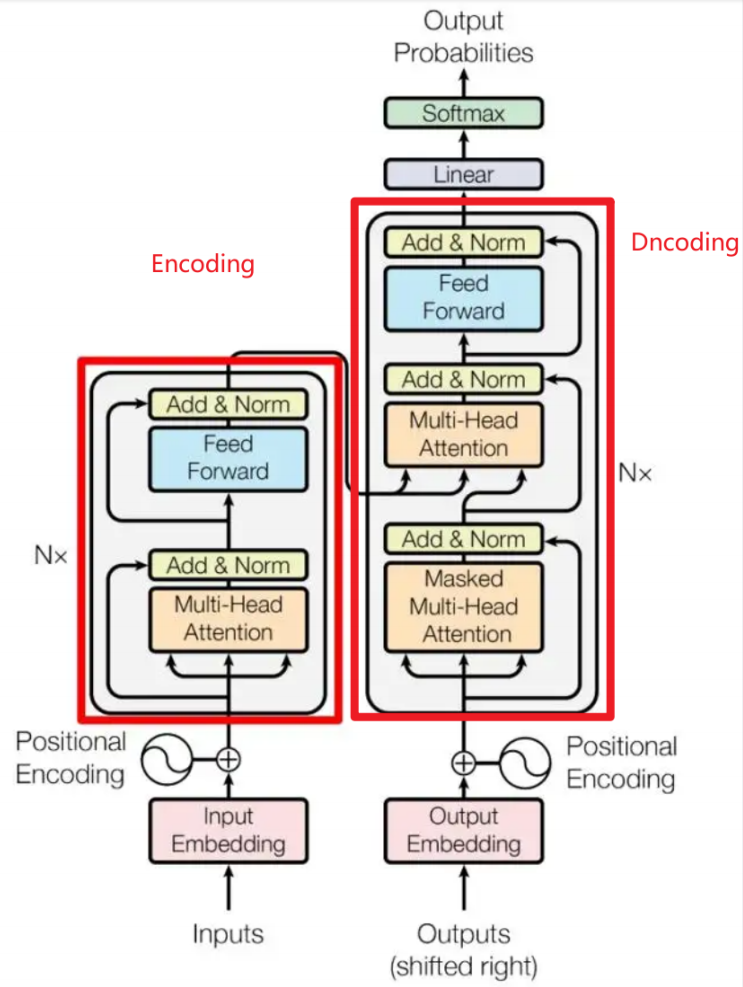
# Transformer详解

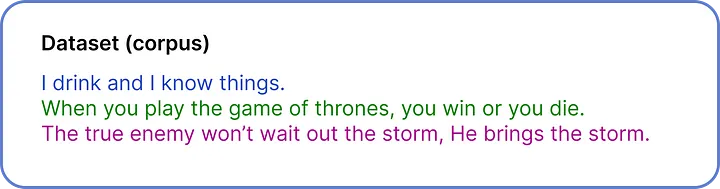
## 一：原理讲解

### 架构图

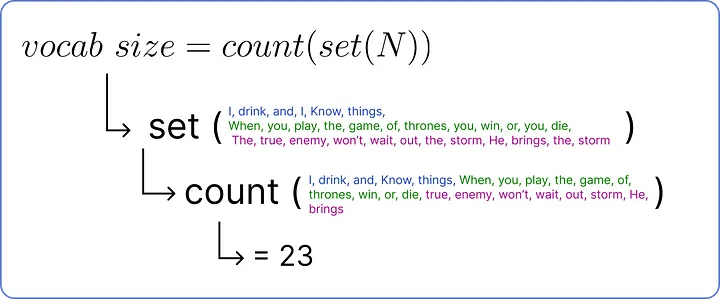


添加图片注释，不超过 140 字（可选）

### 1. Input Embedding



仅包含三个句子



N 表示总数，最后的结果是去重之后的词汇量

#### 1.1 Tokenization

* Token 是指文本中的一个基本单元，通常是词或短语。这个切分 token 的过程，称为分词（Tokenization）。 Tokenization 的本质其实就是一个字符到数字的映射，其维护的是一个字典，而不是权重，也就是说每一个字符/词/短语都有一个唯一确定的数字与其对应。

* 词粒度

* 词粒度基本是最直观的分词手段了，也是最符合我们平时认知的方式。每一个 token 是词典中的一个词。 - 优点： - 符合人类的直觉，词的含义完备。 - 对于拉丁语系很简单，按照空格和标点符号分割即可。 - 缺点： - 对于中文日文，需要专门进行分词，比如中文分词一般都用 jieba（结巴分词），哈工大的 LTP 等。 - 如何构造一份好的词典，以及词典过大的问题。 - OOV （Out of Vocabulary）问题，对于词典中没有词一般会分配一个统一来处理，信息丢失。 - 同义词和误拼会被认为是不同的词。比如 do not 和 don't 虽然含义一样，但是最终的 id 不一样。类似的还有词的不同形态，比如 have, has, had, having 等。 - 词典中的词没有任何关联，低频词不会得到充分的训练。比如 oarfish，这个词虽然第一眼可能不知道什么意思，但是能猜出是一种鱼（皇带鱼）。

