PR #42570 完整报告

vllm-project/vllm

[Refactor] Use shared utils in hermes tool parser

合并时间：2026-05-14 08:35

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42570

执行摘要

- 一句话：Hermes 工具解析器提取公共工具函数
- 推荐动作：建议快速合并。这是良好的代码清理工作，降低重复，提高一致性。值得其他工具解析器参考这一模式。

功能与动机

根据 PR body: 'Replace _partial_tag_overlap and _is_valid_json in hermes_tool_parser.py with shared util methods' 以及 'Move constant token strings and compiled regexes from __init__ to class-level attributes'，旨在统一工具解析器间的公共逻辑，减少重复代码。

实现拆解

1. 导入共享工具函数：在文件头部增加 from vllm.tool_parsers.utils import is_complete_json, partial_tag_overlap，替换原先的本地函数。
2. 删除本地函数：移除模块级别的 _partial_tag_overlap 和 _is_valid_json 两个函数定义。
3. 提升常量和正则表达式为类属性：将 tool_call_start_token、tool_call_end_token、tool_call_regex、scratch_pad_regex 从 __init__ 方法内移到类体，作为类级属性，避免每次实例化时重复编译正则。
4. 更新调用点：在 _extract_content 和 _extract_tool_call_jsons 方法中，将 _partial_tag_overlap 替换为 partial_tag_overlap，将 _is_valid_json 替换为 is_complete_json。
5. 测试：运行 pytest tests/tool_parsers/test_hermes_tool_parser.py 验证无回归。

关键文件：

- vllm/tool_parsers/hermes_tool_parser.py（模块 工具解析器；类别 source；类型 dependency-wiring；符号 _partial_tag_overlap, _is_valid_json, partial_tag_overlap, is_complete_json）：核心变更文件，替换本地工具函数为共享函数，提升常量和正则为类属性。

关键符号：partial_tag_overlap, is_complete_json, _extract_content, _extract_tool_call_jsons

关键源码片段

vllm/tool_parsers/hermes_tool_parser.py

核心变更文件，替换本地工具函数为共享函数，提升常量和正则为类属性。

```
# vllm/tool_parsers/hermes_tool_parser.py（关键变更片段）

# 导入共享工具函数（新增）
from vllm.tool_parsers.utils import is_complete_json, partial_tag_overlap

# 删除私有函数 _partial_tag_overlap 和 _is_valid_json

class Hermes2ProToolParser(ToolParser):
    # 常量和正则表达式提升为类属性（从 __init__ 移出）
    tool_call_start_token: str = "<tool_call>"
    tool_call_end_token: str = "</tool_call>"
    tool_call_regex = re.compile(
        r"<tool_call>(.*?)</tool_call>|<tool_call>(.*)", re.DOTALL
    )
    scratch_pad_regex = re.compile(r"<scratch_pad>(.*?)</scratch_pad>", re.DOTALL)

    def __init__(self, tokenizer: TokenizerLike, tools: list[Tool] | None = None):
        super().__init__(tokenizer, tools)
        if is_mistral_tokenizer(tokenizer):
            logger.error("Detected Mistral tokenizer when using a Hermes model")
            self.model_tokenizer = tokenizer.tokenizer
        # 之前在这里设置 token 和 正则，现在由类属性初始化
        if not self.model_tokenizer:
            raise ValueError(...)
        self._sent_content_idx: int = 0

    # ... 其他方法不变 ...

    def _extract_content(self, current_text: str) -> str | None:
        if self.tool_call_start_token not in current_text:
            overlap_length = partial_tag_overlap( # 使用共享函数
                current_text, self.tool_call_start_token
            )
            sendable_idx = len(current_text) - overlap_length
        # ...

    def _extract_tool_call_jsons(self, text: str) -> list[tuple[str, bool]]:
        # ...
        overlap = partial_tag_overlap(raw, self.tool_call_end_token) # 使用共享函数
        is_complete = is_complete_json(tc_json) if tc_json else False # 使用共享函数
        # ...
```

评论区精华

变更仅收到自动化 bot 的 review（Claude、Gemini），均给出 LGTM，yewentao256 手动批准。无实质性讨论或争议。

- 暂无高价值评论线程

风险与影响

- 风险：低风险。变更仅涉及内部函数替换和属性提升，功能逻辑完全不变。共享函数 partial_tag_overlap 和 is_complete_json 已在 utils 模块中存在且经过测试。但需确保 is_complete_json 与原 _is_valid_json 的行为完全一致（utils 中的 is_complete_json 可能额外处理了不完全 JSON 的情况，但功能上等价）。
- 影响：直接影响 Hermes 工具解析器的使用者，但对功能和性能无可见影响（正则编译从实例化移至类加载，略有性能提升）。降低了后续维护成本，其他工具解析器也可复用相同工具。
- 风险标记：暂无

关联脉络

- 暂无明显关联 PR