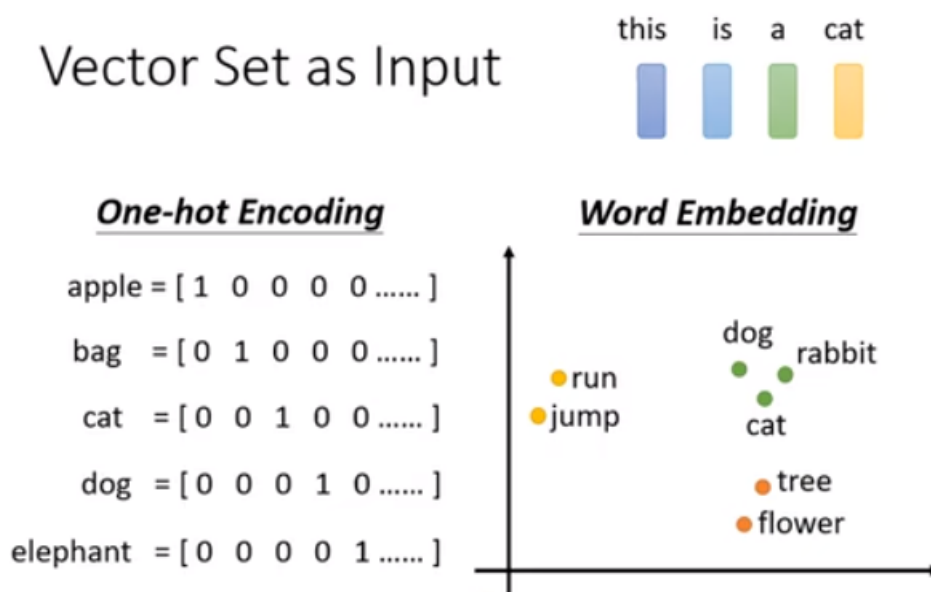


Self attention mechanism

1 文字编码

对文字进行编码，可以使用独热编码，缺点是**无法揭示单词之间的联系**，如cat和dog可能是比较相关的，Word Embedding 就可以很好地解决这一问题，**让不同的单词之间有不同的距离**，类似于分为了不同的类别，Word Embedding的结果可以通过训练得到。



