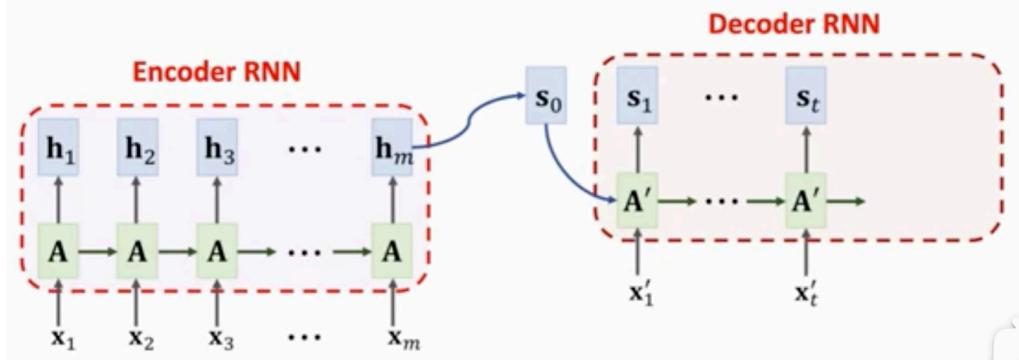


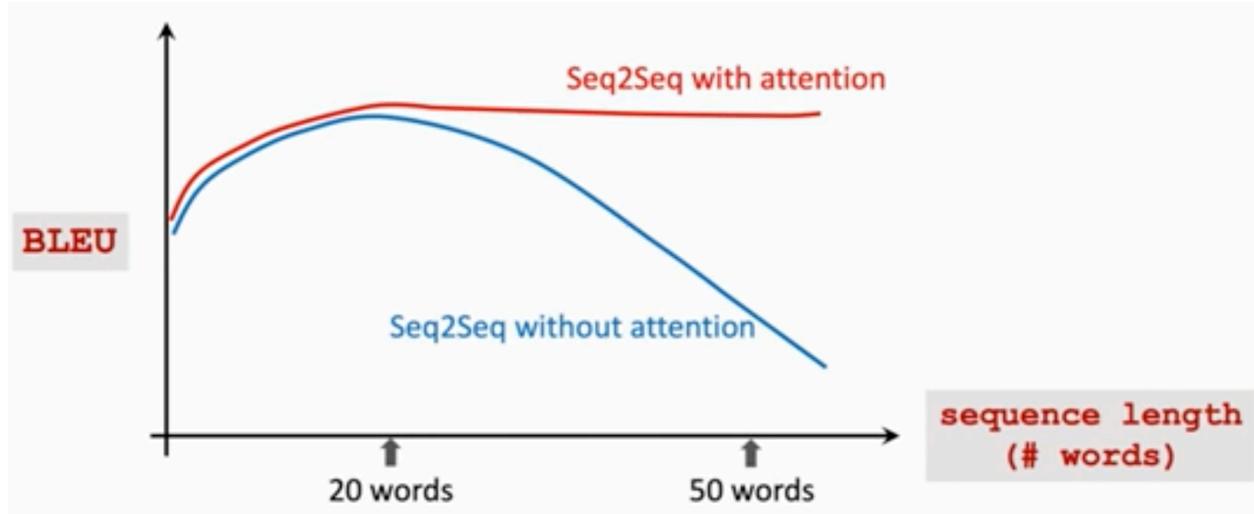
Attention

Seq2Seq Model

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



缺点：RNN模型在处理较长序列时会遗忘



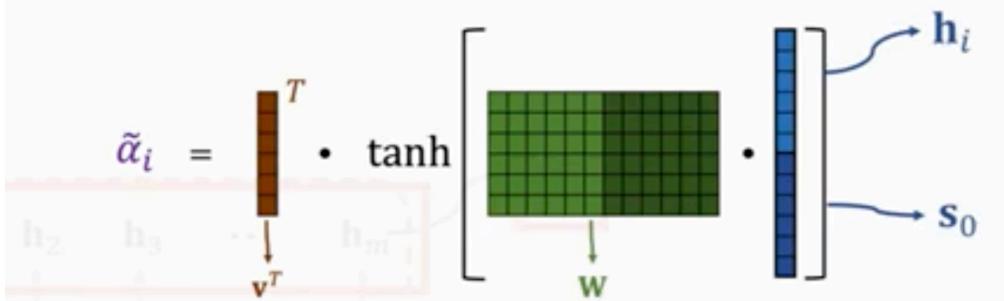
- Attention tremendously improves Seq2Seq model.
- With Attention, Seq2Seq model does not forget source input.
- With Attention, the decoder knows where to focus.
- Downside: much more computation.

计算相关性： $Weight : \alpha_i = align(h_i, s_0)$ 每个输入都计算一个 α , 所有 α 相加得到1

h_i 是每个状态积累的不同信息

最后一个 h_m 积累了所有信息 并将其作为decoder RNN的输入 s_0

Option 1 (used in the original paper):



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

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

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

1. Linear maps:

- $k_i = W_K \cdot h_i$, for $i = 1$ to m .
- $q_0 = W_Q \cdot s_0$.

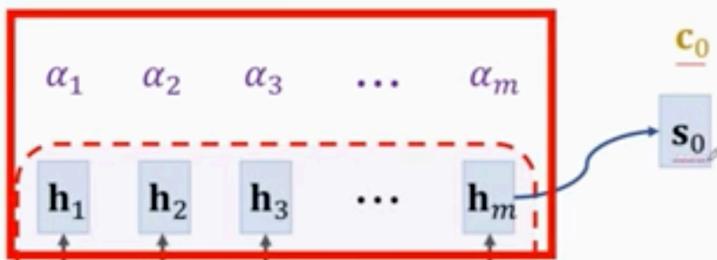
2. Inner product:

- $\tilde{\alpha}_i = k_i^T q_0$, for $i = 1$ to m .

3. Normalization:

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

Context vector: $c_0 = \alpha_1 h_1 + \dots + \alpha_m h_m$.

