



温州大學
WENZHOU UNIVERSITY

深度学习-Transformer

黄海广 副教授

2023年05月

01 Transformer介绍

02 Transformer的工作流程

03 Transformer的训练

04 BERT

1.Transformer介绍

3

01 Transformer介绍

02 Transformer的工作流程

03 Transformer的训练

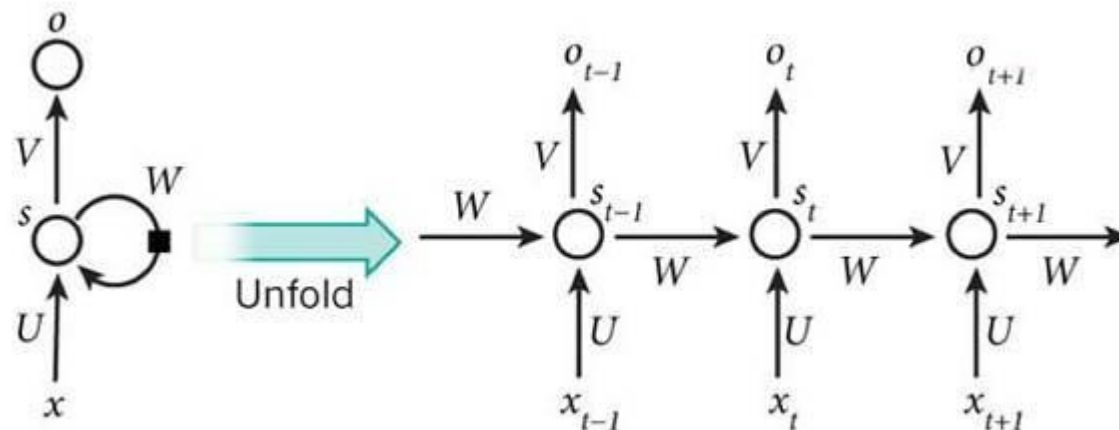
04 BERT

1.Transformer介绍

4

为什么需要用transformer

在没有transformer的时候，我们都是用什么来完成这系列的任务的呢？



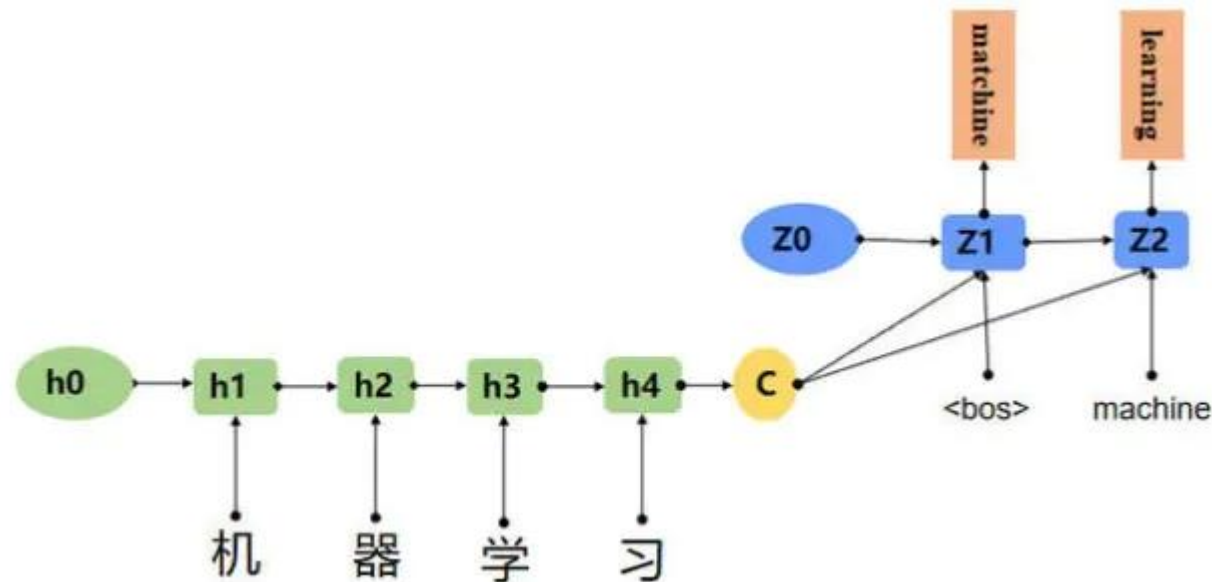
其实在之前我们使用的是RNN（或者是其的单向或者双向变种LSTM/GRU等）来作为编解码器。RNN模块每次只能够吃进一个输入token和前一次的隐藏状态，然后得到输出。它的时序结构使得这个模型能够得到长距离的依赖关系，但是这也使得它**不能够并行计算**，模型效率十分低。

1.Transformer介绍

5

Seq2Seq任务

Seq2Seq 任务指的是输入和输出都是序列的任务，输出的长度不确定时采用的模型，这种情况一般是在机器翻译的任务中出现，将一句中文翻译成英文，那么这句英文的长度有可能会比中文短，也有可能会比中文长，所以输出的长度就不确定了。



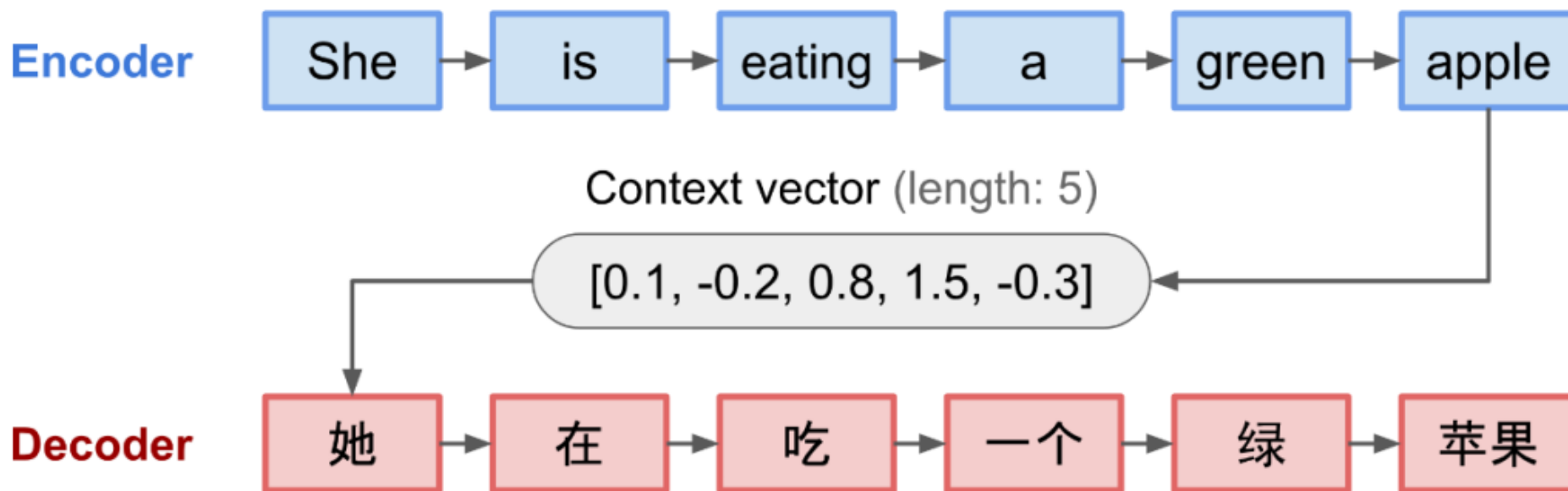
上图，输入的中文长度为4，输出的英文长度为2

1.Transformer介绍

6

Encoder-Decoder模型

通常来说，Seq2Seq任务最常见的是使用Encoder+Decoder的模式，先将一个序列编码成一个上下文矩阵，在使用Decoder来解码。当然，我们仅仅把context vector作为编码器到解码器的输入。

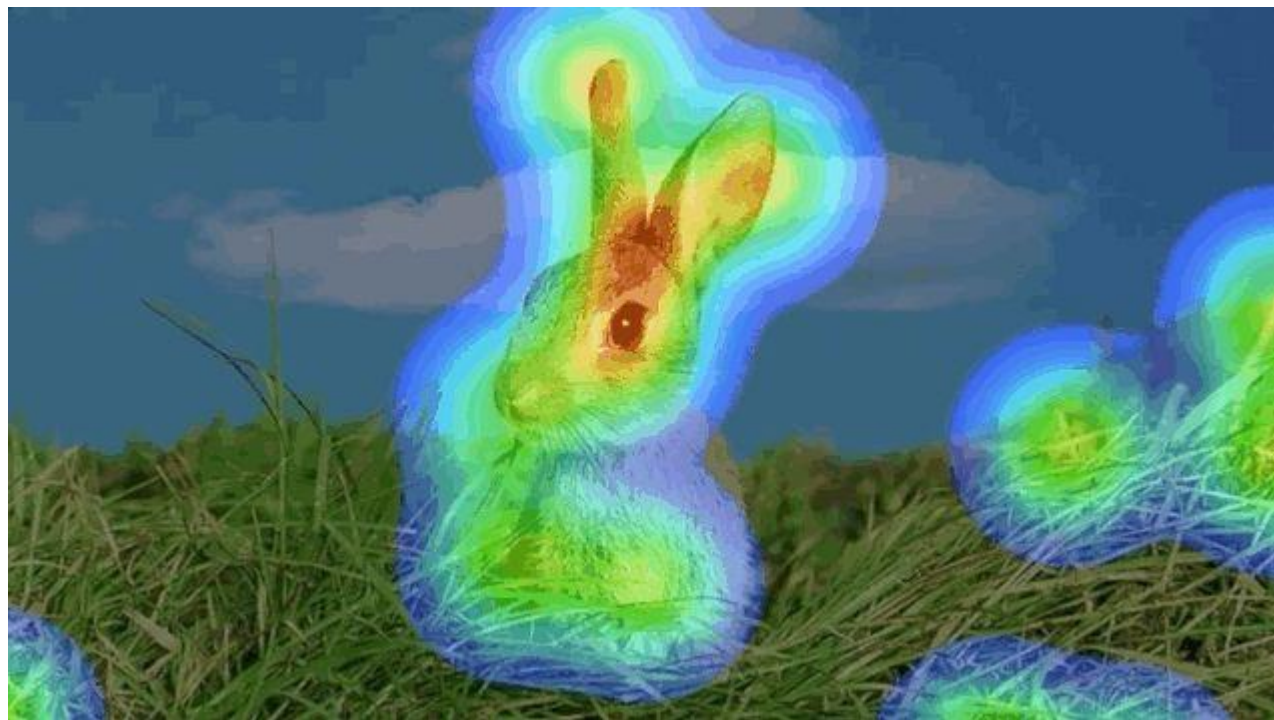


1.Transformer介绍

7

Attention注意力机制

在介绍什么是注意力机制之前，先让大家看一张图片。当大家看到下面图片，会首先看到什么内容？当过载信息映入眼帘时，我们的大脑会把注意力放在主要的信息上，这就是大脑的注意力机制。



1.Transformer介绍

8

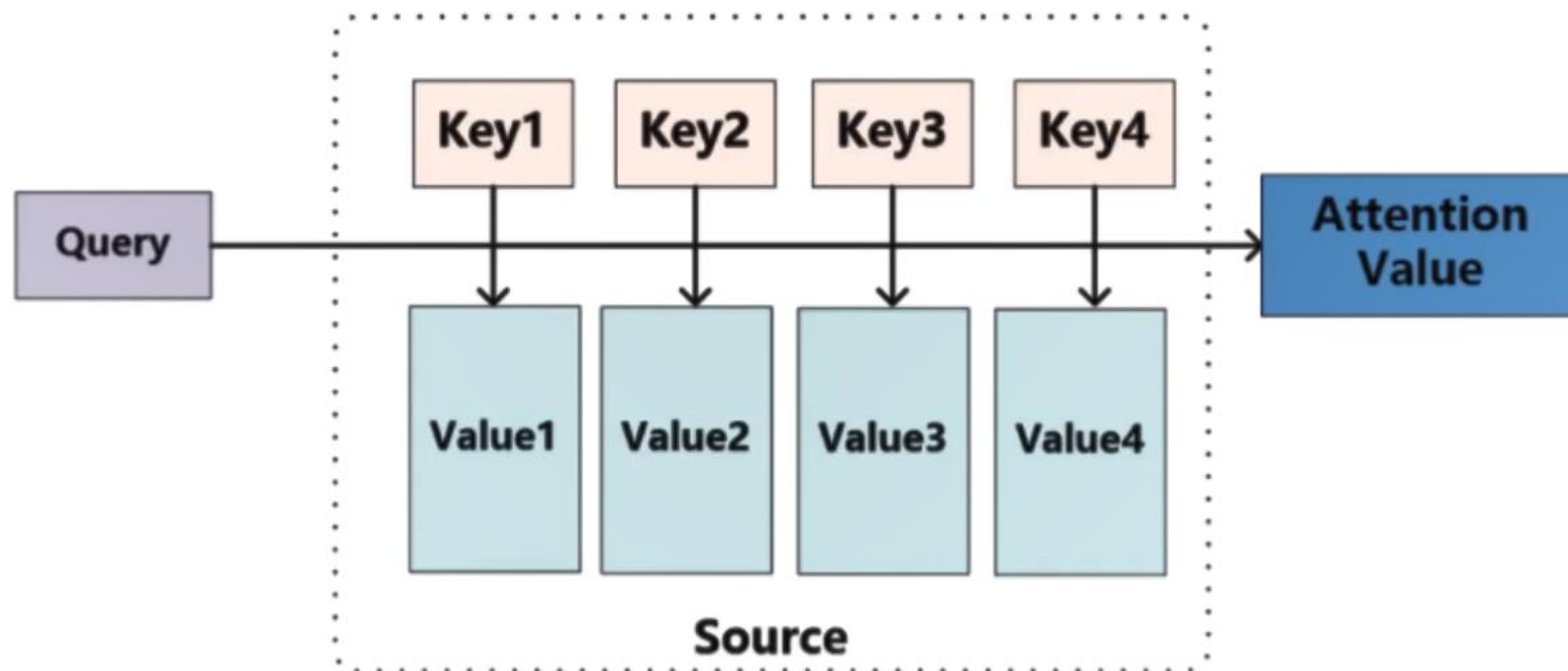
每个词的Attention计算

每个词的Q会跟整个序列中每一个K计算得分，然后基于得分再分配特征

Q: query, 要去查询的

K: key, 等着被查的

V: value, 实际的特征信息



1.Transformer介绍

9

Attention的优点

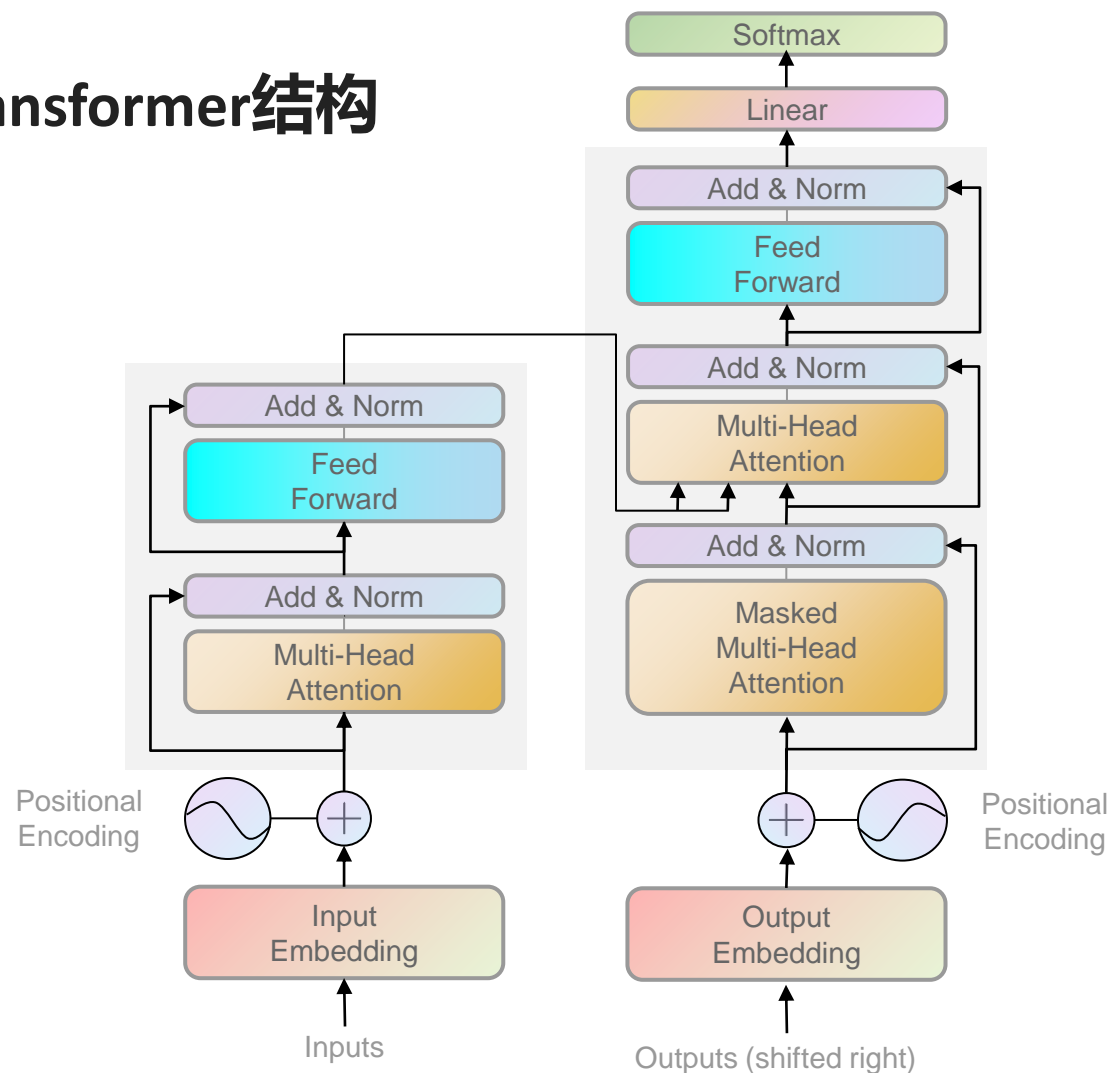
- 1.**参数少**：相比于 CNN、RNN，其复杂度更小，参数也更少。所以对算力的要求也就更小。
- 2.**速度快**：Attention 解决了 RNN及其变体模型不能并行计算的问题。Attention机制每一步计算不依赖于上一步的计算结果，因此可以和CNN一样并行处理。
- 3.**效果好**：在Attention 机制引入之前，有一个问题大家一直很苦恼：长距离的信息会被弱化，就好像记忆能力弱的人，记不住过去的事情是一样的。

