Transformer相关——（4）Poisition encoding

发表于 2021-08-17 | 更新于 2021-08-17 | 深度学习
| 字数总计: 1,158 | 阅读时长: 4分钟 | 阅读量: 1236

Transformer相关——（4）Poisition encoding

引言

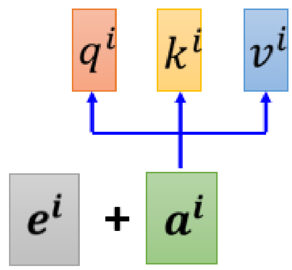
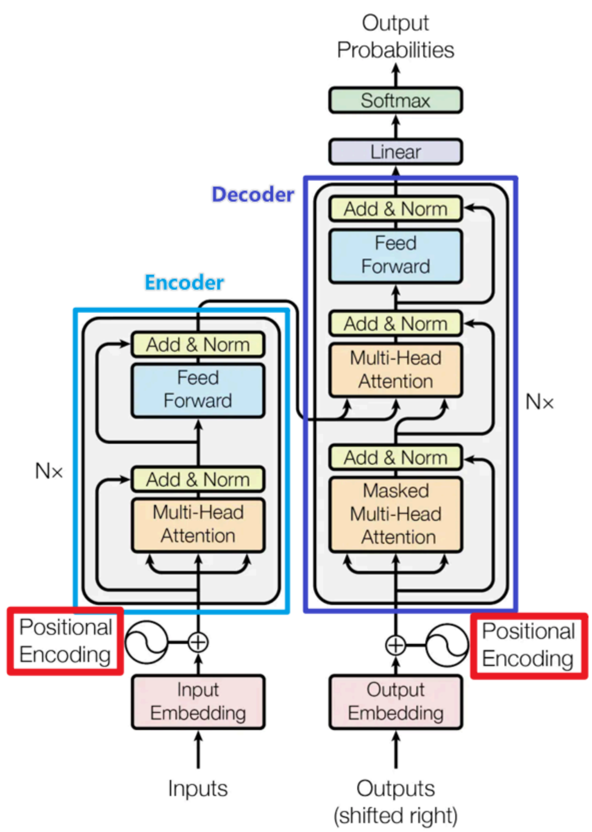
上一篇总结完了attention机制，其中提到Transformer中的self-attention机制可以并行化但是缺乏位置信息，**各个位置完全没有任何差别**，这导致了什么问题呢？比如在一个句子中，某一个词汇它是放在句首的，那它是动词的可能性可能就比较低，这种位置信息可能在NLP的命名实体识别任务中很有用。

positional encoding位置编码就可以补充上述这类位置信息。

Poisition encoding

Poisition encoding应用在哪

Poisition encoding位置编码机制**为每一个位置设定一个 vector，叫做 positional vector**（e^i），**不同的位置都有一个它专属的位置编码**，然后把e^i加到a^i上，再做self-attention操作。

> 
>
>
> 
>
>
> Figure 1: The Transformer - model architecture.

怎么设计Poisition encoding

Poisition encoding是人工设计的，最好能满足以下条件：保证值域固定，且不同长度文本，相差相同字数，差相同值，不同顺序（方向）含义不同。

总的来说可以分为两种类型：函数型和表格型。

> 1. **表格型**：建立一个长度为L的词表，按词表的长度来分配位置id
> 2. **函数型**：通过输入token位置信息，得到相应的位置编码

表格型

1. 位置直接作为编码，$[1,2,3..n]$

   > 这样的问题很明显，没有上界。过大的位置embeding，跟词embeding相加，很容易导致词向量本身含义的丢失。位置向量值不要太大，最好在限定在一个区间内。

2. 位置编码后进行归一化，$[0,1/n,2/n,....1]$

   > 这样词向量的区间就变成[0,1]且具有可比性了，但是在长文本和短文本的的情况下，同样是差两个字，数值差却不同。

函数型

1. $\sin(pos/x)$——周期性函数

   > Sin的值域[-1,1]，对于任意长度的文本，相同相对距离的词之间位置embedding的差值都是相同的。 然而x取值大，则波长大，导致相邻位置的差值变小。x取值过小，则对于长文本来说，很容易就走了几个波峰，导致不同距离，但差值相同。如何取合适的x是一个很关键的问题。

2. 相对位置函数

   在GPT-3论文中给出的公式如下：

$$\vec{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k. t), & \text{if } i = 2k \\ \cos(\omega_k. t), & \text{if } i = 2k+1 \end{cases}$$

首先需要注意的是，上个公式给出的每一个Token的位置信息编码不是一个数字，而是一个不同频率分割出来，和文本一样维度的向量。向量如下：

$$\vec{p_t} = \begin{bmatrix} \sin(\omega_1 \cdot t) \\ \cos(\omega_1 \cdot t) \\ \sin(\omega_2 \cdot t) \\ \cos(\omega_2 \cdot t) \\ \vdots \\ \sin(\omega_{d/2} \cdot t) \\ \cos(\omega_{d/2} \cdot t) \end{bmatrix}$$

