PR #28035 完整报告

sgl-project/sglang

fix(openai): validate assistant tool call arguments before chat template

合并时间：2026-06-16 20:21

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28035

执行摘要

- 一句话：修复历史工具调用参数未验证 JSON 对象导致的 500 错误
- 推荐动作：值得精读。该 PR 展示了如何在消息处理早期建立统一的参数验证层，避免临时性解析散落在各路径中。设计上选择在 _apply_jinja_template 入口处原地转换并深拷贝，兼顾了正确性与可维护性。建议关注空参数 edge case 的后续修复。

功能与动机

PR body 指出：OpenAI-compatible chat history sends assistant.tool_calls[].function.arguments as a JSON string. SGLang currently parses that string before passing messages to Hugging Face chat templates, but it does not verify that the parsed value is a JSON object. Those values can reach templates that call .items() on the parsed arguments and fail with 'str object' has no attribute 'items'. This scenario should actually throw a 400 bad request instead of a 500 Internal Error.

实现拆解

1. 新增核心验证函数（serving_chat.py）：parse_tool_call_arguments(arguments) 使用 orjson.loads 解析字符串，并检查结果是否为 dict，若失败或非对象则抛出 ValueError；normalize_assistant_tool_call_arguments(message) 遍历消息中的助理角色 tool_calls，原地将字符串 arguments 替换为解析后的 dict。
2. 集成到 Jinja 模板路径（serving_chat.py_apply_jinja_template）：在消息深拷贝后、传给编码器前统一调用 normalize_assistant_tool_call_arguments，确保后续所有子路径（包括 DSV 编码）拿到的 arguments 都是 dict。同时调整 deepcopy 位置避免重复解析。
3. 同步适配 DSV 编码路径（encoding_dsv4.py 和 encoding_dsv32.py 的 encode_arguments_to_dsml）：将原有的无条件 json.loads 改为先判断类型（字符串则解析），再校验是否为 dict，非对象直接 raise ValueError。
4. 新增单元测试（test_serving_chat.py）：四个测试用例 —— test_jinja_rejects_non_object_tool_call_arguments 验证 Jinja 路径拒绝字符串 / 数组参数；test_jinja_accepts_object_tool_call_arguments_string 验证正常对象字符串可转换为 dict；test_dsv_encoders_reject_non_object_tool_call_arguments 验证 dsV4/dsv32 编码路径拒绝标量；test_dsv_encoders_accept_object_tool_call_arguments_string 验证编码路径接受对象。

关键文件：

- python/sglang/srt/entrypoints/openai/serving_chat.py（模块 聊天服务；类别 source；类型 core-logic；符号 parse_tool_call_arguments, normalize_assistant_tool_call_arguments）：核心变更文件，新增 parse_tool_call_arguments 和 normalize_assistant_tool_call_arguments 函数，修改 _apply_jinja_template 集成验证逻辑和深拷贝流程。
- test/registered/unit/entrypoints/openai/test_serving_chat.py（模块 测试；类别 test；类型 test-coverage；符号 test_jinja_rejects_non_object_tool_call_arguments, test_jinja_accepts_object_tool_call_arguments_string, test_dsv_encoders_reject_non_object_tool_call_arguments, test_dsv_encoders_accept_object_tool_call_arguments_string）：新增 151 行单元测试，覆盖 Jinja 和 DSV 两条路径的非对象拒绝与对象接受场景，确保验证逻辑正确性。
- python/sglang/srt/entrypoints/openai/encoding_dsv4.py（模块 DSv4 编码；类别 source；类型 core-logic）：在 encode_arguments_to_dsml 中增加 arguments 类型判断与对象验证，保持与主路径一致的错误处理。
- python/sglang/srt/entrypoints/openai/encoding_dsv32.py（模块 DSv32 编码；类别 source；类型 core-logic）：与 encoding_dsv4.py 同步修改，在 encode_arguments_to_dsml 中添加相同验证。

关键符号：parse_tool_call_arguments, normalize_assistant_tool_call_arguments, encode_arguments_to_dsml

关键源码片段

python/sglang/srt/entrypoints/openai/serving_chat.py

核心变更文件，新增 parse_tool_call_arguments 和 normalize_assistant_tool_call_arguments 函数，修改 _apply_jinja_template 集成验证逻辑和深拷贝流程。

```
# 解析并验证工具调用参数必须为 JSON 对象
def parse_tool_call_arguments(arguments: str) -> Dict[str, Any]:
    """Parse OpenAI tool call arguments for chat templates."""
    try:
        parsed_arguments = orjson.loads(arguments)
    except orjson.JSONDecodeError as exc:
        raise ValueError(
            "Assistant tool call function.arguments must be valid JSON."
        ) from exc

    if not isinstance(parsed_arguments, dict):
        raise ValueError(
            "Assistant tool call function.arguments must be a JSON object."
        )

    return parsed_arguments

# 原地将助理消息中的 tool_calls 参数从 JSON 字符串转为 dict
```

```
def normalize_assistant_tool_call_arguments(message: Dict[str, Any]) -> None:
    """Normalize assistant history tool call arguments in-place."""
    if message.get("role") != "assistant" or not isinstance(
        message.get("tool_calls"), list
    ):
        return

    for item in message["tool_calls"]:
        function = item.get("function") if isinstance(item, dict) else None
        if not isinstance(function, dict):
            continue
        if "arguments" in function and isinstance(function["arguments"], str):
            function["arguments"] = parse_tool_call_arguments(function["arguments"])


# 在 _apply_jinja_template 入口统一进行规范化
messages = [msg.model_dump() for msg in request.messages]
for message in messages:
    normalize_assistant_tool_call_arguments(message)
```

评论区精华

gemini-code-assist[bot]在 parse_tool_call_arguments 处评论：若参数为空字符串或仅包含空白字符，orjson.loads 会抛 JSONDecodeError，导致该请求被 400 拒绝。建议默认返回空字典 {} 以增强健壮性。该建议未被作者采纳，PR 合并时保留了当前行为（空字符串直接拒绝）。

- 空参数处理建议 (correctness): 未采纳，PR 合并时保持当前行为：空参数直接抛出 JSONDecodeError，返回 400。

风险与影响

- 风险：
  1. 空参数兼容性风险：部分客户端可能发送空字符串或空白字符作为 arguments（函数无参数时），当前实现会将其视为无效 JSON 并返回 400，可能破坏原有正常调用。
  2. 消息处理流程变更：_apply_jinja_template 中提前对消息进行深拷贝并调用 normalize_assistant_tool_call_arguments，如果下游自定义编码器依赖原始字符串格式则可能受影响。
  3. 测试未覆盖空参数：新增的单元测试未包含空或仅空白字符串的用例，缺失 edge case 防护。 - 影响：用户侧：以往导致 500 的非法参数请求现在会返回 400，更符合 HTTP 语义，错误信息更明确。系统侧：增加少量解析开销（仅对包含 tool_calls 的历史助理消息），对大部分请求无影响。团队侧：代码库统一了参数验证入口，降低了未来模板崩溃的排查成本。 - 风险标记：空参数场景未处理，核心路径变更，消息深拷贝副作用

关联脉络

- 暂无明显关联 PR