PR #1916 完整报告

radixark/miles

(1/2) refactor(rollout): drop --generate-multi-samples and its per-turn sample semantics

合并时间：2026-07-30 02:44

原文链接：http://prhub.com.cn/radixark/miles/pull/1916

执行摘要

- 一句话：移除 --generate-multi-samples，多轮轨迹统一返回标量 Sample
- 推荐动作：值得精读。这是 rollout 输出契约规范化的代表性 PR，展示了「把语义决策从 CLI 开关迁移到生成器类型契约」的设计思路。重点阅读 miles/rollout/generate_hub/multi_turn.py 与 miles/rollout/generate_hub/agentic_tool_call.py 的 head 版本，对比 merge 链路的统一；再看 dynamic_sampling_filters.py 的类型收窄，理解混合 group 过滤的正确姿势。若团队有自定义 reward model 或依赖旧 flag 的脚本，需按此 PR 同步迁移。

功能与动机

PR body 明确指出：Sample boundaries belong to trajectory topology, not a CLI switch; treating turns as siblings gives them one group_index and pollutes group baselines. 也就是说，--generate-multi-samples 让每个 turn 跳过 TITO merge、成为独立的 full-context sample，这些轮次级 sample 被视为同组兄弟，共享一个 group_index，从而污染基于 group 的 reward baseline 统计。移除 flag 同时不收缩输出契约：GenerateFnOutput.samples 仍保持 Sample | list[Sample]，内置线性生成器返回合并后的标量 sample，自定义生成器（multi-agent、tree 轨迹）可继续返回 list，未来还计划补充 tree 型轨迹支持。

实现拆解

整个改造按以下步骤推进：

1. multi_turn.generate 移除 flag 分支：在 miles/rollout/generate_hub/multi_turn.py 中删除 multi_samples 累积列表、每轮 sample = deepcopy(input.sample) 的重置逻辑、payload is None 时对 multi_samples[-1] 置 halt 状态的分支以及末尾 GenerateFnOutput(samples=multi_samples if args.generate_multi_samples else sample) 的条件返回；现在多轮轨迹始终累积在同一个 sample 对象上，TITO merge 天然生效，统一返回标量 Sample。同时从 _add_arguments 删除 --generate-multi-samples 参数注册。
2. agentic_tool_call.generate 同步收敛：在 miles/rollout/generate_hub/agentic_tool_call.py 中删除 if not input.args.generate_multi_samples: ... else: samples[-1].metadata.update(session_metadata) 的条件分支，改为无条件 merge_samples(samples, input.state.tokenizer) 后把 session_metadata 合并到标量 sample 上返回；aborted 路径同样返回标量。--generate-multi-samples 参数一并移除。

这一步保证同一 batch 内不会出现标量与 list 形状混合。

3. 动态过滤器类型契约修正：在 miles/rollout/filter_hub/dynamic_sampling_filters.py 中把 _flatten_samples、check_reward_nonzero_std、check_no_aborted 的参数注解统一为 list[Sample | list[Sample]]；check_reward_nonzero_std 原先直接对 samples 取 reward（遇到嵌套 list 会报错），现在改为经 _flatten_samples 展平后再计算方差，消除了对混合 group 的隐性假设。
4. 测试与 fixture 全面简化：tests/fast/fixtures/generation_fixtures.py 的 VARIANT_TO_GENERATE_FN_PATH 从 4 个条目收敛为 multi_turn、agentic_tool_call 两个，extra_argv_for_variant 不再注入 --generate-multi-samples；tests/fast/rollout/generate_hub/test_multi_turn.py 与 tests/fast/rollout/inference_rollout/integration/test_multi_turn.py 删除 multi_samples 形状断言分支（train 的 list[list[Sample]] 与 eval 展平 list[Sample] 两种形态），统一断言每次generate返回标量Sample；tests/fast/rollout/generate_hub/test_single_turn.py、tests/fast/rollout/inference_rollout/integration/test_agent_metadata.py、tests/manual/session/bench_session_server_overhead.py 同步移除对应变体与参数。
5. 文档更新：docs/user-guide/rollout-endpoints.md 将 "For multi-sample outputs, set --generate-multi-samples and return a list" 改为 "A generate function can set GenerateFnOutput.samples to a Sample or list[Sample]"，把输出形状契约从 CLI 开关解绑到生成器返回值类型上。

