PR #27614 完整报告

sgl-project/sglang

Fix LFM 2 tool parser.

合并时间：2026-07-30 18:15

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27614

执行摘要

- 一句话：修复 LFM2 工具解析器不支持带点函数名和 JSON 字面量
- 推荐动作：建议 LFM2 用户合入此修复。代码设计值得参考：通过字典映射统一处理多种字面量拼写，以及通过 ast.Attribute 遍历处理带点号函数名。

功能与动机

LFM2 模型输出的工具调用格式是 Pythonic 的，工具名可能包含点号（如 Calendar.create_event），但解析器只接受 ast.Name 节点，导致带点号调用被静默丢弃。同时，LFM2 会输出 true、false、null 等 JSON 字面量，解析器未识别这些小写形式。该 PR 修复这两个问题，使解析器正确解析参考解析器接受的合法输出。

实现拆解

1. 新增模块级常量 _PYTHONIC_NAME_LITERALS，将 True/False/None 与 true/false/null 统一映射到 Python 值。
2. 新增 _get_function_name 辅助方法，通过遍历 ast.Attribute 链并拼接，支持提取带点号的函数名（如 Calendar.create_event）。
3. 重构 _get_parameter_value 中处理 ast.Name 的分支，用字典查找替代 if-elif 链，以支持小写 JSON 字面量。
4. 修改 _parse_pythonic_call，使用 _get_function_name 替换原先的 ast.Name 类型检查和直接取 id，同时更新警告信息。

关键文件：

- python/sglang/srt/function_call/lfm2_detector.py（模块 工具调用；类别 source；类型 core-logic；符号 _PYTHONIC_NAME_LITERALS, _get_function_name, _get_parameter_value, _parse_pythonic_call）：唯一变更文件，包含了所有修复：添加模块级字面量映射、新增 _get_function_name 解析带点工具名、修改 _get_parameter_value 和 _parse_pythonic_call。

关键符号：_get_function_name, _get_parameter_value, _parse_pythonic_call

关键源码片段

python/sglang/srt/function_call/lfm2_detector.py

唯一变更文件，包含了所有修复：添加模块级字面量映射、新增 _get_function_name 解析带点工具名、修改 _get_parameter_value 和 _parse_pythonic_call。

```
# _PYTHONIC_NAME_LITERALS 字典同时支持 Python 标准拼写和 LFM2 JSON 小写字面量
_PYTHONIC_NAME_LITERALS = {
    "True": True,
    "False": False,
    "None": None,
    "true": True,
    "false": False,
    "null": None,
}

class Lfm2Detector(BaseFormatDetector):
    # ... 省略 init 和 has_tool_call ...

    def _get_parameter_value(self, val: ast.AST) -> Any:
        # 提取 Python 字面量值（支持嵌套结构）
        if isinstance(val, ast.Constant):
            return val.value
        elif isinstance(val, ast.Dict):
            return {self._get_parameter_value(k): self._get_parameter_value(v)
                    for k, v in zip(val.keys, val.values) if k is not None}
        elif isinstance(val, ast.List):
            return [self._get_parameter_value(v) for v in val.elts]
        elif isinstance(val, ast.Tuple):
            return tuple(self._get_parameter_value(v) for v in val.elts)
        elif isinstance(val, ast.Name):
            # 使用统一字典查找替代 if-elif，支持 true/false/null
            try:
                return _PYTHONIC_NAME_LITERALS[val.id]
            except KeyError:
                raise ValueError(f"Unsupported name reference: {val.id}") from None
        # 其他节点处理（UnaryOp、异常等）

    def _get_function_name(self, func: ast.AST) -> Optional[str]:
        # 从 AST 中提取函数名，支持带点号的属性链，如 Calendar.create_event
        parts: List[str] = []
        while isinstance(func, ast.Attribute):
            parts.append(func.attr)
            func = func.value
        if not isinstance(func, ast.Name):
            return None
        parts.append(func.id)
        return ".".join(reversed(parts))

    def _parse_pythonic_call(self, call, call_index, tool_indices):
        # 主要修改：使用 _get_function_name 替换直接检查 ast.Name
        function_name = self._get_function_name(call.func)
```

```
        if function_name is None:
            logger.warning(
                f"Tool call function must be a name or dotted name, got: {type(call.func).__name__}")
            return None
        # 后续验证和参数解析 ...
```

评论区精华

PR 作者在 body 中详细说明了两个 bug 的场景和表现。维护者 JustinTong0323 批准了 PR。CI 多次失败后，作者确认失败由 main 分支的预存断裂导致（#28002 修复中），与本次变更无关，随后维护者重新触发 CI。

- CI 失败与 PR 无关 (other): 确认 CI 失败与 PR 无关，不阻塞合并。

风险与影响

- 风险：风险较低。变更仅影响 LFM2 格式的工具调用解析路径，且只扩展了支持情况（之前被拒绝的合法输入现在被接受）。但缺少直接对应的单元测试（仅通过现存集成测试覆盖），可能遗漏边界情况如空属性链、混合点号名。
- 影响：对使用 LFM2 模型的用户影响较大：修复后带点号的工具调用（如 `Calendar.create_event`）能被正确识别，JSON 小写字面量也能被解析，使 SGLang 行为与参考解析器一致。不影响其他格式或模型。
- 风险标记：缺少测试覆盖

关联脉络

- PR #28002 Fix server_args test on CPU: 修复了 main 分支的预存 CI 断裂，被 PR 作者引用说明 CI 失败不相关。