# 前言：

一、Transformer是什么？

Transformer是一种用于自然语言处理（NLP）和其他序列到序列（sequence-to-sequence）任务的深度学习模型架构，它在2017年由Vaswani等人首次提出。Transformer架构引入了自注意力机制（self-attention mechanism），这是一个关键的创新，使其在处理序列数据时表现出色。

以下是Transformer的一些重要组成部分和特点：

自注意力机制（Self-Attention）：这是Transformer的核心概念之一，它使模型能够同时考虑输入序列中的所有位置，而不是像循环神经网络（RNN）或卷积神经网络（CNN）一样逐步处理。自注意力机制允许模型根据输入序列中的不同部分来赋予不同的注意权重，从而更好地捕捉语义关系。

多头注意力（Multi-Head Attention）：Transformer中的自注意力机制被扩展为多个注意力头，每个头可以学习不同的注意权重，以更好地捕捉不同类型的关系。多头注意力允许模型并行处理不同的信息子空间。

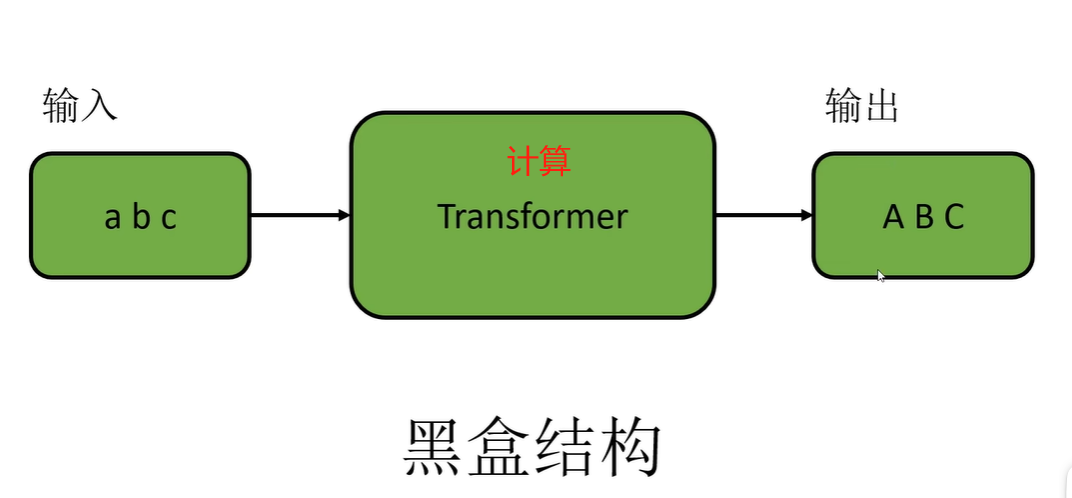
堆叠层（Stacked Layers）：Transformer通常由多个相同的编码器和解码器层堆叠而成。这些堆叠的层有助于模型学习复杂的特征表示和语义。

位置编码（Positional Encoding）：由于Transformer没有内置的序列位置信息，它需要额外的位置编码来表达输入序列中单词的位置顺序。

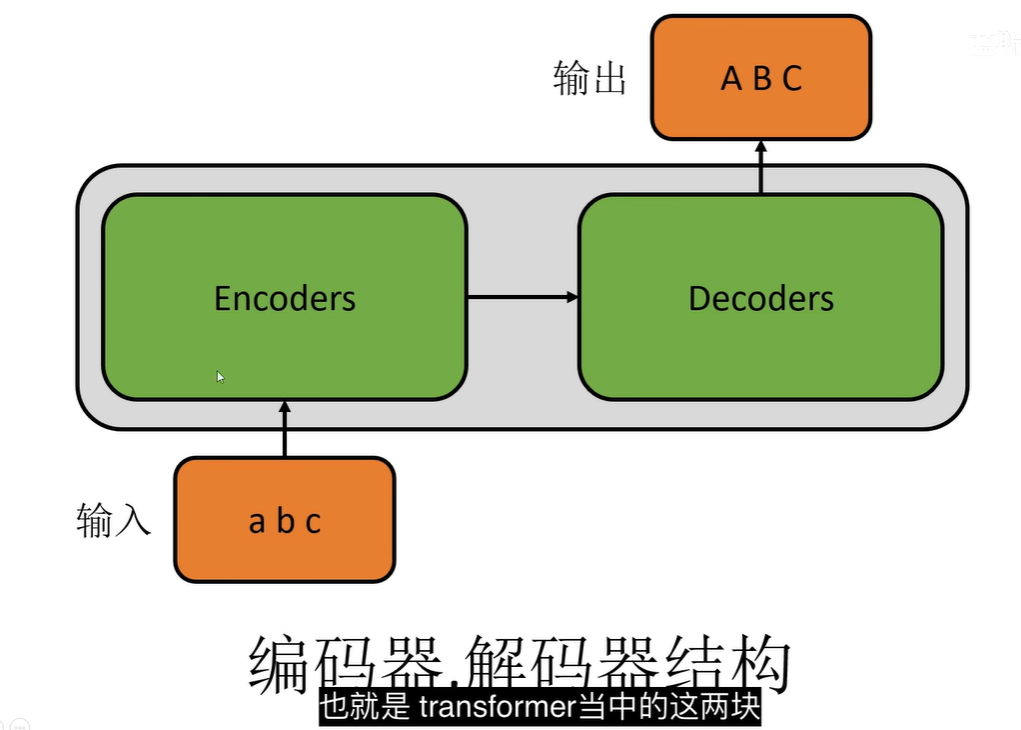
残差连接和层归一化（Residual Connections and Layer Normalization）：这些技术有助于减轻训练过程中的梯度消失和爆炸问题，使模型更容易训练。

