

2021-2022学年秋季学期

自然语言处理

Natural Language Processing



授课教师：胡玥

助 教： 李运鹏

自然语言处理

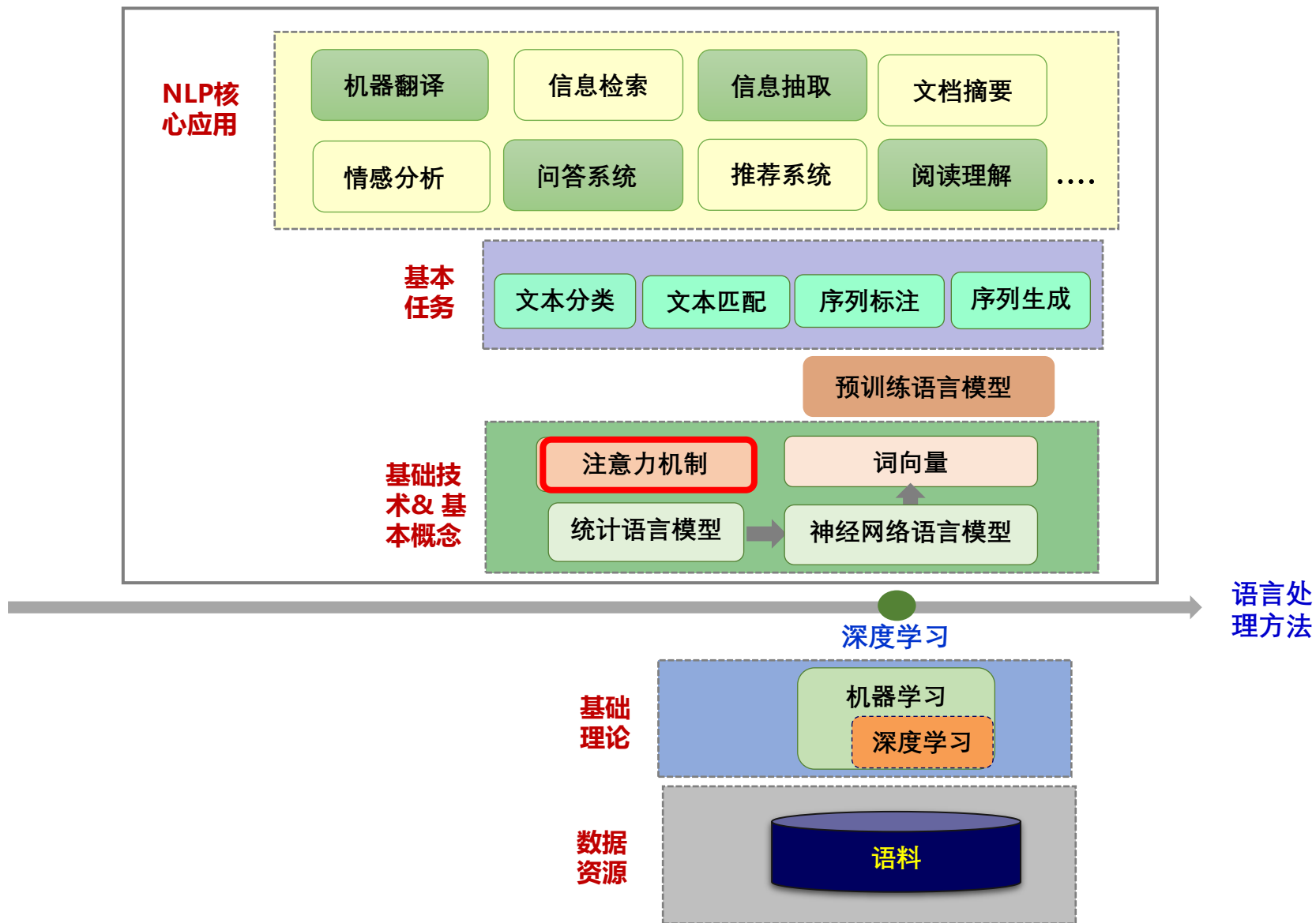
Natural Language Processing

第 5 章 NLP中的注意力机制

授课教师：胡玥

授课时间：2021.10

基于深度学习的自然语言处理课程内容



第 5 章 NLP中的注意力机制

概 要

本章主要内容：

介绍自然语言处理中的注意力机制的内部结构，以及其传统用法和作为编码方式的不同用途。

本章教学目的：

让学生理解并掌握自然语言中注意力机制的含义及不同的用法，并在其他的自然语言处理任务中能够灵活运用。

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

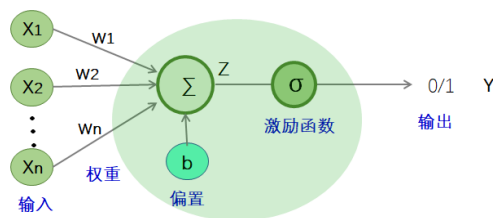
5.1 注意力机制概述

什么是注意力机制？

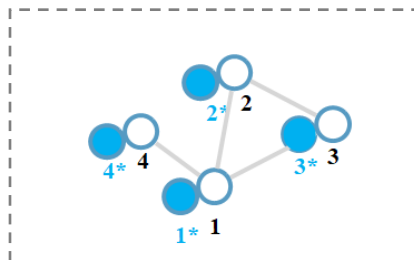
注意力机制 → 加权求和机制/模块

加权求和在神经网络里非常普遍

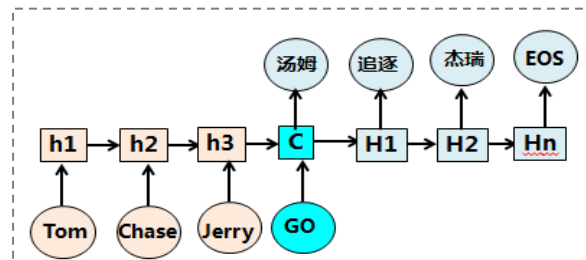
微观如：



宏观如：



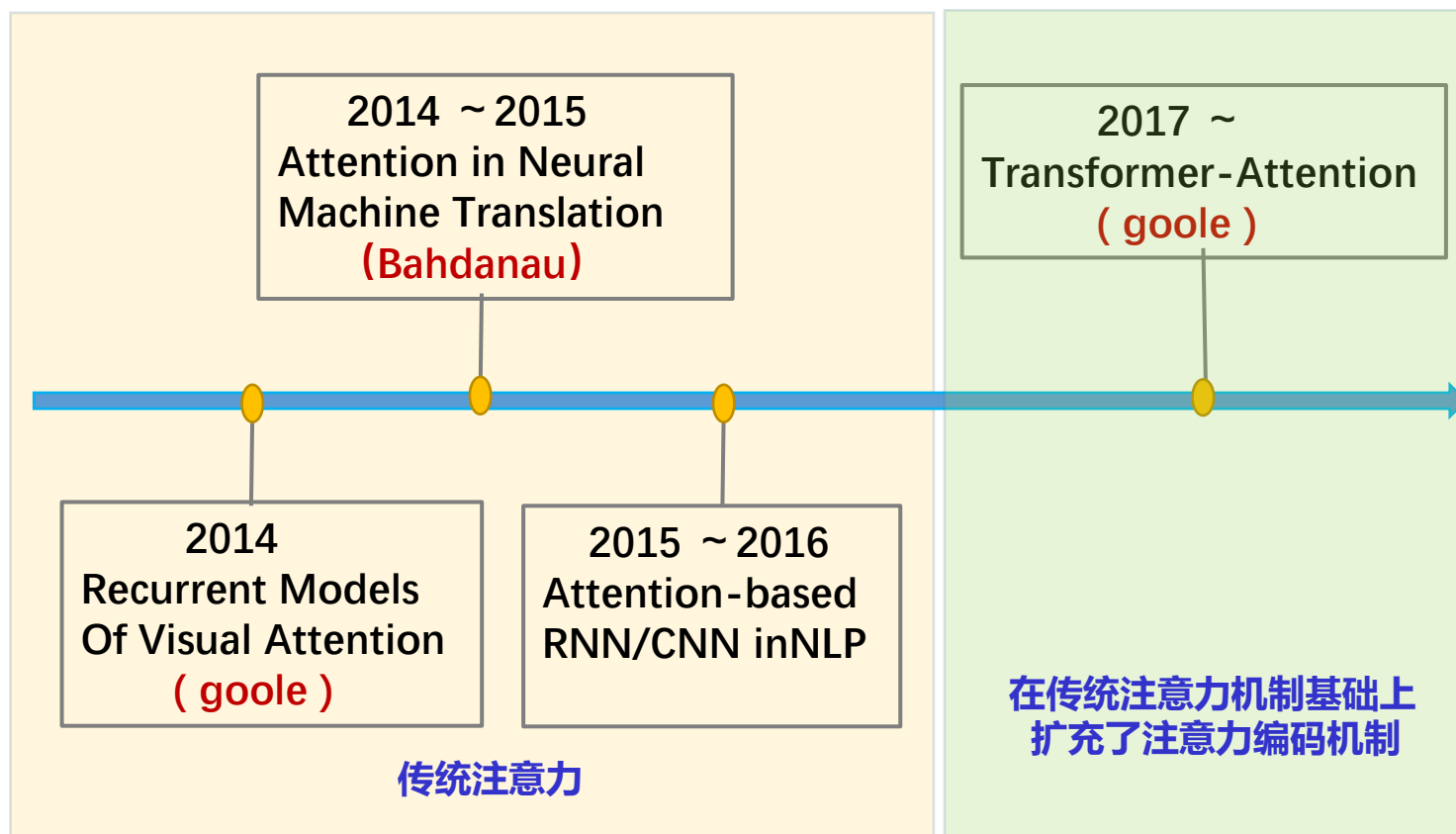
图卷积的邻接节点聚集



机器翻译

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

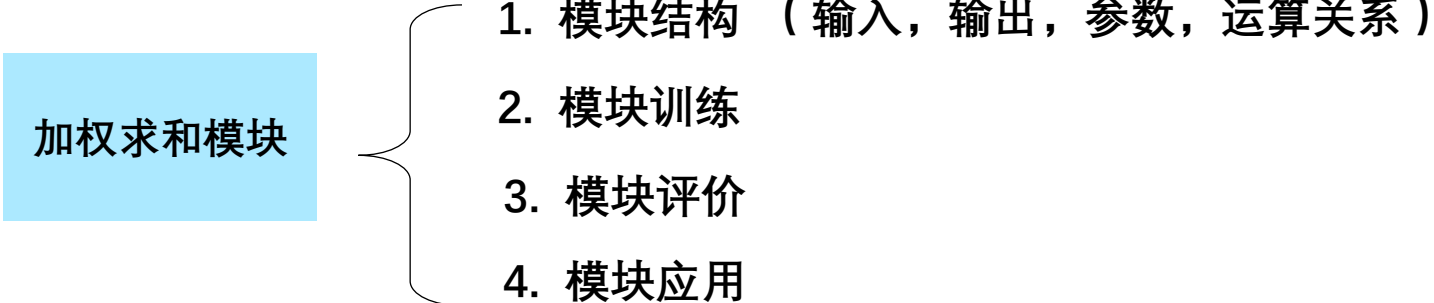
5.3 注意力编码机制

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。

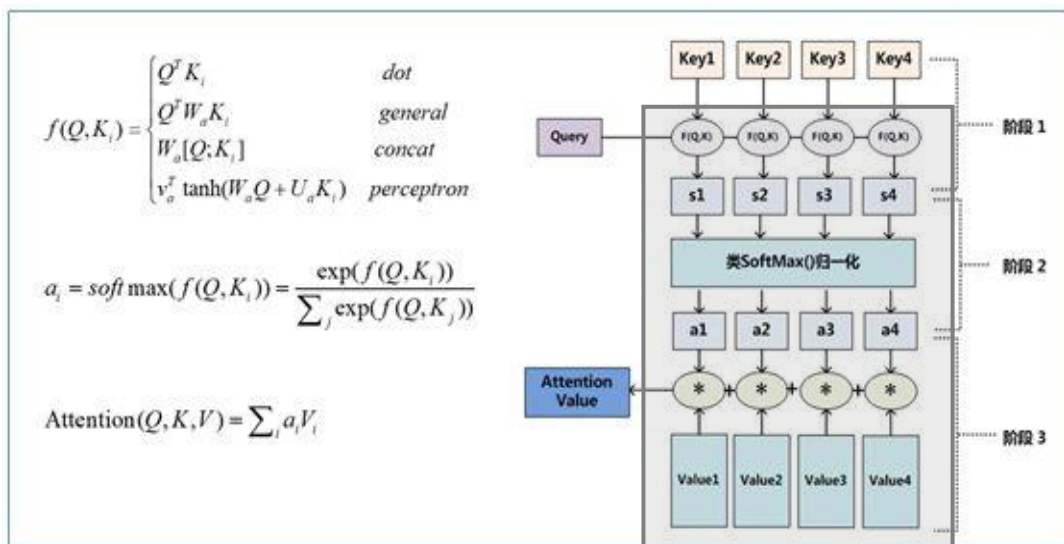
加权求和模块

- 
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. 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)

