1.1 位置编码与长度外推

1.1.1 相对位置编码

原始的 transformer 使用的 Sinusoidal 函数来做绝对位置编码，如式??。我们现在来介绍几种相对位置编码方法。

[1] 提出学习全局共享的相对位置的 embedding。首先，我们看下固定距离的相对位置的计算，对于位置 i 和位置 j，取定长度 k，那么：

$$clip(j-i,k) = \max(-k, \min(k, j-i)) = \begin{cases} k & \text{如果 } j-i > k > 0 \\ j-i & \text{如果 } 0 \le j-i \le k \\ j-i & \text{如果 } -k < j-i < 0 < k \\ -k & \text{如果 } j-i \le -k < 0 \end{cases}$$

注意 k 为正，$j-i$ 可正可负。也即如果 i,j 之间的距离大于 k 就只取 k，否则取实际距离。我们学习两组 $2k+1$ 个相对位置 embedding 向量：$p^k_{-k},\dots,p^k_k$ 和 $p^v_{-k},\dots,p^v_k$。然后 self attention 改变过程如下：

$$\begin{aligned} e_{ij} &= \frac{(x_iW^q)(x_jW^k)^T}{\sqrt{d}} \\ \alpha_{ij} &= \frac{\exp e_{ij}}{\sum_k \exp e_{ik}} \\ z_i &= \sum_j \alpha_{ij}(x_jW^v) \end{aligned} \quad \text{变为} \quad \begin{aligned} e_{ij} &= \frac{(x_iW^q)(x_jW^k + p^k_{clip(j-i,k)})^T}{\sqrt{d}} \\ \alpha_{ij} &= \frac{\exp e_{ij}}{\sum_k \exp e_{ik}} \\ z_i &= \sum_j \alpha_{ij}(x_jW^v + p^v_{clip(j-i,k)}) \end{aligned}$$

通过在每个注意头之间共享相对位置表示来降低存储相对位置表示的空间复杂度。注意，之所以说这是一相对位置编码，而非绝对位置编码，是因为我们学习的是 p^k_{j-i}, p^v_{j-i}，也就是这两个参数只跟两个位置的相对距离有关，而跟两个位置的绝对取值无关。

Transformer-XL [2] 试图对长序列建模。为了建模长度超过 transformer 的输入长度的序列，可以对输入序列分段，然后循环地输入每个段到 encoder。设第 τ 个段在第 n 层 Encoder 的输出为 h^n_τ，那么：

$\widetilde{h}^{n-1}_{\tau+1} = [SG(h^{n-1}_\tau) \quad h^{n-1}_{\tau+1}]$	把本层对前一段的输出跟本层对当前段的输出拼接起来
$q^n_{\tau+1}, k^n_{\tau+1}, v^n_{\tau+1} = h^{n-1}_{\tau+1}W_q, \widetilde{h}^{n-1}_{\tau+1}W_k, \widetilde{h}^{n-1}_{\tau+1}W_v$	q,k,v 变换，注意变换的输入不同
$h^n_{\tau+1} = Encoder(q^n_{\tau+1}, k^n_{\tau+1}, v^n_{\tau+1})$	multi head self attention

其中 SG 表示本操作不要计算梯度。可以看出 $k^n_{\tau+1}, v^n_{\tau+1}$ 是融合了前一段的信息。我们把上式的所有操作概括为函数 f：

$$h^n_{\tau+1} = f(h^{n-1}_\tau, h^{n-1}_{\tau+1})$$

可以看出跟 RNN 是相像的。Transformer-XL 还提出一种相对位置编码方法 (我们这里先忽略分母)：

$$q_ik_j^T = (x_iW^q)(x_jW^k)^T$$

变为:

$$q_i k_j^T = (x_i W^q)(x_j W^k)^T + (x_i W^q)(p_{i-j}\widetilde{W}^k)^T + (uW^q)(x_j W^k)^T + (vW^q)(p_{i-j}\widetilde{W}^k)^T \tag{1.1}$$

