一、注意力机制

1.注意力的定义

从大量输入信息中选择小部分的有用信息来重点处理，并忽略其他信息的能力。

1.1.注意力的类别

自上而下的有意识的注意力（聚焦式注意力）：用脑子，内心想要关注的信息

自下而上的无意识的注意力（基于显著式性的注意力）：被一些外界事物所刺激，如看到一群人有几个人非常亮眼，由此想要着重关注那些亮眼的人。

2.注意力机制（注意力模型）

是将有限的计算资源用来处理优先级更高（更重要）的信息、用来解决信息超载问题的资源分配方案

2.0.注意力分布

为了从N组输入信息中选出与某个任务相关的信息，需要引入一个和某个任务相关的表示，称为查询向量（Query Vector），通过打分函数来计算每个输入向量和查询向量之间的相关性。

注意力分布概率

使用注意力变量$z\epsilon[1,N]$来表示本次"查询：选择出和某个任务相关的信息，即$z=n$表示本次选择了索引为n的输入向量。为了方便计算，采用"软性"的信息选择机制（概率），首先计算在给定*q*和*X*下，选择第n个输入向量的概率(注意力分布)α_n

$$\begin{aligned}\alpha_n &= p(z=n|X,q) &(2.1)\\ &= softmax(s(x_n,q)) &(2.2)\\ &= \frac{exp(s(x_n,q))}{\sum_{j=1}^{N}exp(s(x_j,q))} &(2.3)\end{aligned}$$

注意力打分函数为s，x为输入向量，q为查询向量，W、U、V为可学习参数

$$\begin{aligned}s(x,q) &= x^TWq &(2.4)\\ W &= U^TV，\text{可得} &(2.5)\\ s(x,q) &= (Ux)^T(Vq) &(2.6)\end{aligned}$$

即分别对x和q进行线性变换后计算点积

2.1.注意力机制的计算步骤

1）从输入的N组输入信息中计算注意力分布

2）根据注意力分布来计算输入信息的平均加权

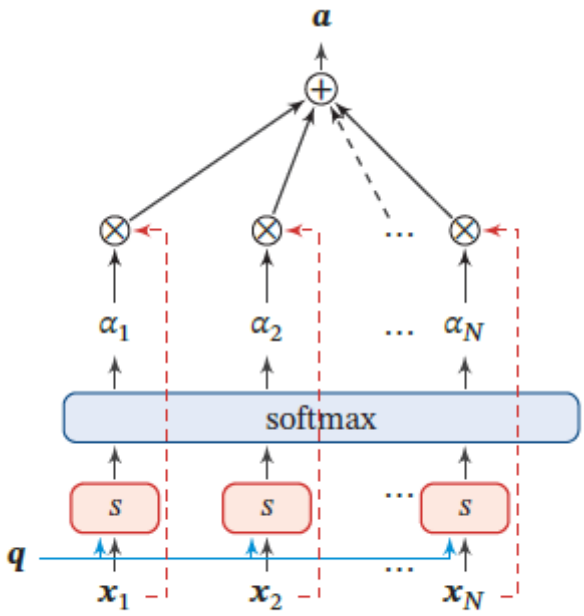
加权平均

使用概率机制对输入信息进行汇总，即

$$att(X,q)=\sum_{n=1}^{N}\alpha_n x_n \tag{2.7}$$

$$=\mathbb{E}_{z\sim p(x|X,q)}[x_z] \tag{2.8}$$

（9）为软性（概率）注意力机制



(a) 普通模式

2.2 注意力机制的变体

2.2.1 硬性注意力（Head Attention)

只关注某一个输入向量

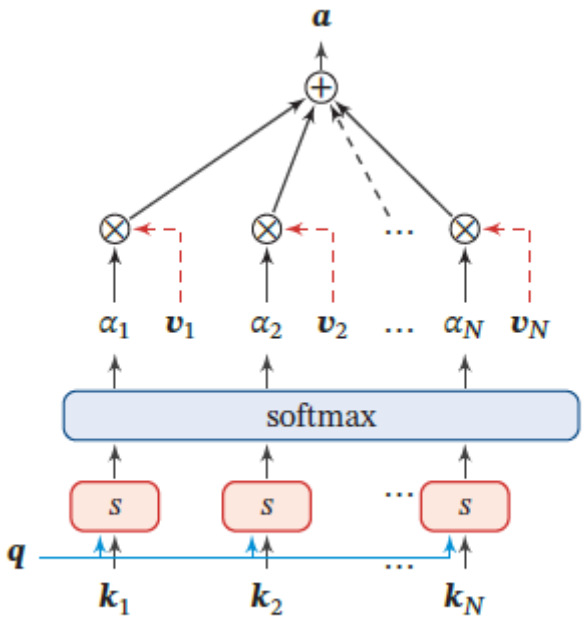
2.2.2 键值对注意力（Key-Value Pair Attention)