编码器和解码器：Transformer通常包括一个编码器用于处理输入序列和一个解码器用于生成输出序列，这使其适用于序列到序列的任务，如机器翻译。

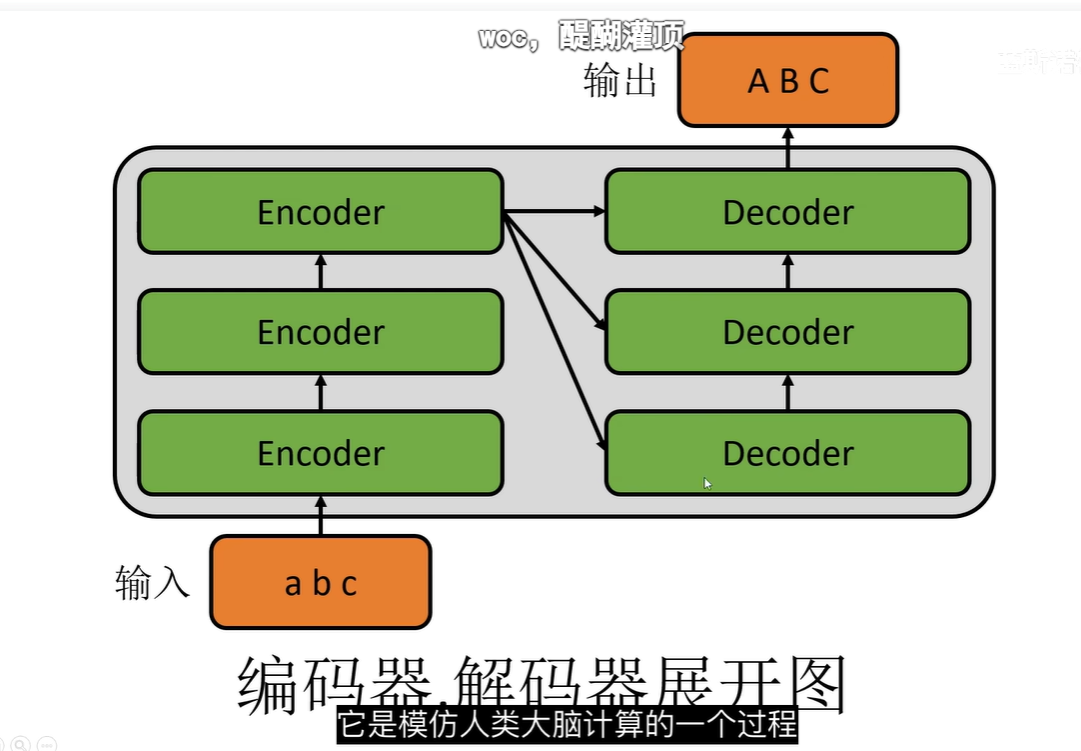
# 总体架构



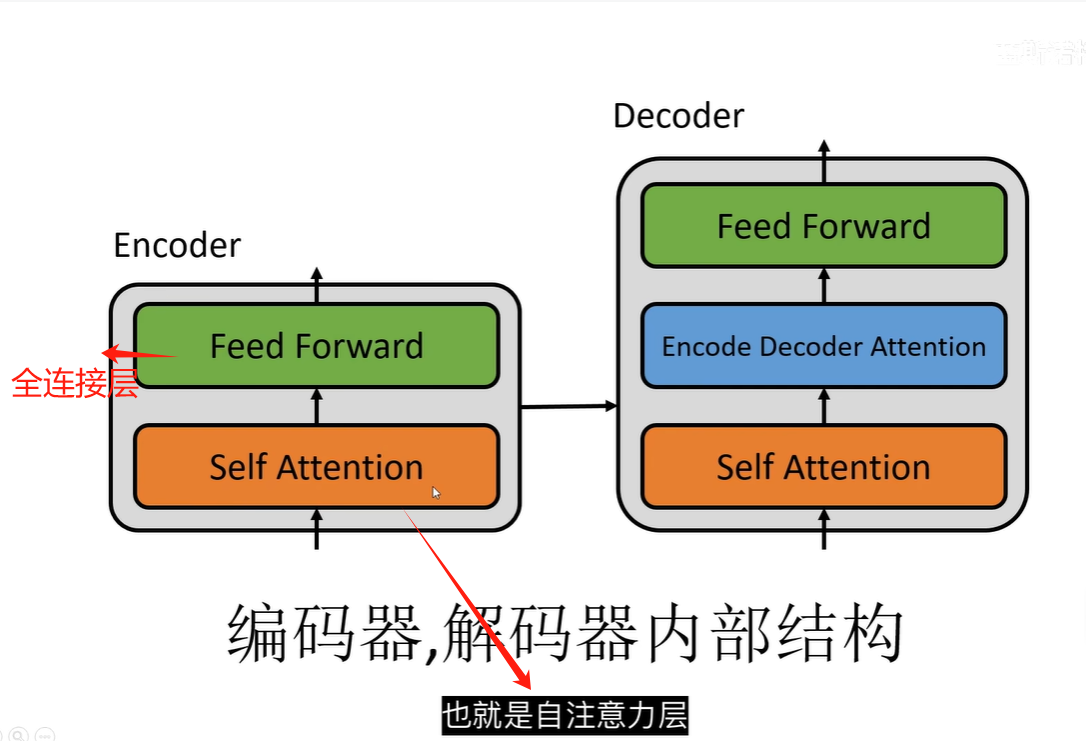
Transformer所做的就是把输入的文本计算成输出的文本，将黑盒的结构展开来看：