其中，t就是每个token的位置，比如说是位置1，位置2，以及位置n，而不同频率是通过w_i来表示的：

$$w_i = \frac{1}{10000^{2i/d_{model}}}$$

我们分别展开看pos和$pos + k$这两个字符的关系。按照位置编码的的公式，可以计算的位置编码，其结果如下：

$$PE_{(pos+k,2i)} = sin(w_i \cdot (pos+k)) = sin(w_i pos)cos(w_i k) + cos(w_i pos)sin(w_i k)$$
$$PE_{(pos+k,2i+1)} = cos(w_i \cdot (pos+k)) = cos(w_i pos)cos(w_i k) - sin(w_i pos)sin(w_i k)$$

其中：

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$
$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

带入之后的结果如下：

$$PE_{(pos+k,2i)} = cos(w_i k)PE_{(pos,2i)} + sin(w_i k)PE_{(pos,2i+1)}$$
$$PE_{(pos+k,2i+1)} = cos(w_i k)PE_{(pos,2i+1)} - sin(w_i k)PE_{(pos,2i)}$$

我们可以知道，距离K是一个常数，所有上面公式中和的计算值也是常数，可以表示为：

$$u = cos(w_i \cdot k), v = sin(w_i \cdot k)$$

这样，就可以将写成一个矩阵的乘法：

$$\begin{bmatrix} PE_{(pos+k,2i)} \\ PE_{(pos+k,2i+1)} \end{bmatrix} = \begin{bmatrix} u & v \\ -v & u \end{bmatrix} \times \begin{bmatrix} PE_{(pos,2i)} \\ PE_{(pos,2i+1)} \end{bmatrix}$$

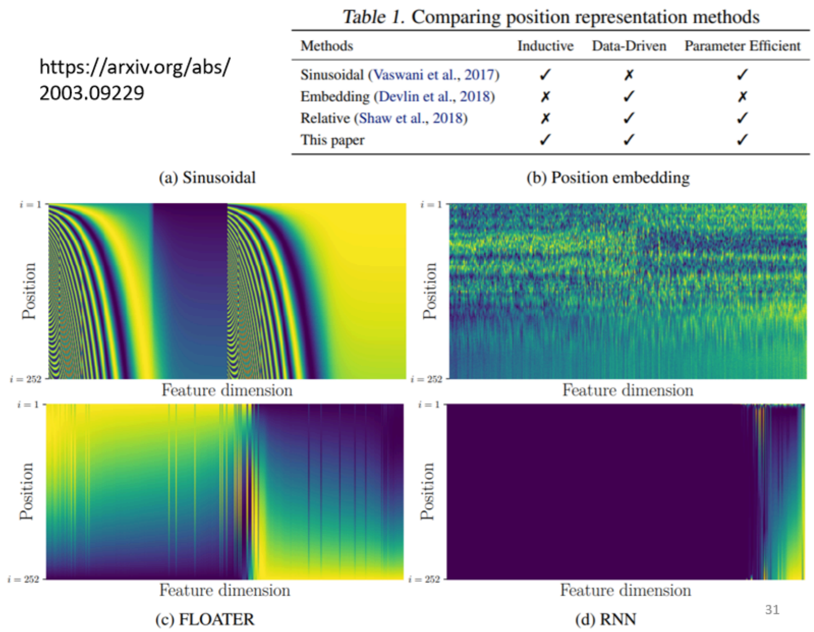
如上所述，该方法计算的相对位置是线性关系，但是位置的方向信息其实是丢失的，即$PE_{pos+k}PE_{pos} = PE_{pos-k}PE_{pos}$。

3 加入方向信息的相对位置函数

> $$Q, K, V = HW_q, H_{d_k}, HW_v, \tag{16}$$
> $$R_{t-j} = [\dots \sin(\frac{t-j}{10000^{2i/d_k}}) \cos(\frac{t-j}{10000^{2i/d_k}}) \dots]^T, \tag{17}$$
> $$A_{t,j}^{rel} = Q_t^T K_j + Q_t^T R_{t-j} + \mathbf{u}^T K_j + \mathbf{v}^T R_{t-j}, \tag{18}$$
> $$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}(A^{rel})V, \tag{19}$$
>
>
>
> 核心是公式(18)，原始的self-attention是只有$Q_t * K_j$，这里把位置信息也跟Q_t相乘了。且R_{t-j}的设定方式也决定它能反映出位置信息。假设$t = 5$，j分别为0，10，则$t - j$分别为5和-5。已知$sin(-x) = -sin(x)$，$cos(-x) = cos(x)$。很明显，我们可以发现对于0和10，值是互为正反，通过这种方式把方向信息就学到了。

4 网络自行学习 positional encoding

把 positional encoding 里面的数值，当作神经网络参数的一部分，直接学习出来，如下图中右上角的可视化结果所示。



positional encoding仍然是一个尚待研究的问题，可以自己设计，核心是尽可能能为序列提供位置信息，比如满足前面提到的几个条件：保证值域固定，且不同长度文本，相差相同字数，差相同值，不同顺序（方向）含义不同。

参考文献

transformer的Position encoding的总结

面经：什么是Transformer位置编码?

(强推)李宏毅2021春机器学习课程

李宏毅老师机器学习课程笔记