PR #2240 完整报告

radixark/miles

feat(session): add configurable replay matching

合并时间：2026-08-13 08:33

原文链接：http://prhub.com.cn/radixark/miles/pull/2240

执行摘要

- 一句话：新增可配置的会话重放匹配，支持三种内置策略
- 推荐动作：值得精读。三个看点：一是 JSON 对象等价的类型打标设计（true vs 1、大数精度、重复键 /NaN 拒绝、递归深度回退），边界考虑细致且测试充分；二是 effective history 的“存储拼写优先”语义，以及 Shi-Dong 补上的 v1 提交修复，是数据一致性上的关键决策；三是“先回退大 PR、收窄后重做”的工程实践。若计划在生产启用 loose_tool_call 或 role_content_only，建议先读 docs/user-guide/customization.md 的风险模型，并对自定义 matcher 先做小流量验证。

功能与动机

PR body 明确指出痛点：Some agent harnesses reserialize tool-call arguments or omit reasoning fields when replaying history. The existing matcher treats those representations as divergent, causing v1 rollback or a new v2 lineage and sample。即 agent harness 重放历史时对工具调用参数做重序列化或省略 reasoning 字段，旧的严格匹配会将其判为“历史发散”，破坏会话连续性并带来 v1 回滚 / v2 新 lineage 的额外计算与样本统计开销。本 PR 让操作者按 harness 特性选择匹配粒度，同时保证可复用存储前缀保持权威。

实现拆解

1. 新建 message_matcher_hub 包集中策略与归一化原语：miles/utils/chat_template_utils/message_matcher_hub/utils.py 持有 SessionMessageMatcher 类型别名、_TEMPLATE_RELEVANT_KEYS / _WIRE_ONLY_TOOL_CALL_KEYS 常量，以及表示层工具 _normalize_value（falsy 哨兵归一）、_normalize_tool_calls（剔除 wire-only 的 index）、_normalize_json_object 与 _tag_json_value（类型打标 + 键排序的 JSON 对象等价归一）、_raw_values_match（类型敏感结构比较）；funcs.py 承载三个内置策略 strict_message_matches、loose_tool_call_message_matches、role_content_only_message_matches、选择器解析 resolve_session_message_matcher、运行时契约包装 _validated 与 append-only 校验 assert_messages_append_only_with_allowed_role。该包只依赖标准库，load_function 在 resolver 中懒加载，避免普通匹配路径拉起 Ray。
2. 迁移 template.py 并保留既有导入面：删除 template.py 中约 107 行匹配实现，将 message_matches 重命名为 strict_message_matches 移入 hub；template.py 顶部以直接别名重新导出 strict_message_matches 与 assert_messages_append_only_with_allow

ed_role，同步更新 tito_tokenizer.py、test_template.py、test_sessions.py 等处的既有引用，保持 from miles.utils.chat_template_utils import ... 的导入面可用。

3. 新增 --session-message-matcher 参数并在组合根装配：miles/utils/arguments.py 注册参数，默认 strict；TestSessionMessageMatcherArgument 验证默认值与"只保留选择器字符串、不提前导入"。miles/rollout/session/sessions.py 的 setup_session_routes 启动时调用 resolve_session_message_matcher 解析一次，注入 v1 SessionRegistry 与 v2 SessionRegistryV2，并注册 SessionMessageMatcherError 异常处理器映射 HTTP 500——matcher 抛异常或返回非 bool 属配置错误，绝不静默判定会话身份。
4. v1/v2 会话核心接入 matcher，统一 effective history 语义：linear_trajectory.py 的 prepare_pretokenized 与 _try_detect_and_rollback_to_assistant_checkpoint 接收 message_matcher，匹配通过后构造 effective_messages = self.messages + request_messages[len(self.messages):] 交给 TITO；update_pretokenized_state 提交同一 effective history，修复 raw replay 前缀改写存储历史的问题。v2 侧 tree_trajectory.py 的 TrajectoryTree.find_attach_point 与 session_state.py 的 position_for_request 增加 message_matcher 参数，决定重放请求是否附着既有节点。全程"先匹配、先 TITO，后改状态"，失败时状态零变更（failure atomicity）。
5. 测试与文档配套：新增 test_message_matcher_hub.py（约 520 行）覆盖三种策略的 JSON 等价边界（空对象拼写、键序、类型保持、大数、重复键、NaN/Infinity、递归深度回退）、契约包装与导入兼容；新增 test_session_message_matcher.py（约 262 行）覆盖 v1/v2 重放匹配、TITO effective history 传递、跨边界 tool-call ID、追加 role 校验与失败原子性。docs/user-guide/customization.md 补充 matcher 契约与风险模型，用户指南文档同步简化语义描述。回归套件 660 passed。

