位置编码总结

xinli*

November 15, 2025

由于 Transformer 本身不具备像 RNN 那样的处理序列顺序的能力，如果没有位置编码，句子" 猫吃鱼" 和 "鱼吃猫" 对于 Transformer 来说是完全一样的，他只关心每个词是什么，而不关心词的顺序，所以我们需要一种方式将单词在句子中的位置信息注入到模型中。

1 Sin-Cos 绝对位置编码

sin-cos 编码的目标是：为序列中的每个位置（例如，句子中的第 1 个词、第 2 个词...）生成一个独一无二的、固定大小的向量，然后将这个向量与单词本身的词向量相加，作为模型的输入。假设我们的词向量（或模型隐藏层的维度）是 d_{model}（在原始论文中为 512）。那么，位置编码也将是一个长度为 d_{model} 的向量。

1.1 具体实现

对于位置为 pos 的词，其位置编码向量的第 i 个元素（i 从 0 开始）的计算方式如下：

- 当 i 是偶数时（即 $i\%2 == 0$），使用正弦函数：

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

- 当 i 是奇数时（即 $i\%2 == 1$），使用余弦函数：

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

其中：pos 指的是词在序列中的位置，例如 $0, 1, 2, \ldots$。i 指的是位置编码向量中的维度索引，从 0 到 $d_{\text{model}} - 1$。d_{model} 指的是编码向量的总维度。

我们也可以把公式写成下面这种等价形式，它把奇偶维度明确分开：

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

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

这里的 k 从 0 到 $d_{\text{model}}/2 - 1$。

*https://github.com/xinli2008

1.2 Sin-Cos 位置编码的特性

1. Sin-Cos 位置编码可以通过预计算，不需要训练。

2. 可以泛化到比训练时更长的序列：由于正弦和余弦函数是周期性的（但每个维度的周期都不同），即使模型在训练时只见过长度为 50 的句子，在推理时遇到长度为 100 的句子，它也能计算出这些新位置的位置编码。虽然效果不一定完美，但至少是可行的。

3. 数值范围稳定：正弦和余弦函数的取值范围是 [-1, 1]，这与经过归一化处理的词向量范围相匹配，使得数值计算更加稳定。

2 可训练的绝对位置编码

如果说 sin-cos 编码是" 手工设计" 的，那么可训练的位置编码就是" 让模型自己学"。它的理念指的是：既然我们不知道什么形式的编码最好，那就把位置编码作为模型参数的一部分，让模型在训练数据中自己学习最优的位置表示。

2.1 具体实现

1. 定义一个位置编码矩阵

假设最大序列长度为 max_seq_len，模型隐藏层维度为 d_model，我们可以创建一个可训练的参数矩阵：$PositionEmbedding = nn.Parameter(torch.randn(max\_seq\_len, d\_model))$，这个矩阵的形状是 $[max\_seq\_len, d\_model]$。

2. 使用方式

对于序列中位置为 pos 的词，我们从矩阵中取出第 pos 行：$pos\_embed = PositionEmbedding[pos]$，然后将这个词向量与位置编码向量相加：$input = word\_embedding + pos\_embed$。

3. 训练过程

这个 PositionEmbedding 矩阵会与模型的所有其他参数（权重、偏置等）一起，通过梯度下降进行优化。

2.2 可训练的绝对位置编码的特性

1. 实现简单：几行代码就能实现，不需要复杂的数学计算。

2. 需要更多数据：需要足够的训练数据来学习有意义的编码。

3. 长度严格限制：无法处理超过 max_seq_len 的序列，缺乏外推能力。

3 相对位置编码

之前的绝对位置编码关心的是每个 token 在序列中的绝对位置，例如 "我是第 3 个词"。二相对位置编码关心的是两个 token 之间的相对距离。

3.1 具体实现

我们在计算自注意力时，有一个 Query 和一系列 Key 向量。相对位置编码的核心操作是：**修改注意力分数的计算过程**。

1. 建立相对位置偏置表

我们需要预先定义一个可以学习的参数表。这个表不是一个位置一个向量，而是一**个相对距离一个偏置值**。例如，我们预设一个最大相对距离，那么所有可能的相对距离就是从 $-k$ 到 $+k$，共 $2k+1$ 种。这个表就有 $2k+1$ 个可学习的标量值（如果是多头注意力，则是 $2k+1$ 个向量，每个头独立学习）。

2. 在注意力机制中注入相对信息

当计算 Token i 对 Token j 的注意力分数时：

原始注意力分数计算为：

$$\text{Score}(i,j) = Q_i \cdot K_j^T$$

引入相对位置偏置后，新的注意力分数为：

$$\text{Score}'(i,j) = Q_i \cdot K_j^T + b_{i-j}$$

其中：Q_i 是第 i 个 token 的 Query 向量；K_j 是第 j 个 token 的 Key 向量；b_{i-j} 是从相对位置偏置表中查找到的、对应相对距离 $i-j$ 的偏置值；$\cdot$ 表示向量点积运算。

