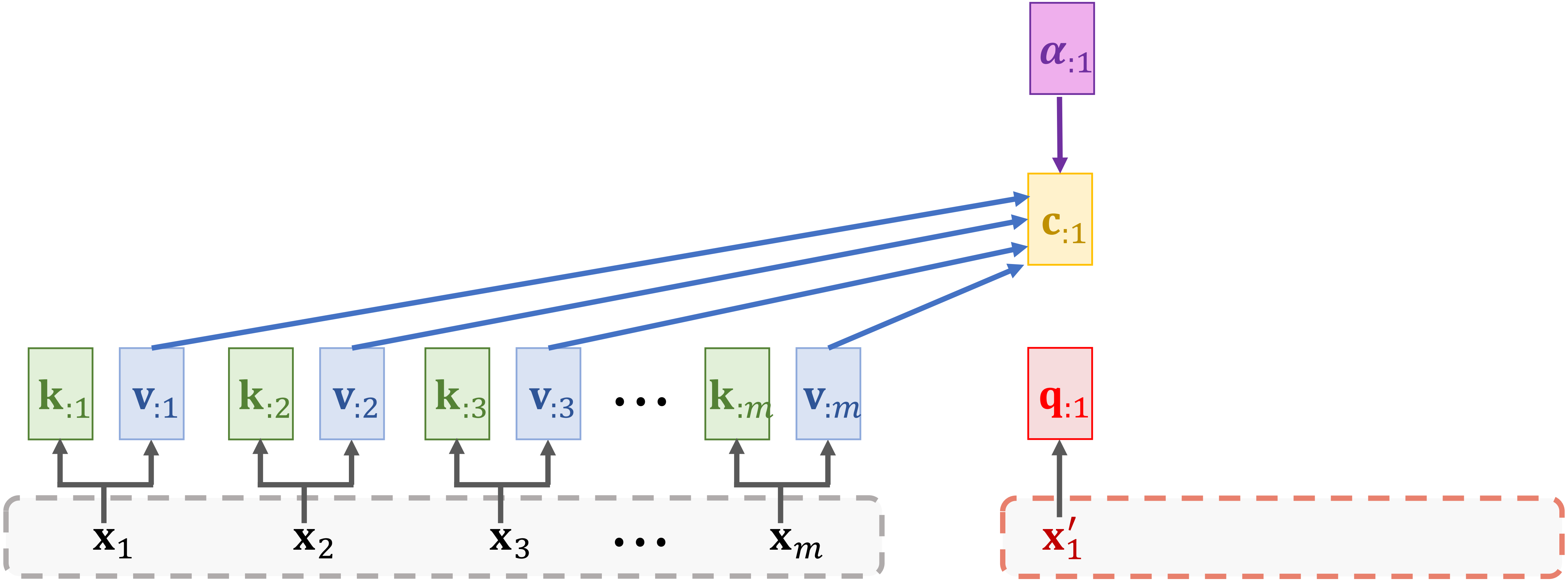
Attention Layer

- Compute weights: $\boldsymbol{\alpha}_{:1} = \mathrm{Softmax}(\mathbf{K}^T \mathbf{q}_{:1}) \in \mathbb{R}^m$.



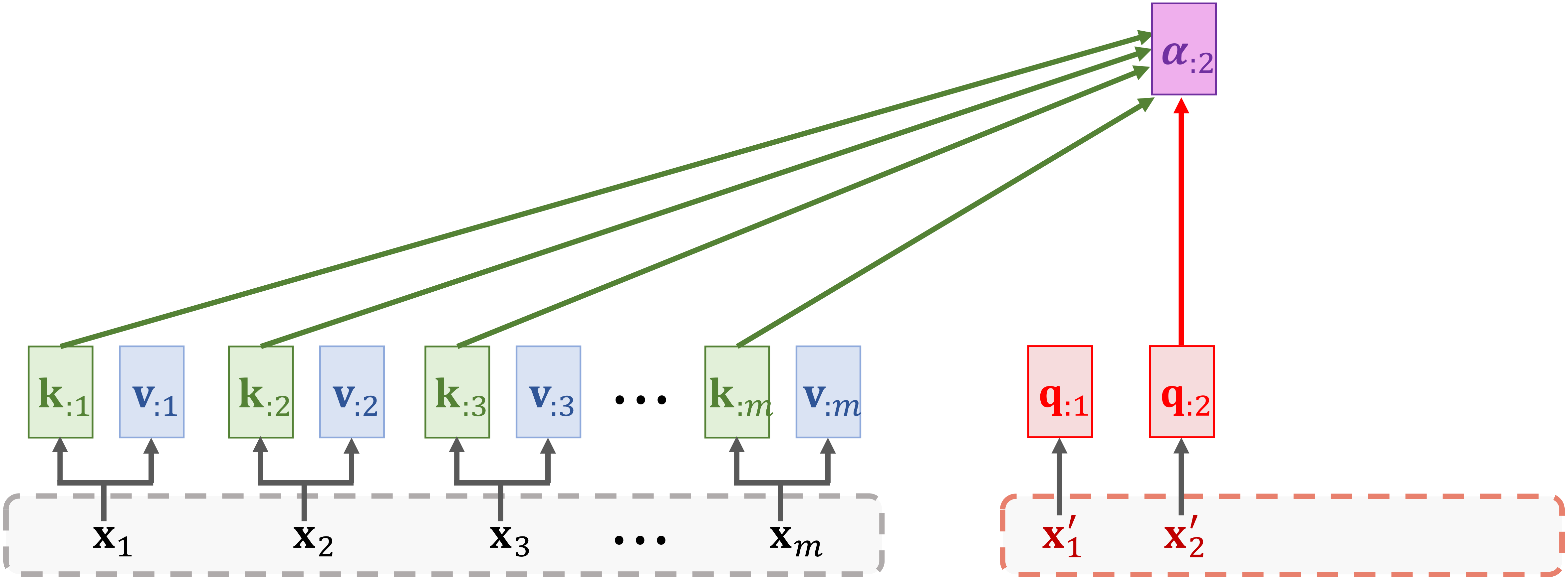
Attention Layer

- Compute context vector: $\mathbf{c}_{:1} = \alpha_{11}\mathbf{v}_{:1} + \cdots + \alpha_{m1}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:1}$.



