

# 自然语言处理

## **Natural language Processing**

授课教师：胡玥

2024.12





## 补充材料1: NLP中的注意力机制

# 内 容 提 要

---

5.1 注意力机制概述

5.2 传统注意力机制

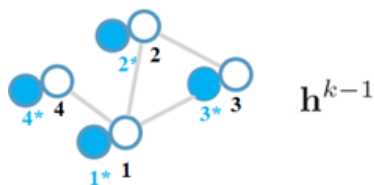
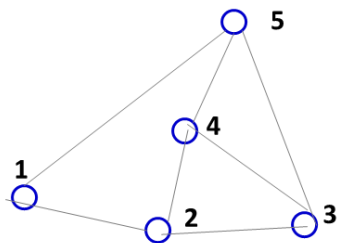
5.3 注意力编码机制

# 5.1 注意力机制概述

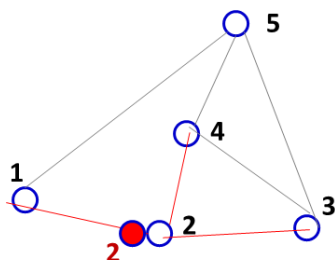
## 什么是注意力机制？

注意力机制：加权求和机制/模块

例：



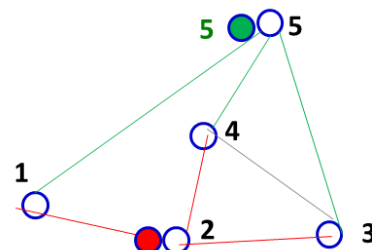
图卷积网邻接节点聚集



输入：1 3 4

求 1, 3, 4 的权重 然后加权求和

输出：2



输入：1 3 4

求 1, 3, 4 的权重 然后加权求和

输出：5

输入：Q, K(集合)

通用：

求 K对Q 的权重 然后加权求和

输出：Att-V

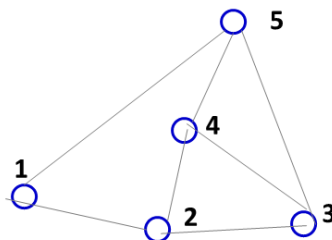
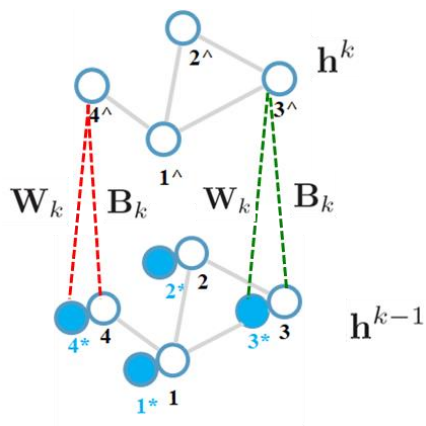
1. 如何求K 对Q 的权重
2. 如何计算最后值

## 5.1 注意力机制概述

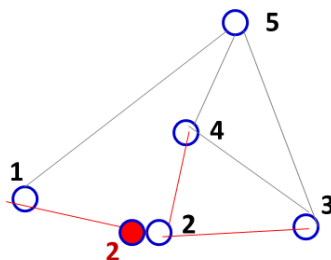
### 注意力机制作用

- 等权求和 → 加权求和

例：



图卷积网邻接节点聚集

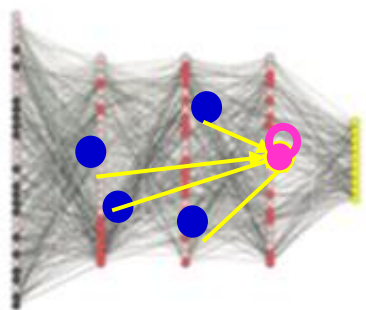


图卷积网邻接节点加权聚集

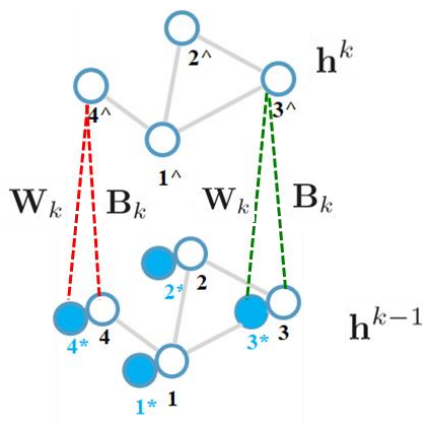
## 5.1 注意力机制概述

### 注意力机制作用

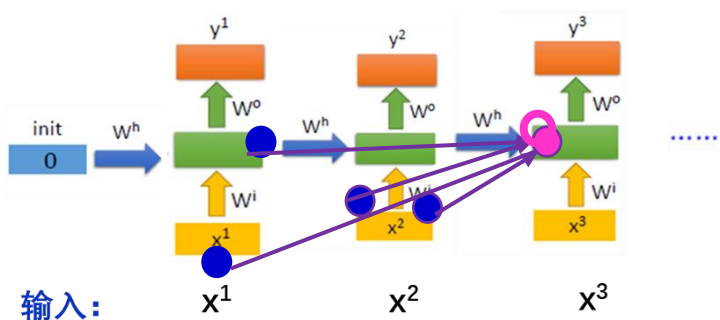
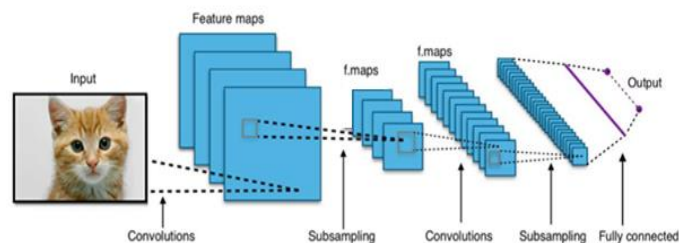
- 任意节点间建立关联关系



- 任意节点间建立关联关系



- 等权求和  $\rightarrow$  加权求和

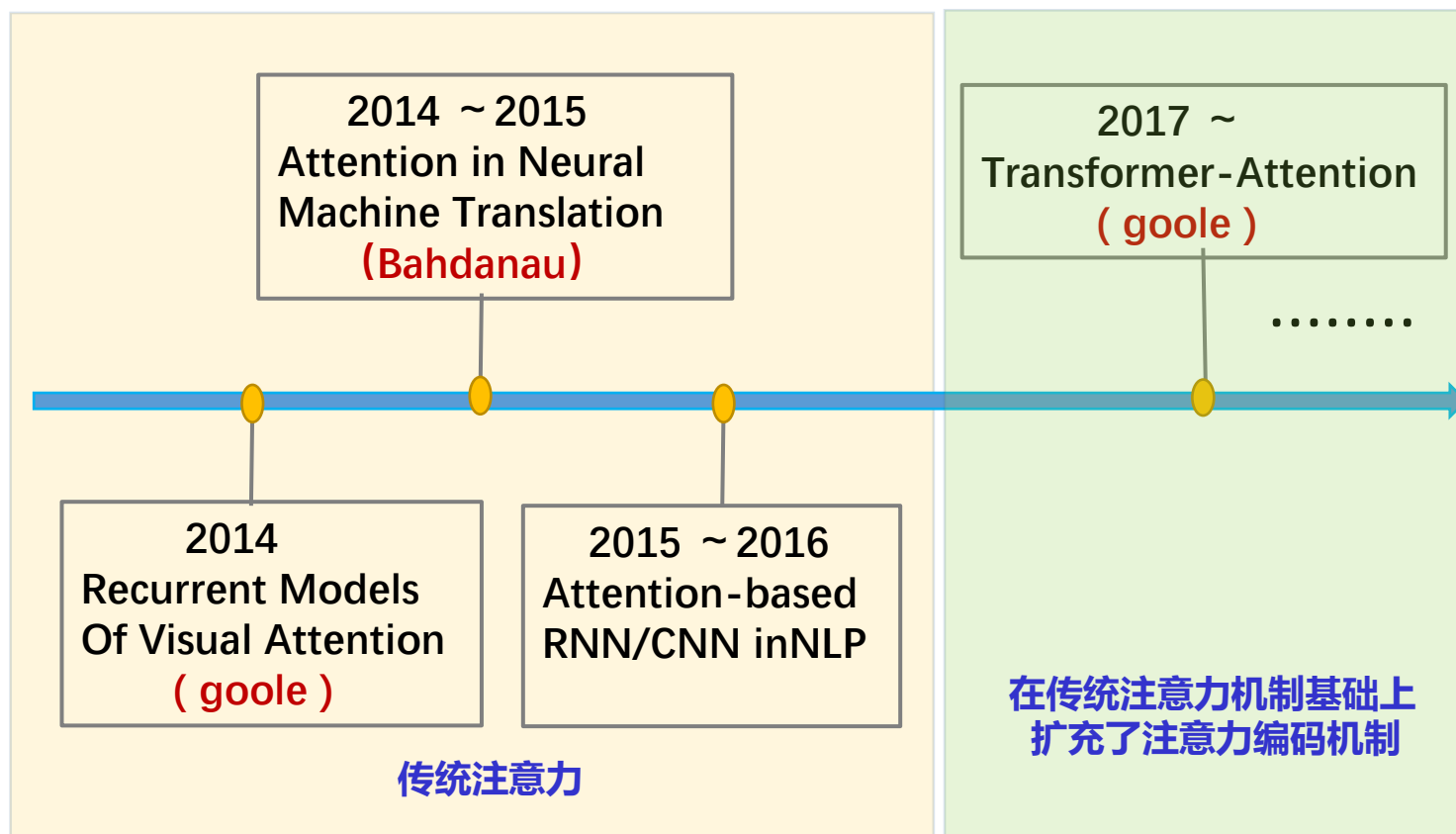


- 任意节点间建立关联关系

已有加权求和链接的不再用

## 5.1 注意力机制概述

### 注意力机制发展历史



# 内 容 提 要

---

5.1 注意力机制概述

5.2 传统注意力机制

5.3 注意力编码机制

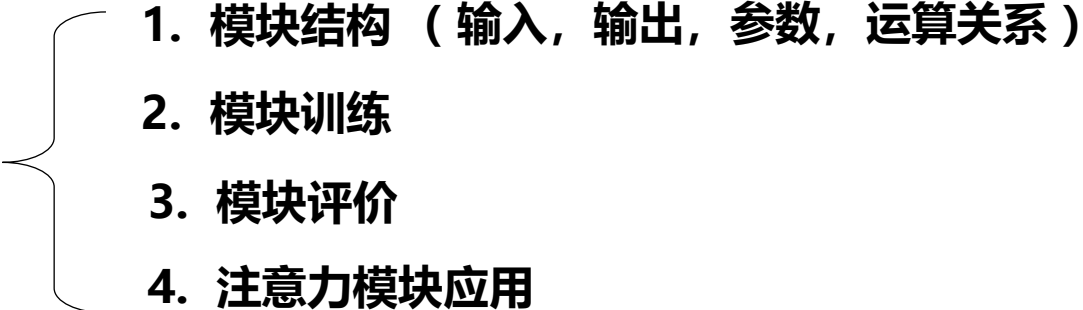


## 5.2 传统注意力机制

### 注意力机制

**加权求和模块：**神经网络中的一个组件，可以单独使用，但更多地用作网络中的一部分。

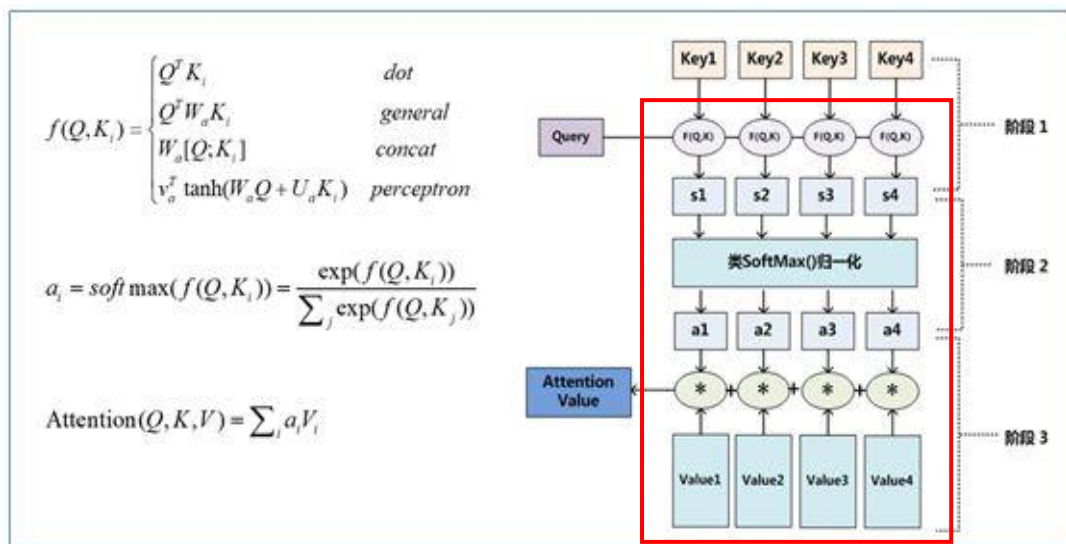
**加权求和模块  
(Attention)**

- 
1. 模块结构（输入，输出，参数，运算关系）
  2. 模块训练
  3. 模块评价
  4. 注意力模块应用

## 5.2 传统注意力机制

### 1. 注意力模块结构：

输入 / 输出：



输入：Q，K(集合)

输出：Att-V

输入：Q，K(集合)

