



深度学习与自然语言处理

第9课：Attention机制

主讲人 郑元春

中科院大数据挖掘与知识管理重点实验室
中科院虚拟经济与数据科学研究中心
从事机器学习与自然语言处理相关研究





课程前言



Seq2Seq



注意力网络原理



代码讲解



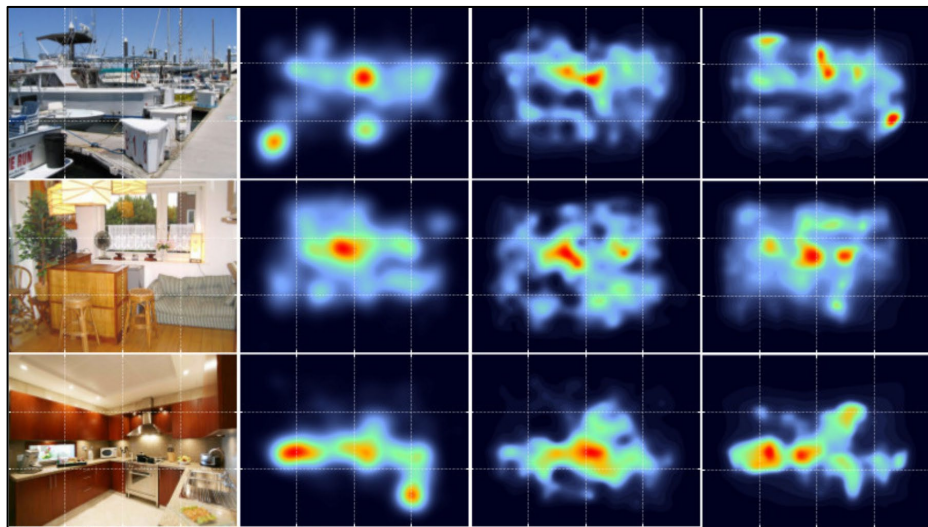
1. 课程前言

什么是注意力(Attention)?

人脑注意力机制的本质:

一种资源分配模型，在某个特定时刻，人类的注意力总是集中在画面（句子）的某个焦点部分，而对其他的部分视而不见。

在图像中，注意力只会在焦点区域开展，使得你大脑分析焦点区域内更加精细的内容。其他视野内的区域只会粗看其轮廓，并不会去关心其精细的内容。





1. 课程前言

什么是注意力(Attention)?



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.



给定图片，为图片生成描述性的文本。

从图中可以看出文本在产生的时候对原先的图像运用了注意力机制，使得单词只关注图像中的部分物体并能有效的描述这个物体。



1. 课程前言

什么是注意力(Attention)?

注意力是一个用来分配有限的信息处理能力的选择机制。

神经网络中可以存储的信息量称为**网络容量 (Network Capacity)**。一般来讲，利用一组神经元来存储信息时，其存储容量和神经元的数量以及网络的复杂度成正比。要存储的信息越多，神经元数量就要越多或者网络要越复杂，进而导致神经网络的参数成倍地增加。

我们人脑的生物神经网络同样存在网络容量问题，人脑中的工作记忆大概只有几秒钟的时间，类似于循环神经网络中的隐状态。而人脑每个时刻接收的外界输入信息非常多，包括来自于视觉、听觉、触觉的各种各样的信息。人脑在有限的资源下，并不能同时处理这些过载的输入信息。大脑神经系统有两个重要机制可以解决信息过载问题：

注意力机制和**记忆机制**。



课程前言



Seq2Seq



注意力网络原理



代码讲解



2. Seq2Seq

Encoder-Decoder框架

这个框架可以看做是深度学习领域的研究模式，应用场景十分的广泛，可看作是由一个句子（或是篇章）生成另一个句子（或是篇章）的通用处理框架。

$$\text{Source} = \langle x_1, x_2, \dots, x_m \rangle$$

$$\text{Target} = \langle y_1, y_2, \dots, y_n \rangle$$

Encoder: 对输入句子Source进行编码，将句子通过非线性的变换转化成一个中间的语句表示，即 $C = F(\text{Source})$

Decoder: 根据中间语义表示 C 和之前的已经生成的历史信息 y_1, y_2, \dots, y_{t-1} 来生成 t 时刻的单词，即 $y_t = G(C, y_1^{t-1})$

机器翻译

(Source: 语言1)

(Target: 语言2)

文本摘要

(Source: 文章)

(Target: 摘要)

问答系统

(Source: 问句)

(Target: 回答)

语音识别

(Source: 语音)

(Target: 文本)

图像描述

(Source: 图像)

(Target: 文本)



2. Seq2Seq

Encoder-Decoder框架

Encoder: 对输入句子Source进行编码, 将句子通过非线性的变换转化成一个中间的语句表示, 即 $C = F(Source)$

Decoder: 根据中间语义表示 C 和之前的已经生成的历史信息 y_1, y_2, \dots, y_{t-1} 来生成 t 时刻的单词, 即 $y_t = G(C, y_1^{t-1})$