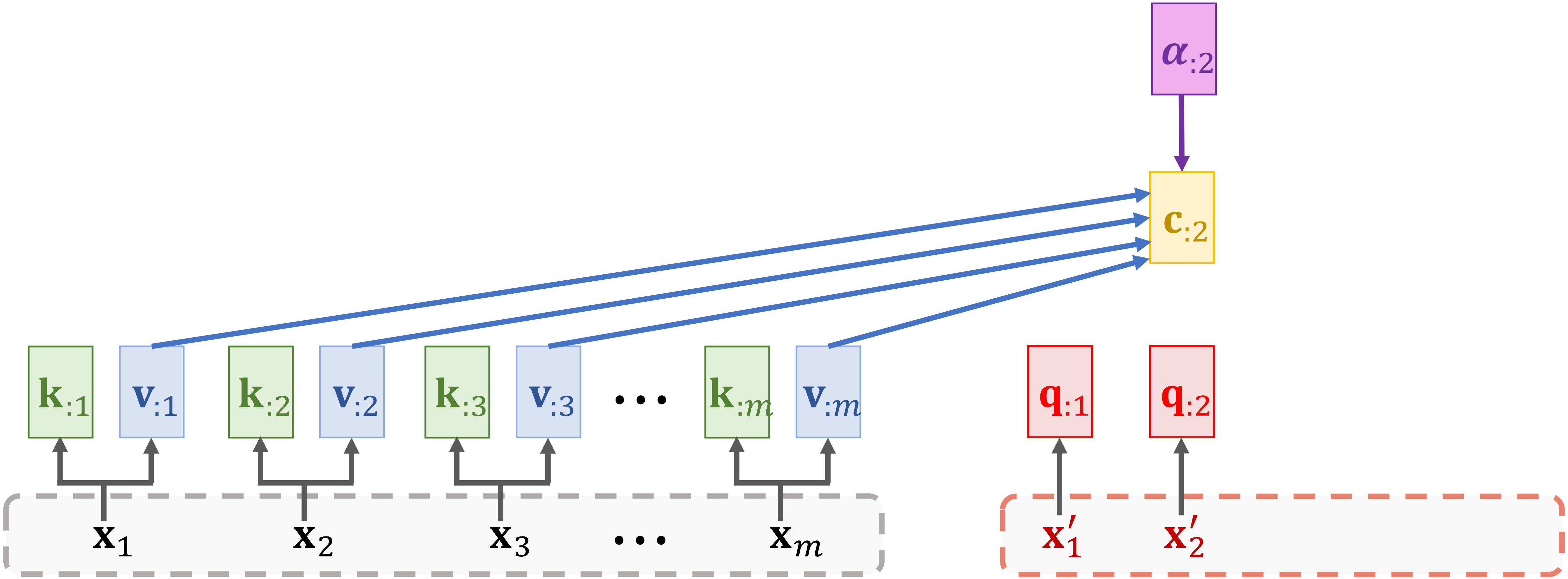
Attention Layer

- Compute weights: $\boldsymbol{\alpha}_{:2} = \mathrm{Softmax}(\mathbf{K}^T \mathbf{q}_{:2}) \in \mathbb{R}^m$.



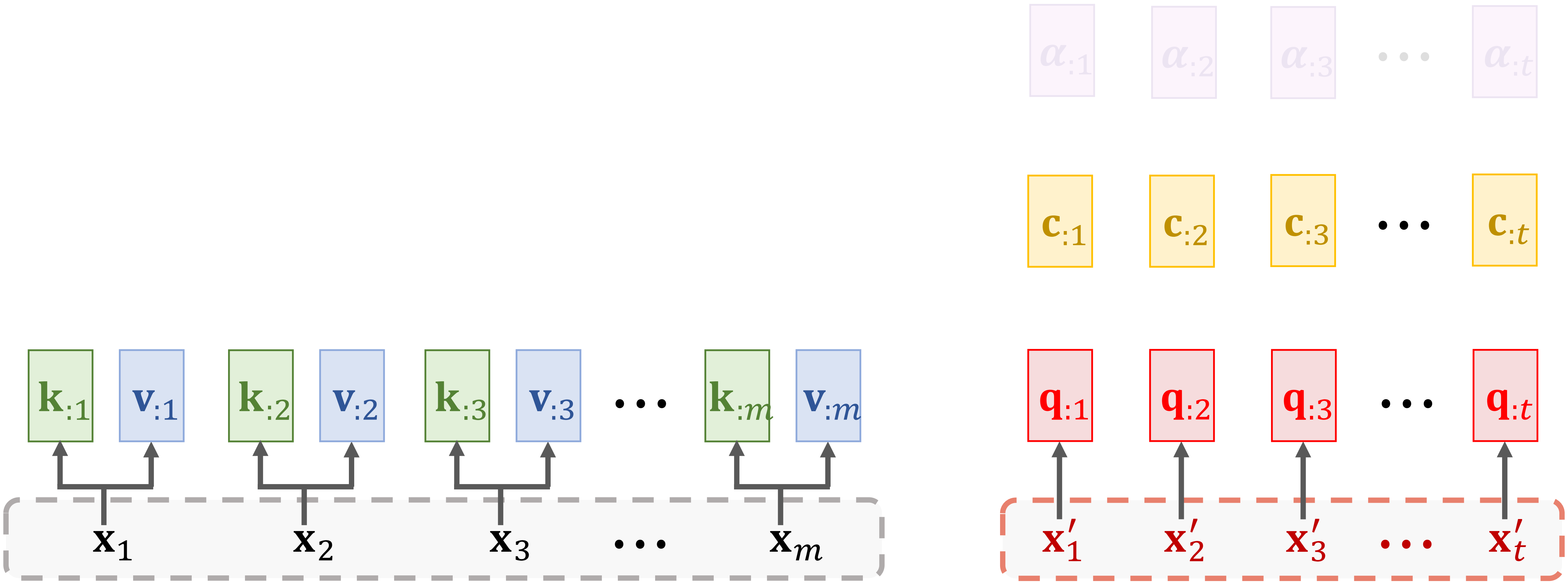
Attention Layer

- Compute context vector: $\mathbf{c}_{:2} = \alpha_{12}\mathbf{v}_{:1} + \cdots + \alpha_{m2}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:2}$.



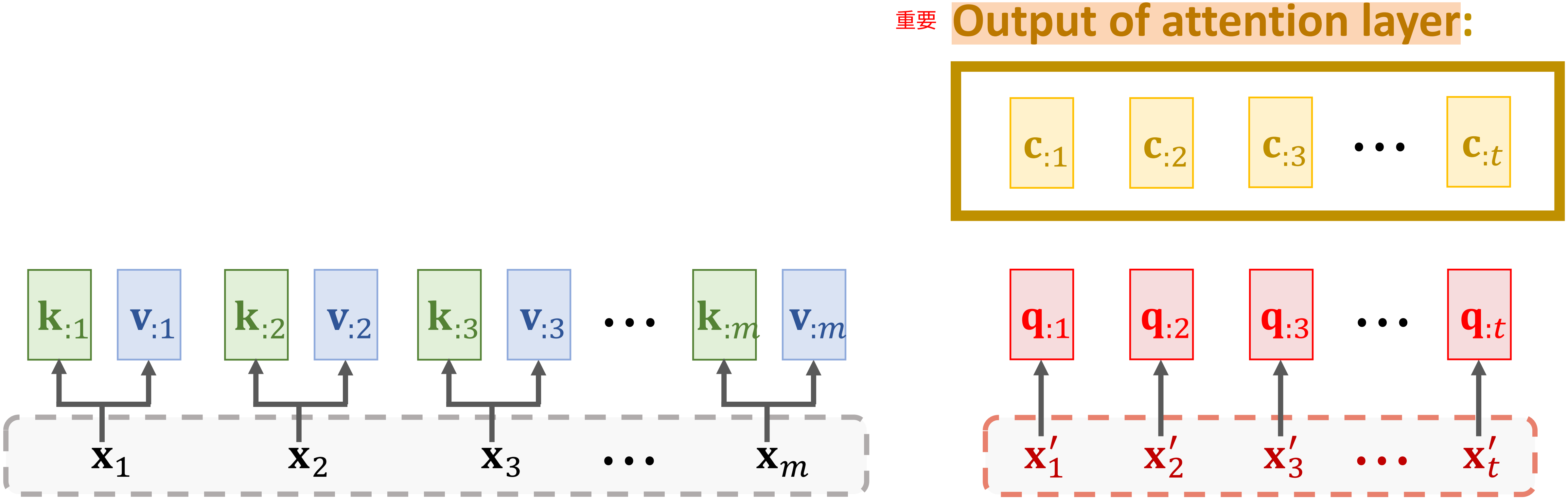
Attention Layer

- Compute context vector: $\mathbf{c}_{:j} = \alpha_{1j}\mathbf{v}_{:1} + \cdots + \alpha_{mj}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:j}.$