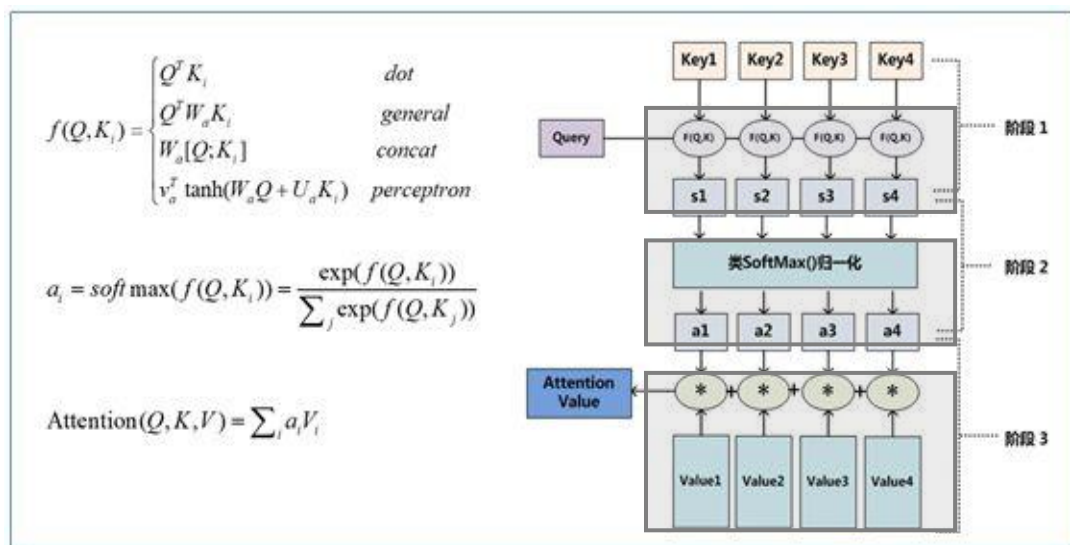
输出: Att-V

功能: 对于集合 K, 求相对 Q 各个元素的权重, 然后按权重相加形成 Q 要的结果

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)

输出: Att-V

步骤1: 计算 $f(Q, K_i)$

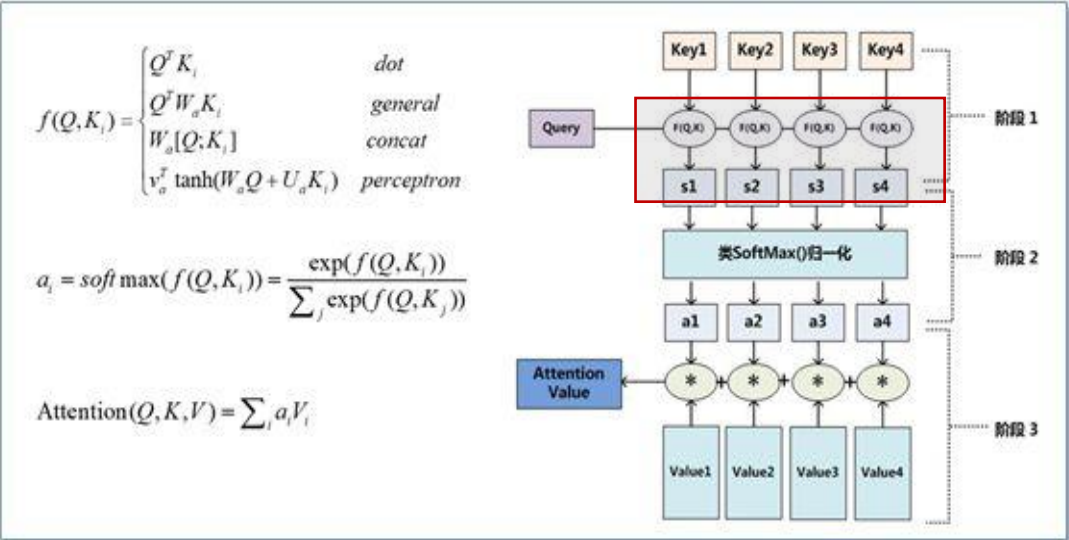
步骤2: $\text{softmax}(f(Q, K_i))$ (计算对于Q 各个 K_i 的权重)

步骤3: 计算输出 (各 K_i 乘以自己的权重, 然后求和)

5.2 传统注意力机制

1. 注意力模块结构:

步骤1: 计算 $f(Q, K_i)$

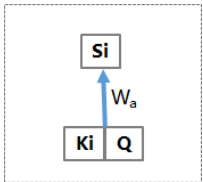


输入: $Q, K(\text{集合})$
输出: Att-V

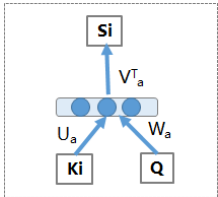
设注意力打分函数 $S = f(Q, K)$

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

$W_a[Q, K_i]$



$V_a^T \tanh(W_a Q + U_a K_i)$

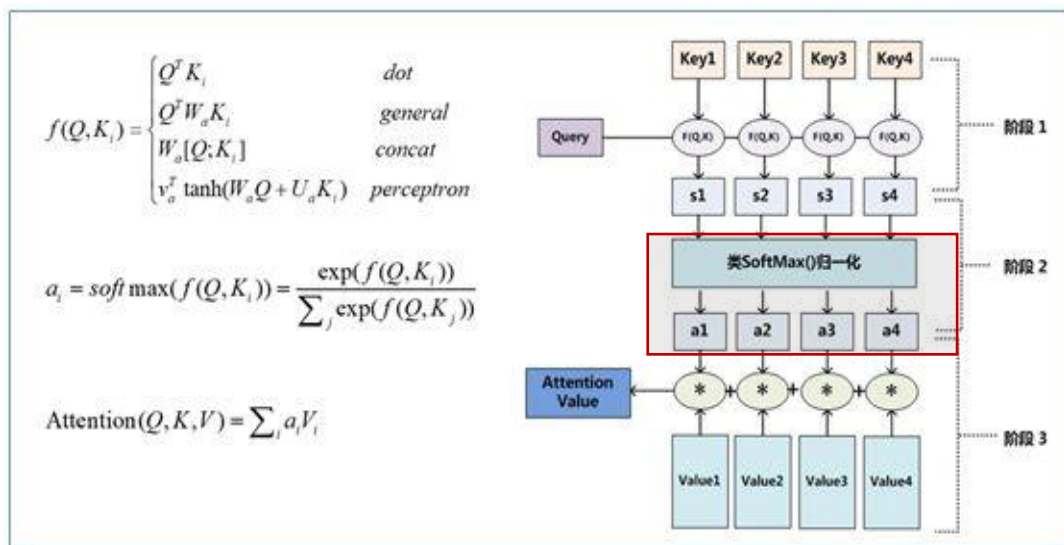


模块参数 ?

5.2 传统注意力机制

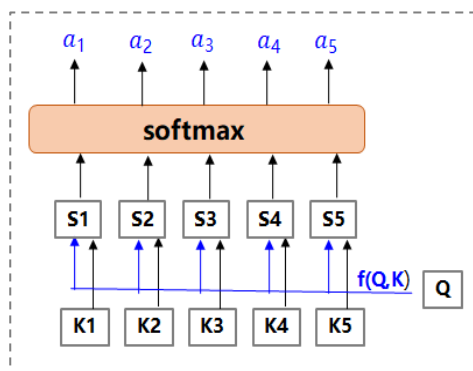
1. 注意力模块结构:

步骤2: 计算对于Q 各个 K_i 的权重



输入: Q, K(集合)

输出: Att-V

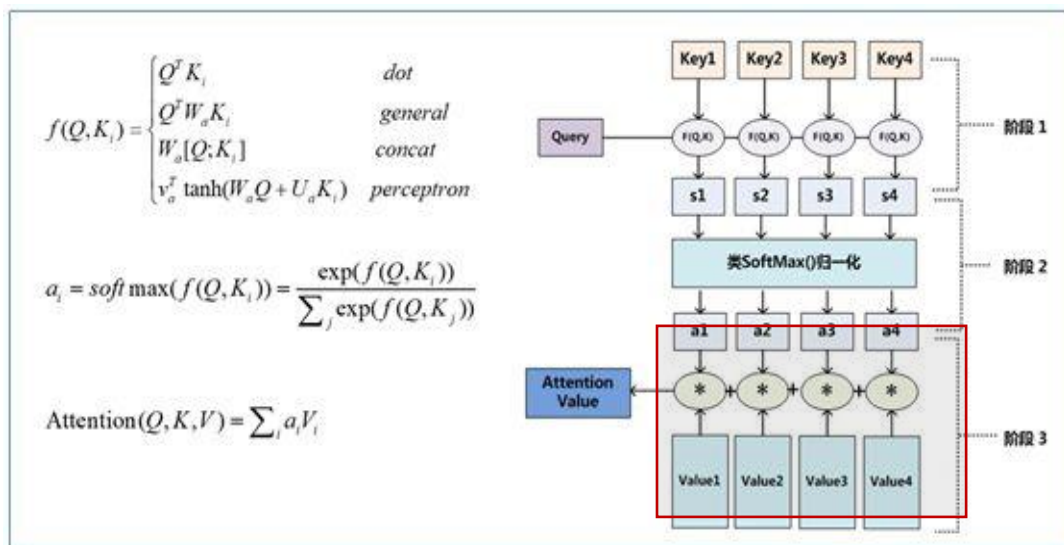


$$a_i = \text{soft max}(f(Q, K_i)) = \frac{\exp(f(Q, K_i))}{\sum_j \exp(f(Q, K_j))}$$

5.2 传统注意力机制

1. 注意力模块结构:

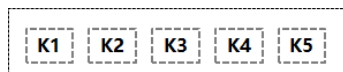
步骤3: 计算输出 Att-V值 (各 K_i 乘以自己的权重, 然后求和)



输入: Q, K(集合)