1.Transformer介绍

10

2017年google的机器翻译团队在NIPS上发表了**Attention is all you need**的文章，开创性地提出了在序列转录领域，完全抛弃CNN和RNN，只依赖**Attention-注意力**结构的简单的网络架构，名为**Transformer**；论文实现的任务是机器翻译。

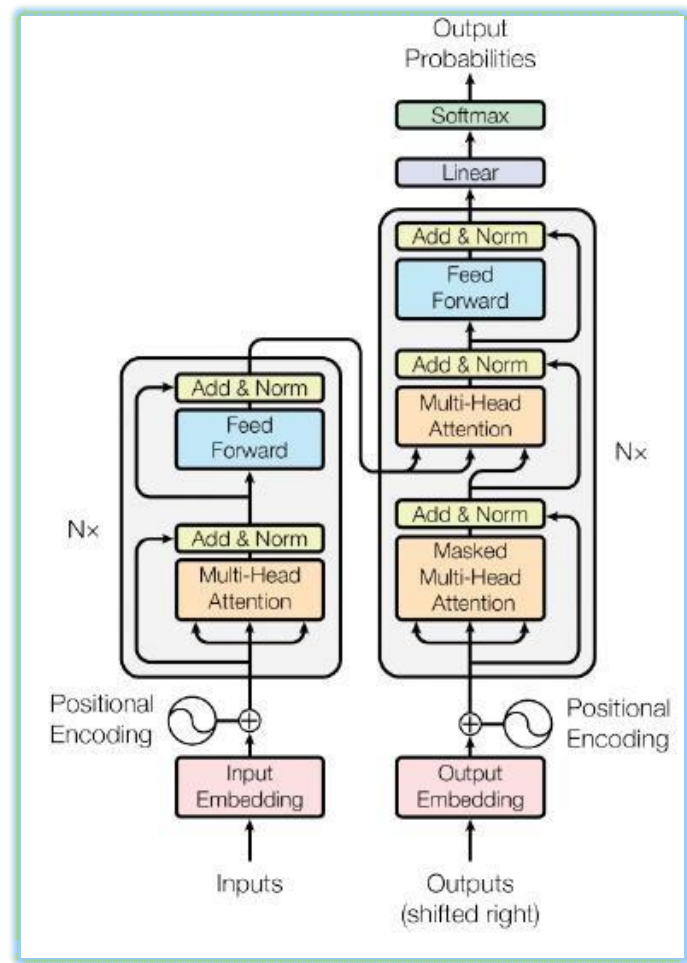
Transformer结构



1.Transformer介绍

11

- **Transformer**摆脱了人工标注数据集的缺陷，模型在质量上更优、更易于并行化，所需训练时间明显更少
- **Transformer**通过成功地将其应用于具有大量和有限训练数据的分析，可以很好地推广到其他任务
- **Transformer**，它完全基于注意力机制，完全不用重复和卷积，因而这些模型在质量上更优，同时更易于并行化，并且需要的训练时间明显更少。
- **Transformer**出现以后，迅速取代了RNN系列变种，跻身主流模型架构基础。（RNN缺陷正在于流水线式的顺序计算）



图：Transformer模型架构

资料来源：《Attention Is All You Need》, Ashish Vaswani et.al 2017

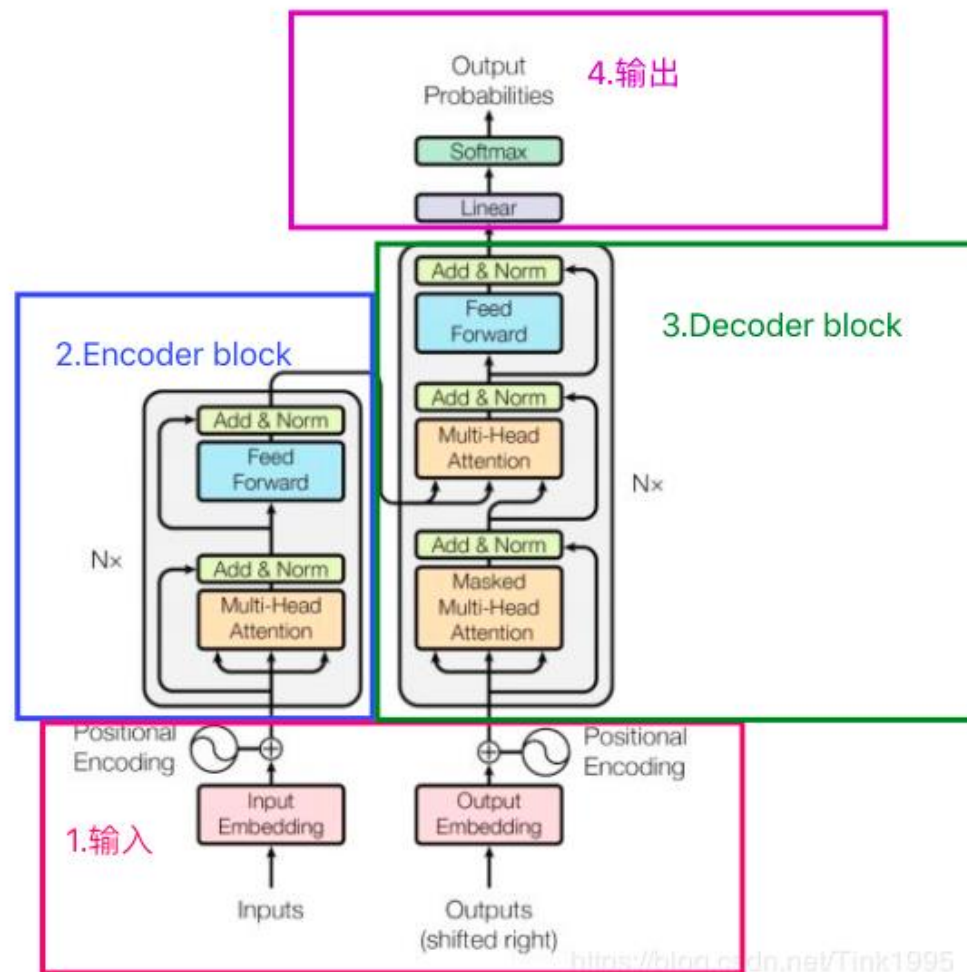
Transformer

12

Transformer —— 大力出奇迹的起点

- 在Transformer提出之后，大模型的基础模型架构基本形成，注意力机制代替卷积神经网络称为主流基础模型组件
 - 有利于模型向更大的参数量扩展
 - Transformer有兼容多模态信息的天生优势特性，这有力地丰富了大模型的应用场景。

参数少 速度快 效果好



2.Transformer的工作流程

13

01 Transformer介绍

02 Transformer的工作流程

03 Transformer的训练

04 BERT

2.Transformer的工作流程

14

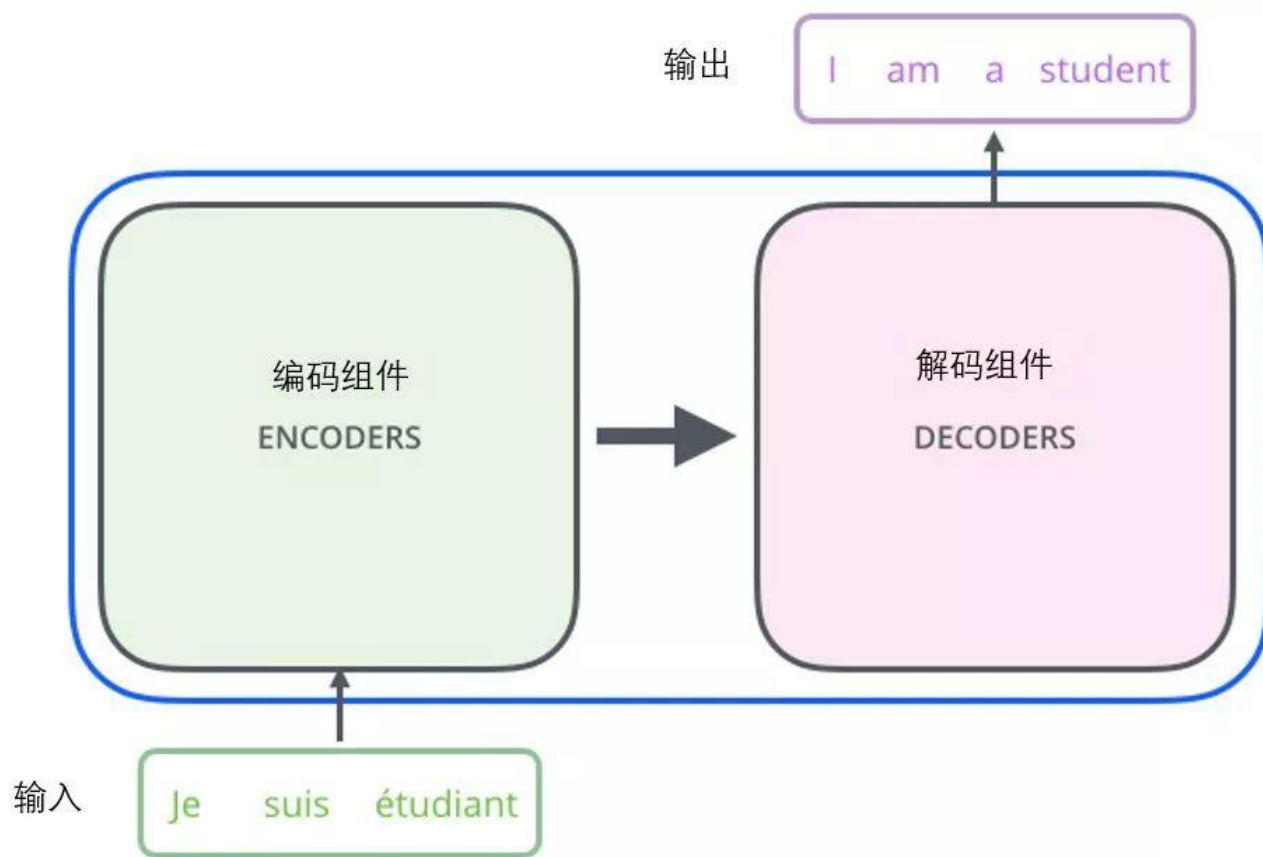
从宏观的视角开始 首先将这个模型看成是一个黑箱操作。在机器翻译中，就是输入一种语言，输出另一种语言。



2.Transformer的工作流程

15

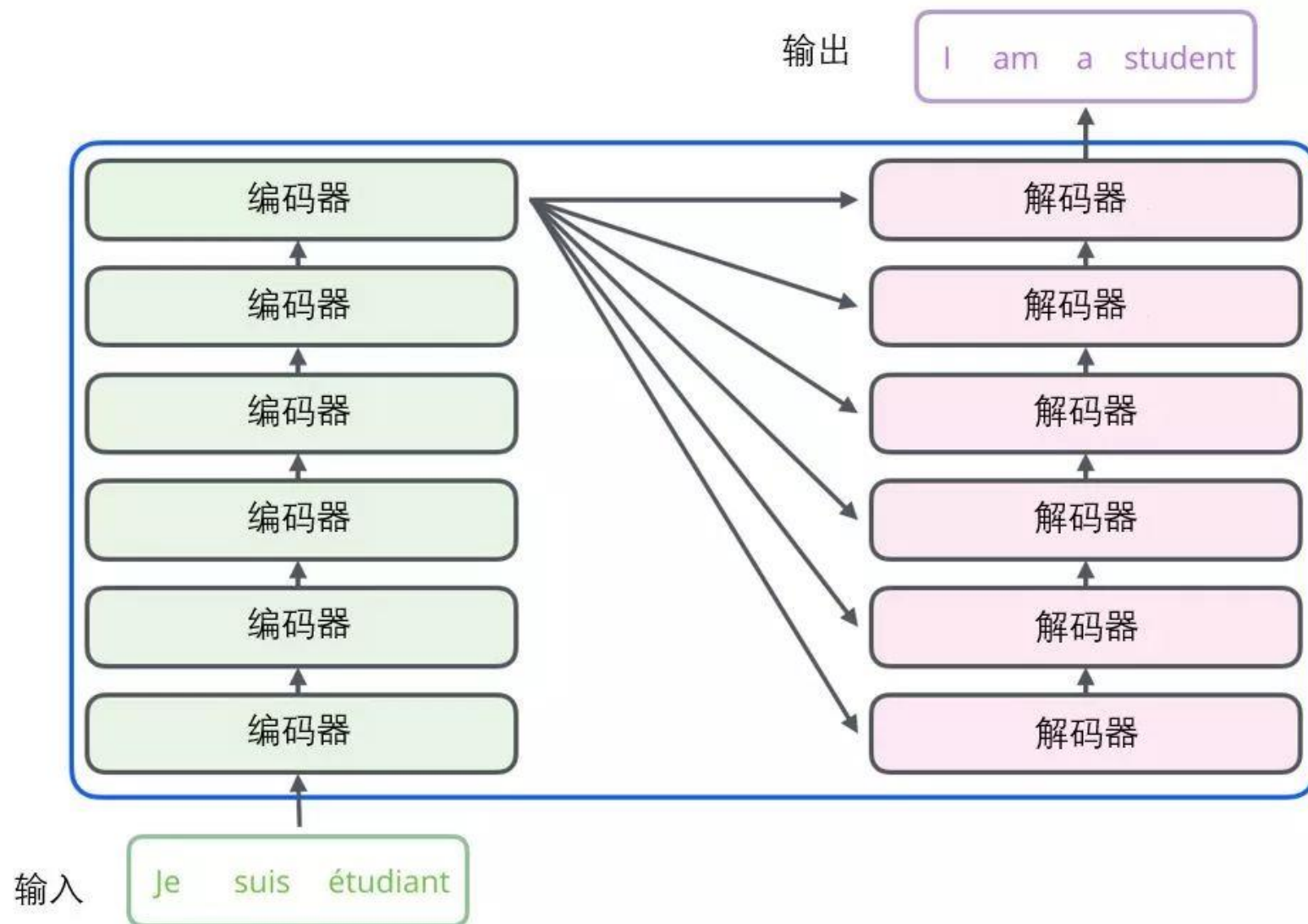
那么拆开这个黑箱，我们可以看到它是由编码组件、解码组件和它们之间的连接组成。



2.Transformer的工作流程

16

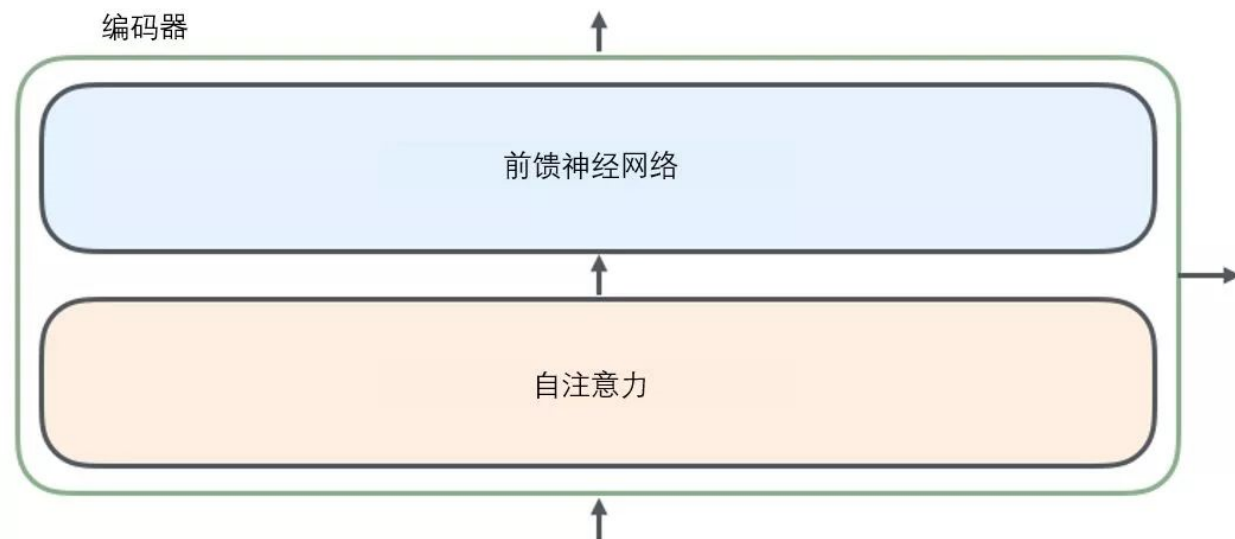
编码组件部分由一堆编码器（encoder）构成（论文中是将6个编码器叠在一起）。解码组件部分也是由相同数量（与编码器对应）的解码器（decoder）组成的。