简单来看可以分成编码与解码的过程，模拟人脑来看就是读和写的过程，编码与解码可以有n层。

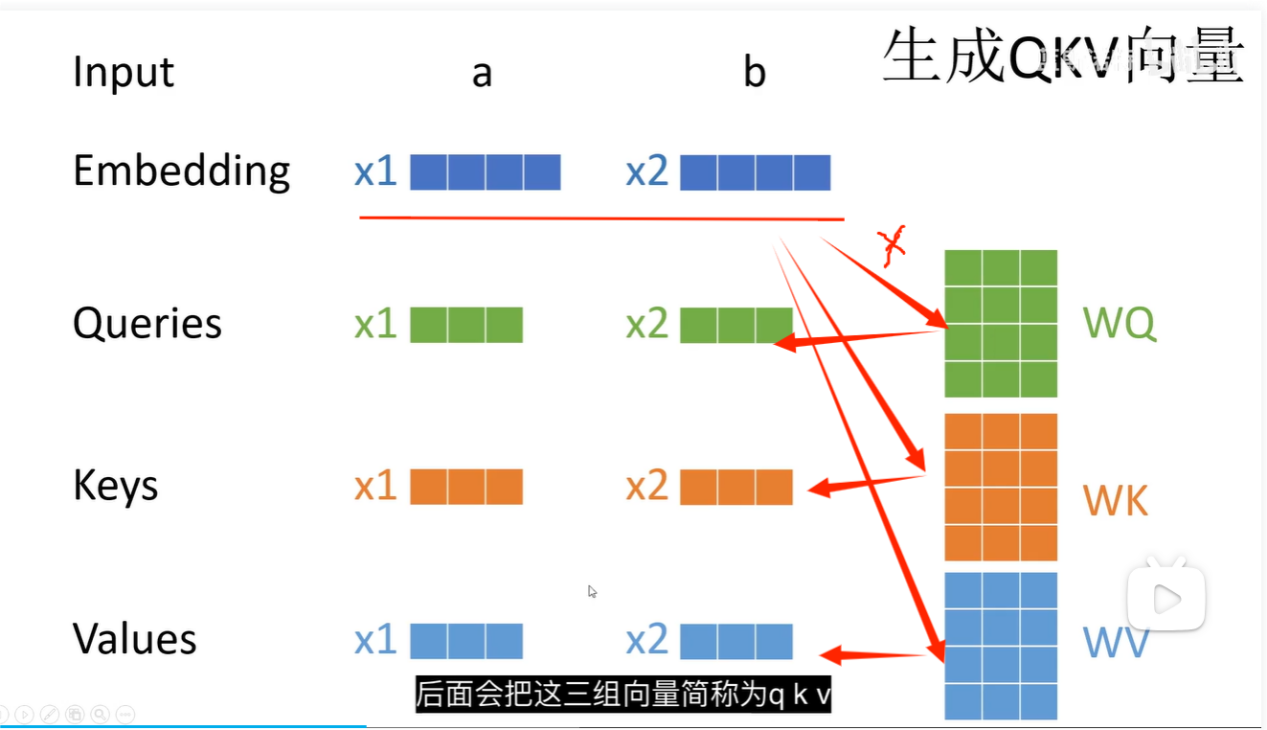


编码器与解码器的内部结构：



解码器比编码器多了一层结构，下文会对这一层详细介绍，首先介绍如何计算注意力。

# **计算注意力**



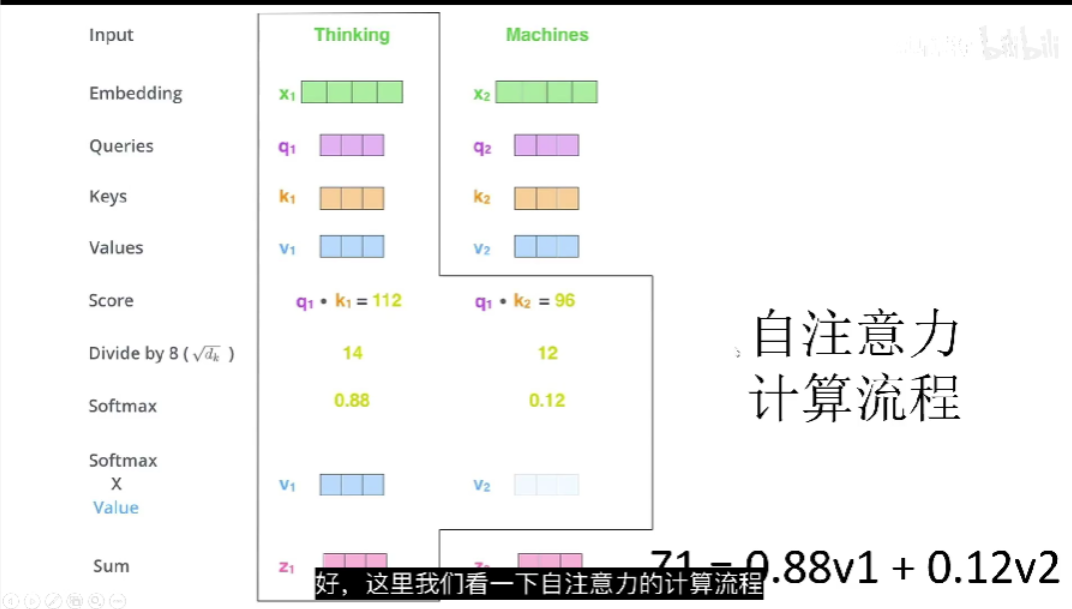
WQ，WK，WV这三个矩阵的参数，都是在训练当中，学习到的。

上图表示输入是两个词a,b。

Embedding表示将输入的两个词进行向量化为两个向量x1和x2。注意：通过图片的给出的向量x1和向量x2发现。两个向量都是一个一行四列的矩阵，这个结论很重要，会与后期的注意力计算机制相联系。

通过图片中的右下角位置，发现一共有三个变量WQ\WK\WV，且三个变量都是四行三列的矩阵。当输入的数据向量化得到的矩阵与之相乘的结果，就是一个一行三列的矩阵。即分别对应着：Queries\Keys\Values三个变量。计算得到的这三个变量很重要，因为将由这三个向量计算整个模型中的注意力机制。后面会将这三个向量简称为q,k,v。

# **自注意力向量形式计算的过程**



上图表示的含义为：

①此时输入的Thinking 和Machines。通过将输入的词进行向量化后得到对应的向量化表示x1，x2。然后将向量化表示的变量分别乘以对应的WQ,WK,WV得到对应的q1,k1,v1。

②上图中用黑线画出的部分就是计算Thinking的自注意力。首先是计算score，计算方式是Thinking的q1,分别乘以k1和k2（这里在计算的过程中，应该涉及到聚举证的转置），得到数值分别为112和96。