关键文件：

- miles/rollout/generate_hub/multi_turn.py（模块 多轮生成；类别 source；类型 core-logic；符号 generate, _add_arguments）：核心改造对象：移除 multi_samples 累积与按轮深拷贝分支，多轮轨迹统一在单一 sample 上累积并以标量 Sample 返回；同时删除 --generate-multi-samples 参数注册。
- miles/rollout/generate_hub/agentic_tool_call.py（模块 智能体生成；类别 source；类型 core-logic；符号 generate, _add_arguments）：agentic 生成路径同步移除 flag 分支，始终 TITO merge 返回标量 Sample；与 multi_turn 共同定义内置线性生成器的输出契约，并保证 batch 内不混入 list 形状。
- miles/rollout/filter_hub/dynamic_sampling_filters.py（模块 动态过滤；类别 source；类型 core-logic；符号 _flatten_samples, check_reward_nonzero_std, check_no_aborted）：动态过滤器是唯一保留混合形状处理的下游消费者；类型注解收紧为 list[Sample | list[Sample]]，并让 check_reward_nonzero_std 先展平再计算，避免嵌套 list 被当作 Sample 使用。
- tests/fast/rollout/generate_hub/test_multi_turn.py（模块 多轮生成；类别 test；类型 test-coverage；符号 is_agentic_variant, variant, test_two_turns_with_tool_call, test_max_turns_reached）：generate-hub 层行为保持的主要验证：variant 从 4 个收敛为 2 个，删除 multi_samples 形状断言分支，统一验证标量 Sample 语义。
- tests/fast/rollout/inference_rollout/integration/test_multi_turn.py（模块 集成测试；类别 test；类型 test-coverage；符号 _VARIANT_NAMES, _verify_samples, _verify_group_samples, _simple_reward_function）：覆盖 train/eval 全链路，确认 n_samples_per_prompt 语义下的输出形状从 list 收敛为标量；删除

_verify_group_samples 的 train/eval 两种复杂分支。

- tests/fast/rollout/generate_hub/test_single_turn.py（模块 单轮生成；类别 test；类型 test-coverage；符号 variant, expected_request, expected_sample）：回归保护单轮与多轮路径在请求构造、loss_mask、truncated 行为上不受 flag 移除影响。
- tests/fast/fixtures/generation_fixtures.py（模块 测试夹具；类别 test；类型 test-coverage；符号 VARIANT_TO_GENERATE_FN_PATH, extra_argv_for_variant）：variant 到 generate 函数路径映射的唯一事实来源，收敛后保证测试与真实 CLI 一致，也证明仓库内已无 flag 注入点。
- tests/fast/rollout/inference_rollout/integration/test_agent_metadata.py（模块 智能体元数据；类别 test；类型 test-coverage；符号 _AGENTIC_VARIANTS）：验证 agentic 标量输出上的自定义 reward 元数据路径（scalar custom reward path）。
- tests/manual/session/bench_session_server_overhead.py（模块 基准测试；类别 test；类型 test-coverage；符号 _build_server_args）：手动基准脚本清理，移除已删除参数的显式赋值，避免过期参数误导后续使用者。
- docs/user-guide/rollout-endpoints.md（模块 用户文档；类别 docs；类型 documentation）：把输出契约从 CLI 开关解绑到生成器返回值类型：GenerateFnOutput.samples 接受 Sample | list[Sample]，并移除 agentic 输出总是 list 的过时警告。

关键符号：multi_turn.generate, multi_turn._add_arguments, agentic_tool_call.generate, agentic_tool_call._add_arguments, dynamic_sampling_filters._flatten_samples, dynamic_sampling_filters.check_reward_nonzero_std, dynamic_sampling_filters.check_no_aborted, tests/.../test_multi_turn.py::_verify_samples, tests/.../test_multi_turn.py::_verify_group_samples, tests/.../test_multi_turn.py::_simple_reward_function