2.Transformer的工作流程

17

所有的编码器在结构上都是相同的，但它们没有共享参数。每个解码器都可以分解成两个子层。



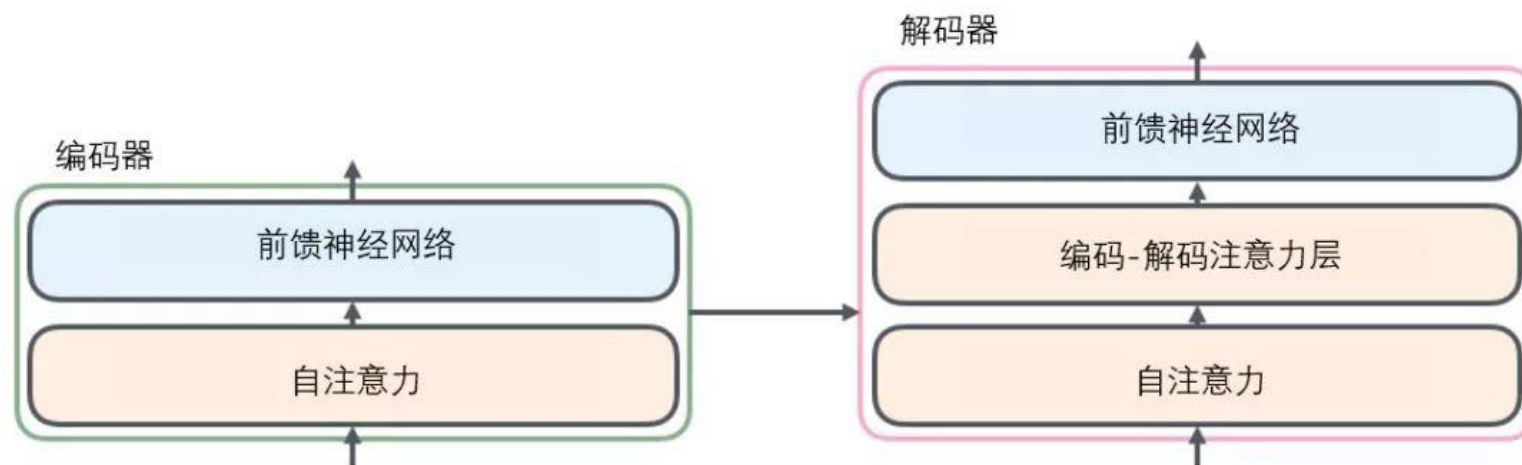
2.Transformer的工作流程

18

从编码器输入的句子首先会经过一个自注意力（self-attention）层，这层帮助编码器在对每个单词编码时关注输入句子的其他单词。

自注意力层的输出会传递到前馈（feed-forward）神经网络中。每个位置的单词对应的前馈神经网络都完全一样（译注：另一种解读就是一层窗口为一个单词的一维卷积神经网络）。

解码器中也有编码器的自注意力（self-attention）层和前馈（feed-forward）层。除此之外，这两个层之间还有一个注意力层，用来关注输入句子的相关部分（和seq2seq模型的注意力作用相似）。

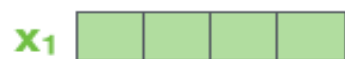


2.Transformer的工作流程

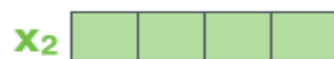
19

各种向量或张量是怎样在模型的不同部分中，将输入转化为输出的。

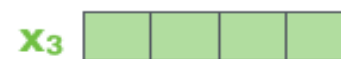
像大部分NLP应用一样，我们首先将每个输入单词通过词嵌入算法转换为词向量。



Je



suis



étudiant

每个单词都被嵌入为512维的向量，词嵌入过程只发生在最底层的编码器中。所有的编码器都有一个相同的特点，即它们接收一个向量列表，列表中的每个向量大小为512维。在底层（最开始）编码器中它就是词向量，但是在其他编码器中，它就是下一层编码器的输出（也是一个向量列表）。

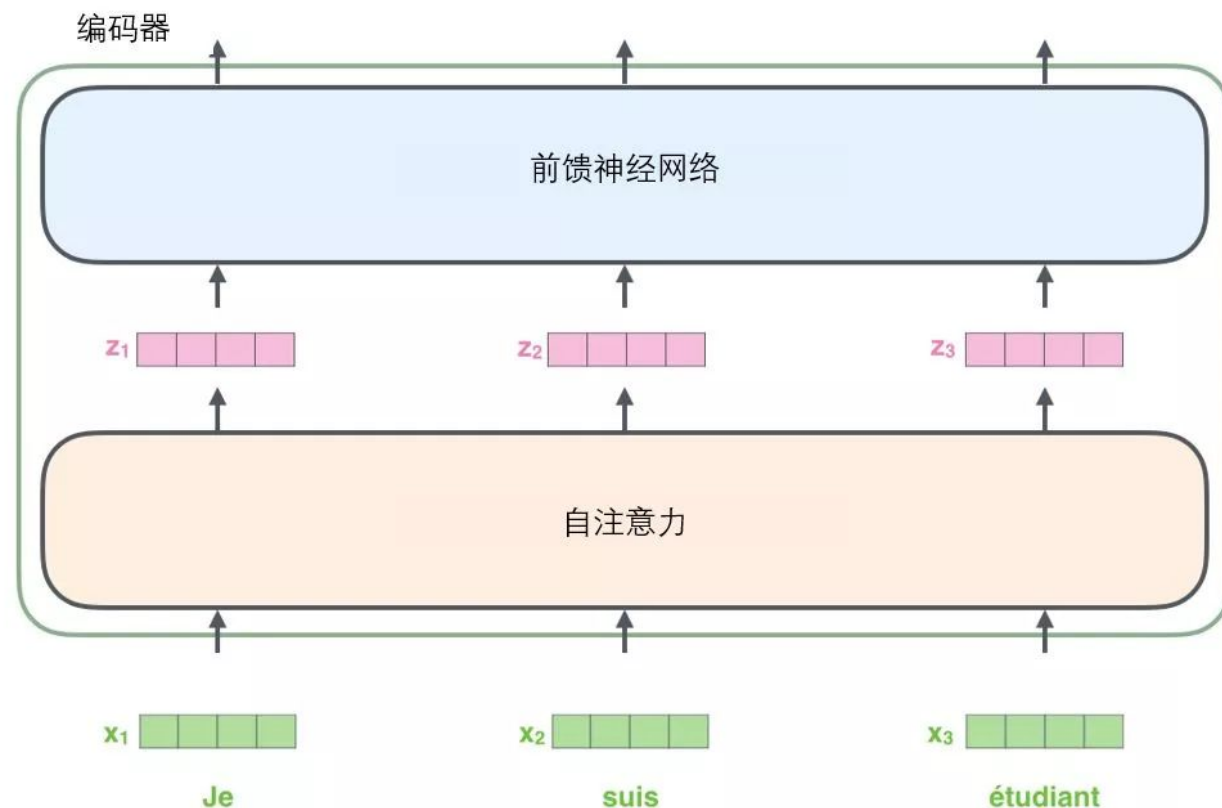
向量列表大小是我们设置的超参数：一般是我们训练集中最长句子的长度。

2.Transformer的工作流程

20

将输入序列进行词嵌入之后，每个单词都会流经编码器中的两个子层。

Transformer的一个核心特性，在这里输入序列中每个位置的单词都有自己独特的路径流入编码器。在自注意力层中，这些路径之间存在依赖关系。而前馈(feed-forward)层没有这些依赖关系。因此在前馈(feed-forward)层时可以并行执行各种路径。

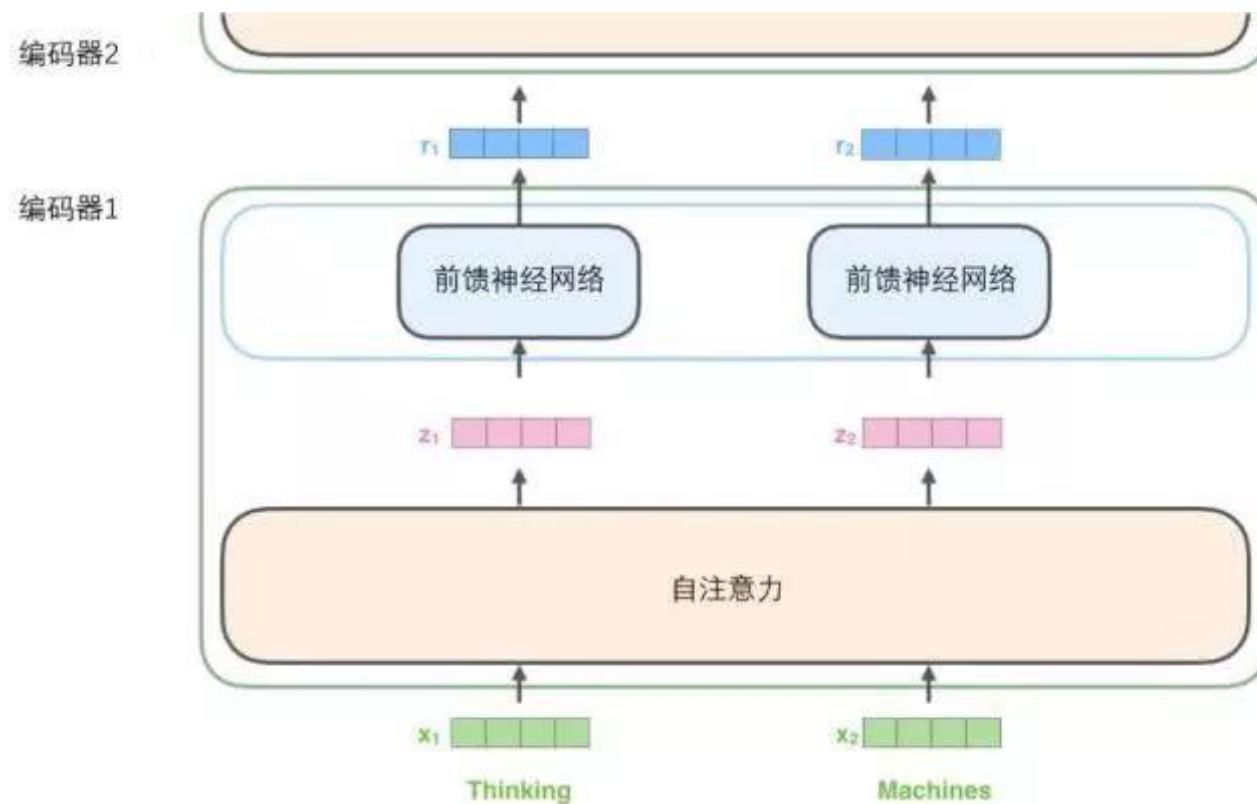


2.Transformer的工作流程

21

编码过程

一个编码器接收向量列表作为输入，接着将向量列表中的向量传递到自注意力层进行处理，然后传递到前馈神经网络层中，将输出结果传递到下一个编码器中。



输入序列的每个单词都经过自编码过程。然后，它们各自通过前向传播神经网络：完全相同的网络，而每个向量都分别通过它。

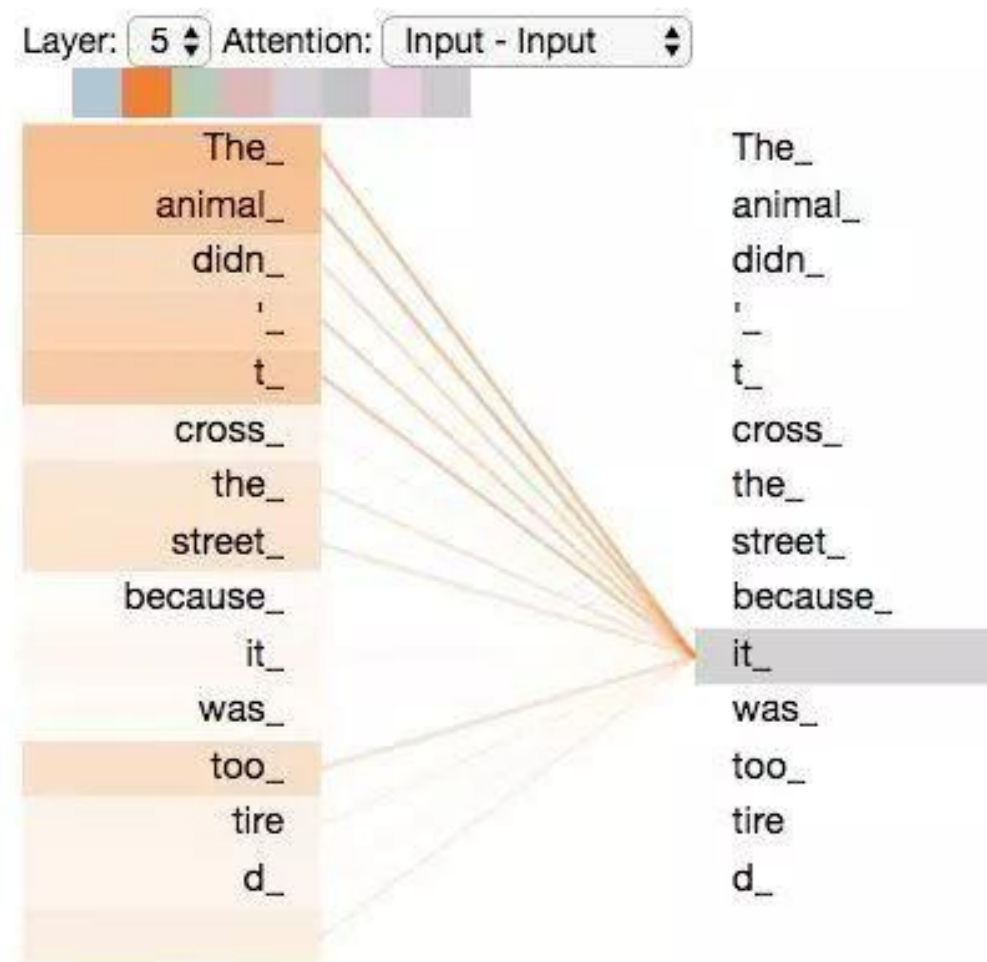
2.Transformer的工作流程

22

从宏观视角看自注意力机制

随着模型处理输入序列的每个单词，自注意力会关注整个输入序列的所有单词，帮助模型对本单词更好地进行编码。

RNN会将它已经处理过的前面的所有单词/向量的表示与它正在处理的当前单词/向量结合起来。而自注意力机制会将所有相关单词的理解融入到我们正在处理的单词中。



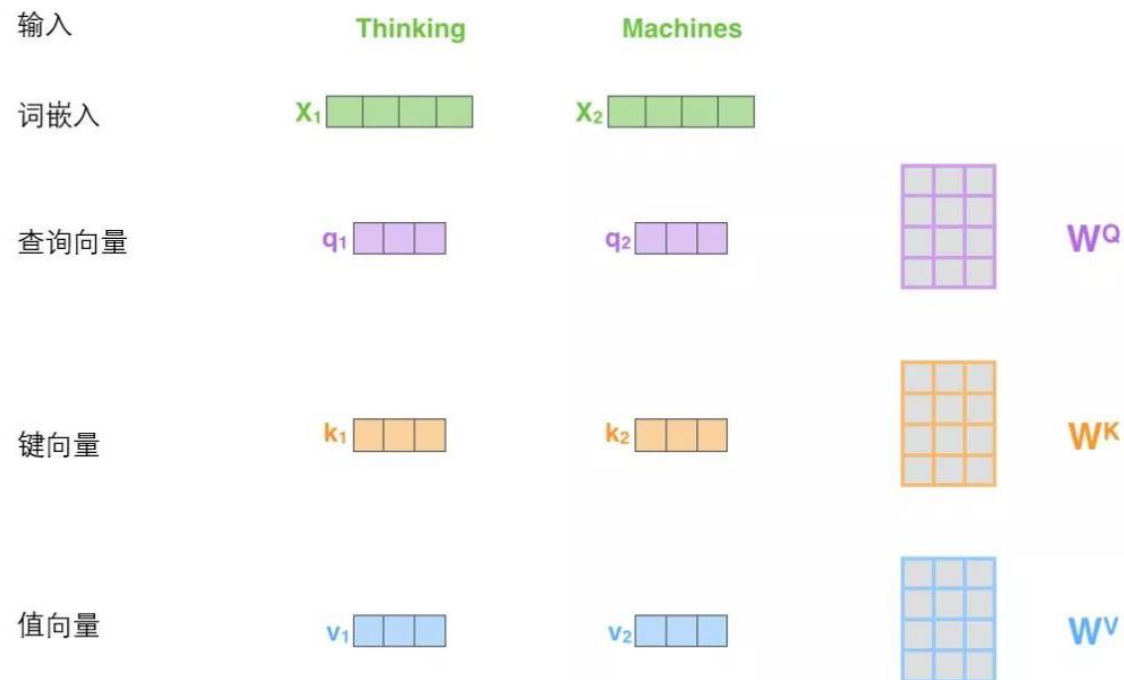
当我们在编码器#5（栈中最上层编码器）中编码“it”这个单词的时，注意力机制的部分会去关注“The Animal”，将它的表示的一部分编入“it”的编码中。

2.Transformer的工作流程

23

从微观视角看自注意力机制

计算自注意力的第一步就是从每个编码器的输入向量（每个单词的词向量）中生成三个向量。也就是说对于每个单词，我们创建一个查询向量(Q)、一个键向量(K)和一个值向量(V)。这三个向量是通过词嵌入与三个权重矩阵后相乘创建的，它们的维度是64，而词嵌入和编码器的输入/输出向量的维度是512。但实际上不强求维度更小，这只是一种基于架构上的选择，它可以使多头注意力（multiheaded attention）的大部分计算保持不变。



x_1 与 W_Q 权重矩阵相乘得到 q_1 ，就是与这个单词相关的查询向量。最终使得输入序列的每个单词的创建一个查询向量Q、一个键向量K和一个值向量V。

2.Transformer的工作流程

24

什么是查询向量Q、键向量k和值向量v?

