# Transformer 之注意力计算原理

**鱼先生**
公众号【Afunby的 AI Lab】

已关注

41 人赞同了该文章

***写在前边：***

学习 Transformer 的过程中，找到了Ketan Doshi 博客中关于Transformer系统的介绍文章，感觉非常棒，于是进行了翻译。**该系列一共有4篇文章。本篇文章为该系列的最后一篇。**

原文链接在文末。翻译主要采用 “DeepL+人工”的方式进行，并加入了一些自己的理解。

第一篇：Transformer 之整体介绍

第二篇：Transformer 之逐层介绍

第三篇：Transformer 之多头注意力

---

**Transformers Explained Visually — Not Just How, but Why They Work So Well**

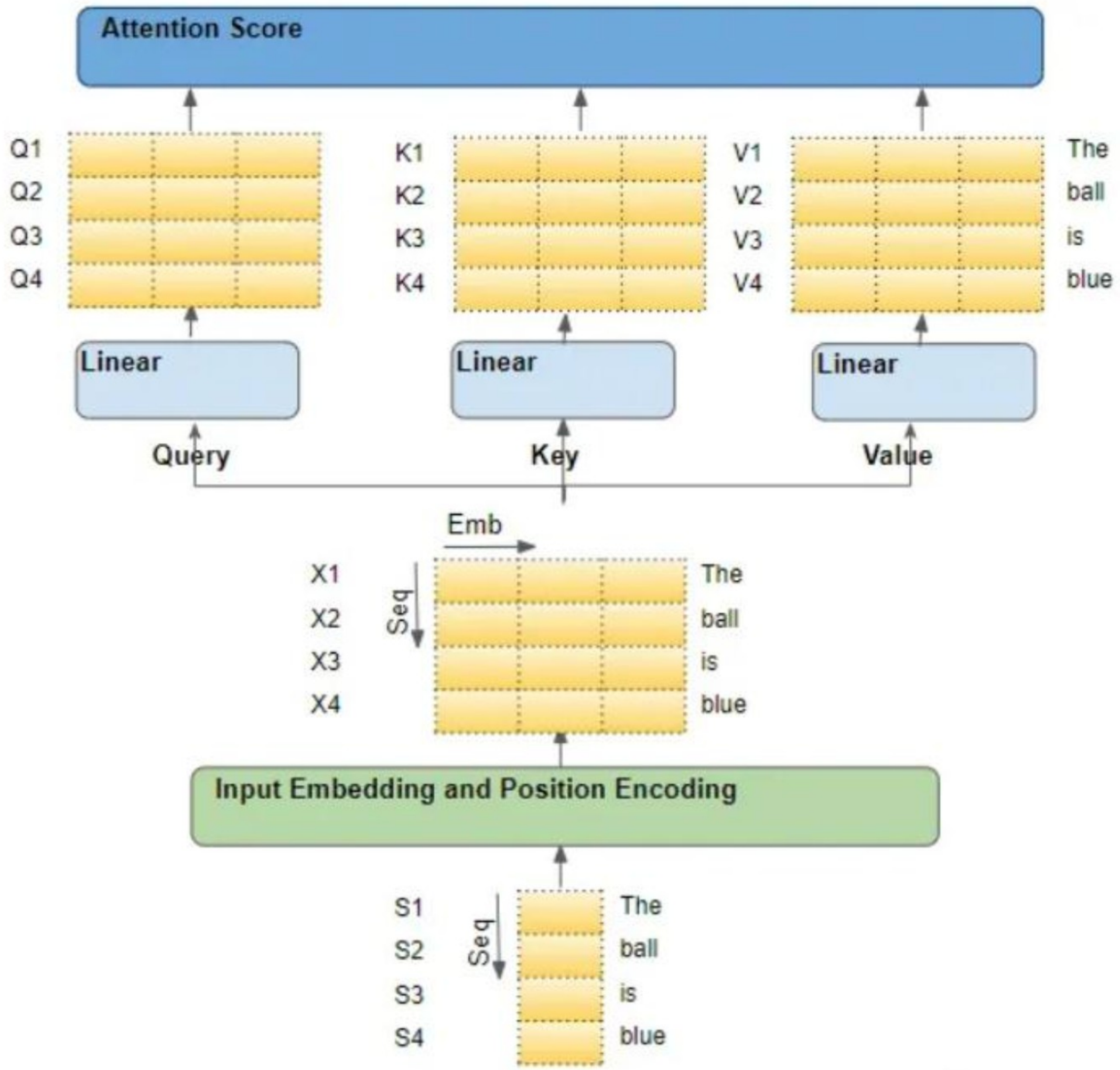
---

## 一、输入序列怎样传入注意力模块？(How does the input sequence reach the Attention module)

假设我们正在处理一个英语到西班牙语的翻译问题，其中一个源序列的样本是 "The ball is blue"。目标序列是 "La bola es azul"。

源序列首先通过嵌入和位置编码层，该层为序列中的每个单词生成嵌入向量。嵌入被传递到编码器，它首先到达自注意力模块。在自注意力模块中，嵌入的序列通过三个线性层，产生三个独立的矩阵--Q、K、V。这三个矩阵被用来计算注意力得分。