通用：

求 K 对 Q 的权重  
然后加权求和

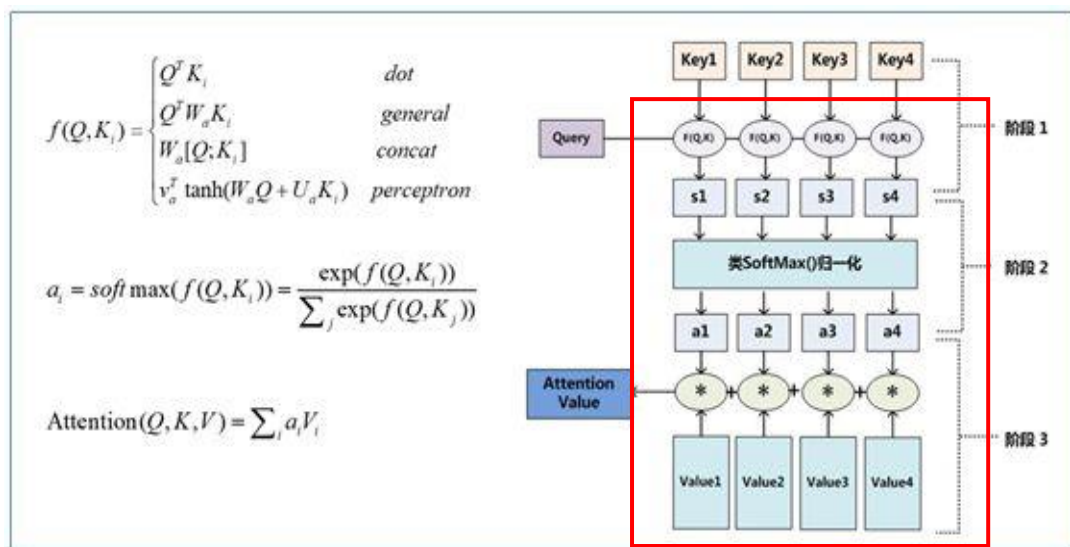
1. 如何求 K 对 Q 的权
2. 如何计算最后值

输出：Att-V

## 5.2 传统注意力机制

### 1. 注意力模块结构:

输入 → 输出 函数关系:



输入: Q, K(集合)

求 K对Q 的权重  
然后加权求和

输出: Att-V

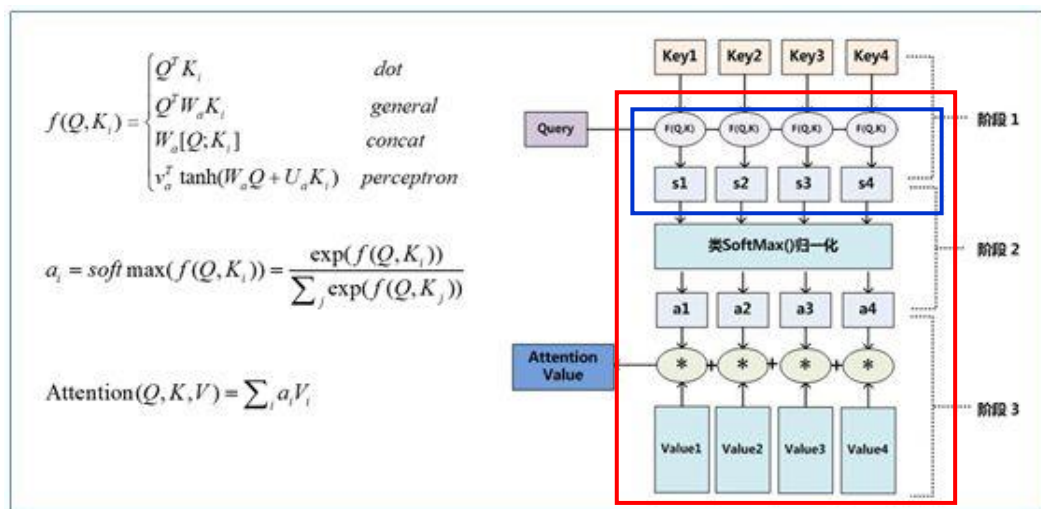
#### 1. 如何求K 对Q 的权

- 计算注意力打分函数  $S = f(Q, K_i)$
- $\text{softmax}(S = f(Q, K_i))$  (计算对于Q 各个  $K_i$  的权重)

## 5.2 传统注意力机制

### 1. 注意力模块结构:

输入 → 输出 函数关系:



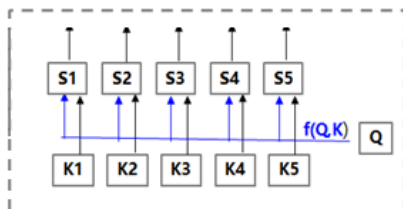
输入:  $Q, K(\text{集合})$

求  $K$  对  $Q$  的权重  
然后加权求和

输出:  $\text{Att-V}$

### 1. 如何求 $K$ 对 $Q$ 的权

- 计算注意力打分函数  $S = f(Q, K_i)$

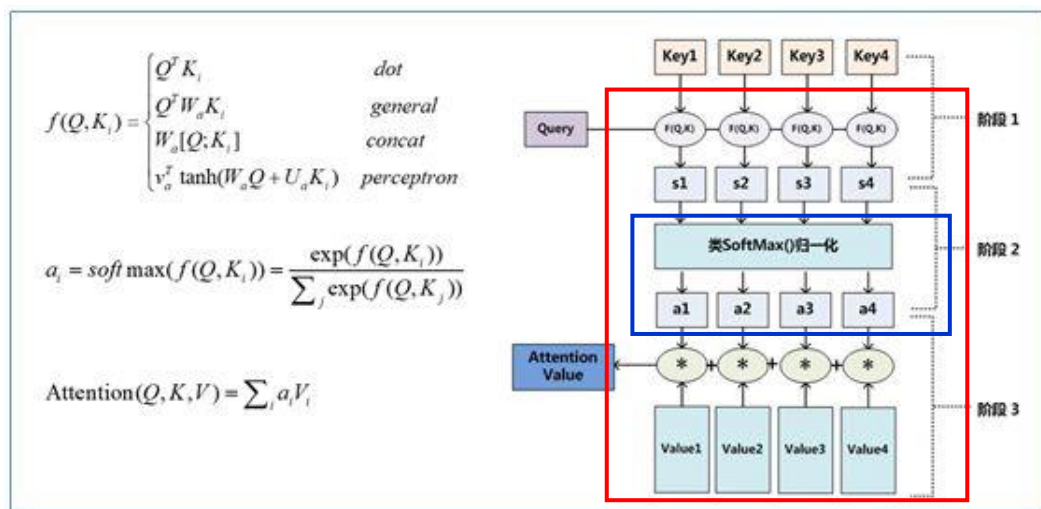


$$S = f(Q, K) = \begin{cases} Q^T K_i & \text{点积模型} \\ \frac{Q^T K_i}{\sqrt{d}} & \text{缩放点积模型} \\ W_a [Q, K_i] & \text{连接模型} \\ Q^T W_a K_i & \text{双线性模型} \\ v_a^T \tanh(W_a Q + U_a K_i) & \text{加性模型} \end{cases}$$

## 5.2 传统注意力机制

### 1. 注意力模块结构:

输入 → 输出 函数关系:



输入: Q, K(集合)

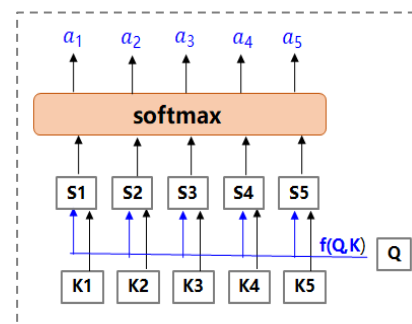
求 K对Q 的权重  
然后加权求和

输出: Att-V

#### 1. 如何求K 对Q 的权

- 计算注意力打分函数  $S = f(Q, K_i)$
- $\text{softmax}(S = f(Q, K_i))$  (计算对于Q 各个  $K_i$  的权重)

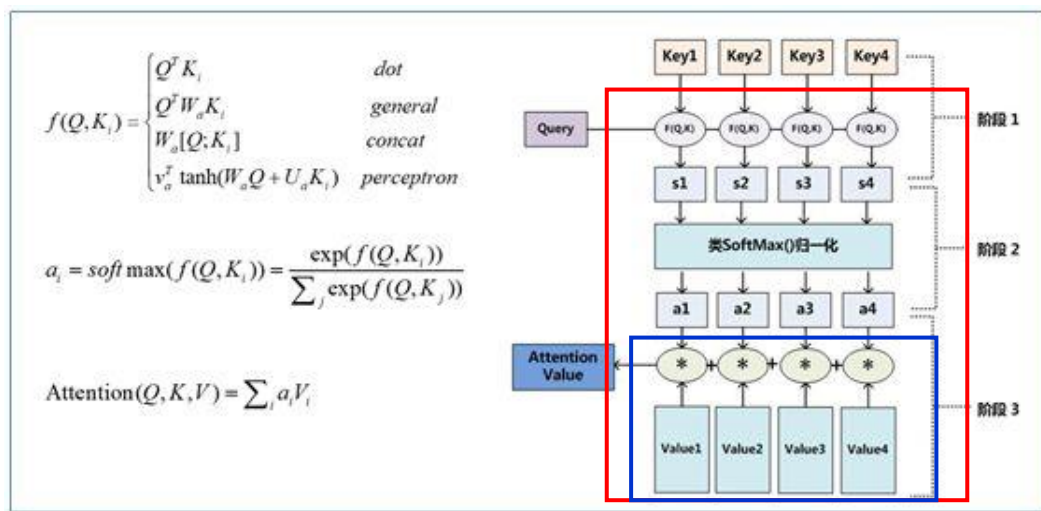
$$a_i = \text{softmax}(f(Q, K_i)) = \frac{\exp(f(Q, K_i))}{\sum_j \exp(f(Q, K_j))}$$



## 5.2 传统注意力机制

### 1. 注意力模块结构:

输入 → 输出 函数关系:



输入: Q, K(集合)

求 K对Q 的权重  
然后加权求和

输出: Att-V

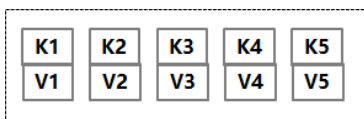
### 2. 如何计算最后值

普通模式



$$\text{Att-V} = a1 \times K1 + a2 \times K2 + a3 \times K3 + a4 \times K4 + a5 \times K5$$

键值对模式

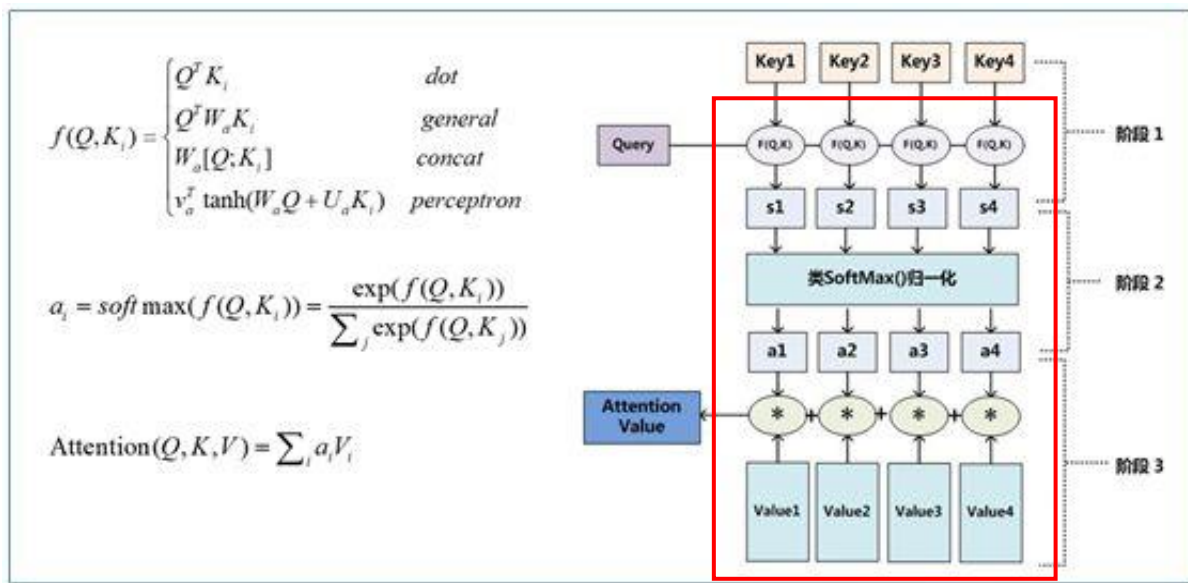


$$\text{Att-V} = a1 \times V1 + a2 \times V2 + a3 \times V3 + a4 \times V4 + a5 \times V5$$

## 5.2 传统注意力机制

### 1. 注意力模块结构:

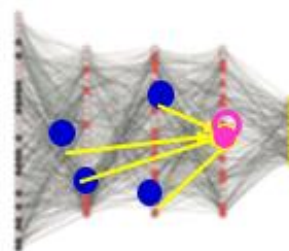
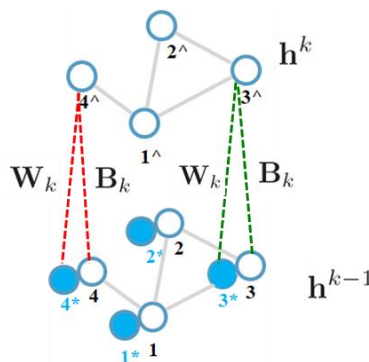
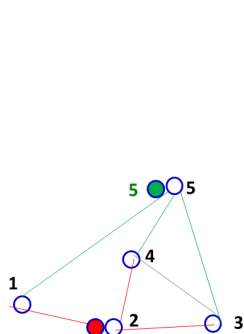
#### 注意力模块