③计算score完成后，然后将计算得到的数值。除以8，分别得到14和12。

除以8，详细解析？——与词向量编码相关

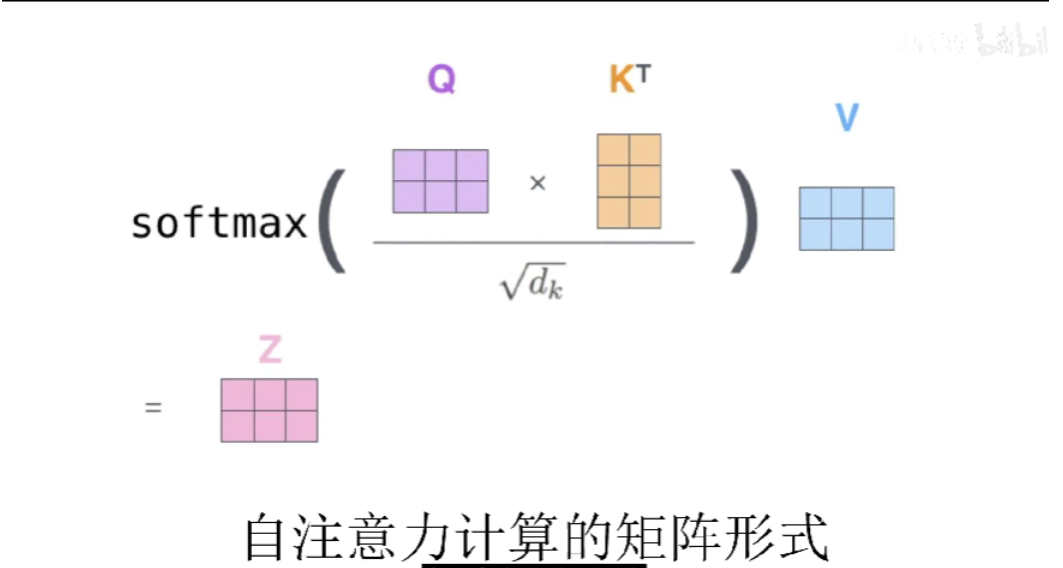
④然后将除以8的到的结果，通过softmax激活函数处理，得到两个小数为0.88和0.12。

​

⑤然后将softmax处理后的结果，分别与输入向量的v相乘。得到两个新的v1和v2向量。

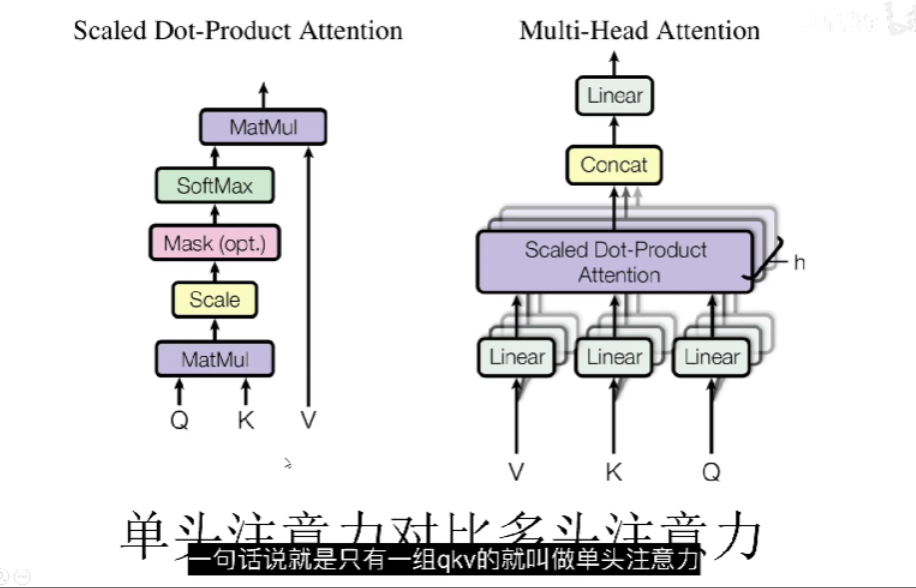
⑥最后将两个新的v1、v2向量进行相加，得到向量z1=0.88v1+0.12v2。且这个变量z就是自注意力计算的结果。

