PR #2595 完整报告

radixark/miles

[RL] Represent and transport rollout sampling support

合并时间：2026-08-30 14:31

原文链接：http://prhub.com.cn/radixark/miles/pull/2595

执行摘要

- 一句话：CSR 采样掩码表示与 rollout→trainer 传输契约落地
- 推荐动作：值得精读。重点学习两点：一是 codec 中通过 ROLLOUT_SAMPLING_MASK_FIELDS 从默认 allowlist 排除新字段、保持 payload 字节级向后兼容的策略；二是 CSR 掩码完整生命周期管理（append 校验、prefix 截断、merge 补观察 token、reset 清空）。建议合入前确认工具调用路径的强制 token 掩码覆盖，并在 #2596 落地 e2e 后回看数据流完整性。

功能与动机

top-p 支持（即每个响应 token 上采样器实际可输出的 token 集合）是 ragged 的，且随每个生成 token 动态变化。PR body 明确说明：Flattened token IDs plus response-aligned offsets preserve the exact realized support without padding it to [response_length, top_k] during rollout, session storage, or object-store transport。把表示与激活分离，使传输契约可独立审查，同时不留下任何半启用的采样行为。

实现拆解

1. 数据结构层扩展（miles/utils/sampling_mask.py）：RolloutSamplingMask 新增 3 个方法：concatenate（classmethod，按 response 顺序拼接多个 CSR 掩码并重新定位全局偏移）、prefix（按 response_length 截断，返回前 N 个 token 的支持集合）、_as_tensors（只读借出内部 ids/offsets 张量供传输）。它们为后续的强制 token 追加、多轮合并、尾部截断提供统一的 CSR 变换入口，并保证掩码始终与 tokens/response_length 对齐。
2. Sample 类型与生命周期（miles/utils/types.py）：Sample 新增 rollout_sampling_mask: RolloutSamplingMask | None = None 字段；validate() 增加掩码长度与 response_length 对齐的断言；strip_last_output_tokens() 截断 tokens 时同步用 prefix 截断掩码；reset_for_retry() 将掩码重置为 None。任何只改 token 不改掩码的路径都会被校验拦截。
3. 强制 token 与多轮合并辅助（新增 miles/rollout/generate_utils/sampling_mask.py；修改 sample_utils.py、tool_call_utils.py）：append_forced_sampling_tokens() 为环境插入的未采样 token（工具调用、观察 token）记录单例支持，并校验追加前掩码长度与 response_length 对齐；merge_sampling_masks() 合并两轮掩码并在中间插入观察 token 的单例支持，两轮均缺时返回 None、单边缺失抛 ValueError。sample_utils._merge_sample_pair 在合并多轮样本时调用它，且 rollout_sampling_mask

被加入 _REPLAY_FIELDS，使 _introduces_replay_gap 能识别掩码缺口并提前断链；tool_call_utils.py 接入强制 token 的记录路径。

4. session codec 与 trainer 传输（修改 codec.py、train_data_conversion.py）：codec.py 新增 sampling_mask codec——encode 时把掩码拆成 {field}.ids.{i} 与 {field}.offsets.{i} 两个 tensor 打包进 safetensors，decode 时严格校验 dtype（ids 必须 int32、offsets 必须 int64）并重建 RolloutSamplingMask。关键兼容性设计是 ROLLOUT_SAMPLING_MASK_FIELDS 被从默认 COMPUTED_FIELDS/COMPUTED_FIELDS_V2 中排除，未启用采样支持的 session 路径 payload 字节级不变，只有显式传入字段才上 wire。train_data_conversion.py 在 ROLLOUT_DATA_TENSOR_DTYPES 中增加 rollout_sampling_mask_ids（int32）与 rollout_sampling_mask_offsets（int64），convert_samples_to_train_data() 检测到任一样本带掩码时要求整个 batch 都携带，否则抛出带 sample_index 与 status 的 ValueError；_package_shards() 把这两个键纳入 DP 分片。
5. 日志与指标隔离（miles/backends/training_utils/log_utils.py）：将 CSR payload 键从训练日志字典中排除，避免 rollout_sampling_mask_ids/offsets 被打点成指标而被平均采样。

测试配套覆盖 4 个方面：表示层 append/merge/strip 的 CSR 结构断言、codec round-trip 与默认不发送、trainer 转换的透传与 batch 完整性校验、日志排除验证，共 95 个聚焦测试在单张 H200 上通过。

