PR #2102 完整报告

THUDM/slime

Support top_p mask

合并时间：2026-06-19 09:56

原文链接：http://prhub.com.cn/THUDM/slime/pull/2102

执行摘要

- 一句话：新增 top_p 掩码支持提升 RL 训练准确性
- 推荐动作：该 PR 设计清晰，值得关注其在分布式训练中传递辅助数据的方法。建议阅读 _build_topp_keep_mask 和 compute_log_probs 的改动，理解如何与 TP/CP 对齐。

功能与动机

在 RL 训练中，当 rollout 采样使用 top_p (nucleus sampling) 时，只有部分 top 概率 token 被考虑。如果训练时计算所有 token 的 log-prob，会导致策略梯度与 rollout 时的采样分布不一致，影响训练效果。该 PR 通过记录和传输 top_p 选中的 token ids，并在训练时仅计算这些 token 的 log-prob，从而对齐 rollout 与训练的概率空间。

实现拆解

1. SGLang 补丁：在 *decode* 阶段记录每个生成 token 的 top-p 候选 token ids，通过 PD 传输通道带回。
2. Rollout 数据处理：新增 _decode_int32_meta_array 等函数解析服务器返回的二进制 / 编码结构，提取 token ids 和 offsets。
3. 训练前馈适配：在 forward_only 中通过 _with_rollout_top_p_token_keys 动态扩展 batch 键，将 top-p 数据传递给损失函数。
4. 掩码构建：_build_topp_keep_mask 基于 TP rank 划分词汇区间，为每个 response token 构建布尔 keep_mask；_fill_topp_mask_rows 填充具体候选位置。
5. Log-prob 计算：compute_log_probs 接受 keep_mask，将非候选位置 logits 置为 -inf，确保 cross-entropy 仅考虑候选 token。
6. 参数与类型：新增 rollout_top_p 参数，在 Sample 类型中增加 rollout_top_p_token_ids/rollout_top_p_token_offsets 字段。
7. 测试覆盖：新增 test_logprob_response_spans.py 验证 cp1、zigzag-cp、allgather-cp 模式下掩码正确性。

关键文件：

- slime/rollout/sglang_rollout.py（模块 rollout 层；类别 source；类型 core-logic；符号 _decode_int32_meta_array, _extract_rollout_top_p_token_data, _merge_rollout_top_p_token_data, _append_rollout_top_p_token_data）：新增 top-p 元

数据解码、提取、合并和追加函数，是 rollout 端数据处理的入口。

- slime/backends/megatron_utils/loss.py（模块 损失函数；类别 source；类型 core-logic；符号 get_rollout_top_p_logprob_kwargs, _fill_topp_mask_rows, _build_topp_keep_mask）：新增 top-p 掩码构建函数和参数提取函数，是训练端 logits 屏蔽的核心。
- docker/patch/latest/sglang-top_p.patch（模块 SGLang 补丁；类别 test；类型 test-coverage；符号 DisaggregationMode, _forward_ascend_backend, _attach_logprobs_to_output, get_token_ids_logprobs_raw）：新增 SGLang 补丁，使服务器端输出 top-p 候选 token ids，是整个流程的基础。
- slime/backends/megatron_utils/model.py（模块 训练流程；类别 source；类型 data-contract；符号 _with_rollout_top_p_token_keys）：修改 forward_only 以传递 top-p 参数，连接 rollout 与 loss 计算。
- tests/test_logprob_response_spans.py（模块 测试；类别 test；类型 test-coverage；符号 _set_cp, _kept_ids, test_top_p_mask_aligns_with_zigzag_cp_response_rows, test_top_p_mask_aligns_with_allgather_cp_response_rows）：新增测试覆盖三种 CP 模式下 top-p 掩码的正确性，保证核心逻辑可靠。
- slime/utils/ppo_utils.py（模块 PPO 工具；类别 source；类型 core-logic；符号 compute_log_probs, calculate_log_probs_and_entropy）：修改 compute_log_probs 和 calculate_log_probs_and_entropy 以接受 keep_mask，是掩码落地的关键。
- slime/ray/rollout.py（模块 Ray 调度；类别 source；类型 core-logic；符号 _compute_top_p_kept_vocab_metrics）：新增 top-p 相关指标计算，用于监控。
- slime/rollout/sglang_streaming_rollout.py（模块 流式 rollout；类别 source；类型 dependency-wiring）：适配 top-p 数据流，与 sglang_rollout 保持一致。
- slime/utils/types.py（模块 类型定义；类别 source；类型 core-logic）：Sample 类型新增 top-p 字段，定义数据结构。
- slime/backends/megatron_utils/actor.py（模块 Actor 模块；类别 source；类型 core-logic）：小修改以传递 top-p 参数。
- slime/backends/megatron_utils/data.py（模块 数据加载；类别 source；类型 core-logic）：数据加载适配 top-p 字段。
- tests/test_sample.py（模块 测试；类别 test；类型 test-coverage）：更新 Sample 测试用例以包含新字段。

