

PR #56299 完整报告

vllm-project/vllm

[Bugfix][Frontend] Support Responses text types in DeepSeek V4.1

合并时间: 2026-09-14 10:43

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56299>

执行摘要

- 一句话: 修复 DeepSeek V4.1 拒绝 Responses API 文本类型, 白名单扩充加回归测试
- 推荐动作: 值得快速浏览, 作为“一行修复 + 精准测试”的范本: 它展示了在自定义 tokenizer 内做 content type 归一化时容易遗漏协议差异 (Chat API 的 text vs Responses API 的 input_text/output_text) 的教训。建议关注两点: 一是 `_normalize_messages()` 这类协议适配层应尽量与通用 content 归一化逻辑复用, 二是可将“协议类型别名”收敛到共享常量以降低未来遗漏风险。

功能与动机

issue #56297 给出了可复现的复现步骤: `POST /v1/responses` 携带 `{"type": "input_text", "text": "Reply with OK."}` 时返回 HTTP 400, 错误信息为 `DeepSeek V4.1 supports text and image content only; got 'input_text'`。根因是 DeepSeek V4.1 渲染器会将原始 messages 直接传给 `vllm/tokenizers/deepseek_v41.py` 的 `_normalize_messages()`, 绕过通用内容归一化; 而该函数只识别 Chat API 的 `text` 类型, 不识别 OpenAI Responses API 的 `input_text/output_text`。Issue 还明确指出当前 `main` 分支同样受影响。

实现拆解

该修复通过以下步骤完成:

1. 定位问题入口: `vllm/tokenizers/deepseek_v41.py` 中的 `_normalize_messages()` 是 DeepSeek V4.1 自定义分词器唯一的内容块过滤入口, `apply_chat_template()` 在编码前会先调用它把结构化 content 拍平成字符串。
2. 根因确认: 函数内 `part_type == "text"` 的单值判断排除了 Responses API 使用的 `input_text` (用户输入) 与 `output_text` (助手输出) 两种内容块类型, 导致落入 `else` 分支抛出 `ValueError`。
3. 一行修复: 将条件改为 `part_type in ("text", "input_text", "output_text")`, 三个类型统一按文本追加, `block.get("text", "")` 保留原有容错逻辑; 图片分支 (`image_url/input_image/image_pil`) 原样保留, 不参与本次改动。
4. 测试配套: 在 `tests/tokenizers/_test_deepseek_v41.py` 新增参数化测试 `test_responses_text_parts_match_chat_text_parts`, 覆盖 (user, input_text) 与 (assistant, output_text) 两组角色 / 类型组合, 断言与 `text` 类型的渲染输出逐字节一致; 本地 32 个测试全部通过, pre-commit 检查通过。

5. CI 与合入流程：期间共触发 4 次 Buildkite CI (#88490、#88512、#88587、#88624)，mergify 提示首轮 pre-commit 失败后作者修复并重新推送，最终由维护者 chaunceyjiang 批准合并。

关键文件：

- vllm/tokenizers/deepseek_v41.py（模块 分词器；类别 source；类型 core-logic；符号 `_normalize_messages`）：修复的核心文件：`_normalize_messages()` 中 `content` 类型白名单从 `text` 扩展为 `text/input_text/output_text`，是本次 bug 的直接根因与修复点。
- tests/tokenizers/_test_deepseek_v41.py（模块 回归测试；类别 test；类型 test-coverage；符号 `test_responses_text_parts_match_chat_text_parts`）：新增参数化回归测试，覆盖 `user/input_text` 与 `assistant/output_text` 两个组合，保证协议适配的修复被持续验证。

关键符号：`_normalize_messages`, `test_responses_text_parts_match_chat_text_parts`

关键源码片段

vllm/tokenizers/deepseek_v41.py

修复的核心文件：`_normalize_messages()` 中 `content` 类型白名单从 `text` 扩展为 `text/input_text/output_text`，是本次 bug 的直接根因与修复点。

```
def _normalize_messages(
    messages: list[ChatCompletionMessageParam],
) -> list[dict[str, Any]]:
    result = [dict(message) for message in copy.deepcopy(messages)]
    for message in result:
        role = message.get("role")
        if role not in ("system", "developer", "user", "assistant", "tool"):
            raise ValueError(f"Invalid role: {role}")
        if "reasoning" in message:
            message["reasoning_content"] = message["reasoning"]
        content = message.get("content")
        if isinstance(content, list):
            parts = []
            for block in content:
                part_type = block.get("type")
                # Responses API 使用 input_text / output_text, Chat API 使用 text,
                # 三者在此统一按文本追加，并保留块间分隔符 "
                # , 保证渲染结果一致
                if part_type in ("text", "input_text", "output_text"):
                    parts.append(block.get("text", ""))
                # 图片类占位符路径不参与本次修复，维持原行为
                elif part_type in ("image_url", "input_image", "image_pil"):
                    parts.append(IMAGE_PLACEHOLDER)
            else:
                raise ValueError(
                    "DeepSeek V4.1 supports text and image content only; "
                    f"got {part_type!r}"
```

```

    )
    message["content"] = "

.join(parts)
return result