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

计算得分

分数除以8，然后通过softmax传递结果。

将每个值向量乘以softmax分数(这是为了准备之后将它们求和)。

对加权值向量求和，然后即得到自注意力层在该位置的输出。

输入

词嵌入

查询向量

键向量

值向量

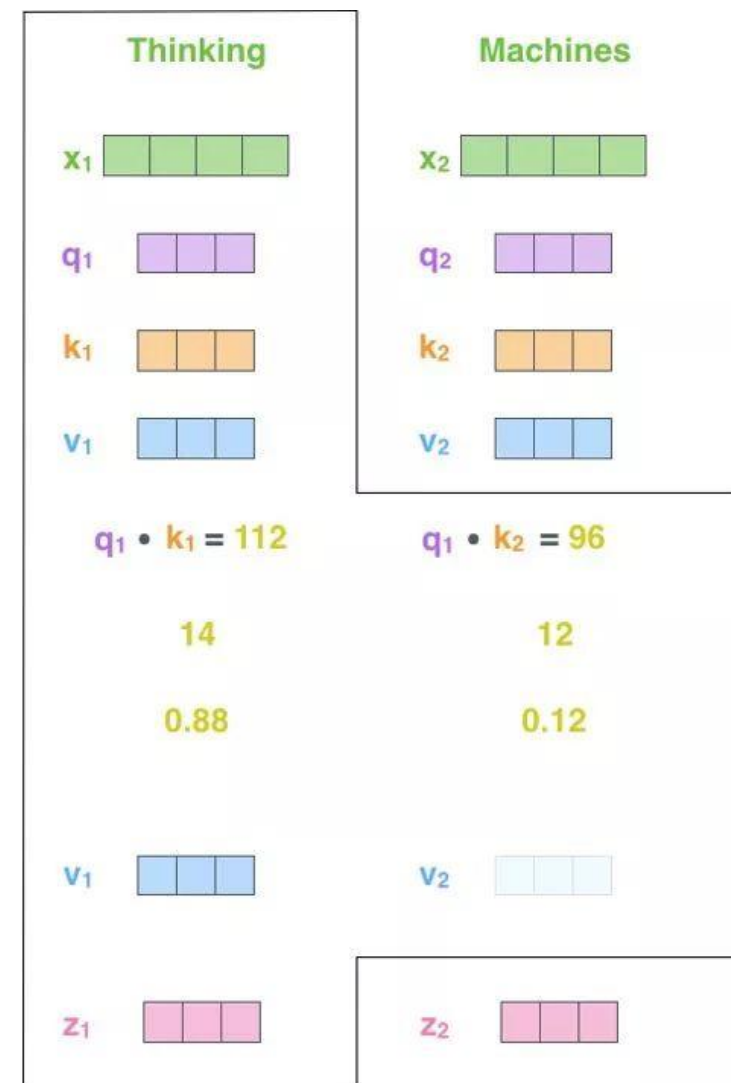
打分

除以8 ($\sqrt{d_k}$)

Softmax

softmax
乘以
值向量

求和



2.Transformer的工作流程

25

通过矩阵运算实现自注意力机制

第一步是计算查询矩阵、键矩阵和值矩阵。为此，我们将输入句子的词嵌入装进矩阵x中，将其乘以我们训练的权重矩阵(W^Q , W^K , W^V)。

x矩阵中的每一行对应于输入句子中的一个单词。我们再次看到词嵌入向量 (512, 或图中的4个格子)和q/k/v向量(64, 或图中的3个格子)的大小差异。

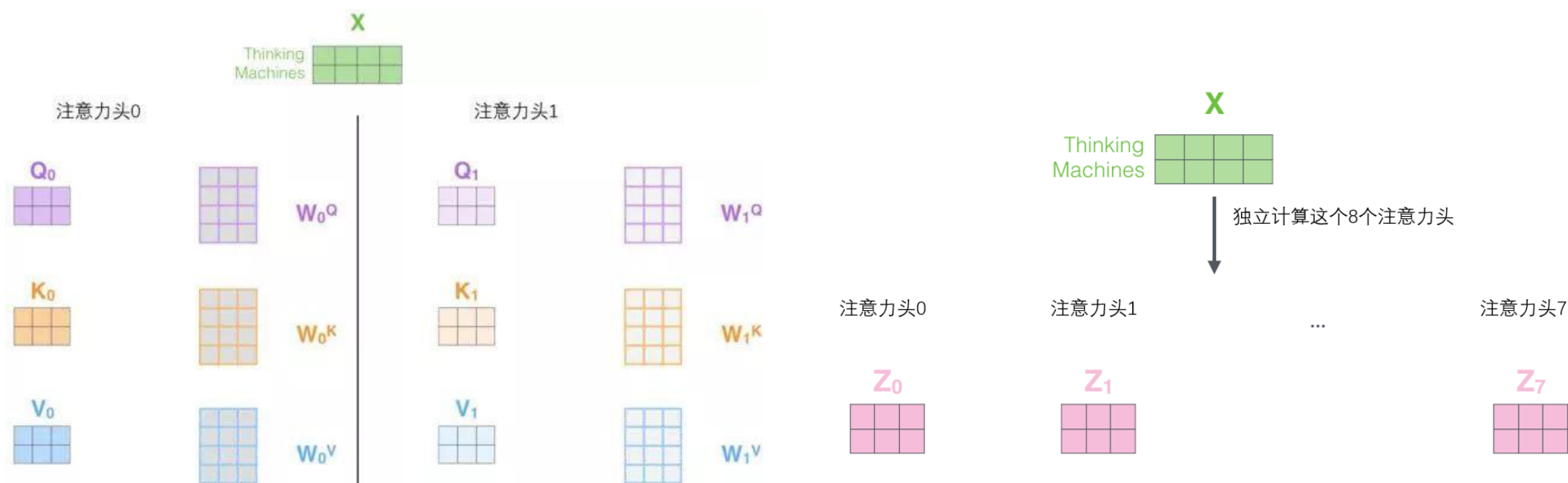
最后，由于我们处理的是矩阵，我们可以用一个公式来计算自注意力层的输出。

$$\begin{aligned} X \times W^Q &= Q \\ X \times W^K &= K \\ X \times W^V &= V \\ \text{softmax} \left(\frac{Q \times K^T}{\sqrt{d_k}} \right) &\times V \\ &= Z \end{aligned}$$

2.Transformer的工作流程

26

“多头”注意力（“multi-headed” attention）的机制



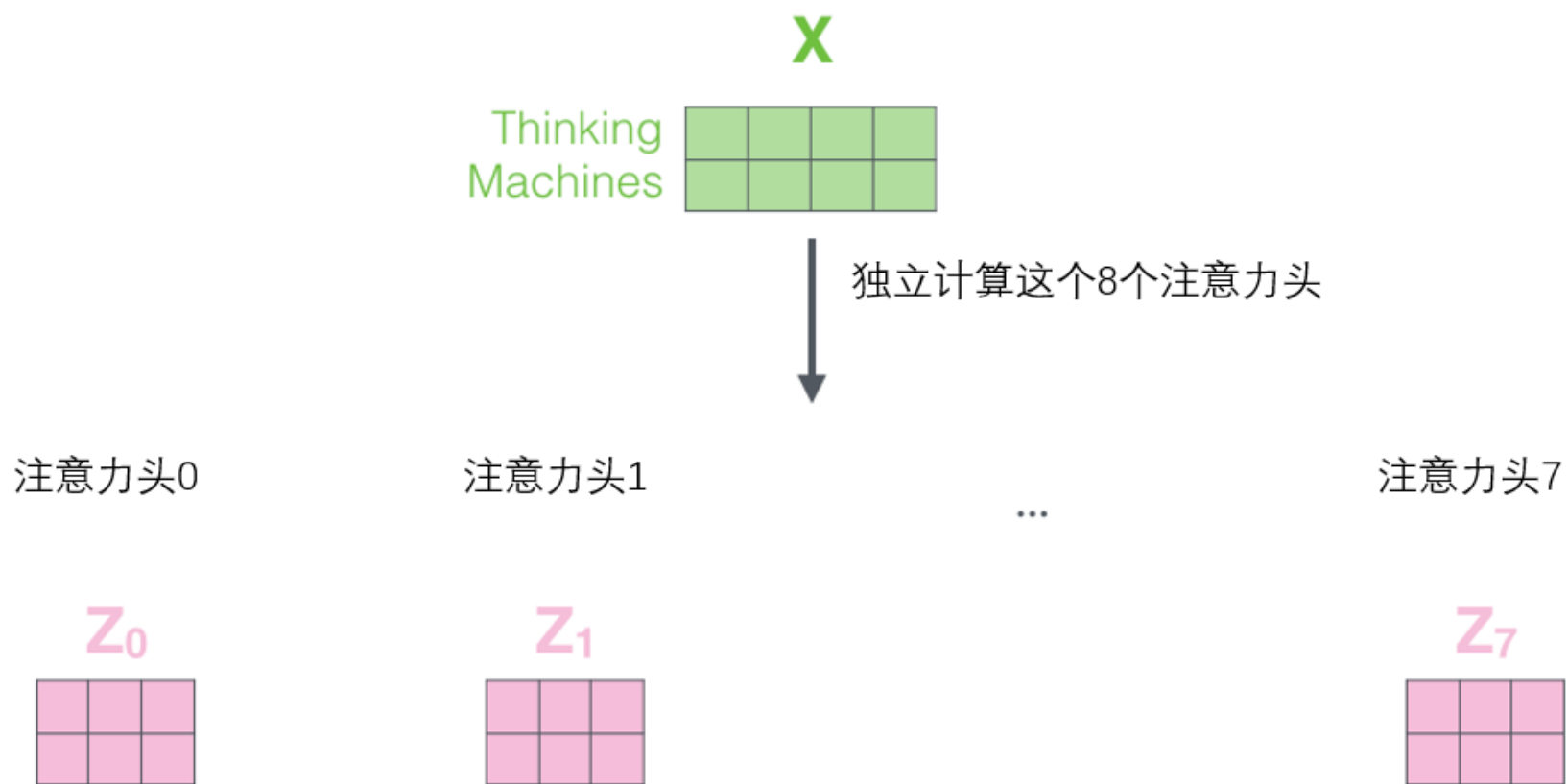
一组Q,K,V得到了一组当前词的特征表达

类似卷积神经网络中的filter提取多种特征？

2.Transformer的工作流程

27

“多头”注意力（“multi-headed” attention）的机制



2.Transformer的工作流程

28

“多头”注意力（“multi-headed” attention）的机制

1) 将所有注意力头拼接起来



2) 乘以矩阵 W^O , 它在模型中是联合训练的

x

3) 结果是一个融合所有注意力头信息的矩阵 Z , 我们可以将其送到前馈神经网络

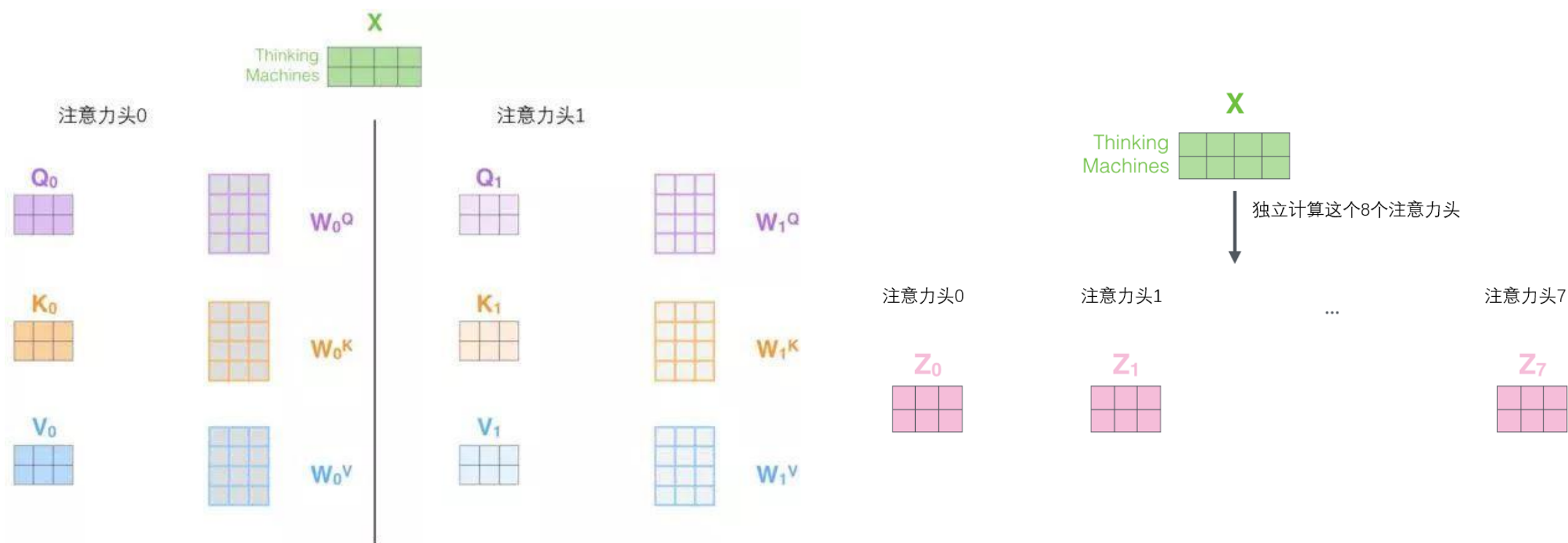


可以看到 Multi-Head Attention 输出的矩阵 Z 与其输入的矩阵 x 的维度是一样的。

2.Transformer的工作流程

29

“多头” 注意力 (“multi-headed” attention) 的机制



2.Transformer的工作流程

30

1) 这是我们的输入句子*

2) 编码每一个单词

3) 将其分为8个头，将矩阵X或R乘以各个权重矩阵

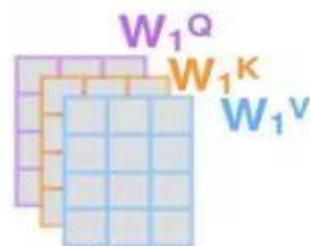
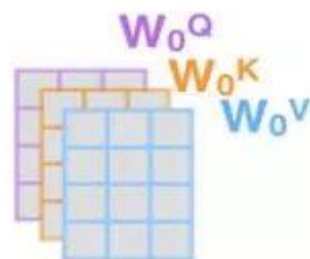
4) 通过输出的查询/键/值 (Q/K/V) 矩阵计算注意力

5) 将所有注意力头拼接起来，乘以权重矩阵 W^O

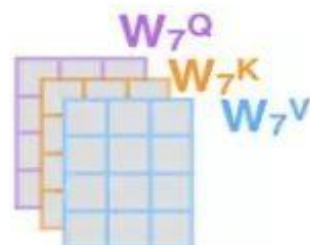
Thinking
Machines



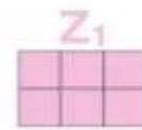
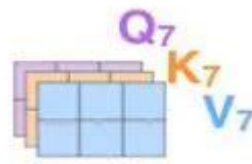
*注：除了第0个编码器，其他编码器都不需要进行词嵌入。它可以直接讲前面一层编码器的输出作为输入（矩阵R）。



...



...



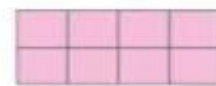
...