* 子粒度

* 把文本按照最小粒度来拆分，对于英文来说就是 26 个字母加上一些标点符号。中文来说就是字。 - 优点： - 实现简单。 - 词典很小。 - 几乎没有 OOV （Out of Vocabulary）问题 - 缺点： - 对于英文来说，单个字母几乎没有含义。 - 难以学习到词汇的真正含义。由于中文的一个字还是有一定的含义，所以在中文勉强可以使用，但是英文几乎没法使用。 - 会让输入变得很长，训练和推理更慢。

* Embedding

* 核心是为了向量化

* 假设我们现在有一个词库(词汇表)，里面有五个词[ I , love , programming , in, Python ]，我们通过索引下标将其标识出来分别对应[1, 2, 3, 4 , 5],但是这样的表示方法，基本很难帮助我们去发现他们之间的关系，比如相似性、多义性，所以我们引入向量空间，帮助我们更好表示不同词token，one-hot就是其中一种：



添加图片注释，不超过 140 字（可选）

* 对于单个词token，比如：I，[ 1, 0, 0, 0, 0] 是个1\* n的矩阵， n的值与词汇表中的词token个数相等。

* 可以看到，**当m个token构成的m\*n矩阵，起码有如下问题**：

* 1、随着词汇量增多，矩阵的维度（n的个数），占用空间将会特别大；

* 2、矩阵过于稀疏，元素含0太多，无法有效进行矩阵计算（两个矩阵一算内积全是0），比如计算相似性（余弦相似性）；

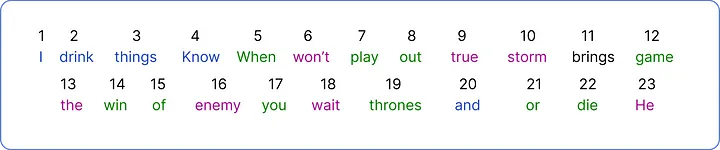
* 类发明了词嵌入embedding解决以上问题。



添加图片注释，不超过 140 字（可选）

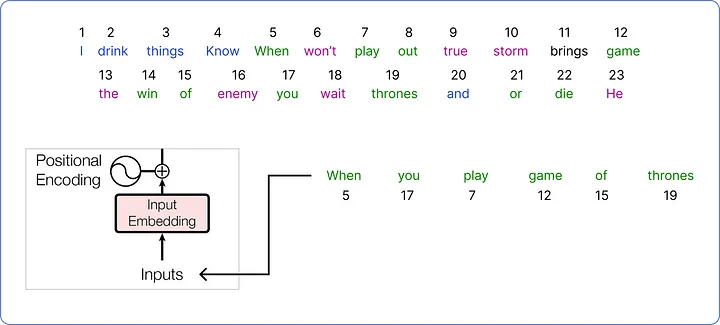
因此，在我们使用使用了字典之后，会进行向量化。使用一组参数，计算出新的向量，用于后面的计算。

