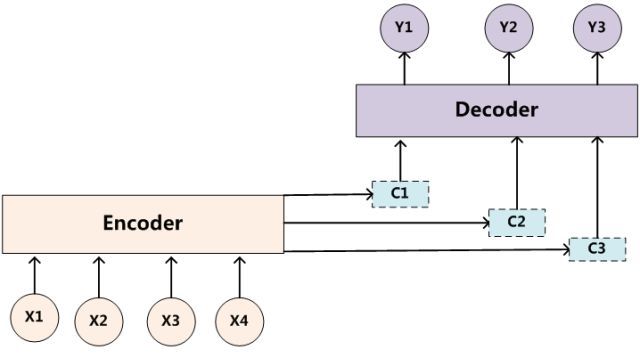
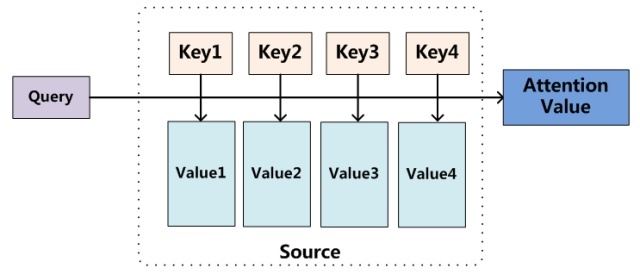
**Attention机制的理解**

参考链接：<https://zhuanlan.zhihu.com/p/35571412>

这样的过程称之为一个分心模型。这是因为sorce中每一个元素对语义编码C的作用是相同的，也就是每一个元素的重要程度都是一样的。可实际上，‘我’这个元素对target中的'I'的结果是影响最大的，其他元素的影响可以说是微乎其微。但是在上述模型中，这个重要程度没有被体现出来，所以这是一个分心的模型。而与其对应的注意力模型就是要从序列中学习到每一个元素的重要程度，然后按重要程度将元素合并。这就表明了序列元素在编码的时候，所对应的语义编码C是不一样的。所以一个体现attention机制运行的图示如下:

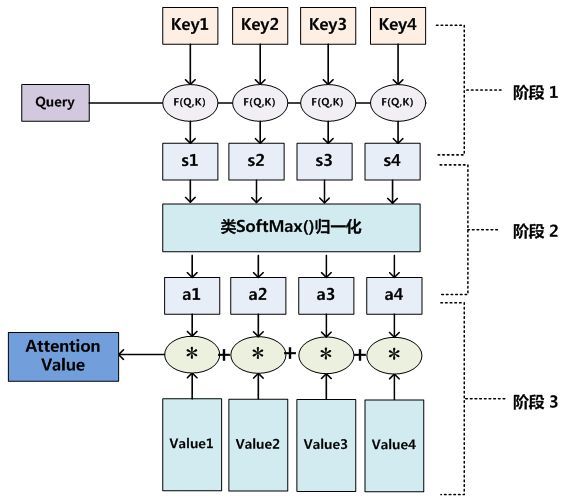


我们现在抛开encoder-decoder来说attention机制。既然attention是一组注意力分配系数的，那么它是怎样实现的？这里要提出一个函数叫做attention函数，它是用来得到attention value的。比较主流的attention框架如下：



上图其实可以描述出attention value的本质：它其实就是一个查询(query)到一系列键值(key-value)对的映射。

这个也就是attention函数的工作实质，如下图所示：



**Attention函数工作机制**

attention函数共有三步完成得到attention value。

- Q与K进行相似度计算得到权值

- 对上部权值归一化

- 用归一化的权值与V加权求和

此时加权求和的结果就为注意力值。公式如下:

Attention Value = similarity(QK^T)V

在自然语言任务中，往往K和V是相同的。这时计算出的attention value是一个向量，代表序列元素$xj$的编码向量。此向量中包含了元素$xj$的上下文关系，即包含全局联系也拥有局部联系。全局联系是因为在求相似度的时候，序列中元素与其他所有元素的相似度计算，然后加权得到了编码向量。局部联系可以这么解释，因为它所计算出的attention value是属于当前输入的$x\_j$的。这也就是attention的强大优势之一，它可以灵活的捕捉长期和local依赖，而且是一步到位的。

**换一个角度理解Attention机制说到这里**

大家应该也明白attention机制具体是个啥吧。其实也不是很玄乎，就是把序列中各个元素分配一个权重系数。上面从attention函数得到了attention机制工作过程。现在换一个角度来理解，我们将attention机制看做软寻址。就是说序列中每一个元素都由key(地址)和value(元素)数据对存储在存储器里，当有query=key的查询时，需要取出元素的value值(也即query查询的attention值)，与传统的寻址不一样，它不是按照地址取出值的，它是通过计算key与query的相似度来完成寻址。这就是所谓的软寻址，它可能会把所有地址(key)的值(value)取出来，上步计算出的相似度决定了取出来值的重要程度，然后按重要程度合并value值得到attention值，此处的合并指的是加权求和。

**优点:**

**一步到位的全局联系捕捉** 上文说了一些，attention机制可以灵活的捕捉全局和局部的联系，而且是一步到位的。另一方面从attention函数就可以看出来，它先是进行序列的每一个元素与其他元素的对比，在这个过程中每一个元素间的距离都是一，因此它比时间序列RNNs的一步步递推得到长期依赖关系好的多，越长的序列RNNs捕捉长期依赖关系就越弱。

**并行计算减少模型训练时间** Attention机制每一步计算不依赖于上一步的计算结果，因此可以和CNN一样并行处理。但是CNN也只是每次捕捉局部信息，通过层叠来获取全局的联系增强视野。

**模型复杂度小，参数少** 模型复杂度是与CNN和RNN同条件下相比较的。

**缺点:**

缺点很明显，attention机制不是一个"distance-aware"的，它不能捕捉语序顺序(这里是语序哦，就是元素的顺序)。