# **自注意力的矩阵形式计算过程**



q和k的转置进行点积运算，然后除以dk的算术平方根。最后将计算的结果经过softmax函数处理，乘以v就得到z。

# **单头注意力机制和多头注意力机制**

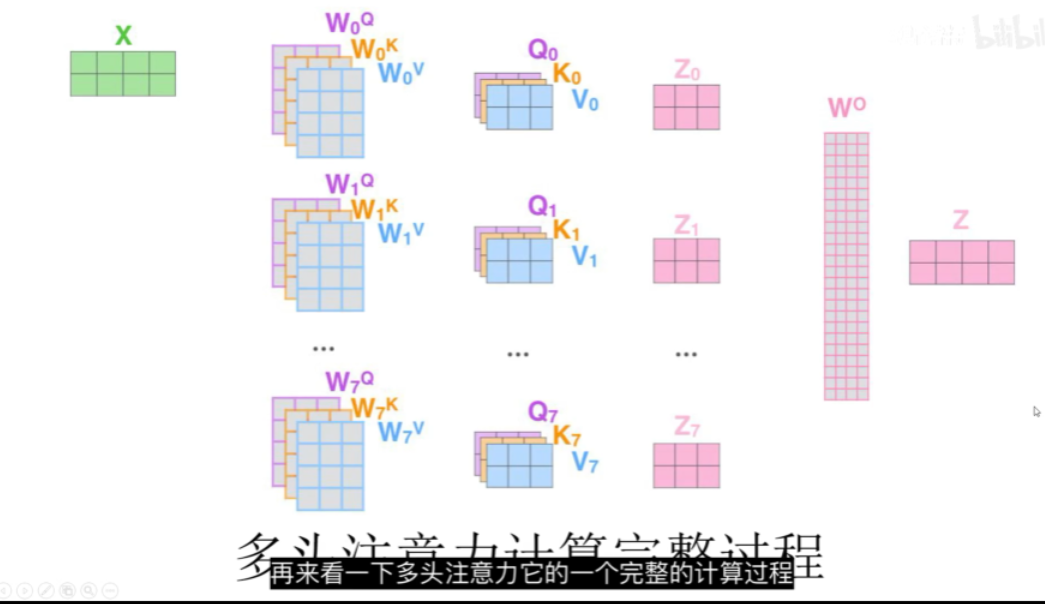


简单理解单头注意力机制和多头注意力机制：

单头注意力机制只有一组q，k,v。（q,k,v是输入数据经过向量化处理后与三个矩阵相乘后得到的向量。）

多头注意力机制有多组q，k,v。（根据多头注意力机制的定义：我们可以知道多头注意力机制，则应该对应着多组向量。且一组向量对应着三个计算qkv向量的矩阵），然后将计算得到的多个向量q,k,v通过concat组合到一起。

多头注意力机制的详细计算过程：



参与计算q,k,v的矩阵是叠加在一起的。而且有8组。所以根据前文介绍的知识，可以知道计算的结果就会对应着八个不同的q,k,v向量。也对应着八组自注意力计算结果。然后通过线性运算wo，将得到八组自注意力机制计算结果进行组合，得到最后的注意力结果z。

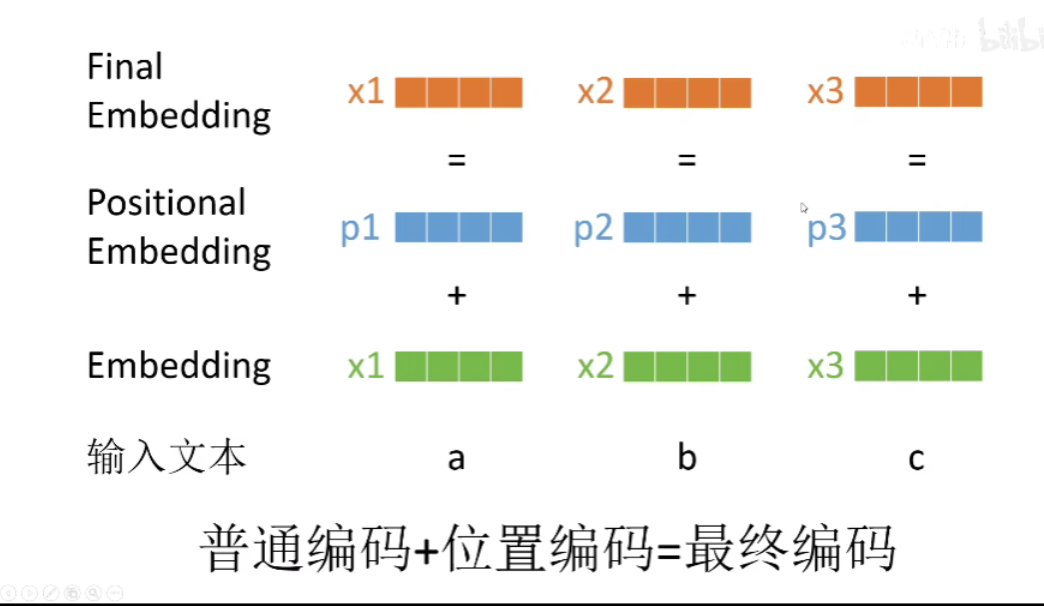
# **词向量编码**

transformer在进行词向量编码的时候，会向数据中加入位置信息，从而使得transformer神经网络可以区分不同位置的相同数据。

例如：

你欠我250

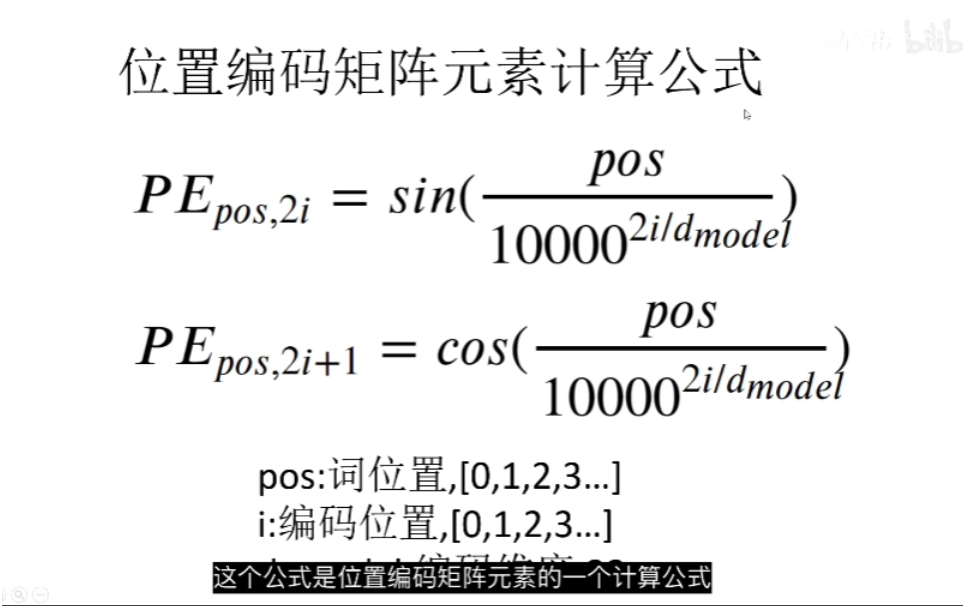
我欠你250



上图是transformer进行词向量编码的过程：

就是现将输入的数据进行编码，然后再将输入数据的位置信息，也对应的生成一个位置向量，最后将位置编码和输入信息编码进行相加就得到了最后的输入数据编码。

## 位置编码矩阵元素计算公式



PE是一个很大的矩阵，然后pos表示是第几个词（表示矩阵的行数）。i表示词向量编码之后的第i个向量（表示矩阵的列数）。

上述两个公式区别：

第一个公式计算偶数列数据、第二个公式计算奇数类数据

# **mask相关知识点**



PAD MASK不计算针对PAD的注意力。也就是减少计算机的错误工作，将计算中心更多的放在词语上而不是填充上。

通过上述的图片我们可以看到第一行的数据为：ab<pad> ，第1列的内容为：

ab<pad>我们将第一行和第一列的内容都看成是一句话。在实际的运算的时候是多句话一起计算的。在多句话的计算过程中，可能出现的问题是句子的长短不一。为了统一计算，我们需要将每句话的长度进行统一。具体的做法为：将短的句子进行补长。上述的例子中，我们是将一个句子的长度扩充为3。