Encoding 编码

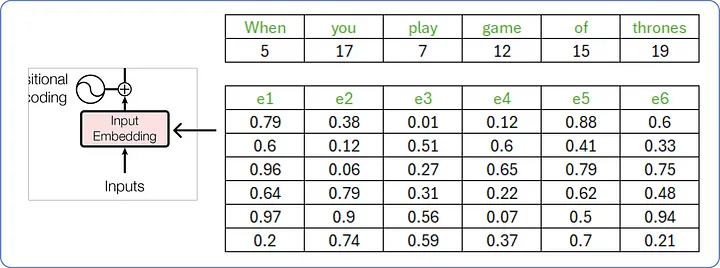


为每个单独的单词分配一个唯一的数字。实际上，大模型并不是一个 Token 对应一个单词，比如 ChatGPT 中，1 个Token = 0.75 个单词。

计算 Embeding

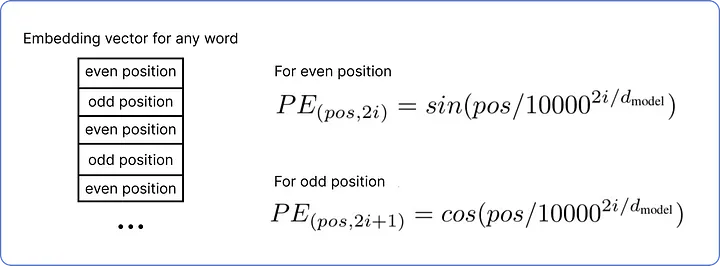


选择一个句子作为输入

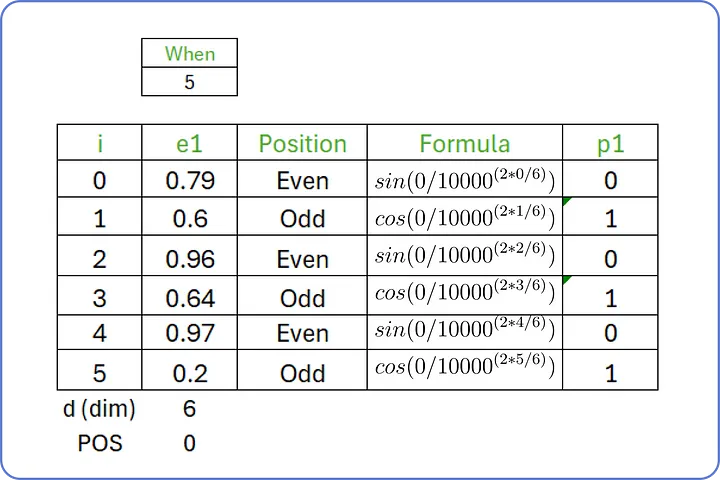


原始论文使用 512 维向量，这里使用嵌入向量的维度为 6，上面是原始数据，下面是解析之后的数据。

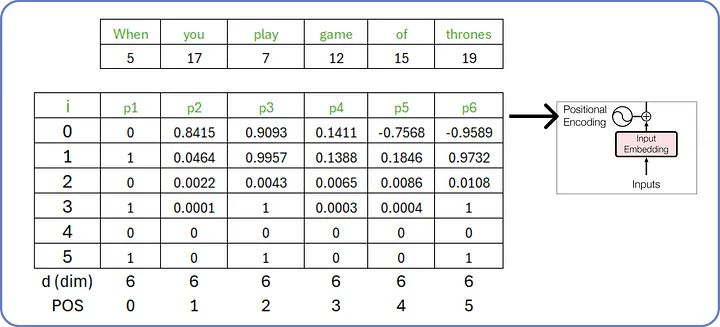
#### 1.2 位置编码



偶数位置用正弦函数，奇数位置用余弦函数。为什么这样设置？因为这样不仅能得到绝对位置，还能得到相对位置信息。相近的更加联系紧密一些。

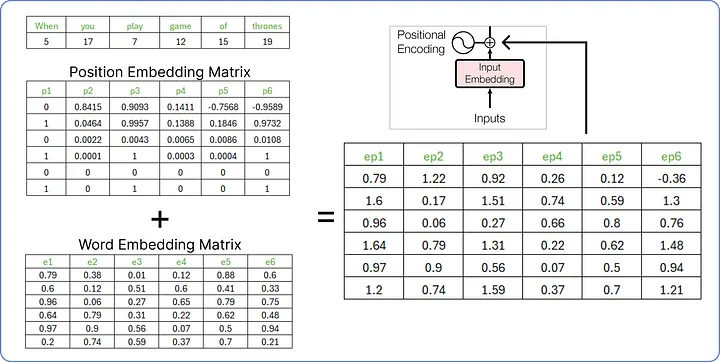


我们的输入句子是“when you play the game of thrones”，起始词是“when” ，起始索引（$pos$）值为 0，维度（$d\_{model}$）为 6。