2 序列化模型的输入与输出

一句话、一段语音、一个图片网络都可以看作为一个序列化信息，输入到模型中可以有多种输出：

1. 输入和输出长度相等

例如**词性标注**



2. 只需输出一个结果

例如**情感分析**



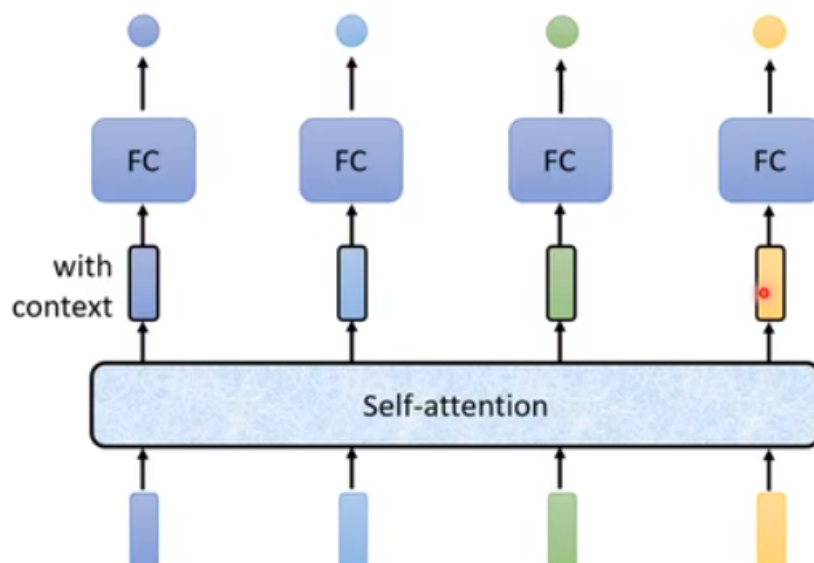
3. 输入和输出不等长

如**翻译**

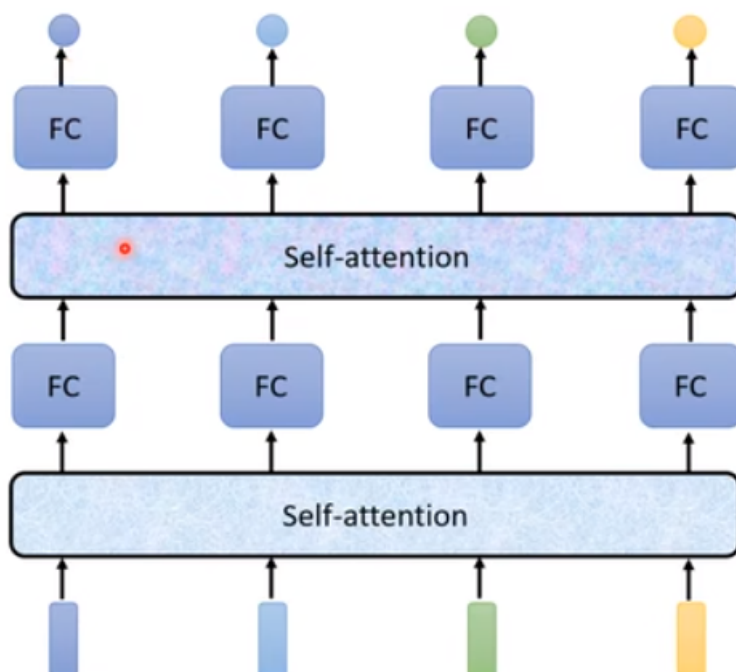


3 引入Self attention mechanism

对于一个序列化模型，我们往往要考虑各个单词之间的前后联系，如果使用的是传统的全连接神经网络，那么就需要针对不同的序列长度进行处理，或者一次性考虑全局信息，但这会极大地增加计算量，因为无法并行。因此便提出了Self attention mechanism，该模型可以**一次性考虑全局的信息**，**同时又可以并行计算**：