在计算的时候每个词对pad的注意力都是mask，但是pad对其他词的注意力，正常计算。

以上就是pad mask方式计算注意力的过程。

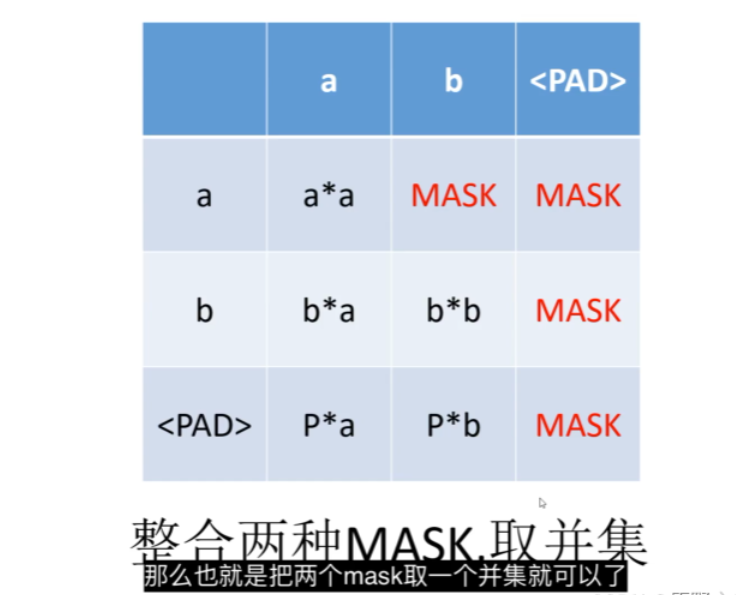


什么意思？就是由a预测b，所以不应计算a对b即c的注意力。

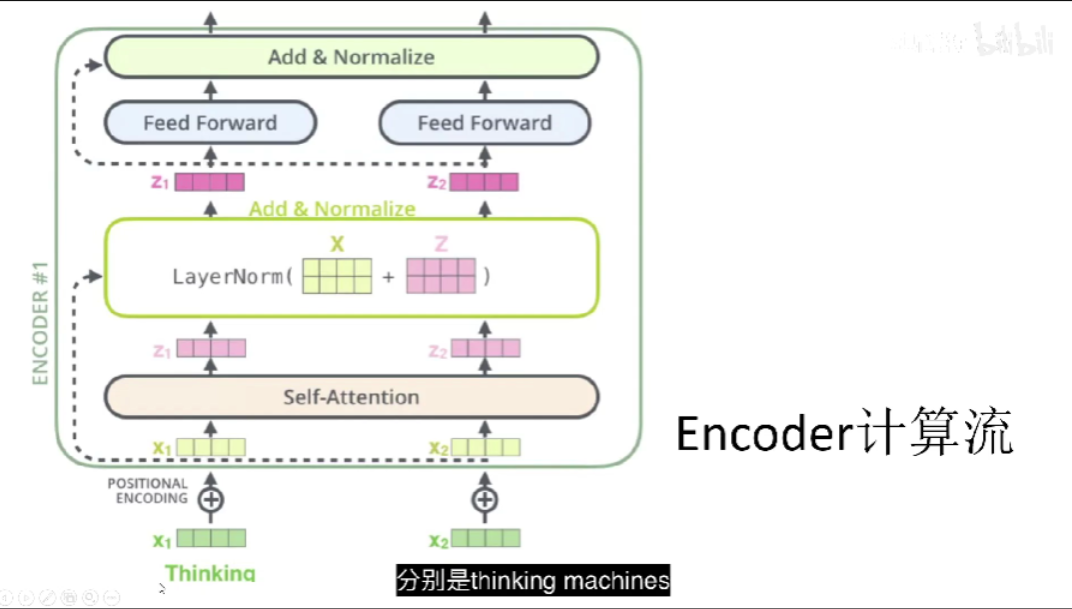
当计算对a的注意力的时候，不关注其他的词。因为b和c是需要让a进行预测的结果。即解码器中的一个计算，需要根据a预测出b和c这个结果。如果让a关注到b这不能起到预测的作用，因为会直接得到结果。

