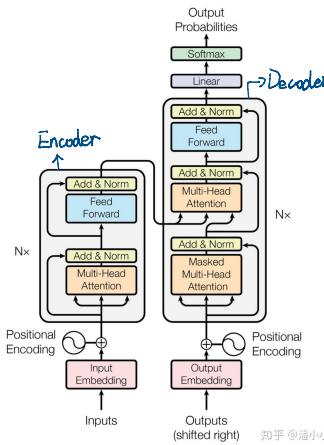


## 1. 原理

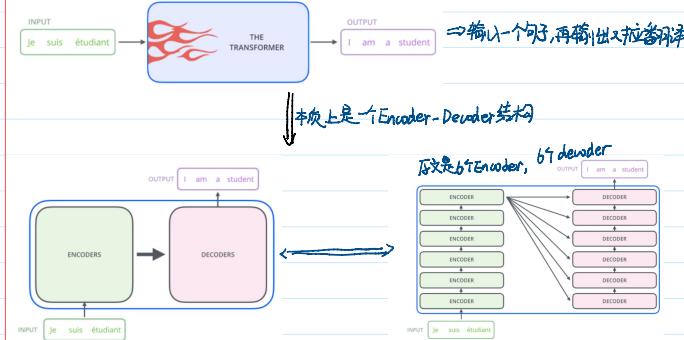
2021年4月7日 19:30

### - Transformer 大览

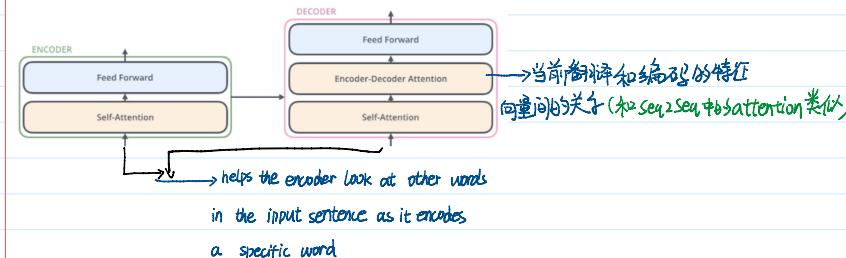


### - Transformer 大览

#### 1. 高层次 transformer

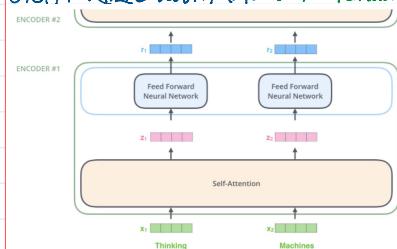


#### 2. Encoder & Decoder



#### 3. 输入编码

① 先将单词通过 embedding layer (只有第一个encoder需要embedding)

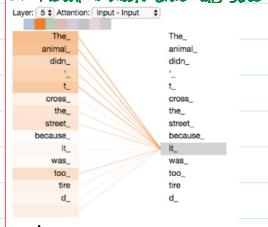


#### 4. Self-attention

核心是将输入向量的每一个词学习一个权重

ex: The animal didn't cross the street because it was too tired

判断 it 代指的内容





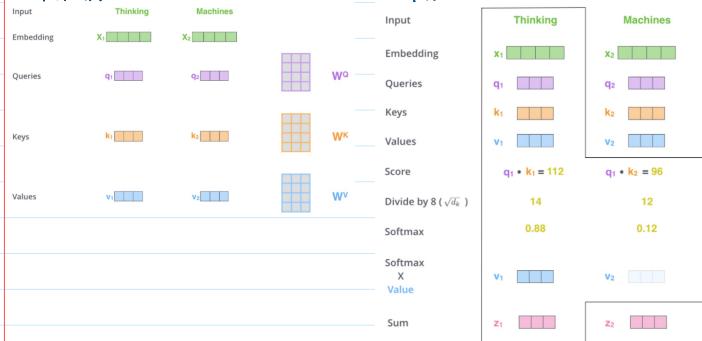
计算流程:

简单单词有3个不同的向量:  
 ↗ Query (Q), 通过 embedding 向量乘以3个权值矩阵得到  
 ↗ Key (K)  
 ↗ Value (V)