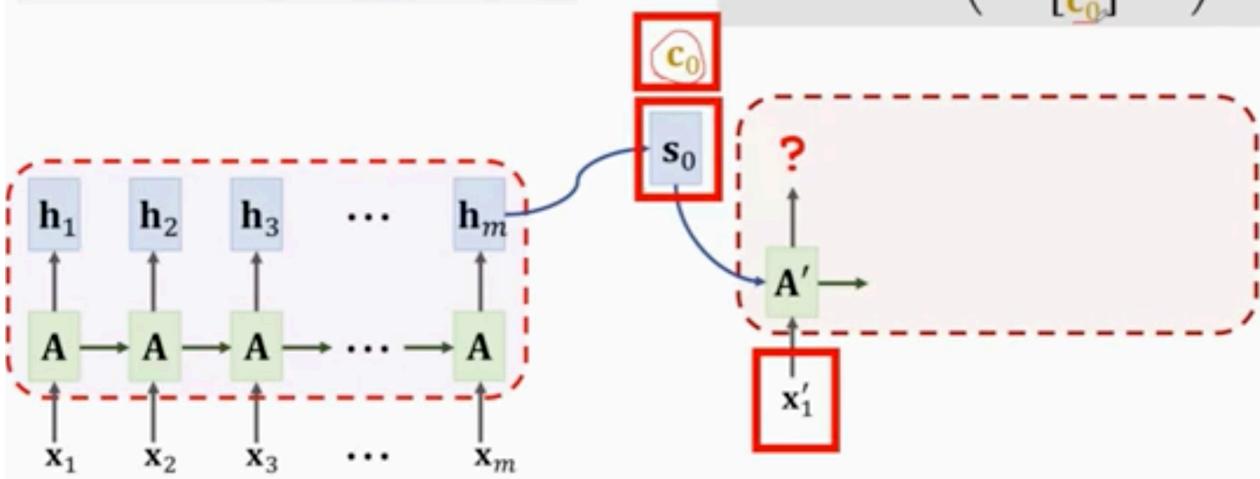


SimpleRNN:

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

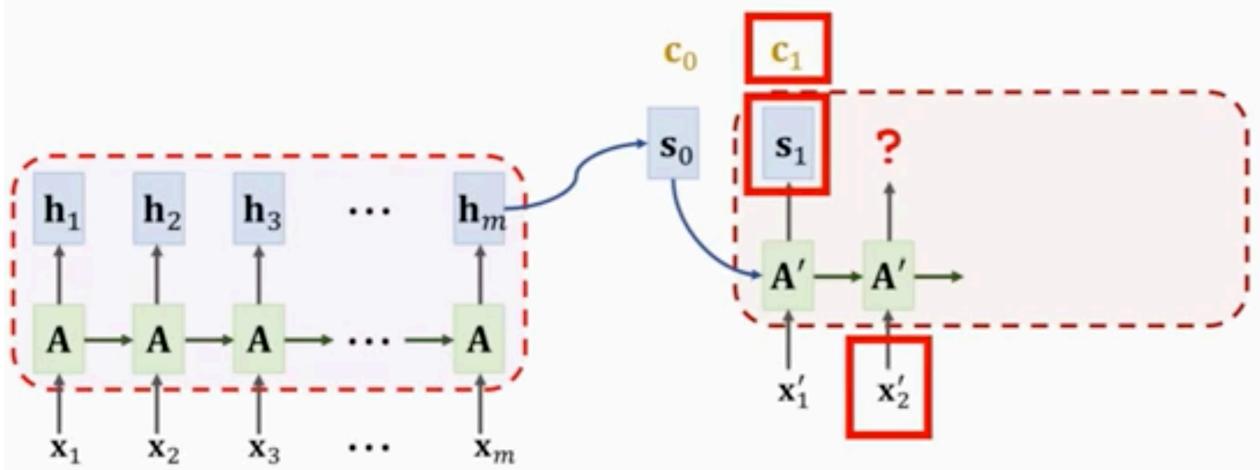
SimpleRNN + Attention:

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

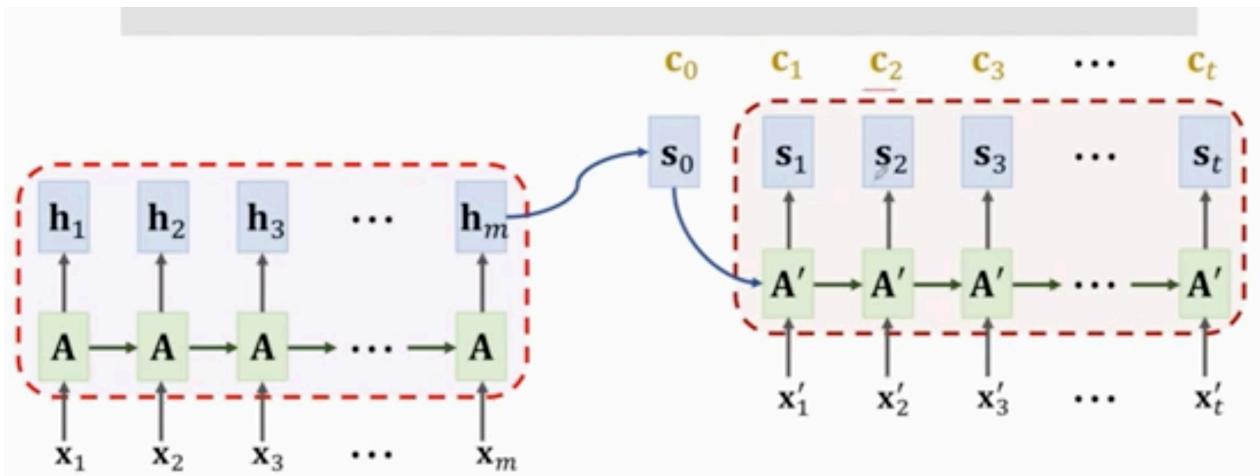


更新每一个decoder中的 s_i 状态都需要重新计算一次相关性

$$s_2 = \tanh \left(A' \cdot \begin{bmatrix} x'_2 \\ s_1 \\ c_1 \end{bmatrix} + b \right)$$



x'_i 是前一个状态的输出 x'_1 是起始符号



- 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

SimpleRNN:

$$h_1 = \tanh(A \cdot \begin{bmatrix} x_1 \\ h_0 \end{bmatrix} + b)$$

c_0

SimpleRNN + Self-Attention:

$$h_1 = \tanh(A \cdot \begin{bmatrix} x_1 \\ c_0 \end{bmatrix} + b)$$

h_0

?

A

x_1

初始状态 c_0 和 h_0 都是全零向量， A 是参数矩阵

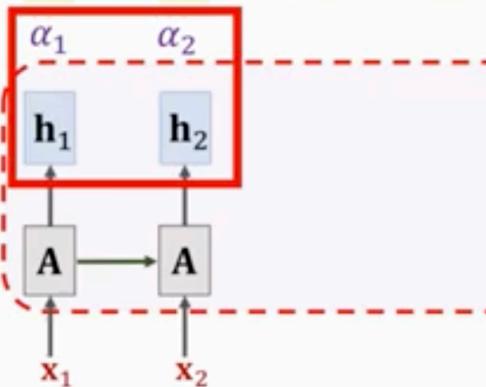
Weights: $\underline{\alpha_i} = \text{align}(\mathbf{h}_i, \mathbf{h}_2)$.

c_1

?