其中“Key”(能否片面的理解为，“Key”个Value，即值为“Value”的数一共有“Key”个)计算注意力分布α（这里不计算值与任务的适配程度，而是看值的数量与该任务的适配程度？），”Value”计算聚合信息

$(K,V)=[(k_1,v_1),\ldots,(k_N,v_N)]$表示N组输入信息，给定任务相关的查询向量q时，注意力函数为

$$att((K,V),q)=\sum_{n=1}^{N}\alpha_n v_n \tag{2.9}$$

$$=\sum_{n=1}^{N}\frac{exp(s(k_n,q))}{\sum_j exp(s(k_j,q))}v_n \tag{2.10}$$



(b) 键值对模式

2.2.3 多头注意力（Multi-Head Attention）

利用多个查询$Q = [q_1, q_2, \cdots, q_M]$来并行的从输入信息中选取多组信息，每个注意力关注输入信息的不同部分：

$$att((K,V),Q) = att((K,V),q_1) \bigoplus \ldots \bigoplus att((K,V),q_M) \tag{2.11}$$

2.2.4 结构化注意力（Hierarchical Attention）

大多数输入信息本身含有层次结构，比如文本可以分为词、句子、段落、篇章等不同粒度的层次，还可以假设注意力为上下文相关的二项分布，使用图模型（上下文为节点，之间的关系为边？）来构建更复杂的结构化注意力分布

2.2.5 指针网络(Pointer Network)

将注意力分布作为软性的网络来指出相关信息的位置，**而不计算注意力分布下输入信息的加权平均**

指针网络是一种序列到序列模型：

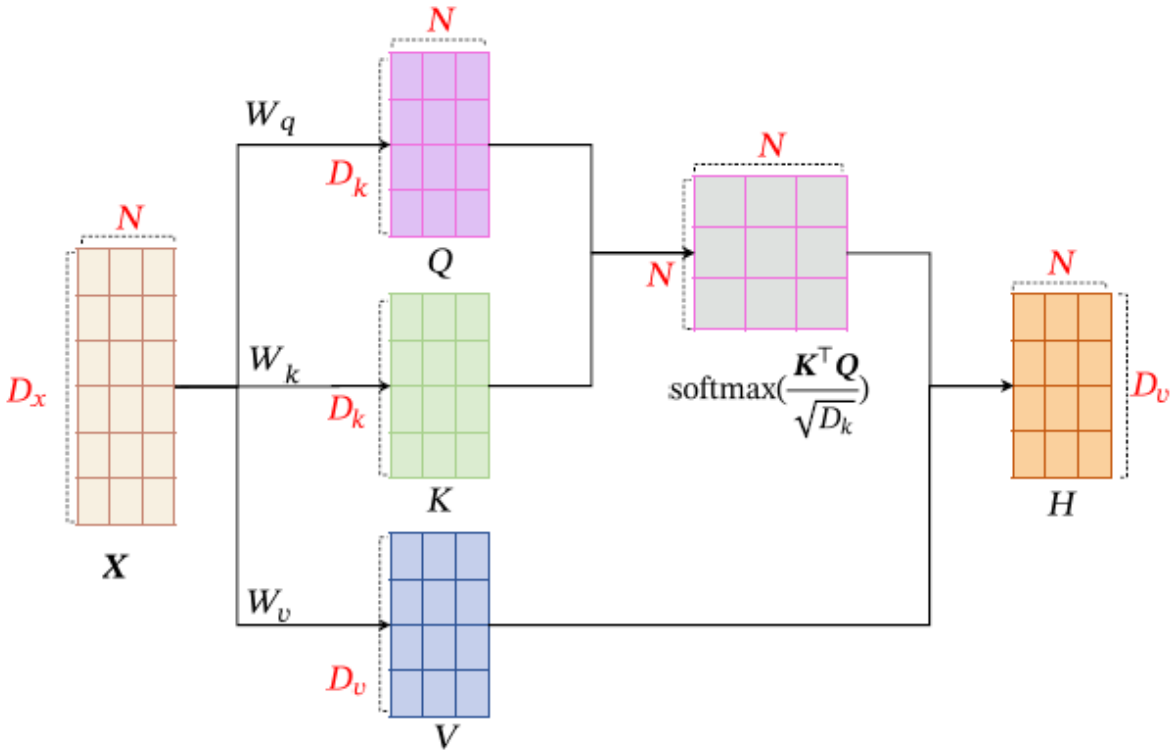
输入：长度为N的向量序列$X = x_1, x_2, \ldots, x_N$，