需要记住的是，这些矩阵的每一 "行 "对应于源序列中的一个词。



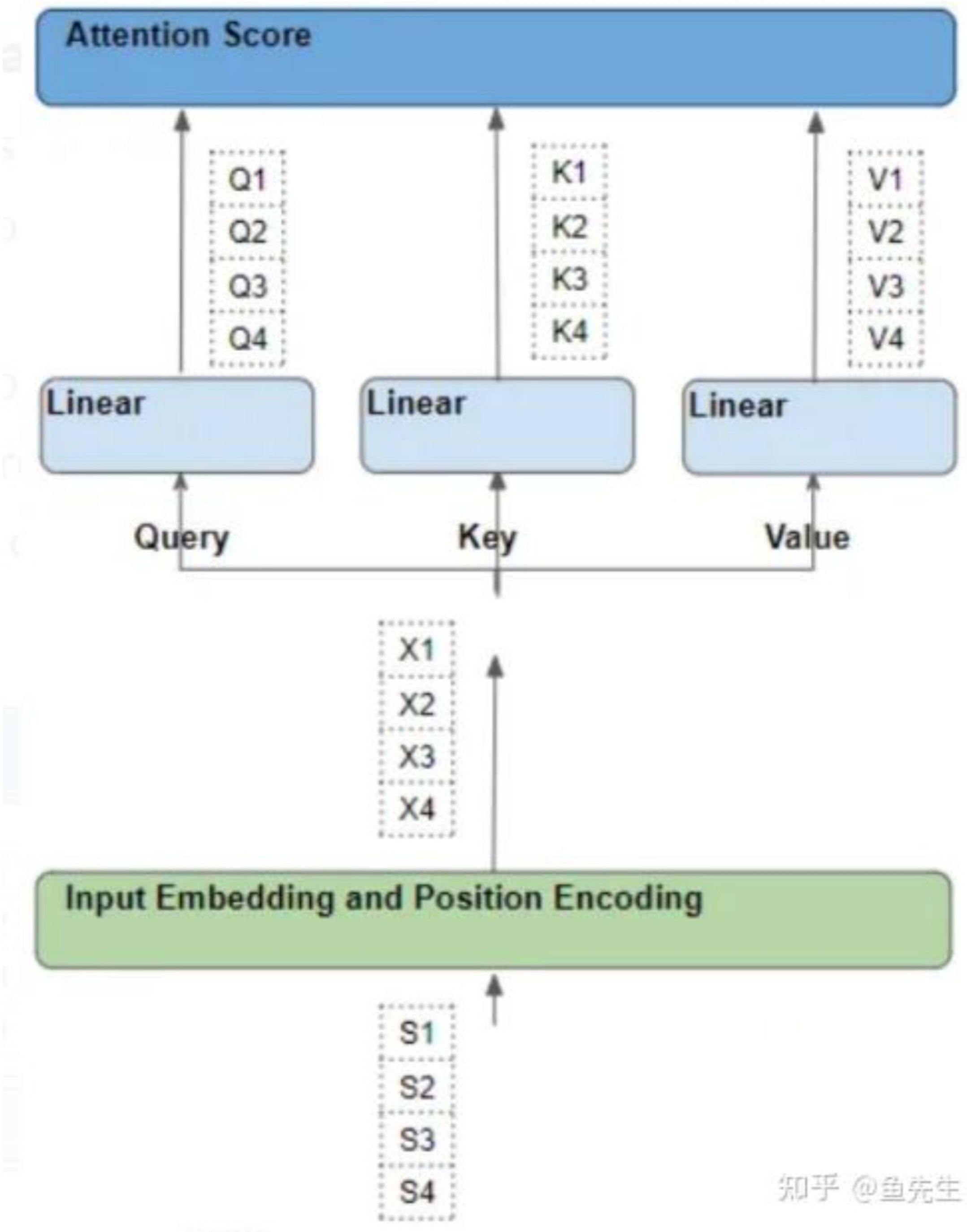

The flow of the source sequence (Image by Author)