关键文件：

- miles/utils/chat_template_utils/message_matcher_hub/funcs.py（模块 匹配策略；类别 source；类型 core-logic；符号 strict_message_matches, loose_tool_call_message_matches, role_content_only_message_matches, resolve_session_message_matcher）：新增的匹配策略核心：三个内置 matcher、选择器解析与运行时契约包装，是整个 PR 的策略中枢。
- miles/utils/chat_template_utils/message_matcher_hub/utils.py（模块 归一化；类别 source；类型 core-logic；符号 _normalize_value, _normalize_tool_calls, _normalize_json_object, _tag_json_value）：表示层归一化原语，是 loose_tool_call 等价语义的精确实现基础，也供自定义 matcher 复用。
- miles/rollout/session/linear_trajectory.py（模块 会话核心；类别 source；类型 core-logic；符号 LinearTrajectory.prepare_pretokenized, LinearTrajectory.update_pretokenized_state, LinearTrajectory._try_detect_and_rollback_to_assistant_checkpoint, SessionRegistry.init）：v1 会话核心：matcher 注入、effective history 构造与提交语义修复都在这里，是行为正确性的关键文件。
- miles/utils/chat_template_utils/template.py（模块 模板工具；类别 source；类型 refactor；符号 message_matches, assert_messages_append_only_with_allowed_role）：匹配逻辑迁移的落点：删除约 107 行旧实现，改为从 hub 别名导入，保持既有 import 面

兼容。

- miles/rollout/session/sessions.py（模块 会话装配；类别 source；类型 entrypoint；符号 setup_session_routes, session_message_matcher_error_handler）：组合根：启动时解析一次 matcher 选择器并注入 v1/v2 registry，新增 SessionMessageMatcherError → HTTP 500 异常处理。
- miles/rollout/session/v2/tree_trajectory.py（模块 会话树；类别 source；类型 core-logic；符号 TrajectoryTree.find_attach_point）：v2 会话树附着判定接入 message_matcher，决定重放请求是附着既有节点还是分裂新 lineage。
- tests/fast/utils/chat_template_utils/test_message_matcher_hub.py（模块 匹配测试；类别 test；类型 test-coverage；符号 test_loose_tool_call_normalizes_empty_argument_objects, test_loose_tool_call_preserves_json_types_values_and_array_order, test_loose_tool_call_preserves_large_finite_json_numbers, test_loose_tool_call_invalid_or_non_object_arguments_use_type_sensitive_raw_comparison）：策略契约测试（约 520 行）：覆盖 JSON 等价边界、类型保持、递归回退、契约包装与导入兼容。
- tests/fast/router/test_session_message_matcher.py（模块 会话测试；类别 test；类型 test-coverage；符号 TestV1ReplayMatching, TestV2ReplayMatching, TestRegistryOwnership, _RecordingTITOTokenizer）：会话层端到端 wiring 测试：v1 回滚 / 提交、v2 附着、TITO effective history 传递与失败原子性。
- miles/utils/arguments.py（模块 参数解析；类别 source；类型 configuration；符号 session_message_matcher）：新增 --session-message-matcher 参数入口，默认 strict，是操作者启用该功能的配置面。
- docs/user-guide/customization.md（模块 用户文档；类别 docs；类型 documentation；符号 matcher）：用户文档补充 matcher 契约与风险模型，是操作者评估启用 loose/role_content_only 的入口。

关键符号：strict_message_matches, loose_tool_call_message_matches, role_content_only_message_matches, resolve_session_message_matcher, _validated, _normalize_json_object, _tag_json_value, assert_messages_append_only_with_allowed_role, LinearTrajectory.prepare_pretokenized, LinearTrajectory.update_pretokenized_state, LinearTrajectory._try_detect_and_rollback_to_assistant_checkpoint, TrajectoryTree.find_attach_point, position_for_request, setup_session_routes