关键文件：

- miles/utils/sampling_mask.py（模块 采样掩码；类别 source；类型 core-logic；符号 concatenate, prefix, _as_tensors）：核心数据结构层：RolloutSamplingMask 新增 concatenate、prefix、_as_tensors，是整条链路所有 CSR 变换的基础。
- miles/rollout/generate_utils/sampling_mask.py（模块 采样辅助；类别 source；类型 core-logic；符号 append_forced_sampling_tokens, merge_sampling_masks）：新增文件，提供环境强制 token 与多轮合并两个核心辅助函数，是采样支持进入 Sample 生命周期的主要入口。
- miles/rollout/session/samples/codec.py（模块 会话编解码；类别 source；类型 data-contract；符号 encode_samples, decode_samples_and_merge_input_sample）：session 端 wire 契约核心：新增 sampling_mask codec，并通过 allowlist 排除策略保证默认 payload 字节级向后兼容。
- miles/ray/rollout/train_data_conversion.py（模块 数据转换；类别 source；类型 data-contract；符号 convert_samples_to_train_data, _package_shards）：trainer 侧传输入口：新增两个 CSR 键、全 batch 完整性校验与 DP 分片透传。
- miles/utils/types.py（模块 样本类型；类别 source；类型 core-logic；符号 Sample.validate, Sample.strip_last_output_tokens, Sample.reset_for_retry）：Sample 数据模型新增字段并同步校验、截断、重置逻辑，保证掩码与 response 逐 token 对齐。
- miles/rollout/generate_utils/sample_utils.py（模块 样本合并；类别 source；类型 core-logic；符号 merge_samples, _merge_sample_pair, _introduces_replay_gap）：多轮样本合并路径接入 merge_sampling_masks，并把采样掩码纳入 replay gap 检测的 _REPLAY_FIELDS。

- tests/fast/rollout/generate_utils/test_sampling_mask.py（模块 采样测试；类别 test；类型 test-coverage；符号 test_forced_tokens_append_singleton_support_and_strip_cleanly, test_merge_sampling_masks_inserts_singleton_observation_supports）：表示层核心测试：验证强制 token 追加、合并与 strip 截断后 CSR 结构与对齐关系。
- miles/backends/training_utils/log_utils.py（模块 日志工具；类别 source；类型 core-logic）：训练日志侧配套：排除 CSR payload 键，避免采样掩码被当成指标平均。

关键符号：append_forced_sampling_tokens, merge_sampling_masks, RolloutSamplingMask.concatenate, RolloutSamplingMask.prefix, RolloutSamplingMask._as_tensors, Sample.validate, Sample.strip_last_output_tokens, Sample.reset_for_retry, merge_samples, convert_samples_to_train_data

关键源码片段

miles/utils/sampling_mask.py

核心数据结构层：RolloutSamplingMask 新增 concatenate、prefix、_as_tensors，是整条链路所有 CSR 变换的基础。

```
# 关键设计：CSR 化表示（扁平 ids + 每 token 偏移）
# 避免把每个 token 的 top-p 支持 padding 成 [response_length, top_k]
@classmethod
def concatenate(cls, masks: Sequence["RolloutSamplingMask"]) -> "RolloutSamplingMask":
    """按 response 顺序拼接完整的 per-token 支持集合。"""
    if not masks:
        return cls(ids=[], offsets=[0])
    if len(masks) == 1:
        return masks[0] # 单元素直接复用，避免无谓拷贝

    ids = torch.cat([mask._ids for mask in masks])
    offsets = [torch.zeros(1, dtype=torch.long)] # 全局首个偏移恒为 0
    id_count = 0
    for mask in masks:
        # 每个 mask 的内部偏移需整体平移 id_count 才能映射到全局坐标
        offsets.append(mask._offsets[1:] + id_count)
        id_count += mask._ids.numel()
    return cls(ids=ids, offsets=torch.cat(offsets))



def prefix(self, response_length: int) -> "RolloutSamplingMask":
    """返回前 ``response_length`` 个 token 的支持，供截断路径使用。"""
    if not 0 <= response_length <= len(self):
        raise ValueError(f"sampling-mask prefix length must be in [0, {len(self)}]")
    if response_length == len(self):
        return self
    id_count = self._offsets[response_length] # 第 response_length 个 token 的结束偏移
    # 切片虽是视图，但构造时 __post_init__ 会做 owned 复制，避免共享 storage
    return type(self)(ids=self._ids[:id_count], offsets=self._offsets[: response_length + 1])
```

```
    def _as_tensors(self) -> tuple[torch.Tensor, torch.Tensor]:
        """借出私有 CSR 张量供即刻只读传输，不复制。"""
        return self._ids, self._offsets
```

