2020-2021学年秋季学期

自然语言处理

Natural Language Processing

授课教师： 胡玥

助　教： 于静

自然语言处理
Natural Language Processing

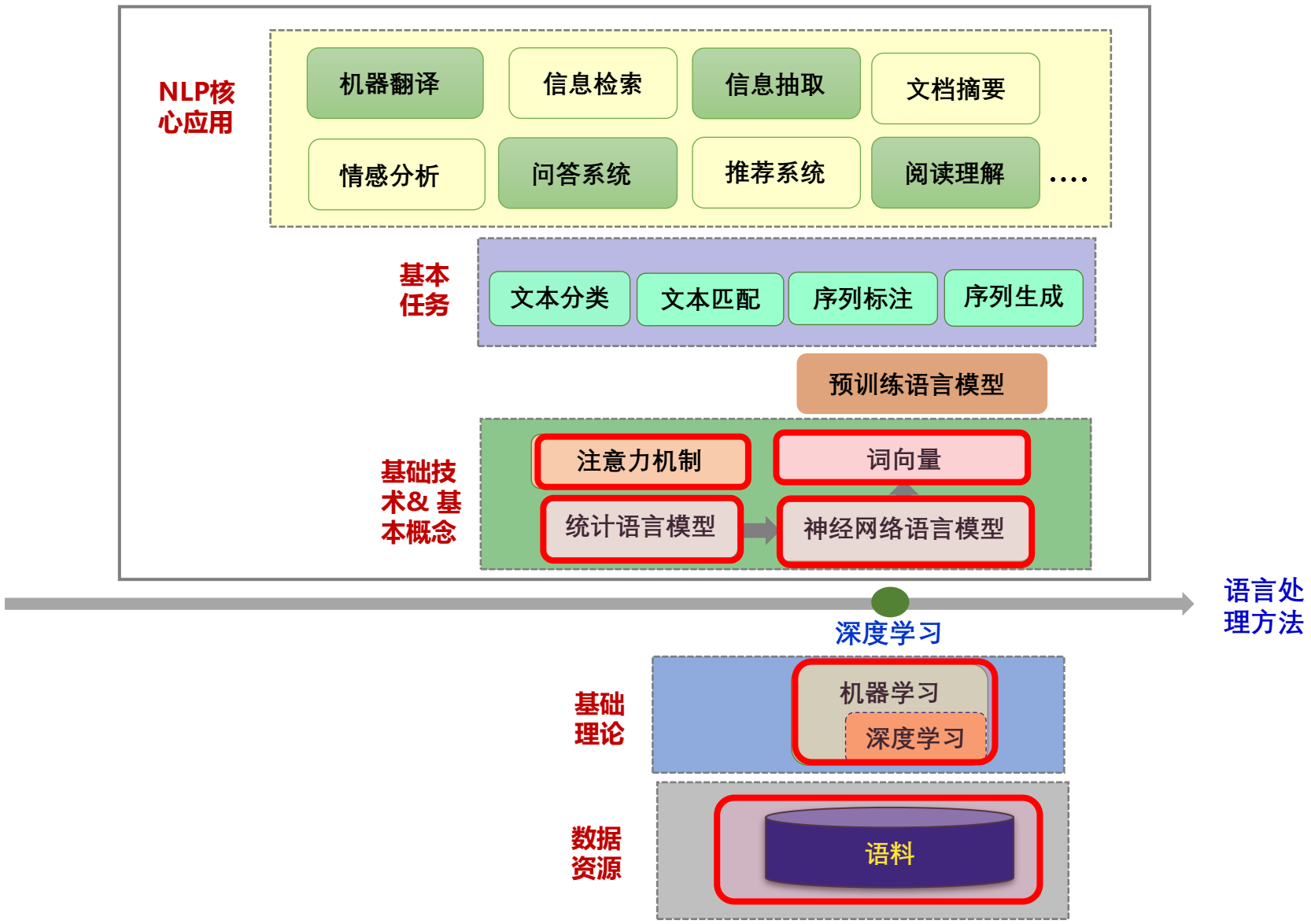
第 10 章　NLP中的注意力机制

授课教师：　胡玥

授课时间：2020.9

课程编码 201M4005H　课程名称 自然语言处理　授课团队名单 胡玥 、于静

基于深度学习的自然语言处理课程内容



第 10 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。

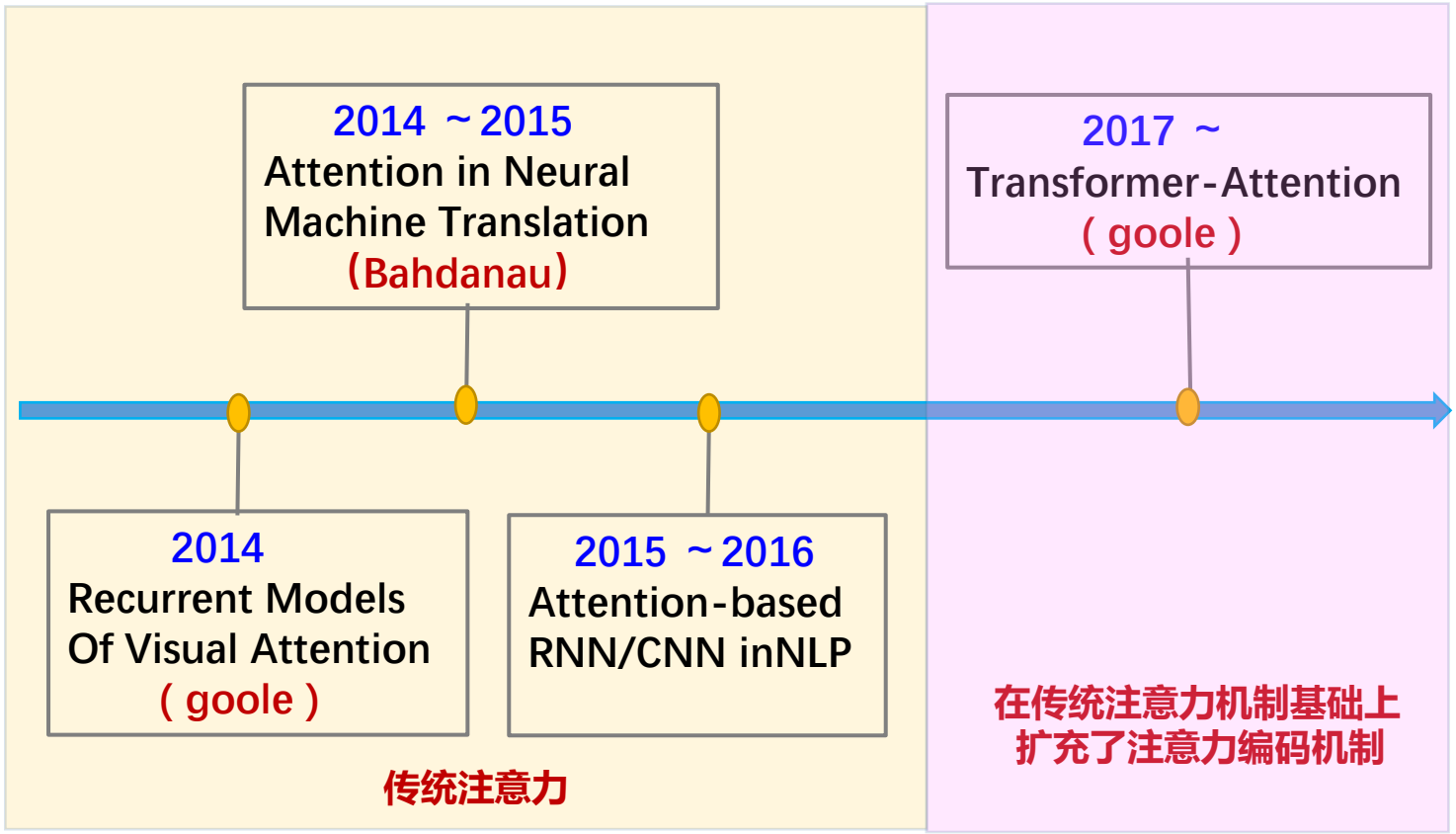
概 述

注意力机制（Attention）：的本质来自于人类视觉注意力机制。人们视觉在感知东西的时候一般不会对一个场景从到头看到尾每次全部都看，而往往是根据需求观察注意特定的一部分。而且当人们发现一个场景经常在某部分出现自己想观察的东西时，人们会进行学习在将来再出现类似场景时把注意力放到该部分上。

将AM模型应用在神经网络模型中可以提高效率和准确性

概 述

注意力机制发展历史：



内 容 提 要

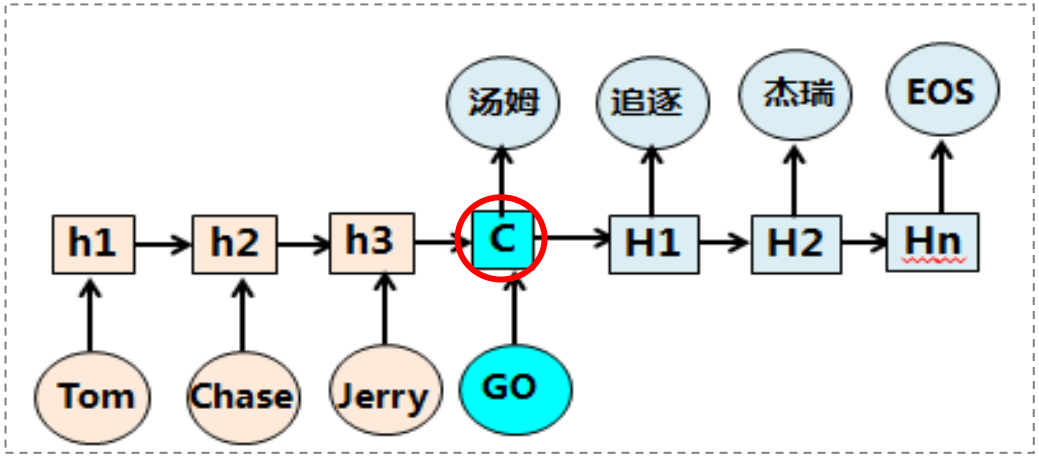
10.1 传统注意力机制

10.2 注意力编码机制

10.1 传统注意力机制

问题引入： **机器翻译例**

Encoder-Decoder RNN



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

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

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

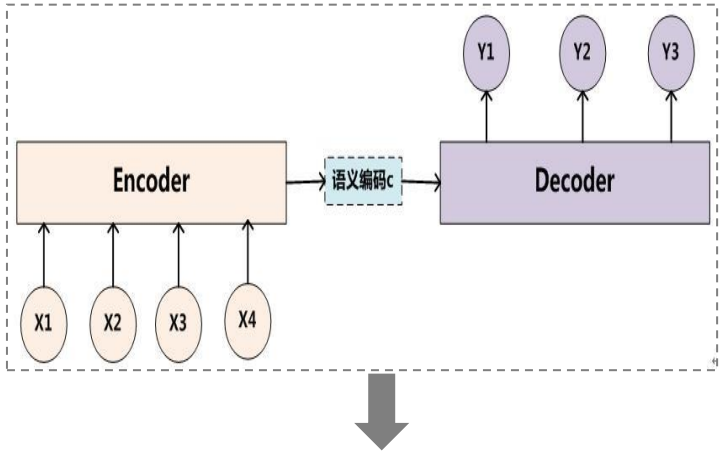
$$y_1 = f(C)$$
$$y_2 = f(C, y_1)$$
$$y_3 = f(C, y_1, y_2)$$
$$y_i = \mathcal{G}(C, y_1, y_2 \dots y_{i-1})$$

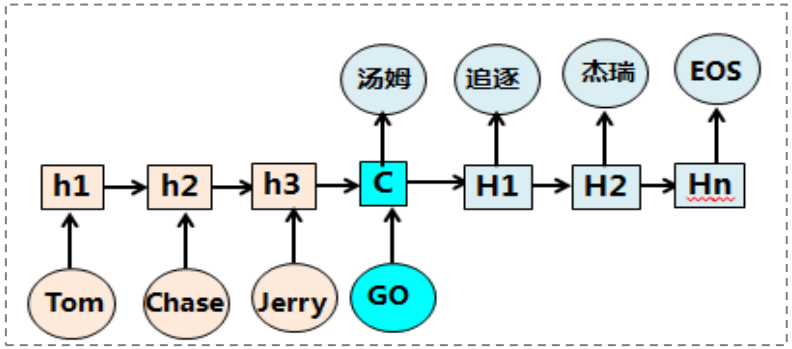
问题: 对不同的输出 Y_i 中间语义表示C 相同

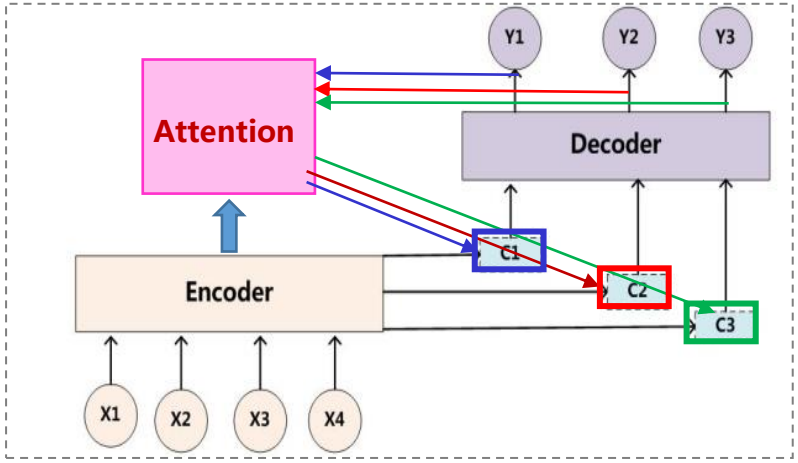
实际应该：在翻译“杰瑞”的时候，体现出英文单词对于翻译当前中文单词不同的影响程度，比如（Tom,0.3）(Chase,0.2) (Jerry,0.5)

问题：对每个输出的词，如何生成针对它的更准确的中间语义单元?