同样在计算b的注意力的时候：不能让b关注c，因为在transformer中的阶码器阶段需要根据b预测c。

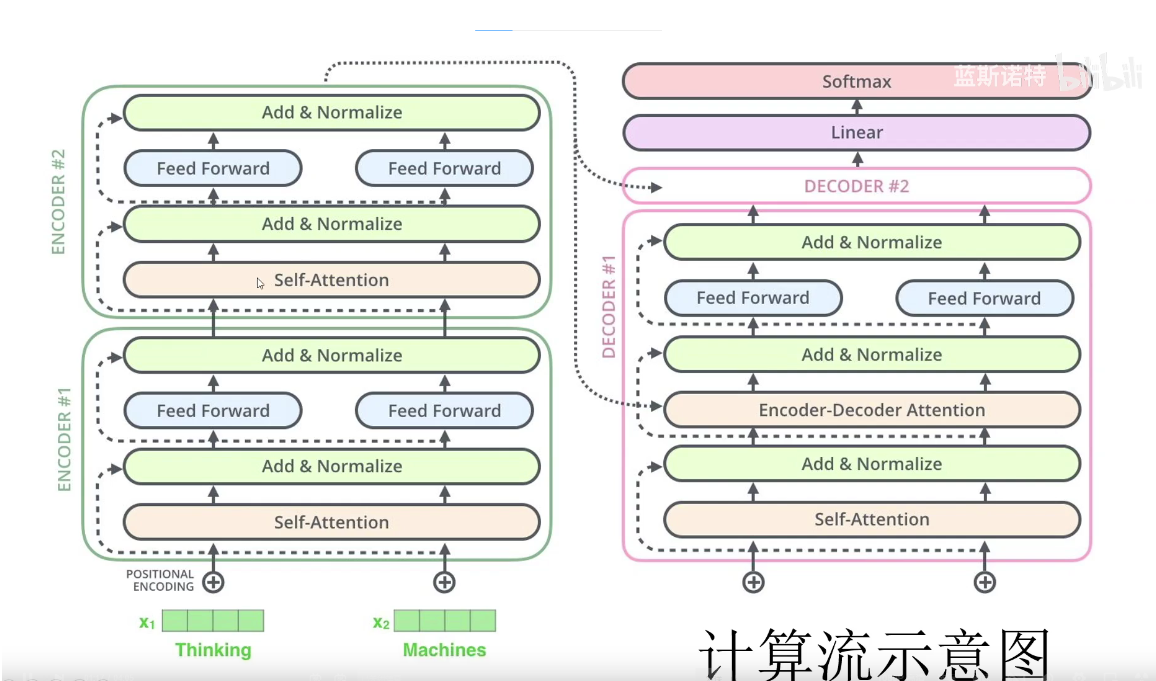
接着整合上三角mask和padmask的计算注意力方式：



# **transformer的完整计算流程**

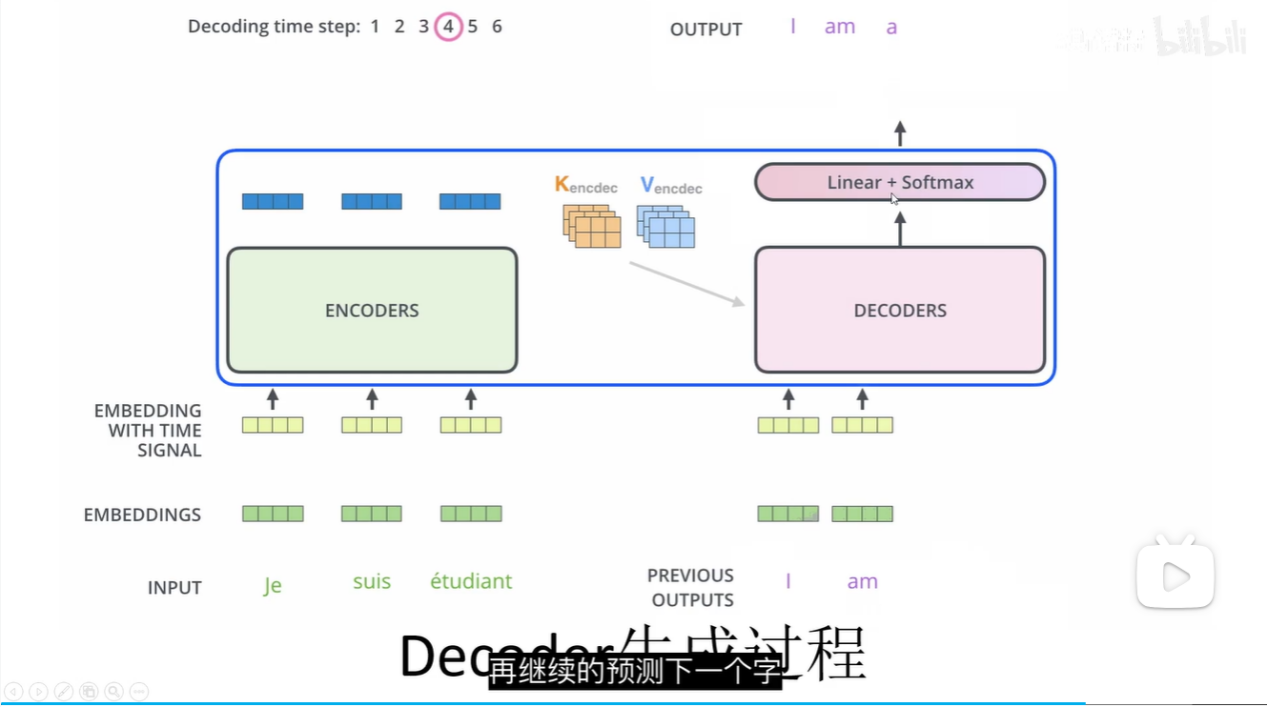


LayerNorm部分表示短接：简单的理解为将xi和zi进行相加。



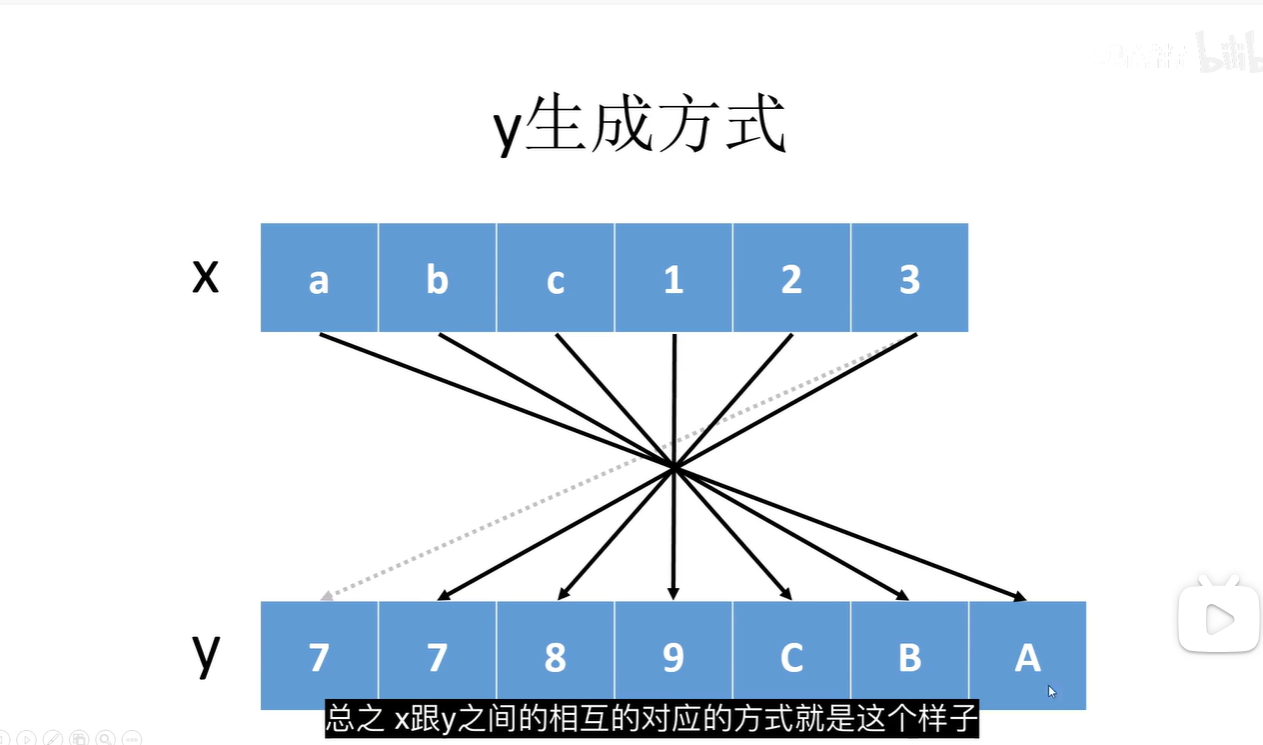
在实际的transformer网络中encoder不只一个，可能存在n个encoder，这n个encoder之间是上下串联的关系。然后将最后计算的结果输出到decoder中计算。