## 5.2 传统注意力机制

### 2. 注意力模块训练

将模块放到整体模型中，不需要额外的训练数据权重可以由模块中的参数学到



网络:  $y = f(x, \theta)$

$\text{Att-V} = \text{Att}(Q, K)$

### 3. 注意力模块评价

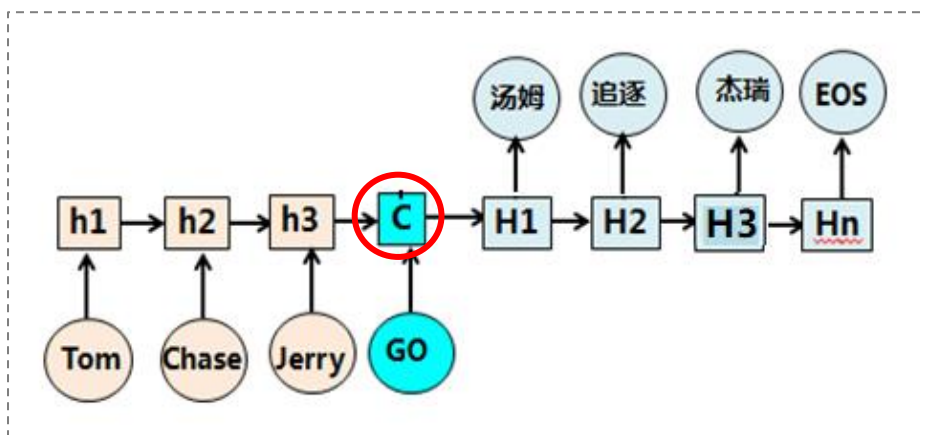
放到各个任务中检验，通过任务指标的提升证明模块的效果



## 5.2 传统注意力机制

### 4. 注意力模块应用

#### 例1：机器翻译例



$$X = \langle x_1, x_2 \dots x_m \rangle$$

$$Y = \langle y_1, y_2 \dots y_n \rangle$$

$$C = \mathcal{F}(x_1, x_2 \dots x_m)$$

$$y_1 = f(C)$$

$$y_2 = f(C, y_1)$$

$$y_3 = f(C, y_1, y_2)$$

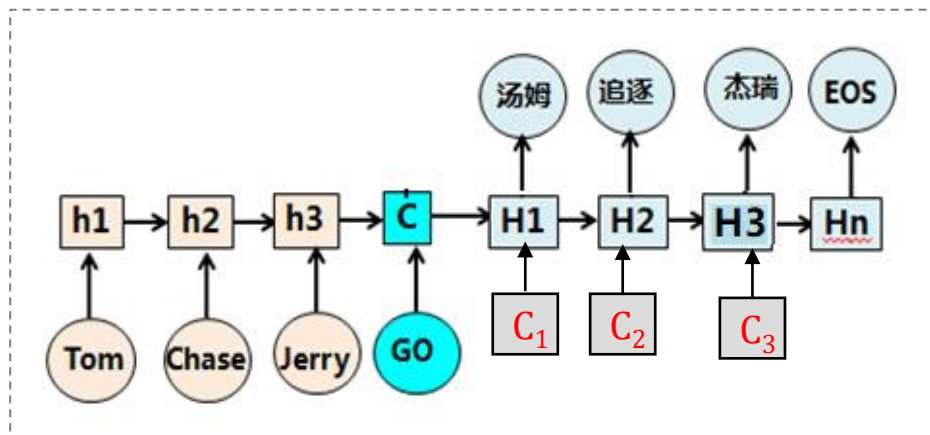
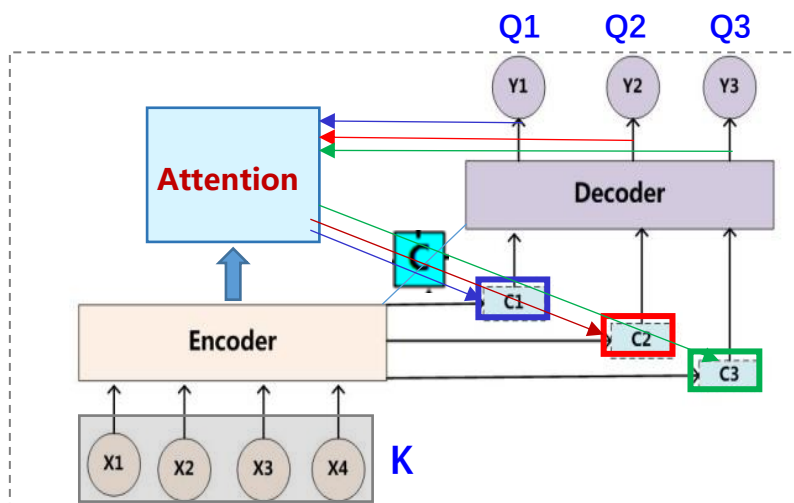
$$y_i = g(C, y_1, y_2 \dots y_{i-1})$$

问题： 对不同的输出  $y_i$  中间语义表示  $C$  相同

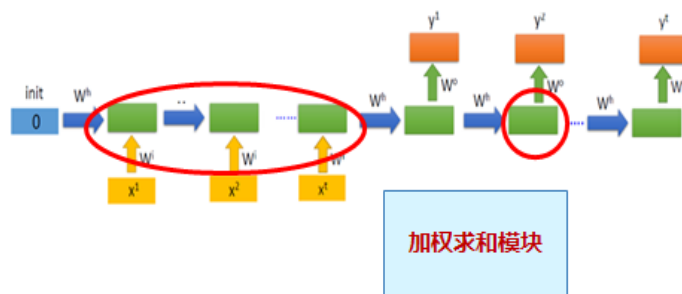
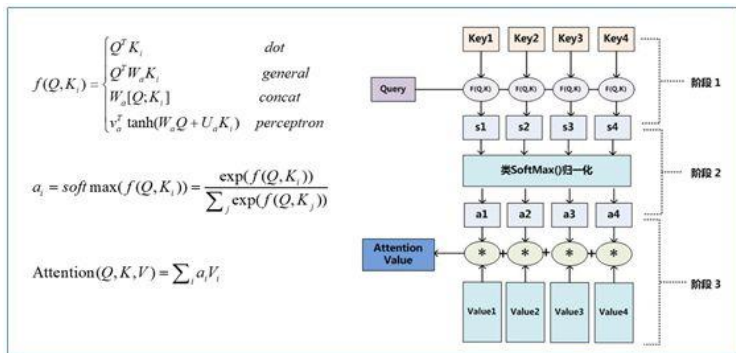
实际应该： 在翻译每个目标语单词时，源语各词对目标词的影响程度是不同的。如翻译“杰瑞”的时候，源语句中各英文单词对于“杰瑞”的影响程度是不同的，如 (Tom,0.3) (Chase,0.2) (Jerry,0.5)

## 5.2 传统注意力机制

解：引入注意力模块

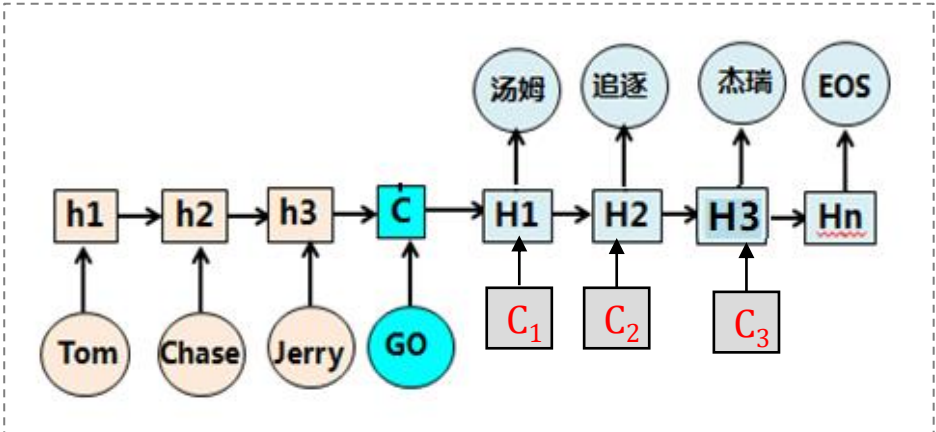
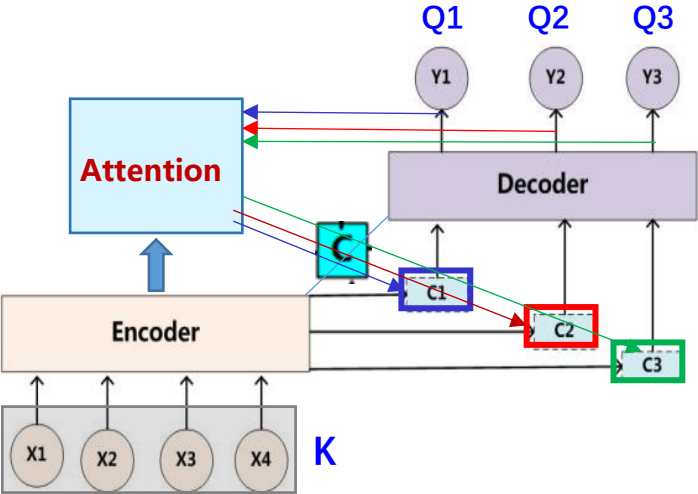


### Attention

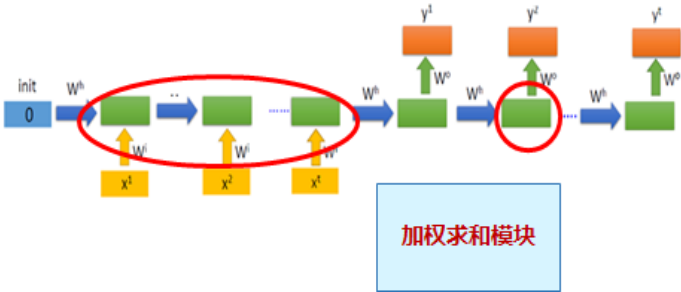
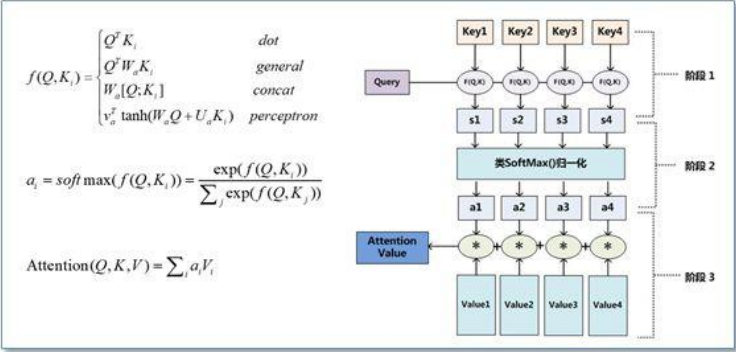


# 5.2 传统注意力机制

解：引入注意力模块

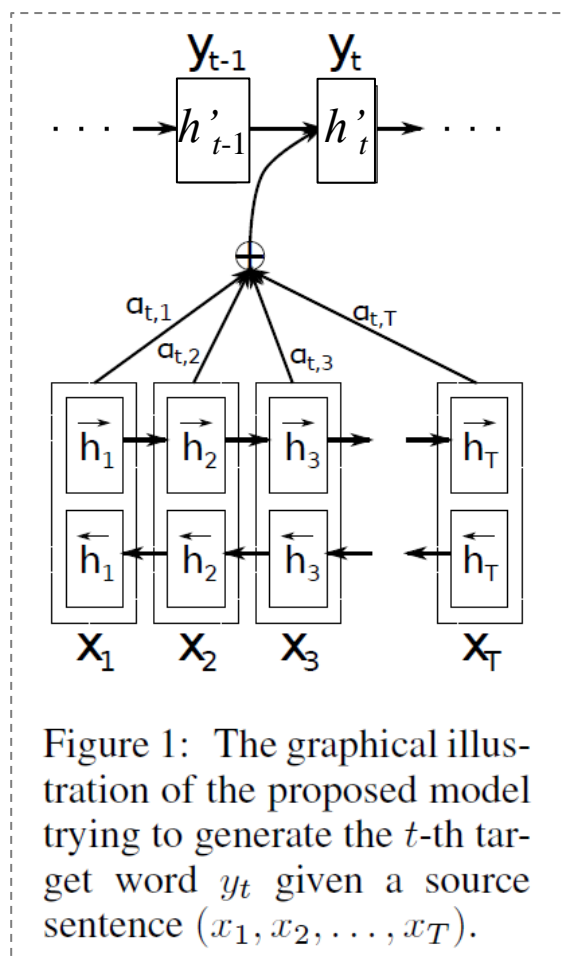


## Attention



## 5.2 传统注意力机制

### Encoder (BiLSTM)-Decoder + Attention



#### ■ 模型结构

编码器采用双向RNN，解码器采用单向RNN

输入：X（源语句子）

输出：Y（目标语句子）

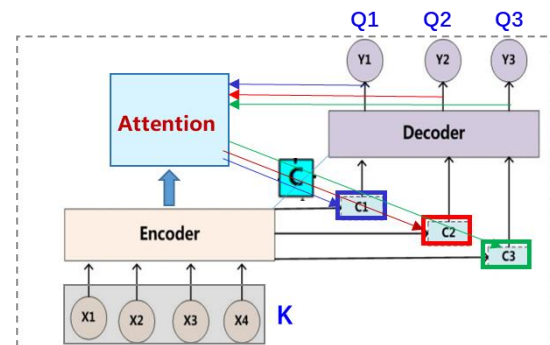
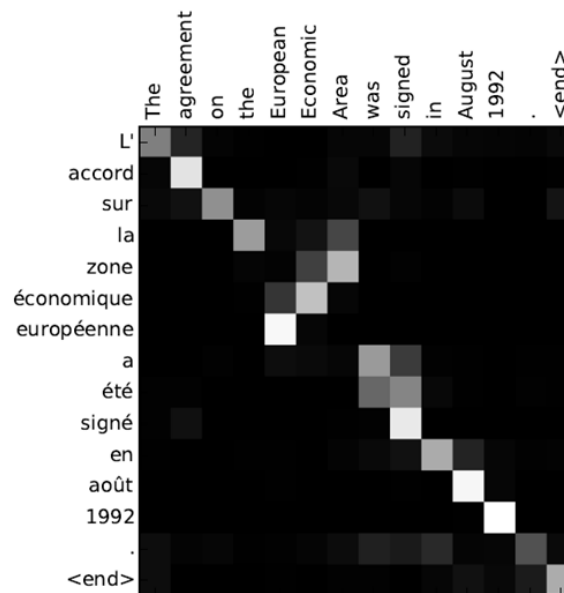
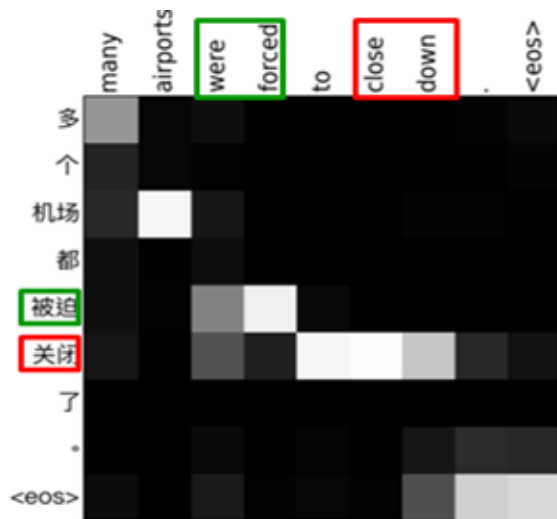
$$p(y_i | y_1, \dots, y_{i-1}, x) = g(y_{i-1}, h_i, c_i)$$