10.1 传统注意力机制





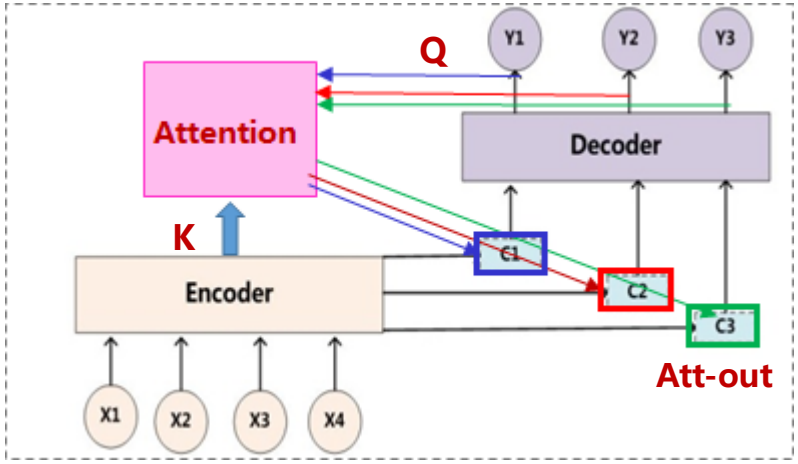


注意力机制：神经网络中的一个组件，可以单独使用，但更多地用作网络中的一部分。

作用：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。

优势：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。

10.1 传统注意力机制



Attention模型：

输入：Q，K

输出：Att-out

输入 → 输出 函数关系：

阶段1：计算 $f(Q, K_i)$

阶段2：softmax（$f(Q, K_i)$）（求得各K_i 对Q 得重要程度）

阶段3：计算输出（各 K_i 乘以自己的权重，然后求和）

10.1 传统注意力机制

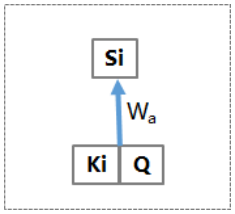
阶段1：计算 $f(Q, K_i)$

阶段2：softmax（$f(Q, K_i)$）（求得各K_i 对Q 得重要程度）

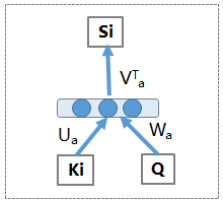
■ **设注意力打分函数 f(Q,K)**

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

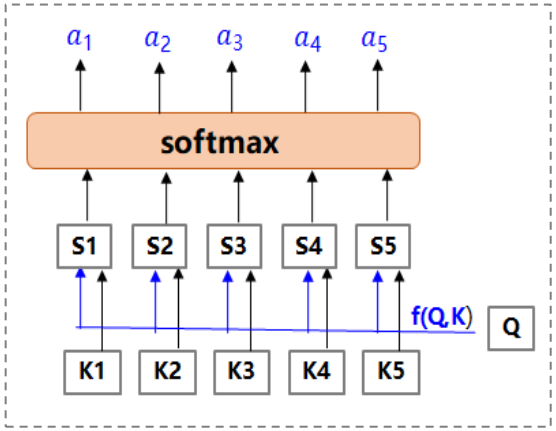
$W_a[Q, K_i]$

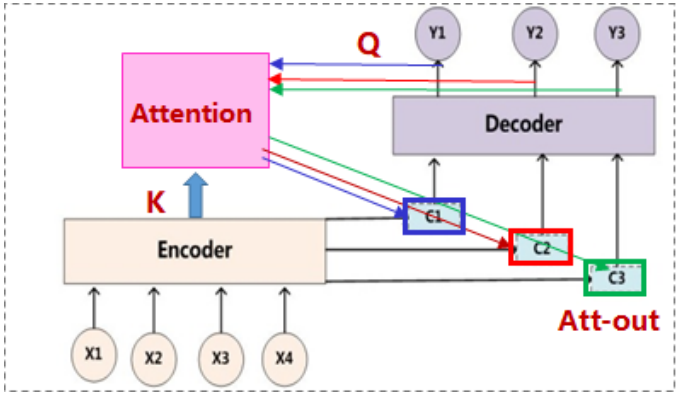


$V_a^T \tanh(W_a Q + U_a K_i)$



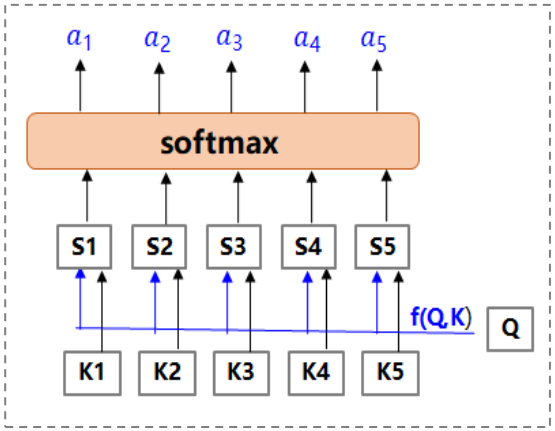
■ **计算各权重**

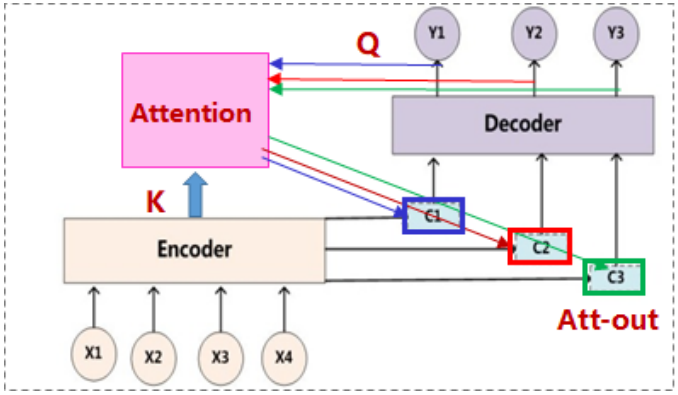




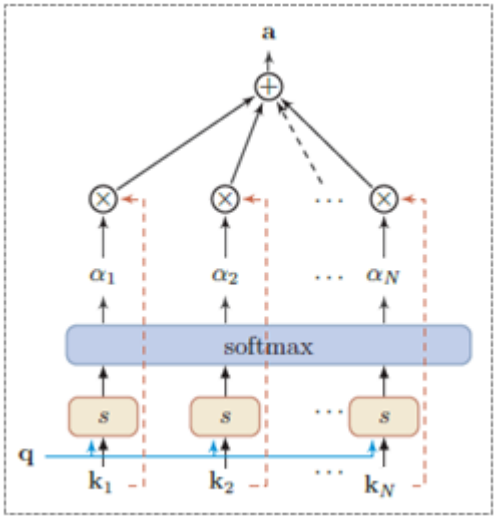
10.1 传统注意力机制

阶段3：计算输出（各 K_i 乘以自己的权重，然后求和 ）

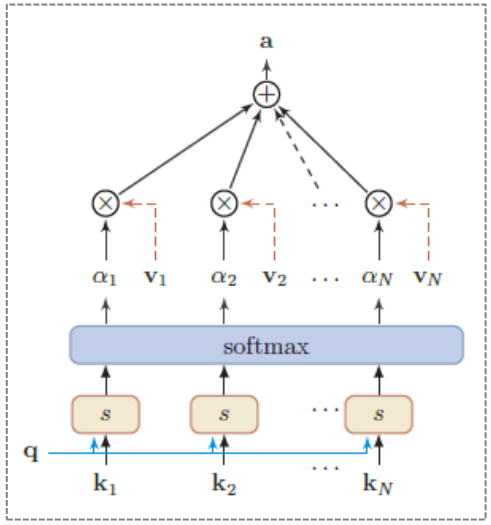