➤ **RNN层**: 无法并行, 由于误差只能有效的传递到前两个历史时刻, 无法很好的学习到全局的结构信息。

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

➤ **CNN层**: 很容易并行, 容易捕捉到一些局部结构信息。

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

➤ **注意力机制**

$$y_t = f(x_t, A, B)$$

注意力机制直接利用整个Encoder序列对当前的目标进行计算, 因此能够很好的获得全局结构信息。同时, 对Decoder中每个单词的注意力计算相互之间没有依赖关系, 因此也能够进行并行计算。

已经被广泛使用的LSTM, GRU, 都并未脱离RNN的递归形式。RNN本身的结构比较简单, 也很适合序列建模, 明显的缺点之一就是无法并行, 因此速度很慢, 这是递归结构的天然缺陷。



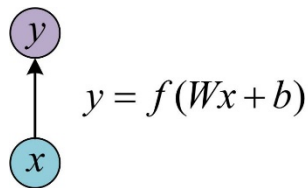
2. Seq2Seq

Encoder-Decoder框架

Encoder: 对输入句子Source进行编码, 将句子通过非线性的变换转化成一个中间的语句表示, 即 $C = F(Source)$

Decoder: 根据中间语义表示 C 和之前的已经生成的历史信息 y_1, y_2, \dots, y_{t-1} 来生成 t 时刻的单词, 即 $y_t = G(C, y_1^{t-1})$

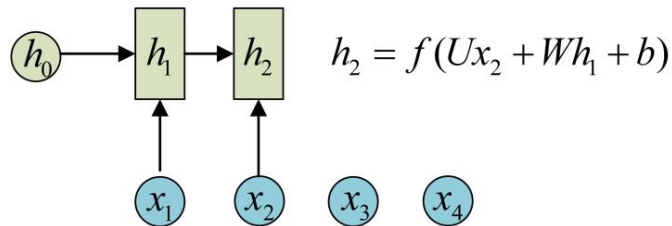
最基本的神经单元的工作模式



单层网络

序列数据:

- 语音处理
- 自然语言处理
- 时序处理



序列型数据



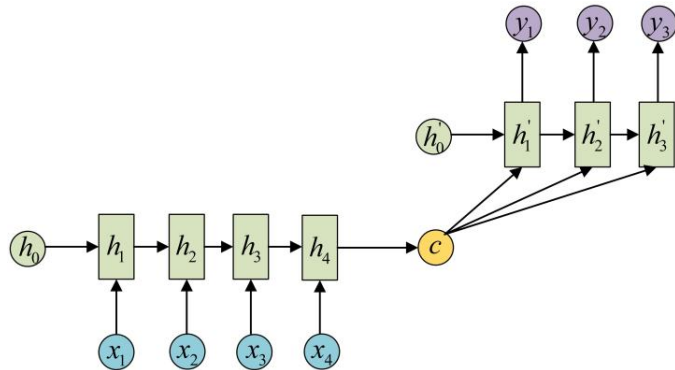
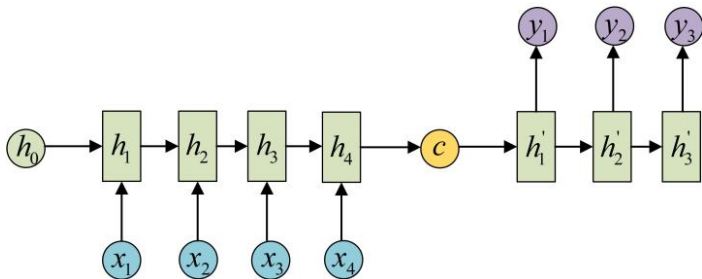
2. Seq2Seq

Encoder-Decoder框架

Encoder: 对输入句子Source进行编码, 将句子通过非线性的变换转化成一个中间的语句表示, 即 $C = F(Source)$

Decoder: 根据中间语义表示 C 和之前的已经生成的历史信息 y_1, y_2, \dots, y_{t-1} 来生成 t 时刻的单词, 即 $y_t = G(C, y_1^{t-1})$