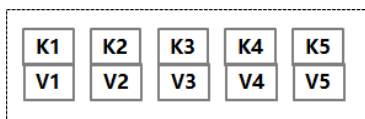
输出: Att-V

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

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

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

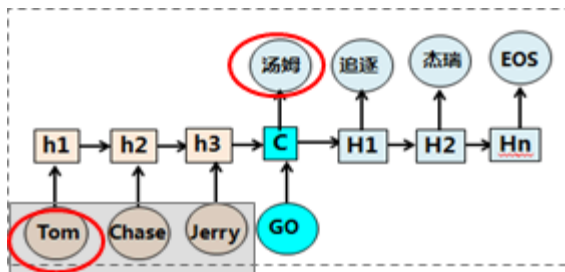
5.2 传统注意力机制

4. 注意力模块应用

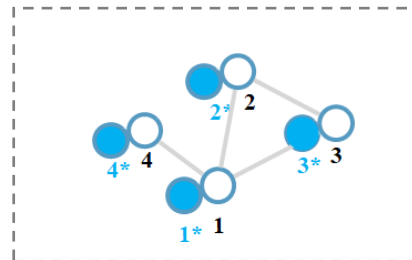
应用场景：网络中有“求和”的地方都可以用，如 图卷积， 机器翻译等

- 优点：**
- 根据不同场景动态选择不同的关注对象
 - 不考虑词之间的距离直接计算依赖关系，提升任务性能

如：



机器翻译

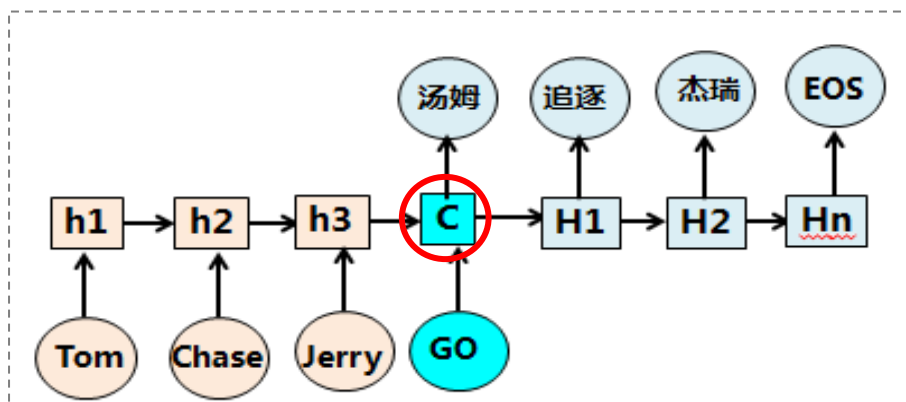


图卷积的邻接节点聚集

作用：等权处理 → 加权处理 → 提升任务效果

5.2 传统注意力机制

例1：机器翻译例



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

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

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

$$y_1 = f(C)$$

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

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

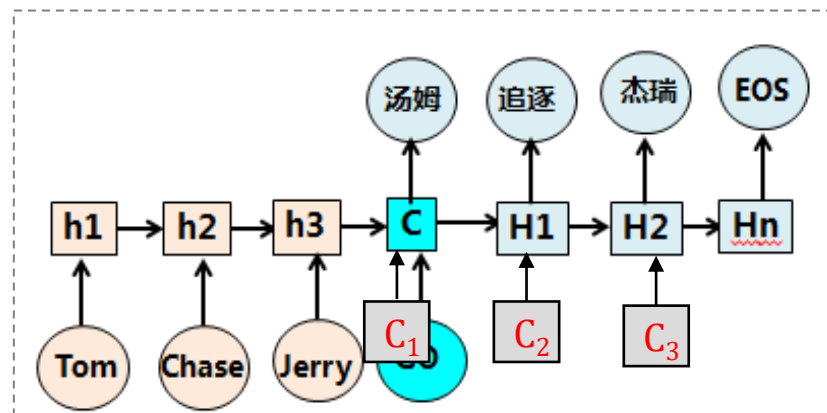
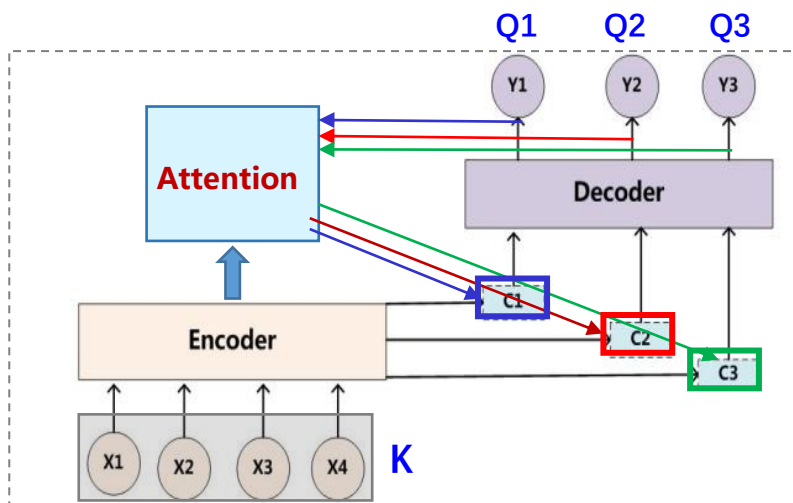
$$y_i = g(C, y_1, y_2 \dots y_{i-1})$$

问题： 对不同的输出 y_i 中间语义表示 C 相同

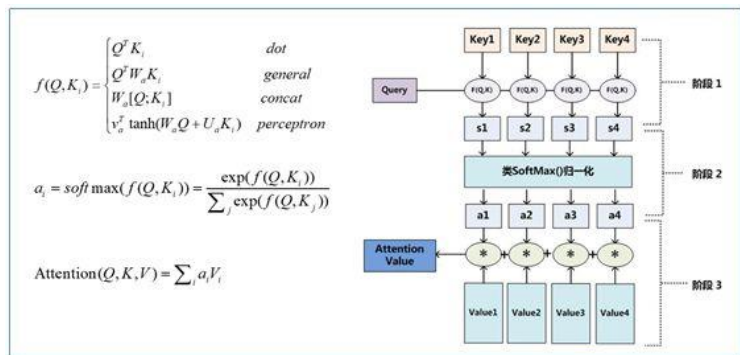
实际应该： 在翻译每个目标语单词时，源语各词对目标词的影响程度是不同的。如翻译“杰瑞”的时候，源语句中各英文单词对于“杰瑞”的影响程度是不同的，如 (Tom,0.3) (Chase,0.2) (Jerry,0.5)

5.2 传统注意力机制

解：引入注意力模块



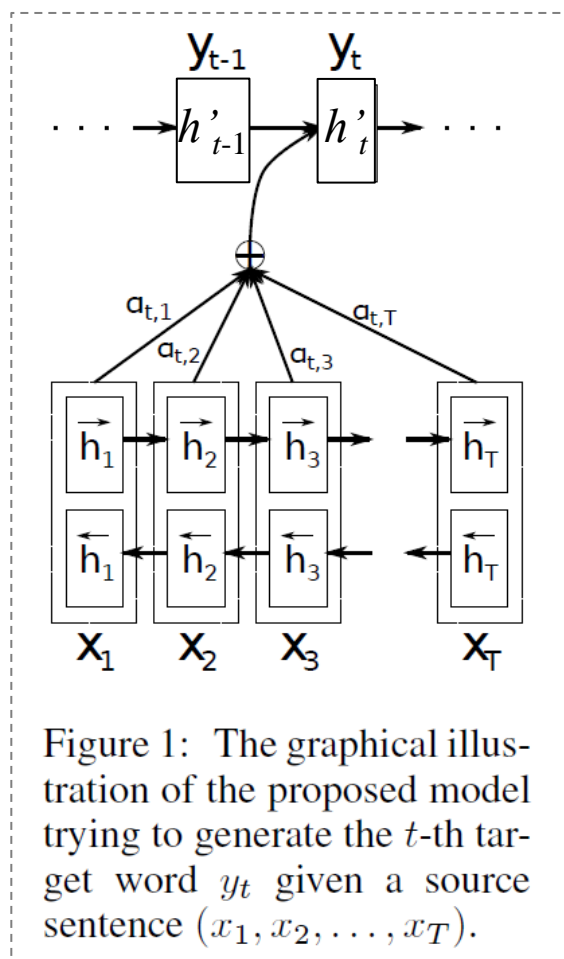
Attention



可实现翻译“杰瑞”的时候，有类似
(Tom,0.3) (Chase,0.2) (Jerry,0.5)
的不同权重

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）

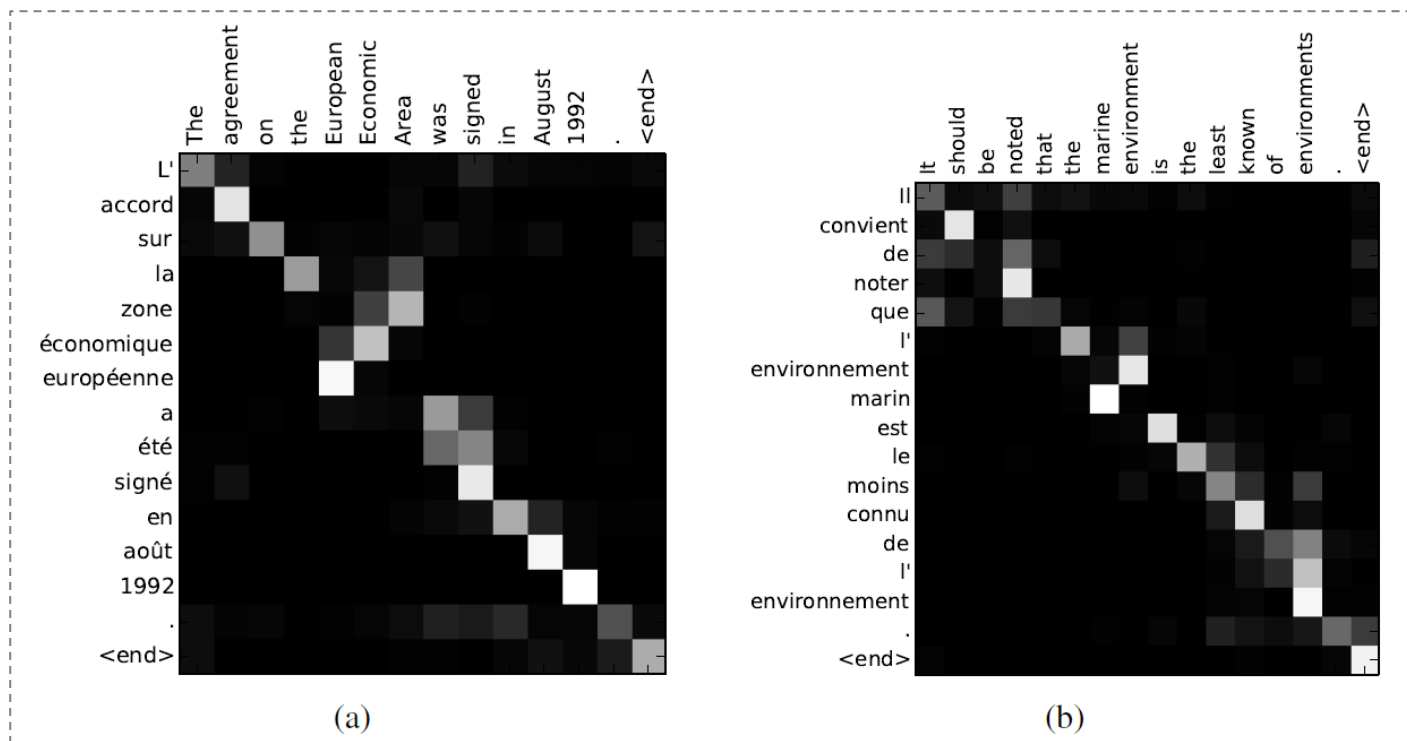
$$p(y_i | y_1, \dots, y_{i-1}, x) = g(y_{i-1}, h_i, c_i)$$

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

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

5.2 传统注意力机制

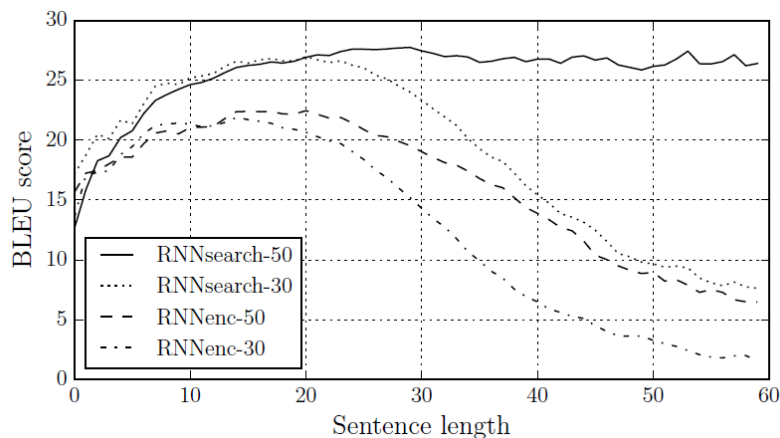
注意力机制可视化效果



注意力机制的双语对齐（英语→法语）效果

5.2 传统注意力机制

注意力机制实验结果

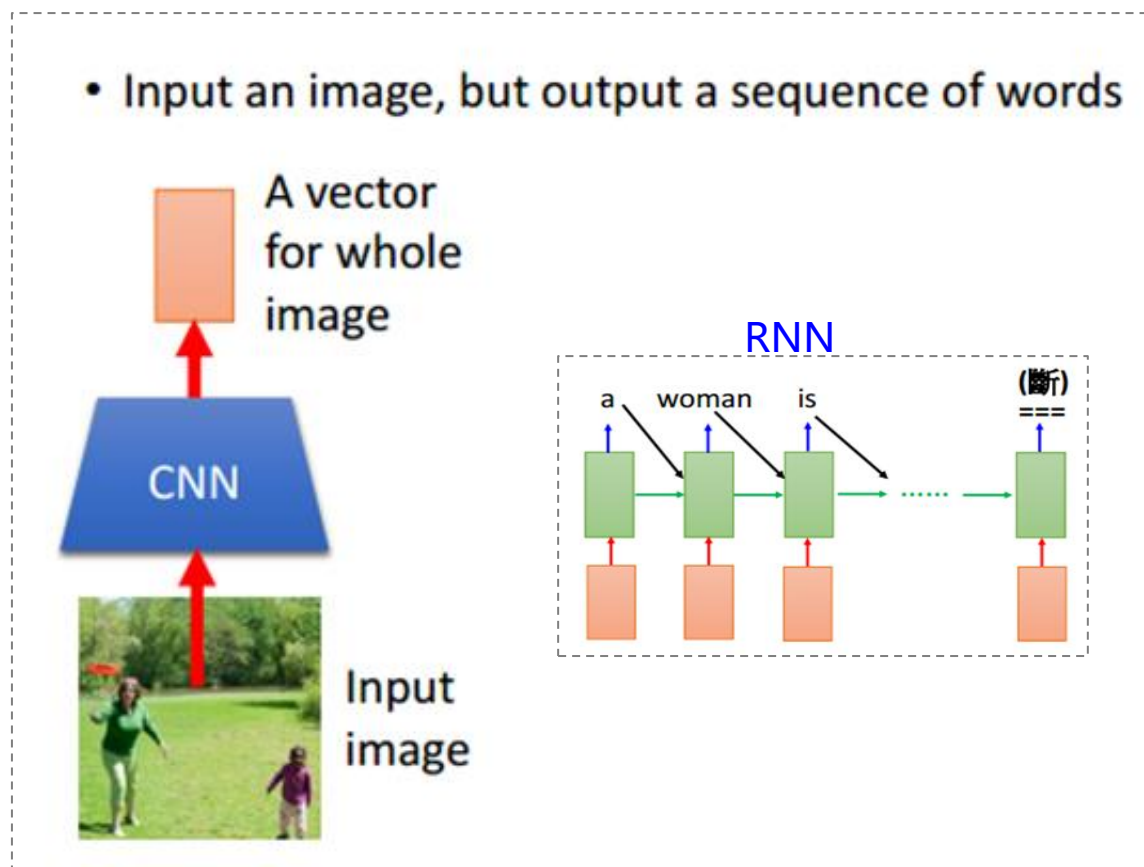


Model	All	No UNK ^o
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50*	28.45	36.15
Moses	33.30	35.63