关键符号：_decode_int32_meta_array, _extract_rollout_top_p_token_data, _merge_rollout_top_p_token_data, _append_rollout_top_p_token_data, get_rollout_top_p_logprob_kwargs, _fill_topp_mask_rows, _build_topp_keep_mask, _with_rollout_top_p_token_keys, compute_log_probs, calculate_log_probs_and_entropy, _compute_top_p_kept_vocab_metrics

关键源码片段

slime/rollout/sglang_rollout.py

新增 top-p 元数据解码、提取、合并和追加函数，是 rollout 端数据处理的入口。

```
import numpy as np
import pybase64

def _decode_int32_meta_array(meta_info: dict[str, Any], keys: tuple[str, ...]) -> list[int] | None:
    # 尝试从多个 key 中获取 top-p 数据，兼容不同命名
    for key in keys:
        if key in meta_info:
            value = meta_info[key]
            break
    else:
        return None

    if value is None:
        return None
    if isinstance(value, str):
        value = pybase64.b64decode(value.encode("ascii"))
    if isinstance(value, bytes):
        return np.frombuffer(value, dtype=np.int32).tolist()
    if isinstance(value, np.ndarray):
        return value.astype(np.int32, copy=False).tolist()
    return [int(x) for x in value]


def _extract_rollout_top_p_token_data(
    meta_info: dict[str, Any],
    *,
    expected_num_tokens: int | None = None,
) -> tuple[list[int], list[int]] | None:
    # 解析 token ids 和 offsets，校验一致性
    token_ids = _decode_int32_meta_array(meta_info, _TOP_P_TOKEN_ID_META_KEYS)
    offsets = _decode_int32_meta_array(meta_info, _TOP_P_TOKEN_OFFSET_META_KEYS)
    if token_ids is None and offsets is None:
        return None
    if token_ids is None or offsets is None:
        raise ValueError("SGLang top-p token replay must include both token ids and offsets.")
    if not offsets or offsets[0] != 0:
        raise ValueError(f"SGLang top-p token offsets must start with 0, got {offsets[:1]}.")
    if offsets[-1] != len(token_ids):
        raise ValueError(f"SGLang top-p token ids/offsets mismatch: offsets[-1]={offsets[-1]},
        len(token_ids)={len(token_ids)}.")
    if expected_num_tokens is not None and len(offsets) != expected_num_tokens + 1:
        raise ValueError(
            "SGLang top-p token offsets length must equal generated token count + 1: "
            f"len(offsets)={len(offsets)}, generated={expected_num_tokens}."
        )
    return token_ids, offsets
```