3.2 相对位置编码的特征

1. 相对位置编码的外推性：模型只学习了预设最大距离 k 以内的偏置。当推理时出现距离 $> k$ 的情况，模型无法推理，属于分布外的问题。

2. 相对位置编码的参数量：相对位置编码的参数量都远少于可学习的绝对位置编码。对于可学习的绝对位置编码来说，参数量公式为 $max\_seq\_len \times d\_model$，而对于 swin transformer 实现的相对位置编码来说，他的参数量是 $(2M-1) \times (2M-1) \times$ num_heads。

4 旋转位置编码 ROPE

ROPE 的全称是 Rotary Position Embedding，即旋转式位置编码。它的核心思想是：不直接将位置信息加在词向量上，而是通过旋转的方式，将绝对位置信息与词向量的表示巧妙地融合在一起。

4.1 核心思想

ROPE 的核心直觉非常巧妙：将词嵌入向量想象成一个在二维平面上的向量，通过旋转这个向量来表示其位置的变化。

首先，想象一个二维坐标系。我们有一个向量 q，它代表一个词（不考虑位置）。如果这个词在序列的第 m 个位置，我们就将向量 $\mathbf{q}$ 逆时针旋转 $m \cdot \theta$ 角度，得到新的向量 $\tilde{\mathbf{q}}$。这个 $\tilde{\mathbf{q}}$ 就是包含了位置 m 信息的向量。现在考虑两个词，位置分别是 m 和 n。它们的嵌入向量在经过各自的旋转后，变成了 $\tilde{\mathbf{q}}_m$ 和 $\tilde{\mathbf{k}}_n$。关键来了，当我们计算 $\tilde{\mathbf{q}}_m$ 和 $\tilde{\mathbf{k}}_n$ 的内积（即 Attention 机制中的核心计算）时，这个内积结果只依赖于它们之间的相对位移 $m-n$，而与它们的绝对位置 m 和 n 无关！

为什么内积只与相对位置有关？

下面我们进行严格的数学推导。

令 $\mathbf{q}, \mathbf{k} \in \mathbb{R}^2$ 为原始词向量（不含位置信息），定义旋转矩阵为：

$$R(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

则位置 m 处的旋转向量为：

$$\tilde{\mathbf{q}}_m = R(m\theta)\mathbf{q}, \quad \tilde{\mathbf{k}}_n = R(n\theta)\mathbf{k}$$

计算它们的内积：

$$\tilde{\mathbf{q}}_m \cdot \tilde{\mathbf{k}}_n = \left(R(m\theta)\mathbf{q}\right)^T \left(R(n\theta)\mathbf{k}\right) = \mathbf{q}^T R(m\theta)^T R(n\theta)\mathbf{k}$$

由于旋转矩阵满足正交性：$R(\phi)^T = R(-\phi)$，因此：

$$R(m\theta)^T R(n\theta) = R(-m\theta)R(n\theta) = R((n-m)\theta)$$

所以：

$$\tilde{\mathbf{q}}_m \cdot \tilde{\mathbf{k}}_n = \mathbf{q}^T R((n-m)\theta)\mathbf{k} = \mathbf{q} \cdot \left(R((n-m)\theta)\mathbf{k}\right)$$

注意到 $n - m = -(m - n)$，且旋转是周期性的，我们可以写成：

$$\tilde{\mathbf{q}}_m \cdot \tilde{\mathbf{k}}_n = \mathbf{q} \cdot \left(R(-(m-n)\theta)\mathbf{k}\right)$$

这表明：内积结果仅依赖于相对位移 $m - n$，而与 m 和 n 的具体值无关！

4.2 RoPE 的具体形式

4.3 ROPE 的特性

1. 不需要训练：RoPE 是一种**确定性**的位置编码方法，其旋转矩阵 $\mathbf{R}^d_{\Theta,m}$ 中的参数是预先根据公式计算得到的：

$$\theta_i = 10000^{-2i/d}, \quad i = 0, 1, \ldots, d/2 - 1$$

其中 d 是嵌入维度，m 是位置索引。这意味着：对于给定的位置 m 和维度 d，旋转角度是固定的，不需要根据梯度下降学习。

2. 外推性：与可学习的位置嵌入（如 BERT 中最多 512 个位置）不同，ROPE 使用的是基于频率的三角函数形式，理论上可以扩展到任意长度（只要频率基底合理）。

3. 显示编码相对距离：ROPE 通过将位置信息编码为旋转矩阵，使得注意力分数天然依赖于相对位置差 m−n。这种设计让模型更容易泛化到未见过的相对距离。