PR #6921 完整报告

verl-project/verl

[rollout] fix: restore turn separator dropped at multi-turn tool agent loop boundaries

合并时间：2026-07-06 10:19

原文链接：http://prhub.com.cn/verl-project/verl/pull/6921

执行摘要

- 一句话：修复多轮 AgentLoop 中回合分隔符 token 丢失
- 推荐动作：值得精读。核心 bugfix 设计优雅：使用 user turn 而非 assistant turn 推导分隔符，避免捕获 think 标签；充分处理 eos_token_id 的 list/tuple 兼容性；测试编写全面。建议关注后续 #6804 多模态 CT PR。

功能与动机

源自 issue #6501 中 @yyDing1 的诊断：多轮 agent loop 增量编码时，每轮 assistant->tool 边界会静默丢弃一个分隔符 token，使得 rollout 序列与完整对话的 apply_chat_template 出现偏差，且每轮累积。PR 在 #6501 评论 (https://github.com/verl-project/verl/issues/6501#issuecomment-4593158066) 指出后专项修复 agent-loop 实例。

实现拆解

1. 新增 initialize_turn_separator 工具函数（verl/utils/tokenizer/chat_template.py）：通过比较渲染空和非空 user 消息的 token 序列，取最大公共后缀，并分割出 close token (eos_token_id)，返回剩余的 turn separator。特意使用 user turn 以避免 assistant turn 中的推理脚手架。
2. 在 AgentLoopBase.__init__ 中缓存分隔符（verl/experimental/agent_loop/agent_loop.py）：当 Continuous Token 启用时设为 []（无操作），否则调用 initialize_turn_separator 并存入 self.turn_separator。
3. 在 _handle_processing_tools_state 中恢复分隔符（verl/experimental/agent_loop/tool_agent_loop.py）：在通用 chat-template 分支渲染 tool response 后，执行 response_ids = self.turn_separator + response_ids，确保后续追加的 response_mask = [0] 将分隔符标记为 context token。
4. 测试配套：新增 tests/utils/test_turn_separator_on_cpu.py，包含 7 个 CPU 回归测试，覆盖 Qwen3 式 think 守卫、multi-token eos_list 等情况。修改 tests/experimental/agent_loop/test_tool_call_id_on_cpu.py，为 mock 添加 turn_separator=[] 属性。

关键文件：

- verl/utils/tokenizer/chat_template.py（模块 工具模块；类别 source；类型 core-logic；符号 initialize_turn_separator）：核心变更，新增 initialize_turn_separator 函数，通过 diff 后缀推导 turn separator，处理 eos_token_id 兼容性
- verl/experimental/agent_loop/agent_loop.py（模块 代理循环；类别 source；类型 dependency-wiring）：导入并调用 initialize_turn_separator，按 Continuous Token 状态分支缓存 turn_separator
- tests/utils/test_turn_separator_on_cpu.py（模块 单元测试；类别 test；类型 test-coverage；符号 ChatMLTokenizer, test_turn_separator_is_the_newline_token, test_no_separator_template_returns_empty）：新增 7 个回归测试，覆盖标准 ChatML、无分隔符模板、Qwen3 式 think 守卫、multi-token eos 等情况，验证推导正确性和增量 / 完整编码一致性
- verl/experimental/agent_loop/tool_agent_loop.py（模块 代理循环；类别 source；类型 core-logic）：在通用 chat-template 分支中实际恢复分隔符，核心修复点
- tests/experimental/agent_loop/test_tool_call_id_on_cpu.py（模块 测试；类别 test；类型 test-coverage）：为 mock 添加 turn_separator=[] 属性使现有测试兼容

关键符号：initialize_turn_separator, AgentLoopBase.init, _handle_processing_tools_state