输出：长度为M的下标序列$c_{1:M} = c_1, c_2, \ldots, c_M, c_m \in [1, N]$，

2.3 自注意力模型（Self-Attention Model）

全连接网络可以直接建模远距离依赖的模型，但无法处理长度变化的输入序列，不同的序列长度，需要用到的权重也不一定相同。由此引入自注意力模型，用来动态的生成不同连接的权重。

为了提高模型能力，自注意力模型经常采用查询-键-值（Query-Key-Value，QKV）模式



假设输入序列为$X = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^{D_x \times N}$，输出序列为$H = [h_1, h_2, \ldots, h_N] \in \mathbb{R}^{D_v \times N}$

自注意力模型的具体计算过程为：

1）对于每个输入x_i,我们首先将其线性映射到三个不同的空间，得到查询向量$q_i \in \mathbb{R}^{D^k}$、键向量$k_i \in \mathbb{R}^{D^k}$、值向量$v_i \in \mathbb{R}^{D_v}$

$$Q = W_q X \in \mathbb{R}^{D^k \times N} \tag{2.12}$$
$$K = W_k X \in \mathbb{R}^{D^k \times N} \tag{2.13}$$
$$V = W_v X \in \mathbb{R}^{D^v \times N} \tag{2.14}$$

其中$W_q \in R^{D_k \times D_x}, W_k \in R^{D_k \times D_x}, W_v \in R^{D_v \times D_x}$为线性映射的参数矩阵（这里的$W_q, W_k, W_v$是动态生成的，应该是指能适应输入向量长度的变化），
$Q = [q_1, q_2, \ldots, q_N], K = [k_1, l_2, \ldots, k_n], V = [v_1, v_2, \ldots, v_N]$分别是由查询向量q、键向量key和值向量value构成的矩阵。

2）对于每个查询向量$q_n \in Q$，利用公式（2.9)中提到的键值对注意力机制，得到输出向量h_n（简单理解为一共有N个查询向量，每个查询向量都能得到一份“加权平均”,现在计算的是第n个查询向量的加权平均，也可以理解为第n个的输出，共N个输出）

$$h_n = att((K, V), q_n) \tag{2.15}$$
$$= \sum_{j=1}^{N} \alpha_{nj} v_j \tag{2.16}$$
$$= \sum_{j=1}^{N} softmax(s(k_j, q_n)) v_j \tag{2.17}$$

缩放点积

$$s(x, q) = \frac{x^T q}{\sqrt{D}} \tag{2.18}$$

若使用缩放点积作为注意力打分函数，最终的输出向量序列可以简写为

$$H = V softmax(\frac{K^T Q}{} \tag{2.19}$$

$$\alpha_{ij} = \text{softmax}\left(\frac{q_j^T k_j}{\sqrt{D_k}}\right) \tag{2.19}$$

自注意力模块可以作为神经网络中的一层来使用，可以在必要的时候替代卷积层、循环层，也可以和其他层一起交叉使用（如：自注意模块的输入X可以是卷积层或循环层的输出），自注意力模型计算的权重α_{ij}只依赖于q_j和k_j的相关性，而忽略了输入信息的位置信息（每次迭代只输入一个x_n，而没有考虑与其他输入向量的关系）。自注意力模型一般需要加入位置编码信息来修正；自注意力模型可以拓展为多头自注意力(Multi-Head Self-Attention)模型，在多个不同的投影空间中捕捉不同的交互信息。