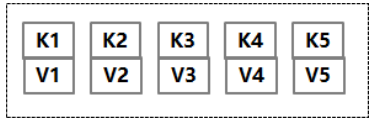
■ **计算输出**



普通模式

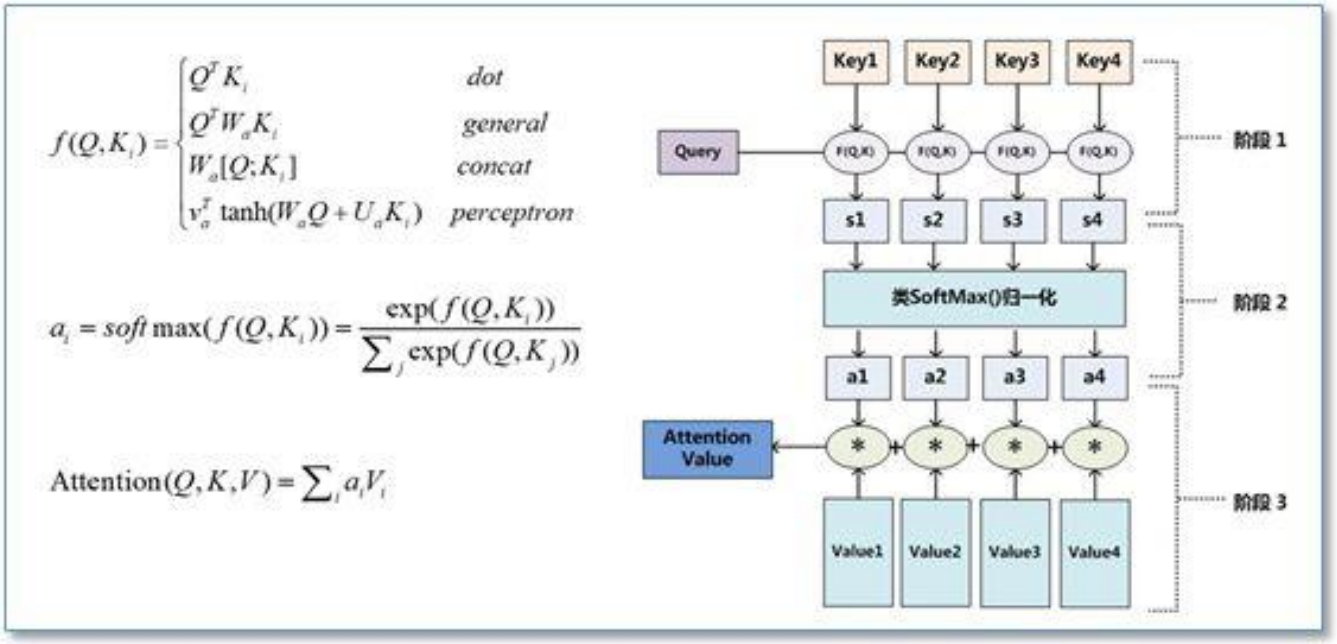


键值对模式



10.1 传统注意力机制

Attention模型表示：



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结果

10.1 传统注意力机制

Encoder (BiLSTM)-Decoder +Attention

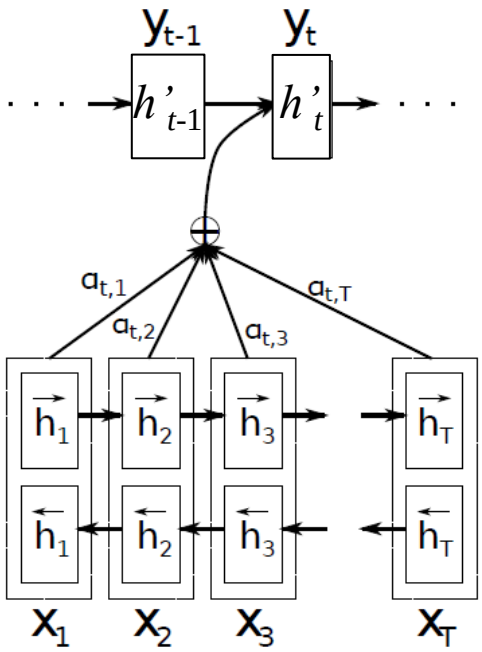


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

■ **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）

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

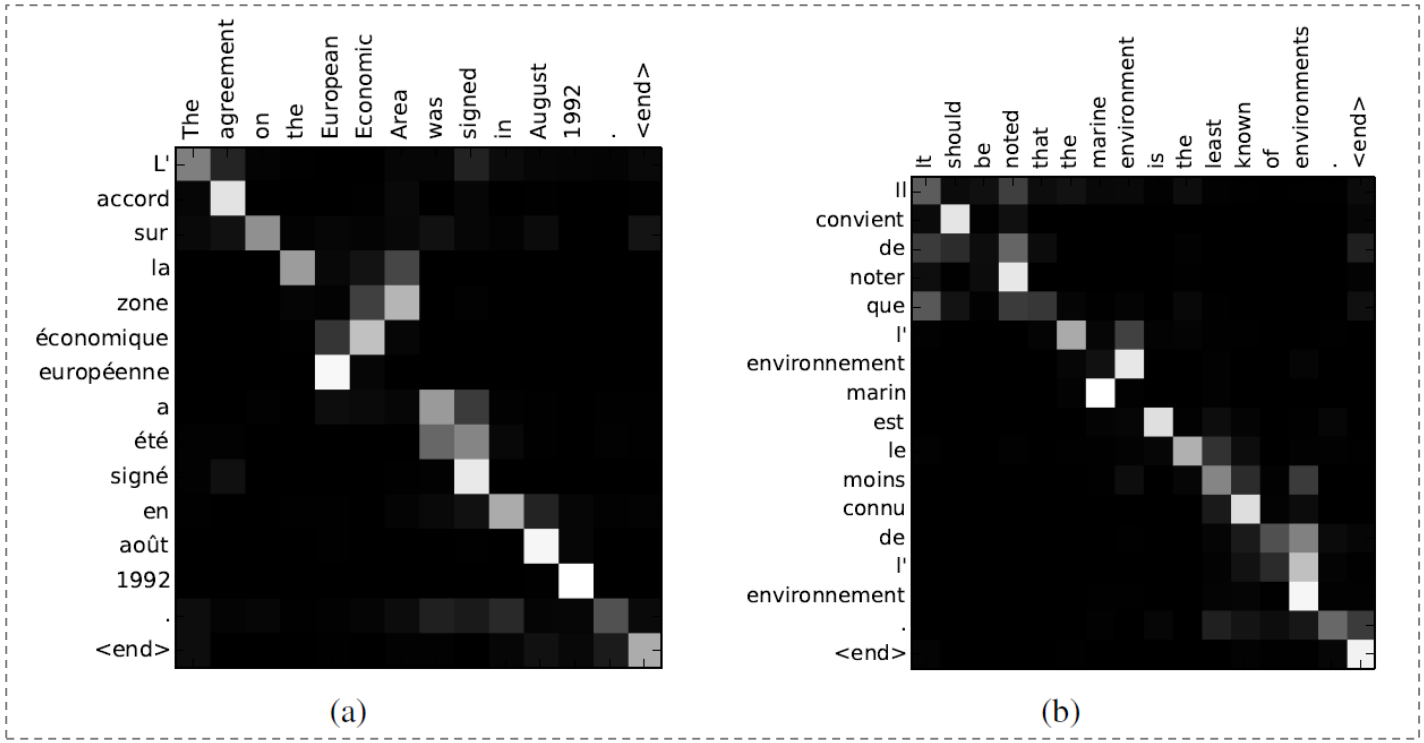
$$h_i^{'} = f(h_{i-1}^{'}, y_{i-1}, c_i)$$

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

Neural machine translation by jointly learning to align and translate. ICLR 2015

10.1 传统注意力机制

注意力机制可视化效果

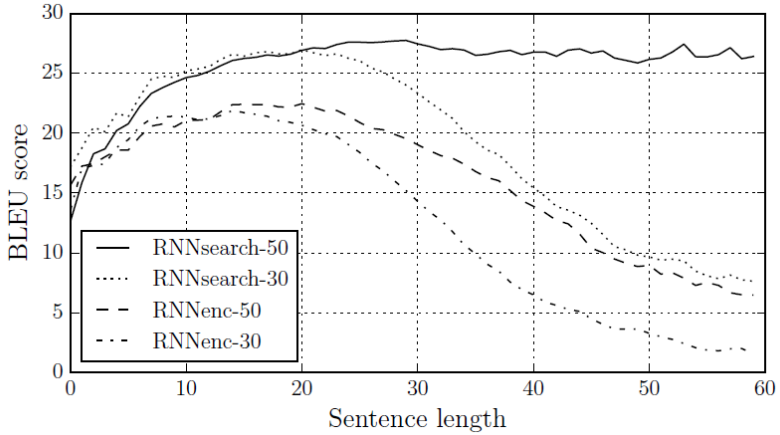


注意力机制的双语对齐（英语→法语）效果

Neural machine translation by jointly learning to align and translate. ICLR 2015

10.1 传统注意力机制

注意力机制实验结果



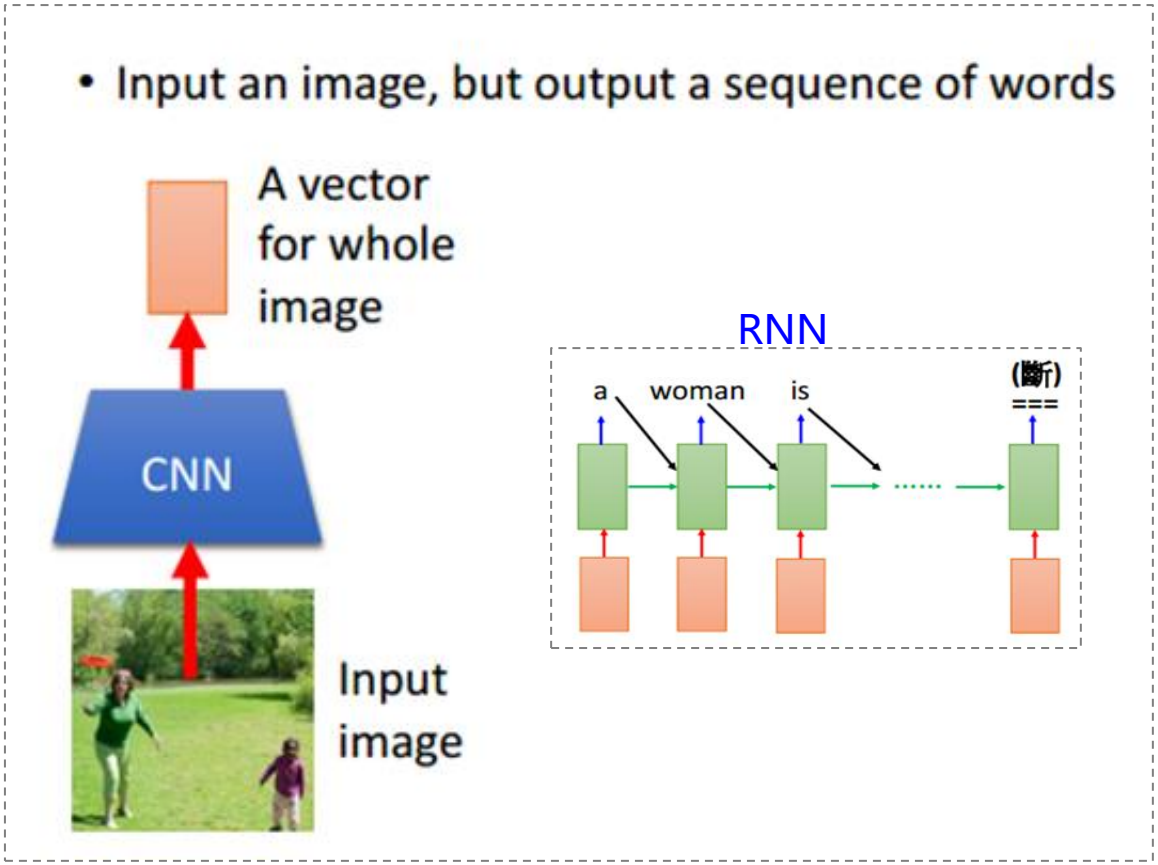
Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50*	28.45	36.15
Moses	33.30	35.63