**二、进入注意力模块的矩阵的每一行，都是原序列中的一个词(Each input row is a word from the sequence)**

一个理解自注意力的方法是，从源序列中的单个词开始，然后跟踪它们在 Transformer 中的路径。我们要关注注意力模块内部的情况，这将帮助我们清楚地看到源序列和目标序列中的每个词是如何与源序列和目标序列中的其他词互动的。

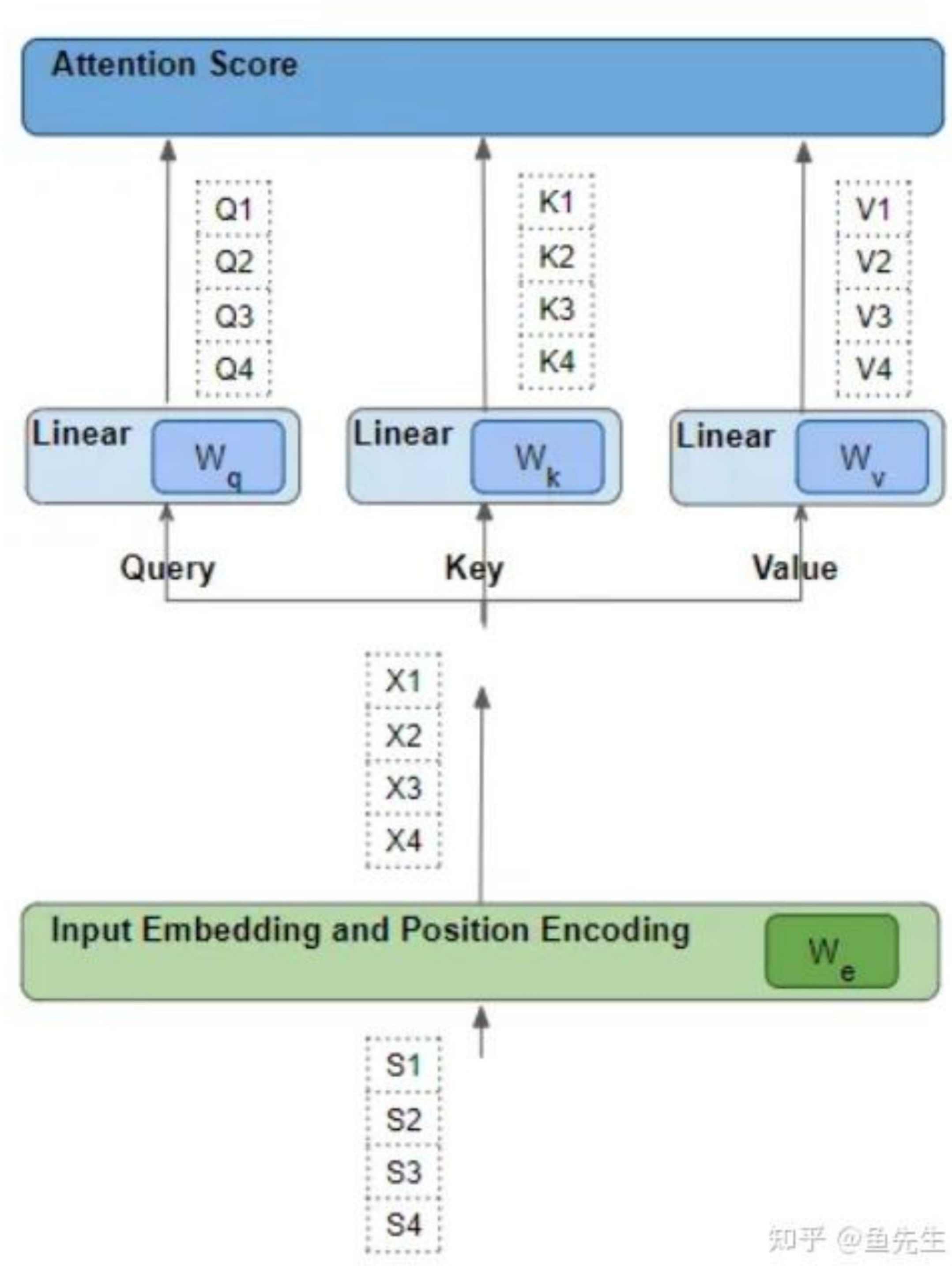
因此，需要特别关注注意力模块对每个词进行的操作，以及每个向量如何映射到原始输入词。我们不需要担心其他细节，如矩阵形状、计算具体细节、多个注意头等等，因为它们与每个词的去向没有直接关系。为了简化解释和可视化，让我们忽略嵌入维度，只跟踪每个词的**"行"**。




### 三、每一行，都会经过一系列可学习的变换操作(Each word goes through a series of learnable transformations)

每个这样的"**行**"都是通过一系列的诸如嵌入、位置编码和线性变换等转换，从其相应的源词中产生。而所有这些的转换都是可训练的操作。这意味着在这些操作中，使用的权重不是预先确定的，