其中 p_{i-j} 采相对位置版本的 Sinusoidal 做位置编码，也就是取值类似式??。$u, v, \widetilde{W}$ 是新引入的参数。也就是在 q, k 做内积的时候引入相对位置编码。Transformer with Untied Positional Encoding (TUPE) [3] 则将式1.1简化如下：

$$q_i k_j^T = (x_i W^q)(x_j W^k)^T + (x_i W^q)(p_{i-j} W^k)^T + (p_{i-j}^T W^q)(x_j W^k)^T \tag{1.2}$$

式1.1和式1.2其实都使用了同一个形式：

$$q_i k_j^T = g(x_i, x_j, i-j)$$

也就是 q_i, k_j 向量内积的结果跟且只跟 k_j^T, x_j 以及相对距离 $i-j$ 有关。

正式介绍 RoPE 前，我们这里补充一些背景知识。一个复数最常见的表示为如下：

$$z = a + bi, \text{其中}, i^2 = -1, Re(z) = a, Im(z) = b$$

a 为实部，b 为虚部，二者皆为实数。复数有一套与实数相兼容的四则运算规则。z 的共轭为:$\bar{z} = a - bi$。在极坐标 (e, ϕ) 中，

$$z = re^{i\phi}, \text{其中}, r = |z| = \sqrt{a^2 + b^2}, \phi = \arctan(y/x)(\text{其实只有 } x > 0 \text{ 本式才成立})$$

根据欧拉公式：

$$e^{i\phi} = \cos\phi + i\sin\phi$$

所以我们对同一个复数 z，有三种等价的定义形式：

$$z = a + bi = |z|e^{i\phi} = |z|(\cos\phi + i\sin\phi)$$

我们还可以引进复数的矩阵形似：

$$a + bi \Longleftrightarrow \begin{pmatrix} a & -b \\ b & a \end{pmatrix} = a\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + b\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

此时实数 1 对应单位矩阵，i 对应 $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$。旋转矩阵是在乘以一个向量后改变向量的方向但不改变大小。旋转矩阵有一些有趣的性质，如下：

- 旋转前后，内积不变，也就是：$ab^T = aM(bM)^T = aMM^Tb^T$，所以选择矩阵是正交的。
- 二维空间中的选择矩阵如下：

  $$R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} = \cos\theta\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \sin\theta\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = \exp\left(\theta\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\right)$$

  这里说的意思就是在原坐标系中点 P 的坐标值分别为 x_a, y_a, 坐标旋转 θ 度后，点 P 的坐标值变为 $[x_b, y_b]^T = R[x_a, y_a]^T$。

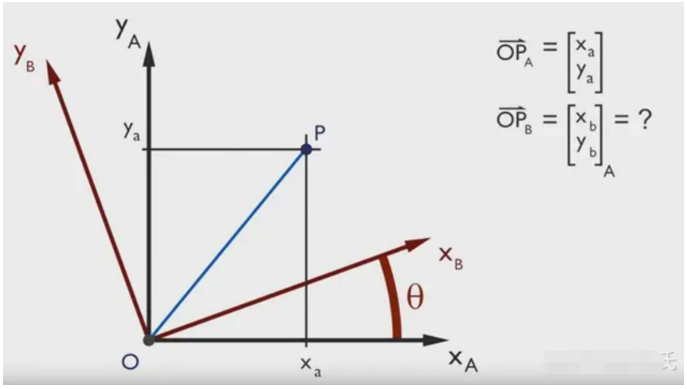

图 1.1: 2 维旋转示意图

在 RoPE [4] 中，当 q, k, v 向量的维度大小为 2 时，位置 m 的 q 向量从

$$W^q x_m^T \quad \text{变为} \quad R_m^2 x_m W^q = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} W^q x_m^T$$

其中，θ 是超参数。当维度为 d 时，把这种方法在每个大小为 2 的子维度空间中套用，可得：

$$R_m^d = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \dots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \dots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \dots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