- 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

Neural machine translation by jointly learning to align and translate. ICLR 2015

10.1 传统注意力机制

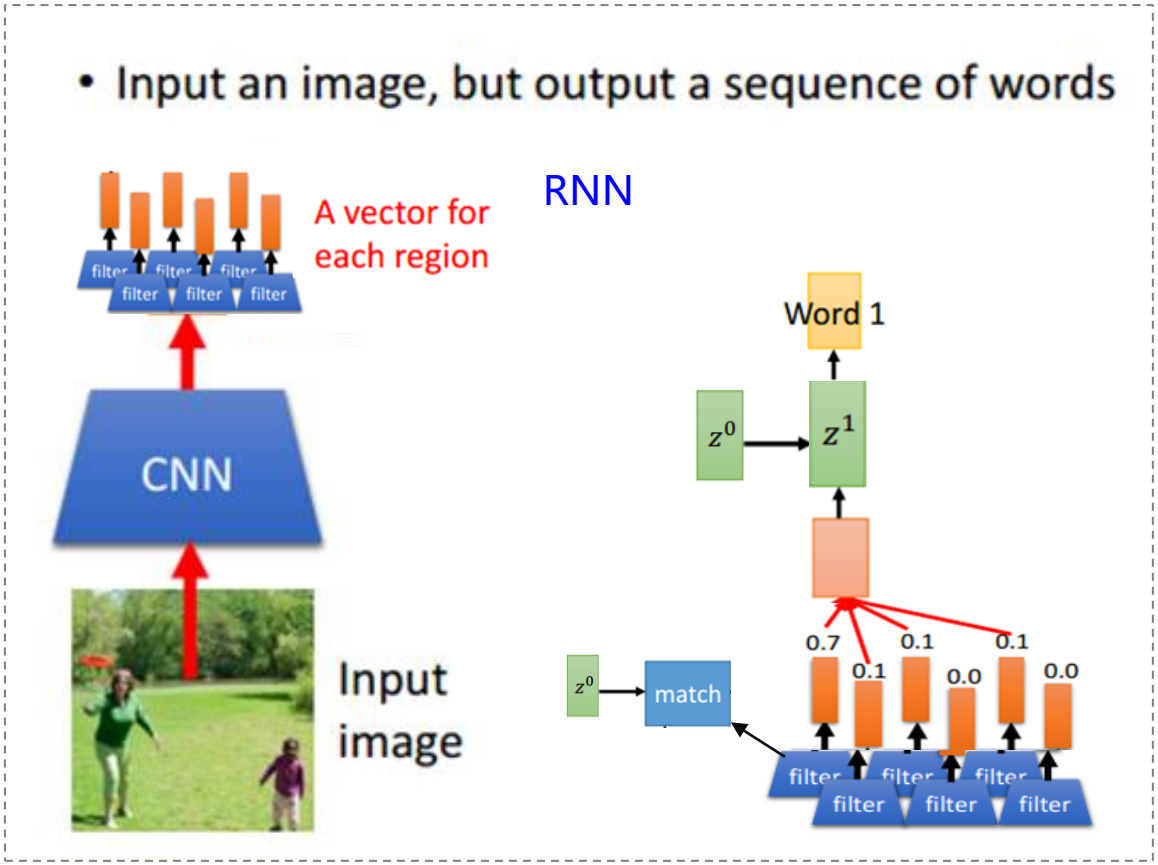
例：　　图片标题生成



A woman is throwing a Frisbee in a park

10.1 传统注意力机制

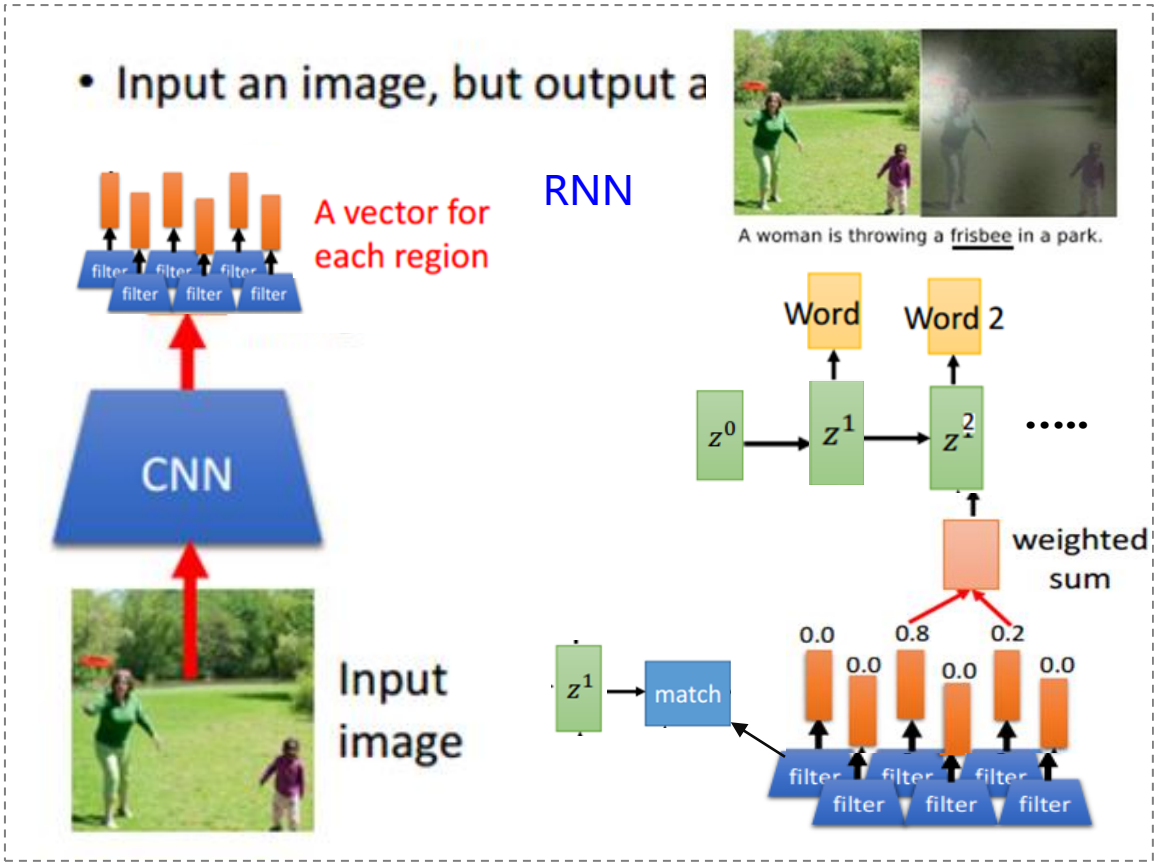
例：　　图片标题生成



A woman is throwing a Frisbee in a park

10.1 传统注意力机制

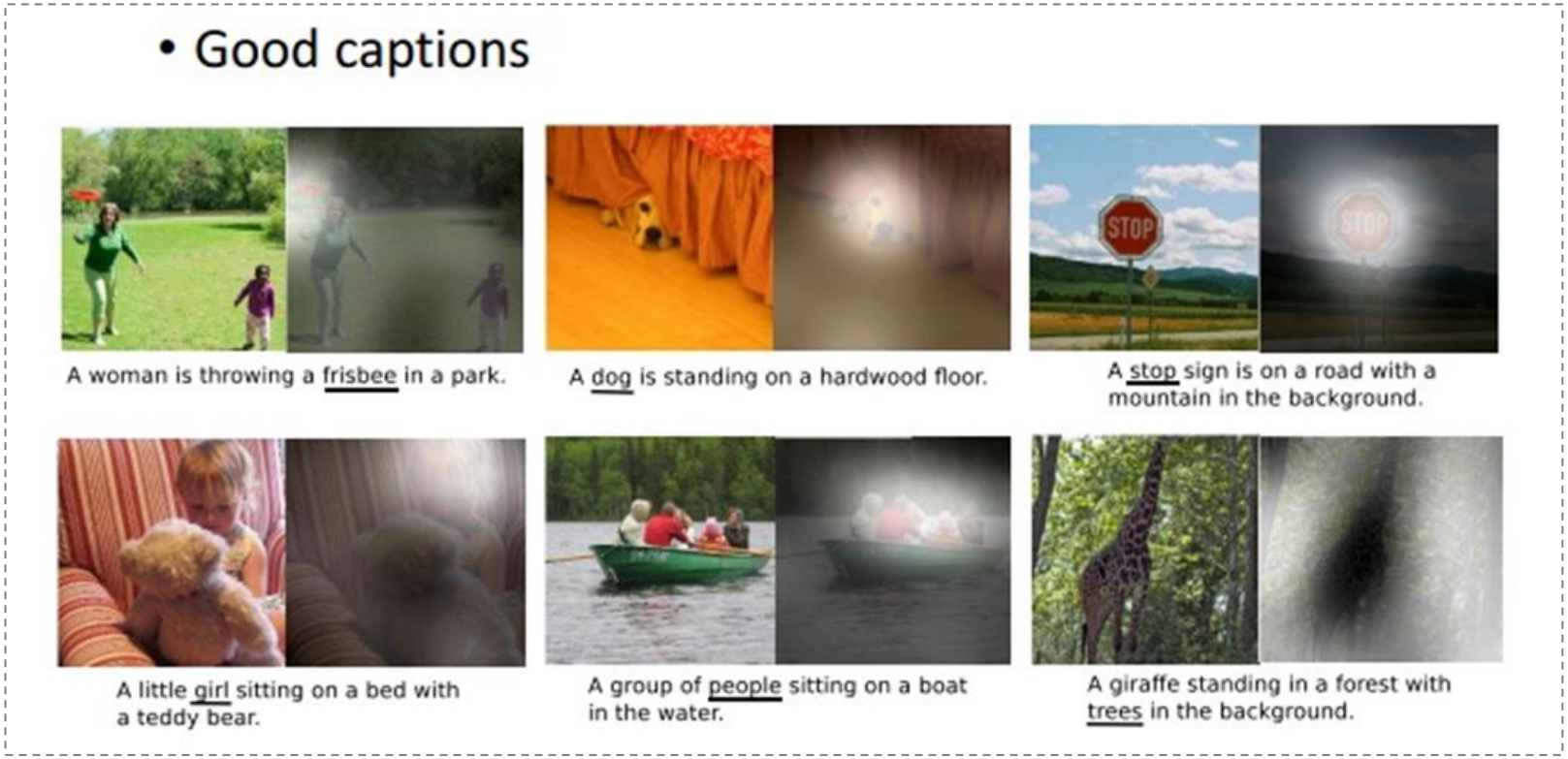
例：　　图片标题生成



A woman is throwing a Frisbee in a park

10.1 传统注意力机制

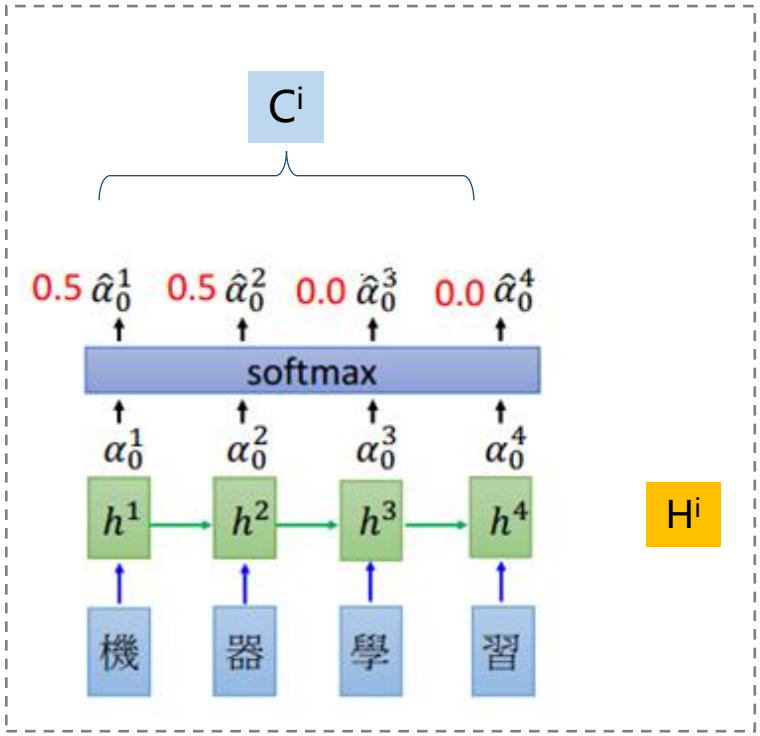
图片标题生成实验



10.1 传统注意力机制

口 软注意力 Hard Attention

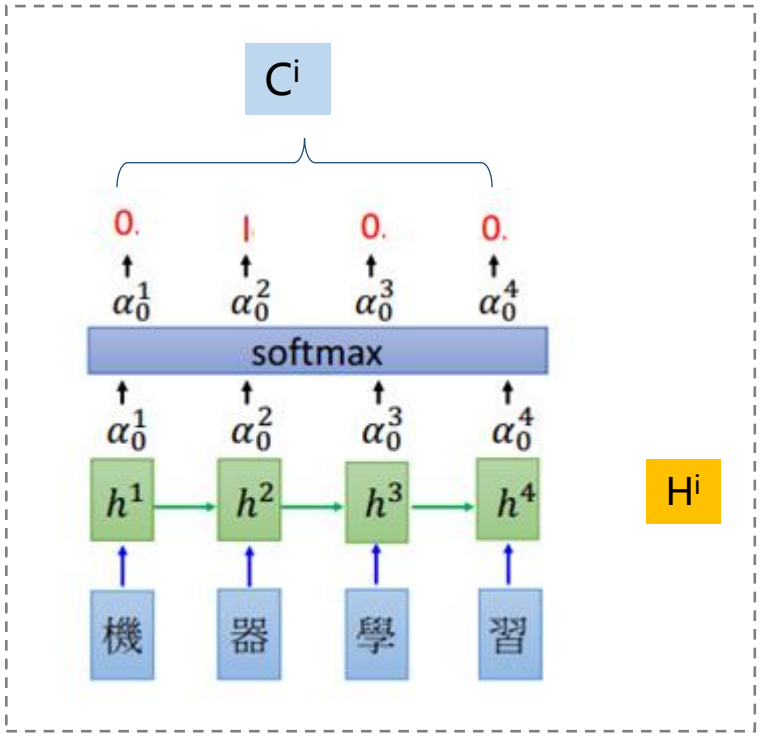
Soft AM：在求注意力分配概率分布的时候，对于输入句子X中任意一个单词都给出个概率，是个概率分布。



10.1 传统注意力机制

口 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

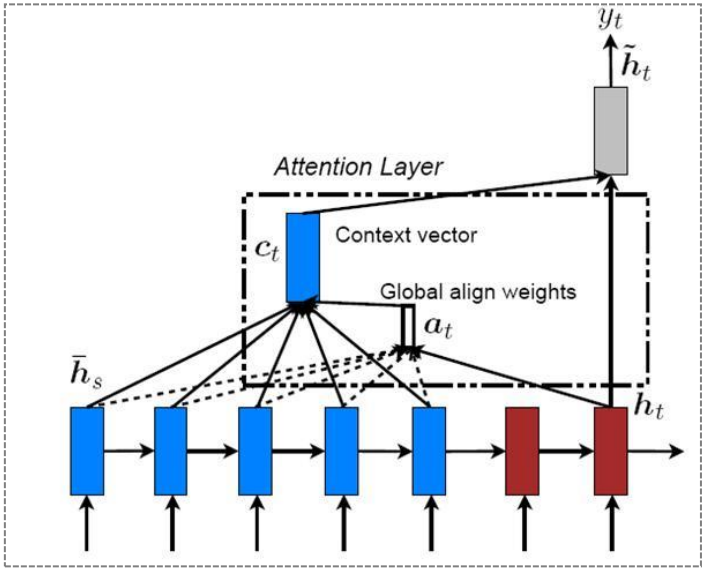


10.1 传统注意力机制

□ 全局注意力 Global Attention

Decode端Attention计算时要考虑输Ecoder端序列中所有的词

Global Attention Model 是Soft Attention Model

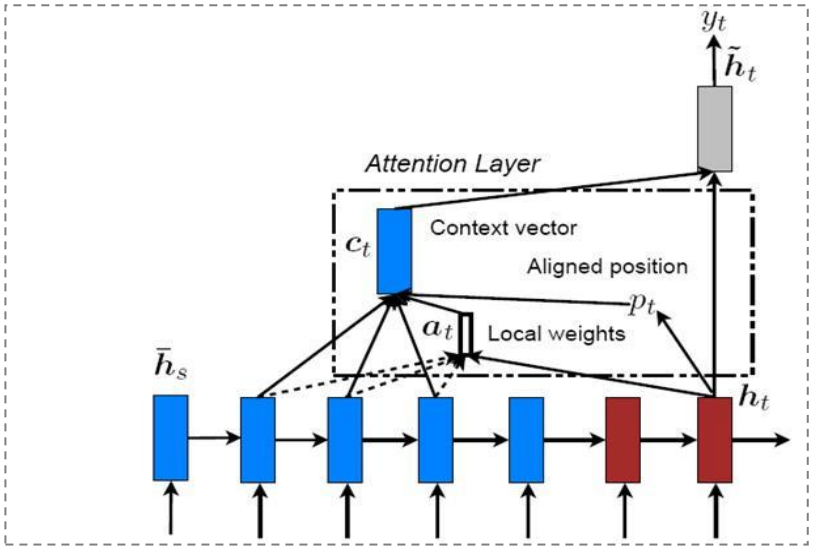


Effective Approaches to Attention-based Neural Machine Translation. ACL 2015.