关键源码片段

miles/rollout/generate_hub/multi_turn.py

核心改造对象：移除 multi_samples 累积与按轮深拷贝分支，多轮轨迹统一在单一 sample 上累积并以标量 Sample 返回；同时删除 --generate-multi-samples 参数注册。

```
async def generate(input: GenerateFnInput) -> GenerateFnOutput:
    # ---------------------- Setup -------------------------
    args = input.args
    sample = deepcopy(input.sample)
    tokenizer = input.state.tokenizer
    assert not args.partial_rollout, 'Partial rollout is not supported'

    url = f'http://{args.sglang_router_ip}:{args.sglang_router_port}/generate'

    execute_tool_function = load_function(args.generate_execute_tool_function_path)
    tool_specs = load_function(args.generate_tool_specs_path)
    tool_call_parser = create_tool_call_parser(tool_specs, args.generate_tool_call_parser)

    # 调试轨迹开关：把多轮 messages 按快照写回 sample.metadata，
```

```
# 后续 turn 继续追加，因此快照是不断增长的轨迹而不是逐轮独立副本
record_trajectory = args.save_debug_trajectory_data is not None
trajectory = (list(sample.prompt) if isinstance(sample.prompt, list) else []) if record_trajectory
else None

# ---------------------- Initial prompts -------------------------
prompt_tokens_ids = compute_prompt_ids_from_sample(input.state, sample, tools=tool_specs)
sample.tokens = prompt_tokens_ids.copy()

for _turn in range(args.generate_max_turns):
    # 调用推理端点；payload 为 None 表示上下文超限，置 halt 状态直接退出
    payload, halt_status = compute_request_payload(args, sample.tokens, input.sampling_
    params)
    if payload is None:
        sample.status = halt_status
        break

    gen_t0 = time.time()
    output = await post(url, payload, headers=compute_routing_headers(args, sample))
    sink = None if input.evaluation else TrajectoryLifecycle().sink
    if sink is not None:
        tokens = output.get('meta_info', {}).get('completion_tokens', '')
        sink.gen_span(sample, gen_t0, time.time(), turn=_turn + 1, detail=str(tokens))
    await update_sample_from_response(args, sample, payload=payload, output=output,
    update_loss_mask=True)
    if record_trajectory:
        trajectory.append({'role': 'assistant', 'content': output['text']})
        sample.metadata['messages'] = list(trajectory) # 快照：后续 turn 继续增长

    # abort / length 终止（tokens 仍保留在 sample 上，便于截断恢复）
    if output['meta_info']['finish_reason']['type'] in ('abort', 'length'):
        break

    # ---------------------- Execute tools -------------------------
    _, tool_calls = tool_call_parser.parse_non_stream(output['text'])
    if len(tool_calls) == 0:
        break

    tool_t0 = time.time()
    tool_messages = await execute_tool_calls(tool_calls, execute_tool_function)
    if sink is not None:
        sink.tool_span(sample, tool_t0, time.time(), turn=_turn + 1, detail=f'{len(tool_calls)}
        calls')
    update_sample_with_tool_responses(sample, tool_messages, tokenizer=tokenizer)
    if record_trajectory:
        trajectory.extend(tool_messages)
        sample.metadata['messages'] = list(trajectory)

# 整个多轮轨迹累积在同一个 sample 上，只返回一个标量 Sample；
```

```
    # --generate-multi-samples 时代按 turn 深拷贝出 list[Sample] 的分支已移除
    return GenerateFnOutput(samples=sample)


def _add_arguments(parser: argparse.ArgumentParser):
    parser.add_argument('--generate-max-turns', type=int, default=16)
    parser.add_argument('--generate-tool-specs-path', type=str)
    parser.add_argument('--generate-tool-call-parser', type=str)
    parser.add_argument('--generate-execute-tool-function-path', type=str)
```

miles/rollout/generate_hub/agentic_tool_call.py

agentic 生成路径同步移除 flag 分支，始终 TITO merge 返回标量 Sample；与 multi_turn 共同定义内置线性生成器的输出契约，并保证 batch 内不混入 list 形状。

```
    # 到这里 samples 已是 session server 按轮次组装的线性记录；
    # agentic 路径始终执行 TITO merge，把多轮记录折叠成一个标量 Sample，
    # 不再有 --generate-multi-samples 开启时返回 list[Sample] 的分支
    sample = merge_samples(samples, input.state.tokenizer)
    sample.metadata.update(session_metadata)
    return GenerateFnOutput(samples=sample)


def _add_arguments(parser: argparse.ArgumentParser):
    parser.add_argument('--custom-agent-function-path', type=str)
    parser.add_argument(
        '--max-seq-len',
        type=int,
        default=None,
        dest='max_seq_len',
        help='Max sequence length in tokens (prompt + completion, including env responses) '
        'per session. Truncates samples on the Miles side and is forwarded to the '
        'Harbor agent server (as max_seq_len) to abort the trial early.',
    )
```