同时，全连接网络和自注意力机制可以交替使用：



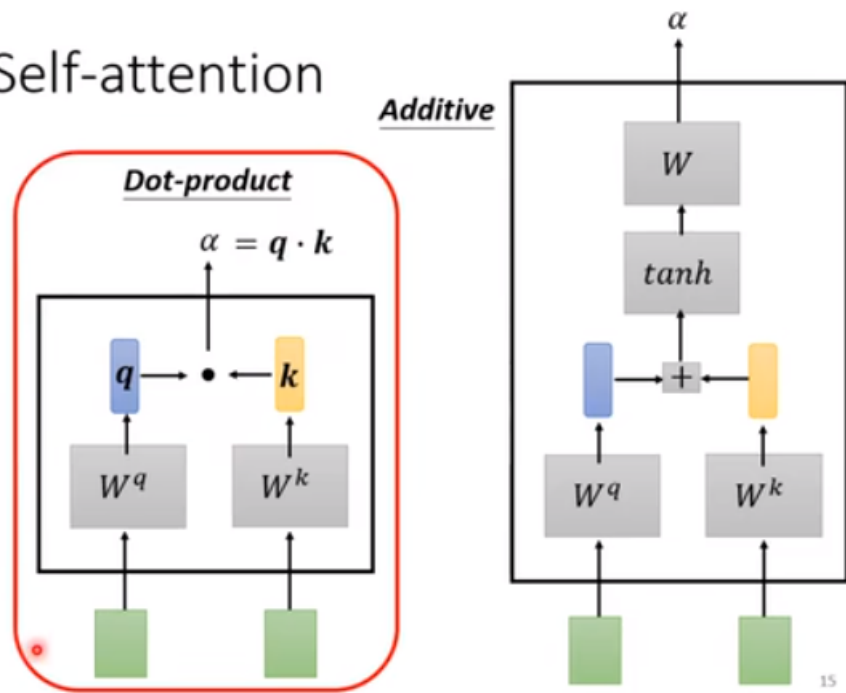
4 Self attention 计算过程

原文公式： $Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$

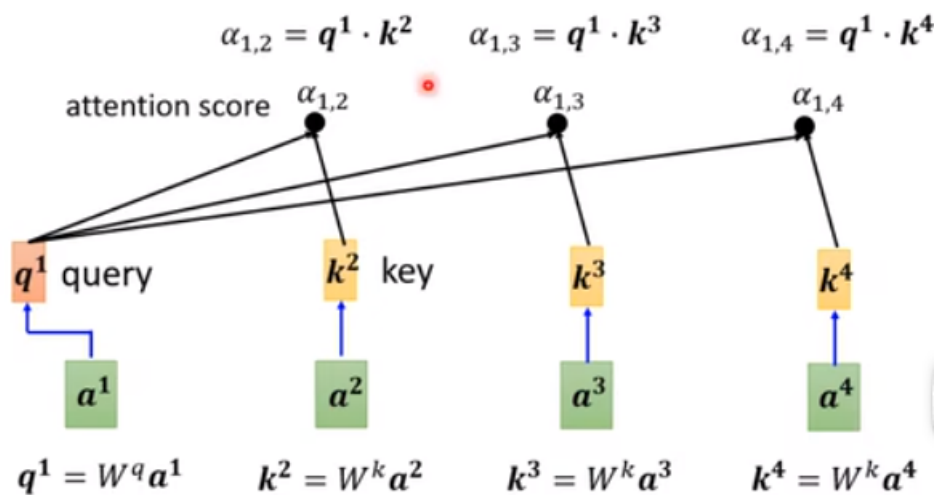
4.1 具体步骤

首先计算一个序列中各个变量之间的关联性 α ，有点**乘注意力**和**加性注意力**两种方式，在transformer中，只使用了Dot-product：

Self-attention



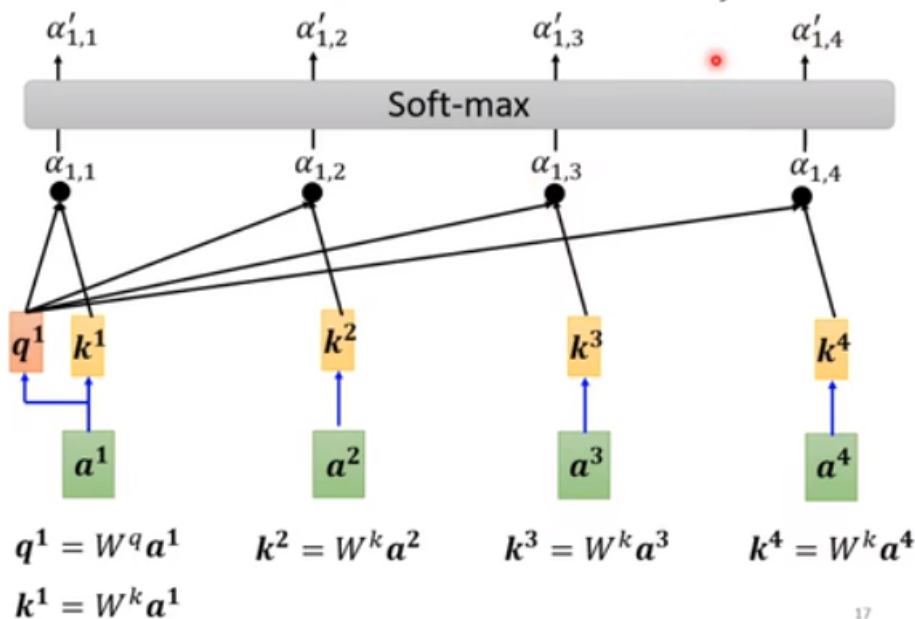
每一个词向量都可以得出 q, k, v ，将 q 与 k 点乘，可以得到二者的关联性（注意：每一个词也要和自己计算关联性）：