10.1 传统注意力机制

□ 局部注意力 Local Attention

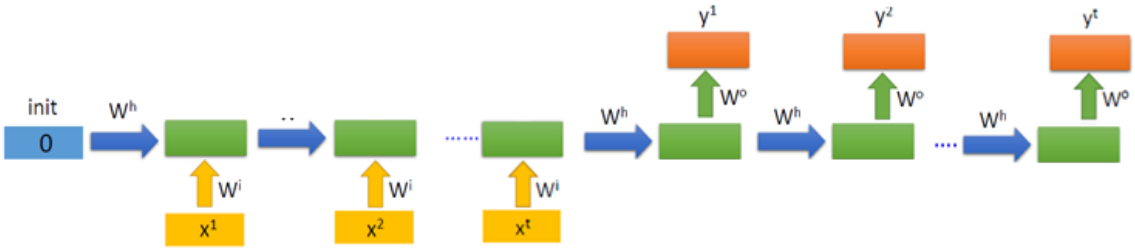
Local Attention Model本质上是Soft AM和 Hard AM的一个混合或折衷。一般首先预估一个对齐位置Pt，然后在Pt左右大小为D的窗口范围来取类似于Soft AM的概率分布。



Effective Approaches to Attention-based Neural Machine Translation. ACL 2015.

10.1 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

10.1 传统注意力机制

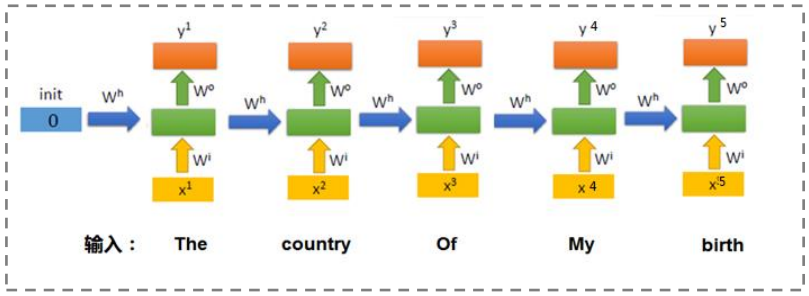
10.2 注意力编码机制

10.2 注意力编码机制

序列编码

深度学习的NLP 方法，基本上都是先将句子分词，然后每个词转化为对应的词向量序列这样一来，每个句子都对应的是一个矩阵 $X=(X_1,X_2,...,X_n)$，其中X_i 代表着第 i 个词的词向量（行向量），维度为 d 维，故 $X\in R^{n\times d}$。于是，各种模型就变成编码这些序列。

- **RNN层：**



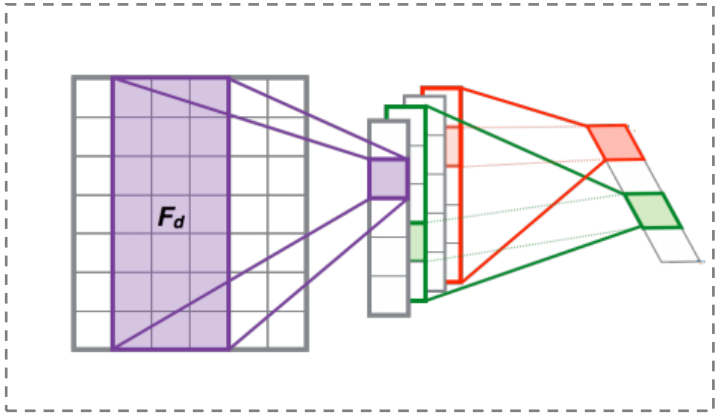
RNN 的方案很简单，递归式进行： $y_t=f(y_{t-1},x_t)$

优点：结构比较简单，适合序列建模；

不足：无法并行，速度较慢，无法很好地学习到全局的结构信息。要用双向 RNN逐步递归才能获得全局信息。

10.2 注意力编码机制

- **CNN 层：** CNN 的方案也是很自然的，窗口式遍历



如，采用卷积核（窗口）为 3 的卷积，CNN为：

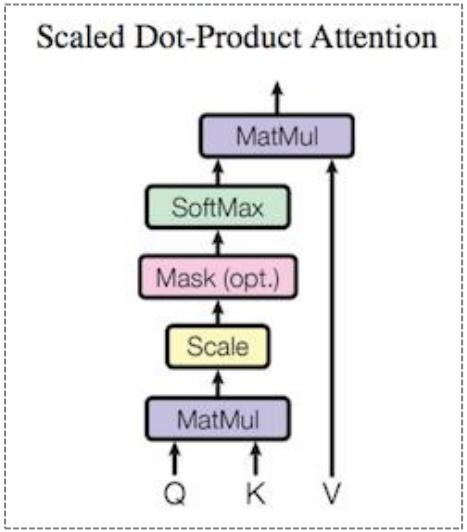
$$y_t=f(x_{t-1},x_t,x_{t+1})$$

CNN 事实上只能获取局部信息，是通过层叠来增大感受野

10.2 注意力编码机制

- **Attention层**

Google 的 Attention 思路也是一个编码序列的方案，可以认为它RNN、CNN 一样，都是一个序列编码的层。



$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

其中，$Q \in R^{n \times dk}, K \in R^{m \times dk}, V \in R^{m \times dv}$

scale factor $\sqrt{d_k}$ 起 到调节作用，使得内积不至于太大（否则softmax会进入饱和区）

如果忽略激活函数 softmax 的话，那么事实上它就是三个 $n \times d_k, d_k \times m, m \times d_v$ 的矩阵相乘，最后的结果就是一个 $n \times d_v$ 的矩阵。这是一个 Attention 层，将 $n \times d_k$ 的序列 Q 编码成了一个新的 $n \times d_v$ 的序列。

10.2 注意力编码机制

Attention用作编码机制

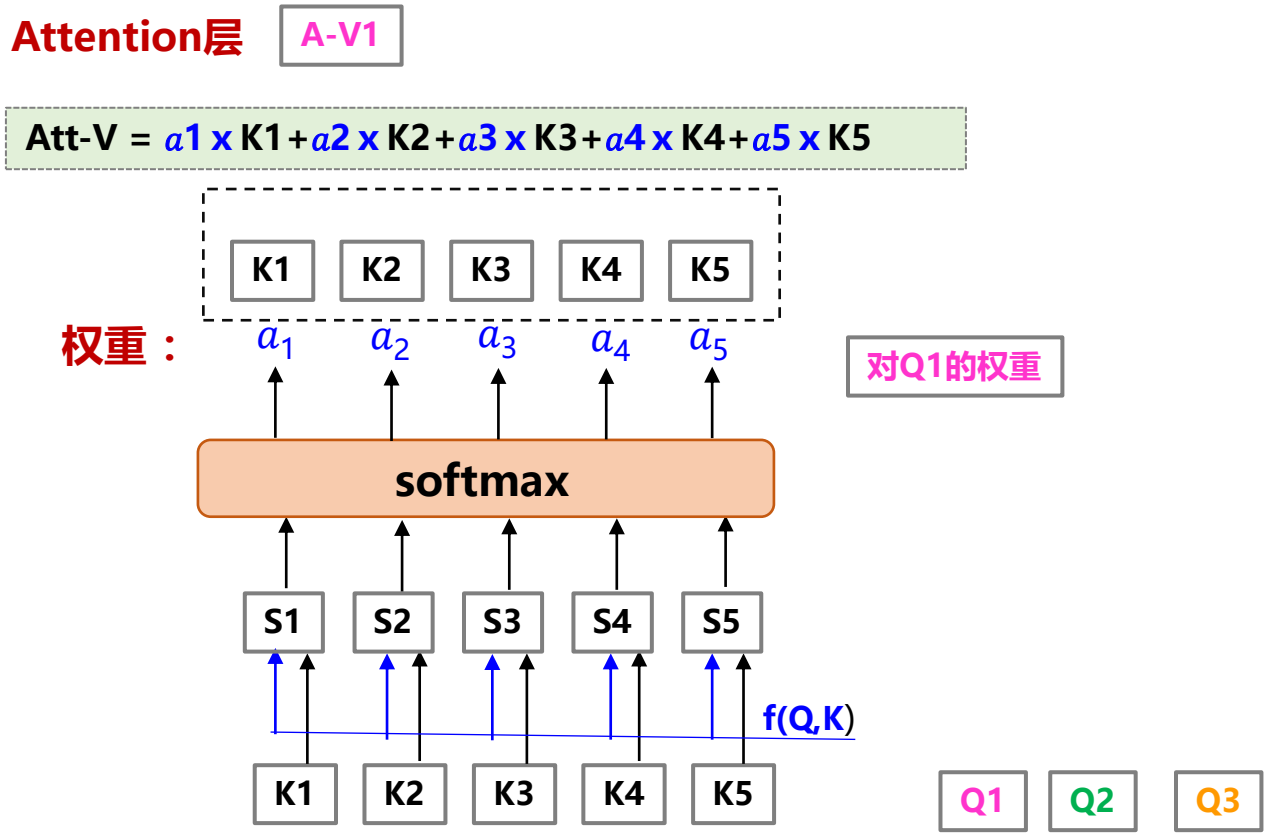
- **不同序列间编码：**可以将2个序列编码成二者的融合表示，匹配任务，阅读理解任务常用

