CS336 作业 1 任务总览

2025 年春季

1 总体目标

本次作业要求从零开始实现并训练一个标准的 Transformer 语言模型 (LM)，覆盖数据预处理、模型组件、训练基础设施以及一系列实验。所有代码需写在仓库提供的框架内，‘adapters.py‘ 中仅填充粘合作用，测试文件不得修改。

2 分任务与细节要求

2.1 Unicode 与 BPE 分词器

- **unicode1 (1 分)**：通过 `chr` 与 `repr` 分析 `chr(0)` 的表现形式及其在字符串中的影响。
- **unicode2 (3 分)**：比较 UTF-8/16/32，解释为何选用 UTF-8；找出错误 UTF-8 解码函数的反例；举例说明无法映射到 Unicode 的字节序列。
- **train_bpe (15 分)**：实现字节级 BPE 训练，输入文本路径、最大词表大小及特殊符号列表，输出 `vocab: dict[int, bytes]` 与 `merges: list[tuple[bytes, bytes]]`。需：
  - 预分词使用给定 GPT-2 正则表达式，使用 `re.finditer`；在预分词前剥离所有特殊符号，确保不跨越文档边界；可并行化预分词，建议参考 `pretokenization_example.py`。
  - 词元对统计需增量维护，仅更新受影响的 pair；频率相同时按字节序字典序最大者合并。
- **train_bpe_tinystories (2 分)**：在 TinyStories (含 `<|endoftext|>`) 上训练 10K 词表，记录用时、内存及最长 token；分析主要耗时步骤。
- **train_bpe_expts_owt (2 分)**：在 OpenWebText 训练 32K 词表，回答最长 token 及与 TinyStories 词表差异。
- **tokenizer (15 分)**：实现可从文件加载/保存的 Tokenizer，支持特殊符号；提供 `encode`、`encode_iterable`（流式输出）、`decode`，编码时逐 pre-token 应用 merges，解码要用 U+FFFD 处理非法序列。
- **tokenizer_experiments (4 分)**：

- 抽样 TinyStories 与 OWT 文档，比较各自分词器压缩率 (bytes/token)。
- 用 TinyStories 分词器处理 OWT 样本，讨论压缩率与质性差异。
- 估算分词吞吐量并推算标注 The Pile 所需时间。
- 将两数据集编码为 `uint16` NumPy 序列并解释类型选择。

2.2 Transformer 模型组件

- **linear (1 分)**：自定义无 bias 线性层，参数以 `nn.Parameter` 形式存储，使用截断正态初始化；适配器需加载提供的权重。
- **embedding (1 分)**：实现自定义嵌入层，矩阵形状 (vocab, d_{model})，使用相同初始化策略。
- **rmsnorm (1 分)**：实现 RMSNorm，前向需先转 float32 计算，再转换回输入 dtype。
- **positionwise_feedforward (1 分)**：实现 SwiGLU 前馈网络，内层维度约为 $\frac{8}{3}d_{\text{model}}$（且为 64 的倍数），包含门控与 SiLU，产出回到 d_{model}。
- **rope (2 分)**：实现 RotaryPositionalEmbedding，可预先注册正弦余弦 buffer，支持任意批次维度和位置索引切片。
- **softmax (1 分)**：实现数值稳定 softmax，按指定维度减去最大值。
- **scaled_dot_product_attention (5 分)**：支持任意批次维度、可选 mask（False 处概率为 0），输出维度匹配值向量。
- **multihead_self_attention (5 分)**：实现多头自注意力，包含 Q/K/V 线性映射、RoPE、因果 mask 以及输出线性层，注意张量重排与批次兼容。
- **transformer_block (3 分)**：实现预归一化块：输入先 RMSNorm 再自注意力并残差，加第二个 RMSNorm + FFN 残差。
- **transformer_lm (3 分)**：串联嵌入、多个块、最终 RMSNorm 与输出线性层，前向返回 (batch, seq, vocab) logits。需通过提供的所有模块测试。
- **transformer_accounting (5 分)**：对 GPT-2 系列 (Small/Medium/Large/XL) 分析参数量、FLOPs、内存占用与主要计算瓶颈；探讨将上下文增至 16,384 的影响。