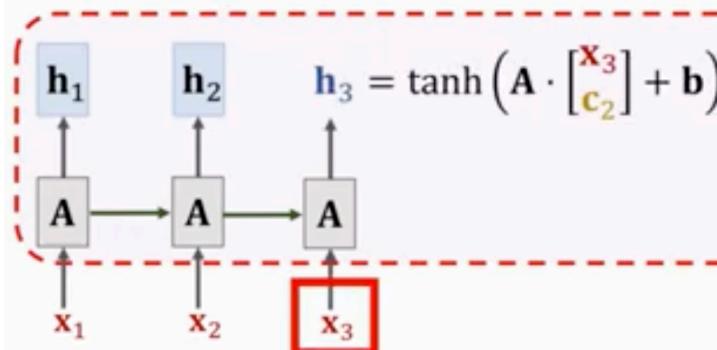
$$c_1 \quad c_2 = \alpha_1 h_1 + \alpha_2 h_2.$$



c_1

c_2

$$h_3 = \tanh(A \cdot [x_3] + b)$$



计算流程

- 首先计算 c_i (先求相似度得到 α 再加权平均)
- 再求得 h_i

Summary

- With self-attention, RNN is less likely to forget.
- Pay attention to the context relevant to the new input.

The
The FBI
The FBI is
The FBI is chasing
The FBI is chasing a
The FBI is chasing a criminal
The FBI is chasing a criminal on
The FBI is chasing a criminal on the
The FBI is chasing a criminal on the run
The FBI is chasing a criminal on the run .

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

- self-attention可以用矩阵运算快速得到相似度
- $E * E^T$ QKV (query、key、value) 在self里面都是一样的

Attention is All You Need

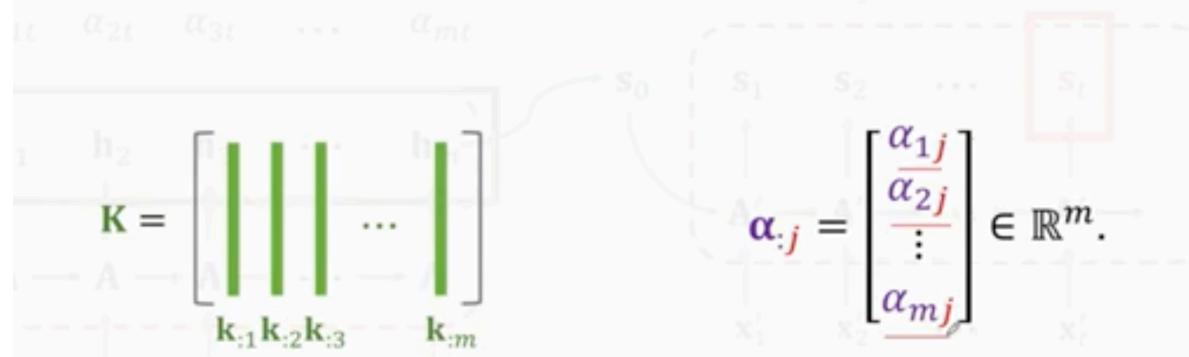
即 Transformer

- Transformer is a seq2seq model.
- Transformer is not RNN.
- Purely based on attention and dense layers. 只有Attention和全连接层
- Higher accuracy than RNNs on large datasets.

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: $\underline{\alpha_{:j}} = \underline{\text{Softmax}(\mathbf{K}^T \mathbf{q}_{:j})} \in \mathbb{R}^m$.



α_{ij} 中的元素都介于 0-1 之间，并且相加和为 1

- **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.)

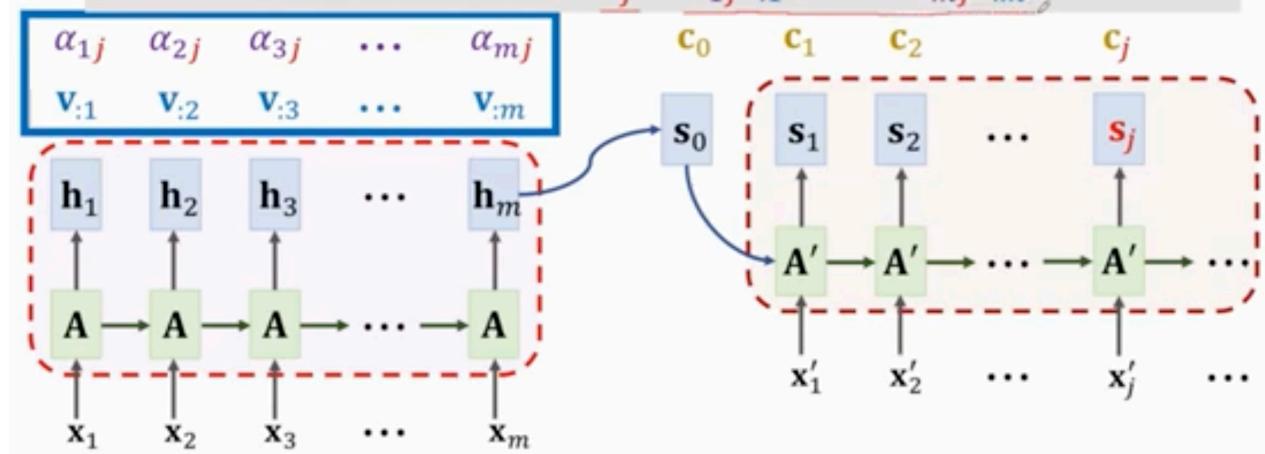
α_{ij} 说明的是 Query 和 Key 的匹配程度

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: $\alpha_{ij} = \text{Softmax}(\mathbf{K}^T \mathbf{q}_{:j}) \in \mathbb{R}^m$.

