PR #24251 完整报告

sgl-project/sglang

[RL][TITO] Preserve whitespace in reasoning parser outputs

合并时间：2026-05-21 03:45

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24251

执行摘要

- 一句话：保留推理输出中的空白字符
- 推荐动作：建议阅读以了解推理解析器在处理空白与标记之间的设计权衡。尤其值得关注 detect_and_parse 中从全局替换到循环去除的转变，以及为何移除 .strip() 调用是可接受的。

功能与动机

根据 PR body，在 TITO（Token-In-Token-Out）路径下，需要保留模型生成消息的原始格式，包括空白。DeepSeek-V4-Flash 测试中发现类似 '（换行）...（换行）（换行）（换行）answer' 的输出中空白被剥离，影响了 RL 训练的正确性。共享的推理解析器在 main 分支也存在同样的问题。

实现拆解

1. 修改 python/sglang/srt/parser/reasoning_parser.py 中的 detect_and_parse 方法：移除全局 replace 加 strip() 的逻辑，改为循环去除前导 <think> 标记（支持重复标记），并去除 split 后的 .strip()，保留末尾所有空白。
2. 修改同一文件中的 parse_streaming_increment 方法：移除返回前对 reasoning_text 的 rstrip() 调用，保留末尾空白。
3. 修改 python/sglang/srt/function_call/deepseekv32_detector.py 中的 detect_and_parse 方法：将提取普通文本时的 .strip() 改为仅移除末尾两个换行符（`removesuffix('

')`），避免去除内部空白。

1. 新增单元测试：在 test_reasoning_parser.py 中新增三个测试用例（非流式空白保留、重复前导标记处理、流式空白保留）；在 test_serving_chat.py 中新增一个集成测试验证 Chat API 响应中的 reasoning_content 和 content 保留空白。

关键文件：

- test/registered/unit/parser/test_reasoning_parser.py（模块 解析测试；类别 test；类型 test-coverage；符号 test_parse_non_stream_preserves_payload_whitespace, test_parse_non_stream_strips_repeated_leading_start_tokens, test_parse_stream_chunk_preserves_payload_whitespace）：新增三个测试用例，验证

非流式、流式空白保留以及重复前导标记处理。

- test/registered/unit/entrypoints/openai/test_serving_chat.py（模块 聊天测试；类别 test；类型 test-coverage；符号 test_non_stream_reasoning_response_preserves_payload_whitespace）：新增集成测试验证 Chat API 响应中空白保留。
- python/sglang/srt/parser/reasoning_parser.py（模块 推理解析；类别 source；类型 core-logic；符号 detect_and_parse, parse_streaming_increment）：核心改进：移除 strip 和 rstrip，支持循环去除前导标记，保留输出空白。
- python/sglang/srt/function_call/deepseekv32_detector.py（模块 DeepSeek 检测；类别 source；类型 core-logic；符号 detect_and_parse）：修改一处剥离逻辑，避免去除内容前的空白。

关键符号：detect_and_parse, parse_streaming_increment, test_parse_non_stream_preserves_payload_whitespace, test_parse_non_stream_strips_repeated_leading_start_tokens, test_parse_stream_chunk_preserves_payload_whitespace, test_non_stream_reasoning_response_preserves_payload_whitespace

关键源码片段

python/sglang/srt/parser/reasoning_parser.py

核心改进：移除 strip 和 rstrip，支持循环去除前导标记，保留输出空白。

```
    def detect_and_parse(self, text: str) -> StreamingParseResult:
        '''
        One-time parsing: Detects and parses reasoning sections in the provided text.
        Returns both reasoning content and normal text separately.
        '''
        in_reasoning = self._in_reasoning or self.think_start_token in text

        if not in_reasoning:
            return StreamingParseResult(normal_text=text)

        # 组合开始标记（如 '<think>' + 可能的 self label）
        think_start_text = self.think_start_token + self.think_start_self_label
        processed_text = text
        # 循环去除所有前导开始标记（支持重复标记，如 <think><think>）
        while processed_text.startswith(think_start_text):
            processed_text = processed_text[len(think_start_text):]

        if (
            self.think_end_token not in processed_text
            and self.think_end_token not in self.previous_content
        ):
            # 检查是否有工具调用开始标记
            if (
                in_reasoning
                and self.tool_start_token is not None
```

```
            and self.tool_start_token in processed_text
        ):
            tool_idx = processed_text.find(self.tool_start_token)
            reasoning_text = processed_text[:tool_idx] # 保留空白，不再 .strip()
            normal_text = processed_text[tool_idx:]
            return StreamingParseResult(
                normal_text=normal_text, reasoning_text=reasoning_text
            )
        return StreamingParseResult(reasoning_text=processed_text)

    if self.think_end_token in processed_text:
        splits = processed_text.split(self.think_end_token, maxsplit=1)
        reasoning_text = splits[0]
        normal_text = splits[1] # 保留空白，不再 .strip()
        return StreamingParseResult(
            normal_text=normal_text, reasoning_text=reasoning_text
        )
    else:
        return StreamingParseResult(normal_text=processed_text)
```

评论区精华

来自 Codex 审查的讨论：

- P2 建议（关于重复起始标记）：detect_and_parse 现在只移除第一个前导标记，对于像 DeepSeekR1Detector 这样文档会发出多个 <think> 的模型，第二个标记会泄漏到 reasoning_text 中，导致格式标记可见。
- P1 建议（关于前导空白）：startswith 条件可能导致 <think> 前存在空白时标记未被移除，造成相同问题。
- 作者回应：<think> 由模板插入，通常不会出现前导空白，因此回归风险低。 结论：团队接受了当前方案，合并了 PR。
- 重复前导标记可能导致 reasoning_text 中残留标记 (correctness): 作者将全局替换改为循环去除前导标记，解决重复标记问题。
- 前导空白导致 <think> 标记未去除 (correctness): 作者回应 <think> 由模板插入，通常无前导空白，回归风险低。团队接受。

风险与影响

- 风险：主要风险在于 reasoning_parser.py 中 detect_and_parse 的行为改变可能影响那些在模板外存在前导空白或重复标记的模型。虽然作者声明由模板保证，但自定义模板可能引入此类情况，导致推理内容中出现未预期的标记。deepseekv32_detector.py 的改动仅限于特定场景，风险较低。回归测试通过，CI 无失败。
- 影响：对用户影响：RL 训练中 TITO 路径下的推理内容将保留原始空白，更忠实于模型输出，有助于训练稳定。对系统影响：解析器行为变更，需要确保所有调用方（如流式和非流式 Chat API）同步更新。测试覆盖增强，降低未来回归风险。
- 风险标记：核心解析路径变更，行为回归风险，模板假设依赖

关联脉络

- 暂无明显关联 PR