- **同一序列自编码：**利用多头Self-Attention编码对一个句子编码可以起到句法分析器的作用

10.2 注意力编码机制

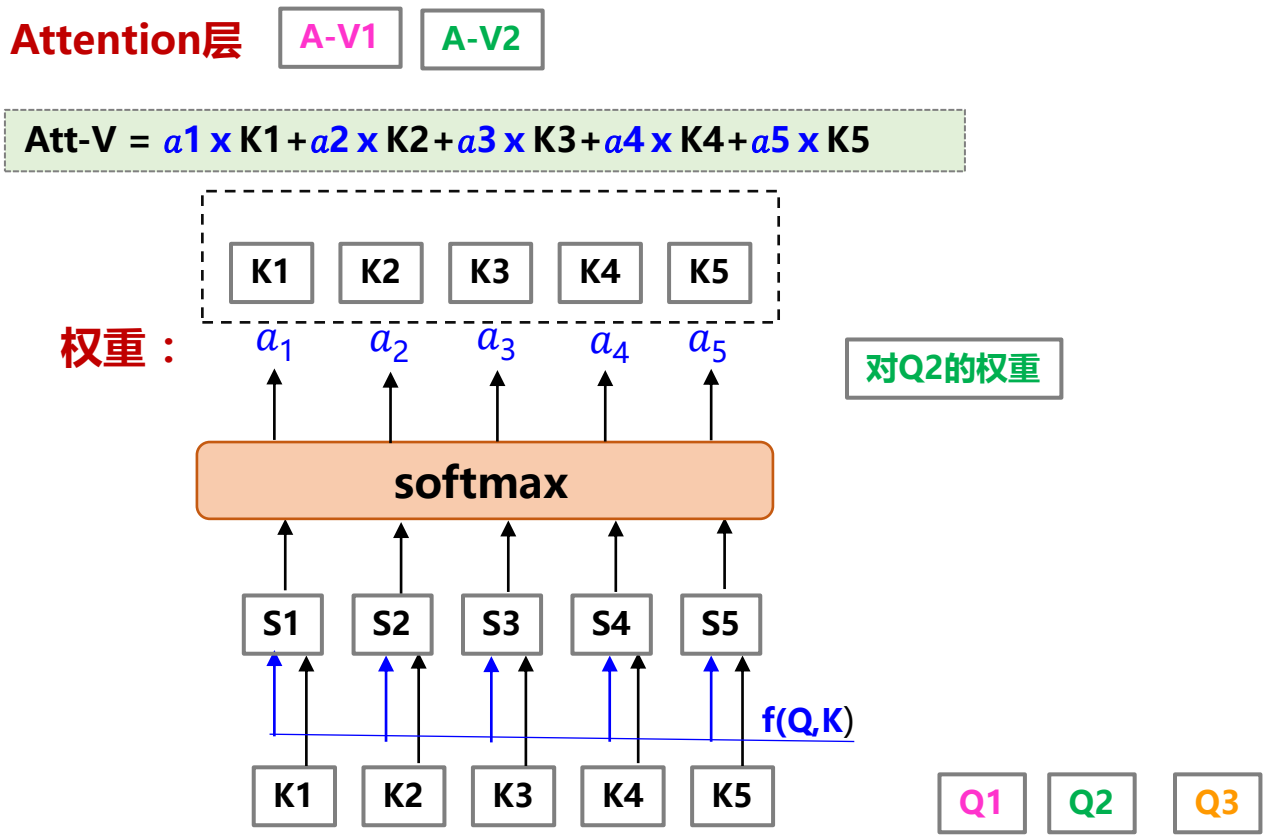
◆ 不同序列间编码：

例：对K序列和Q序列编码

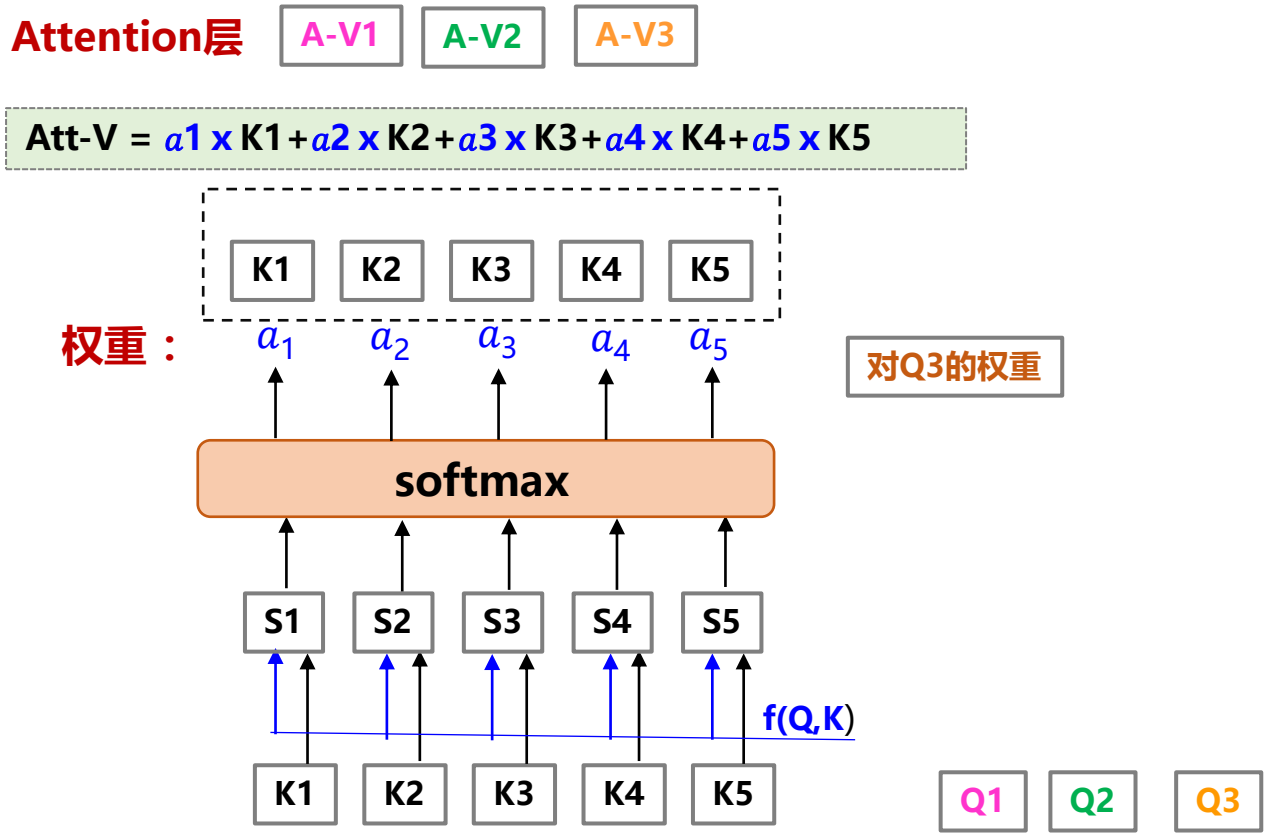


A-V1 序列的含义？

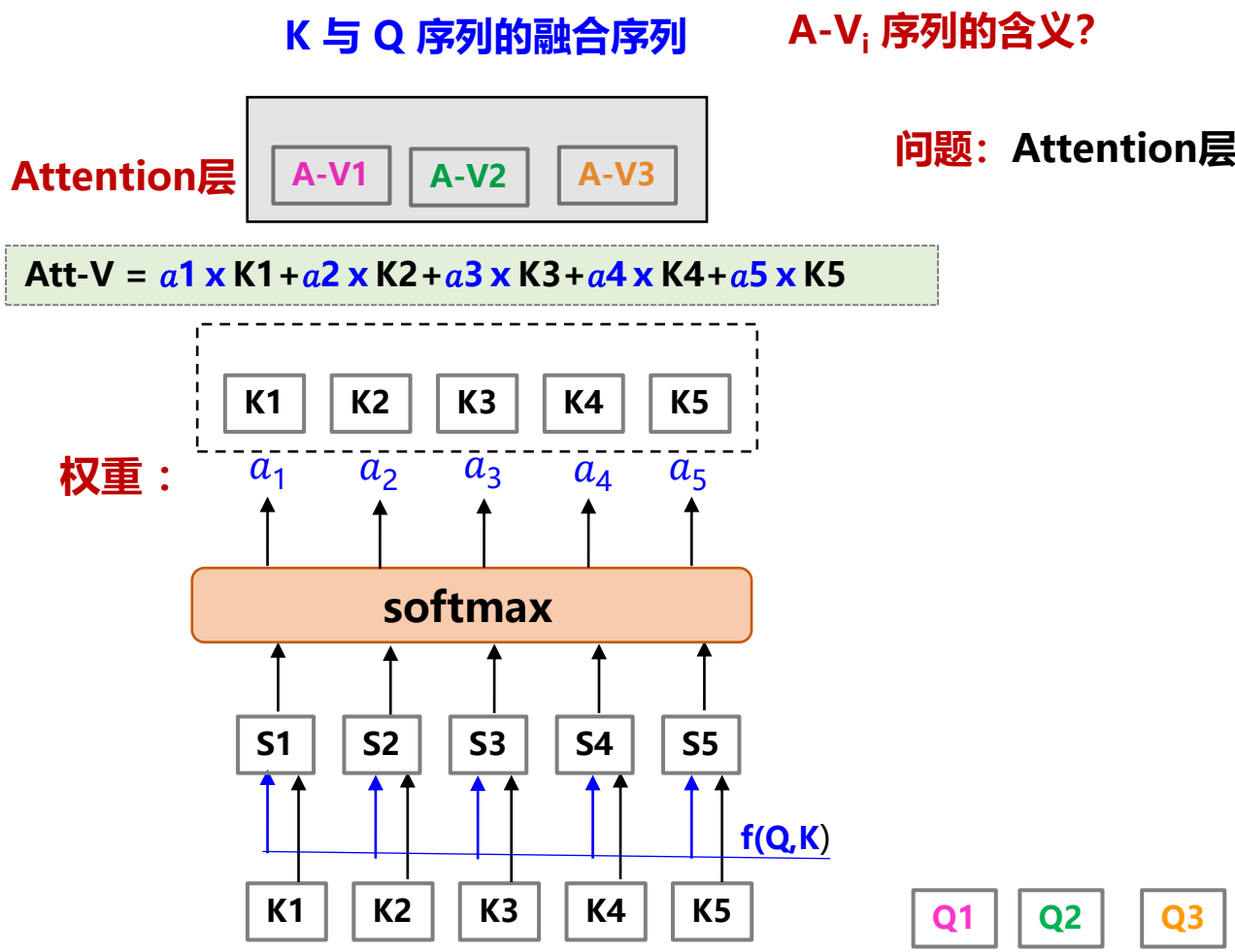
10.2 注意力编码机制


Attention层
A-V1
A-V2
Att-V = a1 x K1 + a2 x K2 + a3 x K3 + a4 x K4 + a5 x K5
K1
K2
K3
K4
K5
权重：
a_1
a_2
a_3
a_4
a_5
对Q2的权重
softmax
S1
S2
S3
S4
S5
f(Q,K)
K1
K2
K3
K4
K5
Q1
Q2
Q3

10.2 注意力编码机制


Attention层
A-V1
A-V2
A-V3
Att-V = a1 x K1 + a2 x K2 + a3 x K3 + a4 x K4 + a5 x K5
K1
K2
K3
K4
K5
权重：
a_1
a_2
a_3
a_4
a_5
对Q3的权重
softmax
S1
S2
S3
S4
S5
f(Q,K)
K1
K2
K3
K4
K5
Q1
Q2
Q3

10.2 注意力编码机制



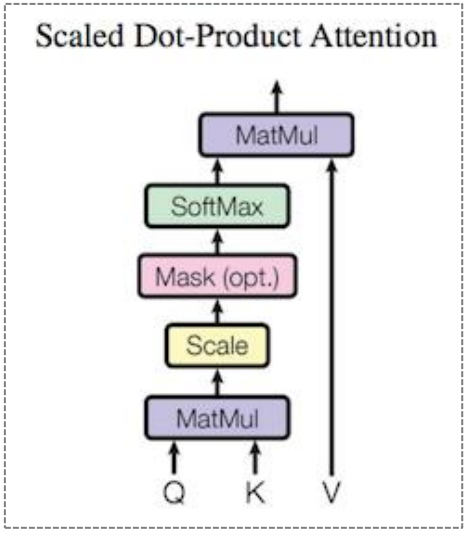
A-V$_i$ 序列元素个数等于 Q序列元素个数

10.2 注意力编码机制

◆ 同一序列自编码：

自注意力 Self-Attention

其实就是 Attention($\mathbf{X},\mathbf{X},\mathbf{X}$)，$\mathbf{X}$ 为输入序，其含义为在序列内部做 Attention，寻找序列内部的联系



$$\mathbf{Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V}$$

其中，　Q=K=V

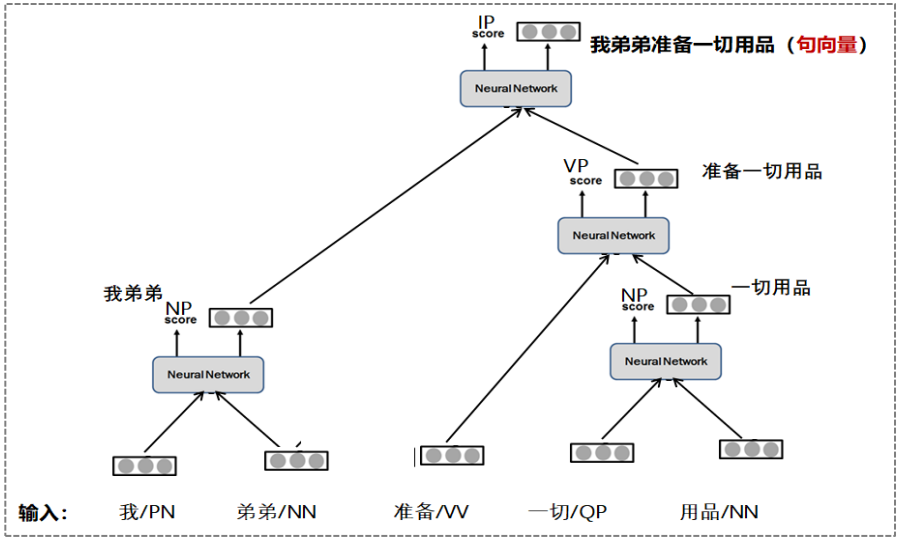
self-attention的特点在于无视词之间的距离直接计算依赖关系，能够学习一个句子的内部结构，实现也较为简单并行可以并行计算（self-attention可以当成一个层和RNN，CNN，FNN等配合使用，应用于其他NLP任务）