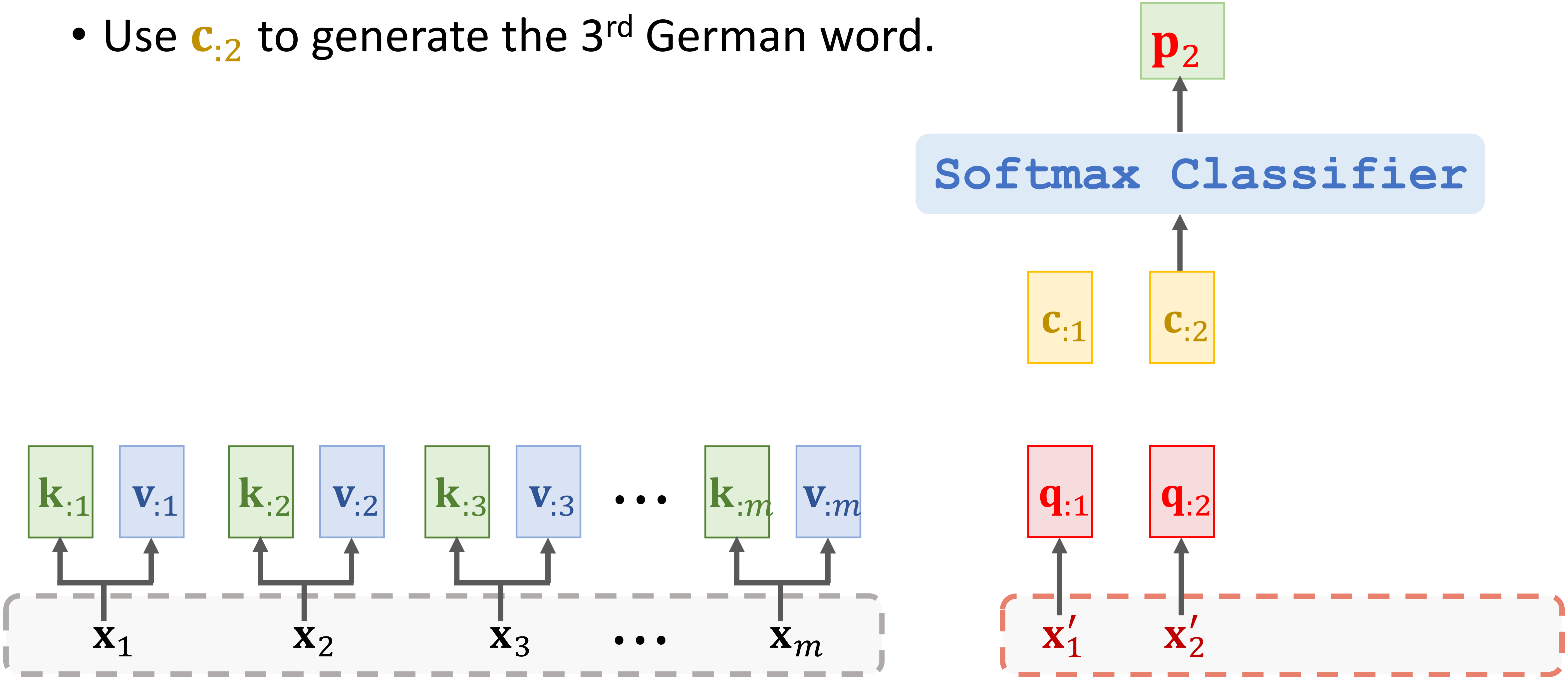
Attention Layer

- Output of attention layer: $\mathbf{C} = [\mathbf{c}_{:1}, \mathbf{c}_{:2}, \mathbf{c}_{:3}, \cdots, \mathbf{c}_{:t}]$.
- Here, $\mathbf{c}_{:j} = \mathbf{V} \cdot \mathrm{Softmax}(\mathbf{K}^T \mathbf{q}_{:j})$.
- Thus, $\mathbf{c}_{:j}$ is a function of $\mathbf{x}'_j$ and $[\mathbf{x}_1, \cdots, \mathbf{x}_m]$.



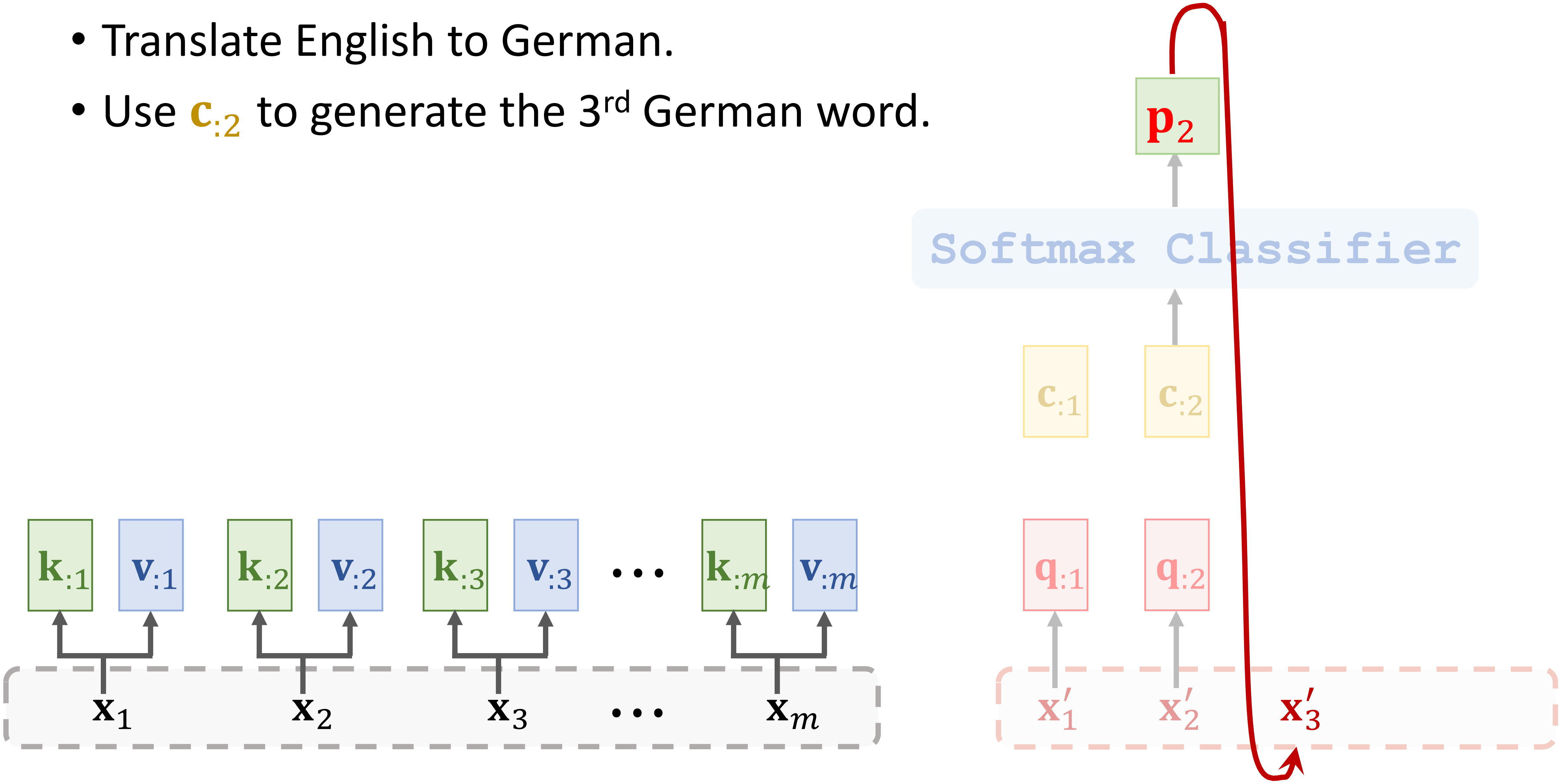
Attention Layer for Machine Translation

- Translate English to German.
- Use $\mathbf{c}_{:2}$ to generate the 3rd German word.