$$h'_i = f(h'_{i-1}, y_{i-1}, c_i)$$

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j$$

## 5.2 传统注意力机制

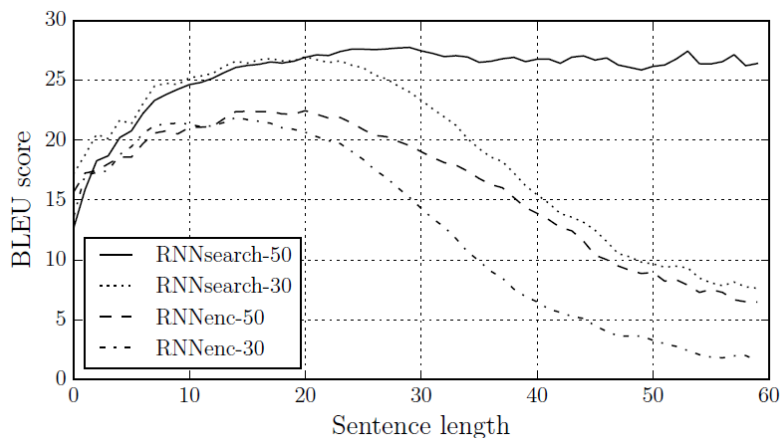
### 注意力机制可视化效果



注意力机制的双语对齐效果

## 5.2 传统注意力机制

### 注意力机制实验结果



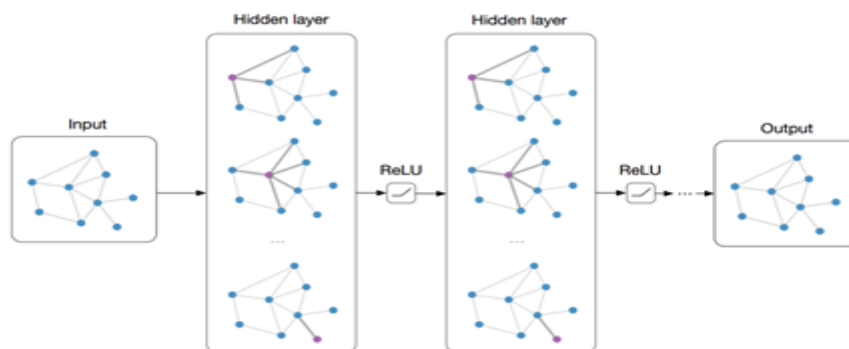
Model	All	No UNK <sup>o</sup>
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50*	28.45	36.15
Moses	33.30	35.63

- 在句子限长为30和50的情况下，加AM模型效果优于不加AM模型
- 句子长度增加时，加AM模型效和不加AM模型的效果均变差，但AM模型鲁棒性较好

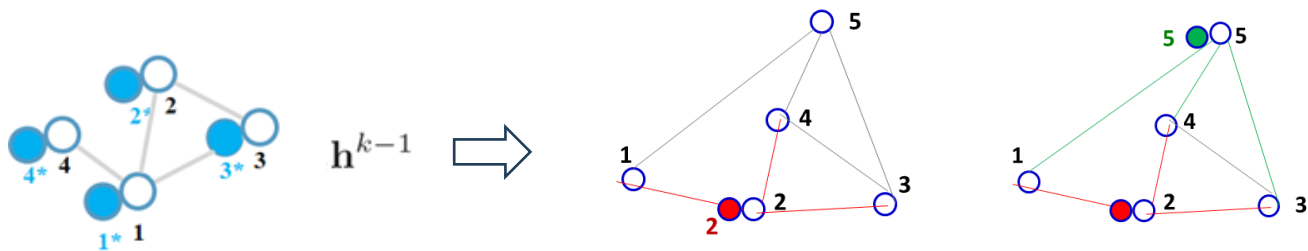
## 5.2 传统注意力机制

### 例2：图卷积中加注意力聚集

#### 图卷积网

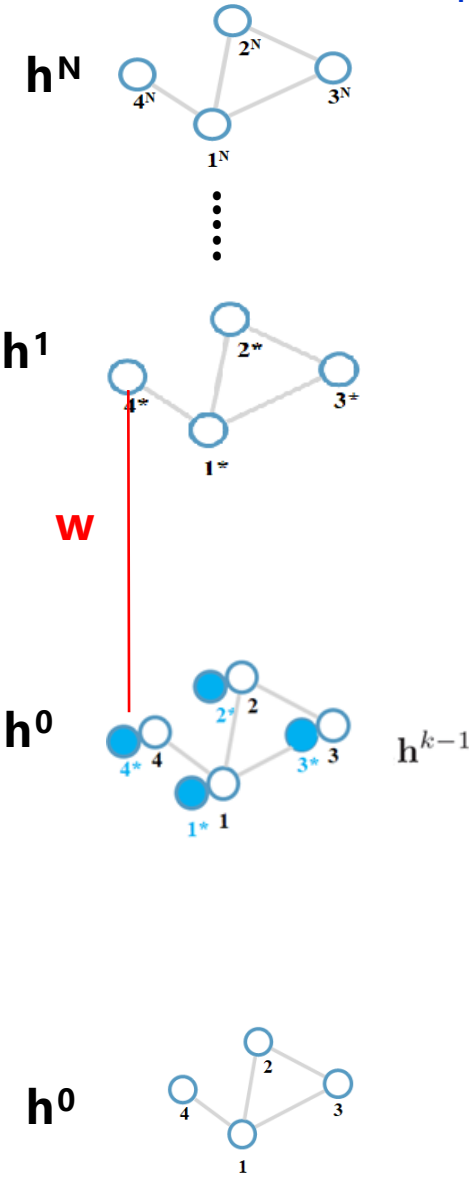


#### 等权聚集 → 加权聚集



例：原图回顾

Output



$$\mathbf{h}_v^k = \sigma \left( \mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1} \right)$$

Step2: Transformation

	f1	f2	f3	f4	f5
Node 1	1.8	0.8	...		
Node 2	0.6	1.0	...		
Node 3	...				
Node 4	...				

feature matrix  $\mathbf{X}' \in \mathbb{R}^{N \times M}$

	k1	k2	k3	k4
w1	0.1	0.7	1.2	1.1
w2	0.3	0.4	0.3	0.1
w3	0.5	0.3	1.5	0.4
w4	1.0	0.1	0.2	0.5
w5	2.0	0.4	0.1	2.4

weight matrix  $\mathbf{W} \in \mathbb{R}^{M \times K}$

	k1	k2	k3	k4
Node 1	2.8	4.8	...	
Node 2	0.6	4.0	...	
Node 3	...			
Node 4	...			

feature matrix  $\mathbf{H} \in \mathbb{R}^{N \times K}$

Step1: Aggregation

	Node 1	Node 2	Node 3	Node 4
Node 1	0	1	1	1
Node 2	1	0	1	0
Node 3	1	1	0	0
Node 4	1	0	0	0

adjacency matrix  $\mathbf{A} \in \mathbb{R}^{N \times N}$

	f1	f2	f3	f4	f5
Node 1	0.1	0.7	1.2	1.1	0.9
Node 2	0.3	0.4	0.3	0.1	1.2
Node 3	0.5	0.3	1.5	0.4	0.6
Node 4	1.0	0.1	0.2	0.5	0.1

feature matrix  $\mathbf{X} \in \mathbb{R}^{N \times M}$

	f1	f2	f3	f4	f5
Node 1	1.8	0.8	...		
Node 2	0.6	1.0	...		
Node 3	...				
Node 4	...				

feature matrix  $\mathbf{X}' \in \mathbb{R}^{N \times M}$

Input:

	Node 1	Node 2	Node 3	Node 4
Node 1	0	1	1	1
Node 2	1	0	1	0
Node 3	1	1	0	0
Node 4	1	0	0	0

adjacency matrix  $\mathbf{A}$

	f1	f2	f3	f4	f5
Node 1	0.1	0.7	1.2	1.1	0.9
Node 2	0.3	0.4	0.3	0.1	1.2
Node 3	0.5	0.3	1.5	0.4	0.6
Node 4	1.0	0.1	0.2	0.5	0.1

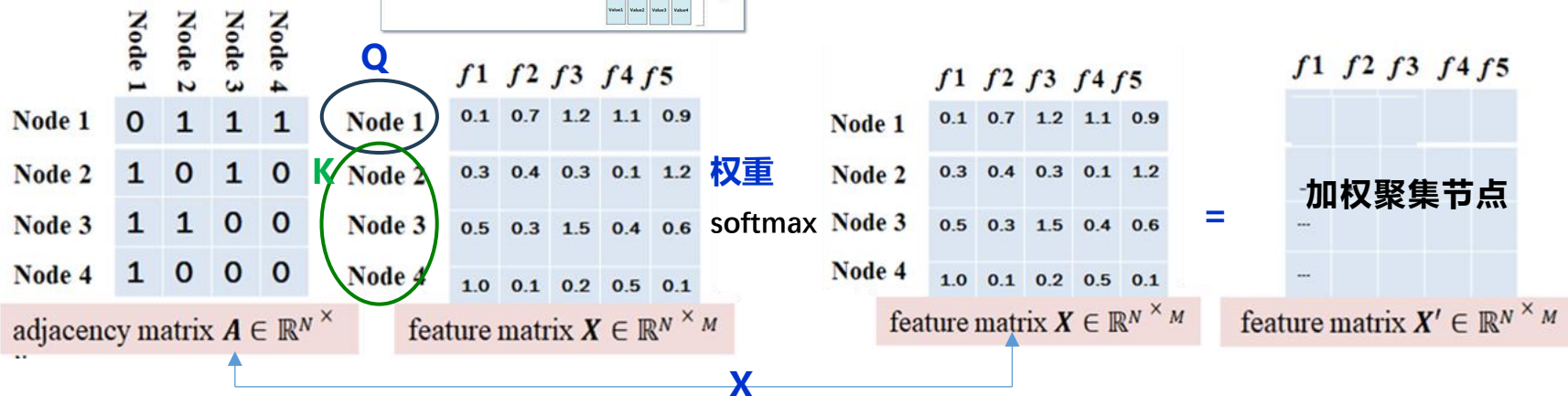
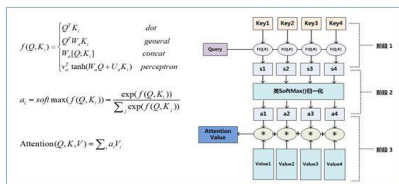
feature matrix  $\mathbf{X} \in \mathbb{R}^{N \times M}$



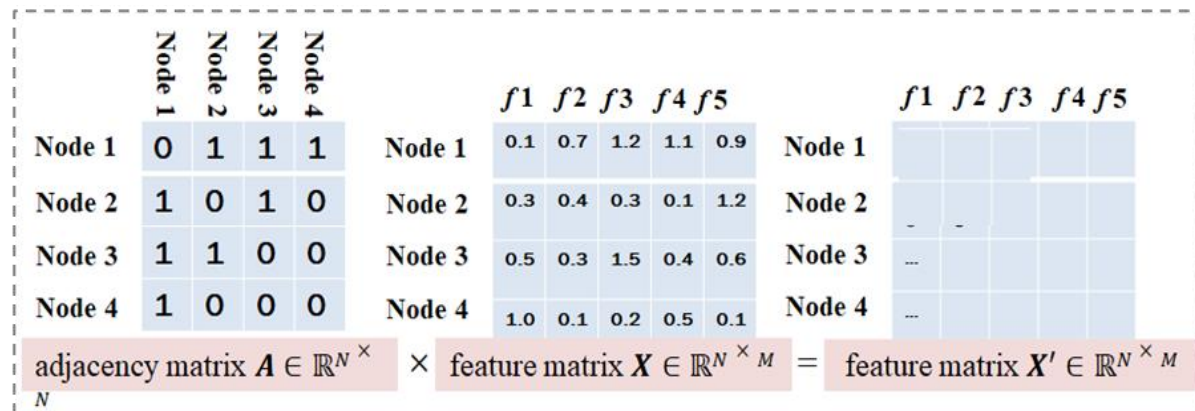
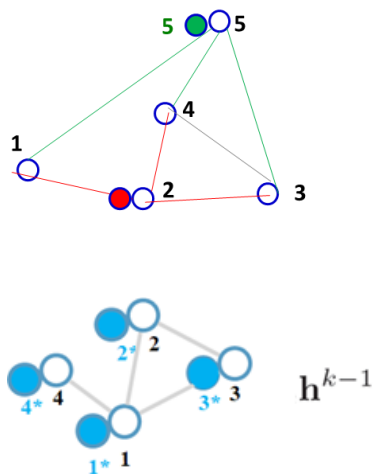
# 例：加权求和

