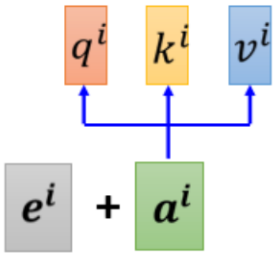
 # Position Encoding

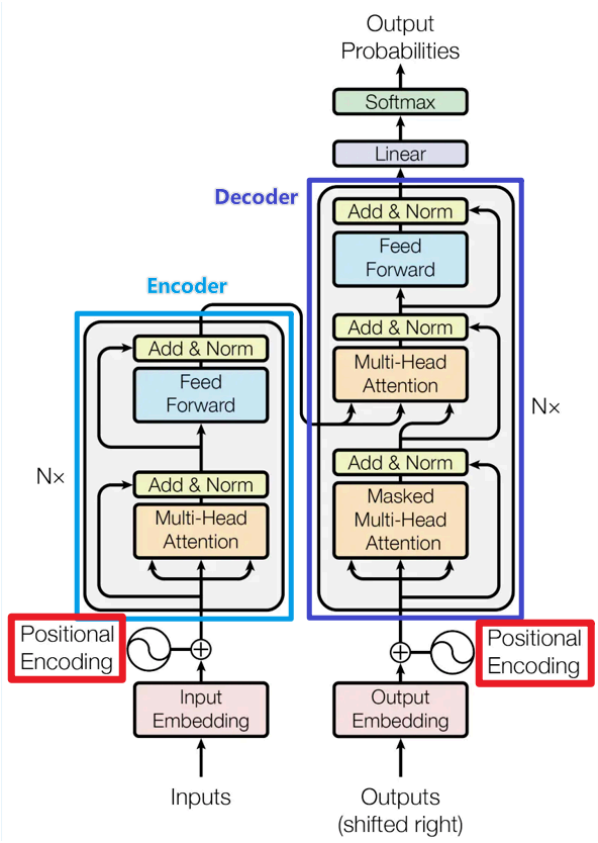
1. 引言

- Tansformer 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。
- 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。

2. 应用位置

- 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加到上a^i，再做 Self-Attention 操作。





3. Position Encoding 设计

- Position 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。
  a) 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
  b) 函数型：通过输入 token 位置信息，得到相应的位置编码

3.1 表格型

- **位置直接作为编码**，$[1,\ 2,\ 3,\ \dots,\ n]$

  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。

- **位置编码后进行归一化**，$[1/n,\ 2/n,\ 3/n,\ \dots,\ 1]$

  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。

3.2 函数型

3.2.1 $Sin(pos/x)$ —— 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。

3.2.2 相对位置函数

- 在 GPT-3 论文中给出的公式如下:

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

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:

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

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:

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

- 在 Transformer 中给出的公式如下:

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

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

其中，pos表示序列中 token（也就是单词）的位置

i表示词向量的维度范围，0到$d_{model}/2-1$之间的整数值

d_{model}表示 embedding 维度，也就是输入的序列维度

- **有一个核心的问题: 相对位置是线形关系，但是位置的方向信息其实是丢失的。**

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:

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

距离K是一个常数，所有上面公式中的计算值也是常数，可以表示为:

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

这样，就可以将其写成一个矩阵乘法:

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

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:

$$PE_{pos+k}PE_{pos} = PE_{pos-k}PE_{pos}$$

- **加入方向信息的相对位置函数**

  核心是公式(18)，原始的 Self-Attention 是只有$Q_t \times K_j$，这里把位置信息也与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和10，则$t - j$分别为5和-5。已知，$\sin(-x) = -\sin(x)$，$\cos(-x) = \cos(x)$。因此，对于0和10，值是互为正反，通过此方式就把方向信息学到了。

$$Q, K, V = HW_q, H_{d_k}, HW_v, \tag{16}$$

$$R_{t-j} = [\ldots \sin(\frac{t-j}{10000^{2i/d_k}}) \cos(\frac{t-j}{10000^{2i/d_k}}) \ldots]^T, \tag{17}$$

$$A^{rel}_{t,j} = Q_t^T K_j + Q_t^T R_{t-j} + \mathbf{u}^T K_j + \mathbf{v}^T R_{t-j}, \tag{18}$$

$$\text{Attn}(Q, K, V) = \text{softmax}(A^{rel})V, \tag{19}$$

- **网络自行学习**

  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。

https://arxiv.org/abs/2003.09229

Table 1. Comparing position representation methods

Methods	Inductive	Data-Driven	Parameter Efficient
Sinusoidal (Vaswani et al., 2017)	✓	✗	✓
Embedding (Devlin et al., 2018)	✗	✓	✗
Relative (Shaw et al., 2018)	✗	✓	✓
This paper	✓	✓	✓