PR #44361 完整报告

vllm-project/vllm

[Bugfix] Responses API assistant EasyInputMessageParam input

合并时间：2026-06-23 20:54

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44361

执行摘要

- 一句话：修复 Responses API 助手消息字符串内容解析错误
- 推荐动作：建议阅读本 PR，特别是 protocol.py 中 input_item_parsing 的改动模式。它展示了在 Pydantic 强制转换前增加类型守卫以绕过验证错误的典型手法。同时关注讨论中关于缺少模型定义的思考，提示团队后续需重构以消除对字典的依赖。

功能与动机

用户在使用 Responses API 时，传入形如 {"role": "assistant", "type": "message", "content": "some string"} 的输入会触发 BadRequestError，错误信息包含 2 validation errors for ValidatorIterator: 0.ResponseOutputTextParam, 0.ResponseOutputRefusalParam。根本原因是 assistant 的字符串内容被不当强制为 ResponseOutputMessage，导致 Pydantic 尝试将字符串中的每个字符解析为输出内容项。

实现拆解

1. 协议层防护（vllm/entrypoints/openai/responses/protocol.py）：在 ResponsesRequest.input_item_parsing 方法中，针对 type: message, role: assistant 的输入，增加对 content 类型的检查。如果 content 不是列表（即字符串），则直接保留原始字典，不再尝试创建 ResponseOutputMessage 对象，避免后续 Pydantic 校验错误。
2. 工具调用构造扩展（vllm/entrypoints/openai/responses/utils.py）：在 _construct_message_from_response_item 函数中新增分支，处理字典类型的 assistant 消息。提取 content 字段，若为字符串直接使用，若为列表则取第一个元素的 text 字段，然后与之前的 assistant 消息合并或生成新消息，确保多轮对话对话历史正确。
3. 测试用例补充（tests/entrypoints/openai/responses/test_function_call_parsing.py）：新增 test_assistant_string_content_stays_easyinput 验证字符串内容的 assistant 消息不会被强制转换；新增 test_assistant_output_style_content_coerced 验证列表内容的 assistant 消息仍然能够正确转换为 ResponseOutputMessage。
4. construct_chat_messages_with_tool_call 测试增强（tests/entrypoints/openai/responses/test_responses_utils.py）：新增参数化用例 reasoning-easyinput-assistant，验证 reasoning 后跟随字典格式 assistant 消息的合并场景，确保回归覆盖。

关键文件：

- vllm/entrypoints/openai/responses/protocol.py（模块 协议层；类别 source；类型 core-logic；符号 input_item_parsing）：核心修复：在 input_item_parsing 中增加字符串内容检查，避免不当强制转换
- vllm/entrypoints/openai/responses/utils.py（模块 工具层；类别 source；类型 core-logic；符号 _construct_message_from_response_item）：辅助修复：扩展 _construct_message_from_response_item 处理字典格式 assistant 消息
- tests/entrypoints/openai/responses/test_function_call_parsing.py（模块 测试；类别 test；类型 test-coverage；符号 test_assistant_string_content_stays_easyinput, test_assistant_output_style_content_coerced）：新增两个单元测试，验证字符串内容保留和列表内容强制转换的正确性
- tests/entrypoints/openai/responses/test_responses_utils.py（模块 测试；类别 test；类型 test-coverage）：增强 construct_chat_messages_with_tool_call 的参数化测试，添加 reasoning-easyinput-assistant 场景

关键符号：input_item_parsing, _construct_message_from_response_item, test_assistant_string_content_stays_easyinput, test_assistant_output_style_content_coerced