关联性计算完成之后，为了使权重之和为1，需要进行softmax：

Self-attention

$$\alpha'_{1,i} = \exp(\alpha_{1,i}) / \sum_j \exp(\alpha_{1,j})$$



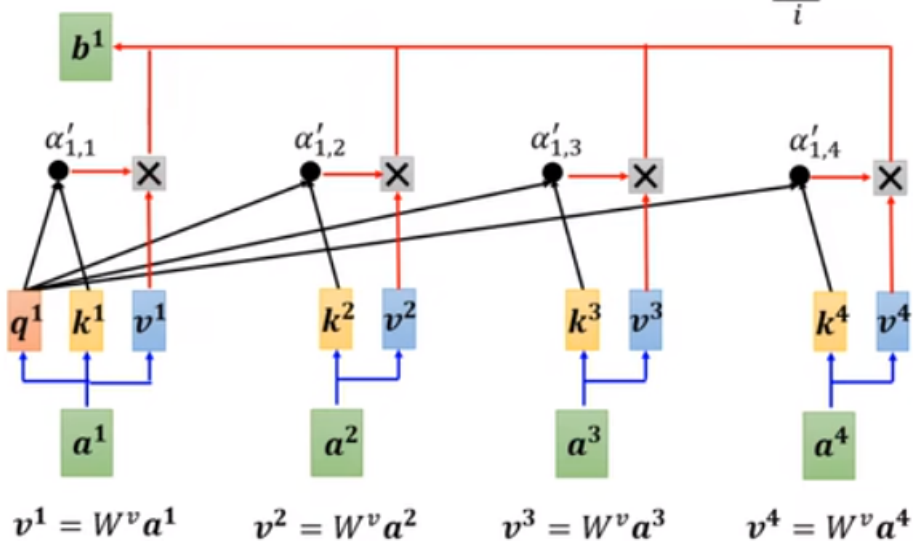
17

得到权重之后与V相乘，得到最终结果：

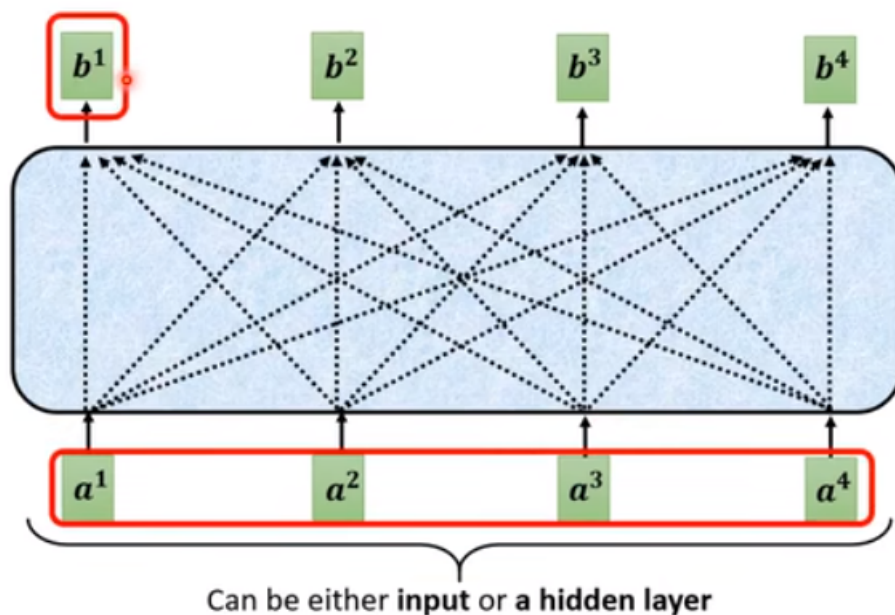
Self-attention

Extract information based on attention scores

$$b^1 = \sum_i \alpha'_{1,i} v^i$$



因此这就得到了某一个词的注意力结果，随后即可计算每一个词的注意力：



4.2 QKV的计算

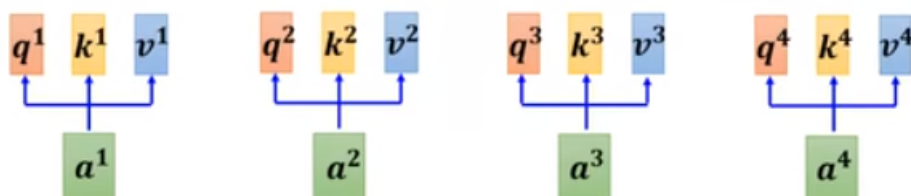
从原始词向量得到QKV:

Self-attention

$$q^i = W^q a^i \quad \begin{matrix} q^1 & q^2 & q^3 & q^4 \\ Q \end{matrix} = \begin{matrix} W^q \\ I \end{matrix} \begin{matrix} a^1 & a^2 & a^3 & a^4 \\ I \end{matrix}$$

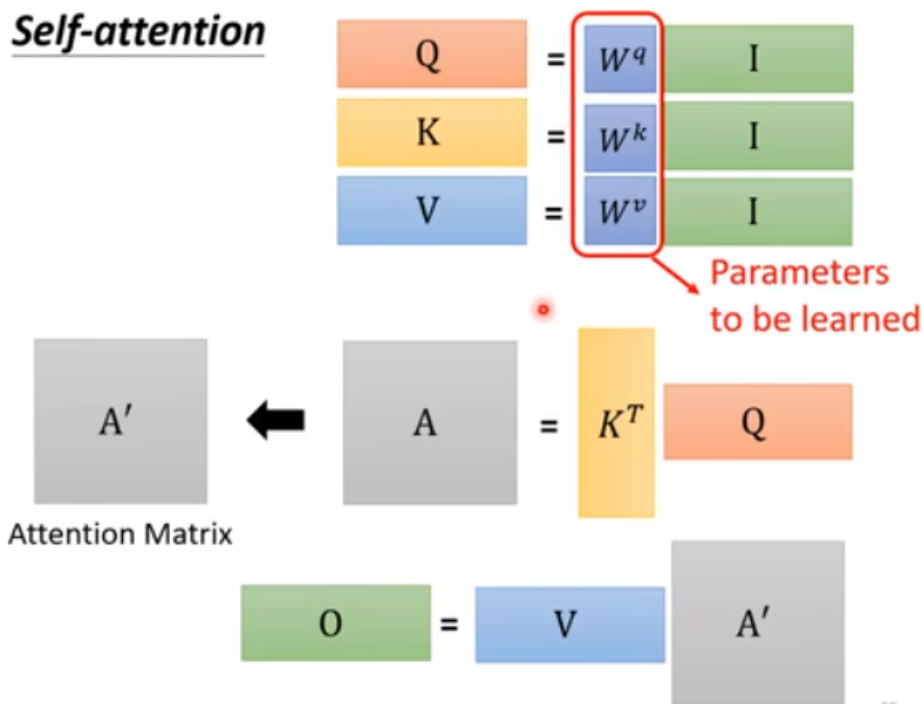
$$k^i = W^k a^i \quad \begin{matrix} k^1 & k^2 & k^3 & k^4 \\ K \end{matrix} = \begin{matrix} W^k \\ I \end{matrix} \begin{matrix} a^1 & a^2 & a^3 & a^4 \\ I \end{matrix}$$

$$v^i = W^v a^i \quad \begin{matrix} v^1 & v^2 & v^3 & v^4 \\ V \end{matrix} = \begin{matrix} W^v \\ I \end{matrix} \begin{matrix} a^1 & a^2 & a^3 & a^4 \\ I \end{matrix}$$