## Step1: Attention –Aggregation

### Attention



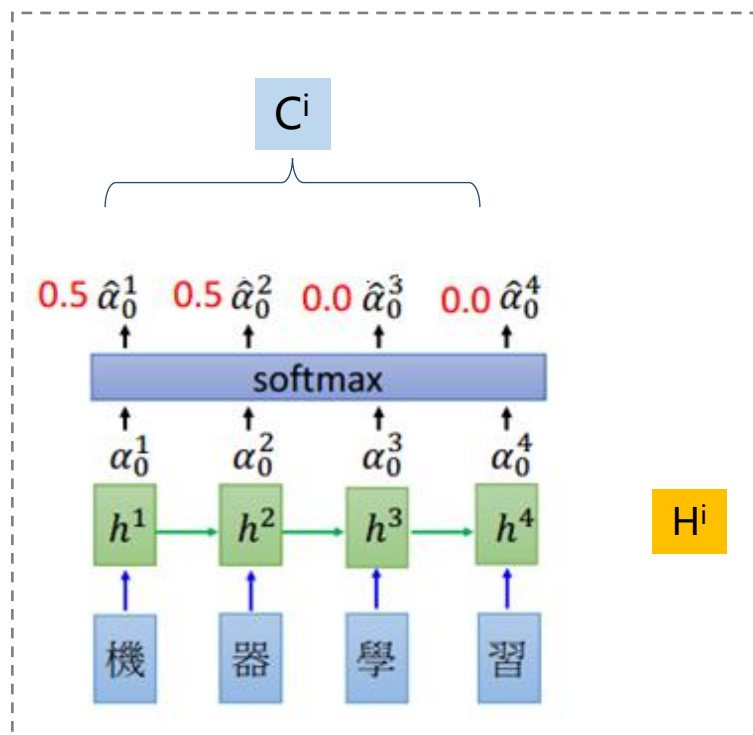
## Step1: Aggregation



## 5.2 传统注意力机制

### □ 软注意力 Soft Attention

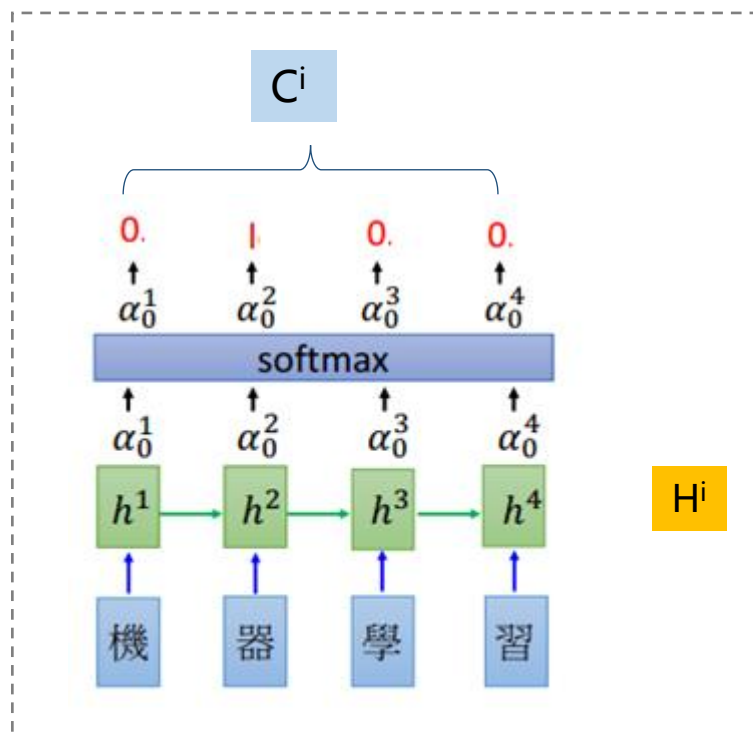
**Soft AM:** 在求注意力分配概率分布的时候，对于输入句子X中任意一个单词都给出个概率，是个概率分布。



## 5.2 传统注意力机制

### □ 硬注意力 Hard Attention

**Hard AM:** 直接从输入句子里面找到某个特定的单词，然后把目标句子单词和这个单词对齐，而其它输入句子中的单词硬性地认为对齐概率为0

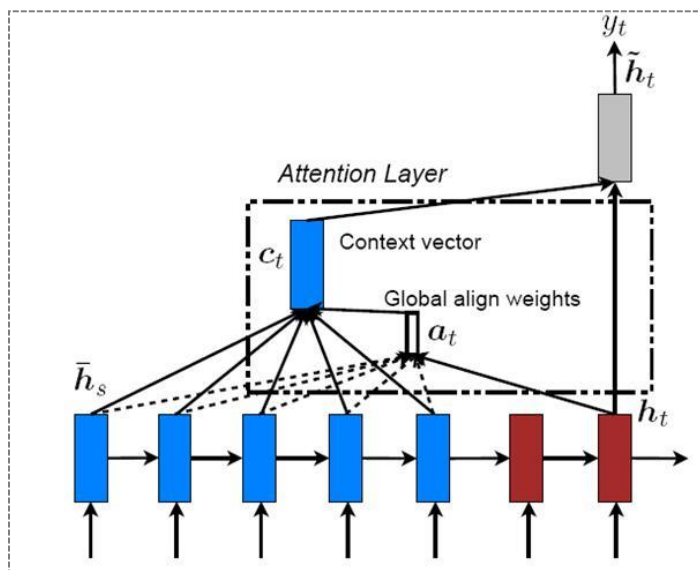


## 5.2 传统注意力机制

### □ 全局注意力 Global Attention

Decode端Attention计算时要考虑Encoder端序列中所有的词

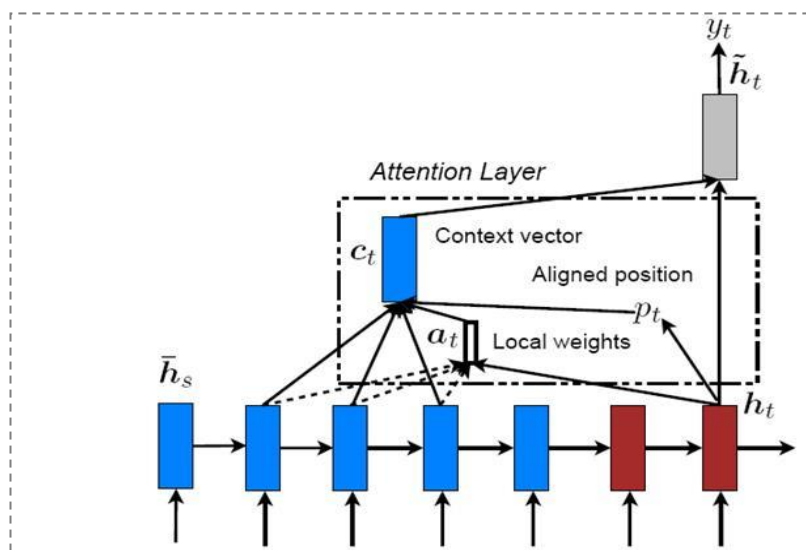
Global Attention Model 是Soft Attention Model



## 5.2 传统注意力机制

### □ 局部注意力 Local Attention

Local Attention Model本质上是Soft AM和 Hard AM的一个混合或折衷。一般首先预估一个对齐位置 $p_t$ ，然后在 $p_t$ 左右大小为 $D$ 的窗口范围来取类似于Soft AM的概率分布。



## 5.2 传统注意力机制

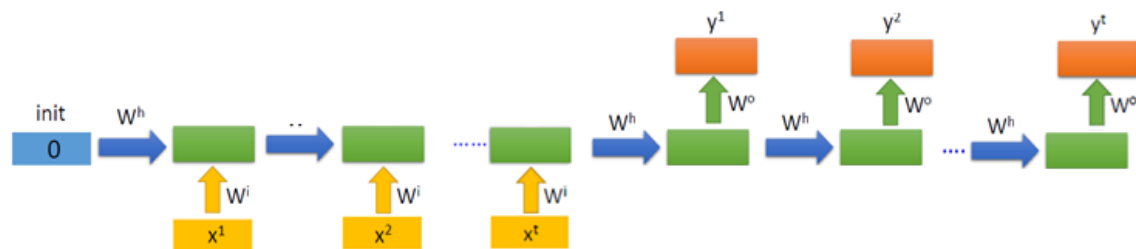
---

### 注意力机制优势

- 让任务处理系统找到与当前任务相关显著的输入信息，并按重要性进行处理，从而提高输出的质量。
- 不需要监督信号，可推理多种不同模态数据之间的难以解释、隐蔽性强、复杂映射关系，对于先验认知少的问题，极为有效。
- 解决长距离依赖问题，提升任务性能

## 5.2 传统注意力机制

**存在问题：** 对RNN有注意力偏置问题



**解决方案：** Coverage机制可以缓解注意力偏置问题

# 内 容 提 要

---

5.1 注意力机制概述

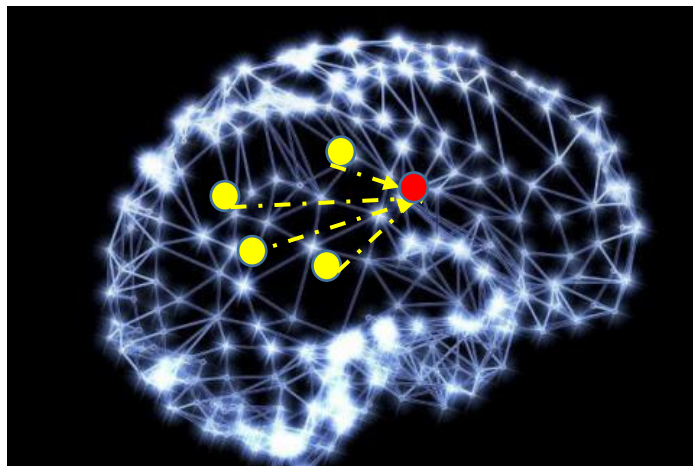
5.2 传统注意力机制

5.3 注意力编码机制



## 5.3 注意力编码机制

### 深度学习编码机制

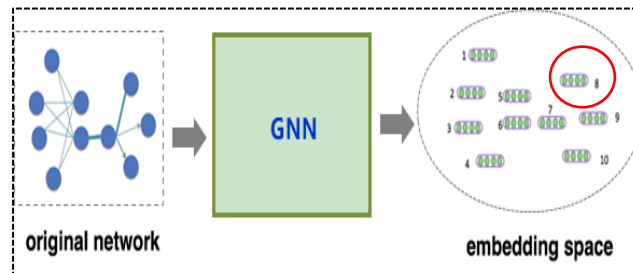
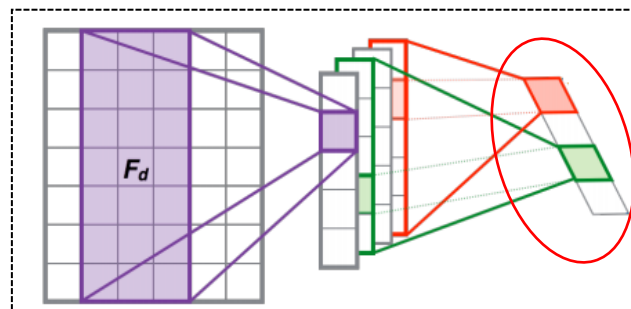
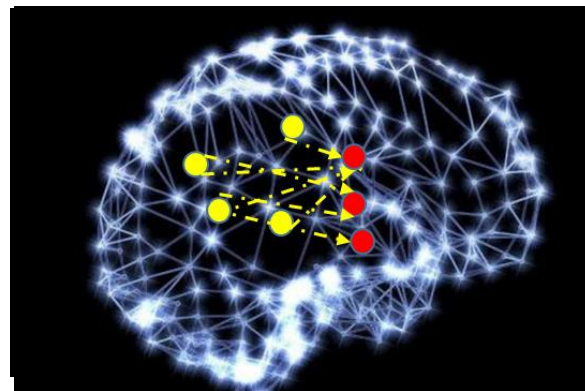
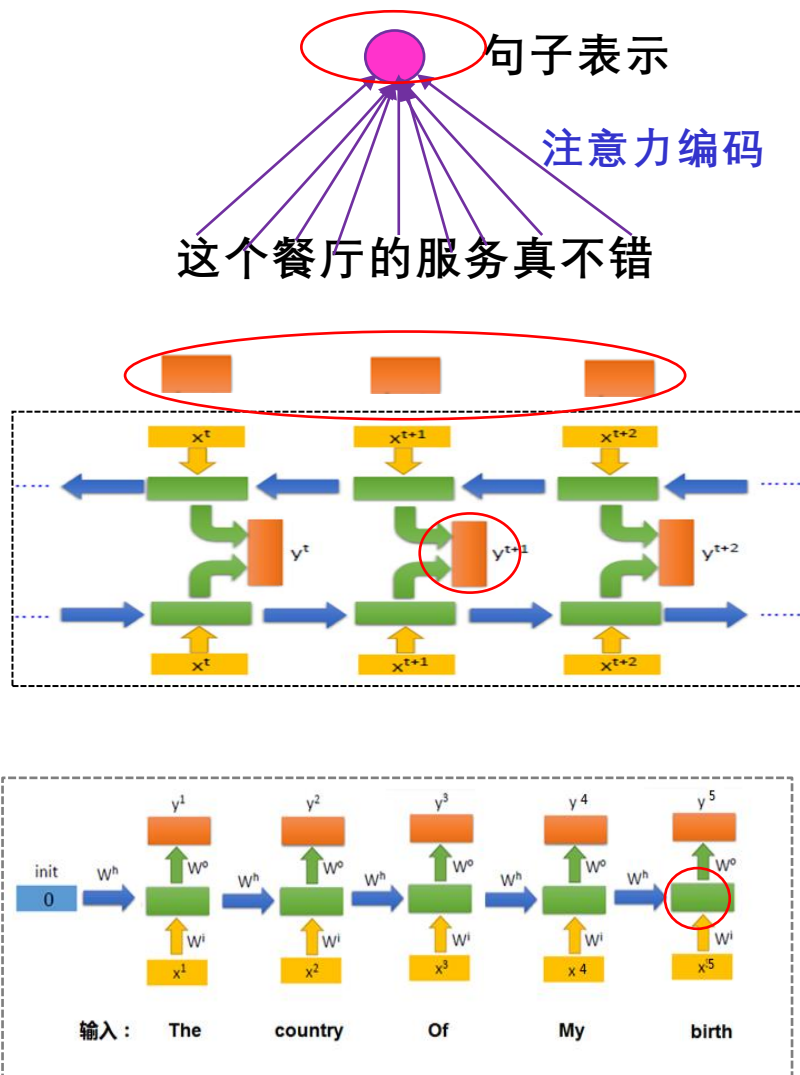