Attention Layer for Machine Translation

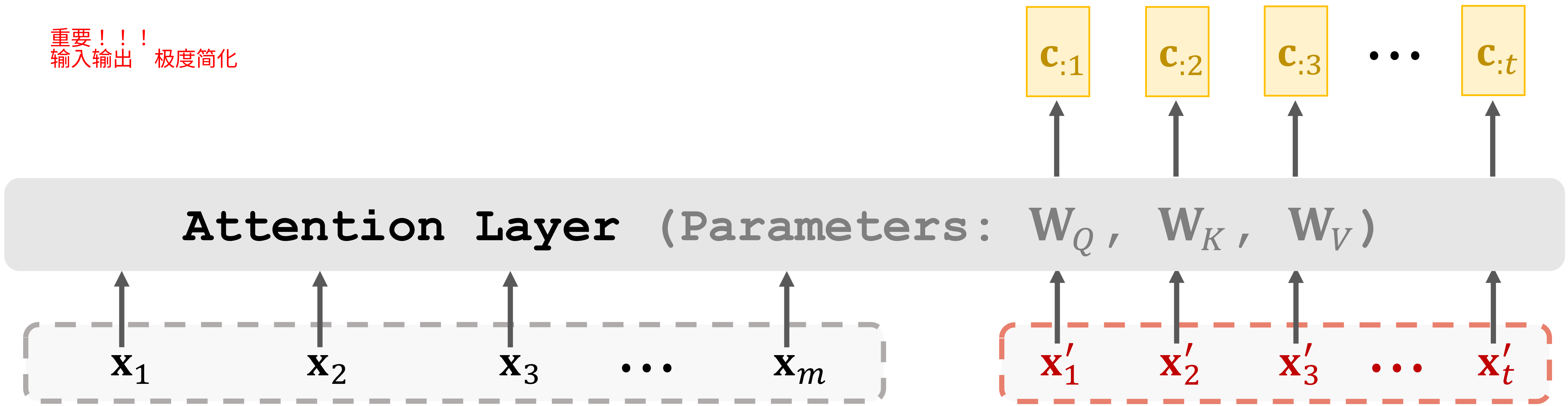
- Translate English to German.
- Use $\mathbf{c}_{:2}$ to generate the 3rd German word.



Attention Layer

- Attention layer: $\mathbf{C} = \mathrm{Attn}(\mathbf{X}, \mathbf{X}')$.
  - Encoder's inputs: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]$.
  - Decoder's inputs: $\mathbf{X}' = [\mathbf{x}'_1, \mathbf{x}'_2, \cdots, \mathbf{x}'_t]$.
  - Parameters: $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$.

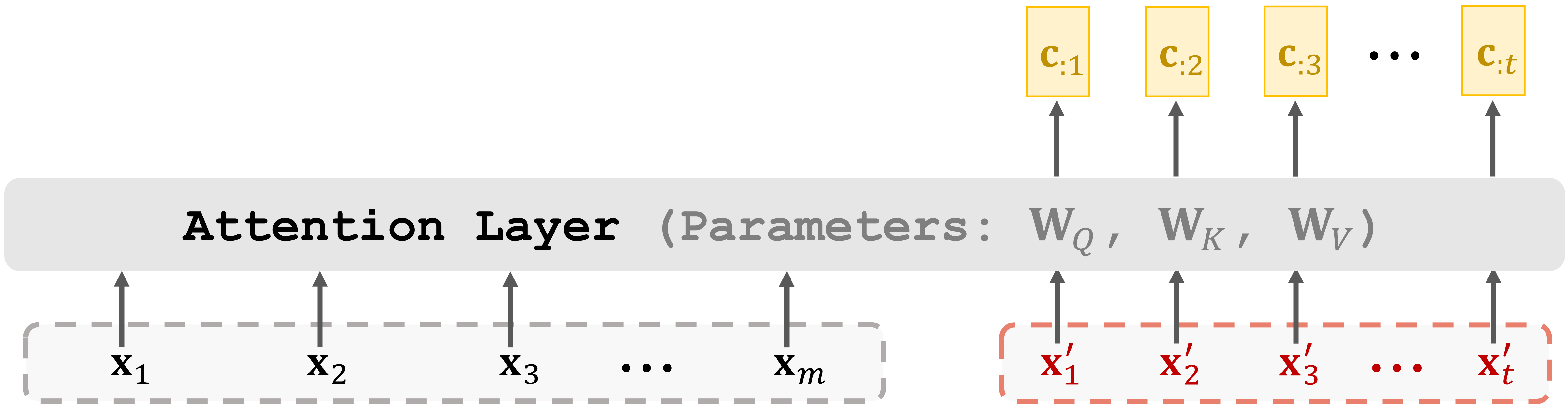
重要！！！
输入输出　极度简化



Self-Attention without RNN

Attention Layer

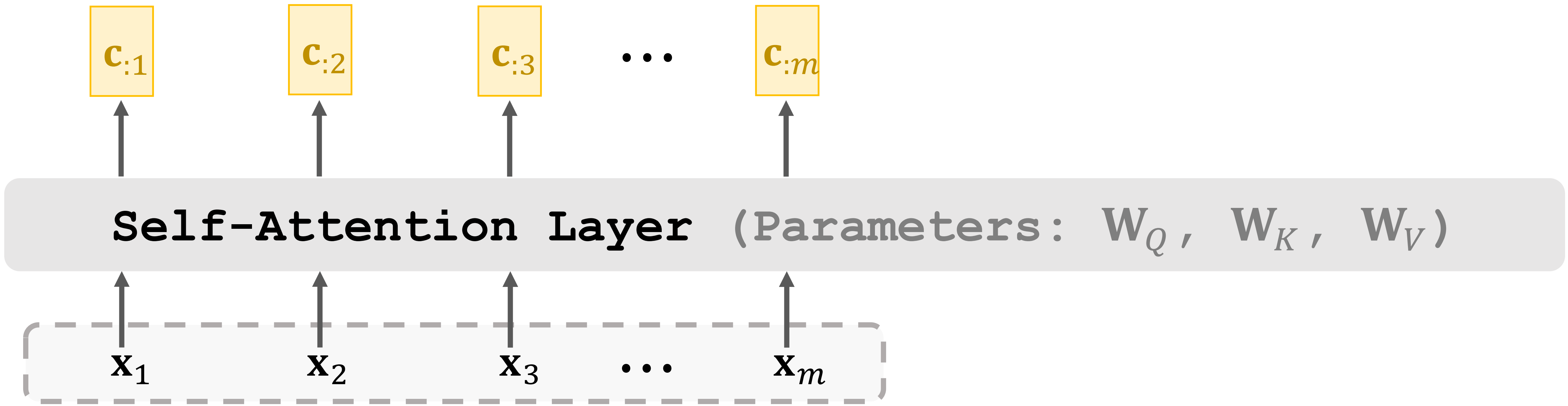
- Attention layer: $\mathbf{C} = \mathrm{Attn}(\mathbf{X}, \mathbf{X}')$.
  - Encoder's inputs: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]$.
  - Decoder's inputs: $\mathbf{X}' = [\mathbf{x}'_1, \mathbf{x}'_2, \cdots, \mathbf{x}'_m]$.
  - Parameters: $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$.



Self-Attention Layer

- Self-attention layer: $\mathbf{C} = \text{Attn}(\mathbf{X}, \mathbf{X})$.
  - Inputs: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]$.
  - Parameters: $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$.

重要！！！
输入输出 极度简化



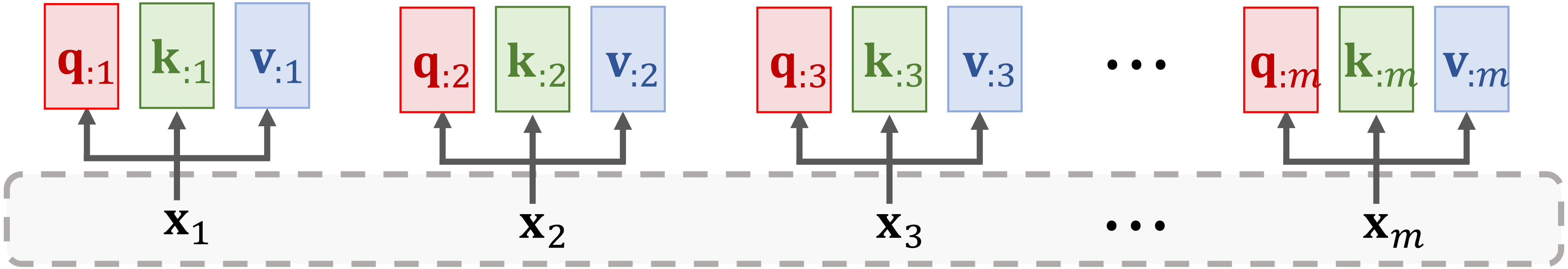
Self-Attention Layer

Inputs:

$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\cdots$ $\mathbf{x}_m$

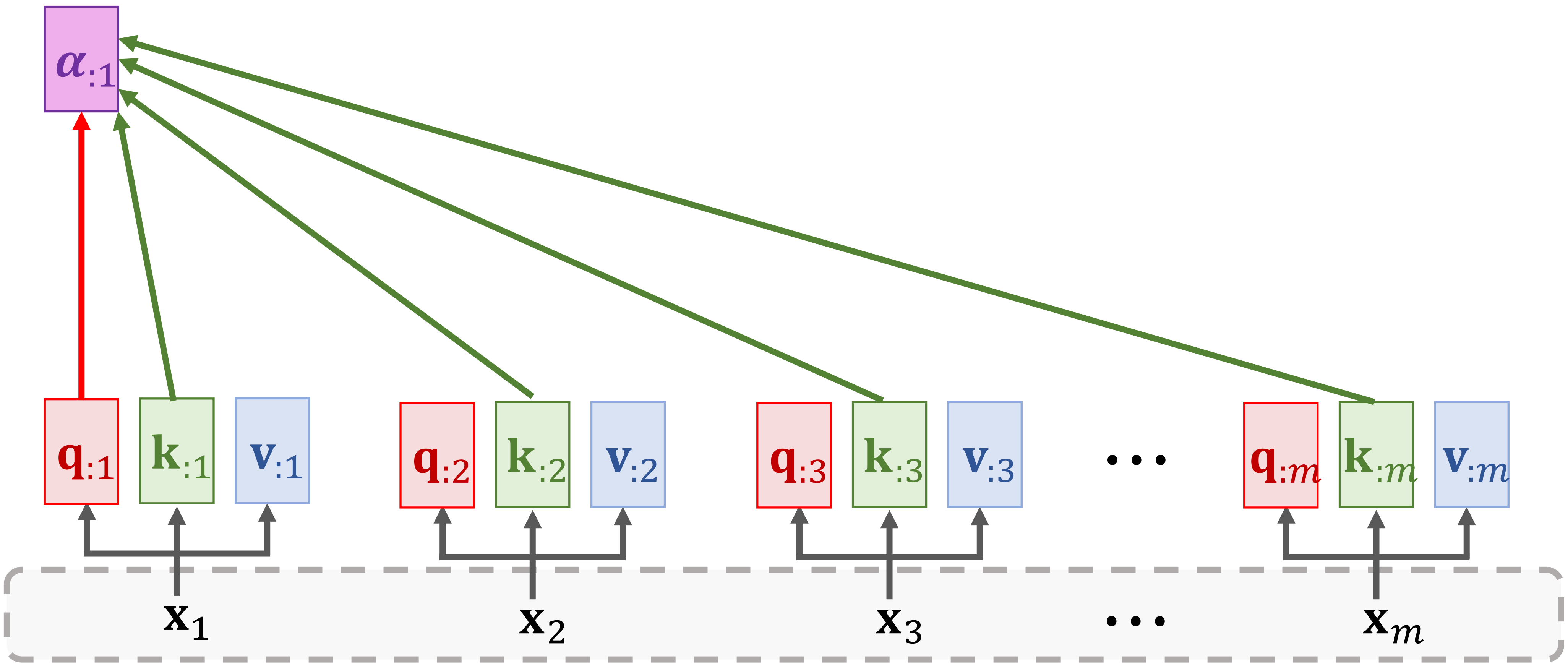
Self-Attention Layer

Query: $\mathbf{q}_{:i} = \mathbf{W}_Q \mathbf{x}_i$, Key: $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{x}_i$, Value: $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{x}_i$.



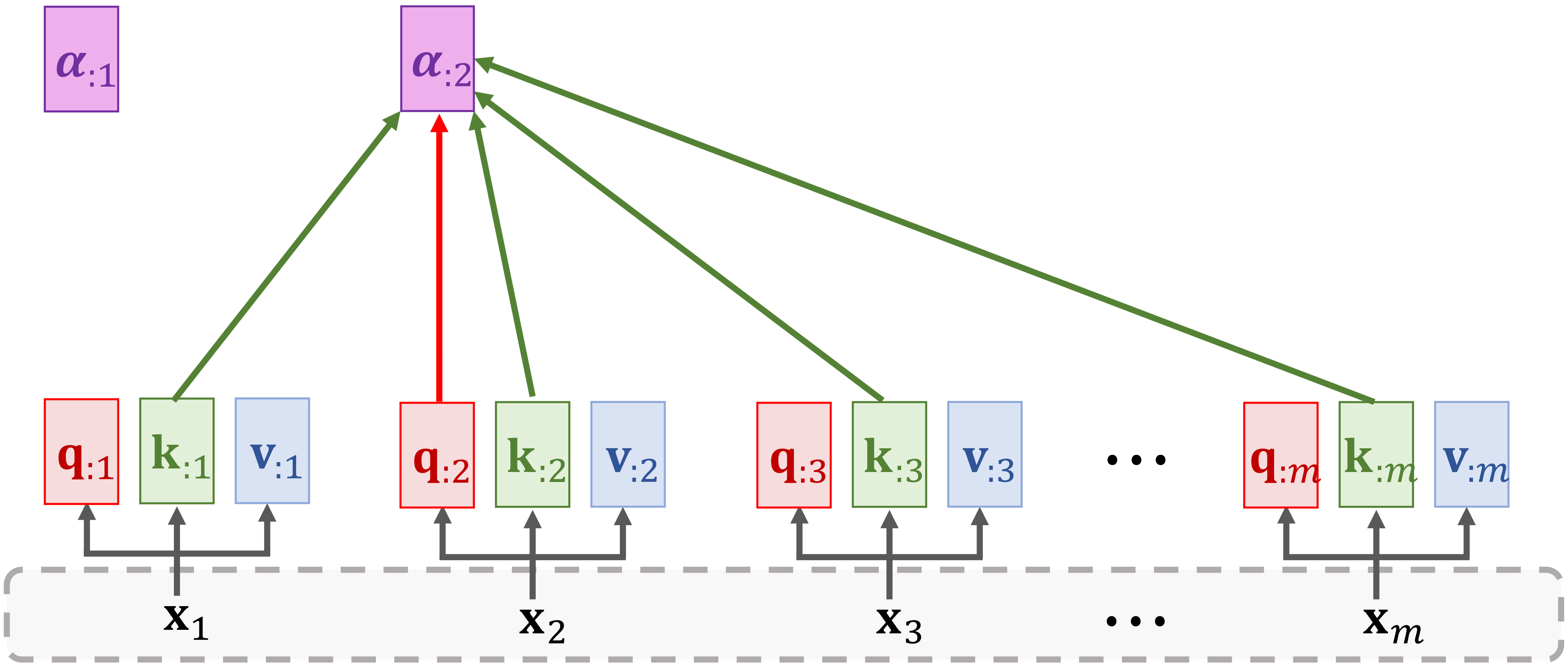
Self-Attention Layer

Weights: $\boldsymbol{\alpha}_{:j} = \mathrm{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m.$



