

PR #48262 完整报告

vllm-project/vllm

[Bugfix] Gemma4 parser: classify channel-less output consistently in streaming and non-streaming

合并时间: 2026-07-14 21:30

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/48262>

执行摘要

- 一句话: 修复 Gemma4 流式解析器无通道输出分类不一致
- 推荐动作: 推荐精读, 特别是 `adjust_initial_state_from_prompt` 的修改逻辑。展示了如何通过精确的预初始化条件修复状态机不一致。对于了解 Gemma4 解析器工作原理的开发者很有价值。

功能与动机

引用 issue #48217: 当启用 `thinking` 时, Gemma4 流式解析器对于以新模型回合结尾的提示总是预初始化为 `REASONING`。如果模型直接给出无通道标记的答案, 每个流式片段都被分类为推理, 永远不会发出内容, 而非流式路径将相同文本分类为内容。

实现拆解

1. 在 `vllm/parser/gemma4.py` 中新增 `_prompt_ends_in_open_reasoning` 方法, 后向扫描提示符 token, 检查是否以打开的 `<lchannel>` 块结束。
2. 修改 `adjust_initial_state_from_prompt`, 将条件从 `not is_reasoning_end` 改为 `_prompt_ends_in_open_reasoning`, 仅在提示符确实在打开的推理通道内时预初始化 `REASONING`。
3. 在测试文件 `tests/parser/engine/test_gemma4_streaming_reasoning.py` 中新增 `TestGemma4ChannelLessOutputConsistency` 测试类, 覆盖无通道输出作为内容流式输出、流式与非流式结果一致两个场景。
4. 重命名原有测试方法 `test_redundant_channel_open_swallowed_after_new_turn` 为 `test_model_emitted_channel_open_after_new_turn`, 更新注释以反映新的引擎启动状态 (`CONTENT` 而非 `REASONING`)。

关键文件:

- `vllm/parser/gemma4.py` (模块 解析器; 类别 `source`; 类型 `core-logic`; 符号 `_prompt_ends_in_open_reasoning`, `adjust_initial_state_from_prompt`): 核心逻辑变更, 新增 `_prompt_ends_in_open_reasoning` 方法, 修改 `adjust_initial_state_from_prompt` 条件
- `tests/parser/engine/test_gemma4_streaming_reasoning.py` (模块 流式推理; 类别 `test`; 类型 `test-coverage`; 符号 `TestGemma4ChannelLessOutputConsistency`,

test_streaming_channel_less_output_is_content, test_streaming_matches_non_streaming, test_model_emitted_channel_open_after_new_turn)：新增测试验证流式与非流式输出一致，重命名和更新现有测试

关键符号：_prompt_ends_in_open_reasoning, adjust_initial_state_from_prompt

关键源码片段

vllm/parser/gemma4.py

核心逻辑变更，新增 _prompt_ends_in_open_reasoning 方法，修改 adjust_initial_state_from_prompt 条件

```
def _prompt_ends_in_open_reasoning(self, prompt_token_ids: Sequence[int]) -> bool:
```

```
    """检查提示符尾部是否处于打开的 `<lchannel>` 块中。
```

```
    从后向前扫描：如果在遇到任何关闭或回合边界 token 之前看到
    `<lchannel>` 起始 token，说明块仍处于打开状态。
```

```
    """
```

```
    start_id = self._reasoning_start_token_id
```

```
    if start_id is None:
```

```
        return False
```

```
    # 将关闭、工具调用、新回合、工具响应视为边界 token
```

```
    boundary_ids = {
```

```
        tid
```

```
        for tid in (
```

```
            self._reasoning_end_token_id,
```

```
            self._tool_call_token_id,
```

```
            self._new_turn_token_id,
```

```
            self._tool_response_token_id,
```

```
        )
```

```
        if tid is not None
```

```
    }
```

```
    for tid in reversed(prompt_token_ids):
```

```
        if tid == start_id:
```

```
            return True
```

```
        if tid in boundary_ids:
```

```
            return False
```

```
    return False
```

```
def adjust_initial_state_from_prompt(self, prompt_token_ids: Sequence[int]) -> None:
```

```
    """仅当提示符以打开的 `<lchannel>` 块结束时预初始化引擎为 REASONING。
```

```
    这覆盖了后工具响应续写场景 (issue #45834)。对于仅仅开始新回合的提示，
    不应预初始化推理状态，因为模型可能直接给出答案 (issue #48217)。
```

```
    当模型确实打开自己的 `<lchannel>` 时，`(CONTENT, THINK_START)` 状态
    转换会处理它，`thought\n` 前缀也会被 `_events_to_delta` 剥离。
```

```
    """
```

```
    if not self._prompt_ends_in_open_reasoning(prompt_token_ids):
```

```
        return
```

```
self._engine.reset(initial_state=ParserState.REASONING)
# 防止后续的 `initialize_streaming()` 覆盖
self._streaming_initialized = True
```