- 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

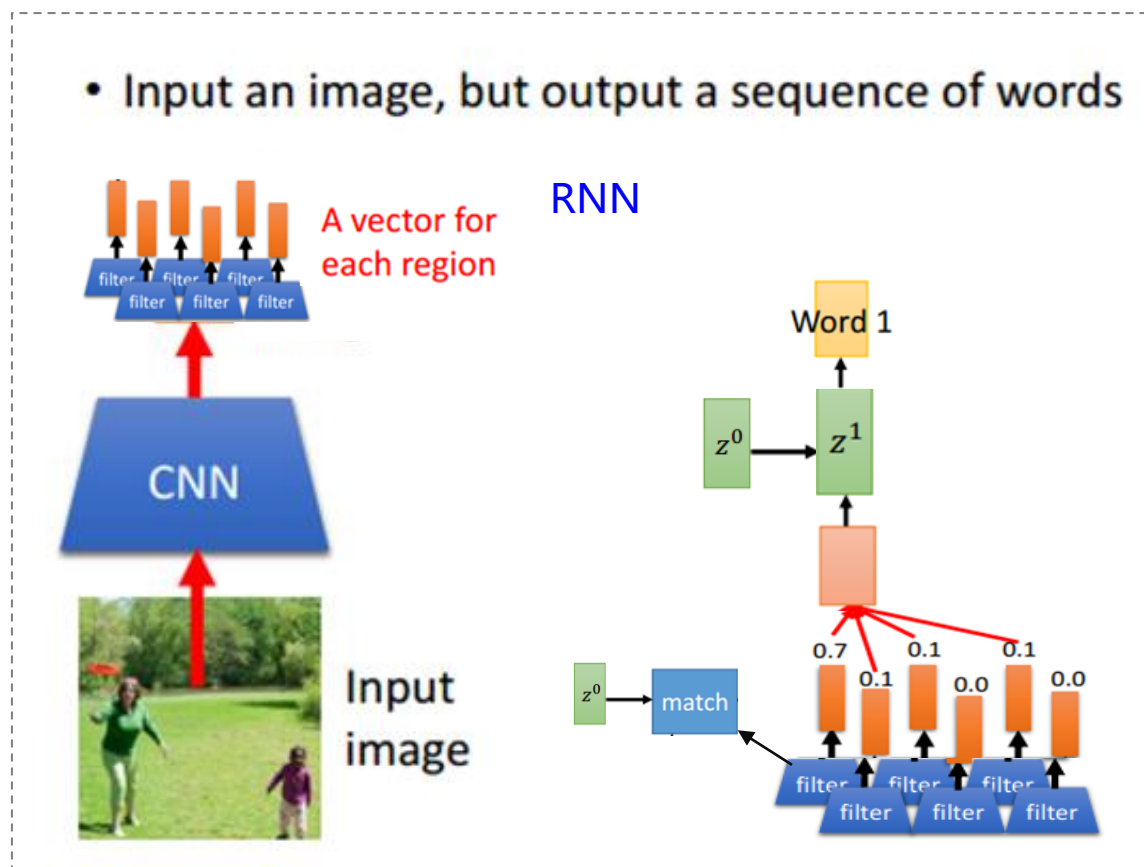
例2： 图片标题生成



A woman is throwing a Frisbee in a park

5.2 传统注意力机制

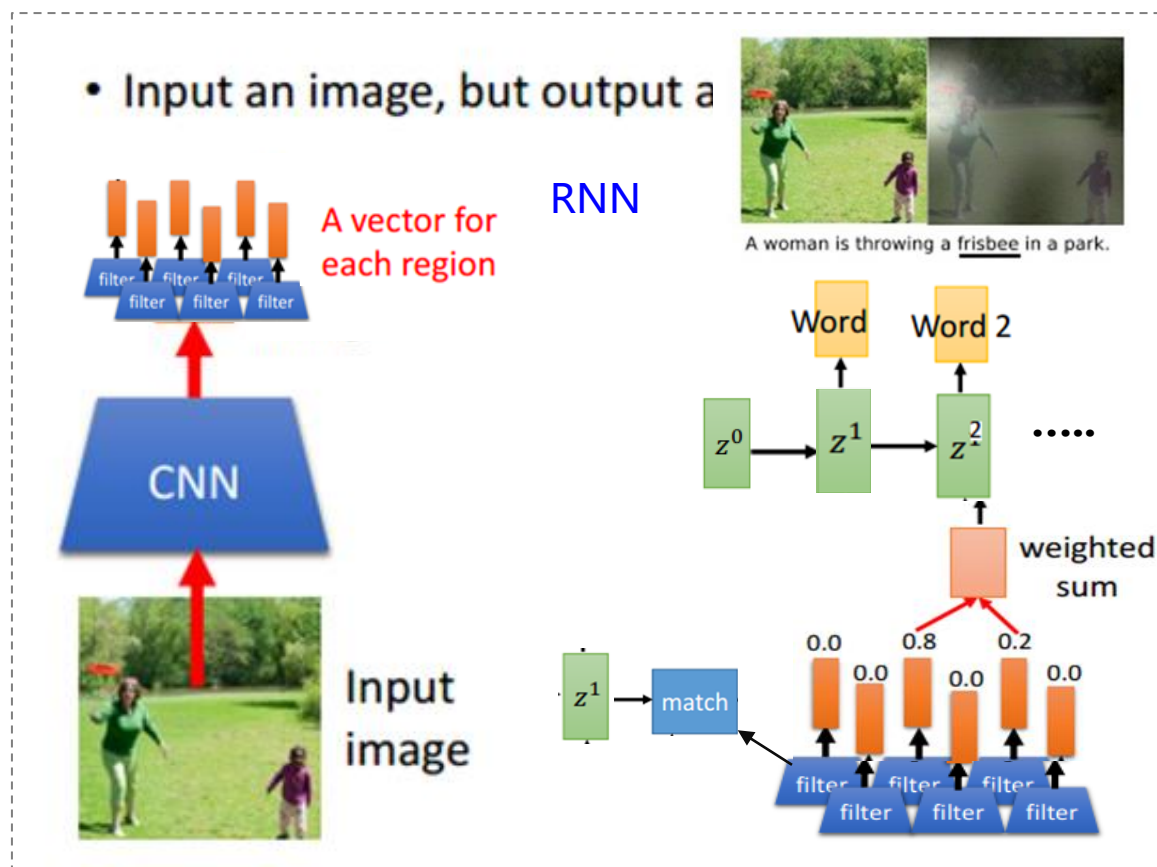
例2： 图片标题生成



A woman is throwing a Frisbee in a park

5.2 传统注意力机制

例2： 图片标题生成



A woman is throwing a Frisbee in a park

5.2 传统注意力机制

图片标题生成实验

- Good captions



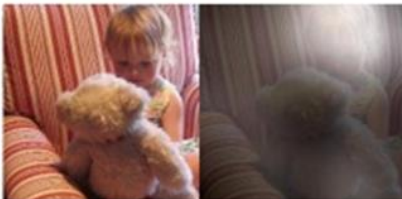
A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.



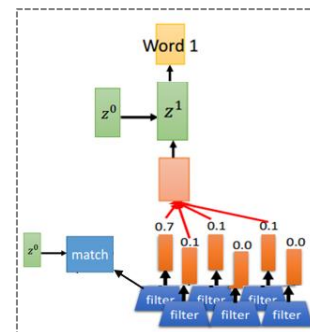
A little girl sitting on a bed with a teddy bear.



A group of people sitting on a boat in the water.



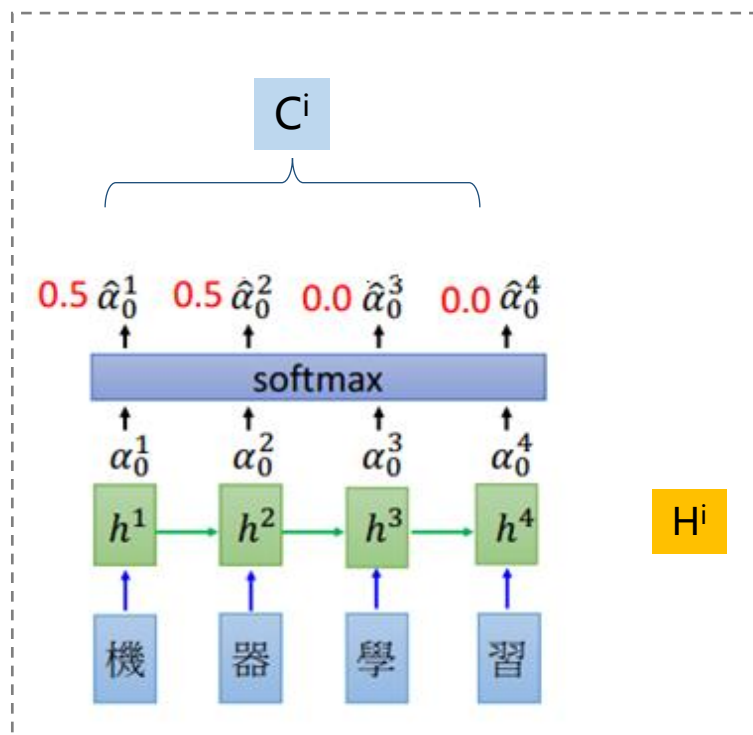
A giraffe standing in a forest with trees in the background.



5.2 传统注意力机制

□ 软注意力 Hard Attention

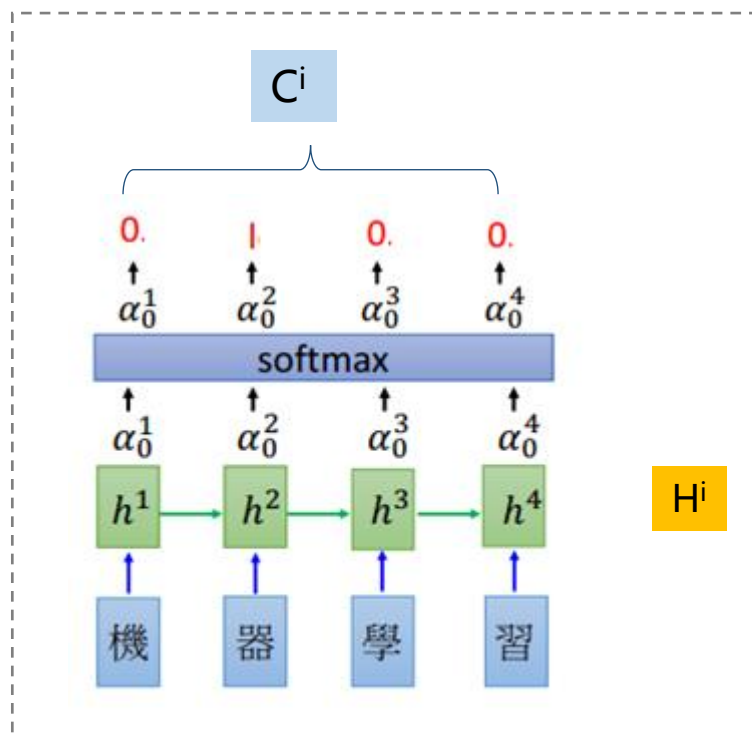
Soft AM: 在求注意力分配概率分布的时候，对于输入句子X中任意一个单词都给出个概率，是个概率分布。