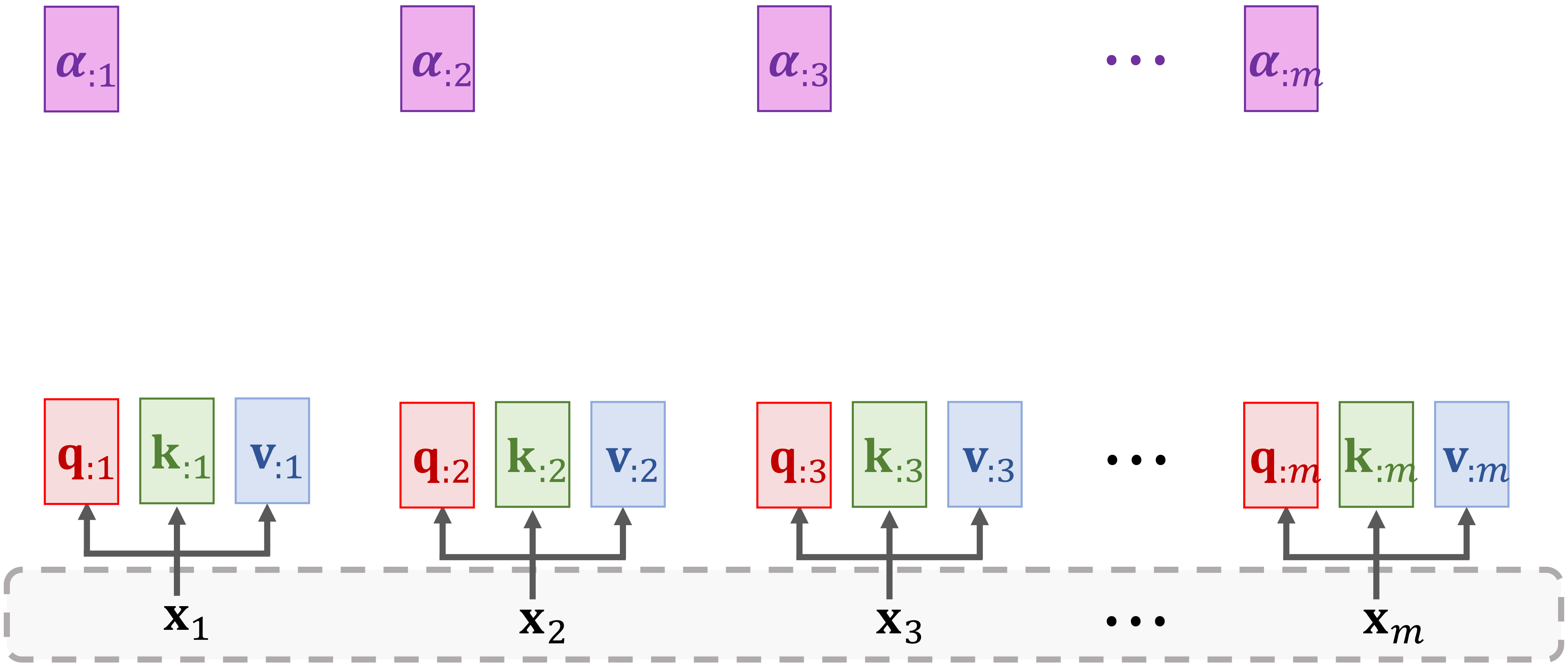
Self-Attention Layer

Weights: $\boldsymbol{\alpha}_{:j} = \mathrm{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m.$



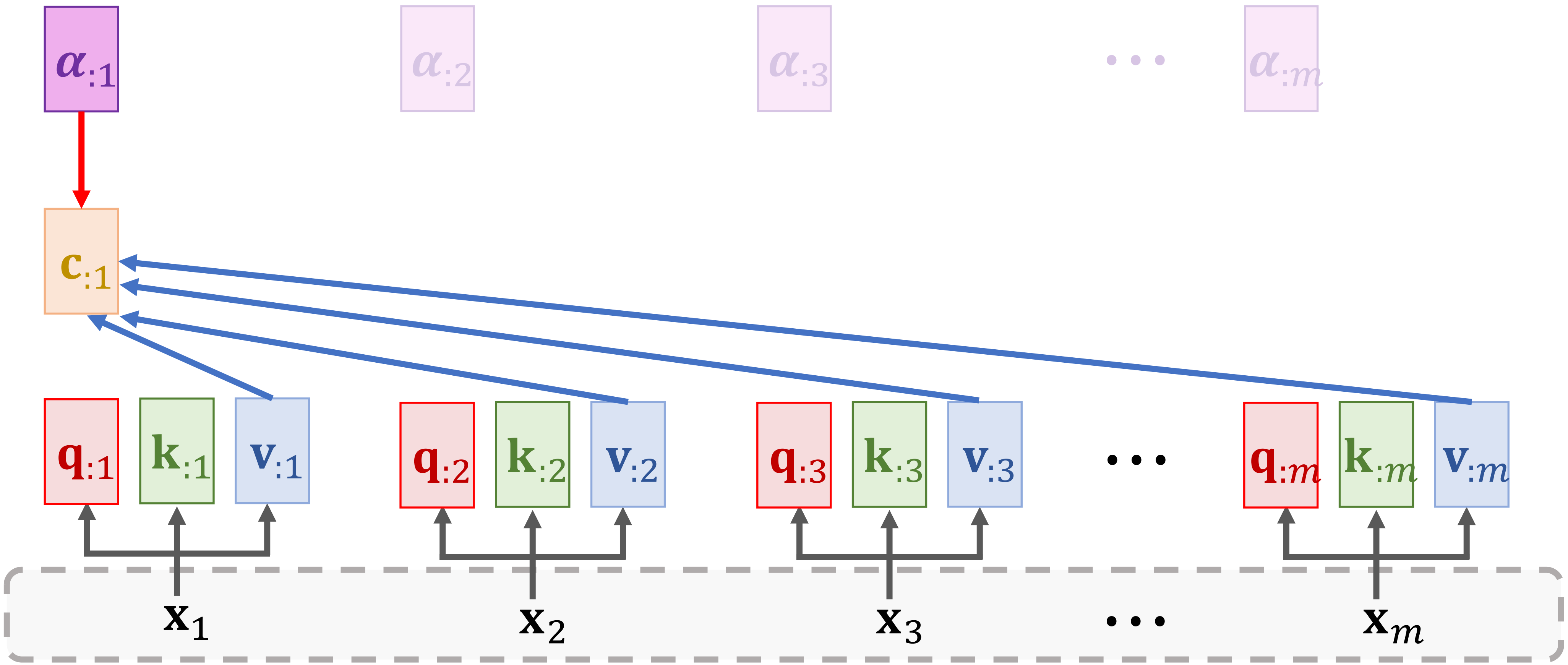
Self-Attention Layer

Weights: $\boldsymbol{\alpha}_{:j} = \mathrm{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m.$