decoder的过程与encoder部分基本相似。不同点在于：encoder decoder attention的计算k，v是encode的计算结果。然后自己的encode 输出的结果作为q。然后通过短接，然后通过全连接层处理。decoder有多个，多个decoder是串联关系。将经过多个decoder计算的结果，使用线性方程（softmax）处理输出预测结果。



# **训练过程**

**该任务是一个字符翻译工作：**

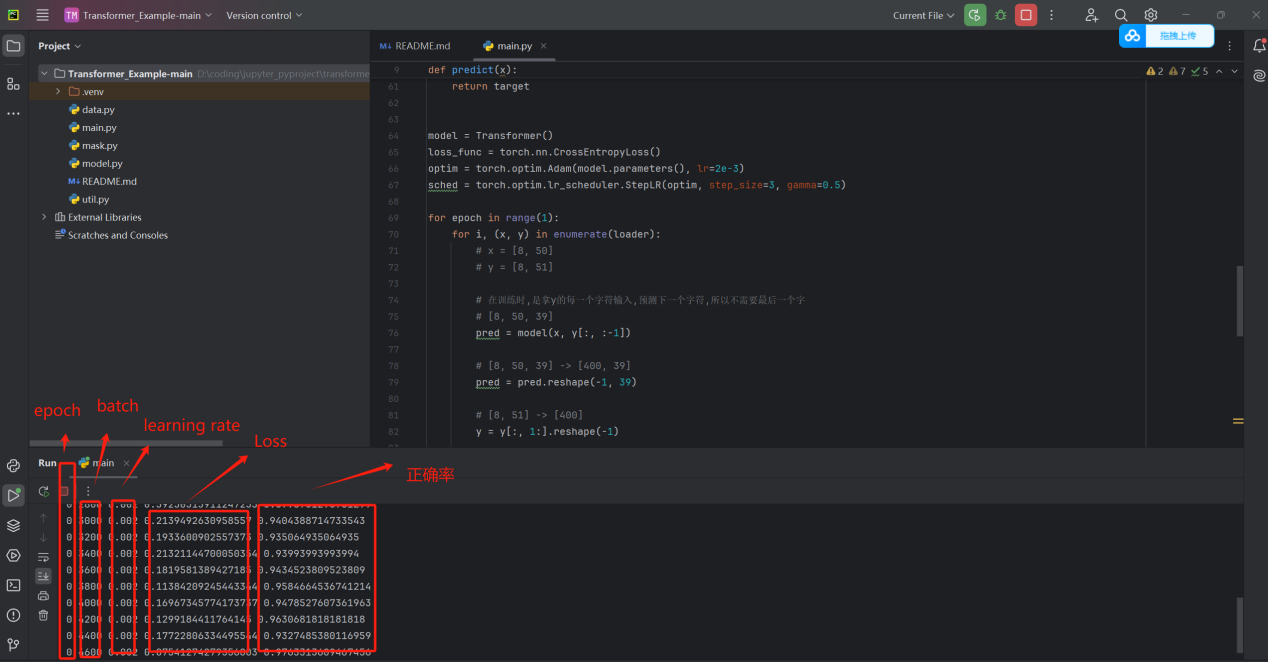


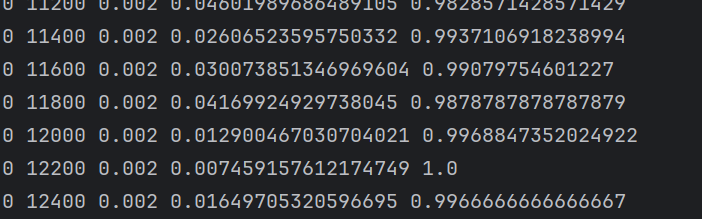
小写变大写，1-9，2-8，3-7.....,逆序输出

源码：



**训练过程：**





训练结束发现正确率已经很高了

测试过程：



测试结果与实际偏差非常小，可见训练效果很好。