miles/rollout/generate_utils/sampling_mask.py

新增文件，提供环境强制 token 与多轮合并两个核心辅助函数，是采样支持进入 Sample 生命周期的主要入口。

```
# 环境强制 token（工具调用、观察 token）不经过采样器，
# 因此在对应响应位置记录 " 单例支持 "（仅允许该 token 本身）。
def append_forced_sampling_tokens(sample: Sample, token_ids: Sequence[int]) -> None:
    """为环境插入的未采样 token 记录单例支持。"""
    sampling_mask = RolloutSamplingMask.from_mask_list([[int(token_id)] for token_id in token_
    ids])
    if sample.rollout_sampling_mask is None:
        # 首次追加必须在任何响应 token 产生之前，否则位置无法对齐
        if sample.response_length != 0:
            raise ValueError("cannot initialize a sampling mask after response tokens have already
            been appended")
        sample.rollout_sampling_mask = sampling_mask
        return

    if len(sample.rollout_sampling_mask) != sample.response_length:
        raise ValueError(
            f"sampling mask length {len(sample.rollout_sampling_mask)} is not aligned with "
            f"response_length {sample.response_length} before appending"
        )
    sample.rollout_sampling_mask = RolloutSamplingMask.concatenate((sample.rollout_sampling_
    mask, sampling_mask))


def merge_sampling_masks(
    first: Sample,
    observation_token_ids: Sequence[int],
    second: Sample,
) -> RolloutSamplingMask | None:
    """合并两段 per-response 的 ragged mask，中间插入强制观察 token 的单例支持。"""
    first_mask = first.rollout_sampling_mask
    second_mask = second.rollout_sampling_mask
    if first_mask is None or second_mask is None:
        if first_mask is None and second_mask is None:
            return None # 两边都没有采样支持，保持 None 即可
        # 单边缺失说明 replay 数据不完整，fail-fast 防止训练数据静默损坏
        raise ValueError("cannot merge samples unless both turns carry a complete rollout
        sampling mask")

    observation_mask = RolloutSamplingMask.from_mask_list([[int(token_id)] for token_id in
    observation_token_ids])
    return RolloutSamplingMask.concatenate((first_mask, observation_mask, second_mask))
```

miles/rollout/session/samples/codec.py

session 端 wire 契约核心：新增 sampling_mask *codec*，并通过 allowlist 排除策略保证默认 payload 字节级向后兼容。

```
# 传输契约：sampling_mask 用两个 tensor（ids + offsets）跨 wire，
# dtype 严格固定：ids = int32、offsets = int64，不匹配直接报错。
SAMPLES_VALUE_SPEC: dict[str, ValueSpec] = {
    "tokens": ValueSpec("tensor_list", np.dtype(np.int64), null=[]),
    "response": ValueSpec("json"),
    "response_length": ValueSpec("json"),
    "loss_mask": ValueSpec("tensor_list", np.dtype(np.uint8)),
    "rollout_log_probs": ValueSpec("tensor_list", np.dtype(np.float64)),
    "rollout_sampling_mask": ValueSpec("sampling_mask"),
    "rollout_routed_experts": ValueSpec("tensor", np.dtype(np.int32), strict=True),
    "rollout_indexer_topk": ValueSpec("tensor", np.dtype(np.int32), strict=True),
    "status": ValueSpec("json"),
    "weight_versions": ValueSpec("json"),
    "prefix_cache_info": ValueSpec("json"),
    "metadata": ValueSpec("json"),
}

# 关键设计：采样掩码默认不跨 wire，未启用该能力时 payload 字节级兼容。
ROLLOUT_SAMPLING_MASK_FIELDS = ("rollout_sampling_mask",)
COMPUTED_FIELDS = tuple(field for field in SAMPLES_VALUE_SPEC if field not in ROLLOUT_
SAMPLING_MASK_FIELDS)

# encode 分支：拆成两个连续 numpy tensor 写入 safetensors
if spec.codec == "sampling_mask":
    if not isinstance(value, RolloutSamplingMask):
        raise TypeError(f"{field} must be a RolloutSamplingMask, got {type(value).__name__}")
    ids, offsets = value._as_tensors()
    tensors[f"{field}.ids.{sample_index}"] = np.ascontiguousarray(ids.numpy())
    tensors[f"{field}.offsets.{sample_index}"] = np.ascontiguousarray(offsets.numpy())
    continue

# decode 分支：严格校验 dtype 后重建 RolloutSamplingMask
if spec.codec == "sampling_mask":
    ids = tensors.pop(f"{field}.ids.{sample_index}")
    offsets = tensors.pop(f"{field}.offsets.{sample_index}")
    if ids.dtype != np.int32 or offsets.dtype != np.int64:
        raise ValueError(
            f"{field} must use int32 ids and int64 offsets, got {ids.dtype} and {offsets.dtype}"
        )
    setattr(
        sample,
        field,
        RolloutSamplingMask(ids=torch.from_numpy(ids), offsets=torch.from_numpy(offsets)),
    )
```