10.2 注意力编码机制

传统句子句法分析树：

如： 我 弟弟 已经 准备 好了 一切 用品

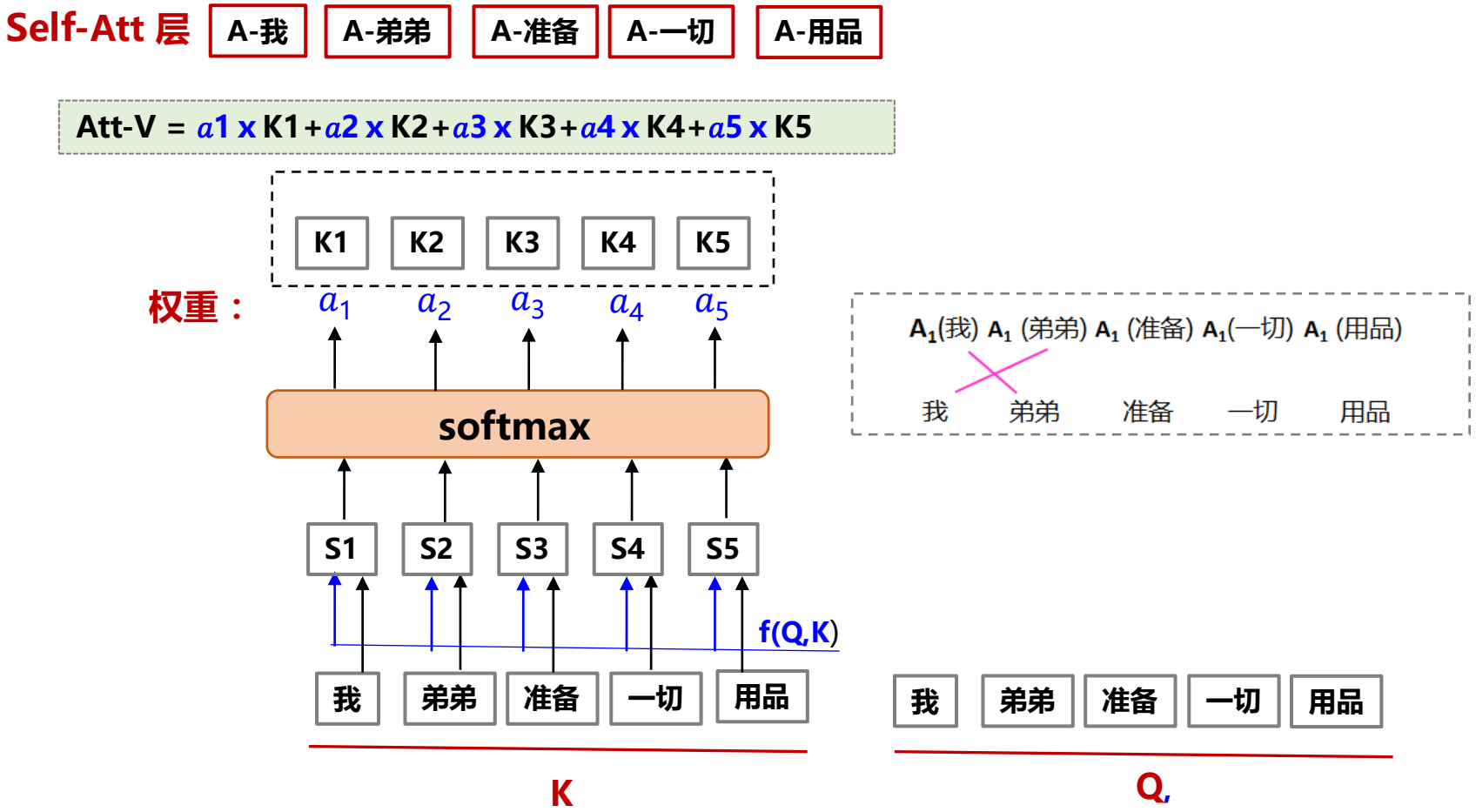
用递归神经网络对句子做句法分析生成短语结构树如下：



短语结构树

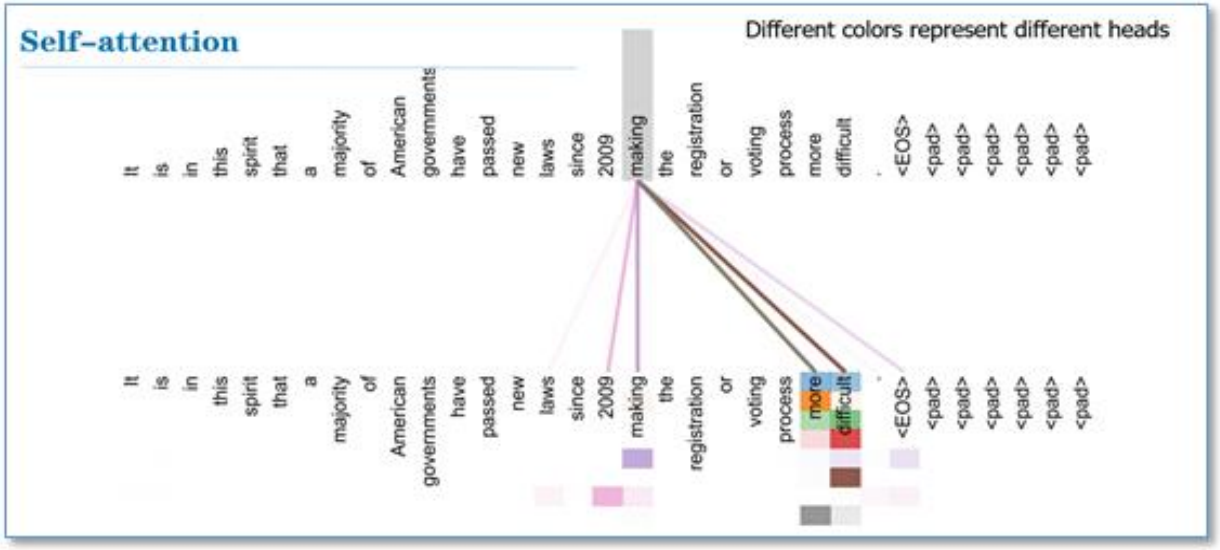
10.2 注意力编码机制

例：对同一序列自注意力编码



10.2 注意力编码机制

Self-Attention可视化的效果



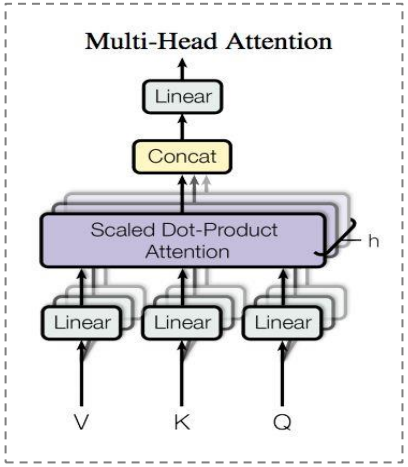
可以看到self-attention在这里可以学习到句子内部长距离依赖"making.....more difficult"这个短语

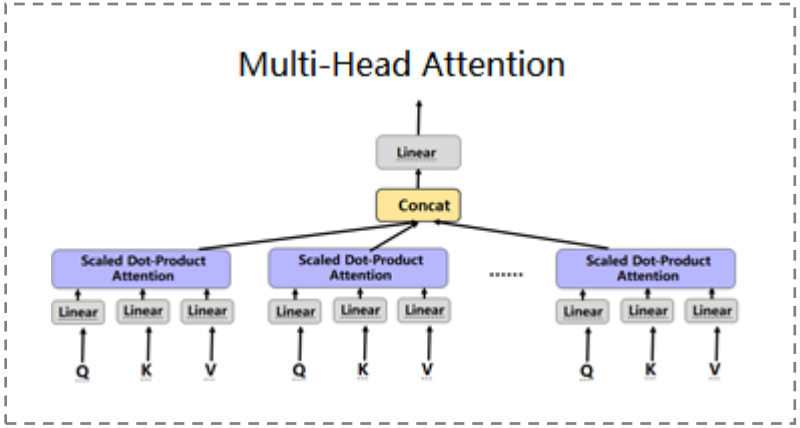
10.2 注意力编码机制

多头注意力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

多头attention通过计算多次来捕获不同子空间上的相关信息。





$$\mathrm{Head}_i = \mathrm{Attention}(QW^Q_i, KW^K_i, VW^V_i)$$

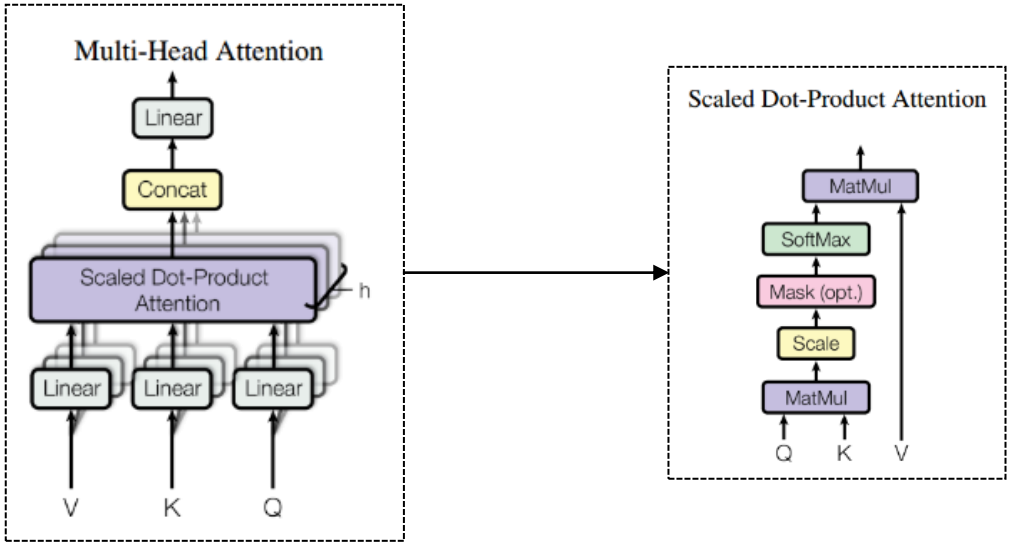
其中，$W^Q_i \in R^{dk \times \sim dk}, W^K_i \in R^{dk \times \sim dk}, W^V_i \in R^{dv \times \sim dv}$

$$\mathrm{Multi\text{-}Head}(Q,K,V) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)$$ **最后得到一个 $n \times (h \sim d_v)$的序列**

10.2 注意力编码机制

多头自注意力 Multi-Head Self Attention（Transformer)

多头自注意力 （Multi-Head Self Attention ）就是多头attention中，每头均为自注意力 Attention($\mathbf{X},\mathbf{X},\mathbf{X}$)



$$\text{MultiHead}(H) = W_O[\mathbf{head}_1;\cdots;\mathbf{head}_M]$$
$$\mathbf{head}_m = \mathbf{self\text{-}att}(Q_m, K_m, V_m)$$
$$Q_m = W_Q^m H, K = W_K^m X, V = W_V^m X$$

$$\mathbf{self\text{-}att}(Q,K,V) = \text{softmax}\left(\frac{K^{\mathrm{T}}Q}{\sqrt{d_h}}V\right)$$

10.2 注意力编码机制

例：对同一序列多头自注意力编码

$$Y=MultiHead(X,X,X)$$

◆ **Head1 Self Attention**

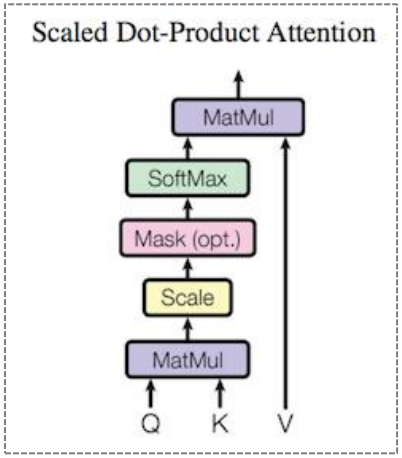
输出：A_1(我) A_1 (弟弟) A_1 (准备) A_1(一切) A_1 (用品)

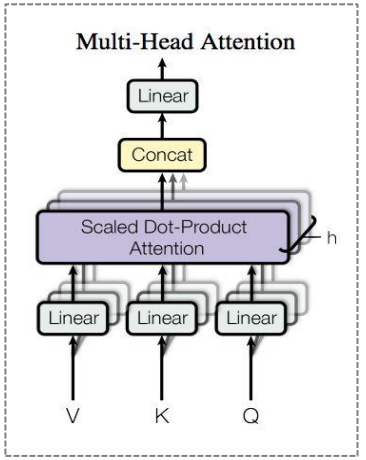
输入：我　弟弟　准备　一切　用品

◆ **Head2 Self Attention**

输出：A_2(我) A_2 (弟弟) A_2 (准备) A_2(一切) A_2 (用品)

输入：我　弟弟　准备　一切　用品





◆ **Multi-Head Self Attention**

输出：$C_{1\text{-}2}$(我) $C_{1\text{-}2}$ (弟弟) $C_{1\text{-}2}$ (准备) $C_{1\text{-}2}$(一切) $C_{1\text{-}2}$ (用品)

Concat

输入：

A_1(我) A_1 (弟弟) A_1 (准备) A_1(一切) A_1 (用品)

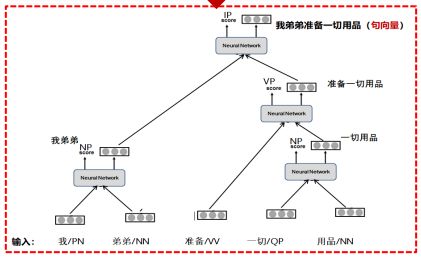
我　弟弟　准备　一切　用品

A_2(我) A_2 (弟弟) A_2 (准备) A_2(一切) A_2 (用品)

我　弟弟　准备　一切　用品

$C_{1\text{-}2}$(我) $C_{1\text{-}2}$ (弟弟) $C_{1\text{-}2}$ (准备) $C_{1\text{-}2}$(一切) $C_{1\text{-}2}$ (用品)