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

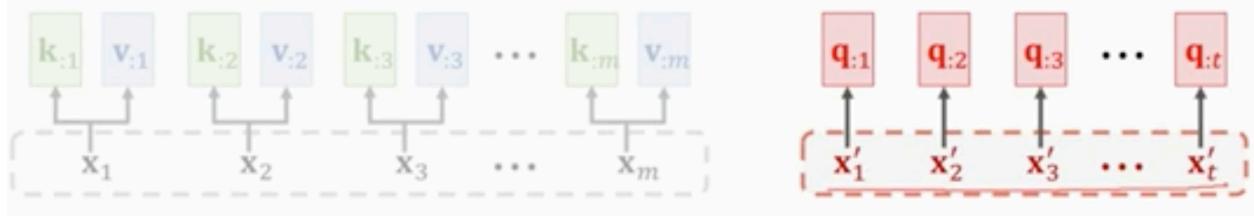


Attention Without RNN

具体步骤

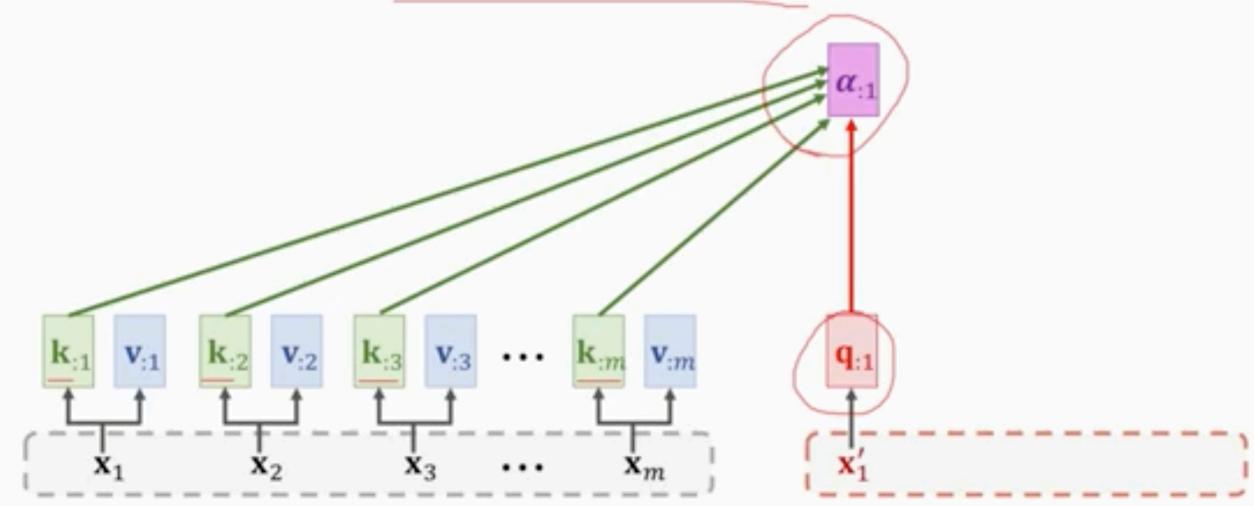
- 首先分别计算K V Q矩阵

- Keys and values are based on encoder's inputs x_1, x_2, \dots, x_m .
- Key: $k_{:i} = \underline{W_K} x_i$.
- Value: $v_{:i} = \underline{W_V} x_i$.
- Queries are based on decoder's inputs x'_1, x'_2, \dots, x'_t .
- Query: $\underline{q}_{:j} = \underline{W_Q} x'_j$.