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

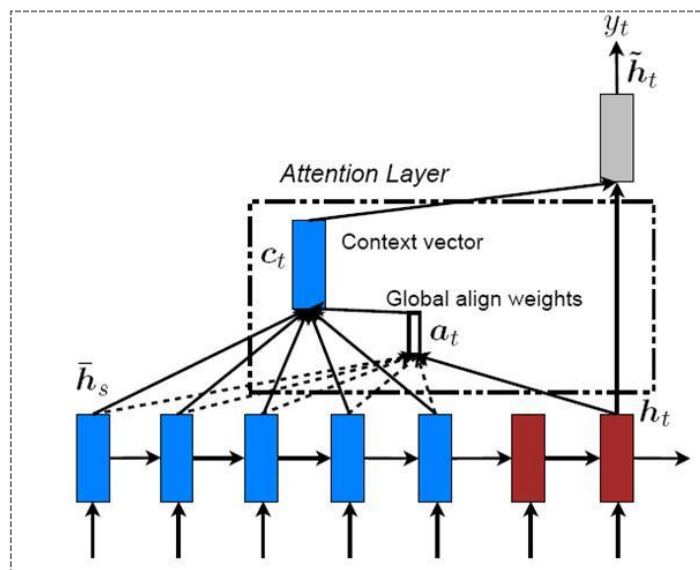


5.2 传统注意力机制

□ 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

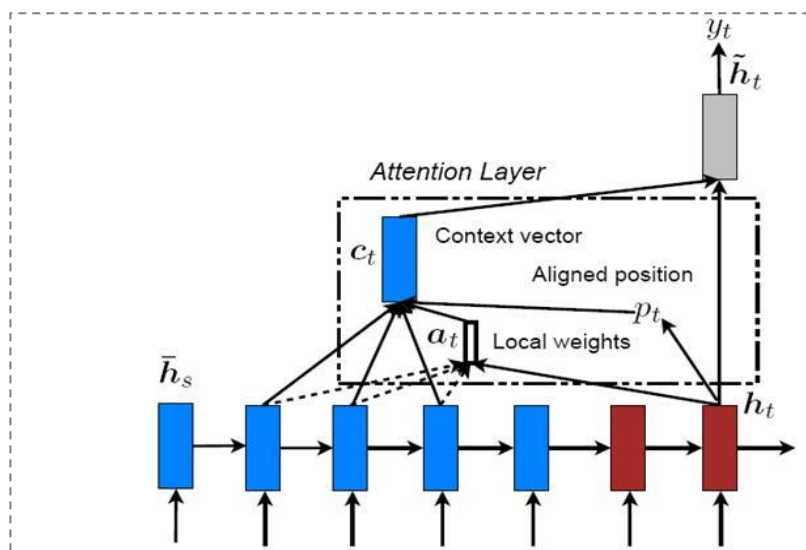
Global Attention Model 是Soft Attention Model



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。



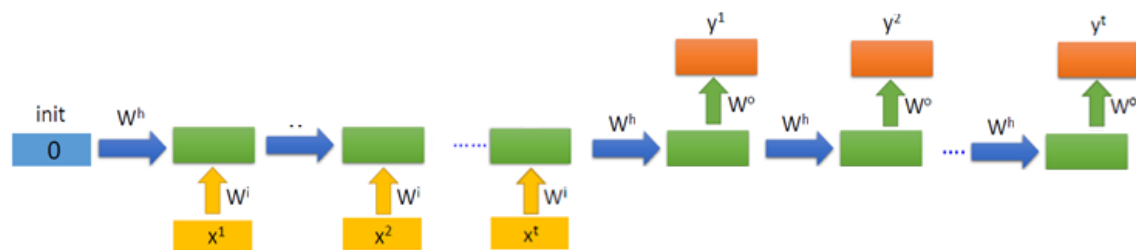
5.2 传统注意力机制

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

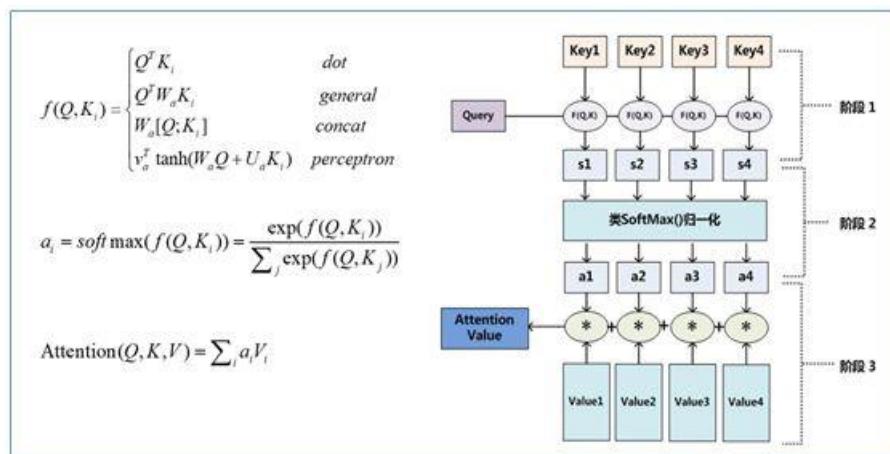
5.3 注意力编码机制

注意力编码机制

注意力机制可以作为一种编码机制，即：通过建立序列各元素之间的关联关系形成一种序列表示（按元素之间关系对序列进行编码）；或通过不同序列元素之间的关联关系形成不同序列间的融合表示。

如：

K: 我爱天安门



Q: X

A-Q

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：

- ◆ **单一向量编码：**将输入序列按规则编码成单一向量表示。如，句表示/篇章表示，某词的上下文表示 等
- ◆ **不同序列间编码：**将2个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
- ◆ **同一序列自编码：**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

◆ 单一向量编码:

通过建立序列K各元素与Q之间的关联关系形成单一向量表示（按元素之间关系对序列进行编码）

★ Q为确定值的句向量编码（句表示）

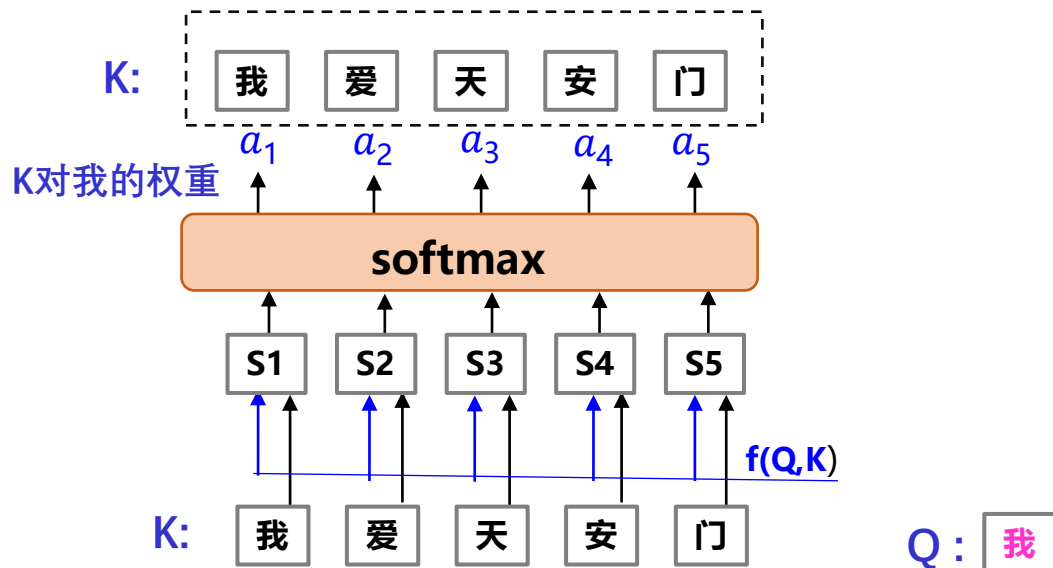
例1:

注意力编码

A-我

A-我 向量的含义？

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

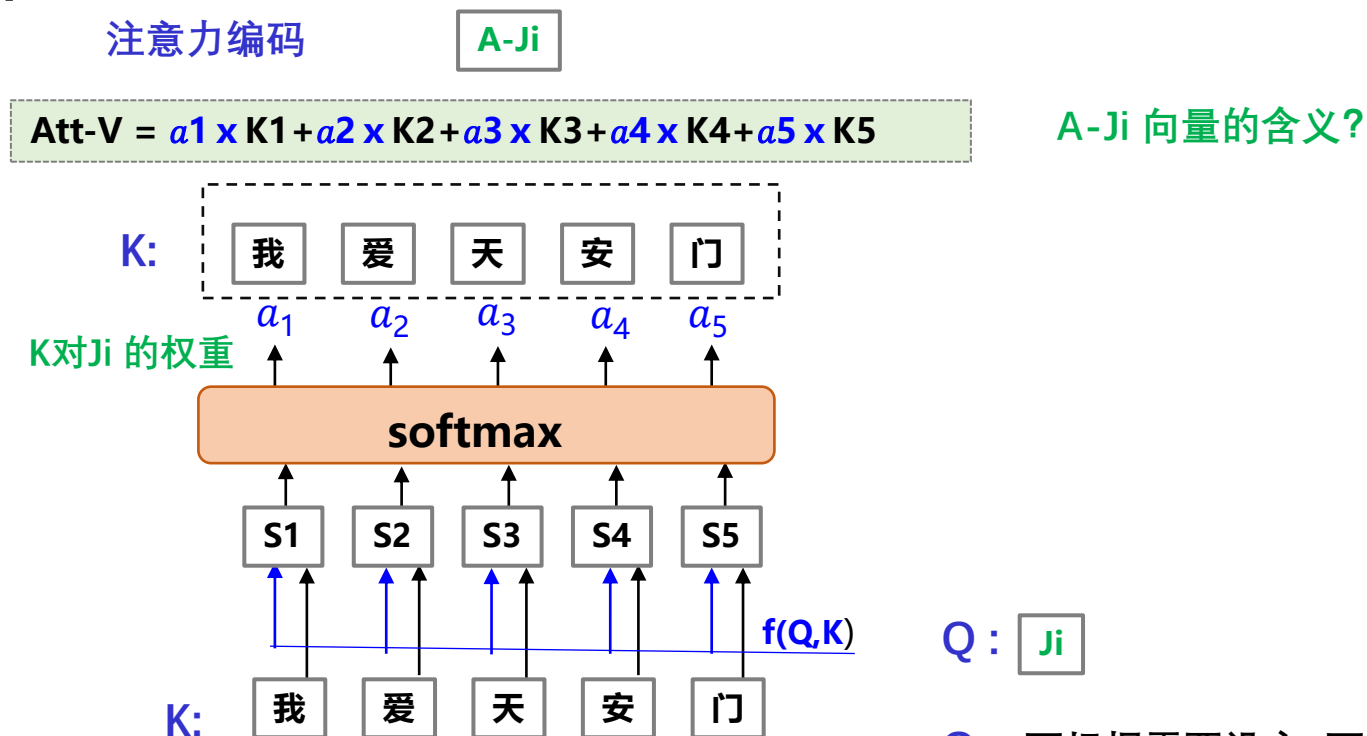


5.3 注意力编码机制

◆ 单一向量编码:

★ Q为隐变量的句向量编码（句表示）

例2:



A-Ji 向量的含义?