而是利用模型通过输出进行学习。



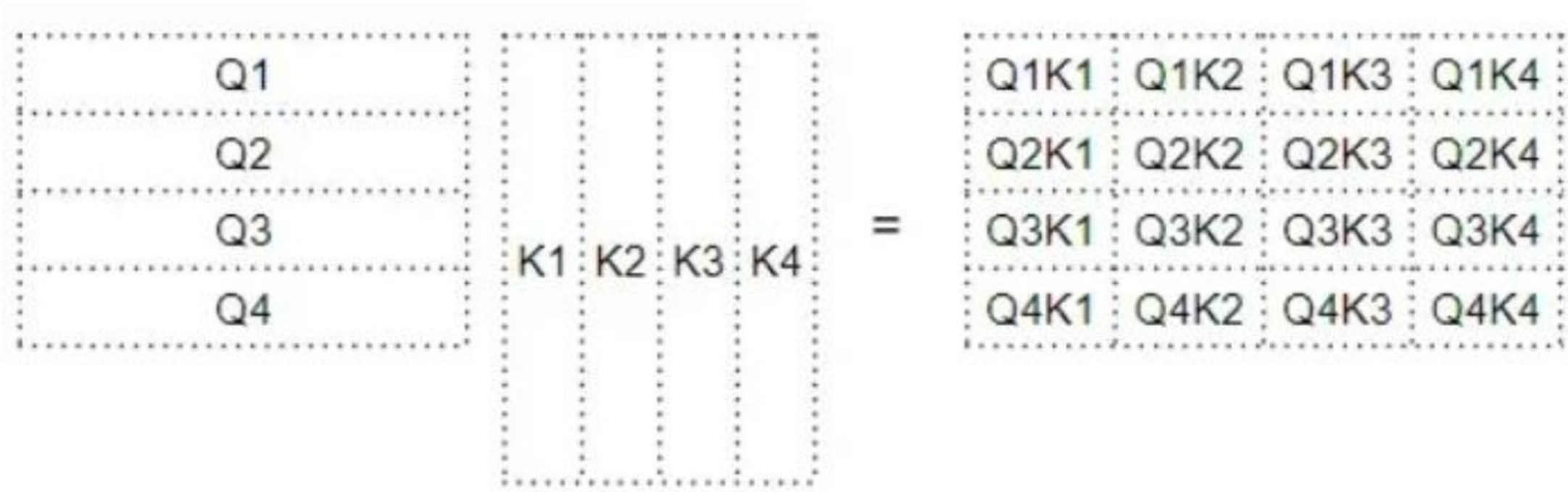
关键问题是，Transformer 如何确定哪一组权重会给它带来最佳效果？请记住这一点，稍后会回到这个问题上。

## 四、注意力得分（Attention Score — Dot Product between Query-Key and Value words）

### 1.the first step — 'factor' matrix

注意力计算的第一步是在 Q 矩阵和 K矩阵的转置之间做矩阵乘法（即点积）。看看每个词会发生什么：

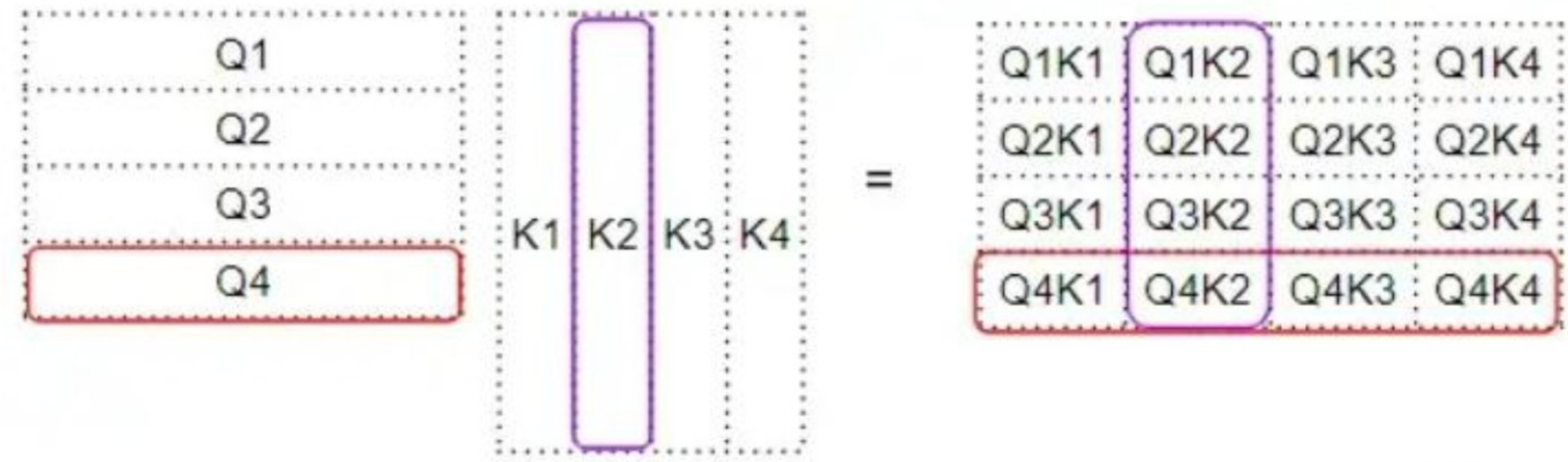
Q 与 T 的转置进行点积，产生一个中间矩阵，即所谓“因子矩阵”。每个单元都是两个词向量之间的矩阵乘法。




