PR #40399 完整报告

vllm-project/vllm

[Responses] Add tool_choice/tools validation to match OpenAI behavior

合并时间：2026-04-23 13:46

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/40399

执行摘要

- 一句话：为 Responses API 添加 tool_choice/tools 验证，对齐 OpenAI 行为。
- 推荐动作：此 PR 展示了如何利用 Pydantic 的 model_validator 在 API 入口层实现业务规则校验，代码简洁且可测试性强。推荐所有涉及用户输入验证的模块参考此实现模式。对于使用 Responses API 的开发者，值得了解其 tool_choice 默认行为的变化。

功能与动机

根据 PR body，目的是对齐 OpenAI Responses API 规范。之前 vLLM 对 tools 和 tool_choice 的非法组合（如无工具时指定 required 或命名工具选择）静默接受或产生意外行为。OpenAI 会返回明确的验证错误，本 PR 复现了这种严格行为。

实现拆解

1. 新增 Pydantic 模型校验器— 在 vllm/entrypoints/openai/responses/protocol.py 的 ResponsesRequest 类中添加了 check_tool_usage 方法（@model_validator(mode="before")），在数据解析前执行验证逻辑。 - 当无工具（tools 缺失或空列表）时，将 tool_choice 转换为 "none"（原默认 "auto"），并对 required 或命名选择抛出 VLLMValidationError。 - 当有工具且 tool_choice 为命名函数时，验证工具名称是否存在于 tools 列表中，不存在则报错。
2. 调整 Harmony 模式工具选择检查— 在 vllm/entrypoints/openai/responses/serving.py 的 _make_request_with_harmony 方法中，将 tool_choice 允许值从 "auto" 扩展为 "auto" 或 "none"，并更新错误提示信息。
3. 新增全面测试覆盖— 创建 tests/tool_use/test_responses_request_validations.py（新文件），包含 12 个测试函数，覆盖无工具 / 空工具 / 有工具下各种 tool_choice 组合的预期行为、合法命名工具匹配以及非法场景的错误验证。

关键文件：

- vllm/entrypoints/openai/responses/protocol.py（模块 入口层；类别 source；类型 core-logic；符号 check_tool_usage）：核心变更文件，新增 check_tool_usage 校验器，定义了所有验证逻辑。
- vllm/entrypoints/openai/responses/serving.py（模块 入口层；类别 source；类型 core-logic）：调整了 Harmony 模式下 tool_choice 允许范围，确保与前端验证一致。

- tests/tool_use/test_responses_request_validations.py（模块 测试；类别 test；类型 test-coverage；符号 test_responses_request_with_no_tools, test_responses_request_no_tools_tool_choice_none, test_responses_request_no_tools_tool_choice_auto, test_responses_request_required_without_tools）：新增 184 行测试，全面验证所有边界场景，确保校验逻辑正确。

关键符号：check_tool_usage, test_responses_request_with_no_tools, test_responses_request_no_tools_tool_choice_none, test_responses_request_no_tools_tool_choice_auto, test_responses_request_required_without_tools, test_responses_request_named_tool_choice_without_tools, test_responses_request_with_tools_default_tool_choice, test_responses_request_with_tools_tool_choice_none, test_responses_request_named_tool_choice_matching

关键源码片段

vllm/entrypoints/openai/responses/protocol.py

核心变更文件，新增 check_tool_usage 校验器，定义了所有验证逻辑。

```
# vllm/entrypoints/openai/responses/protocol.py#L495-L533
@model_validator(mode="before")
@classmethod
def check_tool_usage(cls, data):
    if not isinstance(data, dict):
        return data

    tools = data.get("tools")
    tool_choice = data.get("tool_choice", "auto") # 默认 "auto"
    has_tools = tools is not None and len(tools) > 0
    is_named_tool_choice = (
        isinstance(tool_choice, dict) and tool_choice.get("type") == "function"
    )

    # 情况 1: 无工具
    if not has_tools:
        if tool_choice in ("auto", "none"):
            data["tool_choice"] = "none" # 降级为 none
        elif tool_choice == "required":
            raise VLLMValidationError(
                "Tool choice 'required' must be specified with 'tools' parameter.",
                parameter="tool_choice",
            )
        elif is_named_tool_choice:
            raise VLLMValidationError(
                "Tool choice 'function' not found in 'tools' parameter.",
                parameter="tool_choice",
```

```
            )
    # 情况 2: 有工具，且为命名选择
    elif is_named_tool_choice and tools is not None:
        tool_name = tool_choice.get("name")
        tool_names = {
            t.get("name") if isinstance(t, dict) else getattr(t, "name", None)
            for t in tools
        }
        if not tool_name or tool_name not in tool_names:
            raise VLLMValidationError(
                "Tool choice 'function' not found in 'tools' parameter.",
                parameter="tool_choice",
            )
    return data
```