2.3 训练基础设施

- **cross_entropy (1 分)**：实现数值稳定的交叉熵，支持任意 batch 维并返回平均 loss。
- **adamw (2 分)**：继承 `torch.optim.Optimizer` 实现 AdamW，维护一阶/二阶动量和步数，包含解耦权重衰减；**adamwAccounting (2 分)**：

  - 推导参数/激活/梯度/优化器状态内存表达式，并代入 GPT-2 XL 估算最大可行 batch；
  - 计算一步 AdamW 的 FLOPs；
  - 假设 MFU=50%，估计单张 A100 训练 400K 步 (batch 1024) 所需时间。

- **learning_rate_schedule (1 分)**：实现带线性 warmup 与余弦退火的学习率函数。
- **gradient_clipping (1 分)**：实现全局梯度范数裁剪 (阈值 M，$\epsilon = 10^{-6}$)。

2.4 训练循环与推理

- **data_loading (2 分)**：从 token ID 序列中随机采样 (batch, context) 对及其下一 token 目标，支持 CPU/MPS/GPU；大型文件需使用 `np.memmap`。
- **checkpointing (1 分)**：保存/恢复模型、优化器、调度器状态以及训练进度。
- **training_together (4 分)**：整合分词数据加载、模型、优化器、LR 调度、梯度裁剪、loss 记录与验证。
- **decoding (3 分)**：实现生成接口，支持自定义 prompt、最大生成长度、温度缩放以及 top-p (nucleus) 采样。
- **experiment_log (3 分)**：构建实验追踪与日志系统，记录训练/验证 loss 随步数及墙钟时间的变化，方便附加学习曲线。

2.5 TinyStories 实验

- 基准设置：词表 10K、上下文 256、$d_{\text{model}} = 512$、$d_{ff} = 1344$、4 层 16 头、RoPE $\Theta = 10000$，总 token 数约 3.2768×10^8（batch × steps × context）。
- **learning_rate (3 分)**：在 TinyStories 上进行学习率搜索，绘制多条曲线，并得到验证 loss ≤ 1.45（CPU/MPS 可放宽至 40M token、loss 2.0）。
- **batch_size_experiment (1 分)**：分析批大小对收敛与效率的影响。
- **generate (1 分)**：展示模型生成的 TinyStories 文本例子。
- **layer_norm_ablation (1 分)**：移除 RMSNorm 再训练并讨论影响。
- **pre_norm_ablation (1 分)**：实现后归一化 (post-norm) 版本并比较学习曲线。
- **no_pos_emb (1 分)**：移除 RoPE (NoPE) 与基线对比。
- **swiglu_ablation (1 分)**：用 SiLU FFN (内层 $4d$) 替换 SwiGLU，比较性能。

2.6 OpenWebText 与 Leaderboard

- **main_experiment (2 分)**：在 OWT 上以与 TinyStories 相同架构/训练预算训练，给出损失曲线、与 TinyStories 的 loss 差异解释，以及 OWT 生成示例。
- **leaderboard (6 分)**：在 1.5 H100 小时内（固定数据源）自由改进模型或训练策略，提交验证 loss ≤ 5.0 的结果至课程排行榜，并提供曲线与方法说明。

3 资源与实现建议

- 优化预分词与 BPE 训练可使用 `multiprocessing`，并结合 `cProfile` 等分析瓶颈。
- 模型实现过程中推荐使用 `einops` 或 `einx` 提升可读性与批处理兼容性。
- 对于 CPU/MPS 训练需选取匹配的设备字符串，必要时缩小数据/模型规模；大数据集建议使用 `np.memmap`。
- 调试技巧：尝试过拟合单个 mini-batch、监控激活/权重/梯度范数、使用断点检查张量形状。

4 提交要求

- 课堂提交：上传 `writeup.pdf`（排版书面问题回答）与 `code.zip`（全部实现）。
- Leaderboard：遵循仓库说明通过 PR 提交结果，确保日志记录清晰、实验可复现。