Dot Product between Query and Key matrices (Image by Author)



如下所示，第四行的每一列都对应于第四个 **Q向量** 与每个 **K向量** 之间的点积：

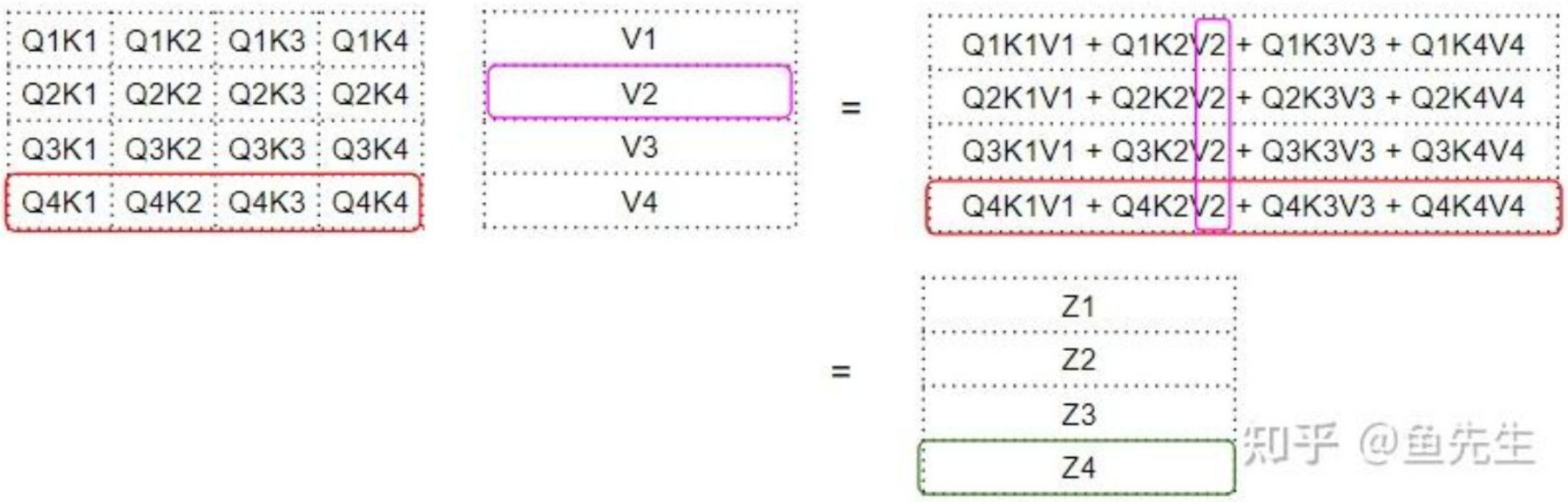



Dot Product between Query and Key matrices (Image by Author)