就是使用原始词向量与对应的权重矩阵W相乘。

4.3 Self attention 整体计算流程图

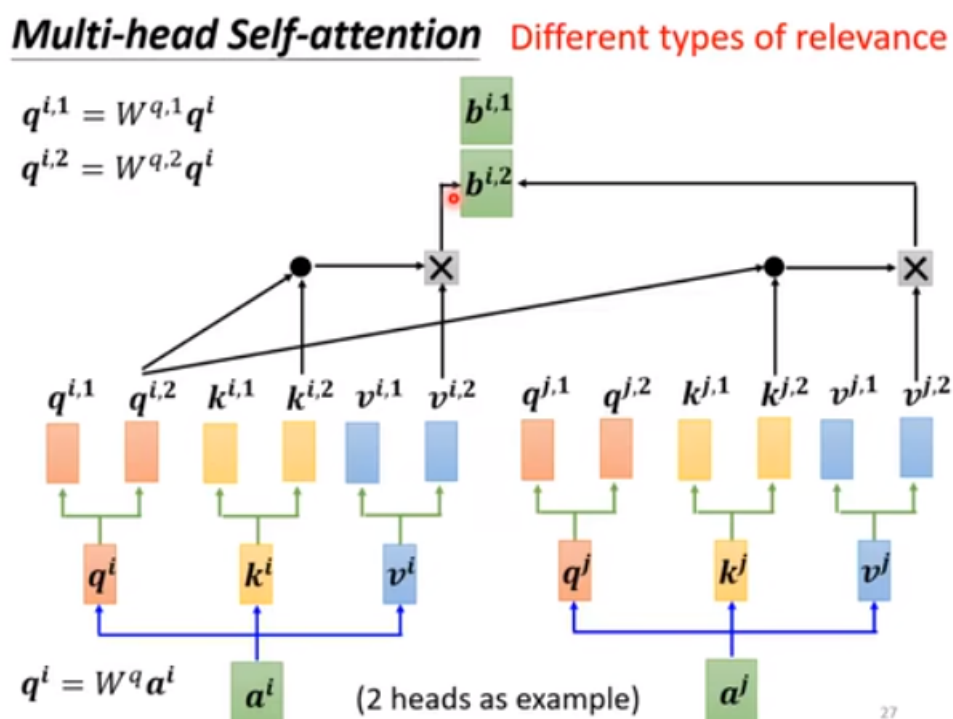


整个Self attention 流程中，我们只需要通过训练数据学习到权重 W ，除此之外，没有其他任何参数。权重 W 的初始值可以随机设置。

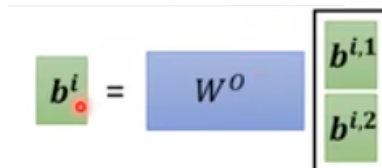
5 Multi-head Self-attention

上述整个Self attention的过程就是使用 q 寻找与之相关的 k ，但是“相关”这件事其实有很多种不同的形式和定义，不同的人可能对两个事物的相关性有完全不同的看法。比如：猫和狗相关，可能是因为猫和狗都可以作为人类的宠物、都有四条腿、身体形状比较形似、都是哺乳动物等。**一个注意力机制可能只能挖掘出一种或少数几种相关**，因此我们就提出了**多头注意力机制 Multi-head Self-attention mechanism**。

以2个head为例，通过原始词向量数据和权重矩阵相乘，得到 q, k, v ，此时的权重矩阵有6个，而不是之前的3个。



通过两个堆叠的注意力层计算得到两个结果，将二者相连接，再乘以权重矩阵，就可以得到最终的结果：


$$b^i = W^O \begin{bmatrix} b^{i,1} \\ b^{i,2} \end{bmatrix}$$

6 Positional Encoding

1. 如何实现位置编码？

- 分配一个0到1之间的数值给每个时间步：不同句子之间的时间步差值没有任何意义。
- 1分配给第一个词，2分配给第二个词，以此类推：模型很有可能没有看到过任何一个这样的长度的样本句子，这会严重影响模型的泛化能力。

2. 一种好的位置编码方案需要满足以下几条要求

- 它能为每个时间步输出一个独一无二的编码；
- 不同长度的句子之间，任何两个时间步之间的距离应该保持一致；
- 模型应该能毫不费力地泛化到更长的句子。它的值应该是有界的；
- 它必须是确定性的。

3. Transformer的位置编码

Transformer的作者通过下列算法计算出位置信息e

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$

实际上第t个位置的位置编码就可以如下表示，特定的词向量维度d和位置t就确定了位置编码的维度和数值大小：

$$w_i = \frac{1}{10000^{2i/d}}$$

$$\vec{p}_t = \begin{bmatrix} \sin(\omega_1 \cdot t) \\ \cos(\omega_1 \cdot t) \\ \sin(\omega_2 \cdot t) \\ \cos(\omega_2 \cdot t) \\ \vdots \\ \sin(\omega_{d/2} \cdot t) \\ \cos(\omega_{d/2} \cdot t) \end{bmatrix}$$

将e与原始词向量相加，这样就添加了位置信息：