经过了编码阶段, 得到了整个句子的隐含语义向量表示。再通过一个解码过程将整个的向量解码出来按照编码的逆顺序过程解码出另一种语言的句子。





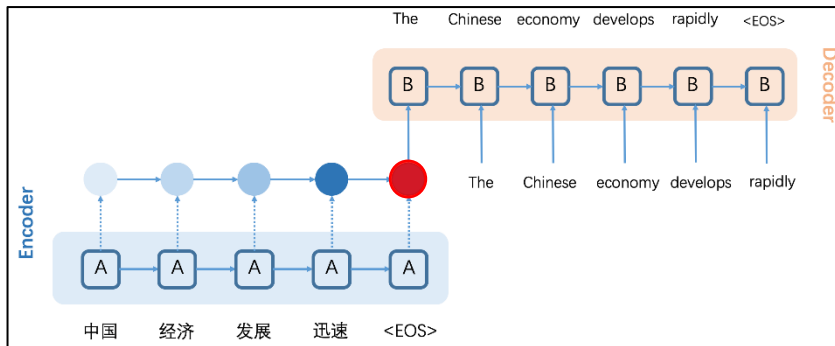
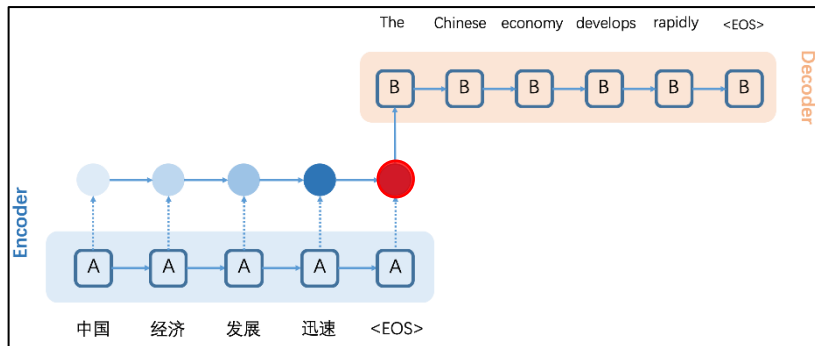
2. Seq2Seq

Encoder-Decoder框架

Encoder: 对输入句子Source进行编码, 将句子通过非线性的变换转化成一个中间的语句表示, 即 $C = F(Source)$

Decoder: 根据中间语义表示 C 和之前的已经生成的历史信息 y_1, y_2, \dots, y_{t-1} 来生成 t 时刻的单词, 即 $y_t = G(C, y_1^{t-1})$

经过了编码阶段, 得到了整个句子的隐含语义向量表示。再通过一个解码过程将整个的向量解码出来按照编码的逆顺序过程解码出另一种语言的句子。





课程前言



Seq2Seq



注意力网络原理



代码讲解



3. 注意力网络原理

注意力机制

Encoder: 对输入句子Source进行编码, 将句子通过非线性的变换转化成一个中间的语句表示, 即 $C = F(\text{Source})$

Decoder: 根据中间语义表示 C 和之前的已经生成的历史信息 y_1, y_2, \dots, y_{t-1} 来生成 t 时刻的单词, 即 $y_t = G(C, y_1^{t-1})$

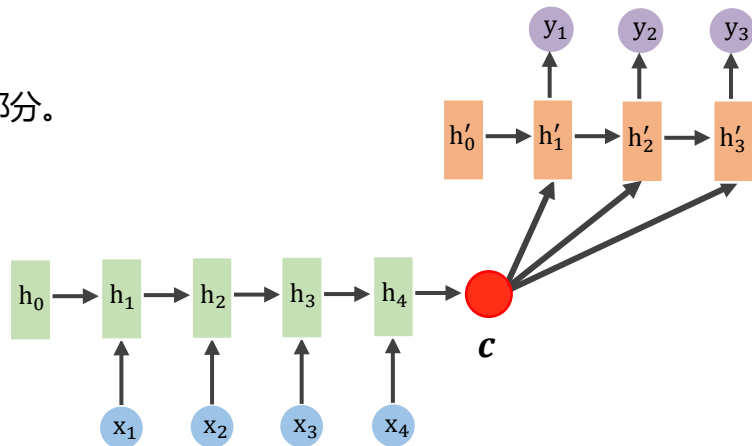
Encoder把一个变长的输入序列 $X = \{x_1, x_2, \dots, x_t\}$ 编码成一个固定长度隐向量(背景向量或上下文向量context) c 。