Q: Ji

Q: 可根据需要设定, 可以是变量

如, 设 Q 为句向量 Ji

5.3 注意力编码机制

◆ 单一向量编码：

★ 对序列中某元素的真正上下文编码（词编码）

例3： 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



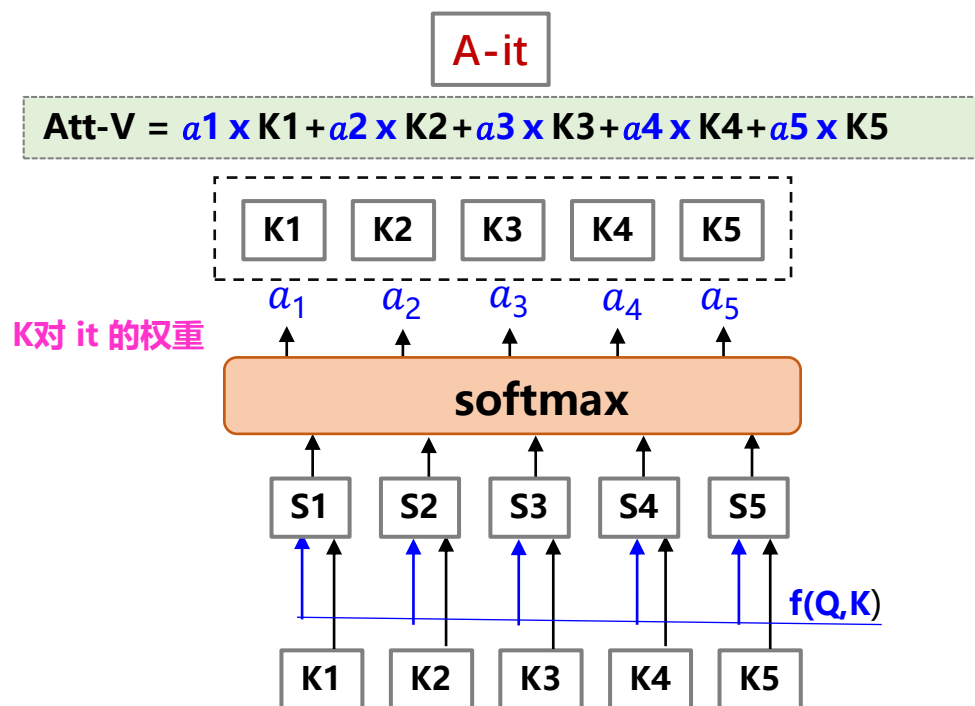
Backward LM: The animal didn't cross the street because **it** was too tired



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

- 采用注意力机制编码词的上下文：



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

用注意力机制编码可以同时看到it的上文 animal, street 和其下文 tired , 所以对 it 的表示信息更加丰富完整, 可以给下游任务提供更丰富的信息

5.3 注意力编码机制

◆ 不同序列间编码：

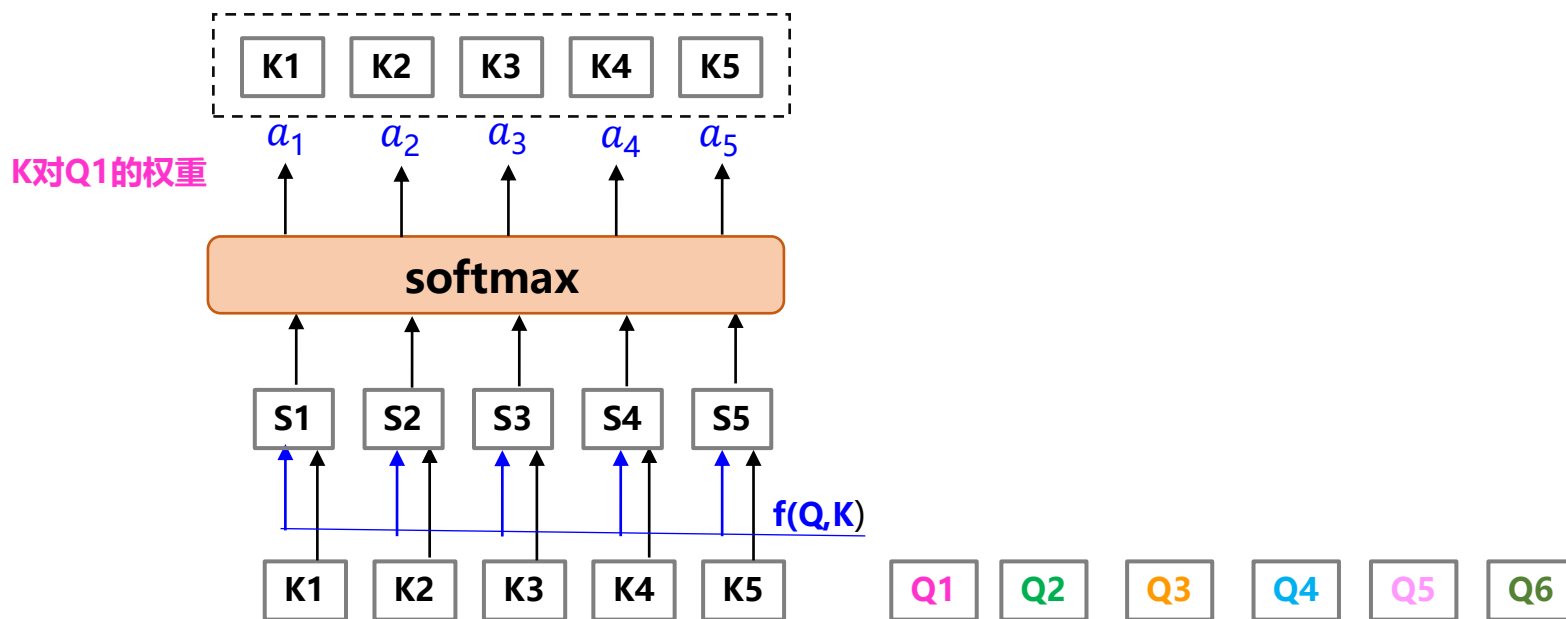
将二个序列编码成二者的融合的代表序列（如，匹配任务和阅读理解任务常用的融合层表示）

例：对K序列和Q序列编码

融合编码层

A-V1

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



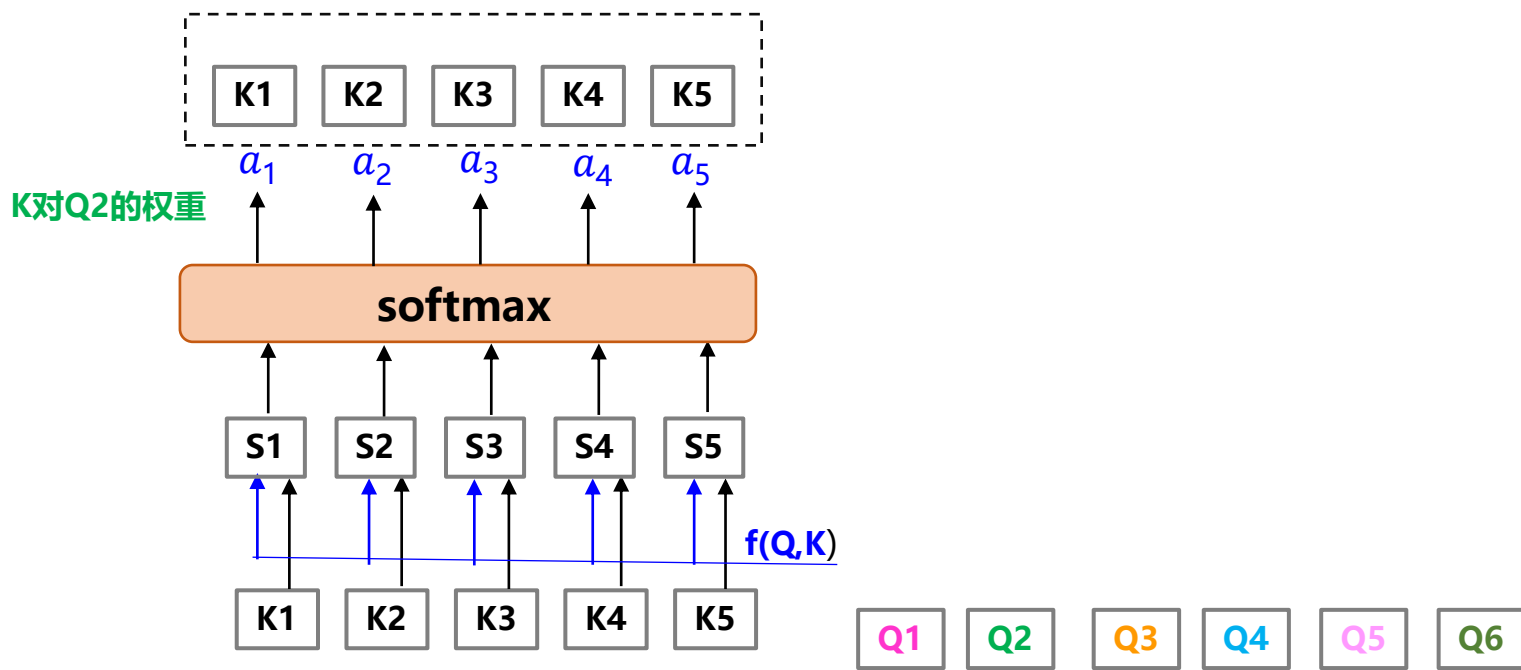
5.3 注意力编码机制

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

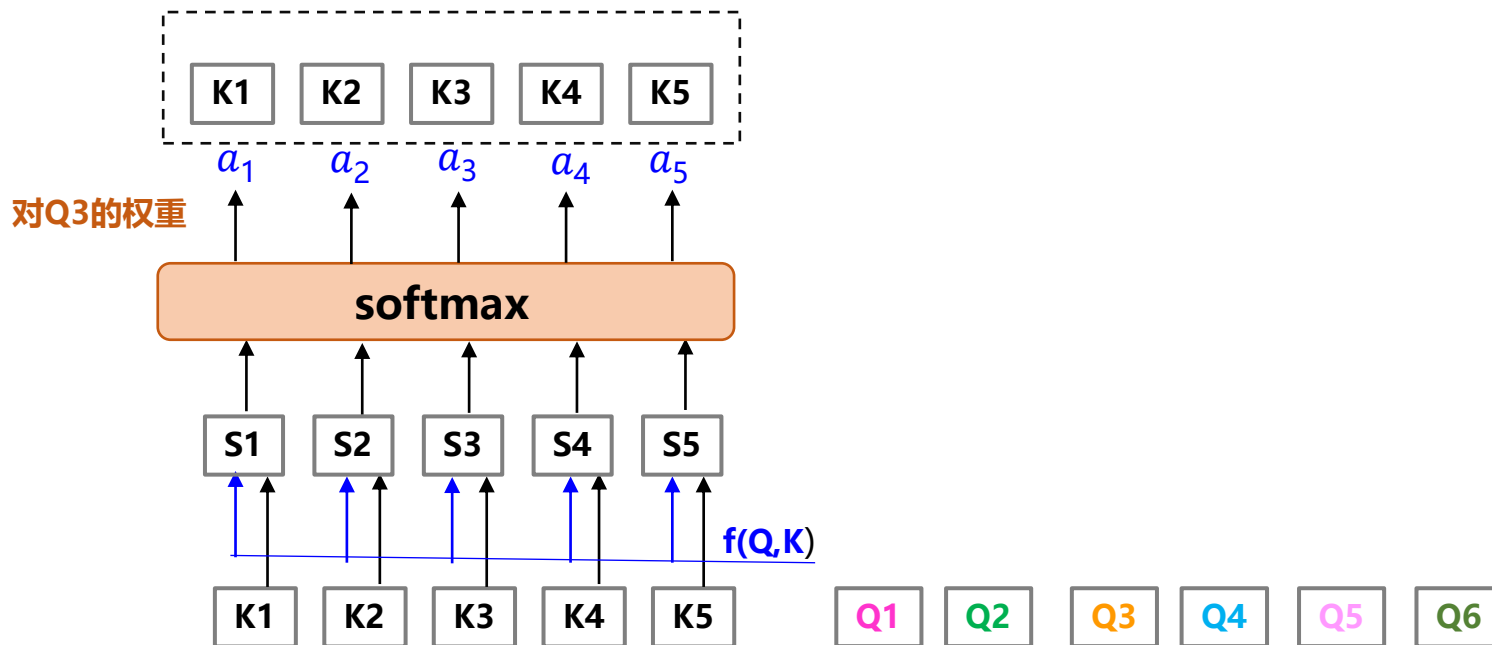
融合编码层

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

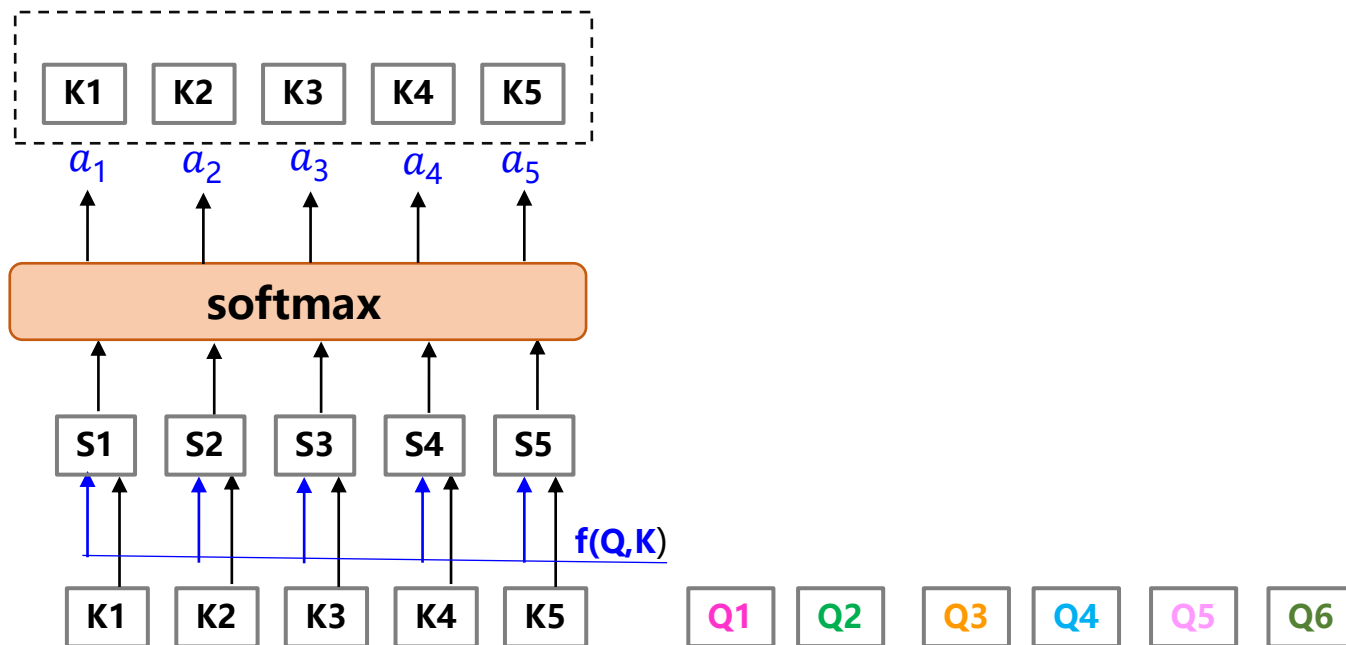
融合编码层



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_i 序列的含义?