关键源码片段

verl/utils/tokenizer/chat_template.py

核心变更，新增 initialize_turn_separator 函数，通过 diff 后缀推导 turn separator，处理 eos_token_id 兼容性

```
def initialize_turn_separator(tokenizer, **apply_chat_template_kwargs) -> list[int]:
    """Tokens a chat template inserts after a message's closing token, before the next turn."""
    # 渲染空 user 消息和非空 user 消息
    empty = normalize_token_ids(
        tokenizer.apply_chat_template(
            [{"role": "user", "content": ""}],
            add_generation_prompt=False,
            tokenize=True,
            **apply_chat_template_kwargs
        )
    )
    filled = normalize_token_ids(
        tokenizer.apply_chat_template(
            [{"role": "user", "content": "x"}],
            add_generation_prompt=False,
            tokenize=True,
            **apply_chat_template_kwargs
        )
    )
    # 找最大公共后缀：后缀相同部分即 close_token + sep
    i = 0
    while i < len(empty) and i < len(filled) and empty[-1 - i] == filled[-1 - i]:
        i += 1
```

```
    suffix = empty[len(empty) - i :]
    if not suffix:
        return []
    # 获取 eos_token_id，兼容 processor 和 list/tuple 类型
    eos_id = getattr(tokenizer, "eos_token_id", None)
    if eos_id is None:
        eos_id = getattr(getattr(tokenizer, "tokenizer", None), "eos_token_id", None)
    eos_ids = {eos_id} if isinstance(eos_id, int) else set(eos_id or [])
    # 在后缀中找最后一个 eos token，其后的即为真正的 turn separator
    last_close = max((i for i, tok_id in enumerate(suffix) if tok_id in eos_ids), default=None)
    if last_close is not None:
        return suffix[last_close + 1 :]
    # 若未找到 eos token，则去掉第一个 token 作为 fallback（较少见）
    return suffix[1:]
```

verl/experimental/agent_loop/agent_loop.py

导入并调用 initialize_turn_separator，按 Continuous Token 状态分支缓存 turn_separator

```
# 导入部分（文件顶部）
from verl.utils.tokenizer.chat_template import apply_chat_template, initialize_system_prompt,
initialize_turn_separator

# 在 AgentLoopBase.__init__ 中（约 L230-255）
if continuous_token_config.enable and self.processor is None:
    # Continuous Token 路径：重新渲染完整消息列表，不需要 turn separator
    self.turn_separator = []
else:
    # Legacy 路径：通过工具函数推导 turn separator
    processing_class = self.processor if self.processor is not None else self.tokenizer
    self.turn_separator = initialize_turn_separator(processing_class, **self.apply_chat_template_
    kwargs)
```

评论区精华

- eos_token_id 兼容性讨论：gemini-code-assist[bot] 指出初始实现假设 eos_token_id 是单整数，但 Llama 3 等 tokenizer 暴露为 list/tuple，会导致分隔符推导错误。作者在 commit 502937b 中修复：归一化为 set，并添加多 token close 的回归测试。
- Continuous Token 关系讨论：wuxibin89 询问启用 data.continuous_token.enable=True 是否能避免此问题。作者确认 Continuous Token 通过重新渲染完整消息列表规避此问题，此 PR 只补丁 legacy 分支，且 CT 路径设定 turn_separator=[] 为无操作。gxlvera 补充多模态 CT（#6804）正在开发，未来将移除 legacy 路径。
  - eos_token_id 可能为 list 的处理 (correctness): 作者在 commit 502937b 中修复，使用 set 归一化并添加多 token close 回归测试。
  - Continuous Token 是否已解决此问题 (question): 明确 CT 已解决此问题，此 PR 作为 legacy 路径的必要修复，待 CT 全面生效后不再需要。

风险与影响

- 风险：
  - 通用 chat-template 分支：修复仅覆盖通用 chat-template 分支，gpt-oss 和 gemma4 分支未修改，若后续需要类似修复需各自手动处理。
  - VLM 多模态未验证：多模态 tool rollout 未进行 GPU e2e 测试，虽然分隔符是纯文本，但需要确认不影响 image placeholder 对齐。
  - Continuous Token 迁移：该修复在 legacy 路径生效，待 Continuous Token 成为默认并覆盖多模态后，此补丁可能成为技术债务。
  - 测试覆盖：CPU 单元测试充分，但缺少 GPU e2e 验证。
- 影响：
  - 用户影响：使用通用 chat template 的 agent loop 用户将获得正确的 token 序列，训练收敛性可能提升。Continuous Token 用户无影响。
  - 系统影响：初始化时增加一次 chat template 推导（极低开销），运行时每 tool turn 多一次 list 拼接（几乎无开销）。
  - 团队影响：需要跟踪 multimodal CT（#6804）进度，考虑 legacy 路径的最终移除。
  - 风险标记：Legacy 路径未来将移除，VLM 未 e2e 测试，缺少 GPU e2e 验证

关联脉络

- PR #6501 initialize_system_prompt can make mistake when the chat template is not strictly append-only: 该 issue 诊断了 PR 修复的 token 序列不匹配问题，@yyDing1 在评论中指出分隔符丢失
- PR #6529 [misc] fix: harden chat template prompt inference: 同一系列修复，但作用域不同（此 PR 修复 agent-loop 的分隔符丢失，而 #6529 修复 initialize_system_prompt 的探测逻辑）