- 作为初始向量, 初始化Decoder。
- 作为背景向量, Decoder每一步都会用它当作输入的一部分。

$$h'_1 = F(c)$$

$$h'_2 = F(c, h'_1)$$

$$h'_3 = F(c, h'_1, h'_2)$$





3. 注意力网络原理

注意力机制

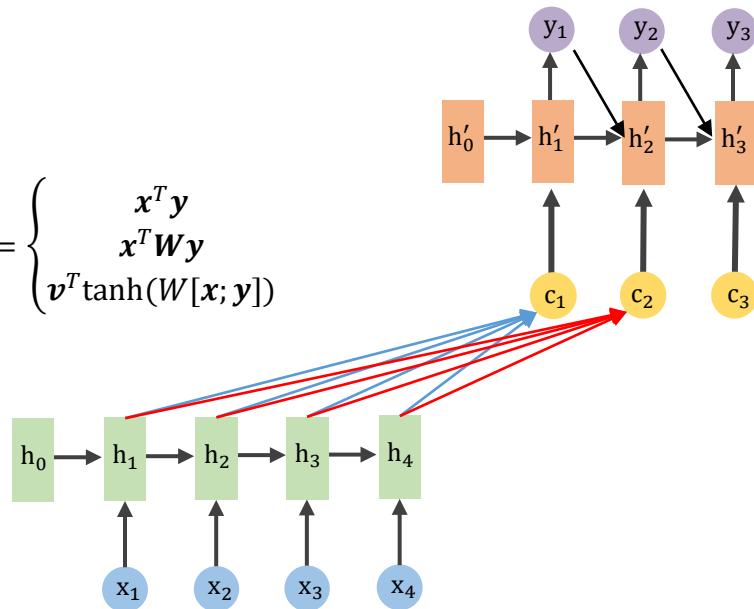
Encoder: 对输入句子Source进行编码, 将句子通过非线性的变换转化成一个中间的语句表示, 即 $C = F(Source)$

Decoder: 根据中间语义表示 C 和之前的已经生成的历史信息 y_1, y_2, \dots, y_{t-1} 来生成 t 时刻的单词, 即 $y_t = G(C, y_{1:t-1}^t)$

$$c_i = \sum_{j=1}^t \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^t \exp(e_{ik})} \quad \text{其中, } e_{ij} = a(h'_{i-1}, h_j), \quad a(x, y) = \begin{cases} x^T y \\ x^T W y \\ v^T \tanh(W[x; y]) \end{cases}$$

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





3. 注意力网络原理

注意力机制

根据注意力计算涉及的范围，以及注意力的计算方式不同，可以将注意力机制划分为以下几种情况：

Soft Attention

Hard Attention

Global Attention

Local Attention



3. 注意力网络原理

注意力机制

根据注意力计算涉及的范围与注意力的计算方式不同，可以将注意力机制划分为以下几种情况：

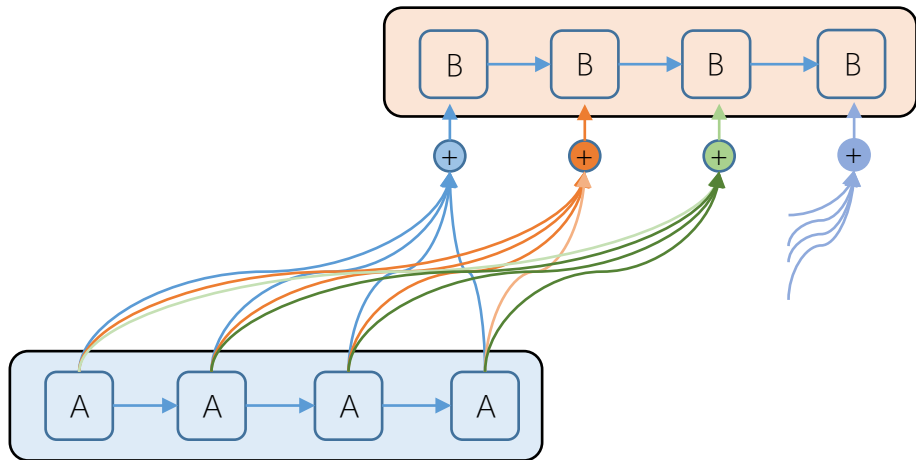
Soft Attention

Hard Attention

Global Attention

Local Attention

由于每一个 c_i 都是通过对原始输入 x 在输出侧的 y_i 上的一个概率分布来进行计算的，因此获得是当前需要解码位置在原始输入序列上的一个注意力分布，这种注意力分布的计算方式称为Soft Attention，可以被嵌入到模型里面去，直接训练。





3. 注意力网络原理

注意力机制

根据注意力计算涉及的范围与注意力的计算方式不同，可以将注意力机制划分为以下几种情况：

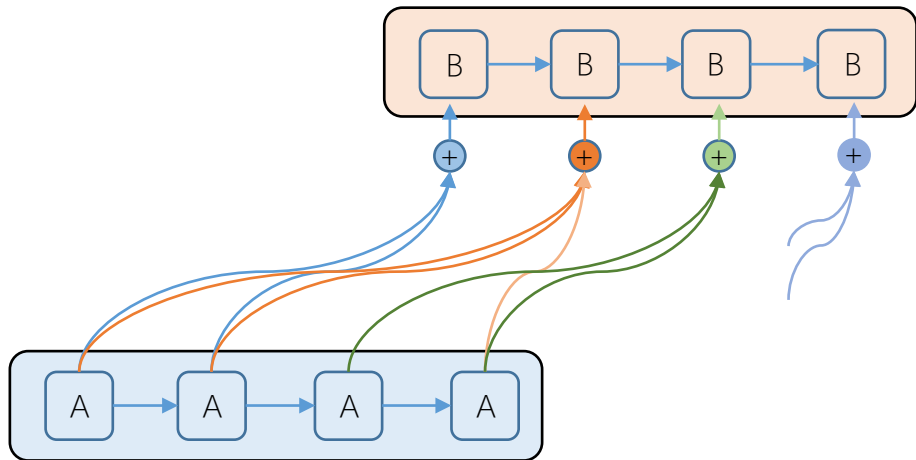
Soft Attention

Hard Attention

Global Attention

Local Attention

hard attention是一个随机的过程，它不会选择整个encoder的输出作为其输入，而是会依据概率来采样encoder端的某些隐藏层作为其attention。为了实现梯度的反向传播，需要使用蒙特卡洛的方法来估计模块的梯度。