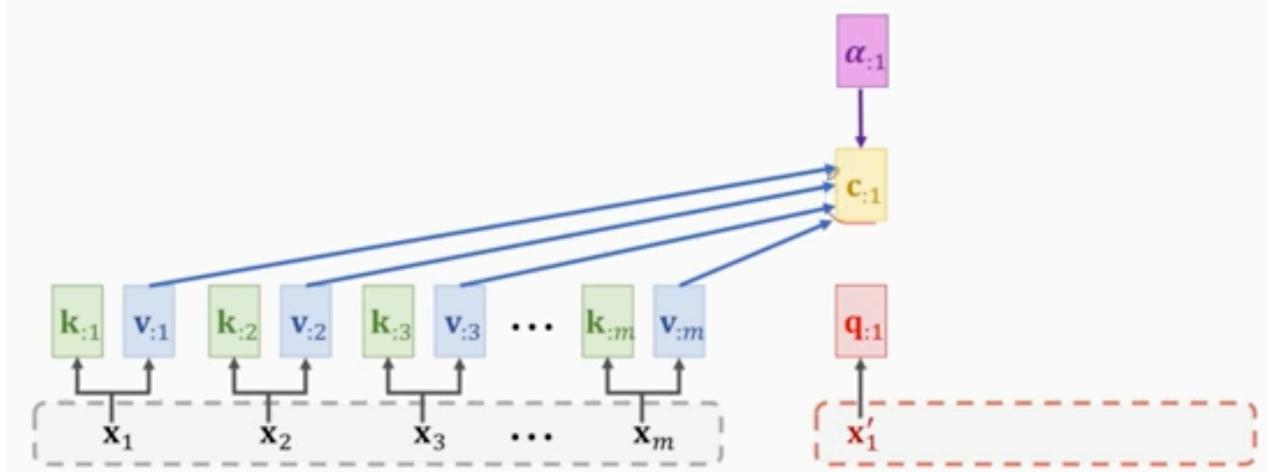
- 计算 $\alpha_{:1}$

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

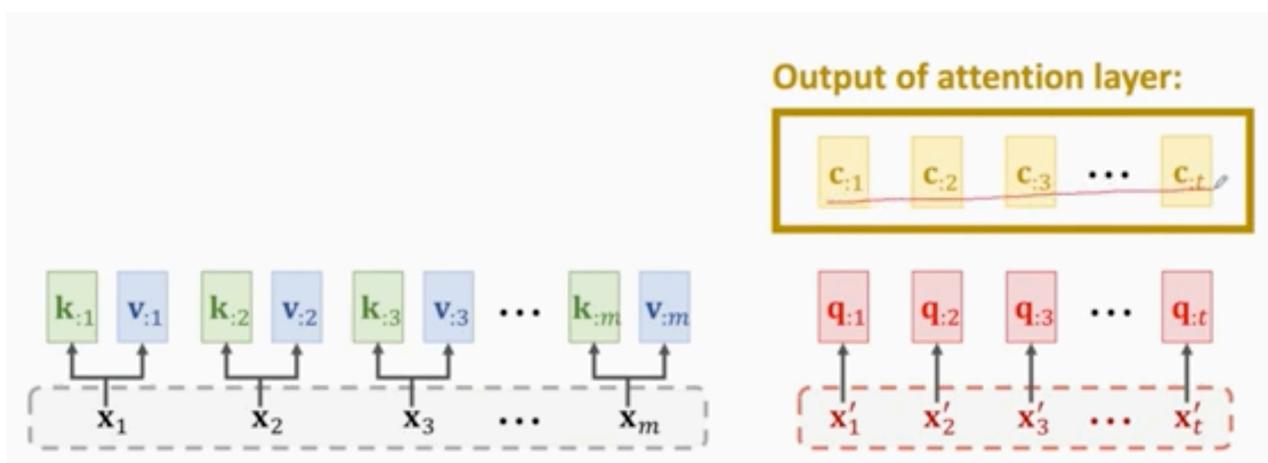


- 计算 $c_{:1}$

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

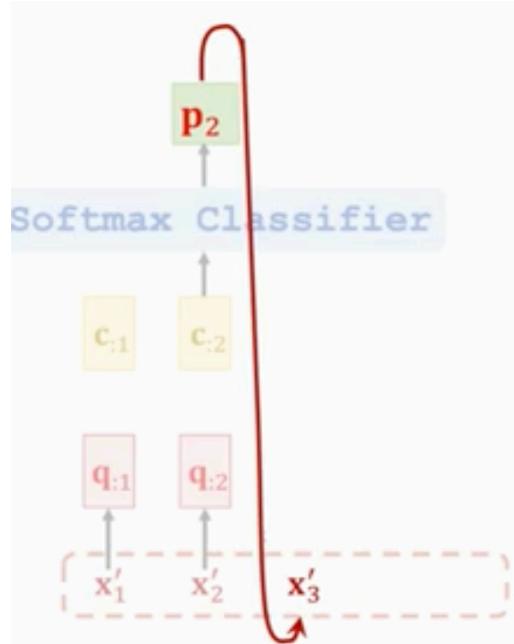


- 得到最终的 c 向量



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

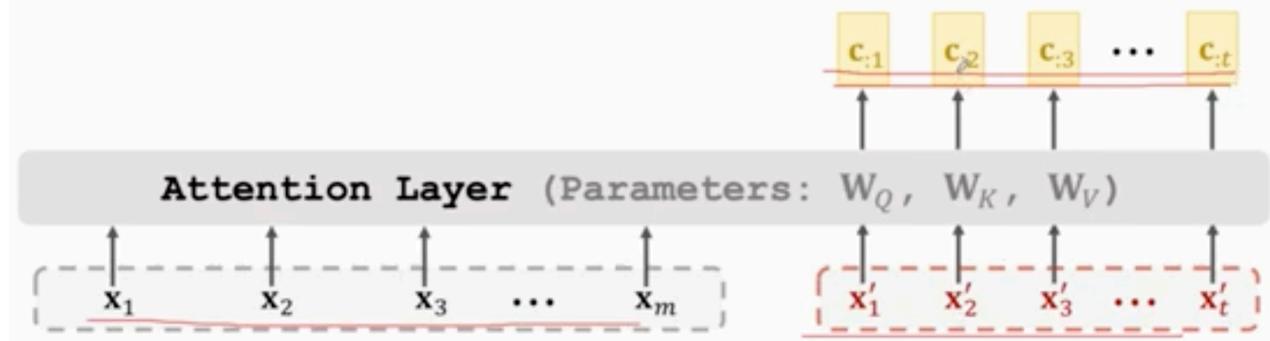
- 将 C 作为特征向量 这样的结构与RNN类似



Attention Layer的最终结构

Attention Layer

- Attention layer: $\text{Attn}(X, X')$.
 - Encoder's inputs: $X = [x_1, x_2, \dots, x_m]$.
 - Decoder's inputs: $X' = [x'_1, x'_2, \dots, x'_t]$.
 - Parameters: W_Q, W_K, W_V .



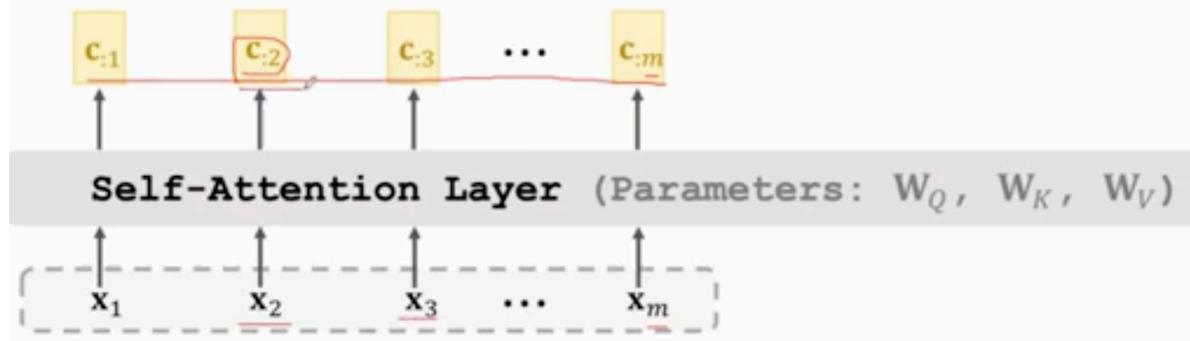
Self-Attention Without RNN

Self-Attention Layer

- Self-attention layer: $C = \text{Attn}(X, X)$.

- RNN's inputs: $X = [x_1, x_2, \dots, x_m]$.

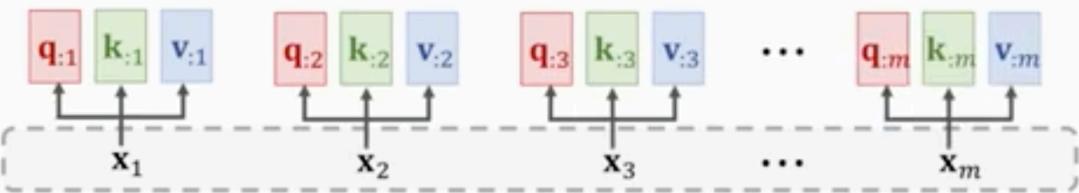
- Parameters: W_Q, W_K, W_V .



具体步骤

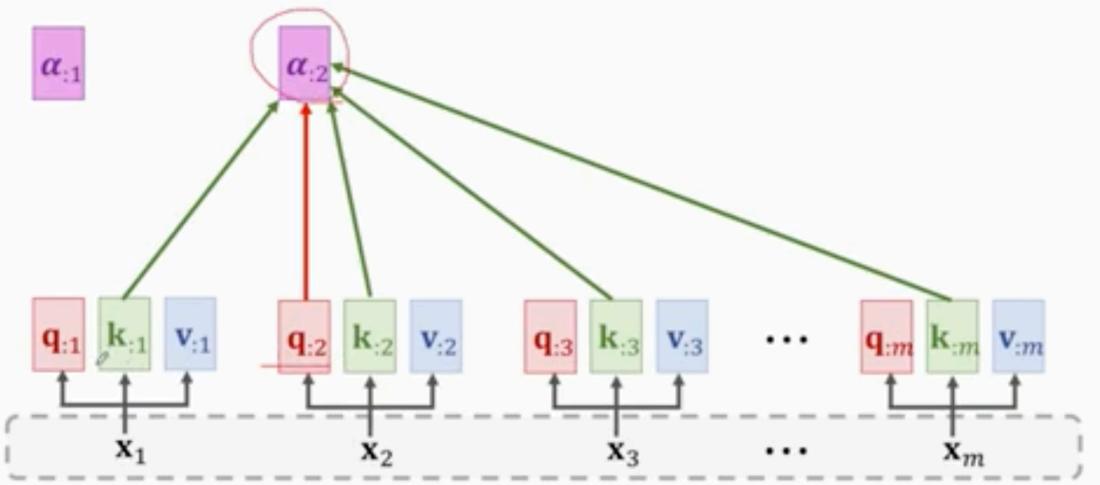
- 分别计算Q、K、V

$$\text{Query: } q_{:i} = W_Q x_i, \quad \text{Key: } k_{:i} = W_K x_i, \quad \text{Value: } v_{:i} = W_V x_i.$$