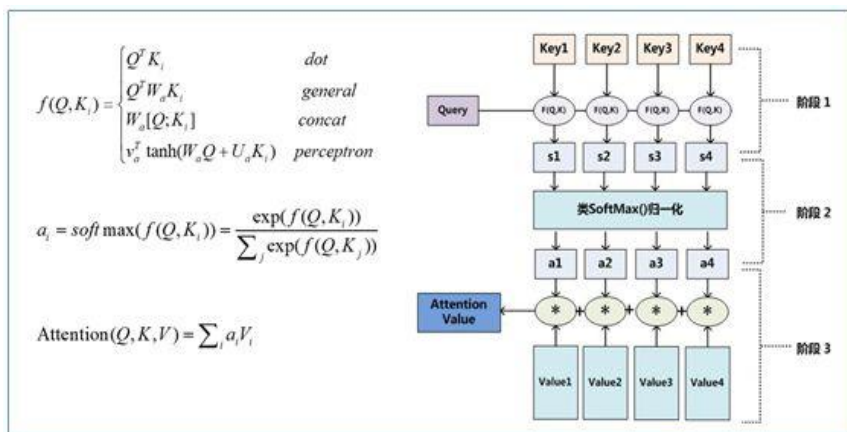
A-V_i 序列元素个数等于 Q 序列元素个数

5.3 注意力编码机制

◆ 同一序列自编码:

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

• 自注意力机制



Attention(Q,K,V)

其中，Q=K=V

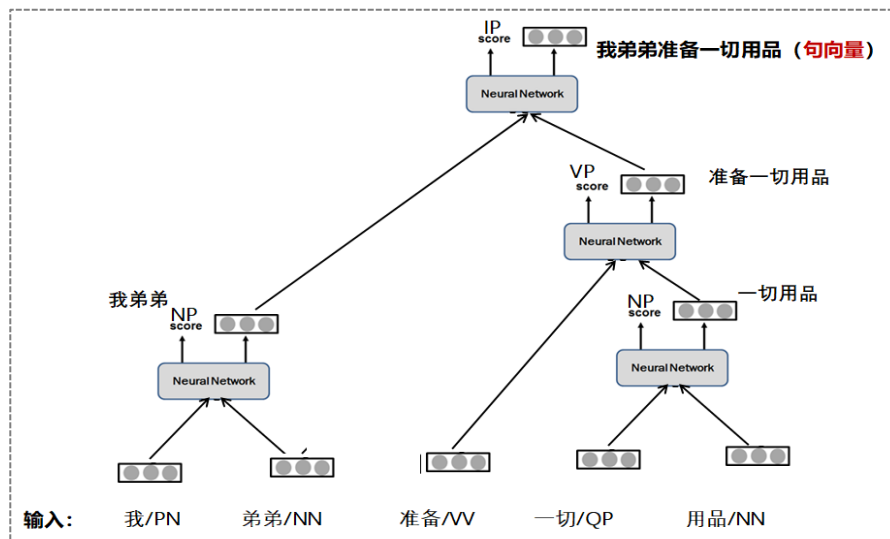
其实就是 Attention(X , X , X), X 为输入序，其含义为在序列内部做 Attention 计算，寻找序列内部词与词之间的关联关系

5.3 注意力编码机制

传统句子句法分析树回顾：

如： 我 弟弟 已经 准备 好 了 一切 用品

用递归神经网络对句子做句法分析生成短语结构树如下：



短语结构树

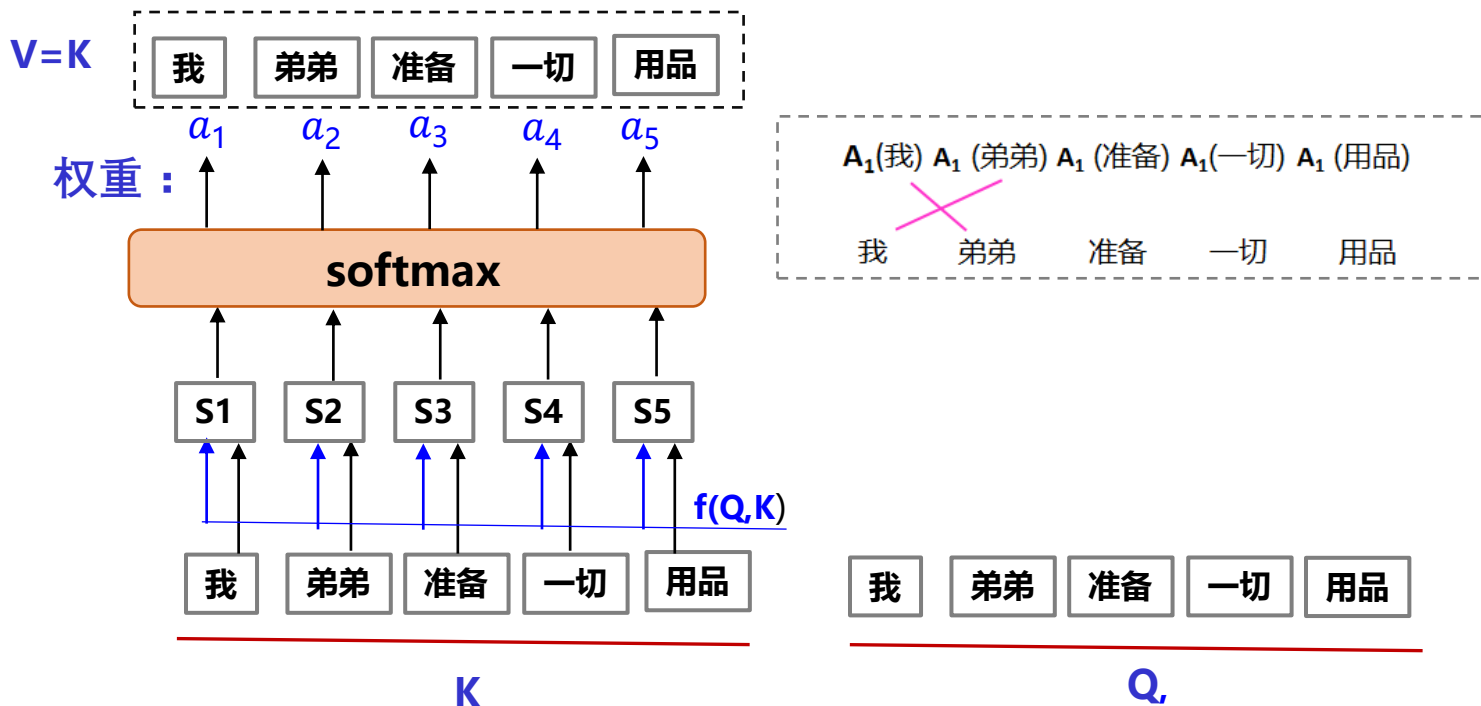
5.3 注意力编码机制

例：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

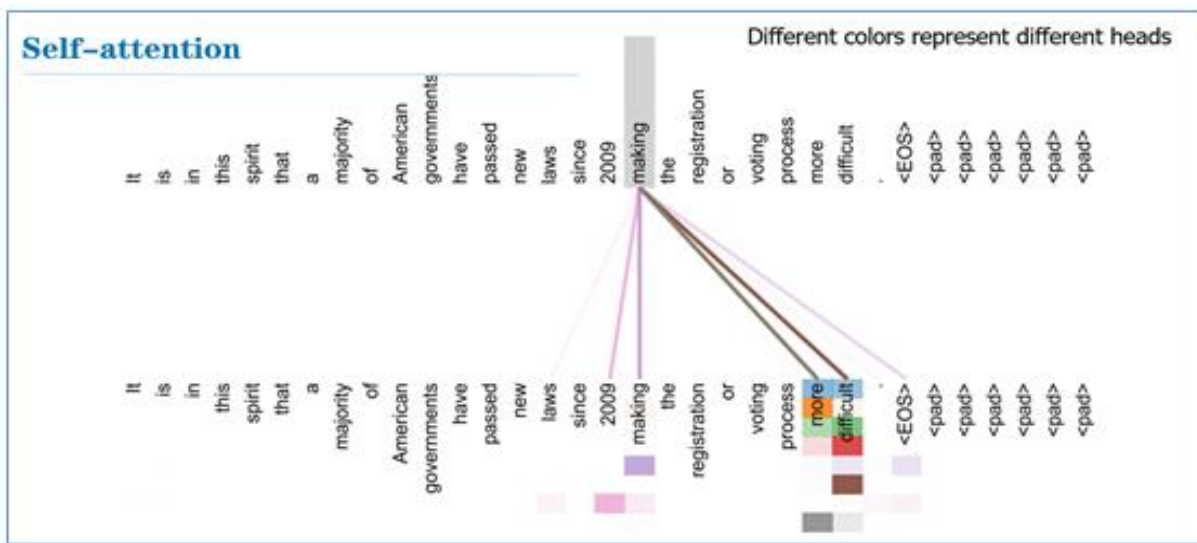
$$\text{Att-V} = a_1 \times K_1 + a_2 \times K_2 + a_3 \times K_3 + a_4 \times K_4 + a_5 \times K_5$$