所有输入单词的位置编码

#### 1.3 位置编码与 embeding 相加



问题： 为什么是相加，而不是concat？

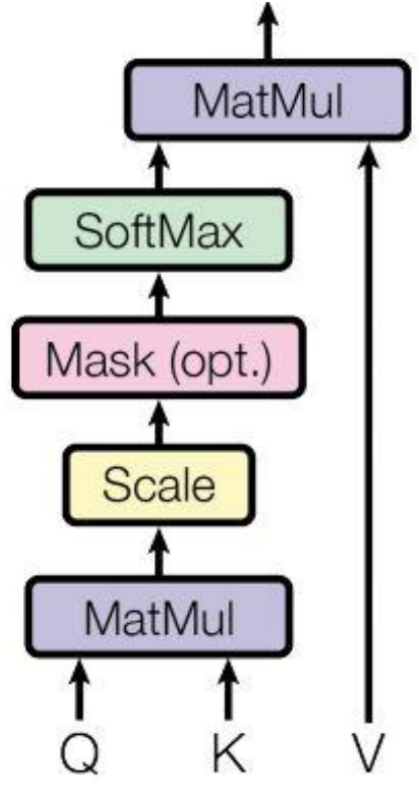
1. concat会增加数据的维度，从而增加计算量。相加则不会。

1. concat 将内容和位置完全分开处理，需要更复杂的机制来融合两种信息，增加了模型学习的难度。

### 2. Encoding

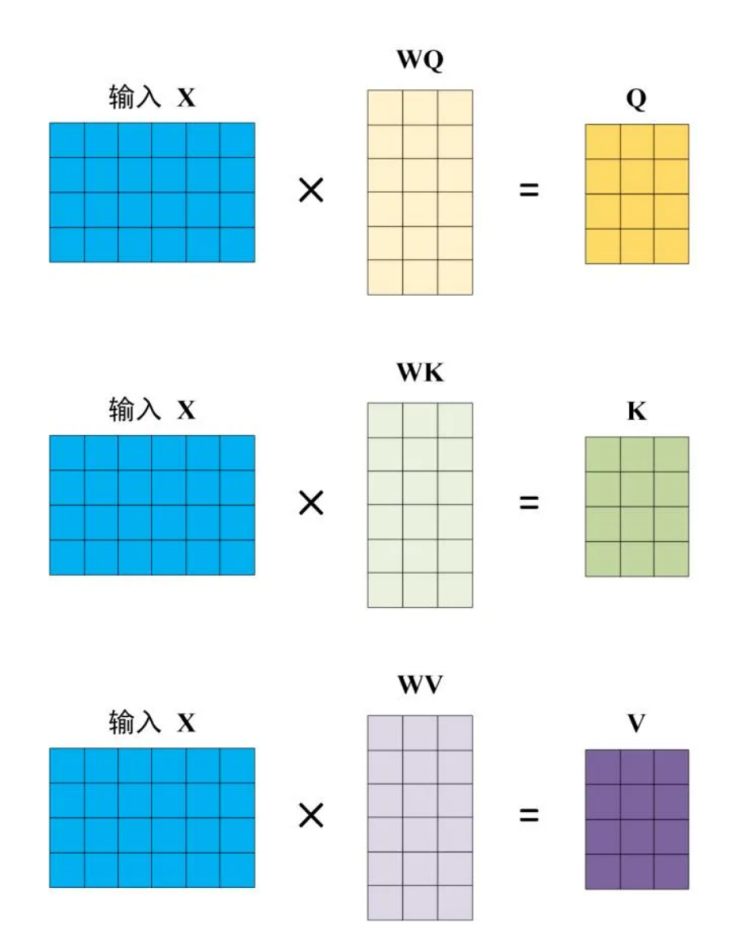
#### 2.1多头注意力机制

##### 2.1.1. 单头注意力机制



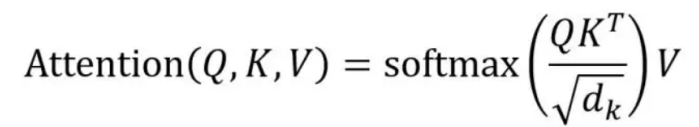
• Q,K,V

可以使用线性变阵矩阵WQ,WK,WV 计算得到 Q,K,V。WQ,WK,WV ，这三个参数在一开始随机或者模型自带的，后续会进行更新。



Q可以理解为自己所带的信息，K可以理解为其他字或者词所带的信息。Q\*K的意思是判断俩者之间关系的紧密程度。V可以理解为权重，俩者关系的一个权重。最后整体算出一个值。