关键源码片段

vllm/entrypoints/openai/responses/protocol.py

核心修复：在 input_item_parsing 中增加字符串内容检查，避免不当强制转换

```
# vllm/entrypoints/openai/responses/protocol.py
# 在 ResponsesRequest.input_item_parsing 中，处理 assistant 消息时
# 增加对 content 类型的判断：字符串内容保留原始 dict，避免 Pydantic 错误

elif item_type == "message" and item.get("role") == "assistant":
    content = item.get("content")
    # 如果 content 是字符串，则是合法 EasyInputMessageParam
    # 不强制转换为 ResponseOutputMessage
    if not isinstance(content, list):
        processed_input.append(item)
        continue

    # 只有列表内容才尝试转为 ResponseOutputMessage
    original_item = item
    item = dict(item)
    if "id" not in item:
        item["id"] = f"msg_{random_uuid()}"
    if "status" not in item:
        item["status"] = "completed"
    # 补充 annotations 字段等逻辑 ...
    try:
        processed_input.append(ResponseOutputMessage(**item))
    except ValidationError:
        processed_input.append(original_item)
```

vllm/entrypoints/openai/responses/utils.py

辅助修复：扩展 _construct_message_from_response_item 处理字典格式 assistant 消息

```
# vllm/entrypoints/openai/responses/utils.py
# 在 _construct_message_from_response_item 中新增 elif 分支
# 处理未被协议层强制转换的 EasyInput 字典格式

elif isinstance(item, dict) and item.get("role") == "assistant":
    content = item.get("content")
    text: str | None = None
    if isinstance(content, str):
        text = content
    elif isinstance(content, list) and content:
        # 注意：这里只取第一个文本片段，多段内容会丢失
        text = content[0].get("text")
    if text is not None:
        if prev_assistant_msg:
            previous_content = prev_assistant_msg.get("content")
            if previous_content is None:
                prev_assistant_msg["content"] = text
                return None
        return {"role": "assistant", "content": text}
```

评论区精华

- bbrowning评论：在 vllm/entrypoints/openai/responses/utils.py 中直接处理字典类型的 assistant 消息，是否意味着缺少一个对应的 Pydantic 模型定义？感觉我们缺少了与 ResponseOutputMessage 对应的简单字符串版本模型。
- yzong-rh回应：EasyInputMessageParam 在 openai 包中被定义为 TypedDict，我们没有创建自己的 Pydantic 模型。可以通过自定义 Pydantic 模型并强制转换来解决，但当前改动镜像了已有行为。这个已知问题已在多个地方存在（如 utils.py 285-295 行），需要更全面的修复。
- 是否需要为字符串内容创建专门 Pydantic 模型 (design): 确认已知问题，本 PR 不解决，作为后续工作

风险与影响

- 风险：
  - 兼容性风险：改动仅限于输入解析路径，对于已存在的 ResponseOutputMessage 列表类型内容行为不变，向下兼容。
  - 功能覆盖不完整：讨论中已指出，construct_chat_messages_with_tool_call 中提取列表内容时只取 content[0].get("text")，对于多段 input_text 类型的列表会丢失部分内容。但此风险在修复前已存在，本 PR 未解决。
  - 测试覆盖：新增的两个单元测试和参数化用例覆盖了核心路径，但缺少对嵌套内容（如拒绝响应 refusal 字段）的测试，属于已知缺口。

- 影响：
  - 用户影响：所有使用 Responses API 并传入字符串内容 assistant 消息的用户将不再遇到验证错误，多轮对话正常工作。涉及非 harmony 和 harmony 两种模式。
  - 系统影响：改动集中在请求输入解析阶段，对推理性能无影响。
  - 团队影响：明确了 EasyInputMessageParam 与 ResponseOutputMessage 的边界，为后续创建专用 Pydantic 模型提供了基础。
  - 风险标记：多部分内容处理不完整，缺少专用 Pydantic 模型

关联脉络

- PR #46030 [Refactor] Responses API parser state into conversation context: 修改了相同的 responses/protocol.py 和 responses/utils.py 文件，属于同一功能线
- PR #44285 [Frontend] Split ServingRender into renderer and entrypoint.: 涉及 Responses API 的渲染和消息构造，与当前改动有间接关联