W^O



Z

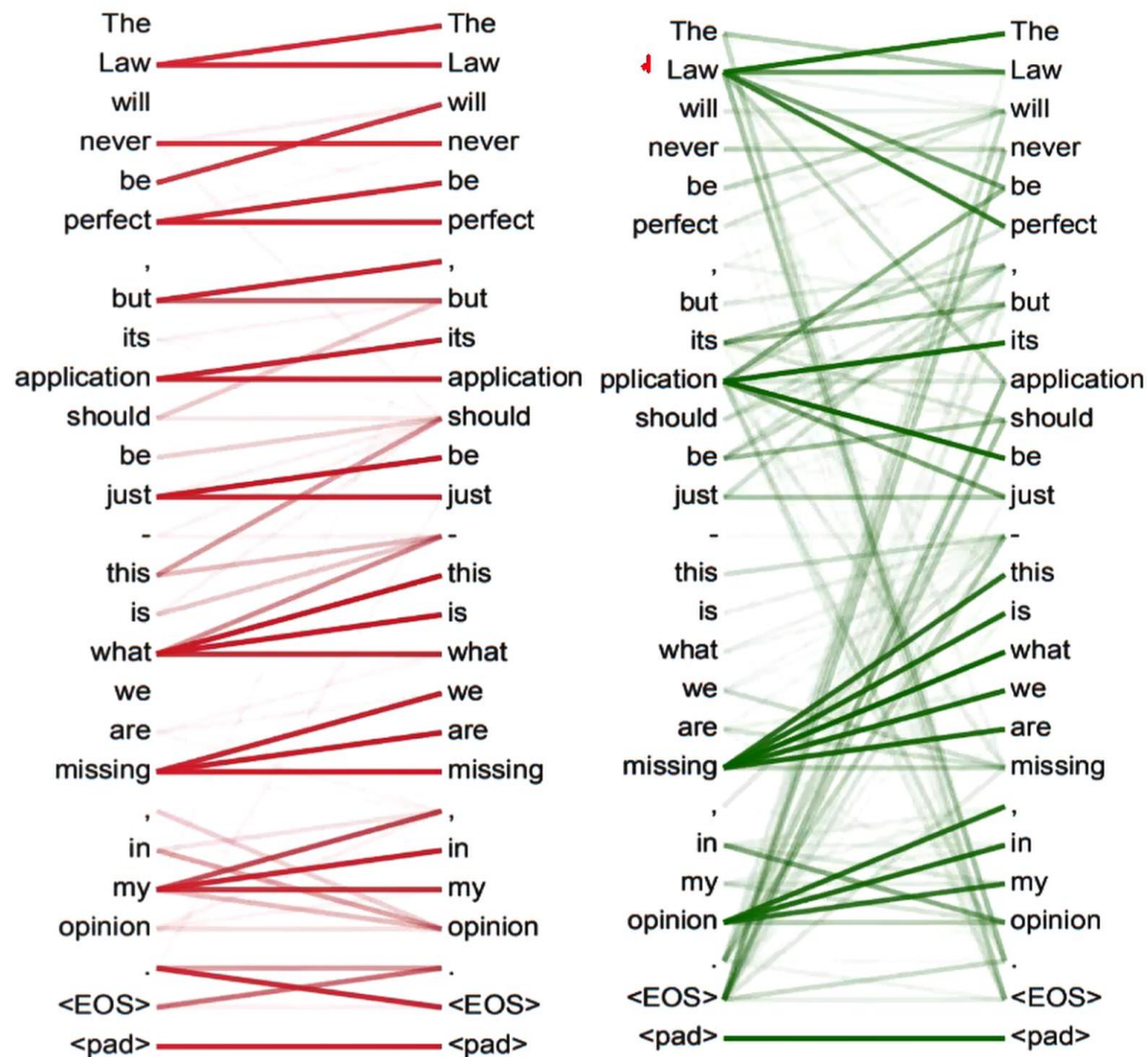


2.Transformer的工作流程

31

multi-headed结果

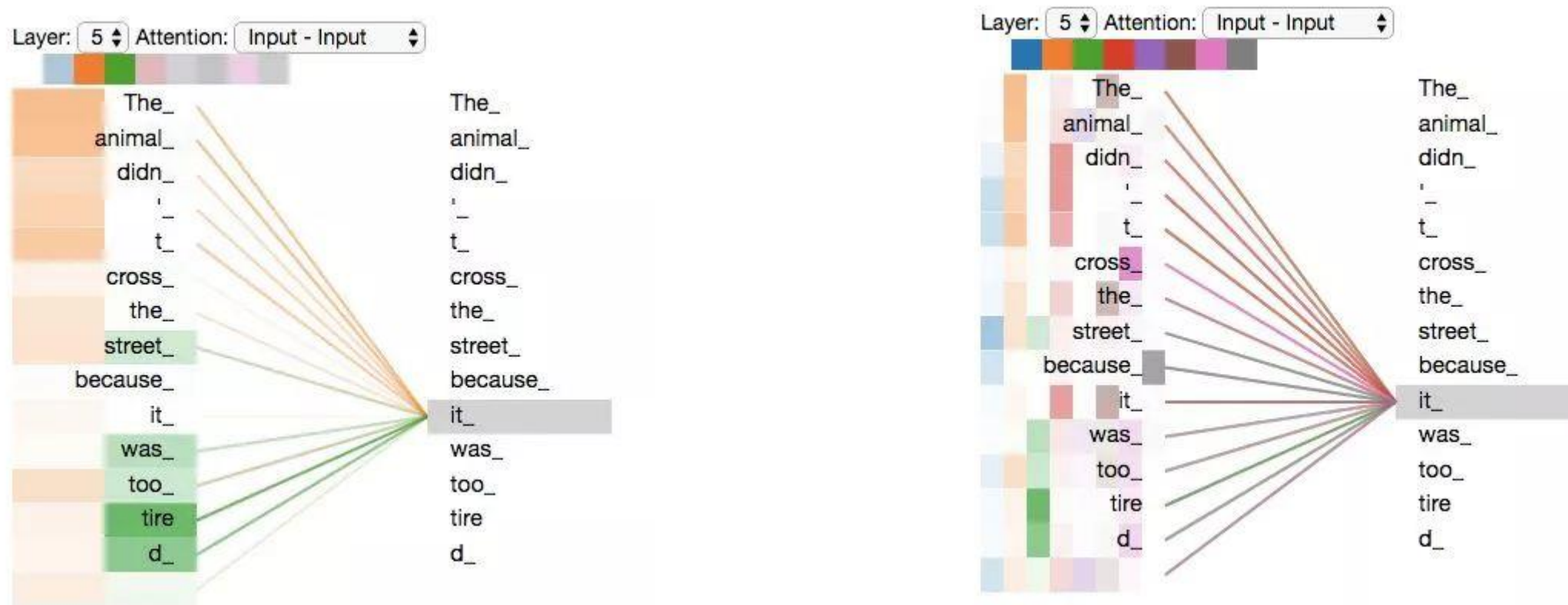
- 不同的注意力结果
- 得到的特征向量表达也不相同



2.Transformer的工作流程

32

当我们编码 “it”一词时，一个注意力头集中在 “animal”上，而另一个则集中在 “tired”上，从某种意义上说，模型对 “it”一词的表达在某种程度上是 “animal”和 “tired”的代表。然而，如果我们把所有的attention都加到图示里，事情就更难解释了：



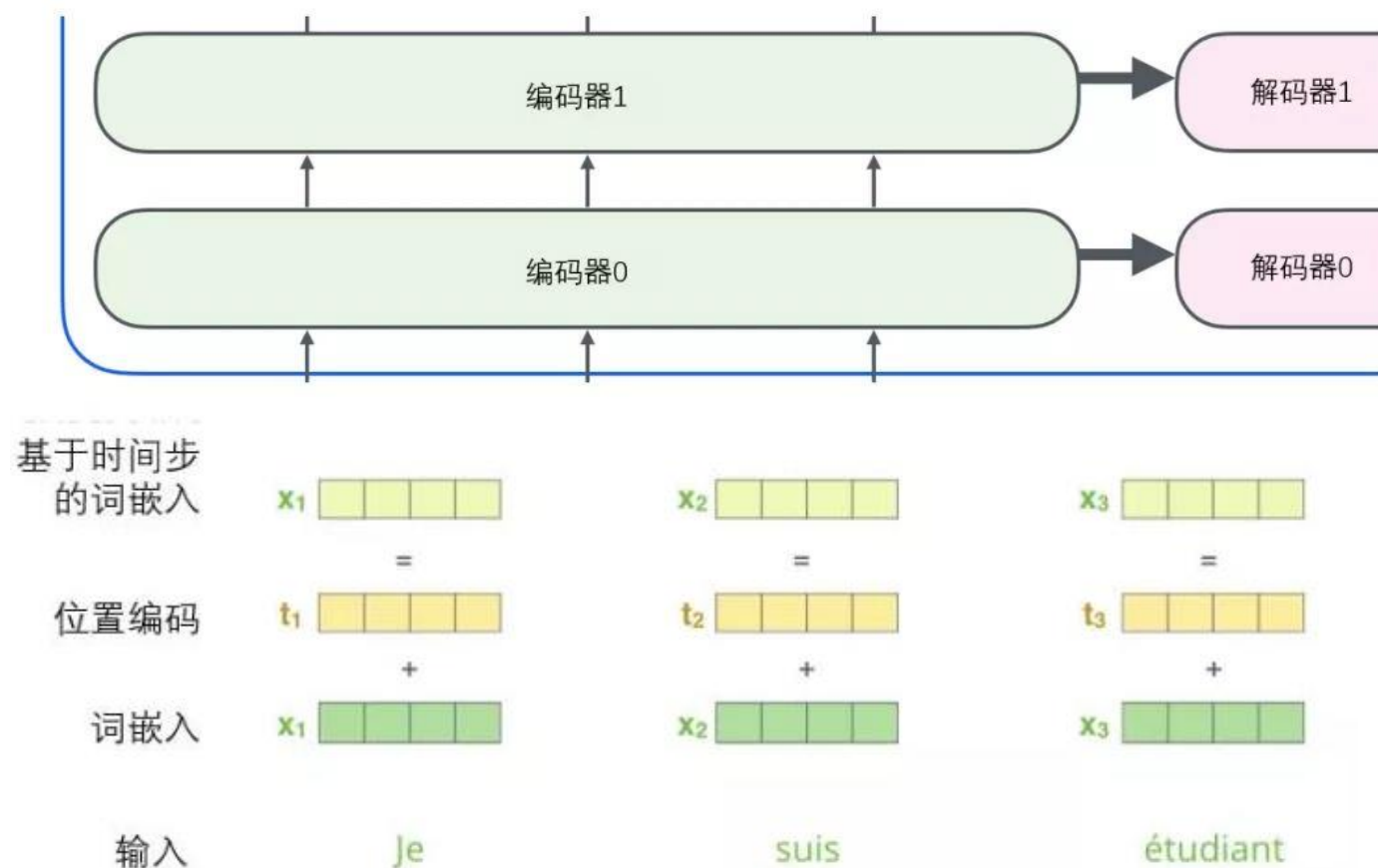
2.Transformer的工作流程

33

使用位置编码表示序列的顺序

到目前为止，我们对模型的描述缺少了一种理解输入单词顺序的方法。

为了解决这个问题，Transformer为每个输入的词嵌入添加了一个向量。这些向量遵循模型学习到的特定模式，这有助于确定每个单词的位置，或序列中不同单词之间的距离。这里的直觉是，将位置向量添加到词嵌入中使得它们在接下来的运算中，能够更好地表达的词与词之间的距离。

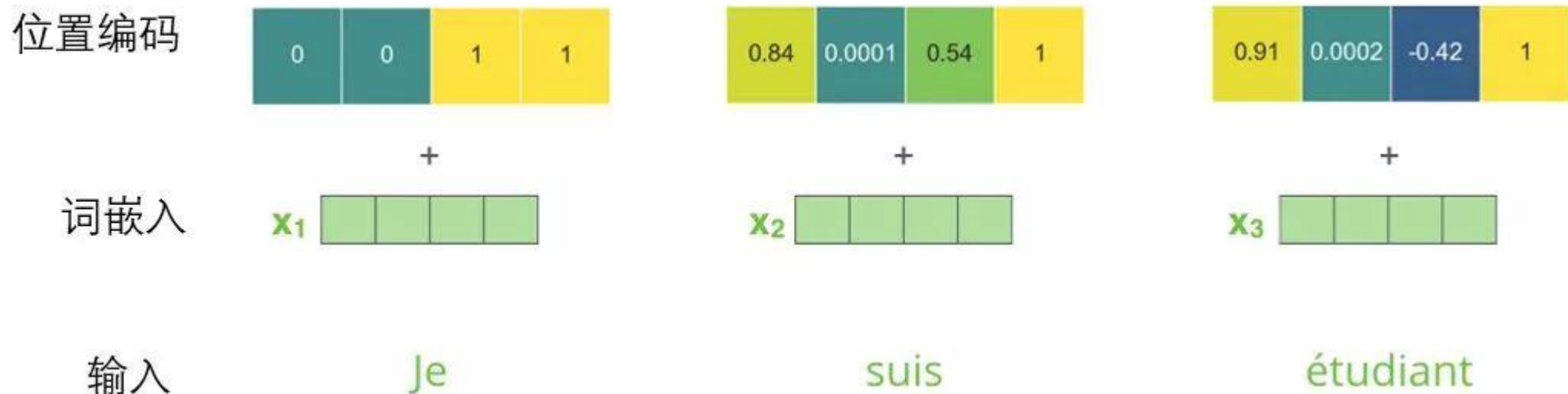


2.Transformer的工作流程

34

为了让模型理解单词的顺序，我们添加了位置编码向量，这些向量的值遵循特定的模式。

如果我们假设词嵌入的维数为4，则实际的位置编码如下：

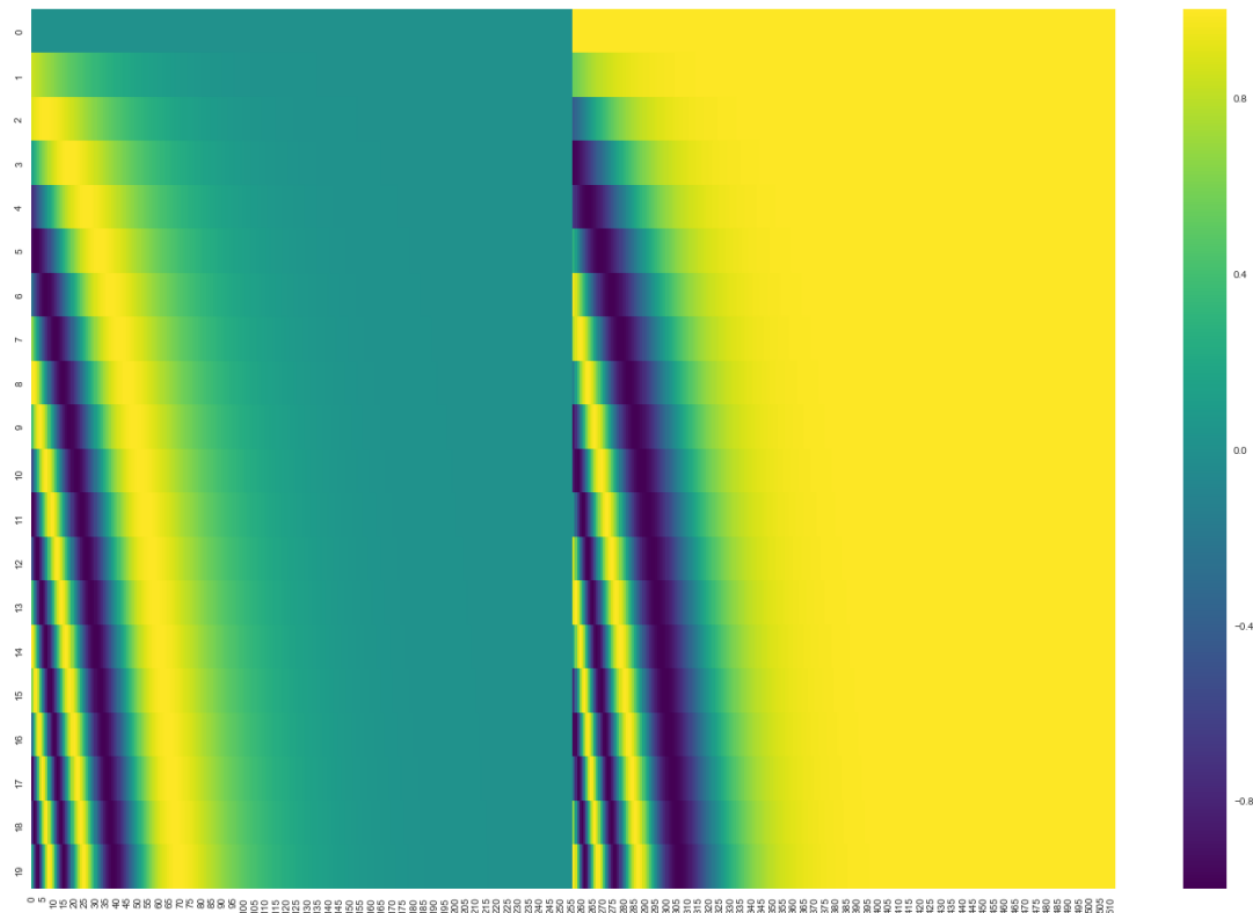


2.Transformer的工作流程

35

20字(行)的位置编码实例，词嵌入大小为512(列)。你可以看到它从中间分裂成两半。这是因为左半部分的值由一个函数(使用正弦)生成，而右半部分由另一个函数(使用余弦)生成。然后将它们拼在一起而得到每一个位置编码向量。

这个编码函数的可视化结果：

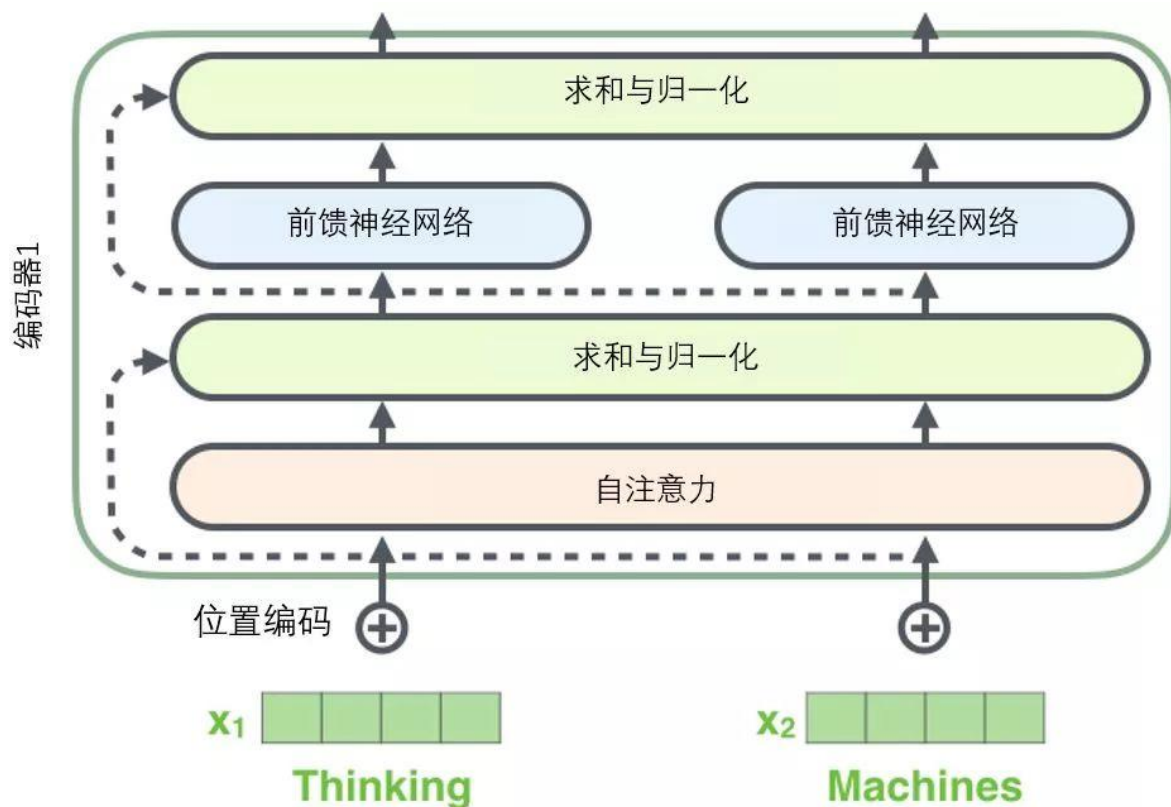


图中，每一行对应一个词向量的位置编码，所以第一行对应着输入序列的第一个词。每行包含512个值，每个值介于1和-1之间。我们已经对它们进行了颜色编码，所以图案是可见的。

