

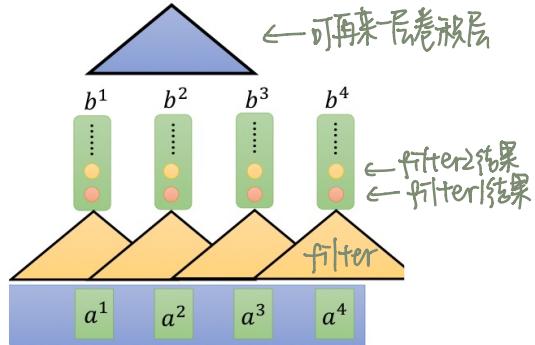
## ▷ Transformer

KEYWORDS: Self-Attention, Seq2Seq, Encoder-Decoder, Transformer

RNN处理有向序列难以并行化 → 希望用CNN(逐句)替代RNN

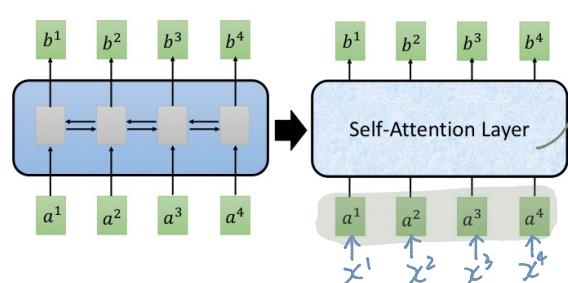
→ 用self-attention替代RNN

### ① CNN:



每个filter对应的输出均可并行.  
但视野不广.要看较大范围需要多个卷积层.

### ② Self-attention 自注意力



每个输出 $b^i$ 均可看到所有输入,且可并行计算.  
所有可用RNN做的均可用self-attention做.

→ 三个向量:

①  $q$ : query, 用来match别人  
 $q^i = W^q a^i$

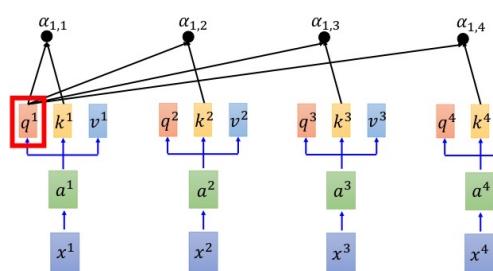
②  $k$ : key, 被match  
 $k^i = W^k a^i$

③  $v$ : 被抽取的信息

$$v^i = W^v a^i$$

即为两个向量的点积 (匹配度)

用向量 $q$ 对所有 $k$ 做attention: 有不同方式, 如 scaled dot-product attention:

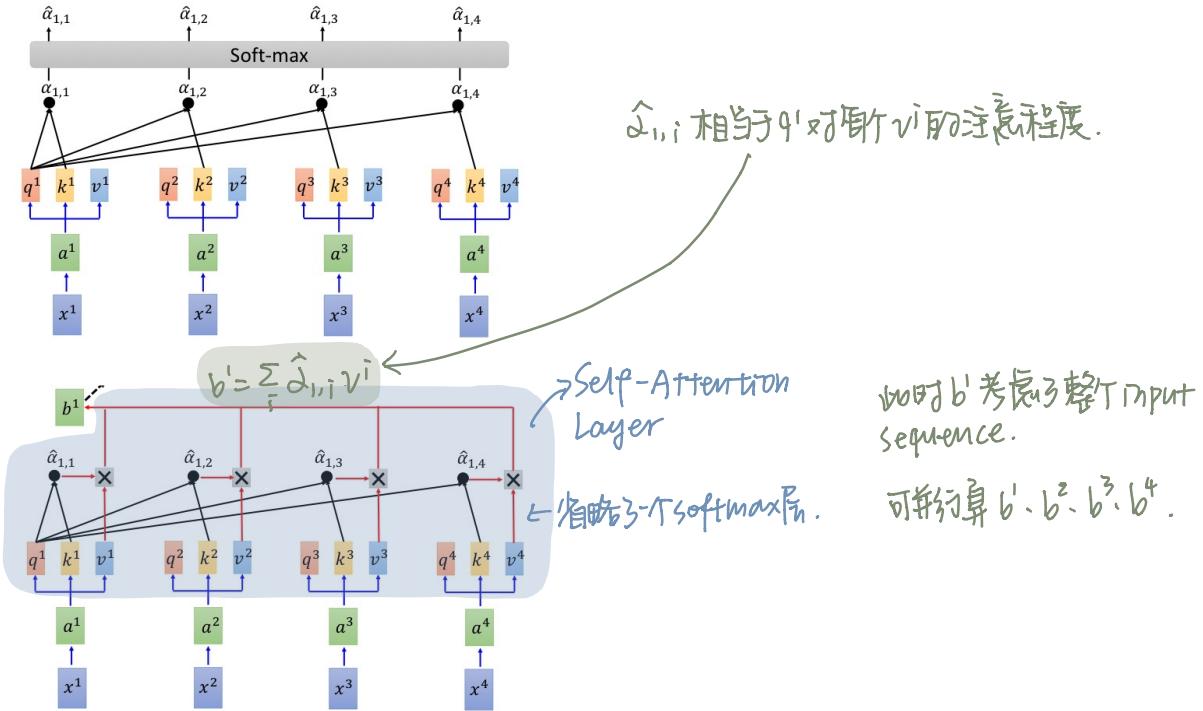


$$\alpha_{1,i} = q^1 \cdot k^i / \sqrt{d}, d \text{ 为 } q \text{ 和 } k \text{ 的维度.}$$

(为何除以  $\sqrt{d}$ ? 因为  $d$ ↑点积↑. 除以  $\sqrt{d}$  平衡了.)  
(也有加性 (additive) attention).

← 如图只画出了 $q^1$ 对所有的attention.

将  $\alpha_{1,i}$  经 softmax 后： $\hat{\alpha}_{1,i} = \exp(\alpha_{1,i}) / \sum_j \exp(\alpha_{1,j})$



矩阵形式 (用 GPU 可加速矩阵乘法)

$$\left. \begin{array}{l} W^q a^i = q^i \\ W^k a^i = k^i \\ W^v a^i = v^i \end{array} \right\} \Rightarrow \begin{cases} W^q [a^1 \ a^2 \ a^3 \ a^4] = [q^1 \ q^2 \ q^3 \ q^4] \\ W^k [a^1 \ a^2 \ a^3 \ a^4] = [k^1 \ k^2 \ k^3 \ k^4] \\ W^v [a^1 \ a^2 \ a^3 \ a^4] = [v^1 \ v^2 \ v^3 \ v^4] \end{cases} \quad \begin{array}{l} W^q I = Q \\ W^k I = K \\ W^v I = V \end{array}$$

$$q^i \cdot k^j = \alpha_{j,i} \Rightarrow \text{值域}/\sqrt{d}$$

$$\begin{bmatrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{bmatrix}^T \cdot q^j = \begin{bmatrix} \alpha_{j,1} \\ \alpha_{j,2} \\ \alpha_{j,3} \\ \alpha_{j,4} \end{bmatrix}$$

$$K^T \begin{bmatrix} q^1 & q^2 & q^3 & q^4 \end{bmatrix} = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} \end{bmatrix} \xrightarrow{\text{Softmax (对角-列)}} \hat{A} = [\hat{\alpha}_1 \ \hat{\alpha}_2 \ \hat{\alpha}_3 \ \hat{\alpha}_4]$$

$$K^T Q = A \rightarrow \hat{A}$$

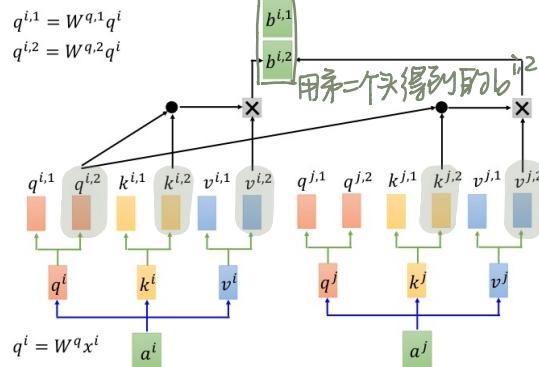
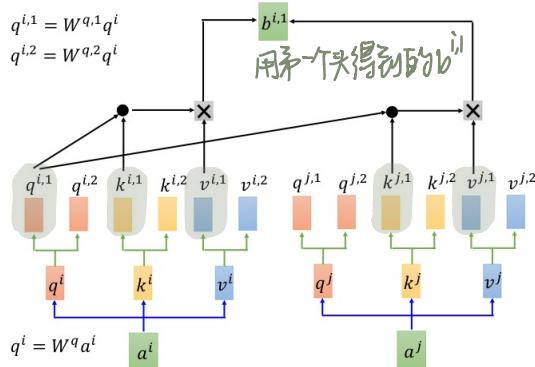
$$\sum_i \hat{a}_{ji} v^i = b^j \Rightarrow \begin{bmatrix} v^1 & v^2 & v^3 & v^4 \end{bmatrix} V \hat{A} = b^j$$

$$V \begin{bmatrix} \hat{d}_1 & \hat{d}_2 & \hat{d}_3 & \hat{d}_4 \end{bmatrix} = \begin{bmatrix} b^1 & b^2 & b^3 & b^4 \end{bmatrix} \quad V \hat{A} = 0$$