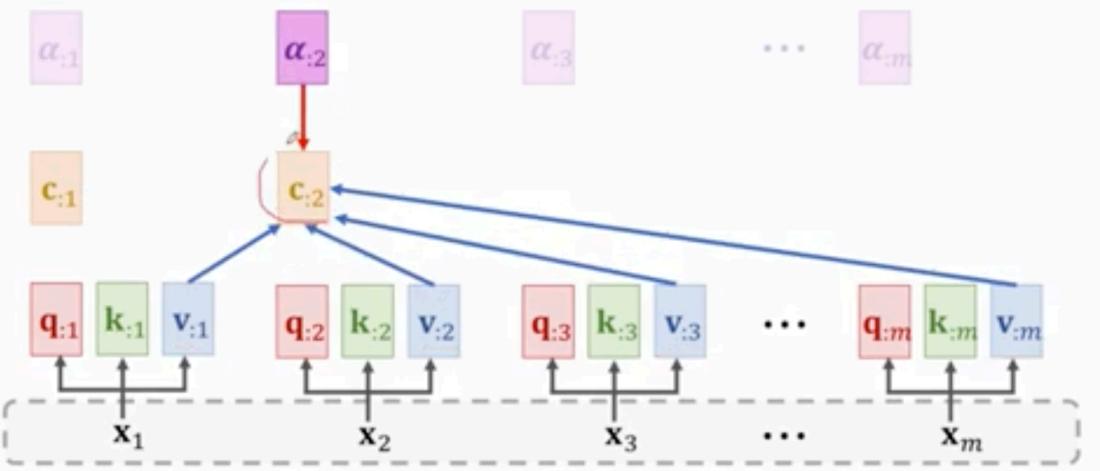
- 计算权重向量

Weights: $\alpha_{:j} = \text{Softmax}(\mathbf{K}^T \mathbf{q}_{:j}) \in \mathbb{R}^m$.



- 计算 C

Context vector: $c_{:2} = \alpha_{12}v_{:1} + \dots + \alpha_{m2}v_{:m} = \mathbf{V}\alpha_{:2}$.



- 总结

- Here, $c_{:j} = \underline{\mathbf{V}} \cdot \text{Softmax}(\underline{\mathbf{K}}^T \underline{\mathbf{q}}_{:j})$.
- Thus, $c_{:j}$ is a function of all the m vectors x_1, \dots, x_m .

Output of self-attention layer:

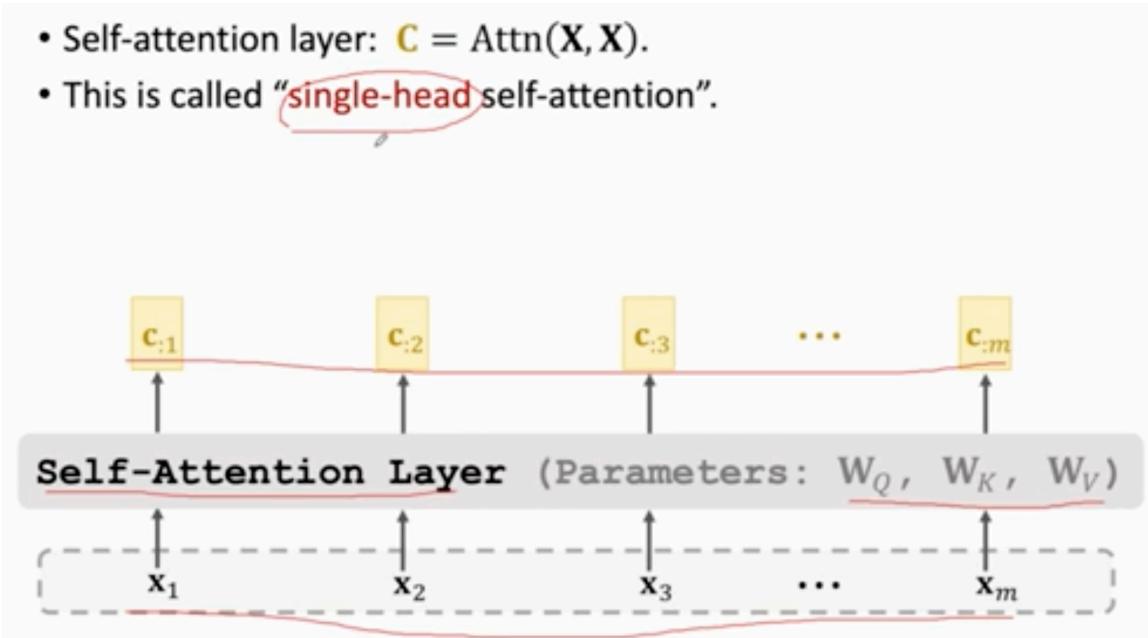


Summary

- Attention was originally developed for Seq2Seq RNN models.
- Self-attention: attention for all the RNN models (no necessarily Seq2Seq models).
- Attention can be used without RNN.

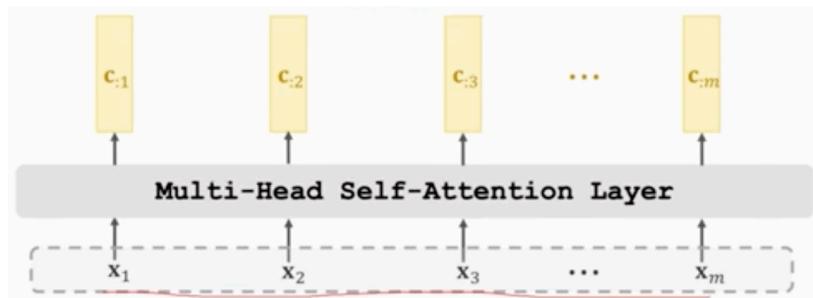
Transformer Model (From Shallow to Deep)

- Self-attention layer: $C = \text{Attn}(X, X)$.
- This is called “single-head self-attention”.

