大规模预训练范式：语料构建、数据治理与系统优化全景

2025 年 10 月 25 日

1 大规模语料构建（CC, The Pile, C4, RefinedWeb）

1.1 语料来源全景与覆盖维度

预训练数据的多样性直接决定模型的语言覆盖范围、知识密度与风格稳健性。代表性开放语料集包括：

- **Common Crawl (CC)**：每月抓取的万亿级网页集合，提供最广泛的主题覆盖，但噪声较大。CC 的子集如 CC-News、CC-Stories 为不同任务裁剪。
- **The Pile**：EleutherAI 构建的 825GB 高质量语料包，由 22 个子语料组成，涵盖学术论文、开源代码、问答论坛和文学作品，强调平衡性与高熵内容。
- **C4 (Colossal Clean Crawled Corpus)**：在 Common Crawl 基础上通过语言识别、重复检测与质量过滤得到的 750GB 英文语料，被 T5 等模型广泛采用。
- **RefinedWeb**：针对 GPT-NeoX-20B 构建的 600GB 高质量网页语料，结合链接权重、URL 过滤和句子级评分以提升知识密度。

在实践中，现实的数据组合通常包含新闻、书籍、代码、对话、百科、社交媒体等多条数据链路，目标是在语言风格、领域知识、编程技能与跨模态描述之间形成平衡。

1.2 语料扩增与地域语言覆盖

构建全球可用的语料库需要兼顾多语种、方言与专业术语：

- **跨语种爬取**：通过站点白名单、语言检测和种子扩散策略获取中文、阿拉伯语、印地语等内容；结合本地新闻站、政府公文、维基百科镜像。
- **协作与众包**：与高校、科研机构合作采集学术论文、教材、课程字幕；通过众包标注收集口语、方言数据。

- **开放数据接口**：利用 arXiv、PubMed、StackExchange、GitHub Archive 等公开 API 持续更新专业语料。

在扩增过程中需关注版权与许可，确保数据遵循 CC 许可、GNU 文档许可或开源协议；对于未明确授权的数据需在合规范围内采样、去标识化或删除。

1.3 分层存储与元数据管理

大规模语料库应在获取阶段写入丰富的元数据以支持后续治理：

- **文档级元数据**：包含来源 URL、抓取时间、语言、字符计数、领域标签、许可证等信息，便于追踪与审计。
- **段落级特征**：使用规则和模型生成毒性评分、可读性指数、主题分布、潜在版权风险标签。
- **版本与快照**：对语料集进行快照管理，记录清洗和采样前后的差异，支持回滚和对比实验。

在工程上常用 HDFS、Delta Lake、Parquet 等格式分级存储，通过 Spark、Ray 或 Dask 批处理以满足 EB 级数据吞吐需求。

2 数据清洗与去重（Deduplication, Filtering）

2.1 多级去重策略

重复内容会放大训练偏差、浪费算力并导致过拟合。常见去重层次：

- **URL 级去重**：基于 URL 或域名的哈希集合快速过滤重复网页。
- **文档级去重**：利用 SimHash、MinHash、LSH Forest 对文档指纹去重，控制语料的局部和全局重复率。
- **片段级去重**：将文档切分为句子或段落，使用 n-gram 指纹或余弦相似度阈值过滤，防止大段重复。

针对学术论文或代码语料，还可配合 AST 抽象语法树匹配与引用网络分析，剔除镜像库或 fork 内容。

2.2 质量过滤与安全治理

清洗流程需要识别噪声、低质量或高风险文本：

- **语言与字符检测**：基于 FastText、CLD3 或字符分布检测非目标语言、乱码和爬虫痕迹。
- **毒性与偏见检测**：使用 Detoxify、Perspective API、self-critique 模型打分，设定阈值或权重抽样，控制模型输出安全性。
- **隐私过滤**：正则匹配与深度模型识别邮箱、身份证、电话号码等敏感字段，并通过模糊化或删除处理。
- **模板与广告识别**：通过页面布局特征、HTML 标签和重复短语识别 SEO 垃圾页面，实现软黑名单。

在生产环境中，数据流水线通常由多级过滤模型串联构成 DAG，结合人类审核样本与反馈闭环持续优化阈值设置。

2.3 加权采样与分布校准

为避免模型被长尾领域或低质量数据污染，需要对清洗后的语料进行加权采样：

- **领域分桶**：依据主题或来源将语料划分为新闻、百科、论坛、代码、学术等分桶，设定目标分布并执行分层抽样。
- **温度采样**：类似重排 softmax 的思想，对得分为 s_i 的文档按照 $p_i = \frac{\exp(s_i/\tau)}{\sum_j \exp(s_j/\tau)}$ 加权，控制高分文档比例。
- **动态再加权**：使用在线指标（如模型困惑度、梯度范数）监测训练进度，针对性增加困难样本或最新数据的采样权重。