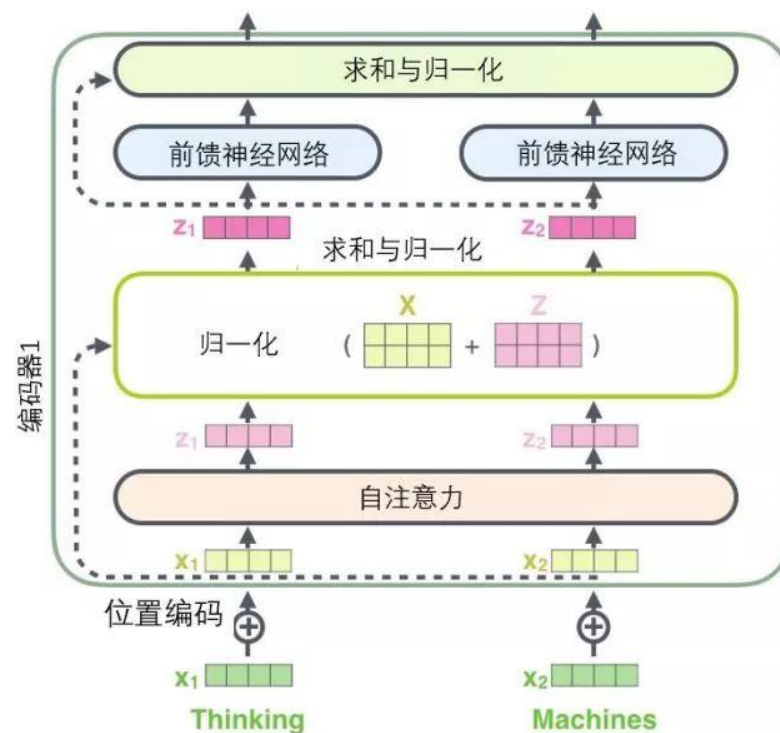
2.Transformer的工作流程

36

在每个编码器中的每个子层（自注意力、前馈网络）的周围都有一个残差连接，并且都跟随着一个“层-归一化”步骤。



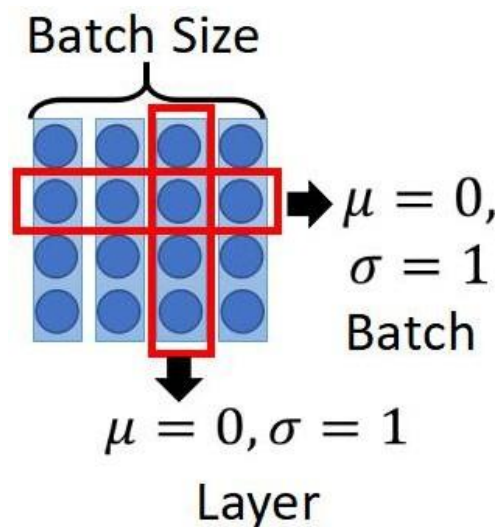
如果我们去可视化这些向量以及这个和自注意力相关联的层-归一化操作，那么看起来就像下面这张图描述一样：



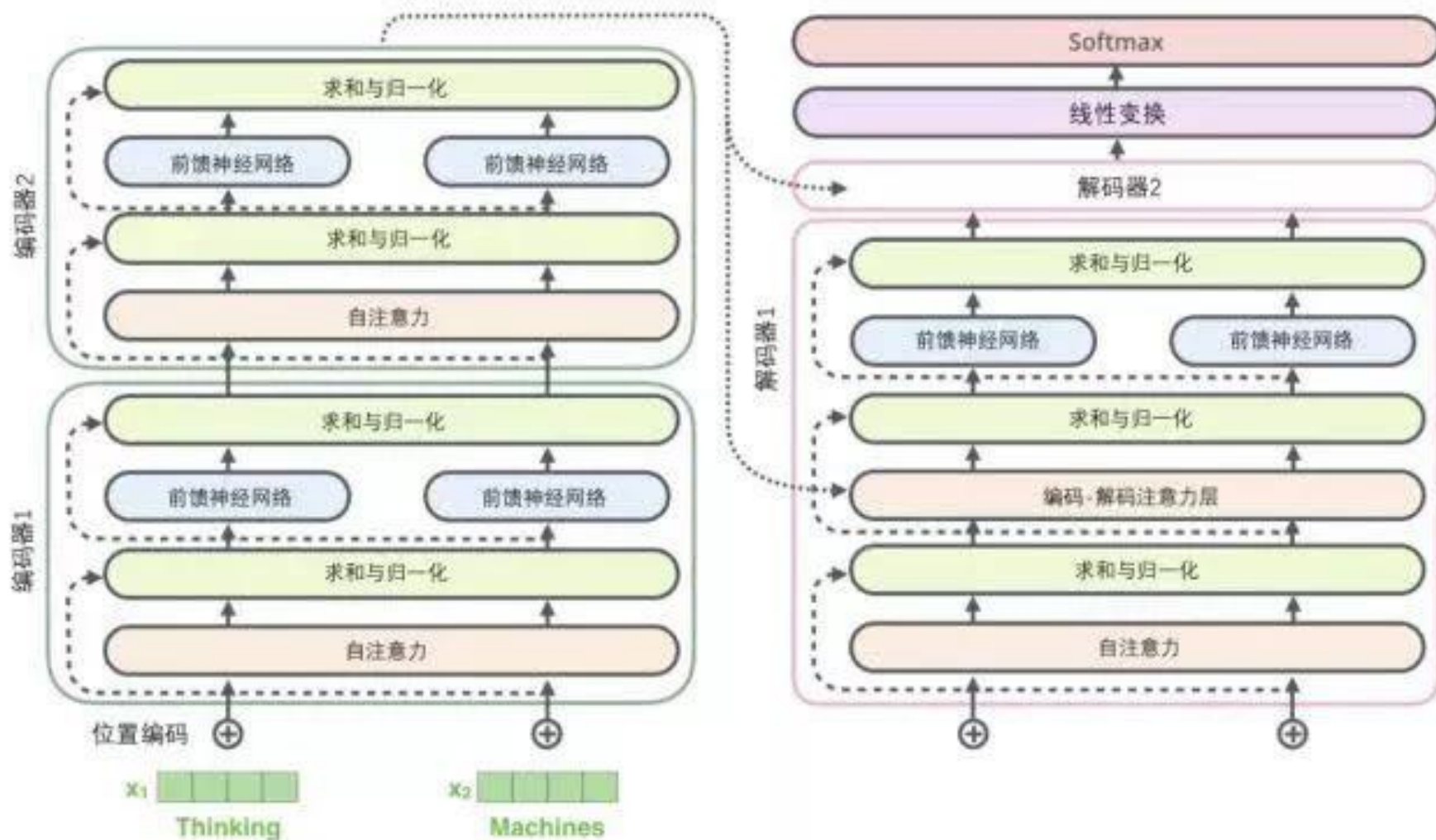
2.Transformer的工作流程

37

归一化:



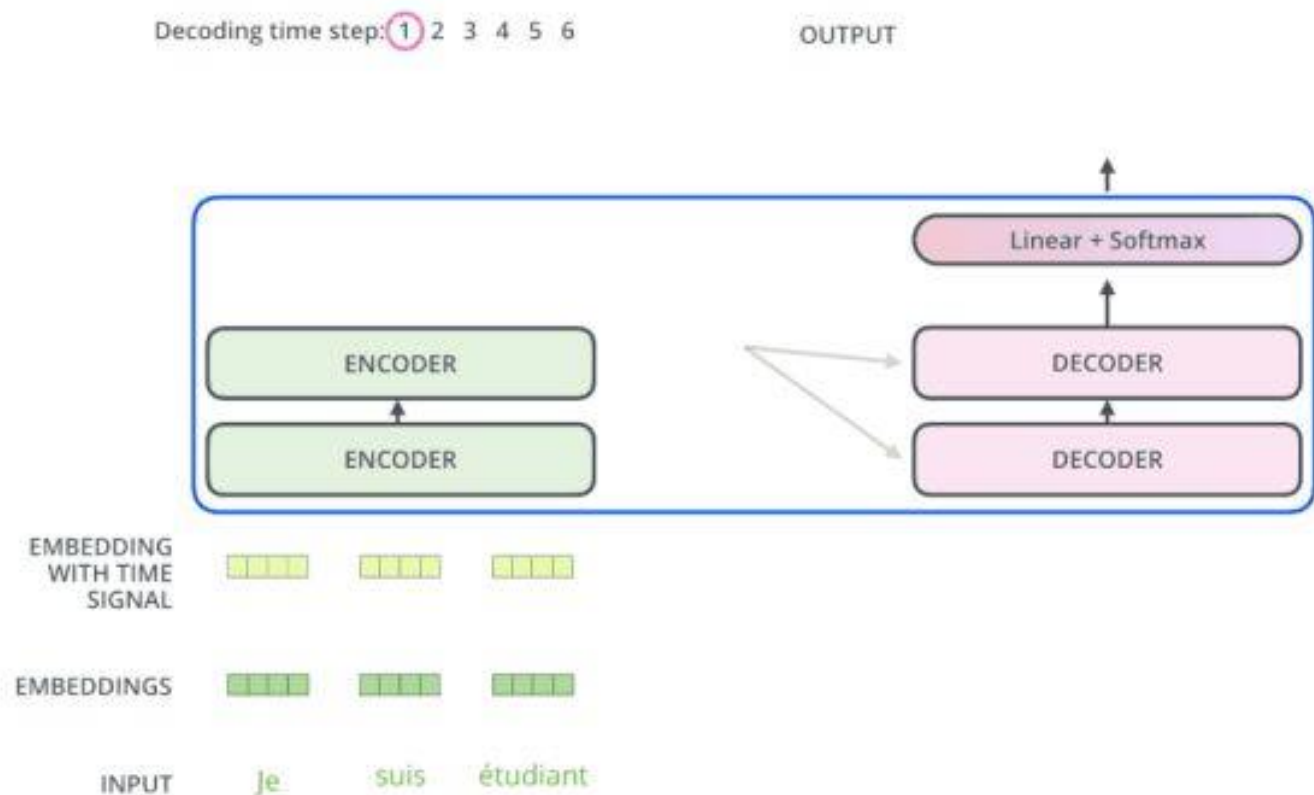
连接: 基本的残差
连接方式



2.Transformer的工作流程

38

编码器通过处理输入序列开启工作。顶端编码器的输出之后会变转化为一个包含向量K（键向量）和V（值向量）的注意力向量集。这些向量将被每个解码器用于自身的“编码-解码注意力层”，而这些层可以帮助解码器关注输入序列哪些位置合适：



在完成编码阶段后，则开始解码阶段。解码阶段的每个步骤都会输出一个输出序列（在这个例子里，是英语翻译的句子）的元素

2.Transformer的工作流程

39

最终的线性变换和Softmax层

解码组件最后会输出一个实数向量。我们如何把浮点数变成一个单词？这便是线性变换层要做的工作，它之后就是Softmax层。

线性变换层是一个简单的全连接神经网络，它可以把解码组件产生的向量投射到一个比它大得多的、被称作对数几率（logits）的向量里。

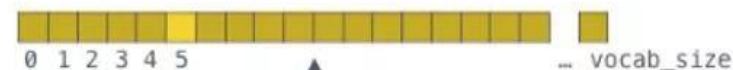
不妨假设我们的模型从训练集中学习一万个不同的英语单词（我们模型的“输出词表”）。因此对数几率向量为一万个单元格长度的向量——每个单元格对应某一个单词的分数。

接下来的Softmax 层便会把那些分数变成概率（都为正数、上限1.0）。概率最高的单元格被选中，并且它对应的单词被作为这个时间步的输出。

查找词表中对应索引的单词

获取最高概率的索引
(argmax)

log_probs



am

5

Softmax

logits



Linear

解码组件输出



这张图片从底部以解码器组件产生的输出向量开始。之后它会转化出一个输出单词。

3.Transformer的训练

40

01 Transformer介绍

02 Transformer的工作流程

03 Transformer的训练

04 BERT

3.Transformer的训练

41

输出词汇

eos: end of sentence的缩写形式



WORD	a	am	I	thanks	student	<eos>
INDEX	0	1	2	3	4	5

我们模型的输出词表在我们训练之前的预处理流程中就被设定好。

3.Transformer的训练

42

一旦我们定义了我们的输出词表，我们可以使用一个相同宽度的向量来表示我们词汇表中的每一个单词。这也被认为是一个one-hot 编码。所以，我们可以用下面这个向量来表示单词 “am”：

输出词表

WORD	a	am	I	thanks	student	<eos>
INDEX	0	1	2	3	4	5

单词“am”的one-hot编码

0.0	1.0	0.0	0.0	0.0	0.0
-----	-----	-----	-----	-----	-----

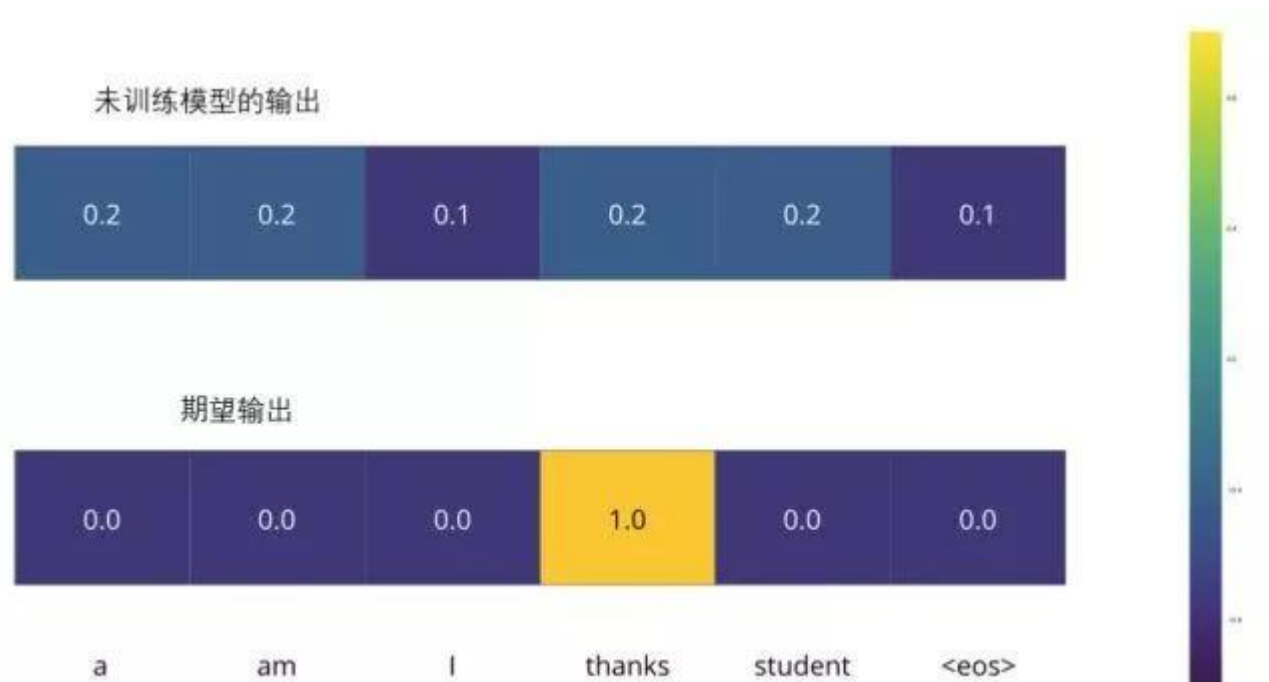
3.Transformer的训练

43

损失函数

比如说我们正在训练模型，现在是第一步，一个简单的例子——把 “merci” 翻译为 “thanks”。

这意味着我们想要一个表示单词 “thanks” 概率分布的输出。但是因为这个模型还没被训练好，所以不太可能现在就出现这个结果。

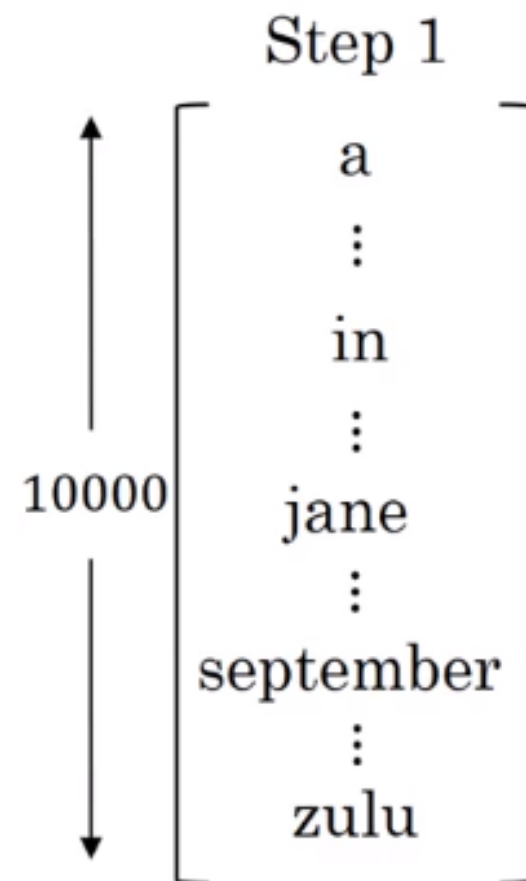


3.Transformer的训练

44

集束搜索(Beam Search)

贪婪算法只会挑出最可能的那一个单词，然后继续。而集束搜索则会考虑多个选择，集束搜索算法会有一个参数 B ，叫做集束宽 (beam width)。在这个例子中 $B=3$ ，这样就意味着集束搜索不会只考虑一个可能结果，而是一次会考虑3个，比如对第一个单词有不同选择的可能性，最后找到in、jane、september，是英语输出的第一个单词的最可能的三个选项，然后集束搜索算法会把结果存到计算机内存里以便后面尝试用这三个词。



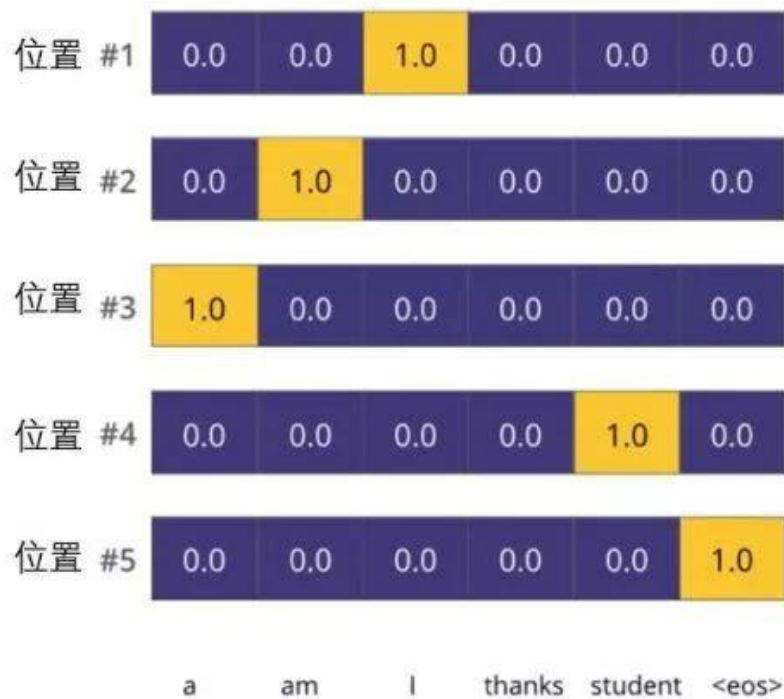
"Jane visite l'Afrique en Septembre." (法语句子)，我们希望翻译成英语，**"Jane is visiting Africa in September"**. (英语句子)