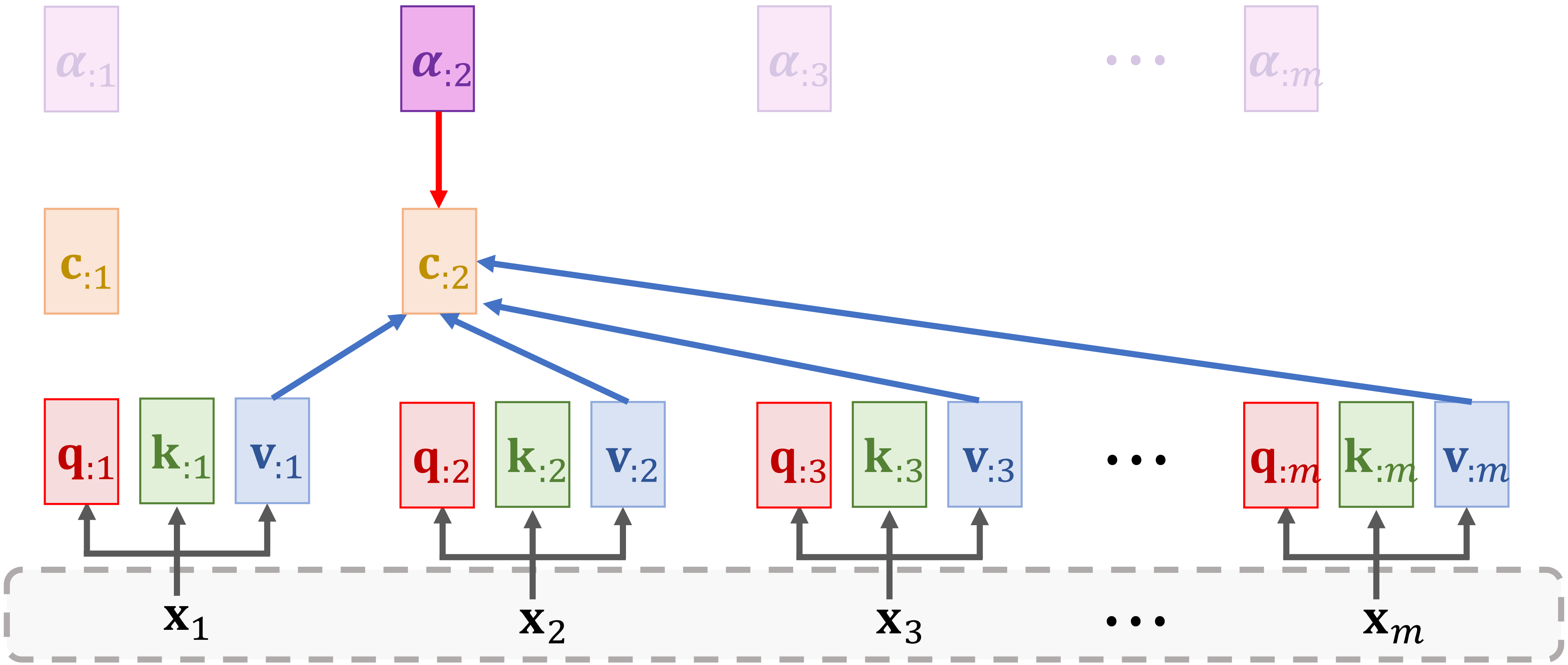
Self-Attention Layer

Context vector: $\mathbf{c}_{:1} = \alpha_{11}\mathbf{v}_{:1} + \cdots + \alpha_{m1}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:1}.$



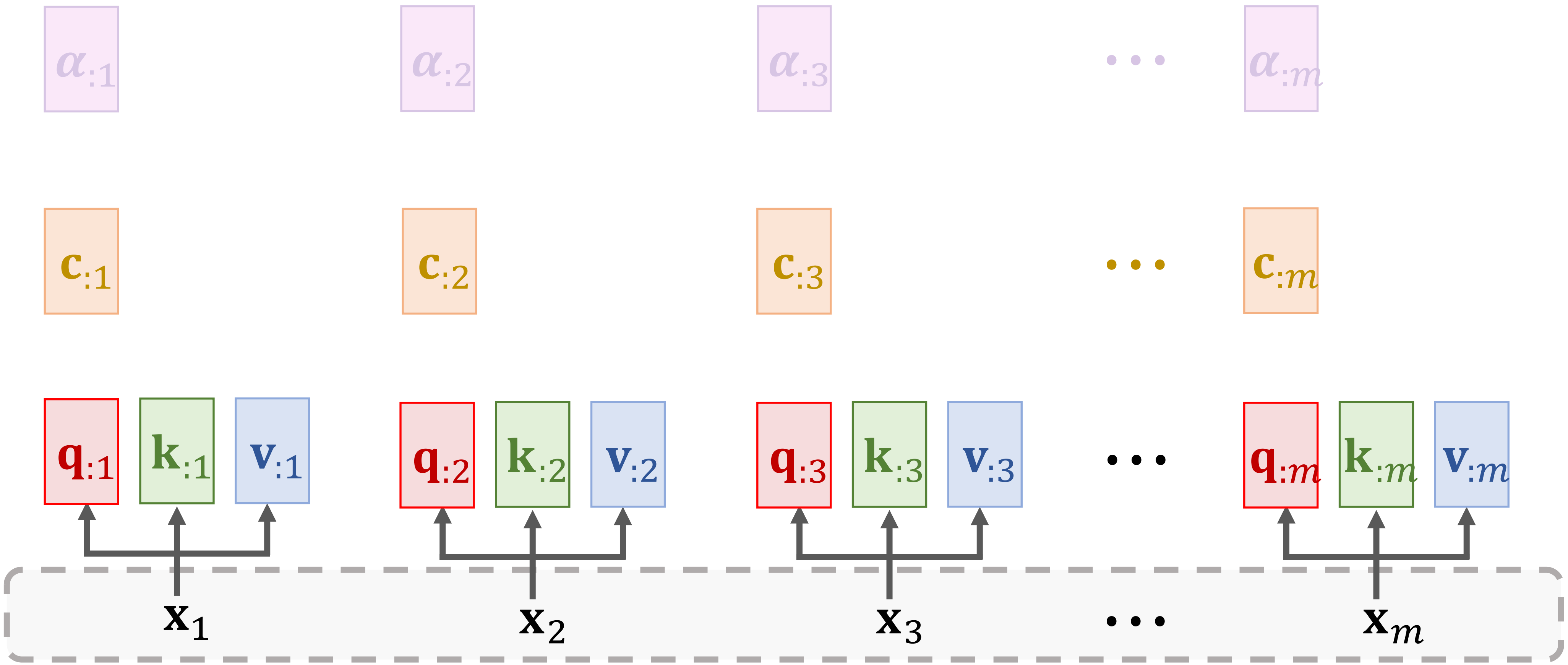
Self-Attention Layer

Context vector: $\mathbf{c}_{:2} = \alpha_{12}\mathbf{v}_{:1} + \cdots + \alpha_{m2}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:2}.$



Self-Attention Layer

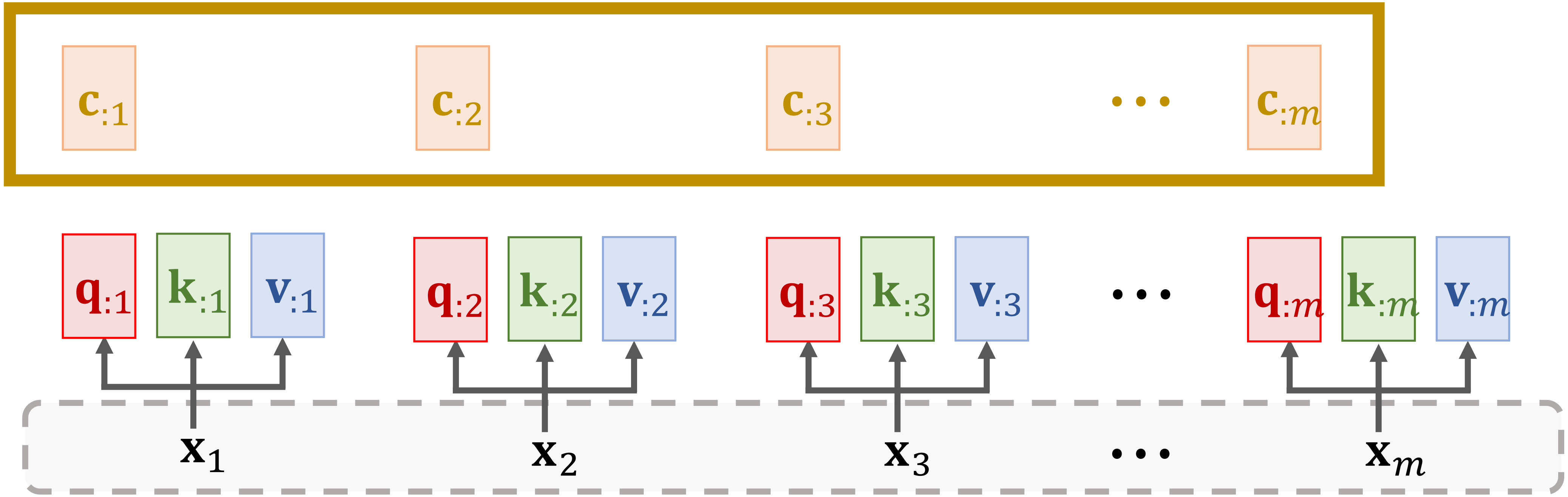
Context vector: $\mathbf{c}_{:j} = \alpha_{1j}\mathbf{v}_{:1} + \cdots + \alpha_{mj}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:j}$.