## 2. produce the attention score

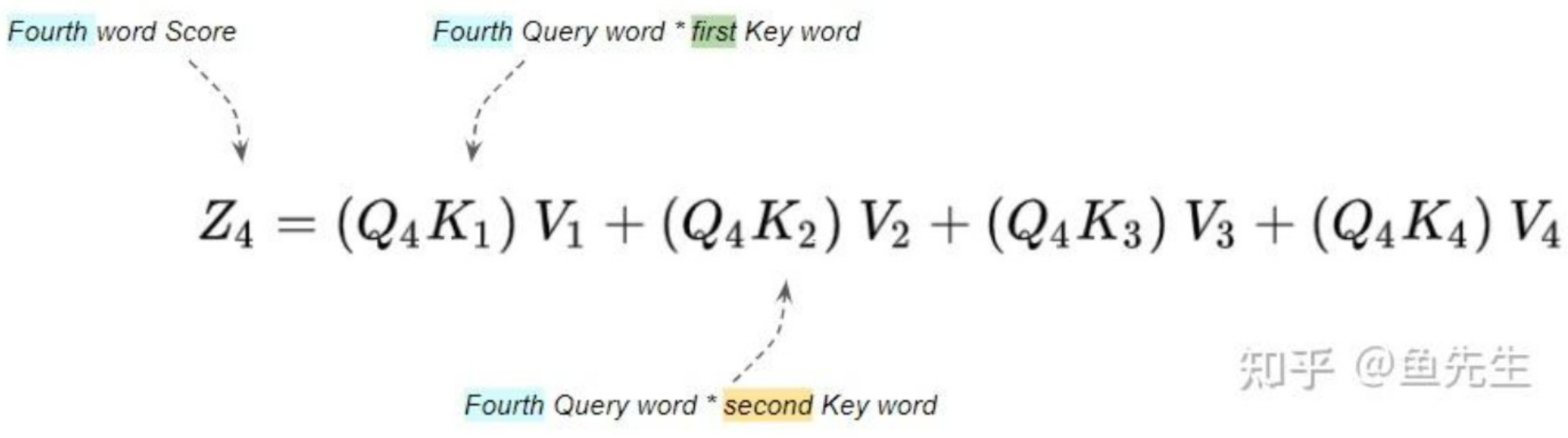
*注意：此部分内容与其他资料中相应内容有出入，未含 Softmax 计算。同时注意力分数的说法也不一样。*

下一步是在这个中间 “因子矩阵”和 V 矩阵之间进行矩阵相乘，以产生由注意力模块输出的注意力分数（**attention score**）。下图中，我们可以看到第四行对应的是第四个 Q 矩阵与所有其他 K 和 V相乘：




这就产生了由注意力模块输出的注意力分数向量（Z）。

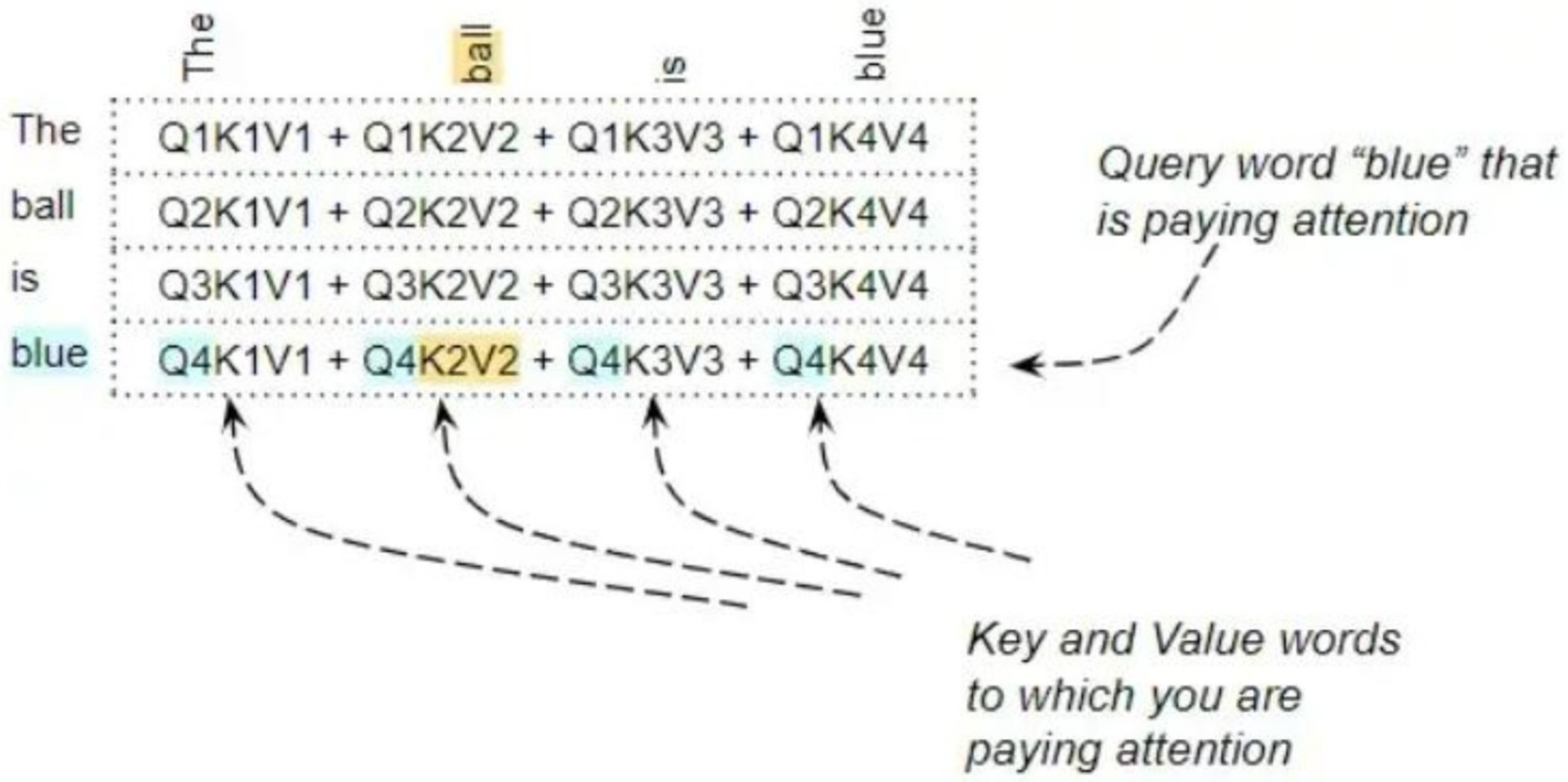
对于 Z 向量的理解是：每个词，**其 V 矩阵中每个词的编码值，由 factor 矩阵加权。factor 矩阵是该特定单词的 Q值 与所有单词的 K值 的点积。**