• 计算

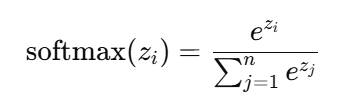


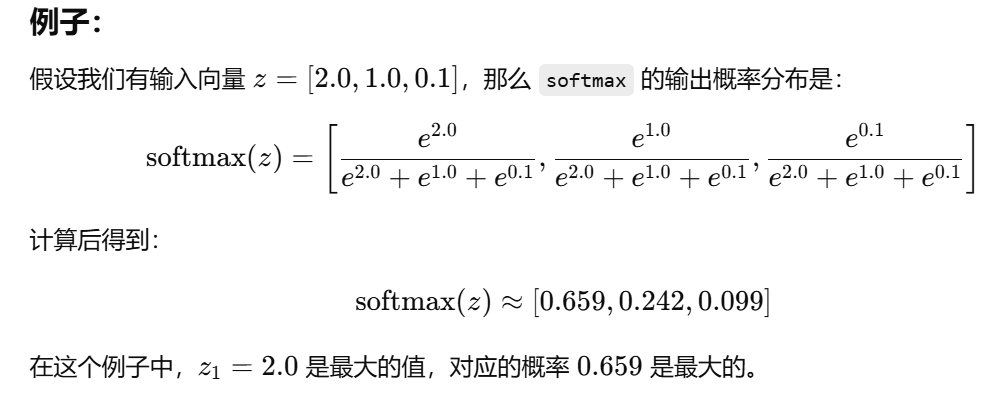
备注：1. 为什么不直接QKV相乘？

**数值不稳定性**：查询和键的维度可能很大，直接相乘会导致得分非常大，导致梯度不稳定或者数值溢出。因此采用缩放的操作

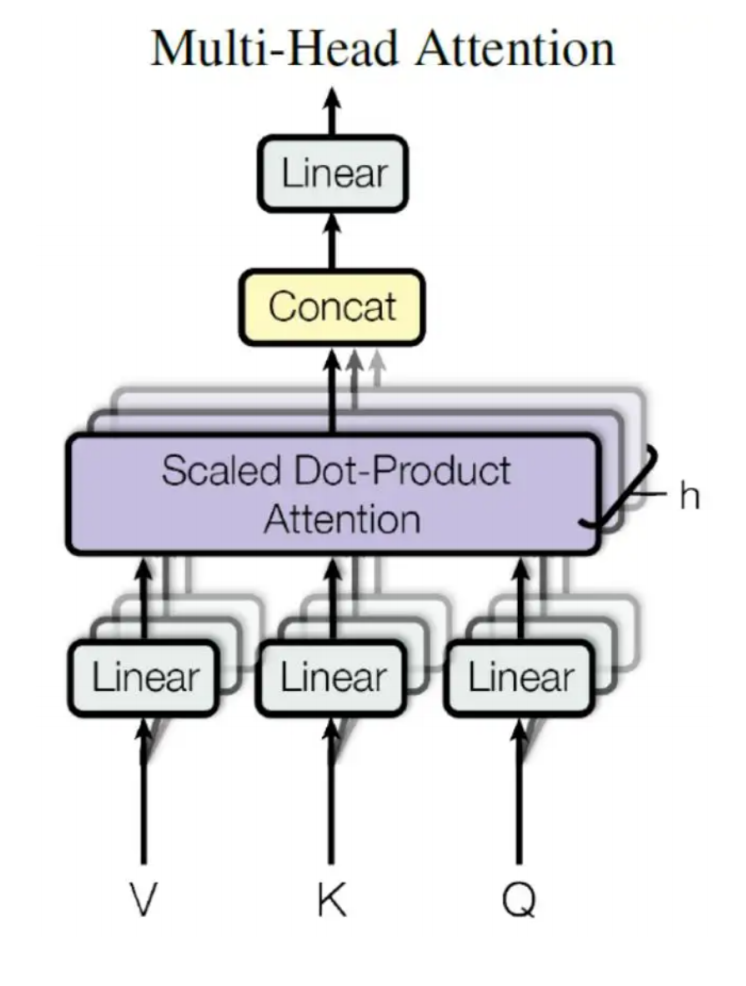
2. 除以平方根，是为了防止内积过大，将结果放在同一个维度进行比较。

3 使用softmax的目的是为了将特征凸显，将较大的值，所占特征更大，从而能够更好的学习到一些信息。

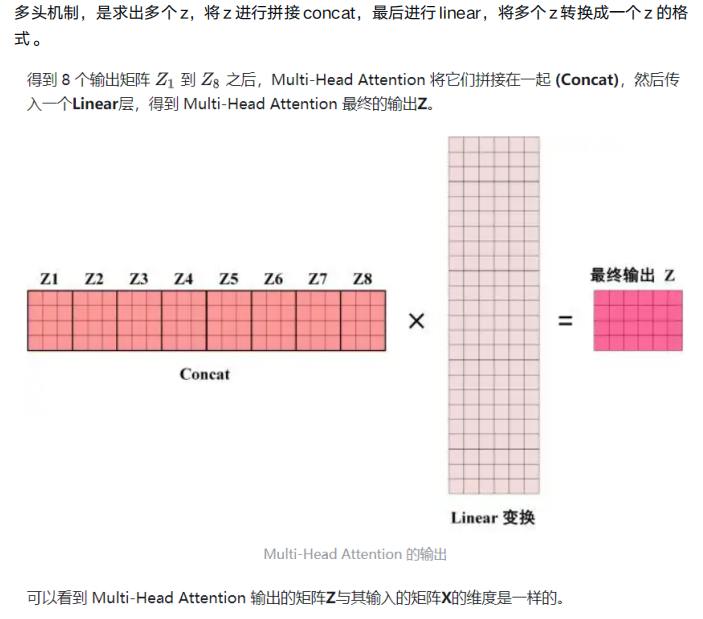




##### 2.1.2. 多头注意力机制



* 理论上来讲，随机事件越多，越能逼近真相。