tests/parser/engine/test_gemma4_streaming_reasoning.py

新增测试验证流式与非流式输出一致，重命名和更新现有测试

```
# —— Channel-less direct answer (no reasoning markers at all) ——

_PLAIN_ANSWER_TEXT = "This is a direct final answer without channel markers."
_PLAIN_ANSWER_TOKENS: list[tuple[int, str]] = []
for _i, _word in enumerate(_PLAIN_ANSWER_TEXT.split(" ")):
    _prefix = " " if _i > 0 else ""
    _PLAIN_ANSWER_TOKENS.append((7000 + _i, _prefix + _word))

class TestGemma4ChannelLessOutputConsistency:
    """流式和非流式必须将无通道输出分类一致。

    启用 `enable_thinking=True` 时，仅以新回合 token 结尾的提示不应
    预初始化推理状态。如果模型直接输出无通道标记的答案，流式流应将其
    作为内容输出，而非推理。
    """

    @pytest.fixture
    def plain_tokenizer(self):
        return _make_tokenizer(_PLAIN_ANSWER_TOKENS)

    @pytest.fixture
    def plain_parser(self, plain_tokenizer):
        return Gemma4Parser(plain_tokenizer)

    def test_streaming_channel_less_output_is_content(
        self, plain_parser, plain_tokenizer, request_obj
    ):
        # 提示符以新回合 token 结尾，不以打开通道结尾
        results = _stream_tokens_batched(
            plain_parser,
            plain_tokenizer,
            request_obj,
            batch_size=1,
            prompt_token_ids=[NEW_TURN_ID, 9100, 9101],
        )
        reasoning, content, _ = _collect_fields(results)
        assert content == _PLAIN_ANSWER_TEXT, (
            f"Expected content to be {_PLAIN_ANSWER_TEXT!r}, got {content!r}"
        )
        assert reasoning is None, (
            f"Expected no reasoning, got {reasoning!r}"
        )
```

```

)

def test_streaming_matches_non_streaming(
    self, plain_parser, plain_tokenizer, request_obj
):
    # 相同输出在流式与非流式模式下分类一致
    results_streaming = _stream_tokens_batched(
        plain_parser,
        plain_tokenizer,
        request_obj,
        batch_size=1,
        prompt_token_ids=[NEW_TURN_ID, 9100, 9101],
    )
    reasoning_s, content_s, _ = _collect_fields(results_streaming)
    # 非流式直接使用 parser.reasoning 属性获取结果
    plain_parser.reset()
    plain_parser.add_request(request_obj)
    for token_id, text in _PLAIN_ANSWER_TOKENS:
        plain_parser.add_token((token_id, text))
    plain_parser.finish()
    reasoning_ns, content_ns = plain_parser.reasoning, plain_parser.content
    assert reasoning_s == reasoning_ns, (
        f"Streaming reasoning {reasoning_s!r} != non-streaming {reasoning_ns!r}"
    )
    assert content_s == content_ns, (
        f"Streaming content {content_s!r} != non-streaming {content_ns!r}"
    )

```

评论区精华

DavidDiao 在评论中报告了另一个问题：当模型输出包含 `<lchannel>` 但没有 `thought\n` 标签时，解析器将所有内容视为推理。作者 adhi29 回复认为这是独立问题，并建议单独跟进。维护者 bbrowning 确认当前修复解决了原始不一致，并建议使用正确的聊天模板。没有未解决的疑虑。

- 关于无 `<channell>` 标记和 `thought` 标签的边角情况 (question): 作者 adhi29 认为这是独立问题，可能需单独跟进；维护者 bbrowning 建议确认使用正确的聊天模板。
- 关于 pre-commit 检查失败 (other): bbrowning 回应指出 CI 尚未运行，预计本地检查通过

风险与影响

- 风险：主要风险在于新逻辑可能遗漏一些边界情况（如自定义 token 或特殊聊天模板），但测试覆盖了两种主要场景（新回合和打开通道）。`is_reasoning_end` 保持不变，结构化输出不受影响。回归风险较低，因为现有测试全部通过（3847 个）。
- 影响：影响仅限于使用 Gemma4 模型且启用 `thinking` 的流式解码用户。修复使流式与非流式输出分类一致，不影响非流式或禁用 `thinking` 的模式。对性能无影响。
- 风险标记：边界情况处理，回归风险

关联脉络

- PR #48217 [Bug]: Gemma4 parser classifies plain output without channel markers differently in streaming and non-streaming mode: 报告了本 PR 修复的不一致问题