可否这样理解：若两个向量相似，如对于图示中的 Q4K2，则二者的因子值就大。再乘以V2时，得分就高。相当于把相似相进行了二次验证和放大。

## 五、查询、键、值的作用（What is the role of the Query, Key, and Value words?）

对某一个查询向量 Query，可以理解为正在计算注意力分数的词。而 Key 向量和 Value 向量是我们正在关注的词，即该词与查询词的相关程度。




Attention Score for the word "blue" pays attention to every other word (Image by Author)

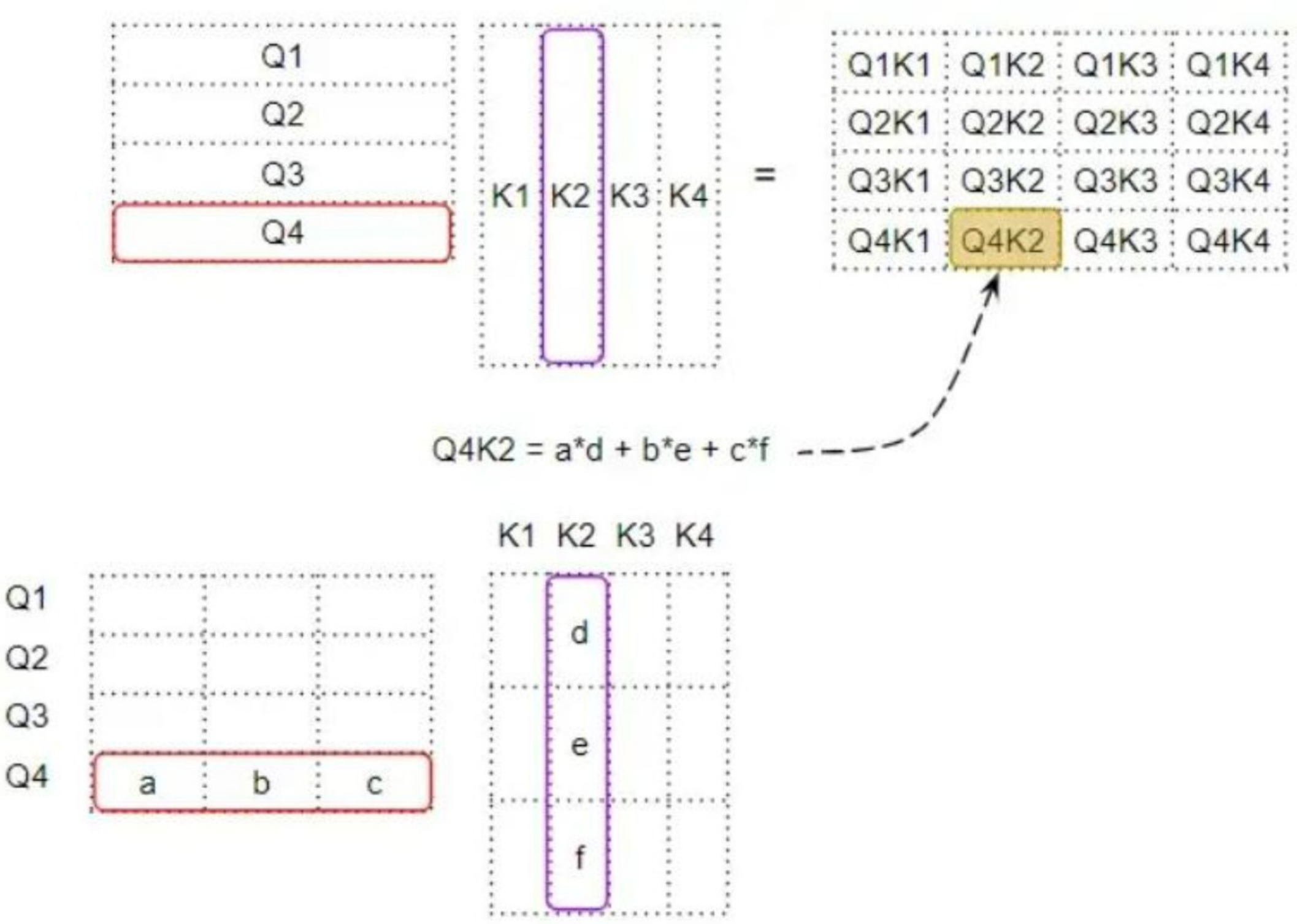
例如，对于 "The ball is blue "这个句子，单词 "blue "这一行包含 "blue "与其他每个单词的注意力分数。在这里，"blue "是 Query word，其他的词是 "Key/Value"。