#### 2.2 Add&Norm

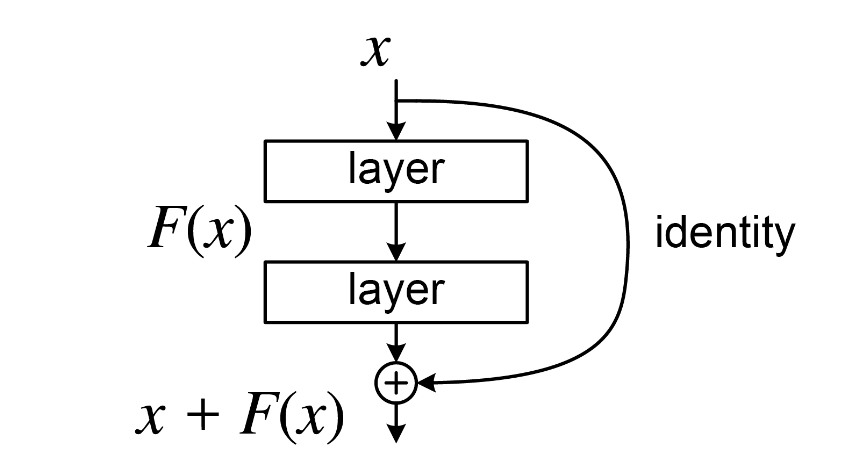
##### 2.2.1 Add（残差连接）

1. 目的

* 1. 为了解决梯度消失的问题

1. 怎么做

* 1. 在tansformer中，将每一层中，将每一层输入值和输出值相加。



b. 刚在这块就是将最原始的输入值，和多头训练完成的值进行相加。

##### 2.2.2 Norm（层级归一化）

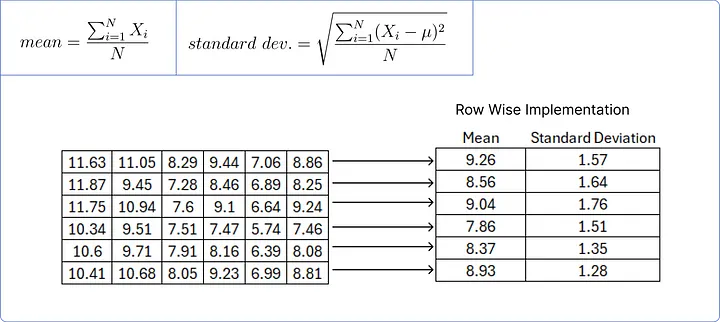
1. 目的

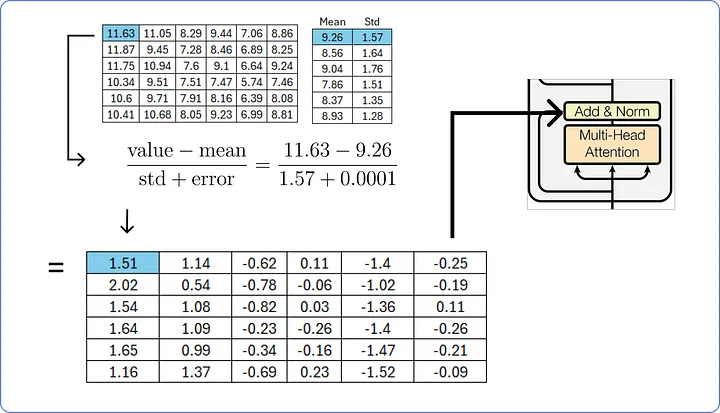
* 1. 为了解决梯度爆炸的问题（数据过大，在梯度更新的时候，参数变动幅度过大，不稳定）

