

PR #7643 完整报告

verl-project/verl

[rollout] feat: continuous_token support Assistant Role in Merge and Encode Context

合并时间: 2026-09-07 13:18

原文链接: <http://prhub.com.cn/verl-project/verl/pull/7643>

执行摘要

- 一句话: 为 Continuous Token 构建器增加 rollback 风格的 assistant 角色上下文合并支持。
- 推荐动作: 此 PR 是一个具有重要设计决策的功能增强。建议相关工程师精读, 特别关注:
 1. merge_context_tokens 和分组编码的新逻辑, 以及 MergeResult 语义的变化。
 2. 模型特定子类 (尤其是 DeepSeek) 如何通过钩子适配新逻辑。
 3. scripts/chat_template_checker.py 中的更新, 以理解 CT 的新行为如何被验证。
 4. 新增的测试用例, 特别是那些模拟 rollback 场景的, 以理解边界条件。

该 PR 的洞察价值较高, 因为它展示了如何通过引入中间层 (钩子) 和约束来解决一个复杂的、与模型模板紧密相关的工程问题, 其设计模式值得借鉴。

功能与动机

此变更旨在支持某些 agent harness 的 rollback 行为, 即它们可能会重写之前的 assistant turn。在这种情况下, 返回给 CT 的上下文增量 (context delta) 可能包含 `role == "assistant"` 的消息。旧的 API (`merge_non_assistant_tokens`) 假定增量消息仅包含 `tool`、`user` 或 `system` 角色, 因此无法处理这种情况。此外, 直接对整个增量消息列表进行全量编码在某些模型 (如 Qwen3) 上会导致模板行为不一致 (例如, 模板会丢弃历史 assistant thinking), 因此需要按角色分组进行增量编码以确保稳定性和正确性。

实现拆解

1. API 重命名与角色扩展: 将 ContinuousTokenBuilder 基类中的 `merge_non_assistant_tokens` 和 `tokenize_non_assistant_incremental_messages` 方法分别重命名为 `merge_context_tokens` 和 `tokenize_context_incremental_messages`。同时, 将 MergeKind 类型从 "non_assistant" 更新为 "context", 并将 MergeResult 的默认 kind 从 "non_assistant" 更新为 "context"。_SUPPORTED_APPEND_ROLES 集合中增加了 "assistant" 角色。
2. 分组增量编码逻辑: 在 `tokenize_context_incremental_messages` 中引入 `processed_messages` 列表来维护处理上下文。该方法不再一次性编码所有追加消息, 而是调用 `_iter_append_groups` 将消息按角色分组, 并对每一组独立进行 delta 编码。通过新增的 `_merge_context_group` 钩子方法, 将每个组的编码结果追加到总增量 token 列表中。
3. 上下文约束与子类适配: 新增规则: 上下文增量消息的最后一条消息不能是 assistant 角色, 否则抛出 ValueError。针对模型特定的子类 (如 DeepSeekV4ContinuousTokenBuilder、

Gemma4ContinuousTokenBuilder) 进行了适配: DeepSeek 子类重写了 `_merge_context_token_ids`, 仅在 `previous_role == "assistant"` 时插入 EOS token, 并在 `tokenize_context_incremental_messages` 中禁用了 `drop_thinking` 以保留历史推理内容。Gemma4 子类则可能通过 `_merge_context_group` 钩子处理其特定的 turn separator 逻辑。

4. Agent Loop 调用点更新: 更新 `verl/experimental/agent_loop/agent_loop.py` 中的 `ct_merge_non_assistant_msg` 方法, 将其重命名为 `ct_merge_context_msg`, 并调用新的 `merge_context_tokens`。同步更新 `verl/experimental/agent_loop/tool_agent_loop.py` 中的调用点。
5. 测试与配套更新: 扩展了 `tests/utils/test_continuous_token_on_cpu.py` 测试文件, 增加了新的 mock tokenizer (如 `_GptOssRollbackTokenizer`, `_DeepSeekToolAppendTokenizer`) 和测试用例, 以覆盖 rollback 场景、分组编码行为以及模型特定的边界处理。更新了 `scripts/chat_template_checker.py` 以调用新的 `merge_context_tokens` API。

关键文件:

- `verl/utils/tokenizer/continuous_token.py` (模块 Token 构建器; 类别 source; 类型 dependency-wiring; 符号 `tokenize_non_assistant_incremental_messages`, `tokenize_context_incremental_messages`, `merge_non_assistant_tokens`, `_merge_context_group`): Continuous Token 构建器的核心实现文件, 包含所有主要 API 重命名、分组编码逻辑引入和 MergeKind 变更。
- `verl/utils/tokenizer/deepseek.py` (模块 DeepSeek 构建器; 类别 source; 类型 core-logic; 符号 `tokenize_non_assistant_incremental_messages`, `tokenize_context_incremental_messages`, `_merge_non_assistant_token_ids`, `_merge_context_token_ids`): DeepSeek V4 构建器的子类适配, 展示了如何通过重写钩子来适配新的上下文合并逻辑, 特别是处理 EOS 插入和 `drop_thinking` 行为。
- `tests/utils/test_continuous_token_on_cpu.py` (模块 CT 单元测试; 类别 test; 类型 test-coverage; 符号 `_DeepSeekBoundaryTokenizer`, `_DeepSeekToolAppendTokenizer`, `_GptOssGenerationRewriteTokenizer`, `apply_chat_template`): 测试文件, 大幅扩展以覆盖新的上下文合并逻辑、分组编码行为、rollback 场景以及模型特定子类 (如 GPT-OSS rollback tokenizer) 的适配。

关键符号: `tokenize_context_incremental_messages`, `merge_context_tokens`, `_merge_context_group`, `_merge_context_token_ids`, `ct_merge_context_msg`

关键源码片段

`verl/utils/tokenizer/continuous_token.py`

Continuous Token 构建器的核心实现文件, 包含所有主要 API 重命名、分组编码逻辑引入和 MergeKind 变更。

```
# continuous_token.py 中 token 增量编码的核心逻辑
# 修改后支持按角色分组编码, 并处理上下文中的 assistant 消息
```

```
def tokenize_context_incremental_messages(
    self,
```

```

previous_messages: list[dict[str, Any]],
updated_messages: list[dict[str, Any]],
*,
tools: list[dict[str, Any]] | None = None,
) -> list[int]:
    # ... 前置检查 ...
    incremental_ids: list[int] = []
    # 用于维护逐步处理的消息历史
    processed_messages = list(previous_messages)

    # 按角色分组追加消息
    groups = self._iter_append_groups(appended_messages)
    fuse_generation_prompt = self._should_fuse_generation_prompt_with_last_group()

    for index, group in enumerate(groups):
        # ... 根据 group 角色进行编码 (tool, user, system, assistant) ...
        role = group[0].get("role")
        if role == "tool":
            # ... token 化工具组 ...
        elif role in {"user", "system", "assistant"}:
            # 新规则: 上下文增量不能以 assistant 结尾
            if role == "assistant" and index == len(groups) - 1:
                raise ValueError("Continuous Token context incremental messages cannot end with assistant")
            # ... token 化单条消息 ...
        # ... 错误处理 ...

    # 使用钩子将编码结果合并到增量列表中
    self._merge_context_group(
        incremental_ids,
        group_token_ids,
        group=group,
        processed_messages=processed_messages,
        tools=tools,
    )
    # 更新已处理的消息历史, 为下一组提供上下文
    processed_messages.extend(group)

    # ... 处理 generation prompt delta ...
    return incremental_ids

# 新增的钩子方法, 子类可用于自定义合并行为
def _merge_context_group(
    self,
    incremental_ids: list[int],
    group_token_ids: list[int],
    *,
    group: list[dict[str, Any]],
    processed_messages: list[dict[str, Any]],

```

```

        tools: list[dict[str, Any]] | None = None,
    ) -> None:
        """Merge one encoded context group into the incremental token stream."""
        incremental_ids.extend(group_token_ids)

```

verl/utils/tokenizer/deepseek.py

DeepSeek V4 构建器的子类适配，展示了如何通过重写钩子来适配新的上下文合并逻辑，特别是处理 EOS 插入和 drop_thinking 行为。

deepseek.py 中 DeepSeekV4ContinuousTokenBuilder 的适配

```

def tokenize_context_incremental_messages(self, previous_messages, updated_messages, *,
tools=None):
    # ... 前置检查 ...
    if appended_messages[-1].get("role") == "assistant":
        raise ValueError("Continuous Token context incremental messages cannot end with
assistant")
    self._assert_prefix_is_stable(previous_messages, tools=tools)
    # 关键适配：编码上下文时禁用 drop_thinking，以保留历史推理内容
    return self._encode(
        appended_messages,
        tools=None,
        add_bos_token=False,
        drop_thinking=False,
    )

def _merge_context_token_ids(
    self,
    runtime_token_ids: list[int],
    appended_token_ids: list[int],
    *,
    previous_messages: list[dict[str, Any]] | None = None,
    **kwargs: Any,
) -> MergeResult:
    prefix = list(runtime_token_ids)
    inserted_token_ids: list[int] = []
    # 根据前一条消息角色决定是否插入 EOS
    previous_role = previous_messages[-1].get("role") if previous_messages else None
    # 只有前一条是 assistant 且当前没有 EOS 时才插入
    if appended_token_ids and previous_role == "assistant" and prefix[-1:] != [self._eos_id]:
        prefix.append(self._eos_id)
        inserted_token_ids.append(self._eos_id)
    return MergeResult(
        token_ids=prefix + list(appended_token_ids),
        appended_token_count=len(appended_token_ids),
        kind="context",
        inserted_token_ids=inserted_token_ids,
    )