```

tests/tokenizers_/test_deepseek_v41.py

新增参数化回归测试，覆盖 `user/input_text` 与 `assistant/output_text` 两个组合，保证协议适配的修复被持续验证。

```

@pytest.mark.parametrize(
    ("role", "responses_type"),
    [("user", "input_text"), ("assistant", "output_text")],
)
def test_responses_text_parts_match_chat_text_parts(role, responses_type):
    # 构造等价的 Responses 消息与 Chat 消息，仅 content 类型字段不同
    responses_message = {
        "role": role,
        "content": [{"type": responses_type, "text": "hello"}],
    }
    chat_message = {
        "role": role,
        "content": [{"type": "text", "text": "hello"}],
    }
    # 关键断言：responses 的 input_text / output_text 必须与 plain text 渲染结果一致
    assert render([responses_message], thinking=False) == render(
        [chat_message], thinking=False
    )

```

评论区精华

该 PR 没有实质性的技术争论，评论区主要由自动化工具构成：

- Claude Code指出该 PR 来自 fork，自动评审被禁用，需要维护者评论 @claude review 才能触发一次性审查。
- mergify[bot]提示 pre-commit 检查失败，要求作者重新运行并提交修复。
- chaunceyjiang多次触发 CI (/ci run) 并最终批准合并，仅回复“Thanks~”，没有留下额外技术意见。

整体属于典型的“小改动、无争议”合入流程。

- fork 仓库的自动评审限制 (other): 仓库对 fork PR 不运行自动 AI 评审，依靠人工维护者把关；本 PR 最终由 chaunceyjiang 人工批准。
- CI 与 pre-commit 流程 (other): 作者修复 pre-commit 问题后重新提交，后续 CI 通过并完成合并。

风险与影响

- 风险：风险整体很低：

- 回归风险: `input_text/output_text` 与 `text` 共用同一分支, 行为完全等价; 图片占位符路径未触碰, 不影响既有多模态请求。
- 测试盲区: 新增测试只覆盖单文本 `part` 的场景, 未覆盖“图片 + 文本”混合的 `Responses` 请求, 也未覆盖 `text` 字段缺失 (依赖既有的 `block.get("text", "")` 容错) 的边界情况, 但这部分属于既有逻辑, 并非本改动引入。
- 扩展性隐患: 白名单硬编码在 `tokenizer` 内部, 若未来 `Responses API` 引入新的文本类 `content` 类型 (如 `input_file` 等), 需要再次同步扩展; 目前尚无统一规范位置管理 `content` 类型别名。
- 无性能与安全影响: 改动不改动计算路径, 不影响模型执行与梯次推理。
- 影响: 影响范围集中在使用 `DeepSeek V4.1` 系列模型 (如 `deepseek-ai/DeepSeek-V4.1-Flash`) 并通过 `/v1/responses` 接口发起请求的用户: 修复前这类请求因 400 被拒, 修复后可正常走完推理。受影响的模块仅是 `vllm/tokenizers/deepseek_v41.py`, 不涉及调度、注意力、`KVCache` 等运行时路径, 因此对系统性能与稳定性无影响; 对团队而言是一次低成本的兼容性补丁, 测试覆盖了角色与类型的两个正交维度, 防止后续回归。
- 风险标记: 协议类型白名单硬编码, 测试未覆盖混合图文场景, 低回归风险

关联脉络

- PR #56366 [Bugfix][Rust Frontend][Multimodal] Align DeepSeek V4.1 and Kimi K3 media with rendered placeholders: 同属 `DeepSeek V4.1` 前端 / 渲染适配修复系列: 该 PR 处理媒体占位符渲染顺序错配, 本 PR 处理 `tokenizer` 文本类型兼容, 共同完善该模型在 `Responses` 前端链路的行为一致性。