关键源码片段

miles/utils/chat_template_utils/message_matcher_hub/funcs.py

新增的匹配策略核心：三个内置 matcher、选择器解析与运行时契约包装，是整个 PR 的策略中枢。

```
# message_matcher_hub/funcs.py：内置匹配策略与运行时契约


def _arguments_match(stored: Any, replayed: Any) -> bool:
    # 把 arguments 归一为"类型打标 + 键排序"的可比较形式（见 utils 的 _normalize_json_object）
```

```
    stored_normalized = _normalize_json_object(stored)
    replayed_normalized = _normalize_json_object(replayed)
    # 任一侧无法归一为 JSON 对象（非法 JSON、顶层非对象等）时，退回类型敏感的原始比较
    if stored_normalized is _INVALID_JSON_OBJECT or replayed_normalized is _INVALID_JSON_
    OBJECT:
        return _raw_values_match(stored, replayed)
    return stored_normalized == replayed_normalized


def _tool_call_matches(stored: Any, replayed: Any) -> bool:
    if not isinstance(stored, dict) or not isinstance(replayed, dict):
        return _raw_values_match(stored, replayed)
    # SGLang 会在非流式 tool call 上序列化 index，聊天模板从不读取它，比较前先剔除
    stored_projected = {key: value for key, value in stored.items() if key not in _WIRE_ONLY_
    TOOL_CALL_KEYS}
    replayed_projected = {key: value for key, value in replayed.items() if key not in _WIRE_ONLY_
    TOOL_CALL_KEYS}
    if stored_projected.keys() != replayed_projected.keys():
        return False
    for key in stored_projected:
        if key == "function":
            if not _functions_match(stored_projected[key], replayed_projected[key]):
                return False
        elif stored_projected[key] != replayed_projected[key]:
            return False
    return True


def loose_tool_call_message_matches(stored: dict[str, Any], replayed: dict[str, Any]) -> bool:
    """strict 的兼容超集：唯一新增的等价关系是 arguments 的受限 JSON 对象表示归一。

    call 的 id、type、function.name、调用顺序、未知扩展字段以及 reasoning_content
    仍严格比较，避免把“换了工具调用”误判为“同一段历史”。
    """
    try:
        if strict_message_matches(stored, replayed):
            return True
    except RecursionError:
        pass # 深层嵌套交给类型敏感比较兜底，不在这里直接抛异常
    for key in ("role", "content", "reasoning_content"):
        if _normalize_value(stored.get(key)) != _normalize_value(replayed.get(key)):
            return False
    stored_calls = _normalize_value(stored.get("tool_calls"))
    replayed_calls = _normalize_value(replayed.get("tool_calls"))
    if not isinstance(stored_calls, list) or not isinstance(replayed_calls, list):
        return _raw_values_match(stored_calls, replayed_calls)
    # 调用数量与顺序必须完全一致，逐项比较
    return len(stored_calls) == len(replayed_calls) and all(
        _tool_call_matches(left, right) for left, right in zip(stored_calls, replayed_calls, strict=True)
```

```
    )
```

miles/rollout/session/linear_trajectory.py

v1 会话核心：matcher 注入、effective history 构造与提交语义修复都在这里，是行为正确性的关键文件。

```
# linear_trajectory.py：v1 会话核心片段

def prepare_pretokenized(self, request_messages, tools=None, *, tito_tokenizer, message_
matcher=None):
    # matcher 缺省为 strict；同一 matcher 贯穿回滚检测与 append-only 校验
    matcher = message_matcher if message_matcher is not None else strict_message_matches

    # 1. 先做重放检测：可配置 matcher 决定逐位置身份，最多回退一个 assistant checkpoint
    self._try_detect_and_rollback_to_assistant_checkpoint(request_messages, matcher)

    if not self.token_ids:
        # 回退到空 checkpoint（或首轮）时从头渲染整段消息
        return tito_tokenizer.apply_chat_template(
            request_messages, tools=tools, add_generation_prompt=True, tokenize=True)

    # 2. 校验存储前缀逐条匹配，且追加消息的 role 被 TITO 模板允许
    assert_messages_append_only_with_allowed_role(
        self.messages, request_messages, tito_tokenizer.allowed_append_roles, message_matcher=
        matcher
    )

    # 3. effective history = 存储拼写的前缀 + 原始重放的新尾部；
    # 前缀 token 一律来自存储 checkpoint，重放里被接受但重序列化的部分不参与 token 化
    effective_messages = self.messages + request_messages[len(self.messages):]
    return tito_tokenizer.merge_tokens(
        old_messages=self.messages, new_messages=effective_messages,
        pretokenized_token_ids=self.token_ids, tools=tools,
    )


def update_pretokenized_state(self, request_messages, assistant_message,
                              prompt_token_ids, completion_token_ids, max_trim_tokens):
    all_token_ids = prompt_token_ids + completion_token_ids
    assert_pretokenized_prefix(self.token_ids, all_token_ids, max_trim_tokens=max_trim_tokens,
                               request_messages=request_messages, assistant_message=assistant_
                               message)

    # 提交与 prepare_pretokenized 相同的 effective history（存储拼写前缀 + 新尾部）：
    # 若提交原始 replay，被 loose 接受的重序列化前缀会改写存储历史，
    # 下一次规范重放将无法 strict 匹配自己的会话（discard_count 超限报错）
    self.messages = self.messages + request_messages[len(self.messages):] + [assistant_
    message]
    self.trajectory_token_ids.append(all_token_ids)
```

```
self.generated_checkpoint_message_ends.append(len(request_messages) + 1)
self.num_assistant = len(self.generated_checkpoint_message_ends)
```