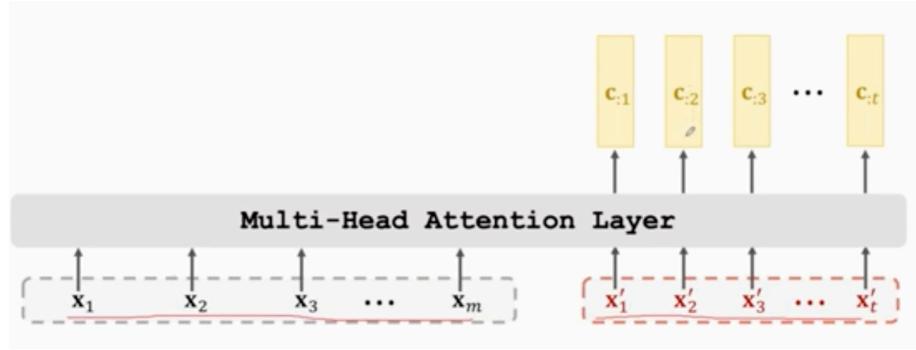


- Using 1 single-head self-attentions (which do not share parameters).
 - A single-head self-attention has 3 parameter matrices: W_Q, W_K, W_V
 - Totally 3l parameters matrices.
 - Concatenating outputs of single-head self-attentions.
 - Suppose single-head self-attentions' outputs are $d*m$ matrices.
 - Multi-head's output shape: $(ld)*m$.

