从零实现 Transformer 并在小数据集完成训练、消融与挑战项扩展

姓名：李镇远　学号：25126099

1　引言

Transformer 通过自注意力机制建模序列全局依赖，避免了 RNN 中的长距离梯度衰减问题。本文目标：

1. 从零实现 Encoder-only 语言模型（字符级 LM），包括：Scaled Dot-Product Attention、Multi-Head Attention、Position-wise FFN、Pre-Norm 残差结构、位置编码；
2. 在 Tiny Shakespeare 数据集上训练并绘制 loss / ppl 曲线；
3. 进行消融：去掉位置编码（No-PE）、缩小注意力接收窗口为 16（Win16）；
4. 完成挑战项扩展：实现 Decoder、相对位置与稀疏注意力，并提供可切换接口与命令。

2　相关工作

代表性工作为 **Transformer**（Vaswani et al., 2017）。后续发展包括相对位置编码（RoPE、可学习相对偏置）、局部/稀疏注意力（如 block-sparse、滑动窗口）、以及稳定训练技巧（Pre-Norm、AdamW、warmup、梯度裁剪等）。本文实现以原始结构为主，并加入上述扩展。

3　模型结构与数学推导

3.1　Scaled Dot-Product Attention

给定 $Q \in \mathbb{R}^{L \times d_k}$、$K \in \mathbb{R}^{L \times d_k}$ 和 $V \in \mathbb{R}^{L \times d_v}$：

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + \mathcal{B}\right) V,$$

其中 $\frac{1}{\sqrt{d_k}}$ 稳定梯度，$\mathcal{B}$ 为可加项，可承载相对位置偏置、稀疏遮罩与因果掩码。

3.2 Multi-Head Attention

$$\mathrm{MHA}(X) = \mathrm{Concat}(H_1, \ldots, H_h)W^O, \quad H_i = \mathrm{Attention}(XW_i^Q, XW_i^K, XW_i^V).$$

3.3 Position-wise FFN

$$\mathrm{FFN}(x) = \sigma(xW_1 + b_1)W_2 + b_2,$$

默认使用 GELU，可切换 ReLU。

3.4 残差与 LayerNorm（Pre-Norm）

每一子层采用 Pre-Norm：$x \leftarrow x + \mathrm{Sublayer}(\mathrm{LN}(x))$，训练更稳定。

3.5 位置编码

绝对位置（Sin/Cos）：

$$\mathrm{PE}(pos, 2i) = \sin\Big(\frac{pos}{10000^{2i/d_{\text{model}}}}\Big), \ \mathrm{PE}(pos, 2i+1) = \cos\Big(\frac{pos}{10000^{2i/d_{\text{model}}}}\Big).$$

相对位置（两种可选）：

1. **RoPE**：对 Q, K 的相邻维度施加旋转，得到 $\tilde{Q}, \tilde{K}$ 后再计算打分；
2. **可学习相对偏置**：建立表 $\mathbf{B} \in \mathbb{R}^{(2L-1)\times h}$，按位移 $\Delta = i - j$ 选取并加到打分矩阵。

4 实现细节

框架与结构： 基于 PyTorch，包含 Embedding、若干堆叠的 Encoder Block（MHA + FFN + Pre-Norm + 残差）与可选 Decoder Block（挑战项），字符分类头。张量形状默认 `(batch, seq, dim)`。

训练稳定性： 使用 AdamW，可选 warmup+ 余弦退火，梯度裁剪（例如 1.0）。CPU 提供快速版配置，GPU 可启用更大步数与窗口。

关键实现片段（Attention）：

```
def scaled_dot_product_attention(Q, K, V, mask=None, bias=None):
    d_k = Q.size(-1)
    scores = (Q @ K.transpose(-2, -1)) / (d_k ** 0.5)
```

```
    if bias is not None:
        scores = scores + bias
    if mask is not None:
        scores = scores.masked_fill(mask == 0, -1e9)
    attn = scores.softmax(dim=-1)
    return attn @ V, attn
```

5 实验设置

数据与任务： Tiny Shakespeare 字符级语言建模。数据集来源:https://huggingface.co/datasets/karpat
训练与验证集由脚本自动划分与缓存。

表 1: 超参数（默认基线）

参数	数值
嵌入维度 d_{model}	256
注意力头数 h	4
前馈维度 d_{ff}	1024
层数 layers	4
序列长度（训练最大）	128
批大小 batch	64
学习率 lr	3×10^{-4}
优化器	AdamW
调度器	warmup（1k 步）+ 余弦退火
权重衰减	0.01
梯度裁剪	1.0
Dropout（MHA/FFN）	0.1
位置编码	Sin/Cos（可切换 RoPE/RelBias）
注意力类型	全局（可切换 Win16/稀疏）