● 可以是词，句子 等后继处理单元

**编码：**将神经网络中分散的信息聚集为某种隐层表示，形成信息量更丰富的表示，以便后继处理

## 5.3 注意力编码机制

例如：



## 5.3 注意力编码机制

### 注意力编码机制

注意力机制作为编码机制主要有：

#### ◆ 编码为单一向量：

- **句编码**：将句子编码为一个句向量
- **词编码**：将序列中的词进行词编码（编后词带有句子的各词权重信息）

#### ◆ 编码为一个序列：将2个序列编码为一个序列

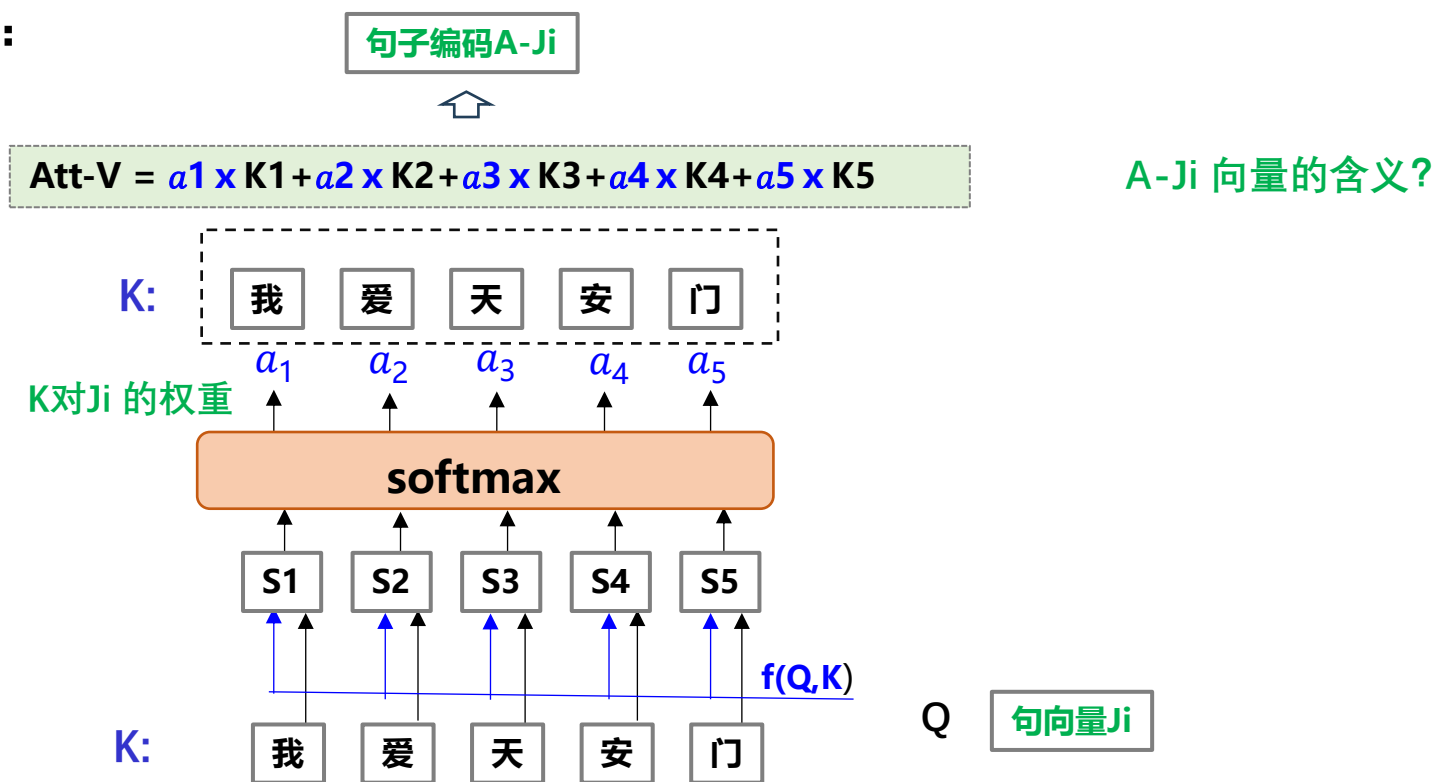
- **不同序列融合编码**：将2个不同的序列编码成二者的融合的表达序列，如，匹配任务和阅读理解任务常用的融合层表示
- **相同序列自注意力编码**：利用多头自注意力编码对一个句子编码可以起到类似句法分析器的作用。如Transformer的编码端

## 5.3 注意力编码机制

### ◆ 编码为单一向量:

- 句编码: 将句子编码为一个句向量

例2:



## 5.3 注意力编码机制

### ◆ 编码为单一向量:

- **词编码**：将序列中的词进行词编码（编后词带有句子的各词权重信息）

例： 在实际的下游任务中，常常需要具有上下文关系的词表示

如： The animal didn't cross the street because **it** was too tired

编码 it 时因为 it 可能指代 animal 也可能指代 street。需要同时利用前后的信息才能更好的编码。如下文是 tired，则 it 指 animal；如下文是 wide，则 it 指 street

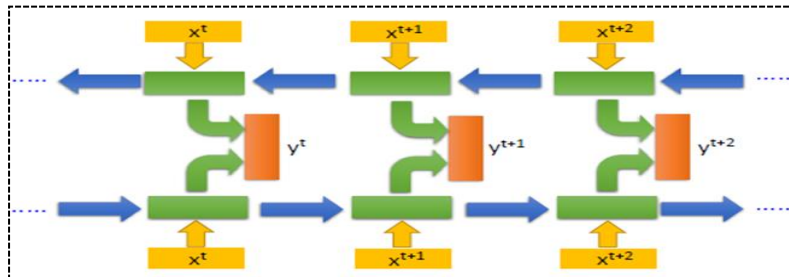
## 5.3 注意力编码机制

- 采用双向RNN语言模型编码词的上下文：

**Forward LM:** The animal didn't cross the street because **it** was too tired

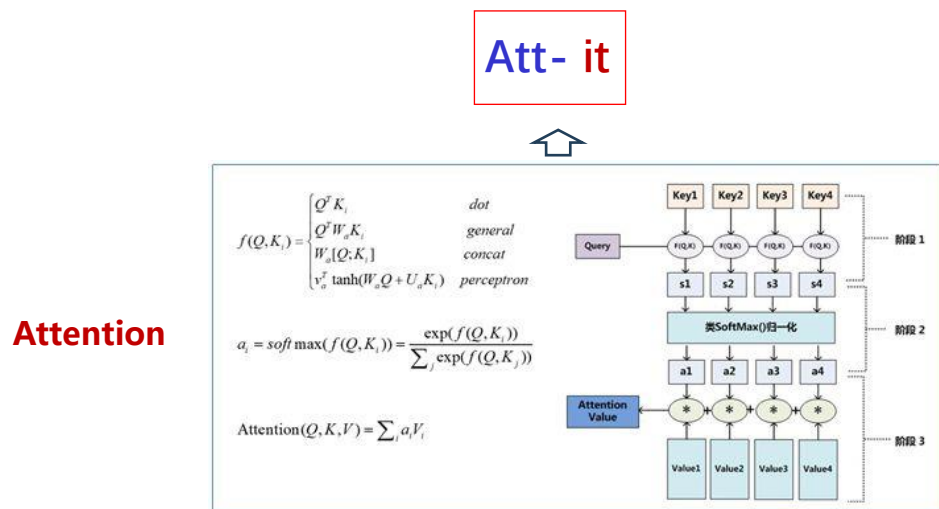
**Backward LM:** The animal didn't cross the street because **it** was too tired

**问题：**双向RNN语言模型实际是单独的两个方向的语言模型，并不能同时观察到上下文。



## 5.3 注意力编码机制

- 采用注意力编码词的上下文:



K: The animal didn't cross the street because **it** was too tired

Q **it**

注意力机制词编码是真正上下文编码

## 5.3 注意力编码机制

### ◆ 编码为一个序列 (不同序列融合编码):

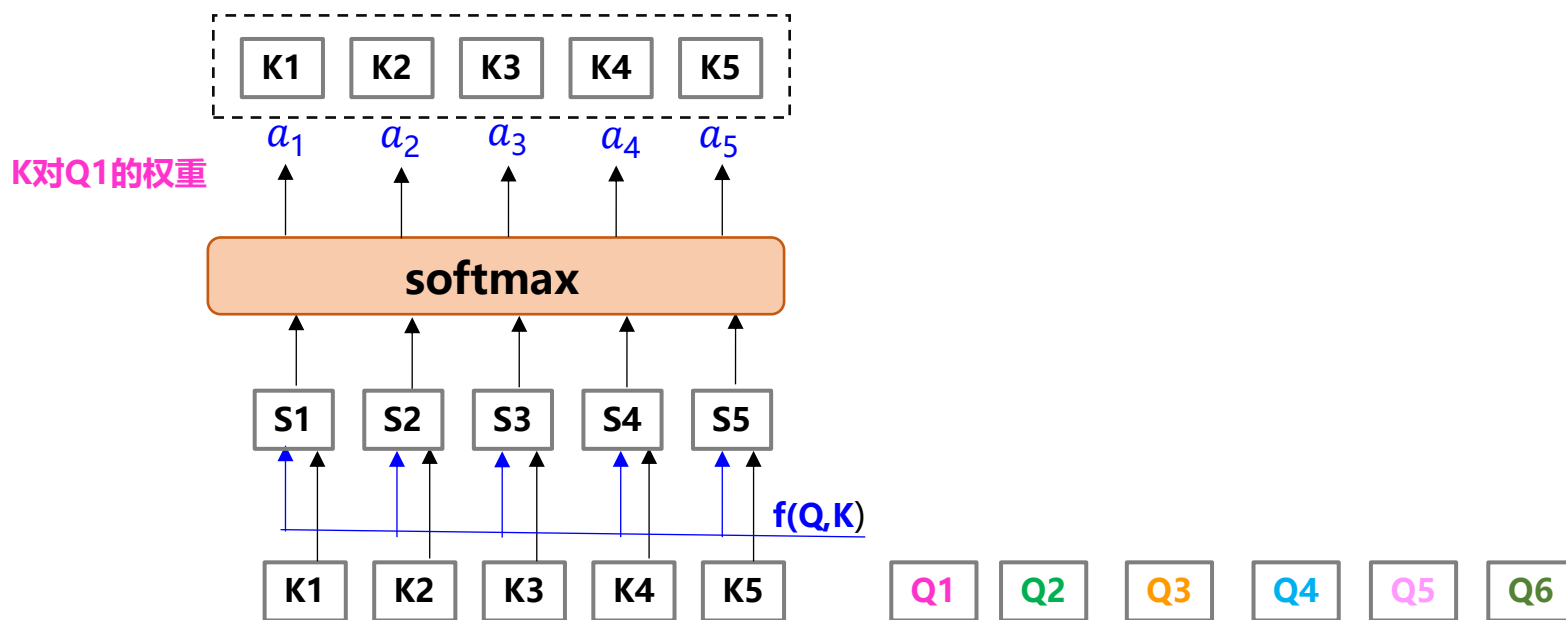
- **不同序列融合编码**: 将2个不同的序列编码成二者的融合的代表序列，  
如，匹配任务和阅读理解任务常用的融合层表示

例：对K序列和Q序列编码

融合编码层

A-V1

$$\text{Att-V} = a_1 \times K_1 + a_2 \times K_2 + a_3 \times K_3 + a_4 \times K_4 + a_5 \times K_5$$