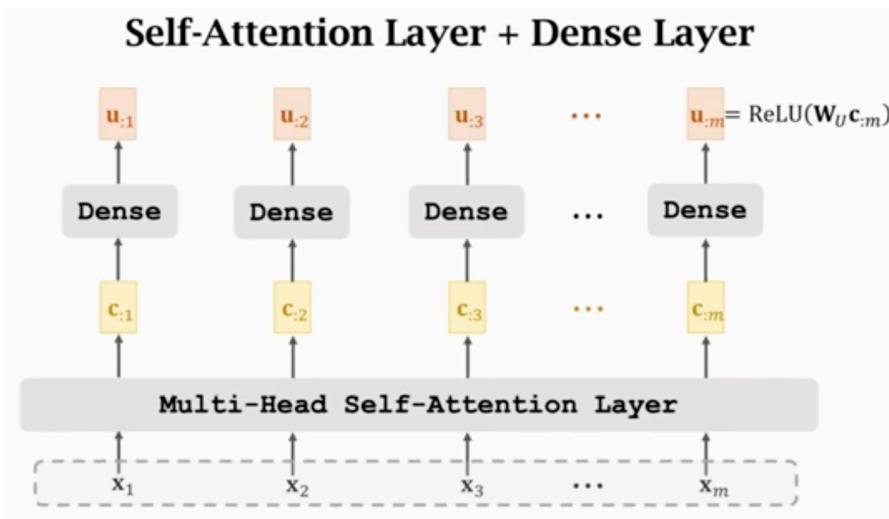
1. Multi-head Attention



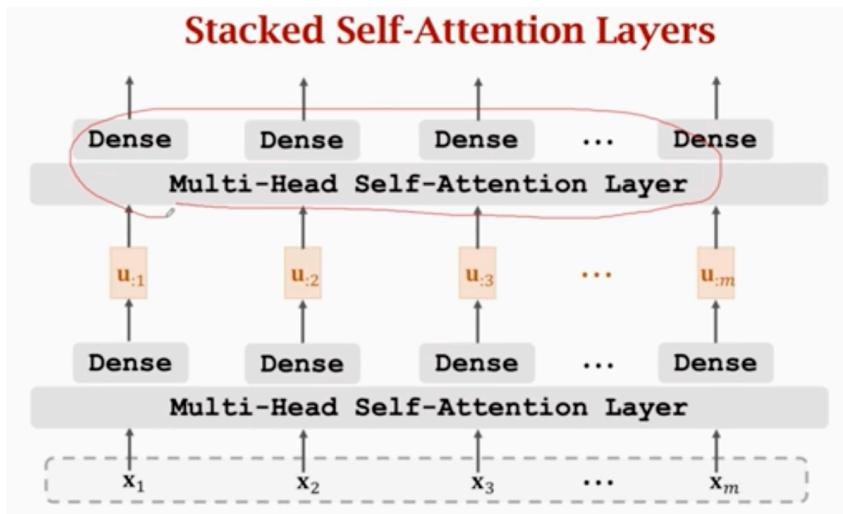
2. Multi-head Self-Attention



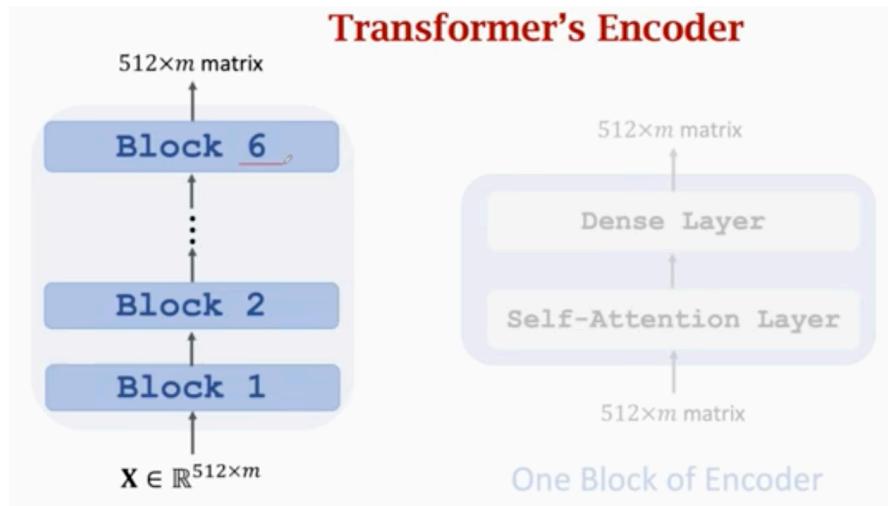
Stacked Self-Attention Layers



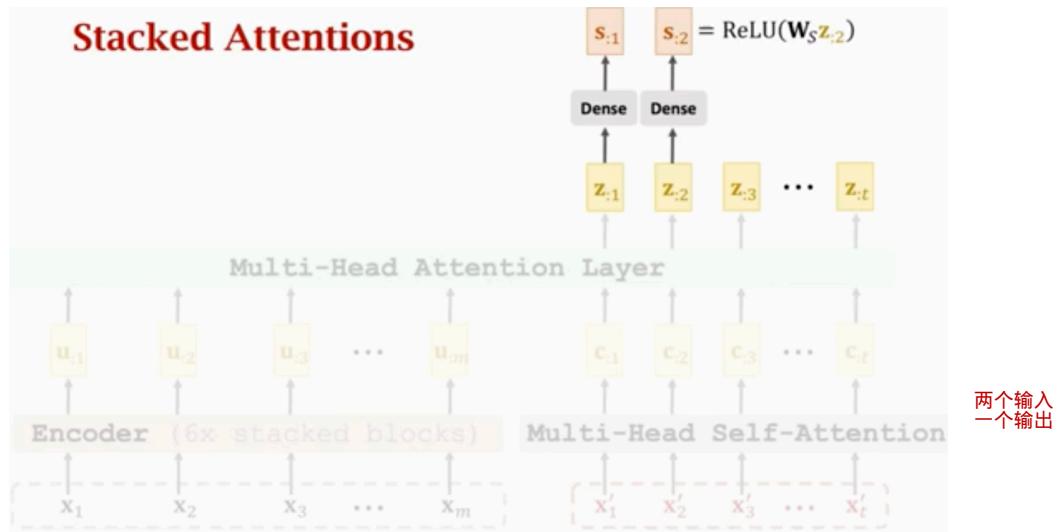
- Dense 表示全连接层 每一个 Dense 层的权重都完全相同
- 输出 u_i 依赖于 $x_i \dots x_m$, 改变任何一个 x 的值都会对 u_i 产生影响, 但是对 u_i 影响最大的还是 x_i



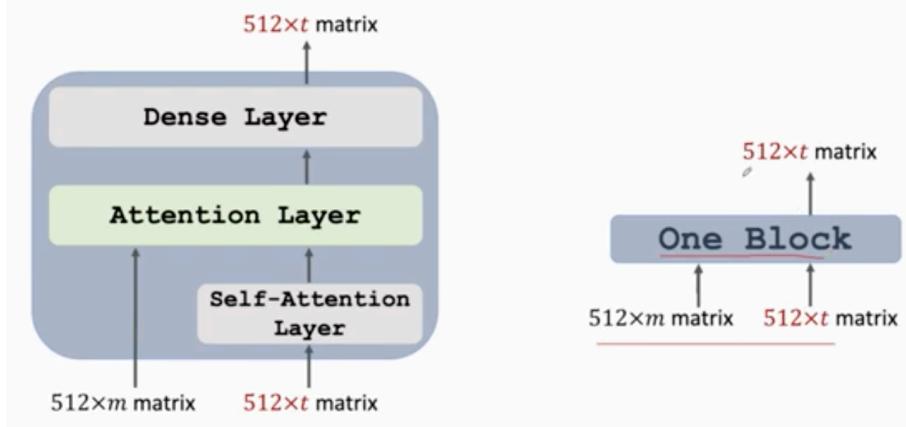
Transformer's Encoder



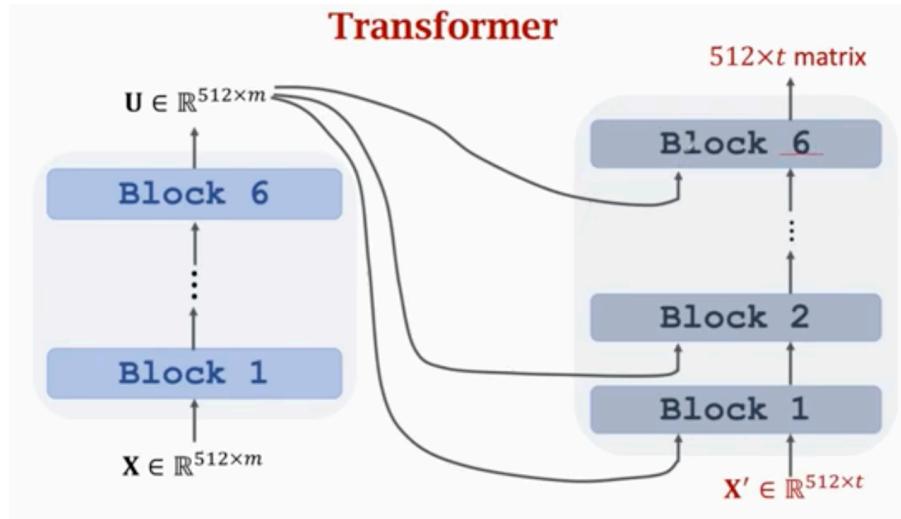
- block之间不共享参数
- Transformer's encoder contains 6 stacked blocks.
- 1block \approx 1 multi-head attention layer + 1 dense layer.



Transformer's Decoder : One Block



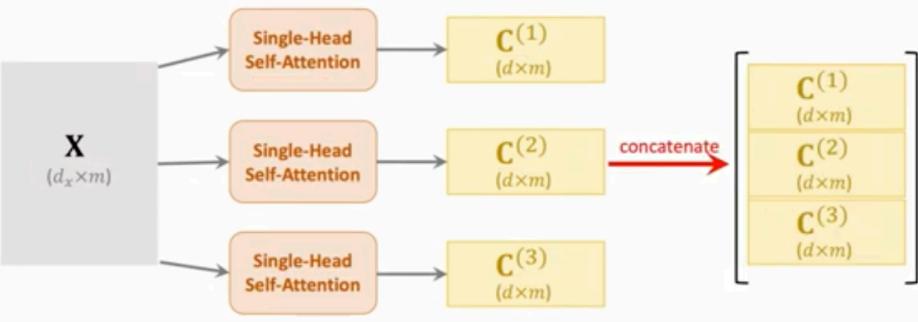
Put Everything Together



Summary

- From Single-Head to Multi-head

• Single-head self-attention can be combined to form a multi-head self-attention.



- Encoder Network of Transformer

• 1 encoder block \approx multi-head self-attention + dense.
 • Input shape: $512 \times m$.
 • Output shape: $512 \times m$.

• Encoder network is a stack of 6 such blocks.

- Decoder Network of Transformer

• 1 decoder block \approx multi-head self-attention + multi-head attention + dense.
 • Input shape: $(512 \times m, 512 \times t)$.
 • Output shape: $512 \times t$.
 • Decoder network is a stack of 6 such blocks.

- Transformer Model

- Transformer is Seq2Seq model; it has an encoder and a decoder.
- Transformer model is not RNN.
- Transformer is based on attention and self-attention.
- Transformer outperforms all the state-of-the-art RNN models.