continue

评论区精华

1. 实现简化：维护者 guapisolo 在评论中表示 Current code looks much simpler than before!，作者回应本分支已收敛为 transport-only 层，激活与 actor replay 全部移到 #2596，使传输契约可独立审查。
2. 合并策略：guapisolo 以 LGTM 审批但附带条件——建议待 #2596 就绪并带上 e2e CI 后再合并，且 #2596 当前被 #1837 的 args refactor 阻塞。这说明本 PR 虽是独立可合入的传输层，但端到端生效仍依赖系列完成。
3. 审查流程：由于是 fork PR，claude[bot] 两次提示自动化 review 被禁用、需维护者手动触发，最终由 guapisolo 人工审批合入。

- 实现简化与分层审查 (design): 将激活逻辑移出本 PR，使传输契约可独立审查。
- 合并时机与 e2e CI (other): 本 PR 作为传输层单独合入，端到端生效依赖 #2596。
- fork PR 自动化 review 关闭 (other): 未触发手动 review，最终由 guapisolo 人工审批合入。

风险与影响

- 风险：
  1. 兼容性（低）：codec 默认 COMPUTED_FIELDS 语义从“全部字段”变为“排除 sampling_mask 的字段”，依赖旧语义的调用方需改用 COMPUTED_FIELDS + ROLLOUT_SAMPLING_MASK_FIELDS 才能拿到全量；_TENSOR_FIELDS 断言也随之放宽。
  2. 数据完整性（中）：train_data_conversion.py 的“全 batch 必须携带掩码”是显式 fail-fast，一旦某个上游路径（如工具调用强制 token）漏记掩码，训练会硬失败；tool_call_utils.py 仅 +3 行，强制 token 覆盖是否完整需要 #2596 的 e2e 验证。
  3. 多轮合并异常路径（中）：merge_sampling_masks() 在单边缺失掩码时抛 ValueError，正确性依赖 _introduces_replay_gap 提前断链；若 gap 检测与掩码语义不一致，可能出现崩溃或掩码缺位，相关注释语义已从 routing gap 泛化为 replay gap。
  4. 性能（低）：concatenate() 每次 append 都对全量 ids/offsets 做 torch.cat 复制，长响应多轮场景下是 O(总 token 数) 的重复拷贝，当前阶段可接受，高频追加时值得留意。
  5. 测试覆盖（中）：仅 fast 单测与单卡聚焦测试，无 session→trainer 的端到端测试，guapisolo 也明确建议等待 #2596 的 e2e CI。 - 影响：默认路径下对用户和系统零启用影响：session wire 契约字节级不变，trainer 数据流只在显式启用采样掩码后新增两个可选键。开启后，采样支持数据可在 rollout、session 存储、对象存储、trainer 转换整条链路上以紧凑 CSR 形式保留，避免了 padding 到 [response_length, top_k] 的显存与带宽浪费。对团队而言，本 PR 采用“表示 → 传输 → 激活”三层交付，大幅降低了大功能 review 的复杂度，并为 #2596 提供了清晰的数据契约基准；影响面跨 rollout、session、ray、training 多个模块，属于中等影响。 - 风险标记：新数据契约跨多模块，默认路径零启用，全 batch 严格校验，缺少 e2e 覆盖，多轮合并异常路径依赖 gap 检测

关联脉络

- PR #2200 Add sampling-support log-probability primitives: PR body 系列第一步：为采样支持提供可选的 log-probability 原语，本 PR 在其之上叠加入口，base_sha 对应该分支。
- PR #2596 Enable bounded top-p capture and actor replay: PR body 系列第三步：启用有界 top-p 捕获与 actor replay，本 PR 只定义数据契约，激活逻辑全部留在 #2596。
- PR #1837 Args refactor: guapisolo 审批意见中提及 #2596 当前被 #1837 的 args refactor 阻塞，间接影响本 PR 的端到端验证节奏。