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

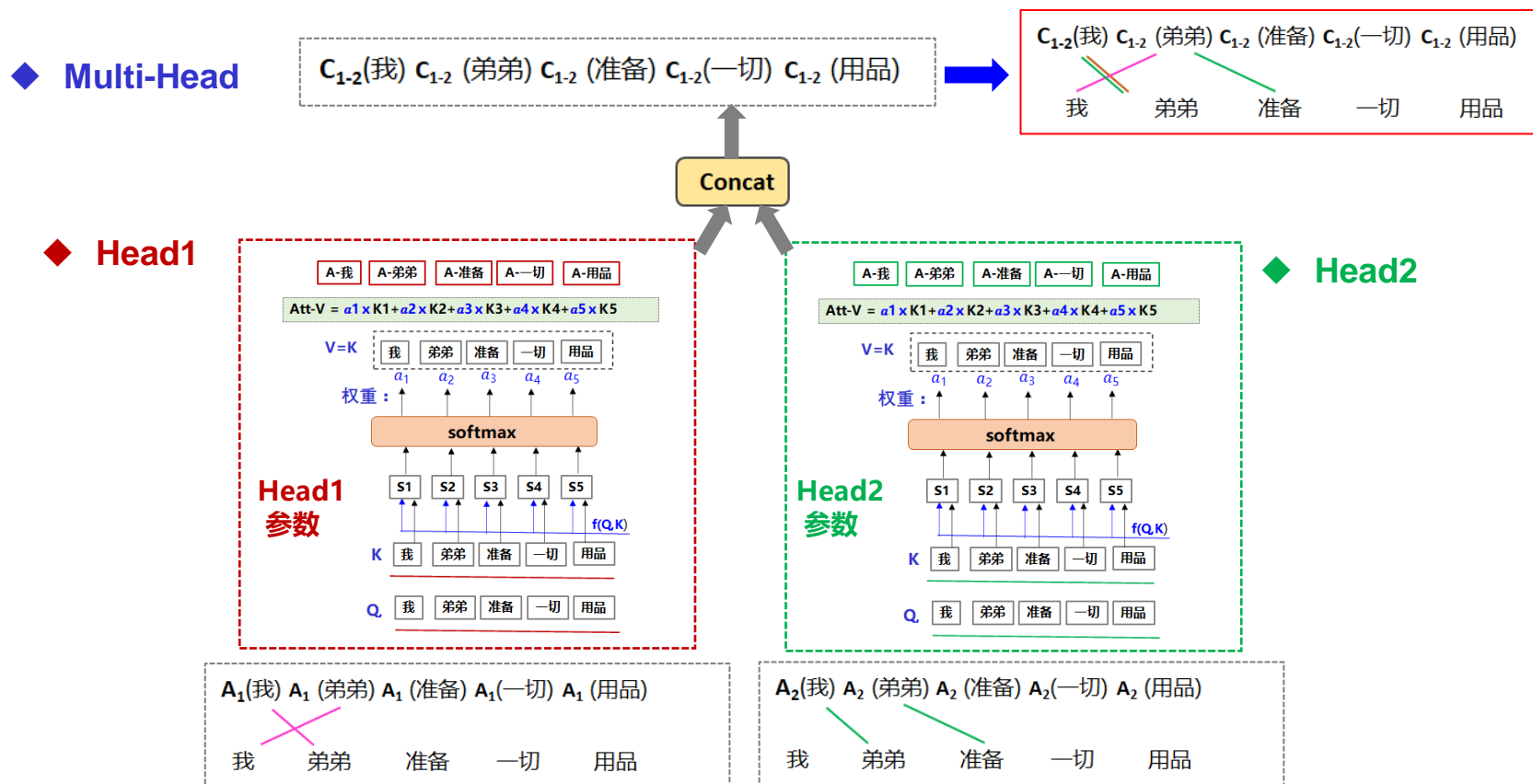


可以看到self-attention在这里可以学习到句子内部长距离依赖"making.....more difficult"这个短语

5.3 注意力编码机制

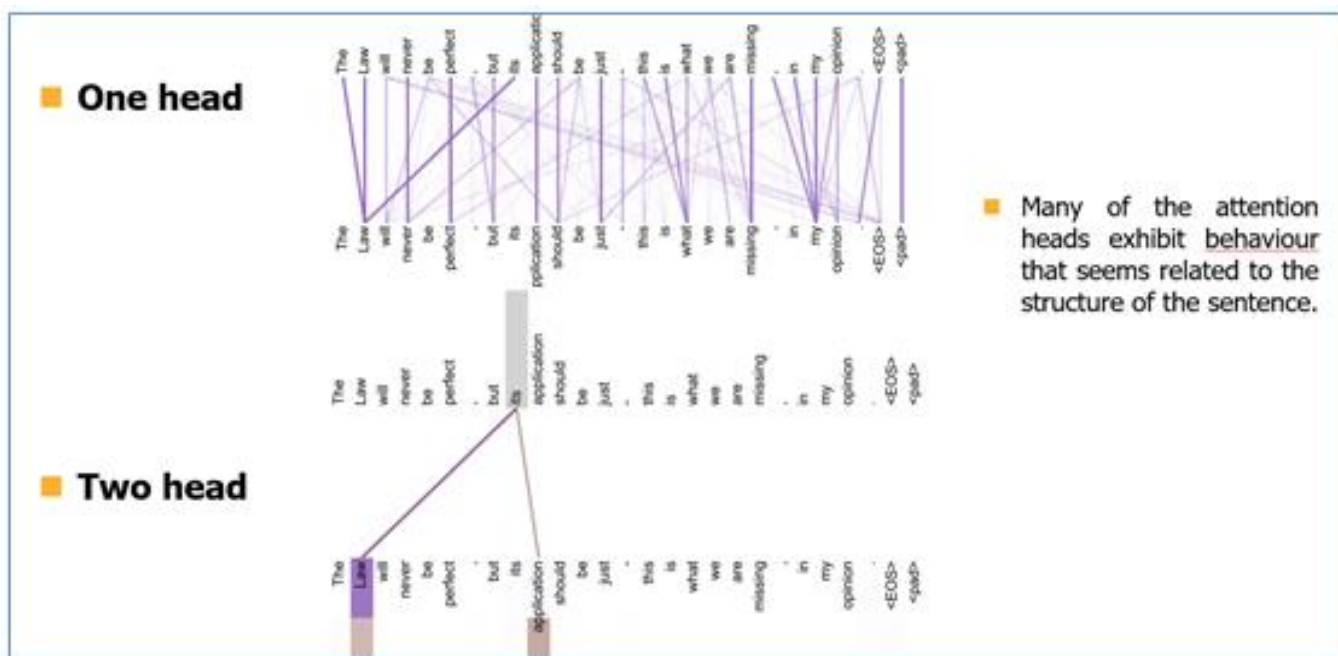
• 多头注意力机制 (Multi-Head Attention)

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



5.3 注意力编码机制

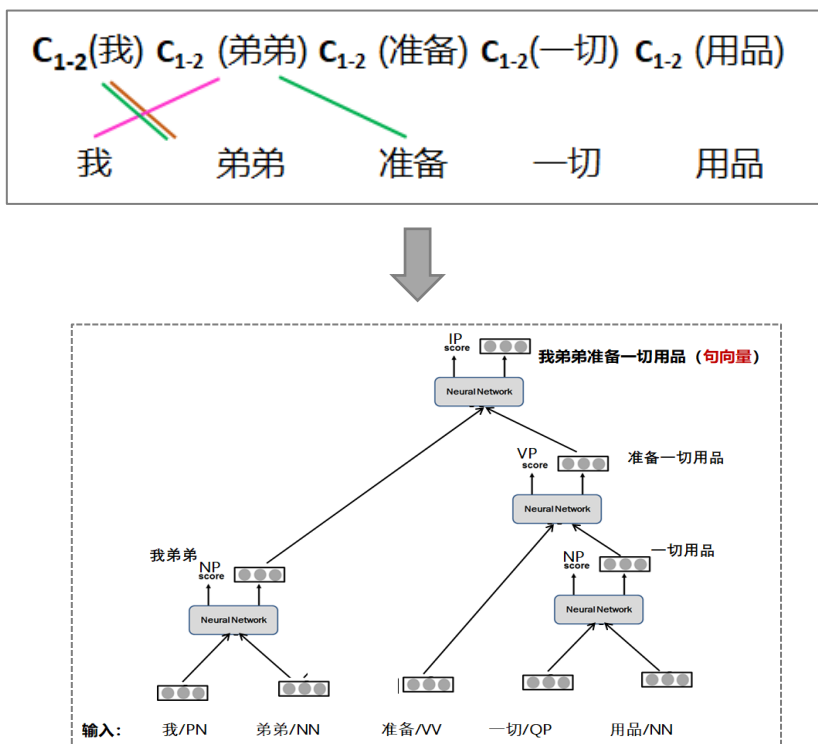
多头自注意力的可视化的效果



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

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

<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

李宏毅课程http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML16.html

在此表示感谢!

谢谢各位！

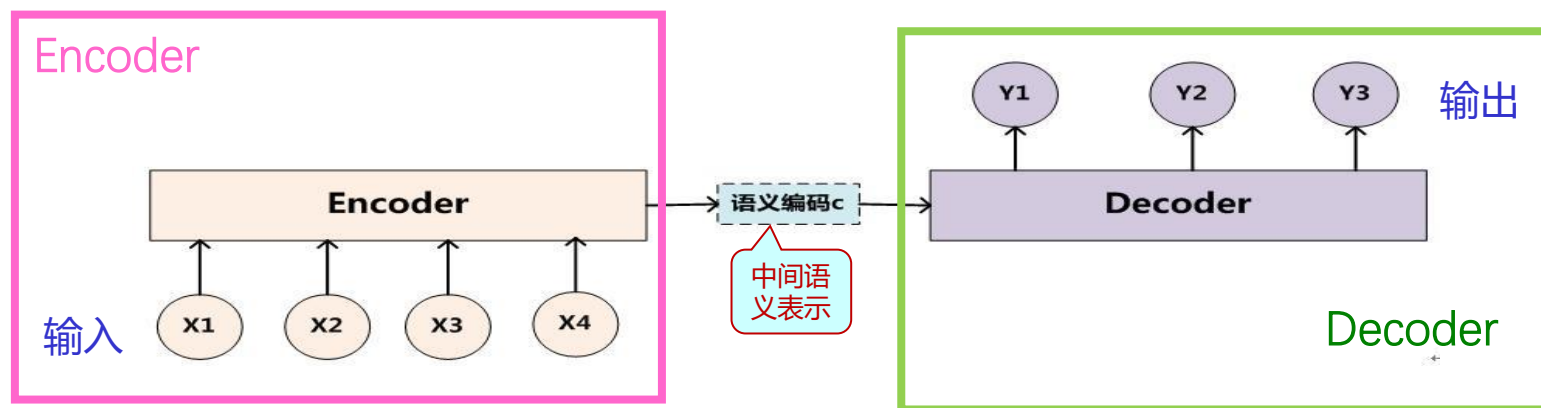


Q&A

附录： Encoder-Decoder 框架

Encoder-Decoder 框架

Encoder-Decoder是个非常通用的计算框架，抽象的表示：



Encoder: 对输入X序列进行编码，通过非线性变换转化为中间语义表示：

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

Decoder: 根据X的中间语义表示C和已经生成的 y_1, y_2, \dots, y_{i-1} 来生成 i 时刻的 y_i

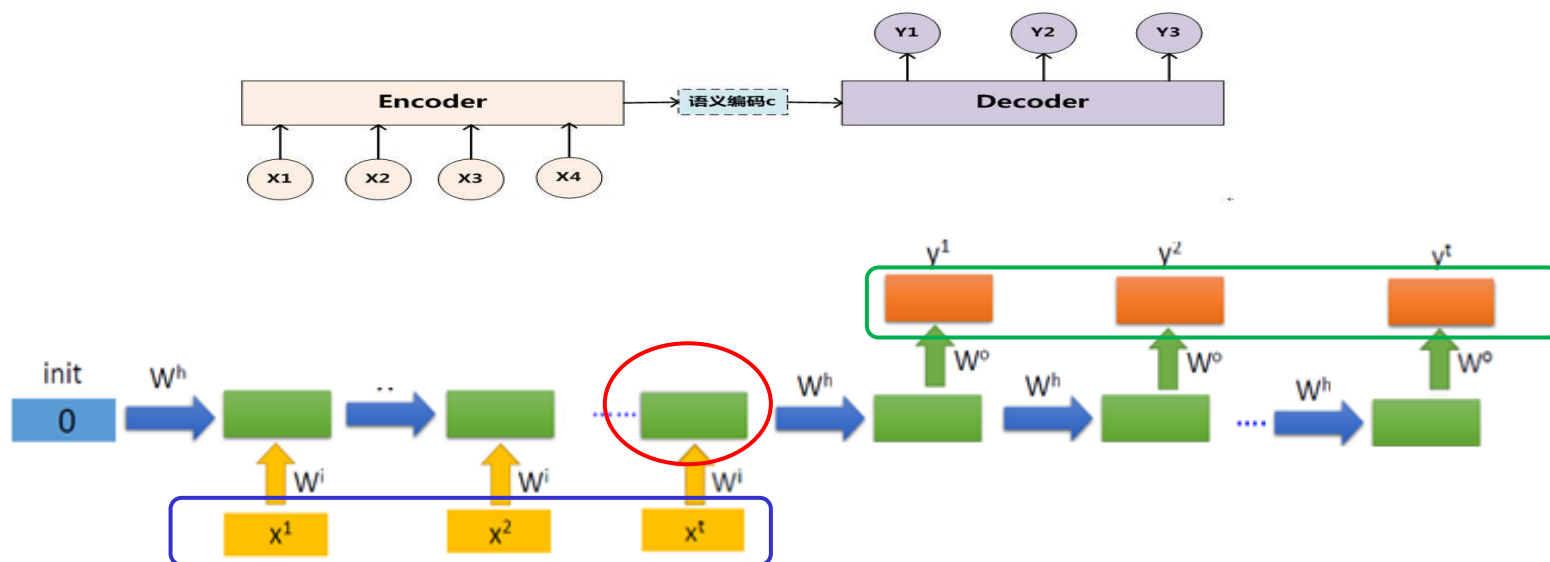
$$y_i = \mathcal{G}(C, y_1, y_2 \dots y_{i-1})$$

Encoder和Decoder具体使用什么模型都是由研究者自己确定。

如，CNN/RNN/BiRNN/GRU/LSTM/Deep LSTM等

Encoder-Decoder 框架

Encoder-Decoder 框架 RNN



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

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

中间语义表示 $C = \mathcal{F}(x_1, x_2 \dots x_m)$

$$y_1 = f(C)$$

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

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

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

C 可以作为X的句向量

附录完