在训练过程中，可通过数据缓冲队列和可复现的伪随机种子保证实验可重复性，并记录每次 epoch 的实际分布供分析。

3 并行训练策略（Data / Model / Pipeline / Tensor）

3.1 数据并行：超大批次与同步协议

数据并行通过复制完整模型到多个设备上独立处理不同 minibatch，再同步梯度。关键技术点包括：

- **梯度压缩**：使用 ZeRO、1-bit Adam、GossipGrad 等压缩方法降低通信带宽。
- **同步拓扑**：NCCL AllReduce、RingReduce、树形聚合；在多机多卡场景需要优化拓扑匹配和通信调度。

- **大批次训练**：使用线性学习率预热与 LARS/LAMB 优化器维持收敛稳定性，可达 10M 级 batch。

3.2 模型并行与张量切分

当模型参数无法放入单卡显存时，需采用模型并行：

- **张量并行（Tensor Parallelism）**：将线性层的权重按列或行切分到多张 GPU，Megatron-LM 的 Column/Row Parallel Linear 是典型实现。
- **序列并行**：在自注意力计算中沿序列维度切分，减少激活冗余与通信。
- **ZeRO Stage-3**：将参数、梯度和优化器状态全面分片，结合 NVMe Offload 扩展到 TB 级模型。

为了减少代价，需要平衡通信和计算；常见做法是在张量并行规模和节点内 NVLink 拓扑之间保持一致。

3.3 流水线并行与调度

流水线并行将模型按层划分到多个阶段，按时间片处理微批次：

- **GPipe 与 1F1B（One Forward One Backward）**：通过微批次调度减少气泡时间，保持流水线饱和。
- **异构流水线**：将注意力密集层与前馈层交错分配，针对 GPU/TPU 差异调整阶段长度。
- **检查点兼容**：管理激活重计算与中间状态同步，确保反向传播一致性。

在大规模集群中，流水线并行常与数据、张量并行组成 3D 并行栈，DeepSpeed、Megatron-Deepspeed、Colossal-AI 等框架提供半自动化配置工具。

4 混合精度与梯度检查点（AMP, Checkpointing）

4.1 自动混合精度（AMP）策略

AMP 通过在保持数值稳定的前提下使用半精度计算，显著降低显存占用与计算成本：

- **静态混合精度**：人工指定半精度与单精度算子，例如 NVIDIA Apex 的 O1/O2 模式。

- **动态混合精度**：PyTorch AMP、TensorFlow Keras Mixed Precision 自动管理计算精度和缩放因子。
- **损失缩放**：使用动态损失缩放避免半精度下的梯度下溢，结合 GradScaler 自动调节。

混合精度需要确保归一化、softmax、层归一等敏感算子保持更高精度，以防梯度爆炸或模式崩溃。

4.2 梯度检查点与激活管理

梯度检查点通过在前向传播时舍弃部分激活，在反向阶段重新计算以换取显存：

- **层级检查点**：将 Transformer Block 作为单位进行 checkpoint，减少约 50% 激活存储。
- **自定义分段**：结合长序列注意力或 MoE 层，针对高成本算子单独设置 checkpoint。
- **Recompute Schedules**：自适应控制重计算层次数，在训练后期降低重计算频率以节省时间。

与 ZeRO Offload、Paged Optimizer 等方案搭配时，需要在训练框架中显式设置检查点策略，以避免重复传输。

4.3 稳定性与验证

混合精度和检查点策略引入新的数值风险，需要配套验证：

- **数值监控**：跟踪梯度范数、损失漂移、NaN/Inf 比率，并记录 AMP 缩放因子的曲线。
- **回归测试**：在小规模数据集上与 FP32 全精度结果对比，验证收敛速度和最终性能差异。
- **服务一致性**：确保训练与推理精度配置一致，避免量化或裁剪引发兼容性问题。

最终，配合自动化实验管理（Weights & Biases、MLflow）和日志审计，可对大规模预训练进行全生命周期追踪与复现。

参考文献

- Gao et al. “The Pile: An 800GB Dataset of Diverse Text for Language Modeling.” arXiv, 2020.

- Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.” JMLR, 2020.
- Smith et al. “Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B.” arXiv, 2022.
- Penedo et al. “The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only.” arXiv, 2023.
- Shoeybi et al. “Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism.” arXiv, 2019.