### ① 算出 Q, K, V



### ② 计算 attention



(1) 对每一个词算出其对应的 Q, K, V

(2) 对每一个词计算它与其它词(包括自己的)的 score

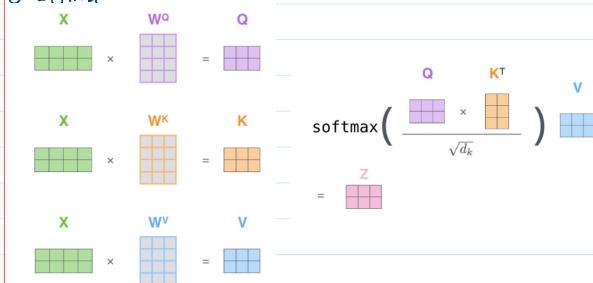
(3) (点乘即是算相似度) parts of the input sentence as we encode a word at a certain position

(4) 将得到的 score 除以  $\sqrt{d_k}$  (根据 key vectors 的维度 d\_k) → 指定权重

(5) 将 values 和对应 score 相乘并加起来

$$z_1 = v_1 \times 0.88 + v_2 \times 0.12$$

### ③ 矩阵形式



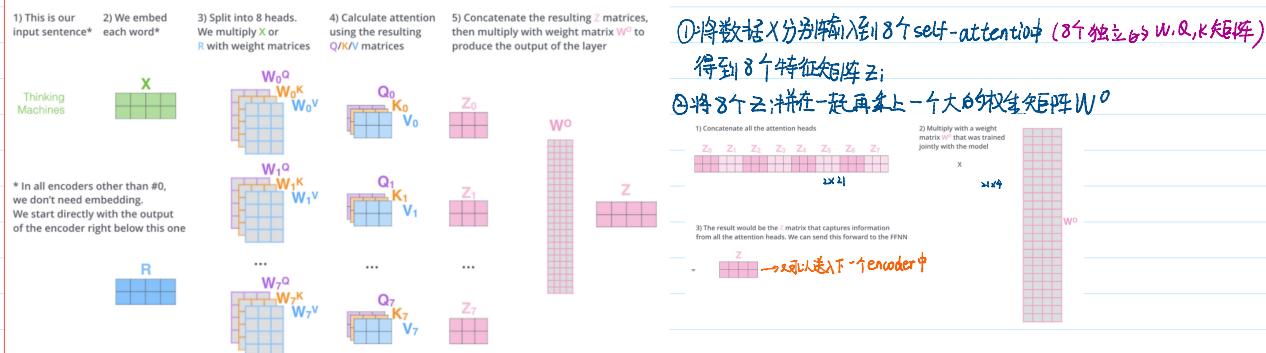
### 5: multi-head attention

self-attention 强版, ① expands the model's ability to focus on different positions

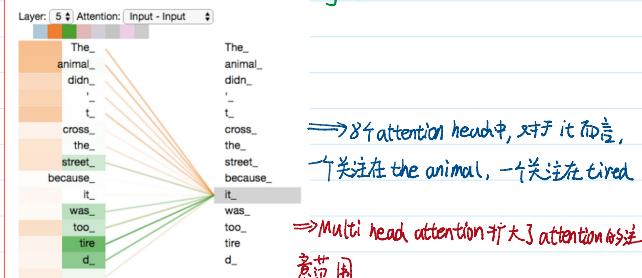
在上例中, 会传播到其他 encoding, 但它在自己位置上的 score 很大, it could be dominated by the actual word itself

### ② 增强表达空间

2. It gives the attention layer multiple "representation subspaces". As we'll see next, with multi-headed attention we have not only one, but multiple sets of Query/Key/Value weight matrices (the Transformer uses eight attention heads, so we end up with eight sets for each encoder/decoder). Each of these sets is randomly initialized. Then, after training, each set is used to project the input embeddings (or vectors from lower encoders/decoders) into a different representation subspace.