3. 注意力网络原理

注意力机制

根据注意力计算涉及的范围与注意力的计算方式不同，可以将注意力机制划分为以下几种情况：

Soft Attention

Hard Attention

Global Attention

Local Attention

➤ Global Attention:

与传统的Attention模式一样。所有的encoder端的hidden state都被用于计算背景向量的权重。

➤ Local Attention:

将Soft & Hard Attention结合起来，每次先为decoder端当前的词，预测一个source端对齐位置 (aligned position) p_t 。然后基于 p_t 选择一个窗口，用于计算背景向量 c_t 。

$$p_t = S \cdot \text{sigmoid}(v_p^T \tanh(W_p h_t))$$

$$a_t(s) = \text{align}(h_t, h'_t) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right)$$



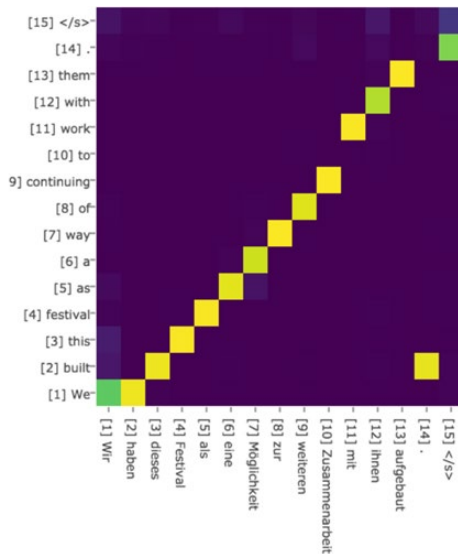
3. 注意力网络原理

注意力在文本上的物理含义

Attention模型的物理含义是什么？

可看作是输出(Target)句子中某个单词和输入(Source)句子每个单词的对齐模型。

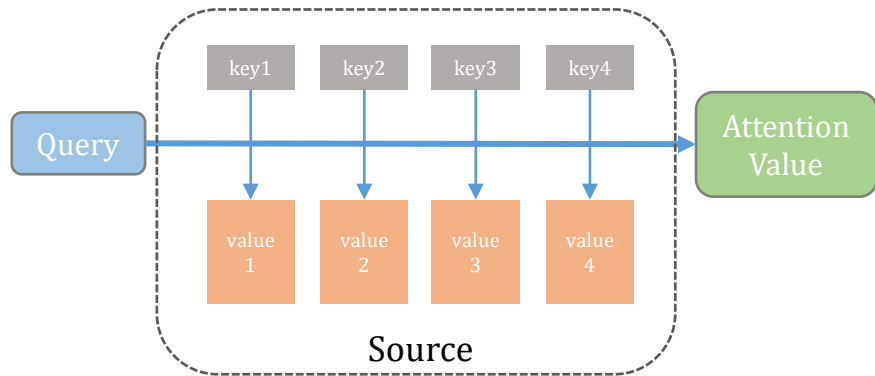
这在机器翻译中是非常直观的：传统的统计机器翻译一般在做的过程中会有一个专门的短语对齐步骤，而注意力模型的作用其实是相同的。





3. 注意力网络原理

Attention机制的思想本质



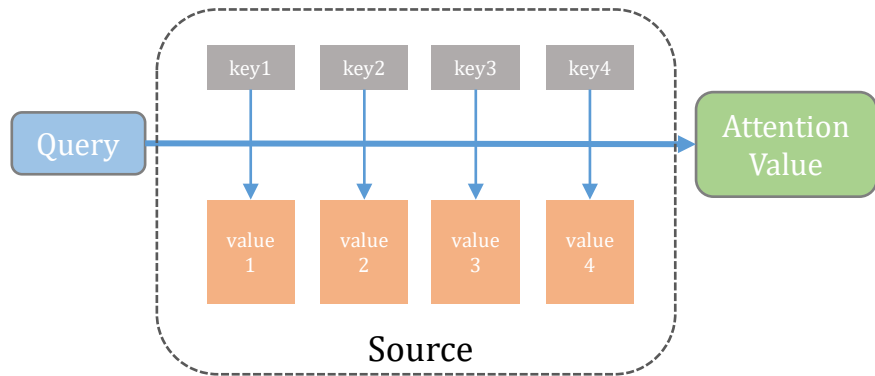
本质上，Attention机制是对source中元素的value值进行加权求和，而query和key用来计算对应value的权重系数。

$$\text{attention}(\text{query}, \text{source}) = \sum_{i=1}^t \text{sim}(\text{query}, \text{key}_i) * \text{value}_i$$