3.Transformer的训练

45

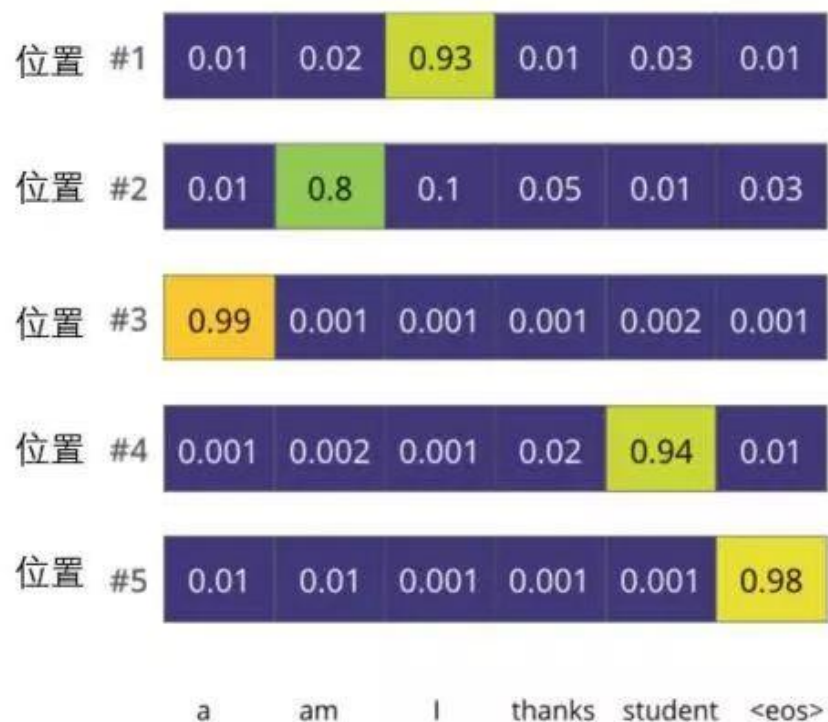
模型的目标输出

输出词表： a am I thanks student <eos>



模型训练后的输出

输出词表： a am I thanks student <eos>



依据例子训练模型得到的目标概率分布

3.Transformer的训练

46

Transformer 总结

- Transformer 与 RNN 不同，可以比较好地并行训练。
- Transformer 本身是不能利用单词的顺序信息的，因此需要在输入中添加位置 Embedding，否则 Transformer 就是一个词袋模型了。
- Transformer 的重点是 Self-Attention 结构，其中用到的 Q, K, V 矩阵通过输出进行线性变换得到。
- Transformer 中 Multi-Head Attention 中有多个 Self-Attention，可以捕获单词之间多种维度上的相关系数 attention 分数。

4.BERT

47

01 Transformer介绍

02 Transformer的工作流程

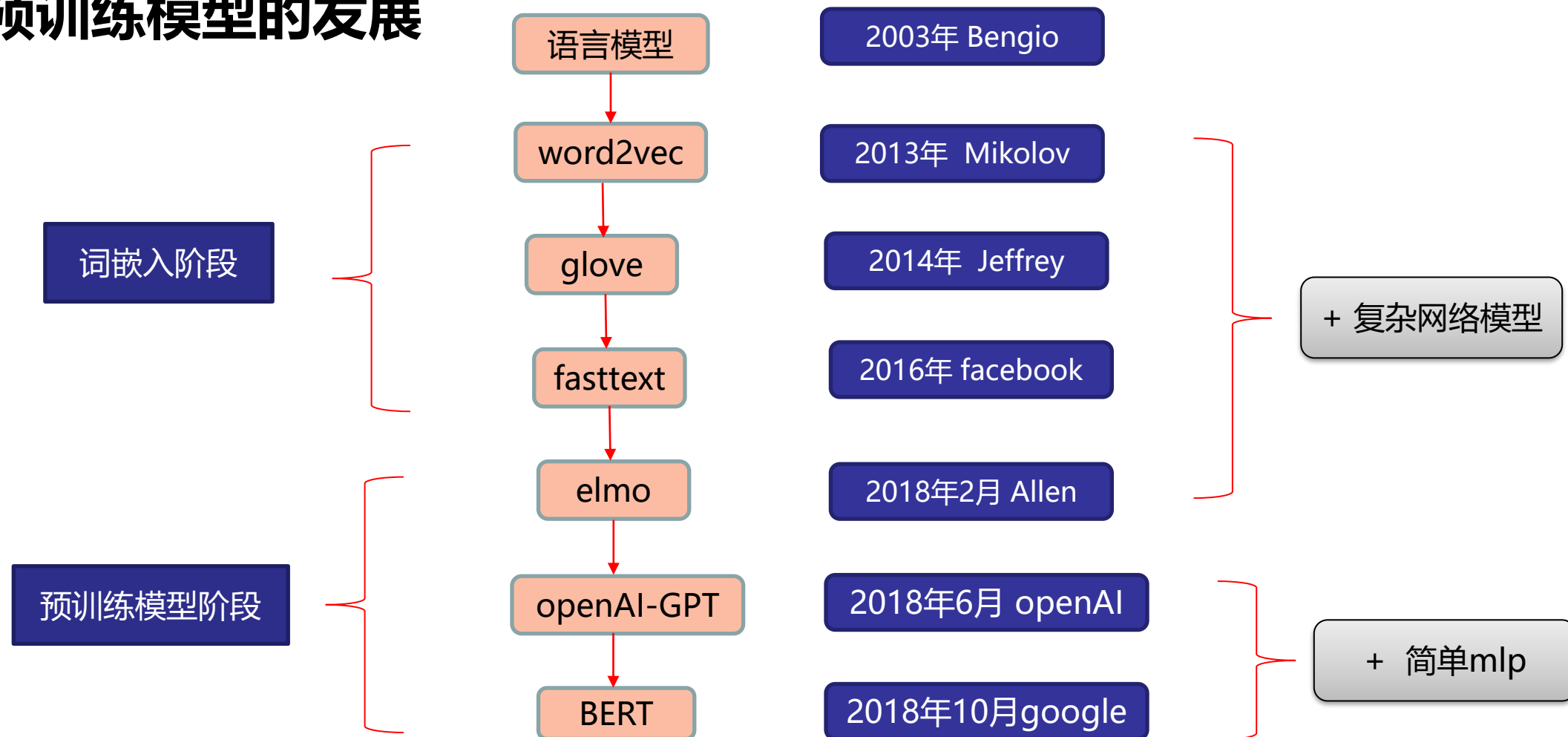
03 Transformer的训练

04 BERT

4.BERT

48

NLP 预训练模型的发展



4.BERT

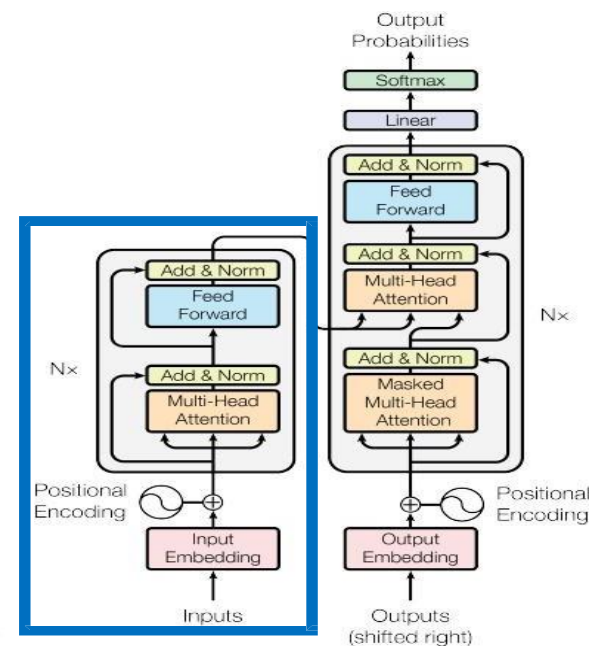
49

BERT训练的词向量有什么不同？

在word2vec中，相同词对应的向量训练好后就固定了

但在不同的场景中，‘干什么’的意思会相同吗？

这俩兄弟都叫transformer:



4.BERT

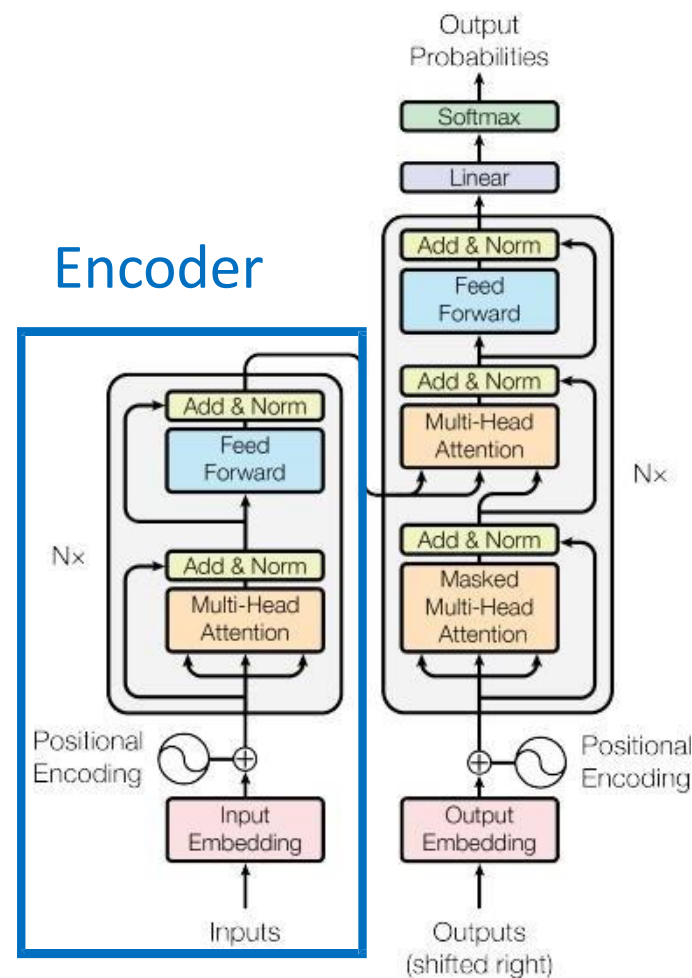
50

Bidirectional Encoder Representations from Transformers

说白了就是transformer的encoder部分

并不需要标签，有语料就能训练了

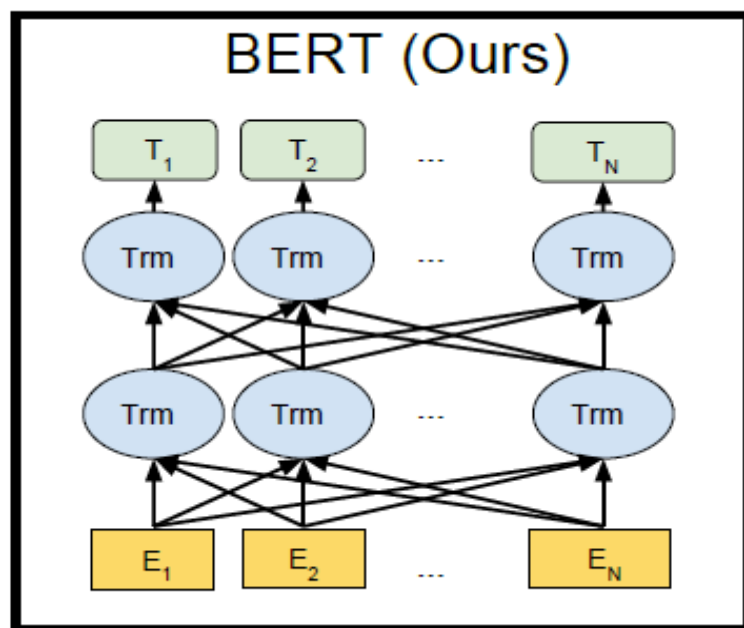
BERT是一个算法模型，它的出现打破了大量的自然语言处理任务的记录。在BERT的论文发布不久后，Google的研发团队还开放了该模型的代码，并提供了一些在大量数据集上预训练好的算法模型下载方式，这使得所有人都可以通过它来构建一个涉及NLP的算法模型，节约了大量训练语言模型所需的时间，精力，知识和资源



4.BERT

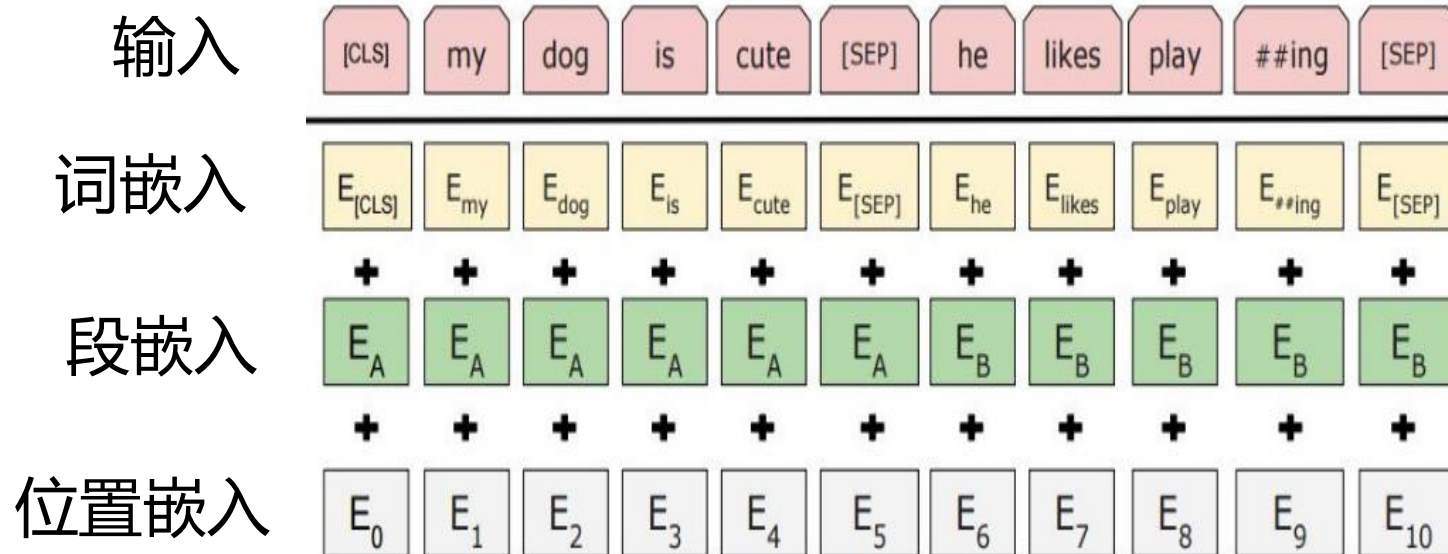
51

BERT—模型结构



特点:

1. 完全的双向, 每一层都是同时关乎上下文
2. transformer 可以对长句子有更强的特征抽取的能力

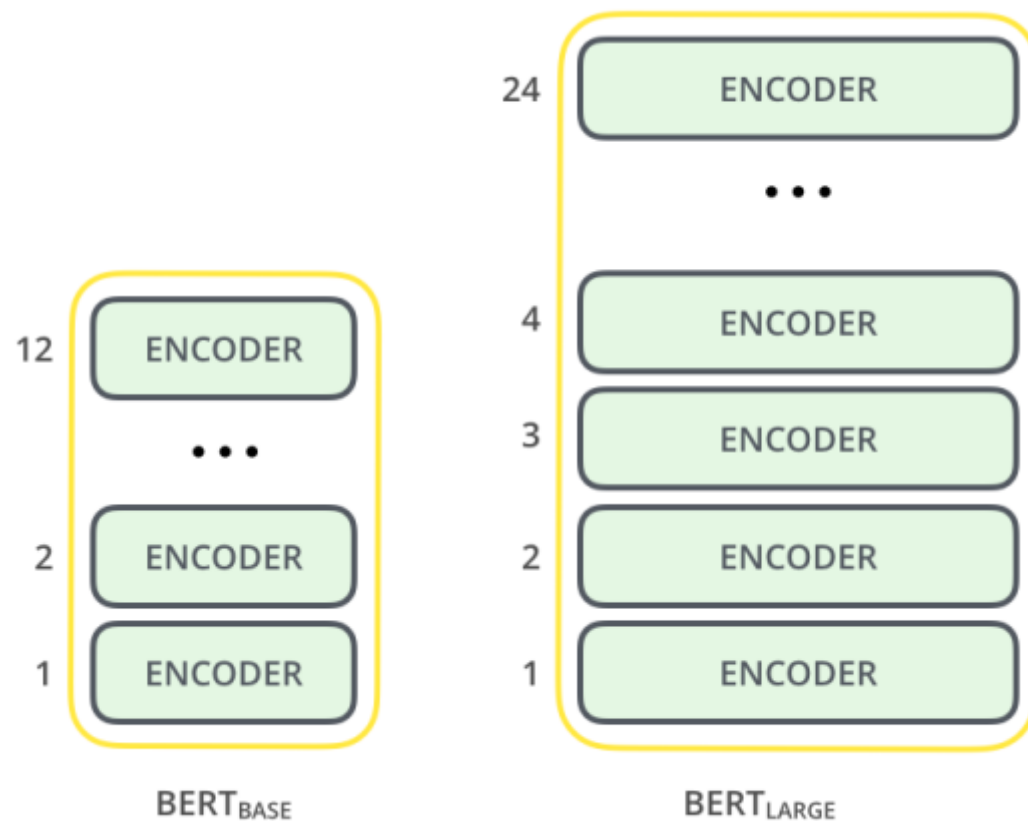


4.BERT

52

BERT—模型结构

2个BERT的模型都有一个很大的编码器层数，（论文里面将此称为Transformer Blocks） - 基础版本就有12层，进阶版本有24层。同时它也有很大的前馈神经网络（768和1024个隐藏层神经元），还有很多attention heads（12-16个）。这超过了Transformer论文中的参考配置参数（6个编码器层，512个隐藏层单元，和8个注意头）