vllm/entrypoints/openai/responses/serving.py

调整了 Harmony 模式下 tool_choice 允许范围，确保与前端验证一致。

```
# vllm/entrypoints/openai/responses/serving.py#L716-L725
# 修改前：if request.tool_choice != "auto":
# 修改后：允许 "auto" 或 "none"
if request.tool_choice not in ("auto", "none"):
    raise NotImplementedError(
        "Only 'auto' or 'none' tool_choice is supported "
        "in response API with Harmony"
    )
```

tests/tool_use/test_responses_request_validations.py

新增 184 行测试，全面验证所有边界场景，确保校验逻辑正确。

```
# tests/tool_use/test_responses_request_validations.py（部分）
# 验证无 tools 时默认 tool_choice 为 none
def test_responses_request_with_no_tools():
    request = ResponsesRequest.model_validate({"input": "Hello", "model": "test-model"})
    assert request.tool_choice == "none"
    # 空 tools 列表同样
    request = ResponsesRequest.model_validate(
        {"input": "Hello", "model": "test-model", "tools": []}
    )
    assert request.tool_choice == "none"

# 验证 required 无 tools 时报错
@pytest.mark.parametrize("tools", [None, []])
def test_responses_request_required_without_tools(tools):
    kwargs = {"input": "Hello", "model": "test-model", "tool_choice": "required"}
    if tools is not None:
        kwargs["tools"] = tools
    with pytest.raises(
        ValidationError, match="Tool choice 'required' must be specified"
```

):
    ResponsesRequest.model_validate(kwargs)

评论区精华

gemini-code-assist[bot] 对 protocol.py 中工具名称提取逻辑提出质疑：

- 认为如果 tools 列表包含 None 或在 Pydantic 完全验证前工具模型为嵌套结构（如 tool.function.name），则 t.get("name") 可能返回 None 或导致 TypeError。
- sfeng33 回复指出：Responses API 的工具格式是扁平的（name 在顶层），Pydantic 会在字段类型验证阶段自动拒绝 None 等无效值，因此该评论无效。
- 最终 chaunceyjiang 批准了 PR，未发生实质性争议。
- 工具名称提取逻辑脆弱性 (correctness): sfeng33 回应：Responses API 工具格式是扁平的（name在顶层），Pydantic字段类型验证会先拒绝无效值，因此该评论不适用于此场景。PR 作者澄清后无进一步讨论。

风险与影响

- 风险：变更集中在请求校验层，不涉及推理核心或模型加载。主要风险：
  - 回归风险（低）：新的验证可能在边缘情况（如自定义工具格式）意外拒绝合法请求。由于测试覆盖了大多数组合，且验证发生在 mode="before"，Pydantic 类型检查后续仍会执行，风险可控。
  - Harmony 模式兼容性：_make_request_with_harmony 的修改允许 "none"，与 Harmony 集成相关，但若 Harmony 实现依赖 tool_choice 为 "auto"，可能引入误用。当前改动仅放宽检查，未影响下游逻辑。
  - 影响：
  - 用户：获得更严格的 API 验证，早期捕获配置错误，体验更接近 OpenAI。尤其工具调用开发者会受益于明确错误信息。
  - 系统：无性能影响，验证仅在请求构建时执行一次。
  - 团队：维护成本低，清晰的校验逻辑和内联注释易于后续扩展。
  - 风险标记：校验逻辑变更，兼容性边缘情况

关联脉络

- 暂无明显关联 PR