评论区精华

Review 主要由作者 guapisolo 自己在文档 diff 上发起讨论，核心交锋集中在「输出契约的文档表述」上：

- guapisolo 对文档初稿 Returning a list[Sample] from a generate function is supported natively; no flag is needed. 提出意见：输出形状不应表述为「需要 flag 或 native support」，而应直接描述契约本身；随后在 eae3bdf 中改为 GenerateFnOutput.samples 接受 Sample | list[Sample]。
- guapisolo 对 <Warning> Agentic output is a list[Sample] 的警告评论 "Not always actually."——因为最终实现把 agentic 生成结果 merge 为标量 Sample，仅在成功与 abort 路径都返回标量；最终移除该 list-only 警告。
- 审阅者 Shi-Dong 直接 APPROVED（"LGTM!"），说明整体设计方向无分歧。

- 文档措辞：输出形状不应绑定 flag (documentation): 作者在 eae3bdf 更新文案为 "A generate function can set GenerateFnOutput.samples to a Sample or list[Sample]"，把契约描述为返回值类型本身，不与任何 flag 绑定。
- agentic 输出是否总是 list[Sample] (correctness): 在 eae3bdf 中移除 list-only 警告，文档改为描述 GenerateFnOutput.samples 同时支持 Sample | list[Sample]；agentic 成功与 abort 路径均返回合并后的标量 Sample。

风险与影响

- 风险：
  1. 破坏性 CLI 变更：--generate-multi-samples 被移除，任何仍在命令行或配置中传该参数的脚本会触发 argparse 报错（unrecognized arguments）。仓库内调用点（fixture、bench 脚本）已同步清理，但外部用户脚本需要自行迁移。
  2. 输出形状变化影响自定义 reward model：agentic_tool_call.generate 在 flag 开启时曾返回 list[Sample]，现在总返回标量 Sample；若自定义 RM 按 list 形状编写（如旧版 _simple_reward_function 中 getattr(args, "generate_multi_samples", False) 分支），不更新会在运行时把 Sample 当 list 遍历而报错。
  3. 多轮截断语义对 flag 用户的改变：multi_turn.generate 首轮 payload 为 None 时，旧版 flag 路径会对 multi_samples[-1] 置 halt 状态并返回 list；新版统一对单一 sample 置 halt_status 并返回标量，行为等价于原先非 flag 路径，回归风险低，但 flag 用户看到的输出结构变化明显。
  4. 动态过滤器行为微调：check_reward_nonzero_std 改为先展平再计算方差，对原纯标量 group 完全等价，对混入 list 元素的 group 从潜在 AttributeError 变为正常工作，属于安全性修复，但理论上可能改变此前「报错即失败」场景下的任务结果。
  5. 测试覆盖收缩：集成测试删除了多轮 multi_samples 的 train/eval 两种形状校验，list 输出路径仅保留在自定义生成器的单元测试层面，后续对 Sample | list[Sample] 契约的回归保护略弱于此前。
- 影响：
  - 对用户：依赖 --generate-multi-samples 的 rollout 命令必须移除该参数；内置生成器输出统一为标量 Sample，批量数据中不再出现标量与 list 混合形状，下游 sample 组装和 reward 计算更可预期。
  - 对系统：sample 边界回归到轨迹拓扑决定，turn 级样本不再共享 group_index，group baseline 统计不再被轮次兄弟污染，对 PPO/RLHF 训练数据质量是正向修正；GenerateFnOutput.samples 契约保持 Sample | list[Sample]，为 multi-agent / tree 轨迹扩展预留空间。
  - 对团队：测试矩阵从 4 个 variant 收敛为 2 个，CI 运行时间略降；文档明确了 generate function 的输出契约；仓库内所有 flag 注入点已清理，后续维护成本降低。
  - 影响程度：中高——涉及 rollout 核心生成路径与输出形状契约，但对未使用该 flag 的默认路径行为不变。
  - 风险标记：破坏性 CLI 变更，生成输出形状变化，自定义 reward model 兼容性，外部脚本需同步更新

关联脉络

- PR #1759 (2/2) refactor(session): assemble training samples on the session server; records never leave it: 与本 PR 同属 rollout/session 重构系列（本 PR 为 1/2，1759 为 2/2）；1759 把训练样本组装移入 session server 并涉及 samples/merge.py 的合并语义，与本 PR 的 TITO merge 规范化方向一致。
- PR #2028 session: collect speculative-decoding counters: 改动了 miles/rollout/session/samples/merge.py 同一合并链路，后续在该合并语义上扩展计数器采集，可作为本 PR 合并行为的回归关注点。