评论区精华

本 PR 没有任何 inline review 评论（review_comments_count=0），Shi-Dong 直接 APPROVED，实质设计交锋都发生在提交历史里：

1) 初版 f5198e6 因"implementation expanded beyond the intended matcher-selector scope"被整体回退（84492b5），随后收窄重做，并删除无读者的 selector 分组指标（767d826）； 2) Shi-Dong 在合入前提交 509bbfa 指出 v1 update_pretokenized_state 直接存储 raw replay 会让 loose 接受的重序列化前缀改写存储历史，下一次规范重放 strict 匹配失败（discard_count=2 超限），修复为提交与 token 化一致的 effective history； 3) 契约检查从组合根下沉到 resolve_session_message_matcher（115809db），保证内置别名与自定义导入路径都被 _validated 包裹。

- 初版实现超出 matcher-selector 范围被整体回退 (design): 重做版本把改动收窄到 --session-message-matcher 选择器、message_matcher_hub 包与 v1/v2 注入，并删除无读者的 selector 分组指标（767d826），保留 matcher 可调用对象注入。
- v1 提交 raw replay 前缀会改写存储历史 (correctness): 改为提交与 prepare_pretokenized 相同的 effective history（存储拼写前缀 + 新尾部），与 v2 的 delta 存储天然免疫对齐。
- 契约检查位置从组合根下沉到 hub resolver (design): 把运行时契约包装移入 resolve_session_message_matcher，内置别名与点路径导入统一包裹，原始 matcher 通过 __wrapped__可达。

风险与影响

- 风险：
  1. 角色投影风险：role_content_only 刻意忽略 tool_calls 与 reasoning_content，存储前缀全面优先；若聊天模板或下游逻辑读取这些字段，重放携带的新工具调用 / 思考内容不会体现，需启用前评估模板行为。
  2. JSON 等价边界：_tag_json_value 对 Python dict 输入不放开 Decimal（allow_decimal 只对字符串解析开启），dict 形式带 Decimal 的 arguments 会被判不等；深层 JSON 触发 RecursionError 时退化为类型敏感比较并判 False，属安全方向但可能带来非预期回滚。
  3. 符号重命名：template.py 的 message_matches 改名，仓库内引用已全部同步，但外部直接导入该符号的代码会中断（strict_message_matches 与 append-only 校验保留别名导出）。
  4. 自定义 matcher 故障模式：matcher 抛异常或返回非 bool 会使会话请求直接 HTTP 500（见 sessions.py 的 session_message_matcher_error_handler），不再静默决定身份；对长会话服务意味着坏 matcher 会打挂请求，需先小流量验证。
  5. v1 提交语义变更：update_pretokenized_state 不再存储原始 replay，任何依赖"会话存储保留请求原样"的审计逻辑（如 replayed_messages）需核对新语义。 - 影响：影响面

覆盖 session server 的 v1/v2 双版本核心路径（linear_trajectory.py、tree_trajectory.py、session_state.py）与 chat_template_utils 公共模块。默认 strict 行为不变，存量用户无感；启用 loose_tool_call 后，重序列化工具参数的 agentharness 可保持会话连续，减少前缀 token 重算与 v2 样本分裂。role_content_only 属高风险显式投影，需人工确认再启用。团队侧新增 message_matcher_hub 作为统一扩展点（自定义点路径 matcher），并把 matcher 契约错误显式暴露为 HTTP 500，避免静默错误判定；同时确立了"存储拼写优先"的 effective history 提交原则。 - 风险标记：role_content_only 高风险投影，v1 提交语义变更，message_matches 符号重命名，自定义 matcher 失败返回 HTTP 500, JSON 等价边界复杂

关联脉络

- PR #2278 feat(session): request and assemble additional R3 rows under in-place weight updates: 同属 session 核心演进（session/core.py、v2/core.py 等），都在扩展会话重放 / 增量语义，目标同为削减多轮重复传输与状态损耗。
- PR #2368 fix(rollout): group session v2 leaf samples: session v2 与 agentic tool call 修复线，同一文件域（v2 会话样本组织），与本次 v2 附着语义改动互相影响。
- PR #2369 fix(rollout): normalize rewards per rollout: 同属 rollout/session 统计正确性修复，共用 tests/fast/router/ 的会话测试面（如 test_session_v1_v2_parity），反映该模块持续强化 v1/v2 一致性。