其中，$\theta_i = 10000^{-2(i-1)/d}$。此时，self attention 由：

$$e_{ij} = q_i^T k_j = \frac{(W^q x_i)^T (W^k x_j)}{\sqrt{d}}$$

变为

$$e_{ij} = \frac{(R_i^d W^q x_i)^T (R_j^d W^k x_j)}{\sqrt{d}} = \frac{x_i^T (W^q)^T (R_i^d)^T R_j^d W^k x_j}{\sqrt{d}} = \frac{x_i^T (W^q)^T R_{i-j}^d W^k x_j}{\sqrt{d}}$$

也即是 q, k 做内积之前，先各自旋转一下，旋转时每两个维度组成的子空间单独旋转 (因为子空间之间的选择角度不一样，且这差异跟绝对位置有关)。可以证明当 $abs(j-i)$ 越大时，e_{ij} 的上界会波动减小，这就是所谓的远程衰减。 [5] 把 RoPE 的推导过程做了详细的复现。

1.1.2 外推算法

长度外推意味着可以在短的上下文窗口上进行训练，在长的上下文窗口上进行推理。我们需要知道，原始的 transformer 可以接受任意长度的序列。 实现外推的一种最简单的方法是：在推理时把长度超过 L_{train} 的输入序列分段，每段长度跟训练时一样，都是 L_{train}，每个段独立输入模型。还有一种方法，在 inference 时，每个 token 只能看到之前的 L_{train} 个 token(单向模型) 或者两边 L_{train} 个 token(双向模型)，并且位置编码采用相对位置编码 (以当前 token 为起点或者中点)。

第一篇明确研究 Transformer 长度外推性的工作应该是 ALiBi(Attention with Linear Biases) [6]。ALiBi 对 attention 部分做如下改变：

$$e_{ij} = \frac{(x_i W^q)(x_j W^k)^T}{\sqrt{d}} \Rightarrow e_{ij} = \begin{cases} \frac{(x_i W^q)(x_j W^k)^T}{\sqrt{d}} - m(i-j) & \text{if } i \geq j \\ 0 & \text{else} \end{cases}$$

其中，m 是每个 head 的斜率，有 n 个 head 时，第 $k, k = 1, 2, \ldots$ 个 head 对应的 m 取值为：$\frac{1}{2^{8k/n}}$。KERPLE [7] 和 Sandwich [8] 都是 ALiBi 的改进。

位置插值 (Position Interpolation, PI) [9] 对 RoPE 做如下改变:

$$e_{ij} = \frac{(R_i^d W^q x_i)^T (R_j^d W^k x_j)}{\sqrt{d}} \Longrightarrow e_{ij} = \frac{(R_{\frac{iL}{L'}}^d W^q x_i)^T (R_{\frac{jL}{L'}}^d W^k x_j)}{\sqrt{d}}$$

其中，L 是训练时的窗口长度，L' 是在推理时期望的目标长度。PI 是将超出训练长度的位置下标等比例缩小，映射到模型已经学习的位置范围内。线性插值法不需要修改模型架构，只需要少量微调即可将 LLaMA 的上下文窗口扩展到 32768。

参考文献

[1] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations, 2018.

[2] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

[3] Guolin Ke, Di He, and Tie-Yan Liu. Rethinking positional encoding in language pre-training, 2021.

[4] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023.

[5] 一文看懂 llama 中的旋转式位置编. [EB/OL]. https://zhuanlan.zhihu.com/p/642884818.

[6] Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation, 2022.

[7] Ta-Chung Chi, Ting-Han Fan, Peter J. Ramadge, and Alexander I. Rudnicky. Kerple: Kernelized relative positional embedding for length extrapolation, 2022.

[8] Ta-Chung Chi, Ting-Han Fan, Alexander I. Rudnicky, and Peter J. Ramadge. Dissecting transformer length extrapolation via the lens of receptive field analysis, 2023.

[9] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation, 2023.