注意力的计算还包含其他操作，如除法和 Softmax 计算，但本文可以忽略它们。它们只是改变了矩阵中的数值，但并不影响矩阵中每个词行的位置。它们也不涉及任何词间的相互作用。

## 六、点积：衡量向量之间的相似度（Dot Product tells us the similarity between words）

Attention Score 是通过做点乘，然后把它们加起来，来捕捉某个特定的词，和句子中其他每个词之间的一些互动。但是，矩阵乘法是如何帮助 Transformer 确定两个词之间的相关性呢?

为了理解这一点，请记住，Query, Key, Value 行实际上是具有嵌入维度的向量。让我们放大看看这些向量之间的矩阵乘法是如何计算的：




Each cell is a dot product between two word vectors (Image by Author)



当我们在两个向量之间做点积时，我们将一对数字相乘，然后相加：

- 如果这两个成对的数字（如上面的'a'和'd'）都是正数或都是负数，那么积就会是正数。乘积会增加最后的总和。
- 如果一个数字是正数，另一个是负数，那么乘积将是负数。乘积将减少最后的总和。
- 如果乘积是正数，两个数字越大，它们对最后的总和贡献越大。

这意味着，如果两个向量中相应数字的符号是一致的，那么最终的和就会更大。

## 七、Transformer 如何学习单词之间的相关性（How does the Transformer learn the relevance between words?）

前述点积的概念也适用于注意力得分的计算。如果两个词的向量更加一致，注意力分数就会更高。我们希望 Transformer 的操作是，对于句子中的两个词，若相互关联，则二者的注意分数就高。而对于两个互不相关的词，我们则希望其得分较低。

例如，对于 "The black cat drank the milk" 这个句子，"milk "这个词与 "drank "非常相关，与 "cat "的相关性可能稍差，而与 "black "无关。我们希望 "milk "和 "drank "产生一个高的注意力分

数，"milk "和 "cat "产生一个稍低的分数，而 "milk"和 "black "则产生一个可以忽略的分数。这就是我们希望模型学习产生的输出。

要做到这一点，"milk "和 "drank "的词向量必须是一致的。"milk" 和 "cat" 的向量会有一些分歧。而对于 "milk "和 "black "来说，它们会有很大的不同。

让我们回到前述的问题—Transformer 是如何找出哪一组权重会给它带来最佳结果的?

词向量是根据词嵌入和线性层的权重生成的。因此，Transformer 可以学习这些嵌入向量、线性层权重等来产生上述要求的词向量。

**换句话说，它将以这样的方式学习这些嵌入和权重：如果一个句子中的两个词是相互关联的，那么它们的词向量将是一致的。从而产生一个更高的关注分数；对于那些彼此不相关的词，词向量将不会被对齐，并会产生较低的关注分数。**

因此，"milk "和 "drank "的嵌入将非常一致，并产生较高的注意分数。对于 "milk "和 "cat"，它们会有一些分歧，产生一个稍低的分数，而对于 "milk "和 "black"，它们会有很大的不同，产生一个非常低的分数。

这就是注意力模块的原理。

## 八、总结（Summarizing — What makes the Transformer tick?）

Query 和 Key 之间的点积计算出每对词之间的相关性。然后，这种相关性被用作一个 "因子 "来计算所有 Value 向量的加权和。该加权和的输出为注意力分数。

Transformer 学习嵌入等，其方式是使彼此相关的词更加一致。

这是引入三个线性层的原因之一，这给了注意力模块一些更多的参数，它能够学习调整词向量的创建。