What are multi-head attention focusing at:



⇒ 8个 attention head, 对于 it 而言,  
一个关注在 the animal, 一个关注在 tired

⇒ Multi head attention 扩大了 attention 的注意范围

### 三: 位置编码

到目前为止, transformer 模型并没有捕捉顺序序列的能力,也就是说不说

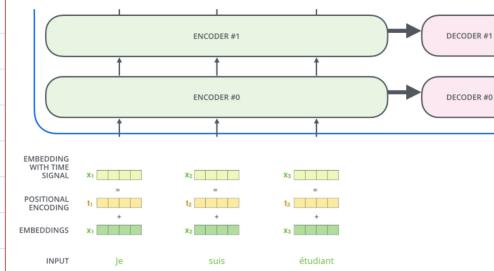
### 三：位置编码

到目前为止，transformer模型并没有捕捉顺序序列的能力，也就是说不论

句子的结构怎么打乱，transformer都会得到类似的结果。

解决方法，在embedding向量中加入一个positional encoding

To address this, the transformer adds a vector to each input embedding. These vectors follow a specific pattern that the model learns, which helps it determine the position of each word and the distance between different words in the sequence. The intuition here is adding these values to the embeddings provides meaningful distances between the embedding vectors once they're projected into Q/K/V vectors and during dot-product attention.



论文给出的编码公式如下：

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (3)$$

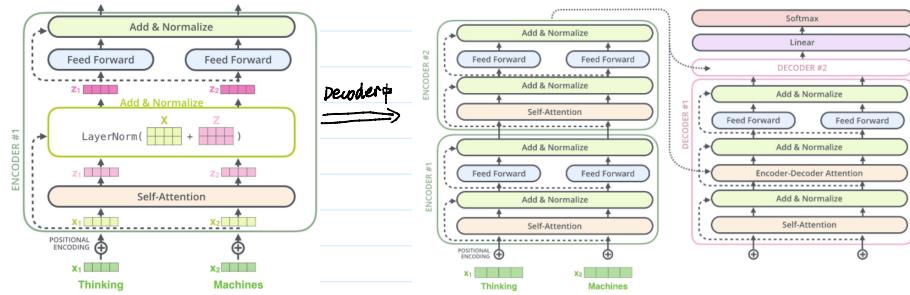
$$PE(pos, 2i + 1) = \cos\left(\frac{pos}{10000^{\frac{2i+1}{d_{model}}}}\right) \quad (4)$$

在上式中， $pos$  表示单词的位置， $i$  表示单词的维度。关于位置编码的实现可在Google开源的算法中 `get_timing_signal_1d()` 函数找到对应的代码。

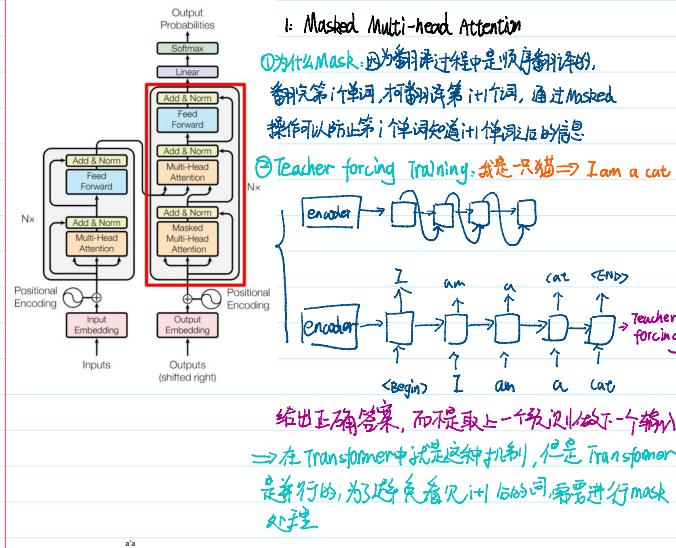
作者这么设计的原因是考虑到在NLP任务中，除了单词的绝对位置，单词的相对位置也非常 important。

根据公式  $\sin(\alpha + \beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$  以及  $\cos(\alpha + \beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$ ，这表明位置  $k + p$  的位置向量可以表示为位置  $k$  的特征向量的线性变化，这为模型捕捉单词之间的相对位置提供了非常大的便利。

### 四：残差结构



### 五：解码部分



第一步：是 Decoder 的输入矩阵和 Mask 矩阵，输入矩阵包含 “<Begin> I have a cat” (0, 1, 2, 3, 4) 五个单词的表示向量，Mask 是一个 5x5 的矩阵。在 Mask 可以 K, V 矩阵。然后计算 Q 和 KT 的乘积 QKT。