总结：由  $W^q, W^k, W^v$  和  $a$  可得  $Q, K, V$ ，由  $K^T$  可得  $A$  进而得  $\hat{A}$ ，由  $V$  和  $\hat{A}$  得  $O$ 。  
 ↑ 转输入  
 ↑ 转输出

### △ Multi-head Self-attention 多头自注意力

e.g. 2头：将  $q, k, v$  分成2个。

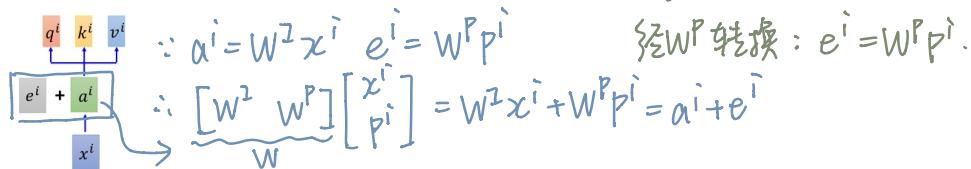


不同 head 注意于不同信息。

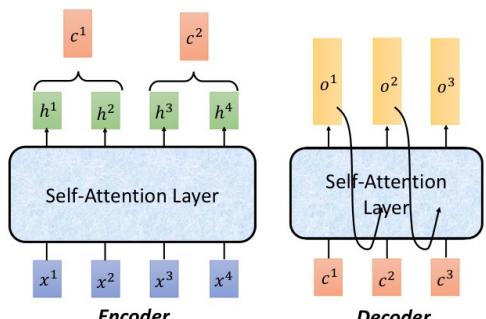
### 位置编码 Positional Encoding

Self-Attention 中没有保存位置信息。

为每个位置指定一个位置向量 (unique). e<sup>i</sup>: e.g. one-hot vector  $p^i = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$  和维。



## Seq2Seq with Attention



Seq2Seq: 转输入序列、输出序列的模型

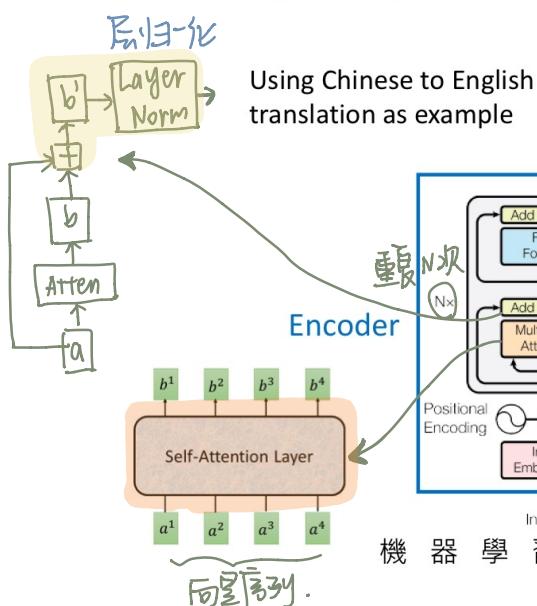
Encoder-Decoder: 一个框架  
RNN/CNN/RNN(BiRNN)/GRU/LSTM等均可  
Seq2Seq是Encoder-Decoder的一个具体模型.

Transformer: 用Attention构造Encoder和Decoder.

(Auto-encoder: 也使用了encoder和decoder,  
但是自成序列的, 即训练时希望输入、输出相同)

★ Transformer 基于Encoder-Decoder框架. 可用Seq2seq的地方均可用Transformer.

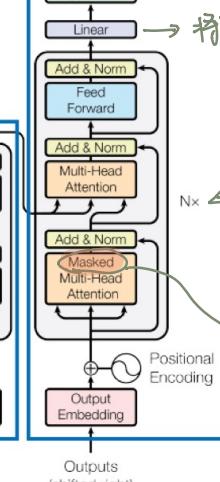
## Transformer



词的位置是每个单词的可能性(维度和单词表大小相同).

machine learning

将向量映射到单词表大小的维度.



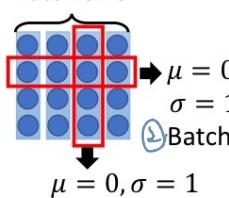
decoder先做一次attention  
再接encoder输出.

Decoder

翻译当前词时只能看到它  
前面的词, 不能看到后面的词.

两种归一化方法:

Batch Size



$\mu = 0, \sigma = 1$   
Batch normalization

① Layer normalization

对data, 对不同dimension归一化, 希望 $\mu=0, \sigma^2=1$ .

对同一batch不同data的同一dimension归一化.  
希望归一化后 $\mu=0, \sigma^2=1$ .