

PR #3115 完整报告

radixark/miles

Scan full responses for repetition metrics

合并时间: 2026-09-10 03:04

原文链接: <http://prhub.com.cn/radixark/miles/pull/3115>

执行摘要

- 一句话: 修复重复度量仅扫描末尾片段的缺陷, 改为全响应重叠窗口扫描并限制计算成本。
- 推荐动作: 这是一个修复度量缺陷的高质量 PR, 值得精读。核心设计决策 (重叠窗口 + 步长动态扩大以控制成本) 简洁有效, 测试覆盖全面且具有针对性。特别值得关注的是作者在第二个提交中对性能问题的快速响应, 体现了良好的工程实践。该变更对提升系统监控的准确性有积极意义。

功能与动机

当前 `rollout/repetition_frac` 仅检查每个响应的最后 10,000 个字符。一个长的响应可能在前部包含严重的重复, 但结尾却是非重复的文本, 导致该指标遗漏了这种行为。扫描有界的重叠窗口使指标能够代表响应中任意位置的重复, 同时在发现重复时保留短路评估的能力。

实现拆解

1. 引入滑动窗口机制: 在 `miles/utils/metric_utils.py` 中新增 `_repetition_windows(text)` 生成器函数, 用于生成响应文本的重叠窗口。窗口大小 (`REPETITION_WINDOW_SIZE_CHARS`) 为 10,000 字符, 初始步长 (`REPETITION_WINDOW_STRIDE_CHARS`) 为 5,000 字符, 并确保总是包含响应的精确末尾片段。
2. 重写 `has_repetition` 函数: 将原来的单次末尾检查重构为使用 `_repetition_windows` 生成器并调用 `any()` 进行短路评估。只要任意一个窗口的压缩比超过阈值 (`REPETITION_COMPRESSION_RATIO_THRESHOLD=10.0`), 即判定存在重复。
3. 添加成本上限机制: 为解决长文本扫描成本线性增长的问题, 引入了 `REPETITION_MAX_WINDOWS=32` 常量。在 `_repetition_windows` 中, 当文本长度超过约 165k 字符时, 动态扩大步长 (`stride = max(REPETITION_WINDOW_STRIDE_CHARS, math.ceil(final_start / (REPETITION_MAX_WINDOWS - 1)))`), 确保扫描的窗口总数不超过常量上限, 从而将计算成本限制在常数级别。
4. 全面测试覆盖: 新增 `tests/fast/utils/test_metric_utils.py` 文件, 包含 8 个测试用例, 覆盖了步长对齐的中间窗口、未对齐的最终窗口、边界长度、短文本短路、严格阈值、长文本窗口数量有界、首尾覆盖以及真实压缩场景下的中间重复检测等关键路径。

关键文件:

- `miles/utils/metric_utils.py` (模块 工具函数; 类别 `source`; 类型 `core-logic`; 符号 `has_repetition, _repetition_windows`): 核心逻辑变更文件, 实现了新的重叠窗口扫描算

法和成本控制机制。

- tests/fast/utils/test_metric_utils.py (模块 测试套件; 类别 test; 类型 test-coverage; 符号 test_has_repetition_checks_stride_aligned_middle_window, test_has_repetition_checks_unaligned_final_window, test_has_repetition_checks_exactly_ten_thousand_characters, test_has_repetition_ignores_short_text) : 新增的全面测试文件, 覆盖了重复度检测新逻辑的多个关键场景和边界条件。

关键符号: _repetition_windows, has_repetition

关键源码片段

miles/utils/metric_utils.py

核心逻辑变更文件, 实现了新的重叠窗口扫描算法和成本控制机制。

```
# miles/utils/metric_utils.py
import math
from collections.abc import Iterator
from typing import Any, Literal

import numpy as np

# 窗口大小与步长常量, 定义扫描粒度
REPETITION_WINDOW_SIZE_CHARS = 10_000
REPETITION_WINDOW_STRIDE_CHARS = 5_000
# 扫描窗口数量上限。由于 has_repetition 在 rollout executor 的同步事件循环中运行,
# 其成本必须与响应长度无关。当文本超过约 165k 字符时, 步长会动态扩大,
# 以确保窗口总数不超过此上限。相邻窗口在约 320k 字符以内仍然重叠,
# 从而保证任意现实响应长度下的覆盖无间隙。
REPETITION_MAX_WINDOWS = 32
REPETITION_COMPRESSION_RATIO_THRESHOLD = 10.0

# ... (dict_add_prefix, compute_pass_rate, compression_ratio 等函数保持不变) ...

def _repetition_windows(text: str) -> Iterator[str]:
    """生成重叠窗口, 包括精确的最终后缀。"""
    if len(text) < REPETITION_WINDOW_SIZE_CHARS:
        return

    final_start = len(text) - REPETITION_WINDOW_SIZE_CHARS
    # 动态计算步长: 取默认步长与根据窗口上限计算出的步长中的较大值。
    # 这确保了总窗口数不超过 REPETITION_MAX_WINDOWS。
    stride = max(
        REPETITION_WINDOW_STRIDE_CHARS,
        math.ceil(final_start / (REPETITION_MAX_WINDOWS - 1)),
    )
```

```

last_start = -1
for start in range(0, final_start + 1, stride):
    yield text[start : start + REPETITION_WINDOW_SIZE_CHARS]
    last_start = start

# 如果步长扩大导致最后一个生成的窗口未覆盖到最终后缀，则补充生成该窗口。
if last_start != final_start:
    yield text[final_start:]

