PR #7197 完整报告

verl-project/verl

[algo, cfg] feat: add token-sum loss aggregation

合并时间：2026-08-04 11:02

原文链接：http://prhub.com.cn/verl-project/verl/pull/7197

执行摘要

- 一句话：新增 token-sum 损失聚合模式，支持 DP 缩放
- 推荐动作：该 PR 代码量小、逻辑直白，但其中“乘 dp_size 抵消 DDP/FSDP 平均”的推导值得精读，是一个典型的并行训练损失语义设计。建议阅读 agg_loss 全函数和三个测试用例，理解 token-sum 与 token-mean 在梯度尺度上的差异。对于计划在多卡或 Megatron 后端使用该模式的团队，建议等待后续专门适配或自行验证。

功能与动机

PR body 明确指出，新增 token-sum 聚合模式的目标是让 DDP/FSDP 的 mean-reduced 梯度等于全局 token 之和。作者还说明，这是从原先范围过大的混合变更中拆分出的单一可审查特性，其他如 importance-sampling/DRO 和标准 PPO 调整将放入独立 PR。

实现拆解

1. 核心逻辑（verl/trainer/ppo/core_algos.py）：在 agg_loss 中新增 elif loss_agg_mode == "token-sum" 分支，实现为 verl_F.masked_sum(loss_mat, loss_mask) * dp_size。原理是 DDP/FSDP 的 AllReduce 默认对梯度做平均，每个 rank 的本地损失乘以 dp_size 后，平均结果恰好等于全局 batch 内所有有效 token 的损失总和。该分支不需要 batch_num_tokens 或 global_batch_size，实现最简。
2. 配置接入（verl/workers/config/actor.py、verl/trainer/config/actor/actor.yaml）：在 ActorConfig.__post_init__的 valid_loss_agg_modes 列表中加入 "token-sum"，并同步更新 docstring 与 yaml 注释，保证用户可以通过 actor_rollout_ref.actor.loss_agg_mode=token-sum 启用。
3. 测试配套（tests/trainer/ppo/test_loss_aggregation_on_cpu.py）：新增 CPU 测试，覆盖三点：① mask 正确性 + dp_size 缩放；② 该模式对 micro-batch 切分不变（即分批聚合结果与整体聚合一致）；③ 模拟 FSDP 的 rank-level mean 归约后，结果等于全局 token sum。这些测试不依赖 GPU，可在 CI 的 CPU job 中运行。

关键文件：

- verl/trainer/ppo/core_algos.py（模块 损失聚合；类别 source；类型 core-logic；符号 agg_loss）：核心实现文件，在 agg_loss 中新增 token-sum 分支，这是整个 PR 的功能主体。

- verl/workers/config/actor.py（模块 演员配置；类别 source；类型 core-logic；符号 ActorConfig.post_init）：配置校验入口，决定 token-sum 是否能被用户配置接受。
- verl/trainer/config/actor/actor.yaml（模块 配置文档；类别 config；类型 configuration）：配置文件注释同步更新，作为用户文档的一部分。
- tests/trainer/ppo/test_loss_aggregation_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 test_token_sum_masks_tokens_and_scales_for_dp, test_token_sum_is_microbatch_invariant, test_token_sum_matches_global_sum_after_fsdp_mean_reduction）：新增 CPU 测试，验证 token-sum 的掩码、微批不变性和 FSDP 平均等价性，是功能正确性的重要保障。

关键符号：agg_loss, ActorConfig.post_init

关键源码片段

verl/trainer/ppo/core_algos.py

核心实现文件，在 agg_loss 中新增 token-sum 分支，这是整个 PR 的功能主体。

```
# verl/trainer/ppo/core_algos.py 中 agg_loss 的聚合分支

def agg_loss(loss_mat, loss_mask, loss_agg_mode, dp_size=1,
             batch_num_tokens=None, global_batch_size=None, loss_scale_factor=None):
    # token-mean：按全局有效 token 数归一化，乘 dp_size 抵消 FSDP/DDP 平均
    if loss_agg_mode == "token-mean":
        if batch_num_tokens is None:
            if dp_size > 1:
                raise ValueError("(global) batch_num_tokens is required when dp_size > 1")
            batch_num_tokens = loss_mask.sum()
        loss = verl_F.masked_sum(loss_mat, loss_mask) / batch_num_tokens * dp_size

    # 新增 token-sum 分支：本 rank 只求掩码 token 之和，乘 dp_size 后
    # 经 AllReduce mean 即可得到全局 token 总和，无需额外的 batch 元信息
    elif loss_agg_mode == "token-sum":
        loss = verl_F.masked_sum(loss_mat, loss_mask) * dp_size

    # seq-mean-token-sum 系列：先逐序列求和，再按序列数平均
    elif loss_agg_mode in ["seq-mean-token-sum", "seq-mean-token-sum-norm"]:
        seq_losses = torch.sum(loss_mat * loss_mask, dim=-1) # 每个序列的 token 求和
        seq_mask = (torch.sum(loss_mask, dim=-1) > 0).float() # 剔除全被掩码的序列
        if global_batch_size is None:
            if dp_size > 1:
                raise ValueError("global_batch_size is required when dp_size > 1")
            global_batch_size = seq_mask.sum()
        loss = verl_F.masked_sum(seq_losses, seq_mask) / global_batch_size * dp_size
        if loss_agg_mode == "seq-mean-token-sum-norm":
            if loss_scale_factor is None:
                horizon = loss_mask.shape[-1]
                loss_scale_factor = horizon
            loss /= loss_scale_factor
```

```
    # seq-mean-token-mean：每个序列先做 token 平均，再对所有序列平均
    elif loss_agg_mode == "seq-mean-token-mean":
        seq_mask = torch.sum(loss_mask, dim=-1) # 每序列有效 token 数
        seq_losses = torch.sum(loss_mat * loss_mask, dim=-1) / (seq_mask + 1e-8)
        seq_mask = (seq_mask > 0).float()
        if global_batch_size is None:
            if dp_size > 1:
                raise ValueError("global_batch_size is required when dp_size > 1")
            global_batch_size = seq_mask.sum()
        loss = verl_F.masked_sum(seq_losses, seq_mask) / global_batch_size * dp_size
    else:
        raise ValueError(f"Invalid loss_agg_mode: {loss_agg_mode}")

    return loss
```