4.BERT

53

如何训练BERT

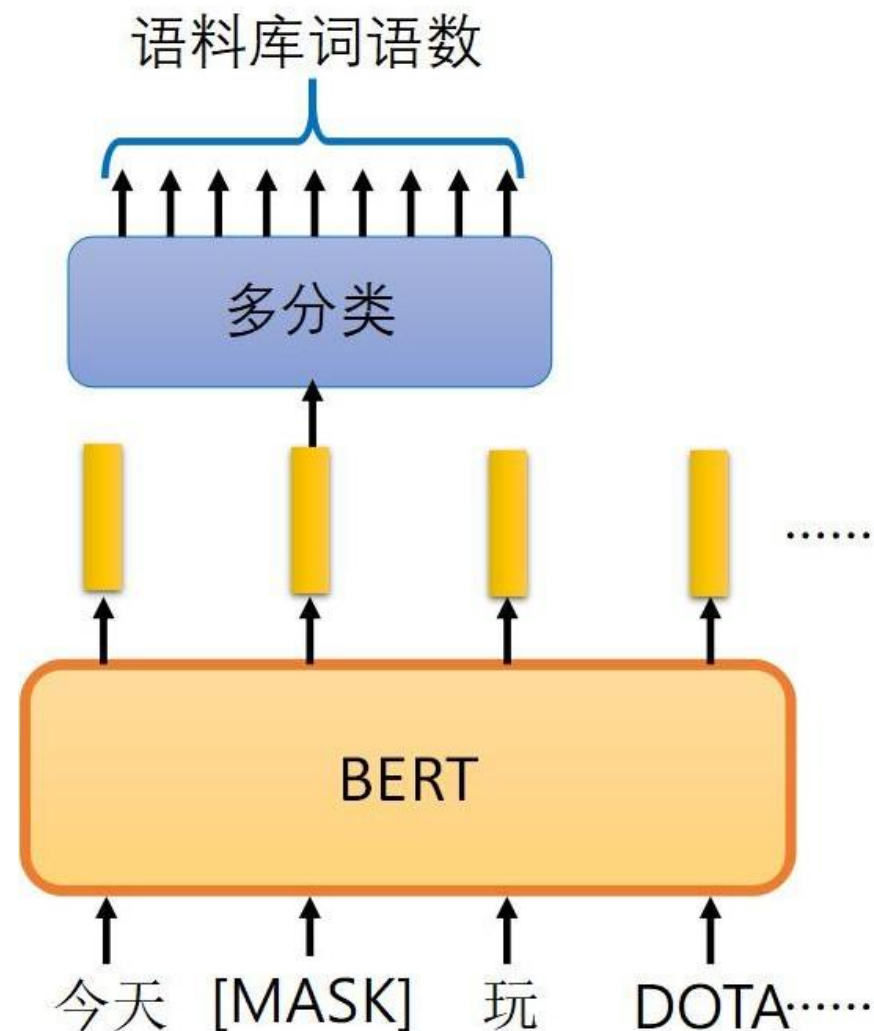
方法1: MLM (Masked Language Modeling)

当前词出现不只是单单依靠上文或者下文，其实应该是同时依赖于上下文深层的双向RNN会互相透露信息。

句子中有15%的词汇被随机mask掉 交给模型去预测被mask的部分到底是什么

词语的可能性太多了，中文一般是字

如果BERT训练的向量好，那分类自然OK



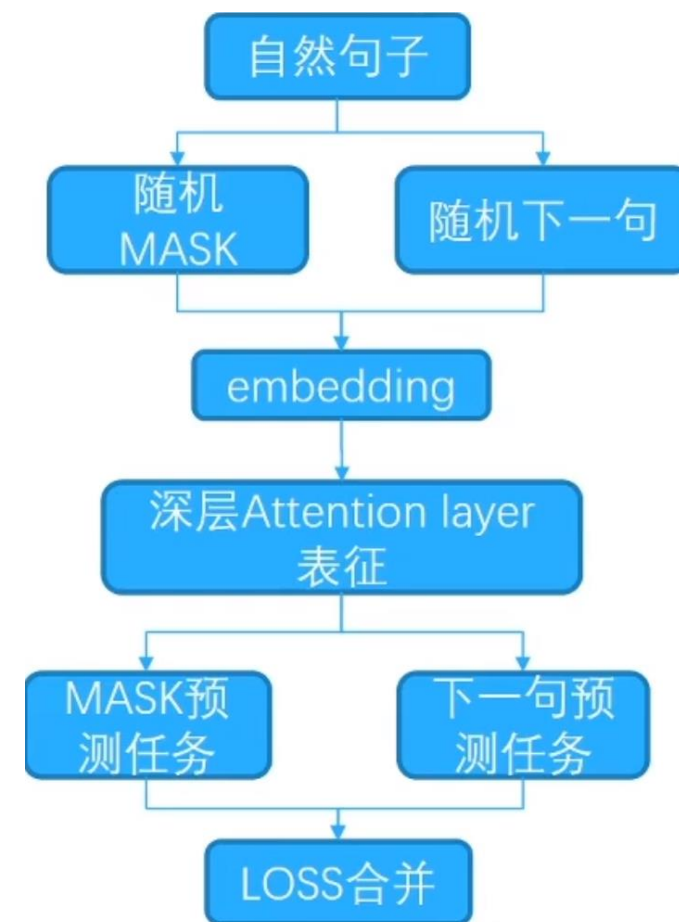
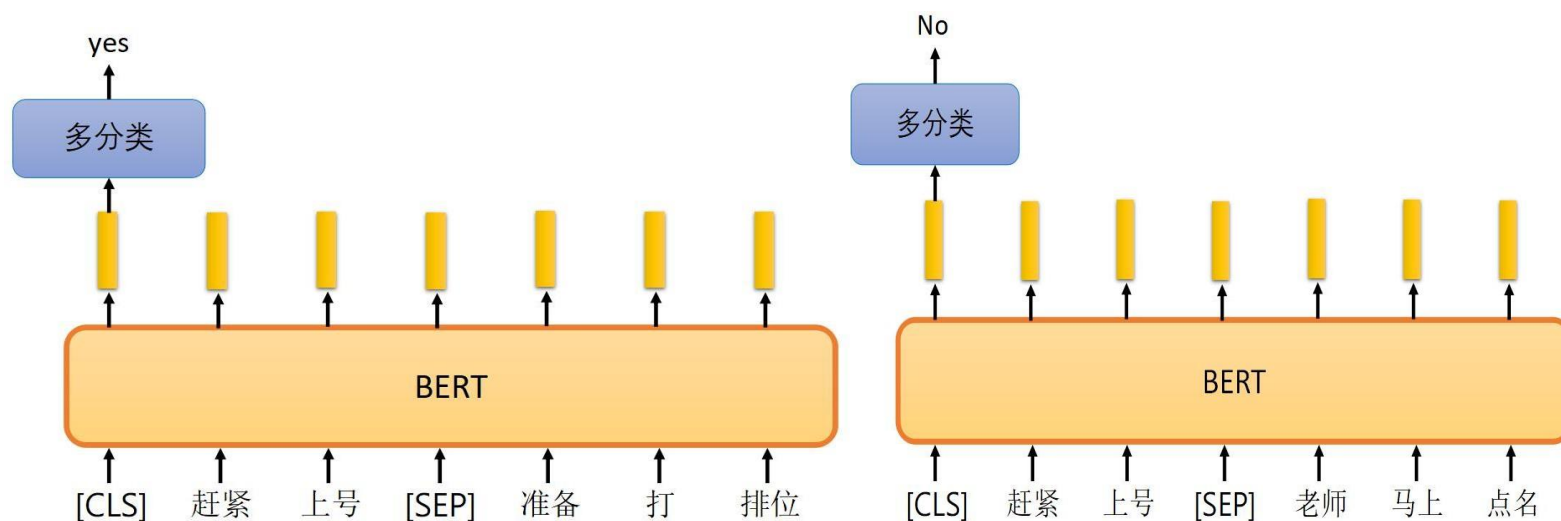
4.BERT

54

如何训练BERT

方法2：预测两个句子是否应该连在一起

[seq]: 两个句子之前的**连接符**, [cls]: 表示要做**分类**的向量



4.BERT

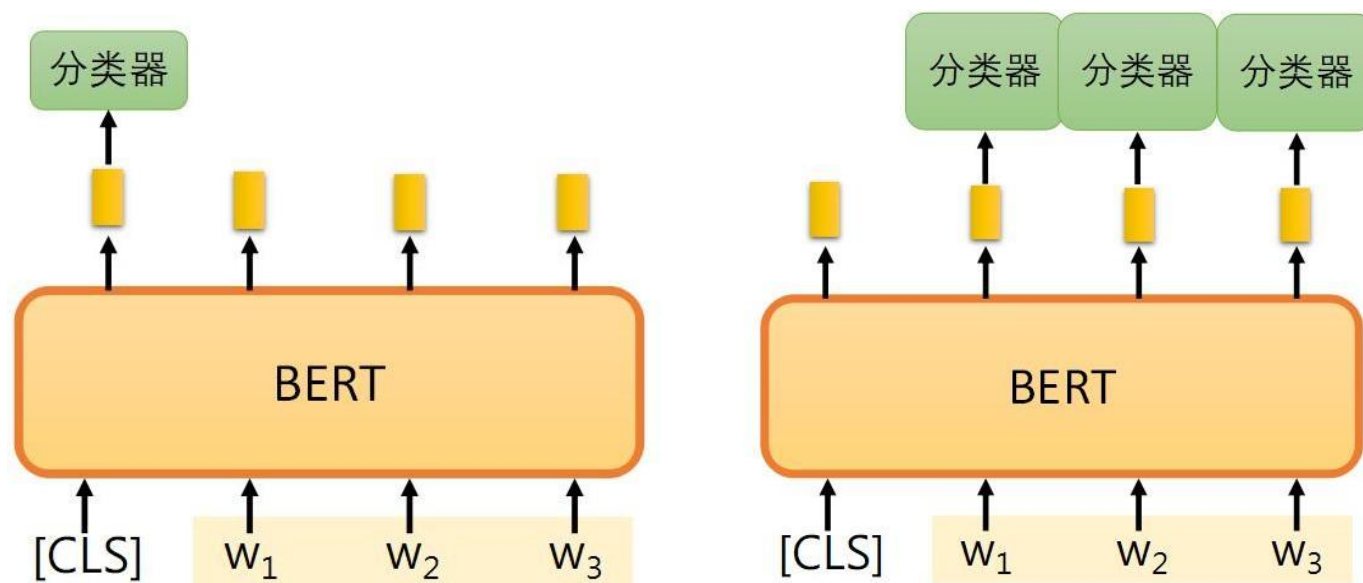
55

如何使用BERT

是不是需要先训练好向量的表达，然后再训练需要的模型呢？

所需的任务融入BERT中即可，它俩一起训练的！

分类任务：



4.BERT

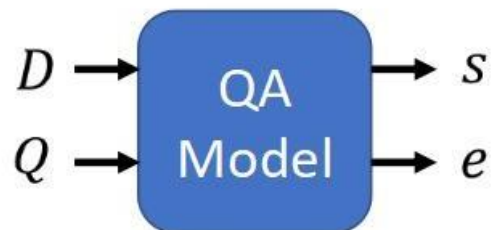
56

如何使用BERT

阅读理解题，输入是文章和问题，
输出是理解的答案位置。

文章: $D = \{d_1, d_2, \dots, d_N\}$

问题: $Q = \{q_1, q_2, \dots, q_N\}$



结果: $A = \{q_s, \dots, q_e\}$

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?

gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?

graupel

Where do water droplets collide with ice crystals to form precipitation?

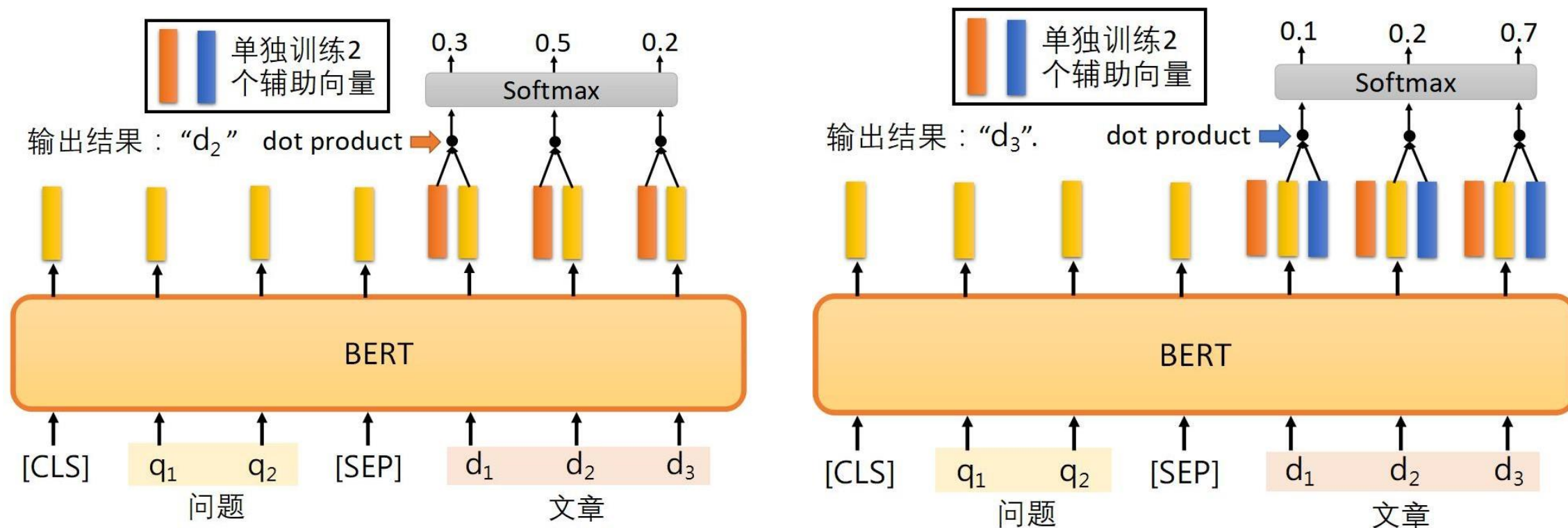
within a cloud

4.BERT

57

如何使用BERT

如何设计网络呢？需要分别计算答案的起始和终止位置



4.BERT

58

Fine tuning 如何设计网络呢？需要分别计算答案的起始和终止位置

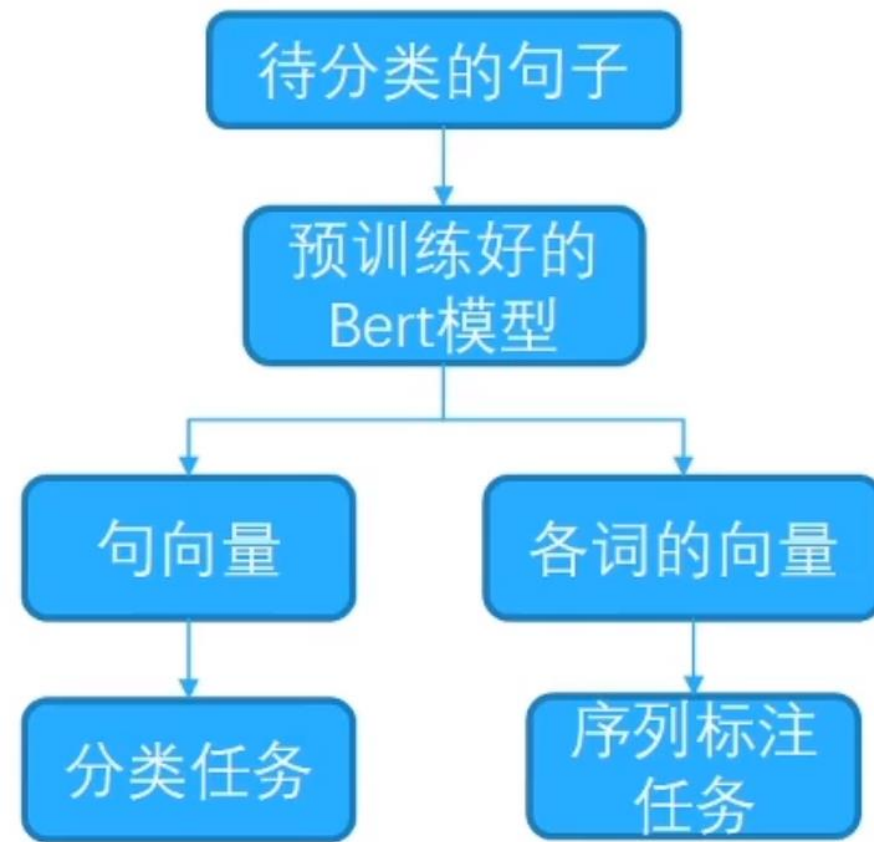
待分类句子A:

Datawhale 是一个 专注于 AI领域 的开源组织

模型输出以下各token的向量:

[CLS]Datawhale 是一个 专注于 AI领域 的开源
组织[SEP]

[CLS]表示的向量一般可以认为是句向量,用[CLS]
向量对接下游文本分类任务,可得到fine tuning的
文本分类模型



1. <https://jalammar.github.io/illustrated-transformer>
2. Andrew Ng, <http://www.deeplearning.ai>
3. https://blog.csdn.net/longxincheng_ml/article/details/86533005
4. 唐宇迪, <https://www.bilibili.com/>
5. 高永伟, <https://www.bilibili.com/>

谢谢!