```

```

def has_repetition(text: str) -> bool:
    """返回任意重叠窗口是否高度可压缩。"""
    return any(
        compression_ratio(window)[0] > REPETITION_COMPRESSION_RATIO_THRESHOLD
        for window in _repetition_windows(text)
    )

```

... (compute_rollout_step 保持不变) ...

tests/fast/utils/test_metric_utils.py

新增的全面测试文件，覆盖了重复度检测新逻辑的多个关键场景和边界条件。

```

# tests/fast/utils/test_metric_utils.py
import hashlib
from unittest.mock import Mock

```

```

from miles.utils import metric_utils

```

```

def test_has_repetition_checks_stride_aligned_middle_window(monkeypatch) -> None:
    """测试步长对齐的中间窗口被正确检测。"""
    text = "a" * 5_000 + "b" * 10_000 + "c" * 5_000
    compression_ratio = Mock(
        side_effect=lambda window: (11.0, 0.0) if window == "b" * 10_000 else (1.0, 0.0)
    )
    monkeypatch.setattr(metric_utils, "compression_ratio", compression_ratio)

    assert metric_utils.has_repetition(text)
    # 验证调用了两次 compression_ratio: 第一次是第一个窗口 (a+b 前半)，第二次是第二个窗口 (b 完整)
    assert compression_ratio.call_count == 2

```

```

def test_repetition_window_count_is_bounded_for_very_long_text(monkeypatch) -> None:
    """扫描在 rollout executor 的事件循环上运行；其成本不能随响应长度缩放。"""
    compression_ratio = Mock(return_value=(1.0, 0.0))
    monkeypatch.setattr(metric_utils, "compression_ratio", compression_ratio)

    # 对于 2,000,000 字符的文本，自然步长会产生约 400 个窗口，但动态调整后应限制。

```

```

assert not metric_utils.has_repetition("x" * 2_000_000)
assert compression_ratio.call_count <= metric_utils.REPETITION_MAX_WINDOWS

def test_repetition_mid_way_in_a_very_long_response_is_still_caught() -> None:
    """在约 320k
    字符以内，扩大的步长仍保持窗口重叠，因此中间的长重复总能落在某个窗口内（使用真实压缩，无
    Mock）。"""
    # 构造约 275k 字符的低压缩比填充文本
    filler = "".join(
        hashlib.sha256(str(i).encode()).hexdigest() for i in range(4_300)
    )
    # 在文本中间插入高度重复的 "spam" 序列
    text = filler[:150_000] + "spam" * 6_250 + filler[150_000:]

    assert metric_utils.has_repetition(text) # 含重复应检出
    assert not metric_utils.has_repetition(filler) # 纯填充不应检出

```

评论区精华

核心讨论由自动化 Review 机器人 [claude\[bot\]](#) 发起。机器人首先指出了新实现 `has_repetition` 函数的成本随响应长度线性增长 ($O(\text{len}/\text{stride})$)，并在 `RolloutExecutor.get` 的异步协程中同步运行，可能阻塞事件循环。作者在后续提交 ([e6ea865](#)) 中通过引入 `REPETITION_MAX_WINDOWS=32` 和动态调整步长直接解决了这个问题。机器人最终确认该修复有效，并验证了生产环境的响应长度限制在无间隙覆盖范围内。争议点（计算成本）已通过明确的常量上限解决。

- 扫描成本随响应长度线性增长 (performance): 作者通过引入 `REPETITION_MAX_WINDOWS=32` 常量和动态调整步长的机制，在后续提交 ([e6ea865](#)) 中直接解决了该问题，确保总扫描窗口数（从而计算成本）不超过小常数，与文本长度无关。

风险与影响

- 风险：
 1. 性能回归风险：变更将原本的单次 zlib 压缩（固定 10KB 窗口）改为最多 32 次压缩（每次 10KB 窗口）。虽然已设置上限，但在非重复长响应上，计算成本仍高于原实现。风险被 `REPETITION_MAX_WINDOWS` 常量和事件循环上下文限制所缓解。
 2. 正确性风险：步长动态扩大后，对于极端长文本（>320k 字符），窗口之间可能出现间隙，导致间隙中的重复内容未被检测到。但根据 Review 讨论，生产配置 `--rollout-max-response-len` 约为 32,768 tokens，远低于 320k 字符阈值，因此该风险在实际场景中不可达。
 3. 测试覆盖风险：测试用例充分覆盖了核心逻辑和边界条件，包括一个不使用 Mock 的真实压缩检测用例，降低了正确性风险。
- 影响：
 1. 对用户的影响：提高了重复度量的准确性，能够更早发现模型生成中出现的重复问题，从而可能影响训练策略或奖励计算。

2. 对系统的影响：rollout executor 的事件循环可能因长响应（非重复）的扫描而增加少量同步阻塞时间，但被严格限制在 32 次压缩操作内，影响可控。
3. 对团队的影响：提供了更可靠的重复检测工具，有助于改进模型训练质量。测试套件的扩展也为后续维护提供了良好基础。 - 风险标记：核心路径变更，同步阻塞事件循环

关联脉络

- 暂无明显关联 PR