tests/trainer/ppo/test_loss_aggregation_on_cpu.py

新增 CPU 测试，验证 token-sum 的掩码、微批不变性和 FSDP 平均等价性，是功能正确性的重要保障。

```
# tests/trainer/ppo/test_loss_aggregation_on_cpu.py 的代表性用例

def test_token_sum_masks_tokens_and_scales_for_dp():
    # loss_mat 形状为 (bs, response_length)，loss_mask 标记有效 token
    loss_mat = torch.tensor([[1.0, 2.0, 30.0], [4.0, 50.0, 6.0]])
    loss_mask = torch.tensor([[1.0, 1.0, 0.0], [1.0, 0.0, 1.0]])

    loss = agg_loss(loss_mat, loss_mask, loss_agg_mode="token-sum", dp_size=4)

    # 有效 token 求和为 1+2+4+6 = 13，再乘以 dp_size 4
    assert loss.item() == pytest.approx(13 * 4)


@pytest.mark.parametrize("dp_size", [2, 4])
def test_token_sum_matches_global_sum_after_fsdp_mean_reduction(dp_size):
    loss_mat = torch.arange(1, 25, dtype=torch.float32).reshape(8, 3)
    loss_mask = torch.tensor([[1.0, 1.0, 0.0]] * 8)
    rank_step = loss_mat.shape[0] // dp_size

    # 模拟各 rank 只看到部分数据，且各自乘上 dp_size
    rank_losses = [
        agg_loss(loss_mat[i : i + rank_step], loss_mask[i : i + rank_step],
                 loss_agg_mode="token-sum", dp_size=dp_size)
        for i in range(0, loss_mat.shape[0], rank_step)
    ]
    # FSDP 的 AllReduce mean 等价于对 rank_losses 求平均
    fsdp_reduced = torch.stack(rank_losses).mean()
    # 全局 token sum 直接对完整 loss_mat 聚合
    global_sum = agg_loss(loss_mat, loss_mask, loss_agg_mode="token-sum")
```

```
torch.testing.assert_close(fsdp_reduced, global_sum)
```

评论区精华

该 PR 没有实质性的 review 评论。唯一的提交记录显示 reviewer Luosuu 直接批准（APPROVED）。PR body 中作者主动披露了两点：其一，本 PR 是从一个范围更大的变更中拆分而来，只保留 token-sum 这一特性；其二，使用了 OpenAI Codex 辅助拆分和整理测试，并承诺合并前人工复核所有改动并重跑测试。这种"AI 辅助 + 人工复核 + 缩小范围"的协作方式值得团队参考。

- 暂无高价值评论线程

风险与影响

- 风险：梯度尺度风险：token-sum 模式下损失值随全局 token 数线性增长，不再被归一化，可能导致梯度范数显著大于 token-mean 模式，用户直接切换且不调整学习率时可能出现训练不稳定，需在文档中提示。后端适配风险：当前实现仅乘以 dp_size，对 FSDP/DDP 平均梯度场景语义正确；Megatron 后端在 PP 调度下还需要考虑 num_microbatches 和 cp_size 的额外缩放（agg_loss docstring 已有说明），Megatron 用户使用该模式时需自行验证或后续补充专门处理。测试覆盖局限：新测试全部在 CPU 上运行，只验证了前向聚合的数学等价性，未覆盖真实分布式反向传播和不同并行配置下的端到端行为，也缺少对非法 dp_size=1 与 token-sum 组合的边界检查。
- 影响：对用户：新增一个可选配置项，默认行为不变，不影响现有训练流程。对系统：loss 聚合函数分支增加，仅在显式选择 token-sum 时生效，不会改变其他模式语义。对团队：提供一种新的梯度聚合语义，可能用于未来强调 token 总损失而非均值的算法（如某些 DRO 或 importance-sampling 场景），并为后续相关 PR 打下基础。
- 风险标记：新聚合模式未归一化可能影响梯度尺度，缺少 Megatron 后端验证，测试仅覆盖 CPU 前向

关联脉络

- PR #7225 [algo] fix: micro-batch normalization for distillation loss: 同属损失语义相关改动，涉及 loss 在并行训练中的归一化 / 聚合方式；前者修正蒸馏损失的微批归一化，后者新增 token 求和聚合模式，共同体现 verl 在训练损失数值语义上的持续打磨。