```

评论区精华

审查者 [wuxibin89](#) 提出了一个关键问题：Gemma4 和 DeepSeek 为什么需要专门的 `tokenize_context_incremental_messages`？是否可以将其合并到基类中？

作者 [gxlvera](#) 回应称，最初为这两个子类重写整个方法是因为它们在合并上下文消息组内（例如，`[tool_response, user_prompt]`）时有特殊的边界行为。但随后作者发现，这种特殊行为可以通过更小的钩子函数来实现，而不是重写整个方法。因此，作者重构了代码：

1. 在基类中新增了一个 `_merge_context_group()` 钩子函数，用于处理上下文消息列表中合并不同消息组时的特殊边界行为。
2. 让 Gemma4 和 DeepSeek 子类复用基类的 `tokenize_context_incremental_messages` 和 `merge_context_tokens`，只重写更小的钩子函数，如 `_merge_context_group` 和 `_merge_context_token_ids`。这样减少了重复代码，并提高了代码复用性。

这个讨论和随后的重构显著提升了代码设计的清晰度和可维护性。

- 子类钩子化重构 (design): 采用钩子化重构，提高代码复用性和可维护性。子类仅需重写必要的钩子。

风险与影响

- 风险：

1. API 兼容性风险： `merge_non_assistant_tokens` 等 API 被重命名为 `merge_context_tokens`， `MergeKind` 的枚举值从 "non_assistant" 变为 "context"。这是一个 breaking change。任何直接调用这些 API 或依赖 `MergeResult.kind` 值为 "non_assistant" 的外部代码或集成都会受到影响。需要确保所有内部调用点（如 agent loop）已全部更新。
2. 行为变更风险：对于不支持或不预期 assistant 角色出现在上下文增量中的下游逻辑，此变更可能引入意外行为。虽然 `merge_context_tokens` 会确保上下文中的 assistant token 的 loss mask 和 logprob 为 0，但调用方需要理解这一语义变化。
3. 模型模板兼容性：分组编码策略旨在解决特定模型（如 Qwen3）的模板不一致性，但引入的“不允许上下文以 assistant 结束”规则是一个强约束。需要确保所有支持的 agent harness 和模型组合都满足此约束，否则会引发 `ValueError`。
4. 回归风险：对 `ContinuousTokenBuilder` 核心类和多个子类（DeepSeek, Gemma4）的修改，以及 agent loop 调用点的更改，涉及关键的多轮对话 token 构建路径。尽管有大量测试，但仍需关注在复杂、长对话或特殊模型配置下的回归风险。
 - 影响：用户影响：对于使用支持 rollback 风格 agent harness 的用户，此 PR 是必要的功能补充，使得 CT 能够正确处理被重写的 assistant turn。对于其他用户，此 PR 是一个透明的内部重构，但需要注意 API 重命名带来的升级适配。
 - 系统影响：此 PR 增强了 CT 模块的灵活性和健壮性，使其能够更准确地模拟全量 `apply_chat_template` 的行为，尤其是在处理复杂、多变的对话上下文时。这对于提升基于 CT 的多轮强化学习训练的稳定性和性能至关重要。
 - 团队影响：此 PR 是 CT 功能演进的重要一步，标志着其从仅支持单向追加到支持更复杂上下文管理的转变。相关的代码重构（引入更小的钩子）也为未来的模型适配工作提供了更好的扩展性。

- 风险标记：核心路径变更，API Breaking Change

关联脉络

- PR #7630 [rollout] fix: DeepSeek continuous token builder cannot render tool appends: 同属 Continuous Token 模块的功能修复和演进。本 PR 的 `tokenize_context_incremental_messages` 重构部分解决了该 PR 提及的 DeepSeek 工具追加渲染问题。
- PR #7703 [training_utils] fix: honor fused output-head backend: 无直接关联，但同属近期高活跃度的核心功能修复 / 增强 PR，共同展示了仓库在模型兼容性和功能演进方面的进展。