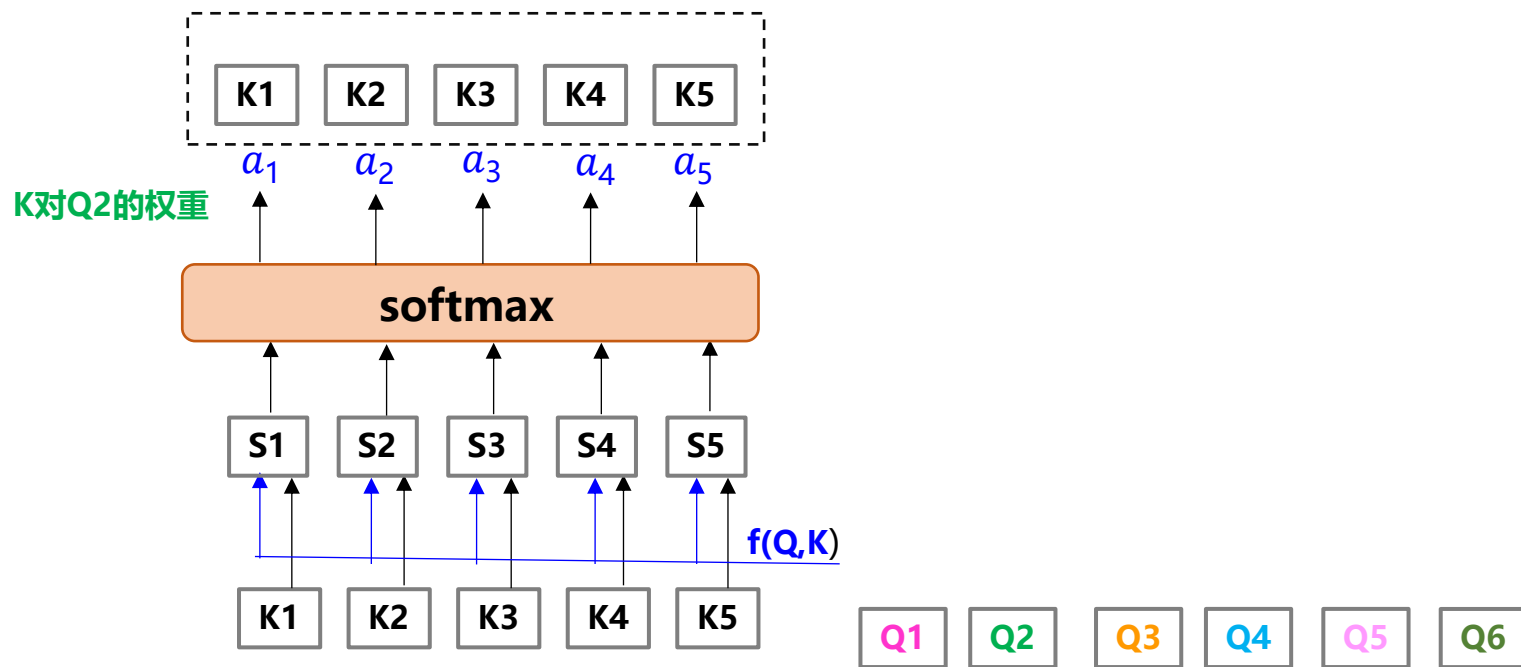
## 5.3 注意力编码机制

融合编码层

A-V1

A-V2

$$\text{Att-V} = a_1 \times K1 + a_2 \times K2 + a_3 \times K3 + a_4 \times K4 + a_5 \times K5$$



## 5.3 注意力编码机制

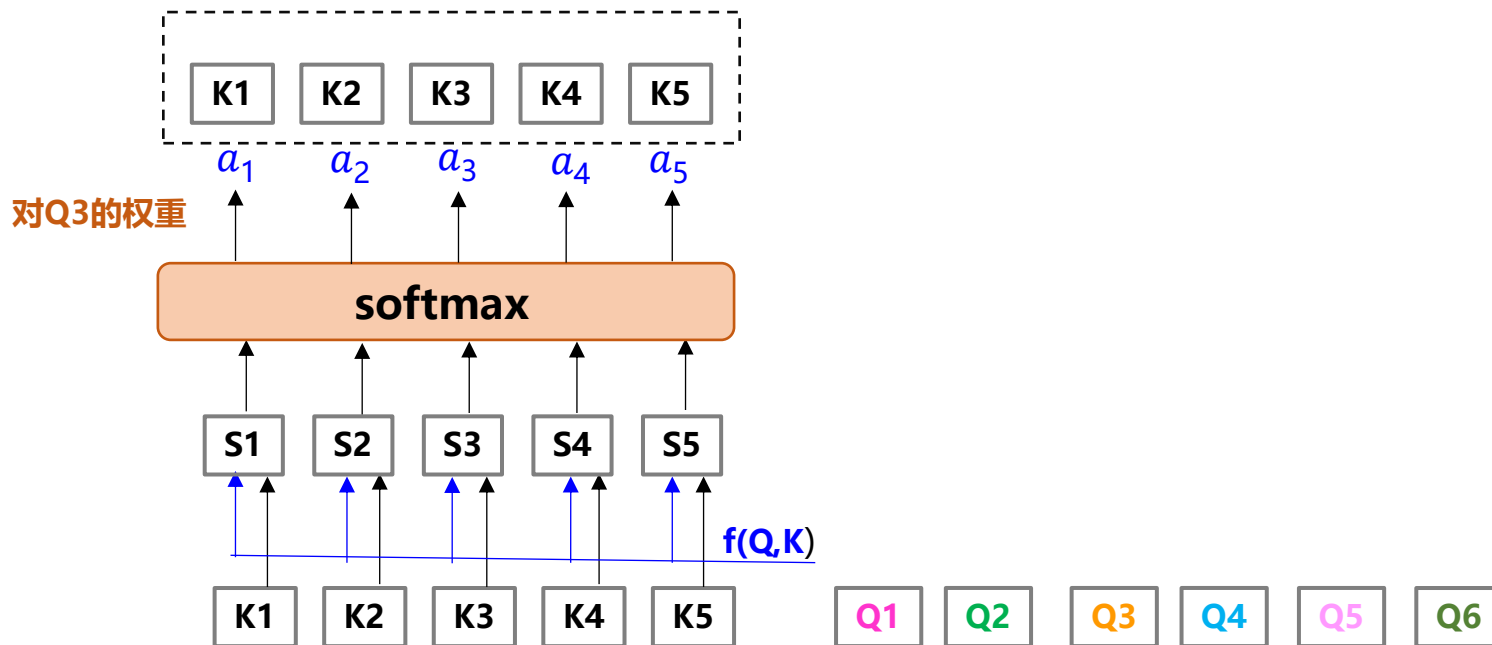
融合编码层

A-V1

A-V2

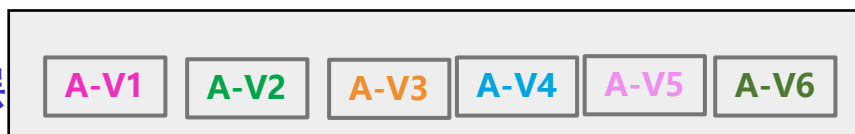
A-V3

$$\text{Att-V} = a_1 \times K1 + a_2 \times K2 + a_3 \times K3 + a_4 \times K4 + a_5 \times K5$$



## 5.3 注意力编码机制

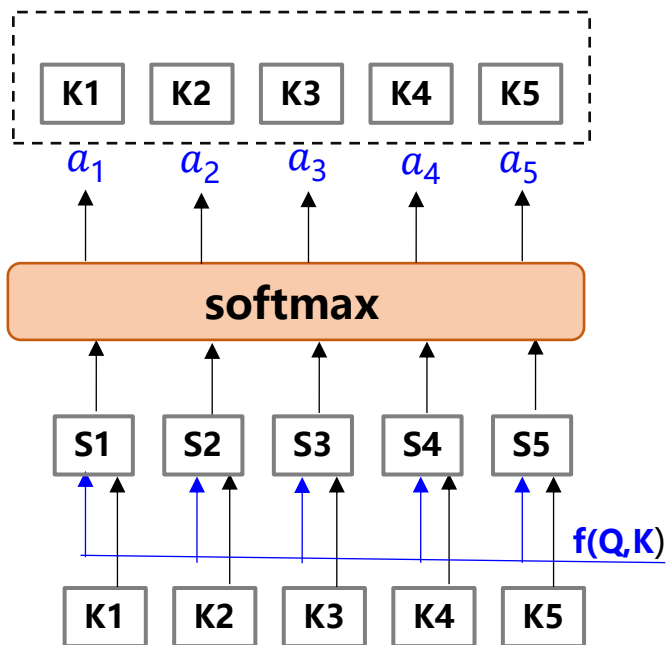
融合编码层



K 与 Q 序列的融合序列

$$\text{Att-V} = a_1 \times K_1 + a_2 \times K_2 + a_3 \times K_3 + a_4 \times K_4 + a_5 \times K_5$$

A-V<sub>i</sub> 序列的含义?



问题: Attention层是词袋模型

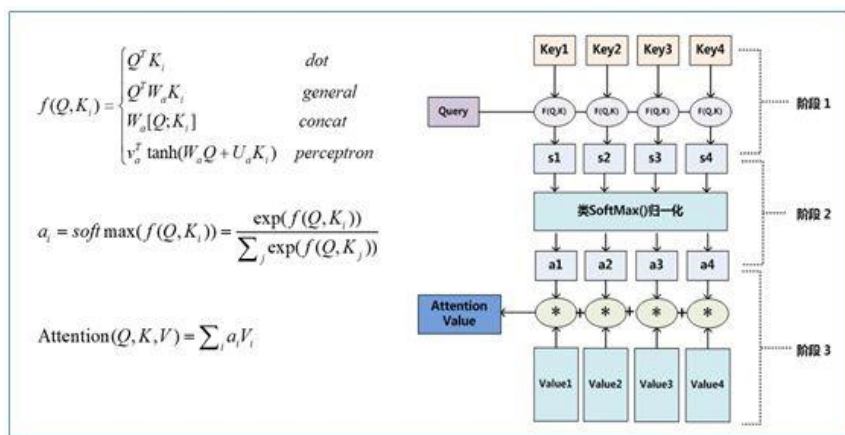
A-V<sub>i</sub> 序列元素个数等于 Q序列元素个数

## 5.3 注意力编码机制

### ◆ 编码为一个序列 (相同序列自注意力编码):

- **相同序列自注意力编码**: 利用多头自注意力编码对一个句子编码可以起到类似句法分析器的作用。如Transformer的编码端

### 自注意力机制



Attention(Q,K,V)

其中，Q=K=V

其实就是 Attention( X , X , X ), X 为输入序，其含义为在序列内部做 Attention 计算，寻找序列内部词与词之间的关联关系

## 5.3 注意力编码机制

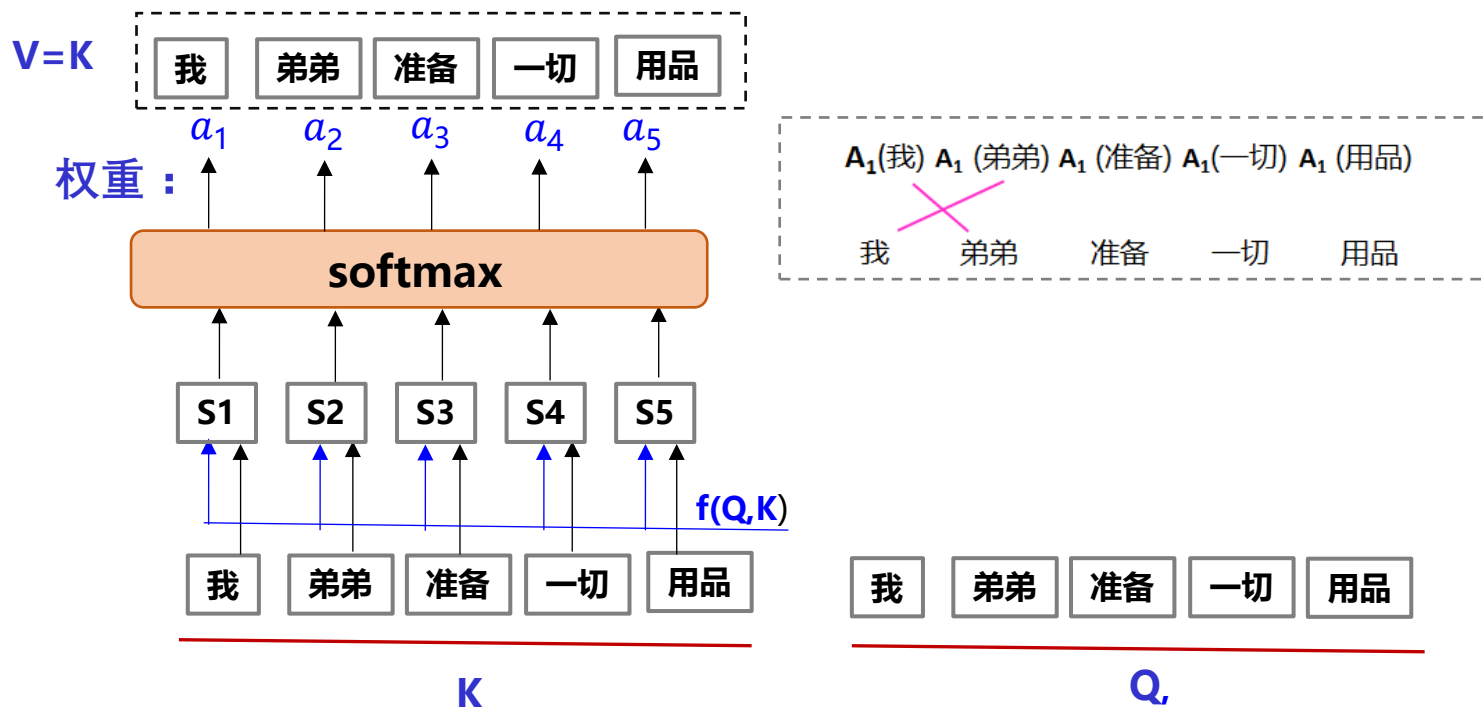
### ◆ 编码为一个序列 (相同序列自注意力编码):