3. 注意力网络原理

Attention机制的思想本质



Attention仍然可以理解为从大量信息中有选择的筛选出少量重要信息并聚焦到这些重要信息上，忽略大多数不重要的信息。

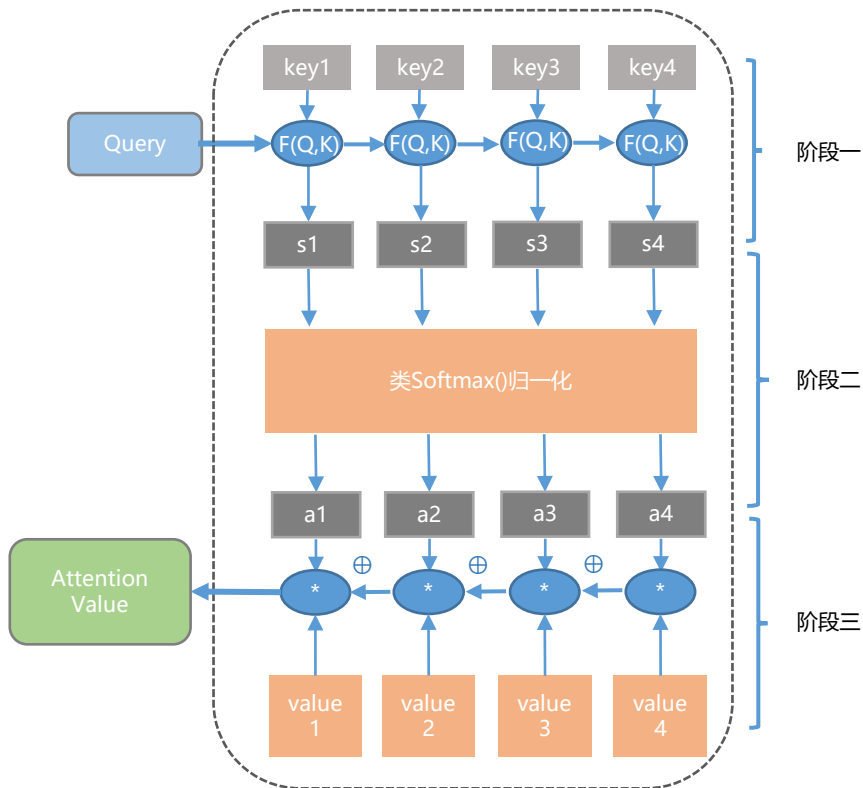
聚焦的过程体现在权重系数的计算上，权重越大越聚焦到其对应的value上，即权重代表了信息的重要性，而value是其对应的信息。



3. 注意力网络原理

Attention机制的思想本质

- 阶段一：根据query \leftrightarrow key计算相似性或相关性
- 阶段二：原始分值归一化、概率化
- 阶段三：根据权重系数对value进行加权求和





3. 注意力网络原理

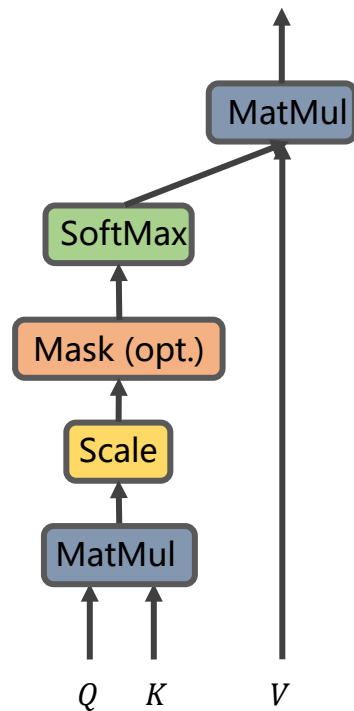
Attention机制的思想本质

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

$$Q \in R^{n \times d_k}, K \in R^{m \times d_k}, V \in R^{m \times d_v}$$

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，最后的结果也是一个 $n \times d_v$ 的矩阵。

即， $n \times d_k$ 的序列 Q 编码成了一个新的 $n \times d_v$ 的序列。



Scaled Dot-Product Attention



3. 注意力网络原理

Attention机制的思想本质

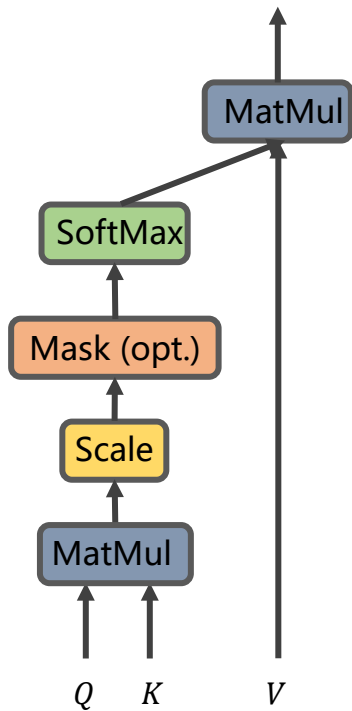
那么怎么理解这种结构呢？

$$Attention(q_t, K, V) = \sum_{s=1}^m \frac{1}{Z} \exp\left(\frac{\langle q_t, k_s \rangle}{\sqrt{d_k}}\right) v_s$$