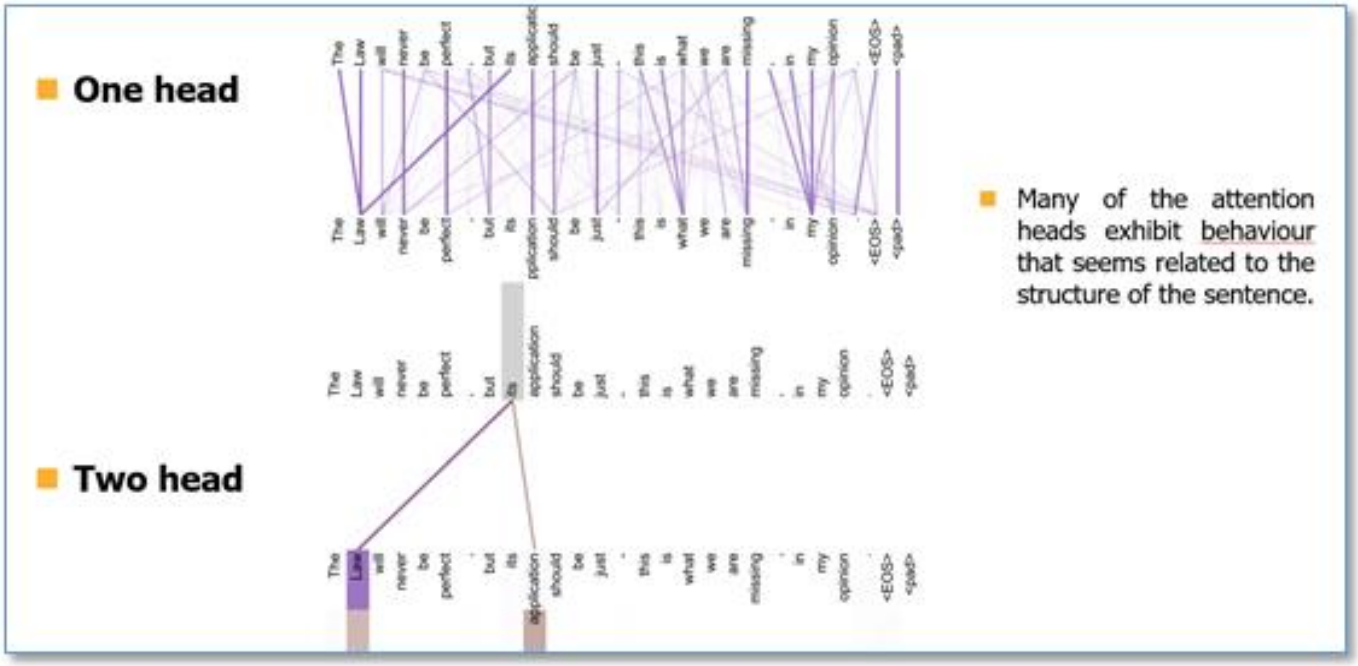
我　弟弟　准备　一切　用品



Multi-Head Self Attention 序列的含义？

10.2 注意力编码机制

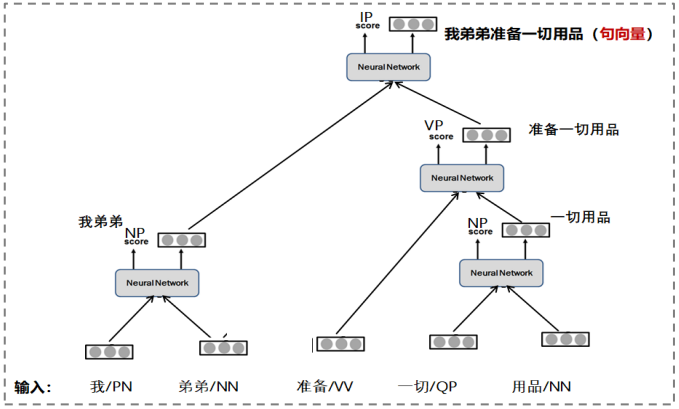
多头自注意力的可视化的效果



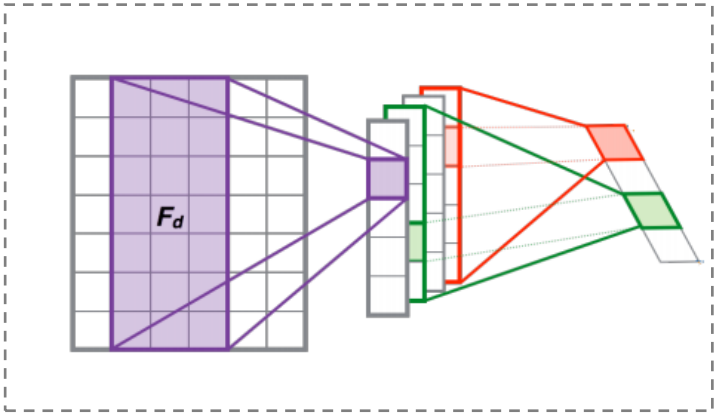
在两个头和单头的比较中，可以看到单头"its"这个词只能学习到"law"的依赖关系，而两个头"its"不仅学习到了"law"还学习到了"application"依赖关系。多头能够从不同的表示子空间里学习相关信息

10.2 注意力编码机制

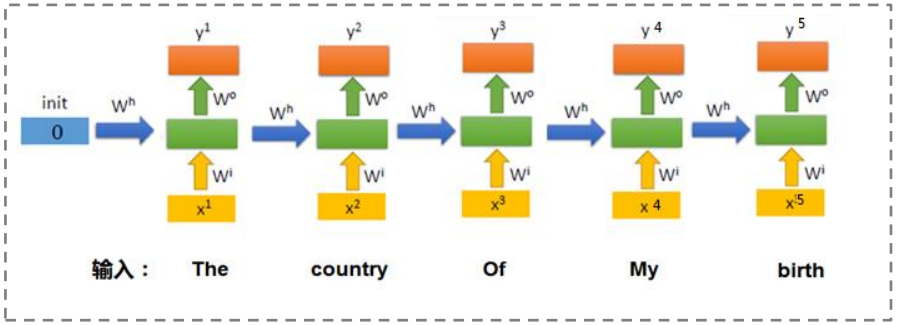
句向量



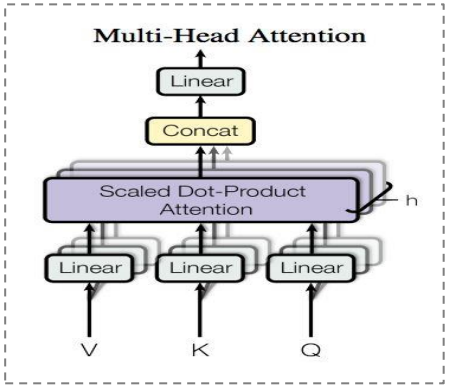
RvNN 句向量



CNN 句向量



RNN 句向量



Multi-Head Self Attention 句向量

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!

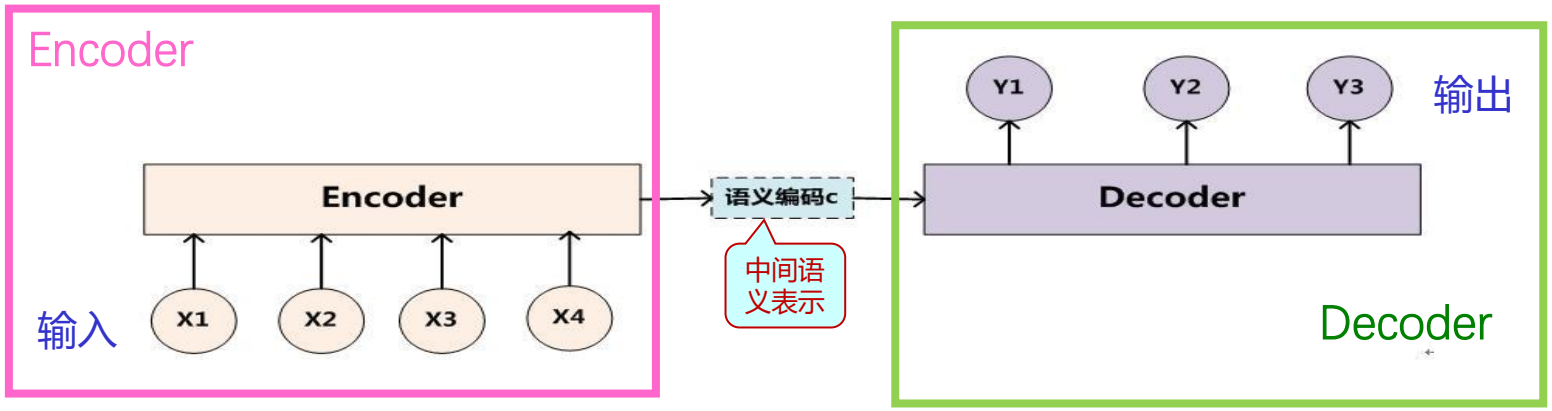
谢谢各位！

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

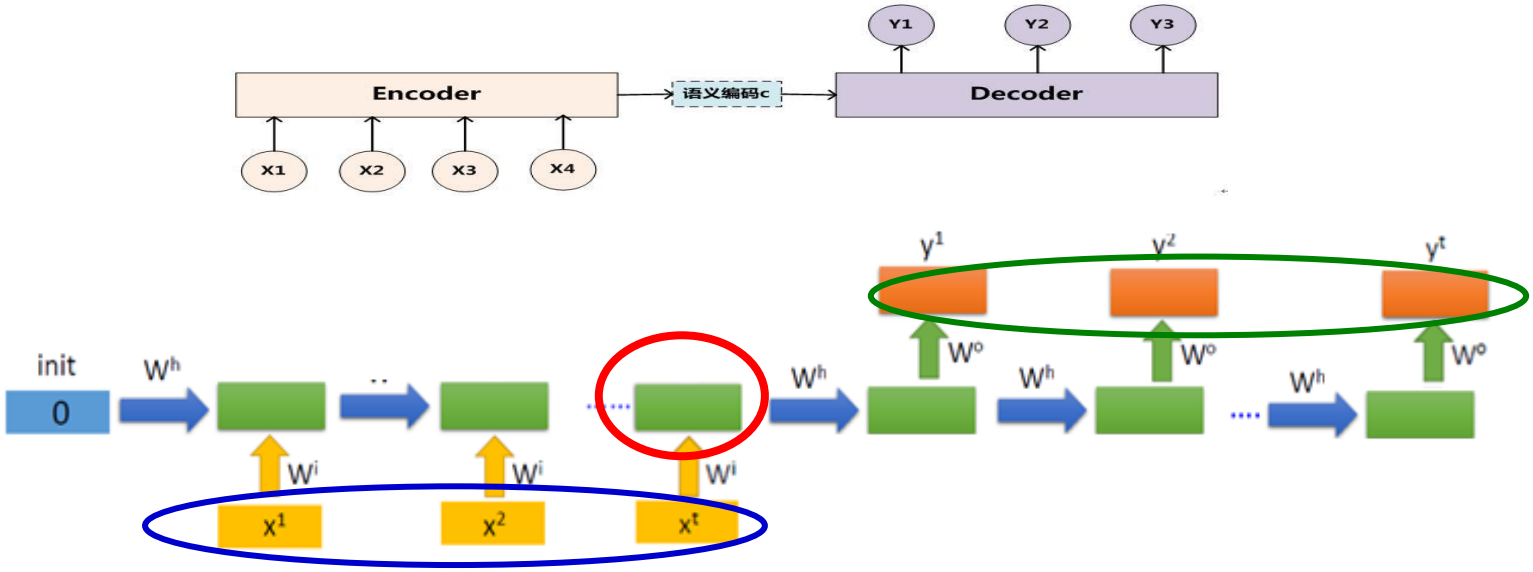
$$y_i = \mathcal{G}(C, y_1, y_2 \dots y_{i-1})$$

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

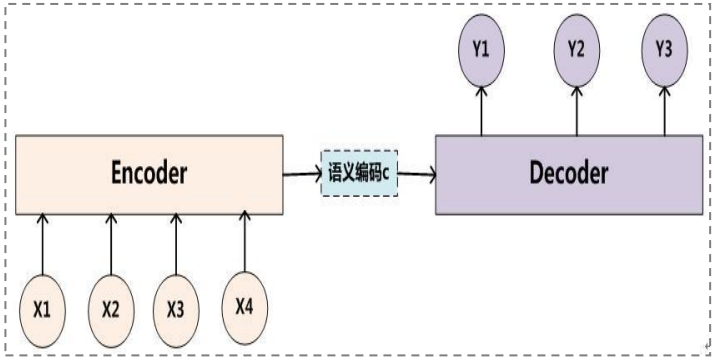
$$y_1 = f(C)$$
$$y_2 = f(C, y_1)$$
$$y_3 = f(C, y_1, y_2)$$
$$y_i = \mathcal{G}(C, y_1, y_2 \dots y_{i-1})$$

C 可以作为X的句向量

Encoder-Decoder 框架

Encoder-Decoder 框架 VS Attention机制

Encoder和Decoder一般采用RNN模型，对于句子比较长的情形，LSTM/GRU模型效果要明显优于RNN模型。但当句子长度超过 30 以后，LSTM模型的效果会急剧下降，一般此时会引入Attention模型。



Encoder-Decoder框架

附录完