发现单词 0 只能使用单词 0 的信息，而单词 1 可以使用单词 0, 1 的信息，即只能使用之前的信息。

0	1	2	3	4
0	1	2	3	4
1	1	2	3	4
2	2	2	3	4
3	3	3	3	4
4	4	4	4	4

输入矩阵 X

0	1	2	3	4
0	1	2	3	4
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4

Mask 矩阵

QKT

$$\text{score} \propto QK^T$$

$$\begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 1 & 1 & 2 & 3 & 4 \\ 2 & 2 & 2 & 3 & 4 \\ 3 & 3 & 3 & 3 & 4 \\ 4 & 4 & 4 & 4 & 4 \end{matrix} = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{matrix}$$

第二步：接下来的操作和之前的 Self-Attention 一样，通过输入矩阵 X 计算得到 Q, K, V。Mask 可以 K, V 矩阵。然后计算 Q 和 KT 的乘积 QKT。

发现单词 0 只能使用单词 0 的信息，而单词 1 可以使用单词 0, 1 的信息，即只能使用之前的信息。

第三步：在得到 QKT 之后需要进行 Softmax，计算 attention score，我们在 Softmax 之前需要使用 Mask 矩阵遮挡每一个单词之后的信息，遮挡操作如下：

0	1	2	3	4
0	1	2	3	4
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4

QKT

按位相乘

Mask 之前 Mask

Mask QKT

Softmax

得到 Mask QKT 之后在 Mask QKT 上进行 Softmax，每一行的和都为 1。但是单词 0 在单词 1, 2, 3, 4 上的 attention score 都为 0。

第四步：使用 Mask QKT 与矩阵 V 相乘，得到输出 Z，则单词 1 的输出向量 Z1 是只包含单词 1 信息的。

0	1	2	3	4
0	1	2	3	4
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4

Mask 之前 Mask

Mask QKT

按位相乘

Softmax

得到 Mask QKT 之后在 Mask QKT 上进行 Softmax，每一行的和都为 1。但是单词 0 在单词 1, 2, 3, 4 上的 attention score 都为 0。

第五步：通过上述步骤就可以得到一个 Mask Self-Attention 的输出矩阵 Zi。和 Encoder 类似，通过 Multi-Head Attention 拼接多个输出 Zi，然后计算得到第一个 Multi-Head Attention 的输出 Zi，Zi 与输入 X 维度一样。

### 六：最后一层 (Linear + softmax)

主要区别：其中的 self-attention 的 K, V 矩阵不是使用上一个 Decoder block 的输出 Z 计算得到的，而是使用 Encoder 的编码信息直接计算（最后一个 Encoder 输出的 Z）

根据 Encoder 的输出 C 计算得到 K, V，根据上一个 Decoder block 的输出 Z 计算 Q (如果是第一个 Decoder block 则使用输入矩阵 X 进行计算)，后续的计算方法与之前描述的一致。

这样做的好处是在 Decoder 的时候，每一位单词都可以利用到 Encoder 所有单词的信息（这些信息无需 Mask）。

### 六：最后一层 (Linear + softmax)

Linear: (`inputsize`, `outputsize=vocab_size`)  $\rightarrow$  将每个词映射为 vocab\_size

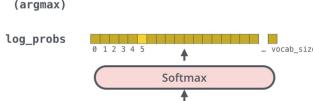
softmax: 将归一化，并取概率最大的词做输出

Which word in our vocabulary is associated with this index?

am

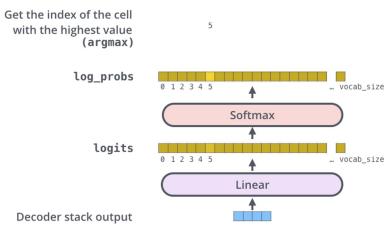
Get the index of the cell with the highest value (`argmax`)

5

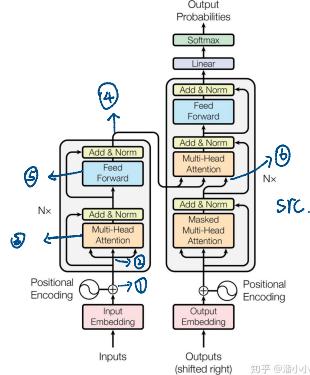


Which word in our vocabulary  
is associated with this index?

am



## 七、最终理解（结合代码）



接下来求真正的 Query, key, value (乘上参数矩阵  $W^Q, W^K, W^V$ ) 可以直接看作  
通过一个线性层的操作