Self-Attention Layer

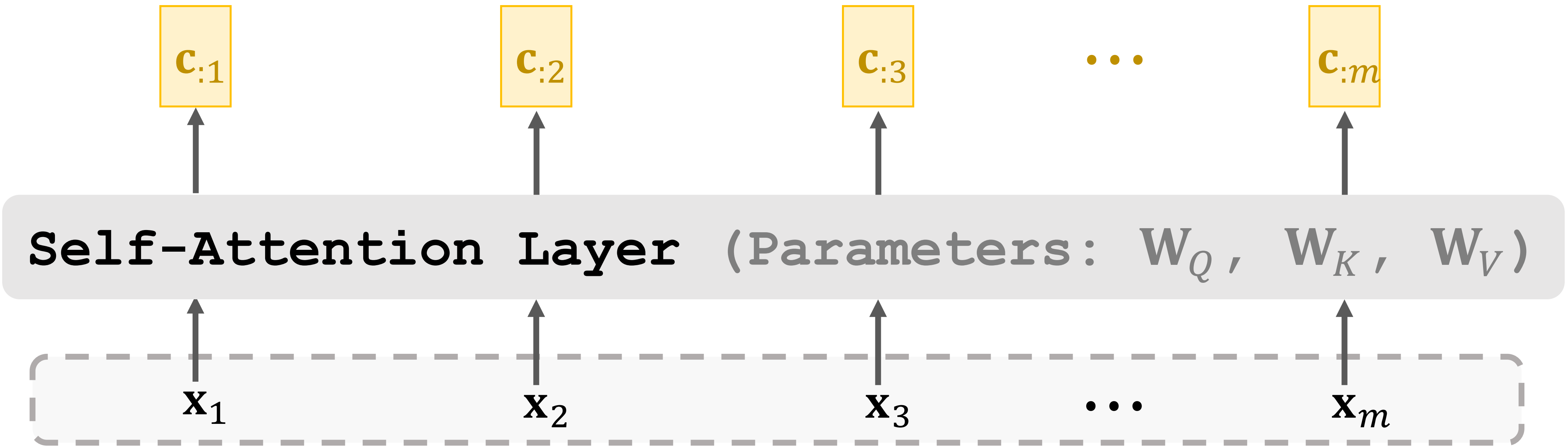
- Here, $\mathbf{c}_{:j} = \mathbf{V} \cdot \mathrm{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right)$.
- Thus, $\mathbf{c}_{:j}$ is a function of all the m vectors $\mathbf{x}_1, \cdots, \mathbf{x}_m$.

Output of self-attention layer:



Self-Attention Layer

- Self-attention layer: $\mathbf{C} = \mathrm{Attn}(\mathbf{X}, \mathbf{X})$.
  - Inputs: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]$.
  - Parameters: $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$.



Summary

Summary

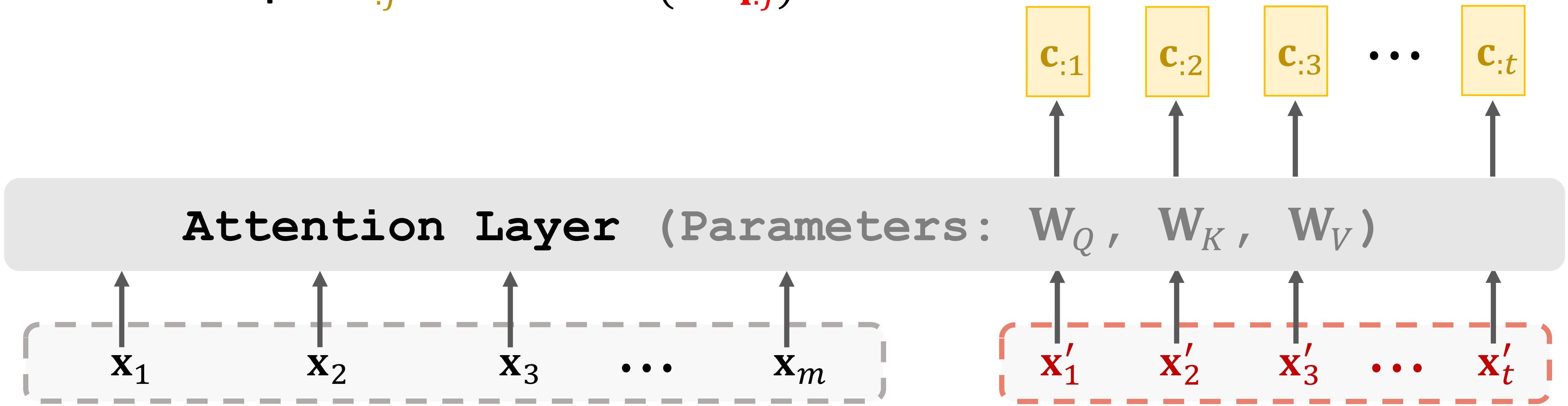
- Attention was originally developed for Seq2Seq RNN models [1].
- Self-attention: attention for all the RNN models (not necessarily Seq2Seq models [2].
- Attention can be used without RNN [3].
- We learned how to build attention layer and self-attention layer.

Reference:

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.
3. Vaswani et al. Attention Is All You Need. In *NIPS*, 2017.

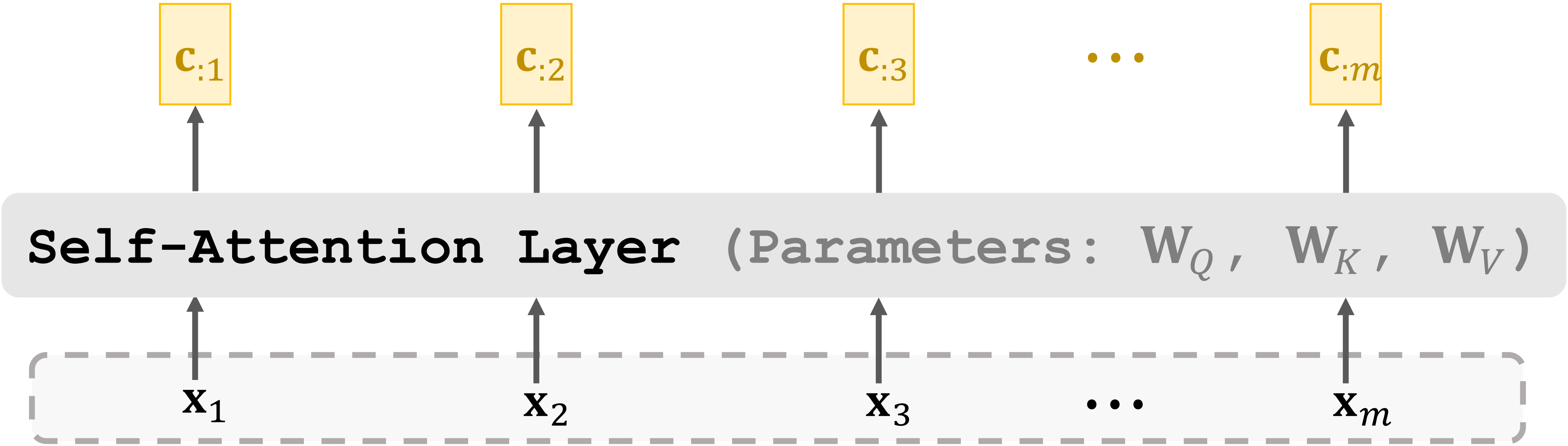
Attention Layer

- Attention layer: $\mathbf{C} = \mathrm{Attn}(\mathbf{X}, \mathbf{X}').$
  - **Query:** $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{x}'_j,$
  - **Key:** $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{x}_i,$
  - **Value:** $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{x}_i.$
  - **Output:** $\mathbf{c}_{:j} = \mathbf{V} \cdot \mathrm{Softmax}(\mathbf{K}^T \mathbf{q}_{:j}).$



Self-Attention Layer

- Attention layer: $\mathbf{C} = \mathrm{Attn}(\mathbf{X}, \mathbf{X}')$.
- Self-Attention layer: $\mathbf{C} = \mathrm{Attn}(\mathbf{X}, \mathbf{X})$.



Thank you!