1. 怎么做

* 1. 为了规范化上述矩阵，需要计算每行的均值和标准差。

* 1. 用矩阵中每个值减去对应行的平均值，然后除以对应的标准差。添加一个小的误差值可以防止分母为零，从而避免使整个项趋于无穷大。





#### 2.3 Free Forword

##### 2.3.1 激活函数

1. 为什么使用

* 1. 为了降低过拟合情况

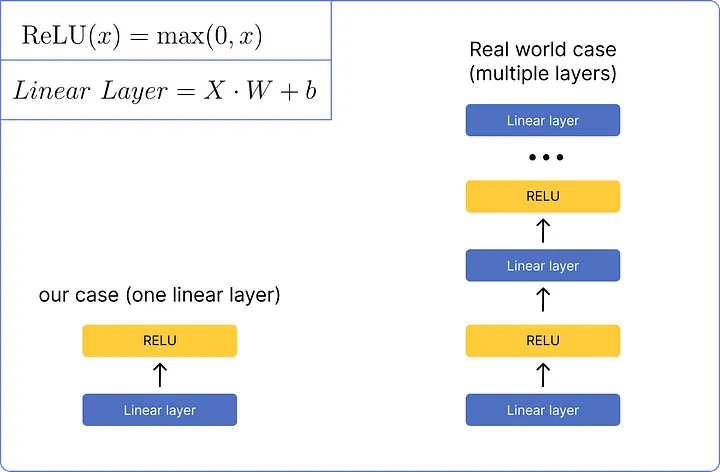
1. 怎么使用

* 1. 简单来说，对于输入 xxx，如果 xxx 大于等于 0，则输出等于 xxx；如果 xxx 小于 0，则输出

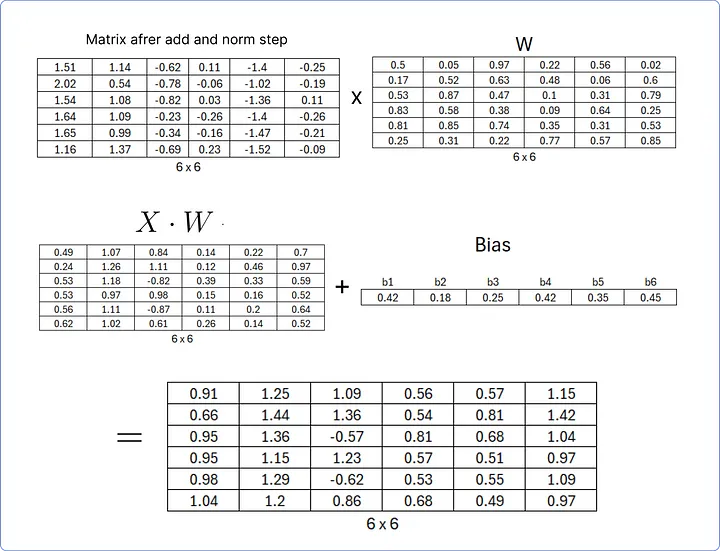
为 0。ReLU 函数的图像为一条从原点开始的直线，当 xxx 大于等于 0 时，函数的斜率为 1，而