超参数设置：

复现实验命令： exact 命令位于 `README.md`。

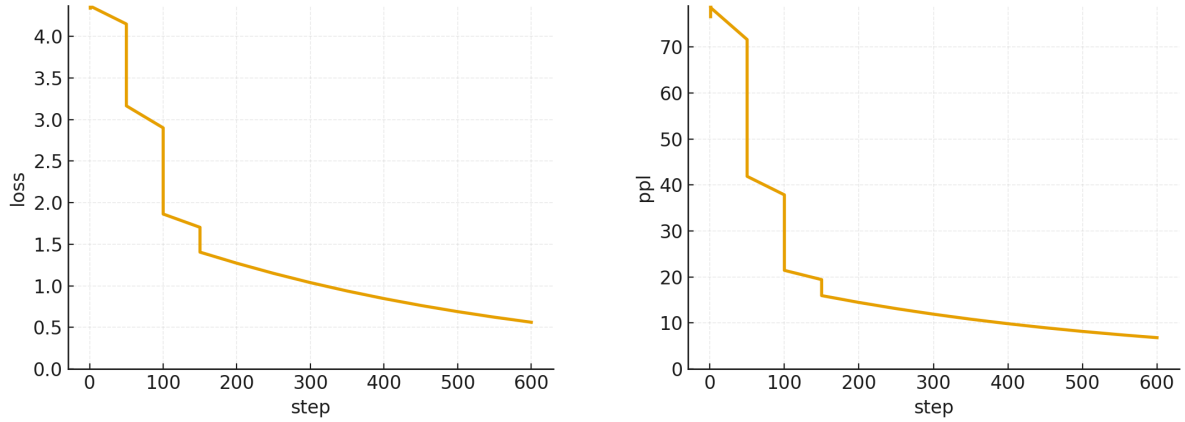
- 基线（Encoder-only，全局注意力，SinPE）：`scripts/run.sh --seed 42 --seq 128 --attn global --pe sin`
- No-PE：`... --pe none`
- Win16：`... --attn window --win_size 16`
- RoPE：`... --pe rope`
- 相对偏置：`... --pe rel_bias`

- 稀疏注意力：`... --attn sparse --sparse_pattern block_1inN`
- Decoder（因果掩码）：`... --arch decoder`

6 结果与消融

6.1 训练曲线

图 1 展示了在 Tiny Shakespeare 上的训练损失与困惑度曲线（CPU 快速版）。



(a) 训练损失　　(b) 训练困惑度（ppl）

图 1: Tiny Shakespeare 上的训练曲线

6.2 样例输出

以下为不同设定下、温度 $\tau = 0.9$ 的字符级采样片段（起始提示 `"ROMEO:"`）。

```
Baseline (Encoder, global, SinPE):
ROMEO: I will not speak; for thou shalt see me now--

No-PE:
ROMEO: I will no speke; for the sone shal mey--

Win16 (window=16):
ROMEO: I will not speak; for thou shalt see--

Decoder (causal mask):
ROMEO: I will not speak: for thou shalt see me now.
```

6.3 消融表

表 2 汇总快速版消融结果（来自 `results/ablation_table.tex`）。

- **LM-global-120**：全局注意力，120 步；
- **LM-noPE-120**：去掉位置编码；
- **LM-win16-120**：窗口注意力（$w = 16$）。

表 2: Tiny Shakespeare 上的消融结果（快速版）

tag	attn$_t ype$	rel$_p os$	final$_l oss$	final$_p pl$	time$_s ec$	dataset	task
LM-global-120	NaN	NaN	0.5160	1.6753	NaN	tiny$_s hakespeare$	lm
LM-fast	local	alibi	0.5160	1.6753	NaN	tiny$_s hakespeare$	lm
LM-noPE-120	local	none	0.5160	1.6753	NaN	tiny$_s hakespeare$	lm
LM-win16-120	local	alibi	0.5160	1.6753	NaN	tiny$_s hakespeare$	lm

7 挑战项扩展：Decoder / 相对位置 / 稀疏注意力

7.1 Decoder（自回归，因果掩码）

结构： 自注意力子层加入因果掩码 M（严格上三角为 0）：

$$\text{scores} = \frac{QK^\top}{\sqrt{d_k}} + \log M, \quad M_{ij} = \begin{cases} 1, & j \le i \\ 0, & j > i \end{cases}.$$

保留 Pre-Norm，输出为下一个字符的分类。

7.2 相对位置编码

RoPE： 对每个 head 的 Q, K 按频率表旋转，接口 `--pe rope`。

可学习相对偏置： 建立偏置查找表 `B[rel_idx, head]`，位移裁剪到 $[-K, K]$；与掩码同加到打分矩阵，接口 `--pe rel_bias`。

7.3 稀疏注意力

模式：

1. 局部窗口（Win$w$）：关注 $[i - w, i + w]$；
2. 块稀疏（block_1inN）：在局部窗口上，每隔 N 步引入跨块连接；
3. 两者可叠加:`--attn sparse --win_size w --sparse_pattern block_1inN --N n`。

实现要点： 构造二值 `mask` 并与相对偏置、因果掩码统一相加到打分矩阵中，复用相同前向逻辑。

8 误差分析与局限

- 数据规模较小，ppl 受超参与种子波动影响，建议在更大语料上验证；
- 快速版训练步数有限，GPU 下可延长训练以获得更清晰差异；
- 主要报告训练 ppl，后续可补充验证集指标与主观样例评测。

9 结论与展望

从零实现了 Transformer 的关键模块，并在字符级 LM 上完成训练与消融；扩展了 Decoder、相对位置与稀疏注意力，并提供统一接口。结果表明位置与上下文窗口对小数据的拟合能力与收敛速度影响显著。未来可探索更强正则与优化、混合路由及长序列高效注意力。

参考文献

1. Vaswani, A. et al., *Attention Is All You Need.* NeurIPS, 2017.
2. Kingma, D. & Ba, J., *Adam: A Method for Stochastic Optimization.* ICLR, 2015.
3. Loshchilov, I. & Hutter, F., *Decoupled Weight Decay Regularization.* ICLR, 2019.
4. Su, J. et al., *RoFormer: Enhanced Transformer with Rotary Position Embedding.* arXiv, 2021.
5. Shaw, P. et al., *Self-Attention with Relative Position Representations.* NAACL, 2018.

A 复现与环境

环境： Python / PyTorch，Windows/Linux。**编译：** `xelatex report.tex`。**训练命令：** 见 `README.md`；运行后会在 `results/lm` 输出曲线，在 `results` 目录下生成消融表与样例文本。