K 和 V 是一一对应的，每次拿一个 q_t 和各个key做内积并softmax，得到 q_t 与各个 v_s 的相似程度，然后加权求和，得到一个 d_v 的向量。

其中，因子 $\sqrt{d_k}$ 起到调节的作用，使得内积不至于太大（太大的话，softmax就是非0即1了，不够soft）。

这个定义只是注意力的一种形式，还有一些其他的选择，比如query和key的运算方式并不一定是点乘，还可以是拼接后再内积一个参数向量，甚至是权重都不一定要归一化等。



Scaled Dot-Product Attention



3. 注意力网络原理

Mutil-Head Attention

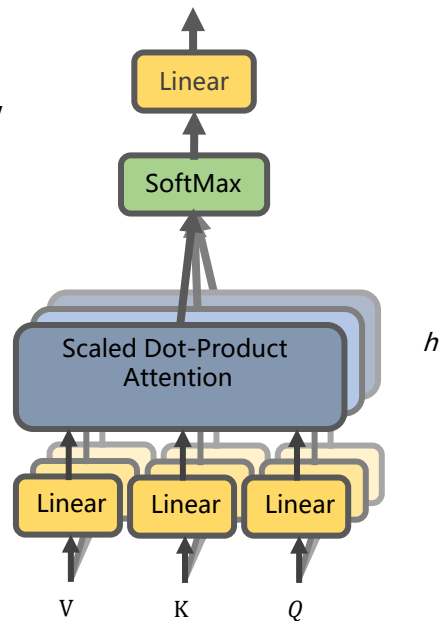
Mutil-Head Attention是Google提出的新概念，是Attention机制的完善，不过从形式上看，它其实就是把Q,K,V通过参数矩阵映射一下，然后再做Attention，把这个过程重复做了h次，结果拼接起来就行了。

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$$W_i^Q \in R^{d_k \times \check{d}_k}, W_i^K \in R^{d_k \times \check{d}_k}, W_i^V \in R^{d_v \times \check{d}_v}$$

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)$$

最后得到一个 $n \times (h\check{d}_k)$ 的序列，所谓的“多头”，就是多做几次同样的事情（参数不共享），然后把结果进行拼接。



Multi-Head Attention



3. 注意力网络原理

Self Attention

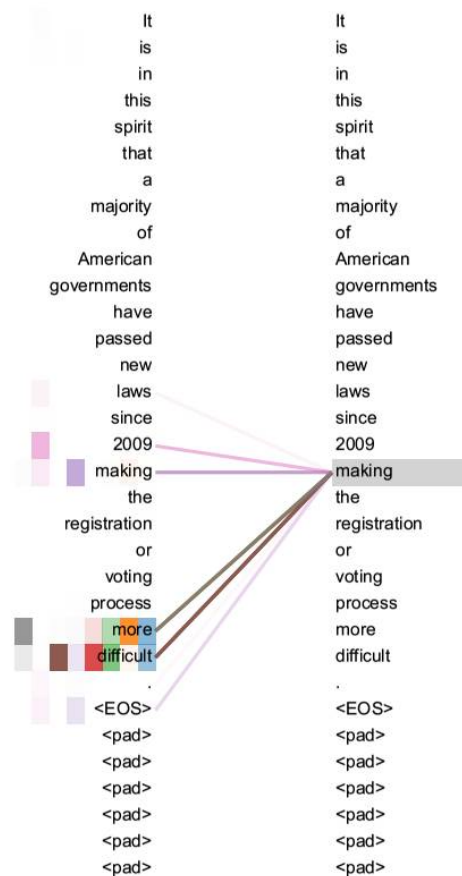
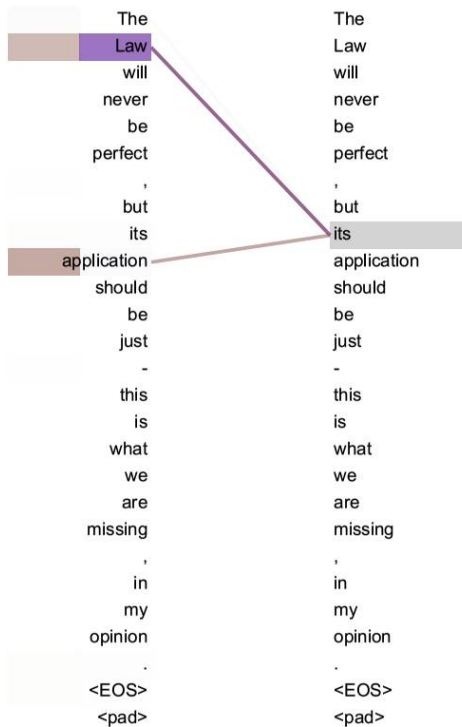
- 传统的Attention是基于source端和target端的隐变量计算的。得到的结果是source端的每个单词与target端的每个词之间的依赖关系。
- Self Attention分别在source端和target端进行，但是分别在两端捕捉自身的词与词之间的依赖关系，然后再把source端得到的self Attention加入到target端得到的self Attention中，捕捉source端和target端词与词之间的依赖关系。
- Self Attention要比传统的Attention Mechanism效果好，主要原因之一是：传统的Attention机制忽略了source/target端自身的词之间的依赖关系。