当 xxx 小于 0 时，函数的斜率为 0。

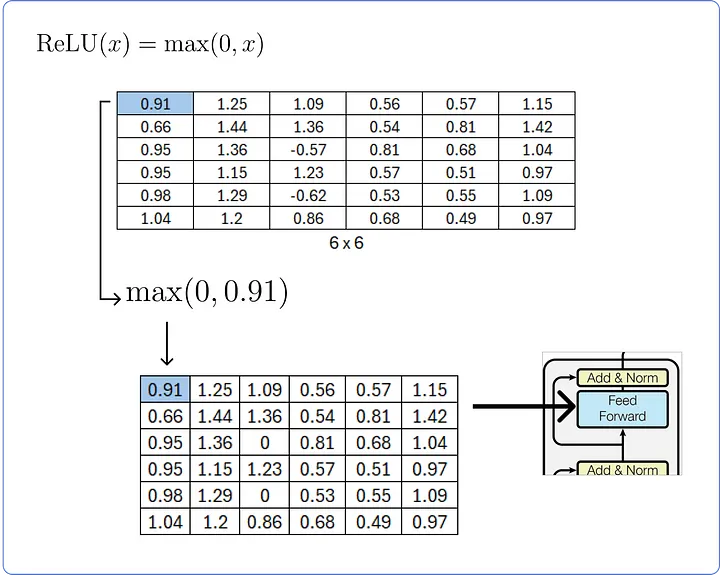
##### 2.3.2 具体操作



1. 进行一次线性连接



1. 进行Relu激活函数



#### 2.4 Add&Norm

1. 继续进行一次残差连接和层级归一化。

### 3. Dncoding

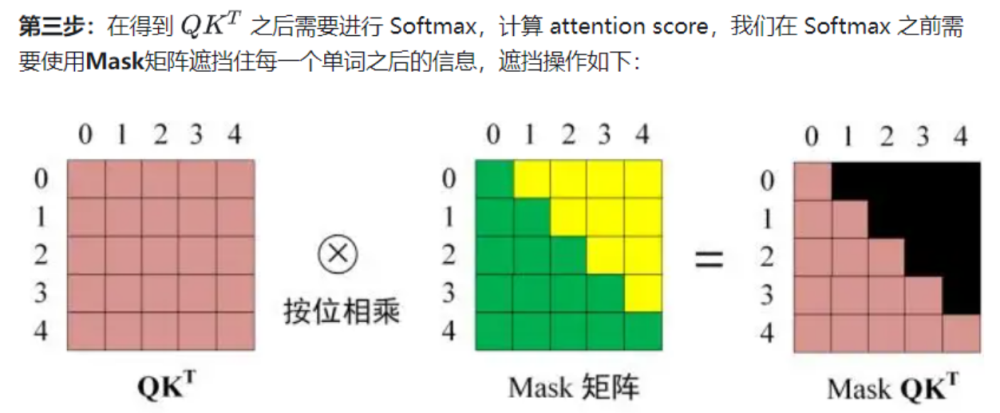
重复性的东西不再讲述

#### 3.1. Masked

1. Decoder block 的第一个 Multi-Head Attention 采用了 Masked 操作，因为在翻译的过程中

是顺序翻译的，即翻译完第 i 个单词，才可以翻译第 i+1 个单词。通过 Masked 操作可以防止

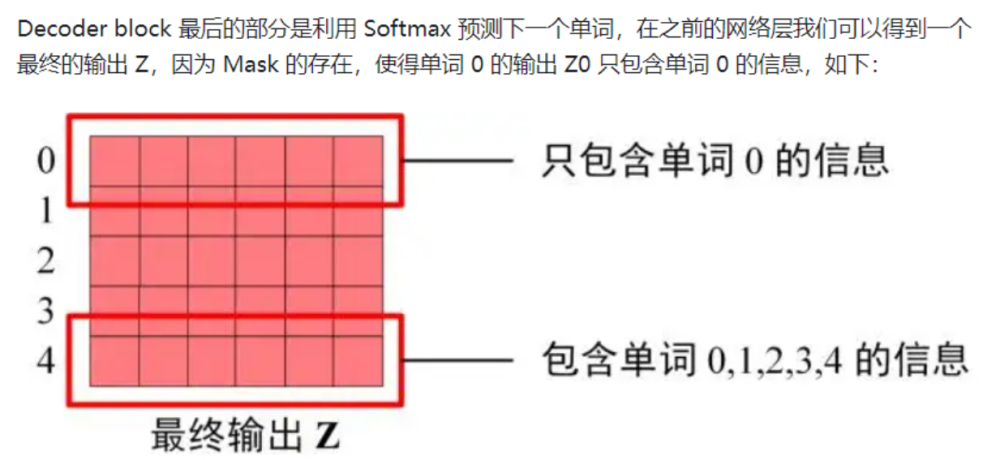
第 i 个单词知道 i+1 个单词之后的信息。

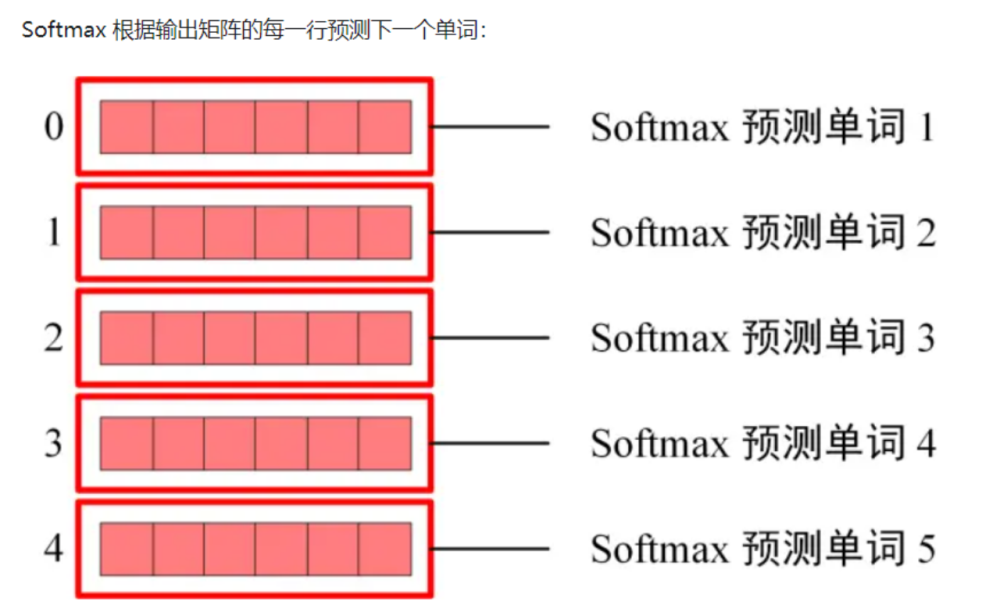


#### 3.2. Multi-Head Attention

1. 和Encoding不同的是，在decoding中，Q是decoding自己所带的，K，V是Encoding提供的。进行一起训练。

#### 3.3. Softmax 预测输出单词





## 二：代码梳理