- 对同一序列自注意力编码

自注意力编码层 A-我 A-弟弟 A-准备 A-一切 A-用品

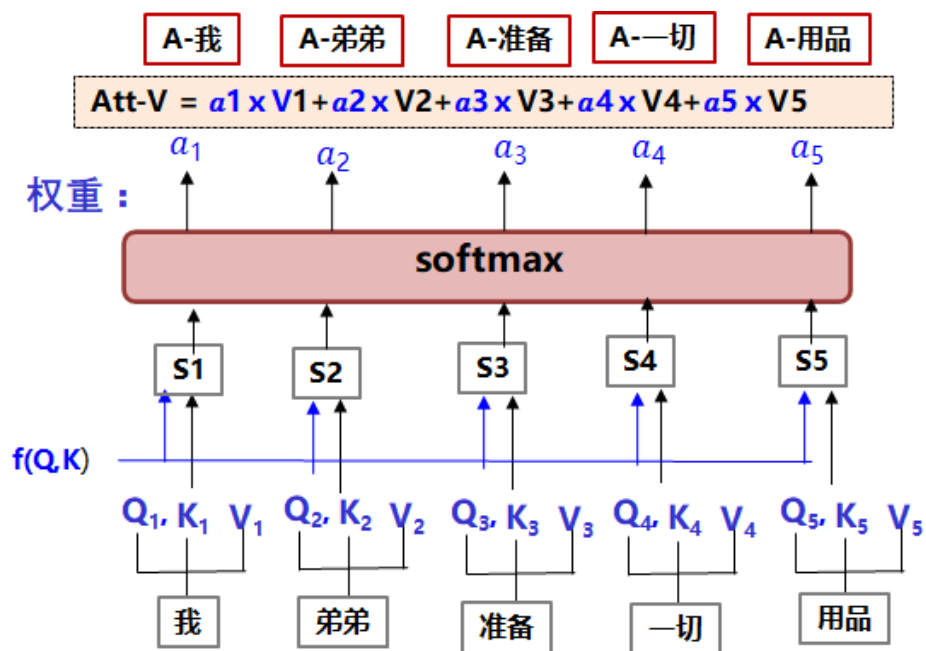
$$\text{Att-V} = a_1 \times K_1 + a_2 \times K_2 + a_3 \times K_3 + a_4 \times K_4 + a_5 \times K_5$$

问题：Attention层是词袋模型



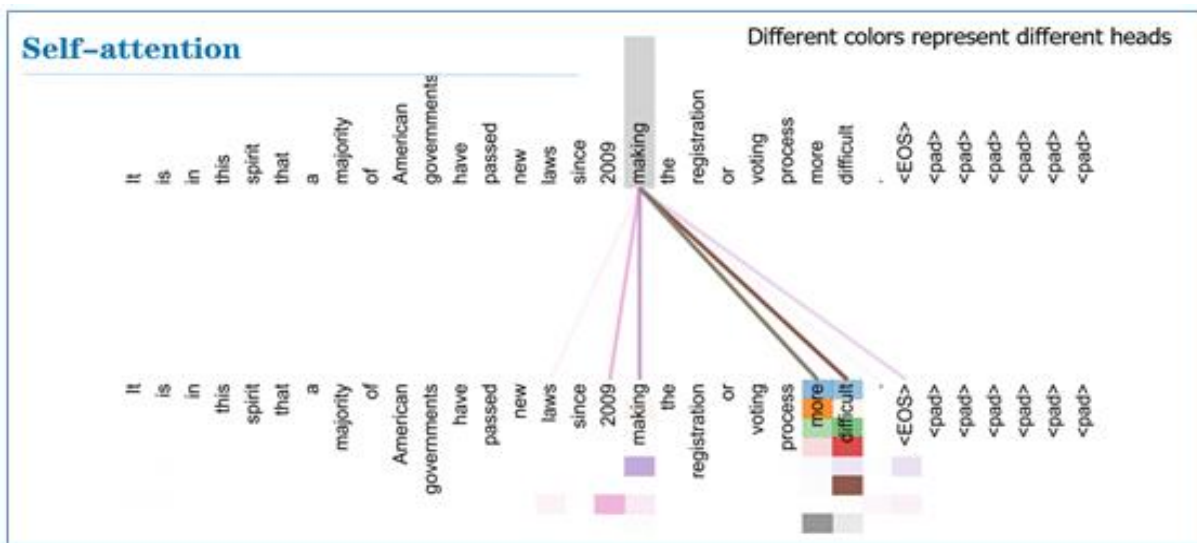
## 5.3 注意力编码机制

自注意力编码可并行计算



## 5.3 注意力编码机制

### 自注意力可视化的效果



可以看到self-attention在这里可以学习到句子内部长距离依赖"making.....more difficult"这个短语

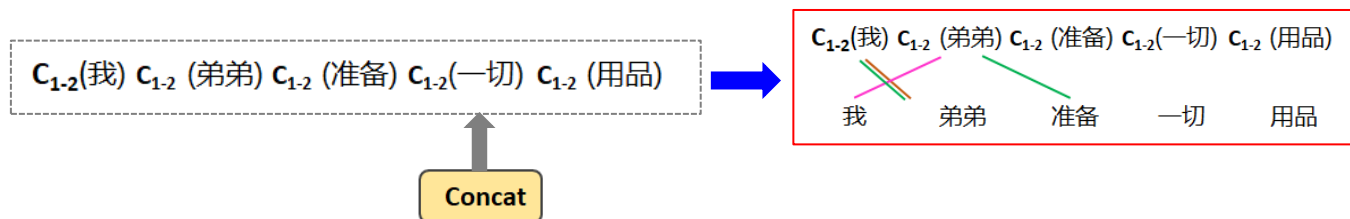
## 5.3 注意力编码机制

### ◆ 编码为一个序列 (相同序列自注意力编码):

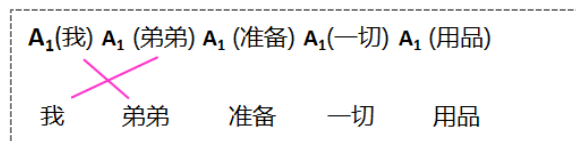
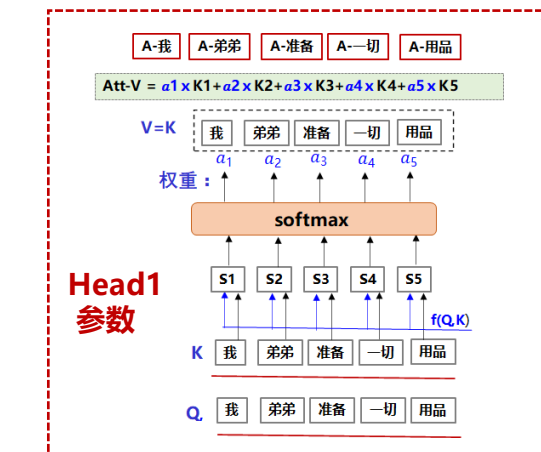
#### • 多头注意力机制 (Multi-Head Attention)

多头 (Multi-Head) 就是做多次同样的事情 (参数不共享)，然后把结果拼接

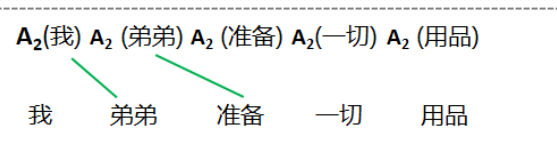
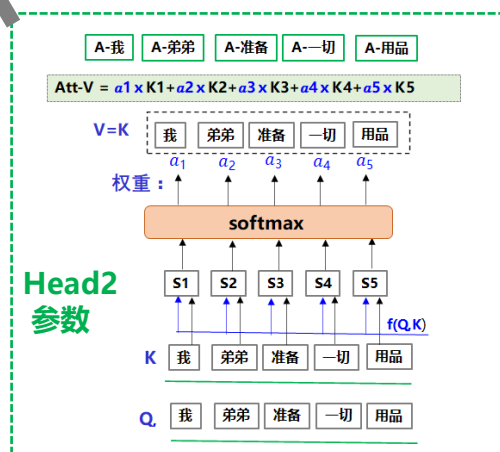
#### ◆ Multi-Head



#### ◆ Head1



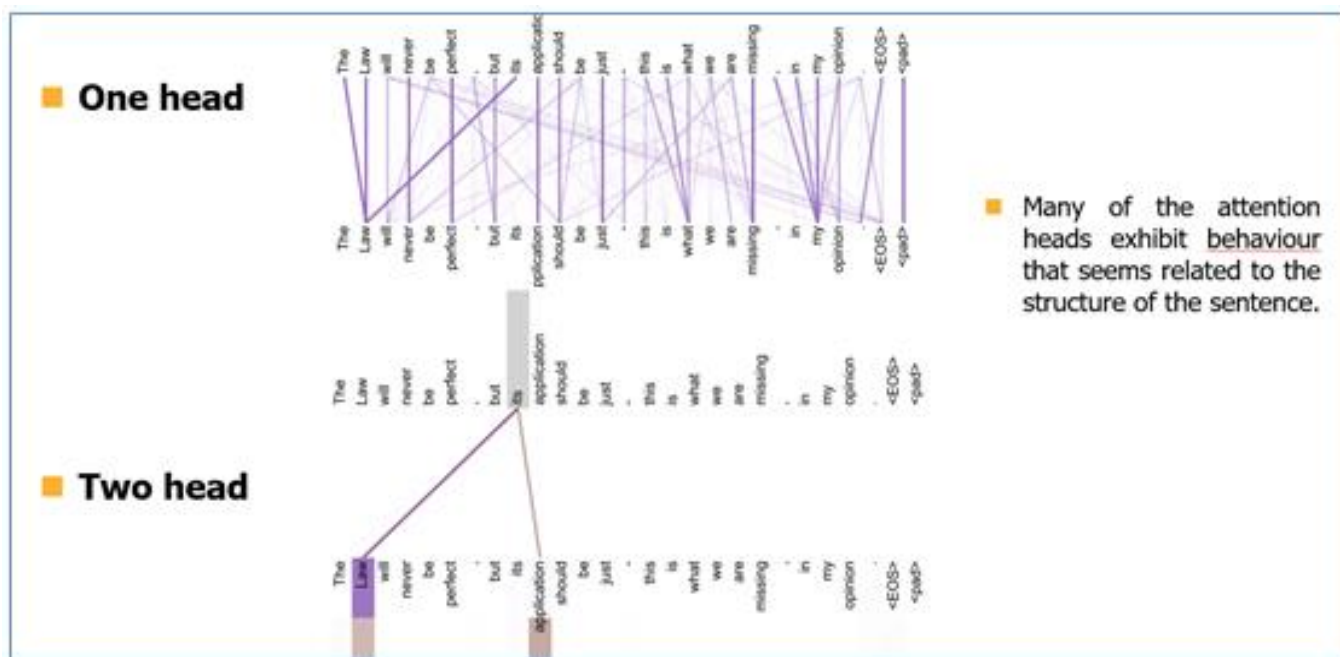
#### ◆ Head2





## 5.3 注意力编码机制

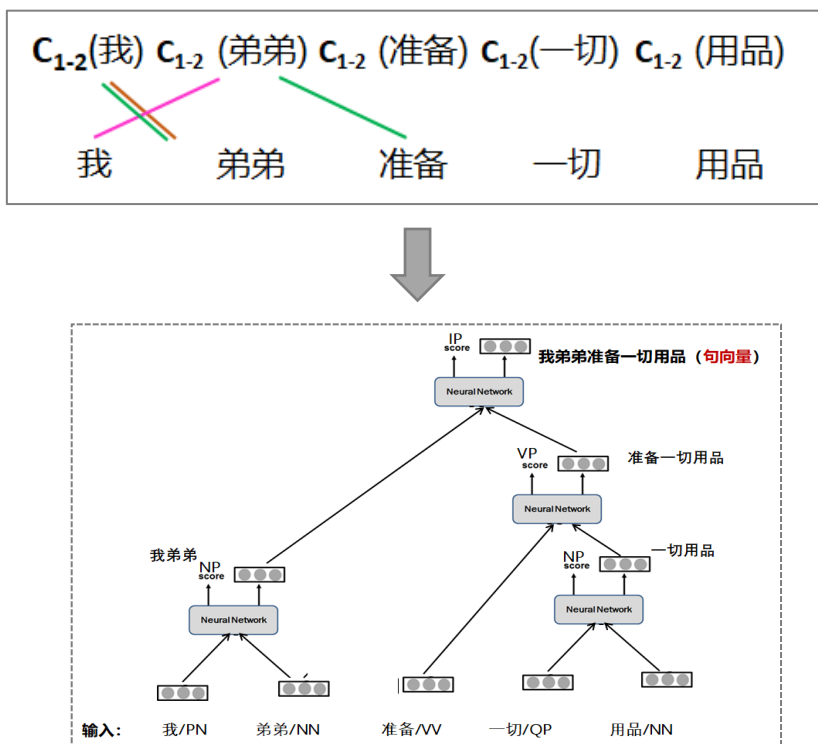
### 多头自注意力的可视化的效果



在两个头和单头的比较中，可以看到单头"its"这个词只能学习到"law"的依赖关系，而两个头"its"不仅学习到了"law"还学习到了"application"依赖关系。多头能够从不同的表示子空间里学习相关信息

## 5.3 注意力编码机制

利用多头自注意力编码对一个句子编码可以起到类似句法分析器的作用



注意力机制典型应用见 Transformer

## 参考文献:

---

张俊林, 深度学习中的注意力机制(2017版),  
<https://blog.csdn.net/malefactor/article/details/78767781>

苏剑林, 《Attention is All You Need》浅读 (简介+代码)  
<https://kexue.fm/archives/4765>

<https://blog.csdn.net/Mbx8X9u/article/details/79908973>

[https://www.sohu.com/a/242214491\\_164987](https://www.sohu.com/a/242214491_164987)

<http://xiaosheng.me/2018/01/13/article121/#ii-attention层>

<https://cloud.tencent.com/developer/article/1086575>

<https://blog.csdn.net/guoyuhaoaaa/article/details/78701768>

[https://blog.csdn.net/sinat\\_31188625/article/details/78344404](https://blog.csdn.net/sinat_31188625/article/details/78344404)

李宏毅课程[http://speech.ee.ntu.edu.tw/~tlkagk/courses\\_ML16.html](http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML16.html)

**在此表示感谢!**

谢谢！

**Thank you**