slime/backends/megatron_utils/loss.py

新增 top-p 掩码构建函数和参数提取函数，是训练端 logits 屏蔽的核心。

```
def _build_topp_keep_mask(
    T: int,
    vocab_local: int,
    device: torch.device,
    top_p_token_ids: list[list[int]],
    top_p_token_offsets: list[list[int]],
    total_lengths: list[int],
    response_lengths: list[int],
    allgather_cp: bool,
) -> torch.Tensor:
    # 构建 [T, vocab_local] 布尔掩码，屏蔽非 top-p 候选 token
    cp_size = mpu.get_context_parallel_world_size()
    tp_rank = mpu.get_tensor_model_parallel_rank()
    vocab_start = tp_rank * vocab_local
    vocab_end = vocab_start + vocab_local

    # 统一为 Python list 方便索引
    top_p_token_ids = [t.tolist() if torch.is_tensor(t) else list(t) for t in top_p_token_ids]
    top_p_token_offsets = [t.tolist() if torch.is_tensor(t) else list(t) for t in top_p_token_offsets]

    keep = torch.ones((T, vocab_local), dtype=torch.bool, device=device)

    if cp_size > 1 and not allgather_cp:
        # zigzag CP：每个 rank 持有交替 chunk
        local_base = 0
        for ids, offsets, total_length, response_length in zip(
            top_p_token_ids, top_p_token_offsets, total_lengths, response_lengths, strict=False
        ):
            prompt_length = total_length - response_length
            chunk_size_cp, chunks_offset, logits_offset, tokens_offset = get_logits_and_tokens_
            offset_with_cp(
                total_length, response_length
            )
            for half, base in ((0, local_base), (1, local_base + chunk_size_cp)):
                local_start = base + (logits_offset[half][0] - chunks_offset[half][0])
                length = logits_offset[half][1] - logits_offset[half][0]
                _fill_topp_mask_rows(
                    keep, ids, offsets,
                    response_start=0, local_start=local_start, length=length,
                    vocab_start=vocab_start, vocab_end=vocab_end
                )
            local_base += 2 * chunk_size_cp
    # 其他情况（cp1 或 allgather_cp）类似，此处省略
    return keep
```

slime/utils/ppo_utils.py

修改 compute_log_probs 和 calculate_log_probs_and_entropy 以接受 keep_mask，是掩码落地的关键。

```
def compute_log_probs(
    logits: torch.Tensor,
    tokens: torch.Tensor,
    process_group: dist.ProcessGroup | None,
    keep_mask: torch.Tensor | None = None,
):
    # 如果提供了 keep_mask，先强制保留 sampled token 所在 shard 的位置
    if keep_mask is not None:
        from megatron.core import mpu
        keep_mask = keep_mask.clone()
        vocab_local = keep_mask.size(-1)
        vocab_start = mpu.get_tensor_model_parallel_rank() * vocab_local
        local_tokens = tokens - vocab_start
        on_shard = (local_tokens >= 0) & (local_tokens < vocab_local)
        rows = torch.nonzero(on_shard, as_tuple=False).squeeze(-1)
        if rows.numel() > 0:
            keep_mask[rows, local_tokens[rows]] = True
        # 将非候选位置 logits 置为 -inf，确保 cross-entropy 只计算候选 token
        logits = logits.masked_fill(~keep_mask, float("-inf"))

    # 原始 cross-entropy 计算
    from megatron.core.fusions.fused_cross_entropy import fused_vocab_parallel_cross_entropy
    logits = logits.unsqueeze(1)
    tokens = tokens.unsqueeze(1)
    log_probs = fused_vocab_parallel_cross_entropy(logits, tokens, process_group)
    return log_probs
```

评论区精华

该 PR 未收到 review 评论，合入流程较为直接。

- 暂无高价值评论线程

风险与影响

- 风险： 1) SGLang 补丁引入外部依赖，需保持与上游同步，若上游重构可能导致维护成本。 2) top-p token ids 传输增加网络开销，尤其在词汇表大且保留 token 多时可能成为瓶颈。 3) TP 环境下 keep_mask 构建依赖正确的 vocab_start/vocab_end 计算，切分错误会导致 masked-logloss 不准确。 4) 新字段在非 top-p 场景下需保证向后兼容，目前当 rollout_top_p == 1.0 时跳过。
  - 影响：对用户：启用 top_p < 1.0 后训练自动获得更准确的梯度估计，无需额外配置。对系统：新增 batch 字段及 SGLang 补丁，增加少量内存和网络开销。对团队：需维护 SGLang 补丁并关注其兼容性。
  - 风险标记：核心路径变更，依赖外部补丁，大规模数据传输，分布式一致性

关联脉络

- PR #2088 Add rollout_data_transport nixl: 都涉及 rollout 数据传输机制的改进，可能共享 rollout 数据流架构。
- PR #2072 [docker] upgrade sglang to v0.5.13: SGLang 补丁依赖具体版本，此 PR 升级了 sglang，影响补丁兼容性。
- PR #2082 Overlapping data loading and sglang initialization: 改动 sglang_rollout.py，与当前 PR 的 rollout 数据处理有重叠区域。