3. 注意力网络原理

Self Attention

Self Attention有什么增益或是好处?





3. 注意力网络原理

Self Attention

- 在Google的论文中，大部分的Attention都是Self Attention。
- Self Attention就是 $\text{Attention}(X, X, X)$, X 是前面说的输入序列，也就是在序列内部做Attention，寻找序列内部的联系。
- Google论文的主要贡献之一就是它表明了内部注意力在机器翻译（甚至在一般的seq2seq）任务的序列编码上是相当重要的，而之前关于seq2seq的研究基本都只是把注意力机制用在解码端。



3. 注意力网络原理

Position Embedding

- Attention模型并不能捕捉序列的顺序，换句话说，如果将K,V按行打乱顺序（相当于句子中的词序打乱），那么Attention模型的结果还是一样，这就表明了它顶多是一个非常巧妙的BOW模型。
- 顺序对于时间序列至关重要，如果学习不到顺序信息，那么效果会大打折扣。
- Position Embedding将每个位置编号，然后每个编号对应一个向量，通过结合位置向量和词向量，就给每个词都引入了一定的位置信息，这样Attention就可以分辨出不同位置的词了。



3. 注意力网络原理

Position Embedding

- 以前的RNN, CNN模型中都出现过Position Embedding, 但是那些模型本身就能够学习到序列的位置信息, 所以Position Embedding(PE)不是必须的。但是在Attention模型中, PE是位置信息的唯一来源, 因此它是模型的核心成分之一, 并非仅仅是简单的辅助手段。
- 在之前的PE中, 基本都是根据任务训练出来的向量, 而Google直接给出了一个构造PE的公式:

$$PE_{2i}(p) = \sin\left(\frac{p}{10000^{\frac{2i}{d_{pos}}}}\right)$$

$$PE_{2i+1}(p) = \cos\left(\frac{p}{10000^{\frac{2i}{d_{pos}}}}\right)$$



3. 注意力网络原理

Position Embedding

- Position Embedding 本身是一个绝对位置的信息，但是在语言中，相对位置也很重要，Google选择前述的位置向量公式一个重要的原因是：由于有 $\sin(\alpha + \beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$ 和 $\cos(\alpha + \beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$,这表明位置 $p + k$ 的向量可以表明位置为 p 的向量的线性变换，这提供了表达相对位置的可能性。
- 结合位置向量和词向量的几个可选方案：可以把他们拼接起来作为一个新向量，也可以把位置向量定义为跟词向量一样大小，然后两者加起来。Facebook的论文和Google的论文中用的都是后者。直觉上相信会导致信息损失，似乎是不可取，但是实验证明相加也是很好的方案。



3. 注意力网络原理

缺点

- Attention层的好处就是能够一步到位的捕捉全局的联系，因为它直接把序列两两比较（代价是计算量变为 $O(n^2)$ ）。相比之下，RNN需要一步步地递推才能捕捉到，而CNN需要通过层叠扩大感受野，这是Attention层的明显优势。
- Attention虽然跟CNN没有直接联系，但是事实上充分的借鉴了CNN的思想，比如Multi-Head Attention就是Attention做多次然后拼接，这个CNN的多个卷积核思想是一致的，论文用到的残差结构都源于CNN网络。
- 无法对位置信息进行很好的建模，PE的引入无法从根本上解决这个问题。
- 并非所有的问题都是需要长程的全局的依赖，也有很多的问题只是依赖于局部结构，这时候用纯的Attention不是很好。所以论文中还是提到了一个受限Attention的概念。

图层类型	每层复杂度	顺序操作	最大路径长度
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
循环	$O(n \cdot d^2)$	$O(n)$	$O(n)$
卷积	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$
Self-Attention (受限)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n / r)$

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., & Gomez, A. N., et al. (2017). Attention is all you need. arXiv.

n 是句子长度， d 是词向量维度， k 为卷积核大小， r 为受限Attention的邻域大小。