PR #23476 完整报告

sgl-project/sglang

fix(tool_call): normalize non-standard JSON Schema types in tool params

合并时间：2026-05-26 15:24

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23476

执行摘要

- 一句话：规范化工具参数中非标准 JSON Schema 类型映射
- 推荐动作：该 PR 设计清晰，测试完备，建议合并。后续可考虑：
  1. 增加最大深度限制以防御堆栈溢出。
  2. 将 _PREFIX_BOUNDARY_CHARS 与前缀规则文档化，供用户参考。
  3. 在 CHANGELOG 中标注此兼容行为。

功能与动机

数据库/ORM 导出的工具参数 schema 常使用非标准类型名（如 "varchar"、"enum"、"int32"），这些会被 _validate_request 中 Draft202012Validator.check_schema 拒绝返回 HTTP 400，导致请求无法到达模型。该问题已在生产部署中报告。

实现拆解

1. 核心兼容层：在 python/sglang/srt/function_call/utils.py 中新增 normalize_json_schema_types 函数及其辅助 _normalize_single_type 和 _matches_type_prefix。前者递归遍历 schema 的 type 字段，将非标准字符串映射到标准类型；后者通过 _JSON_SCHEMA_TYPE_ALIASES 精确映射和 _PREFIX_RULES 前缀匹配处理参数化类型名（如 int32、list[str]）。
2. 请求验证钩子：在 python/sglang/srt/entrypoints/openai/serving_chat.py 的 _validate_request 方法中，在调用 Draft202012Validator.check_schema 之前插入 normalize_json_schema_types 调用，并捕获 RecursionError 以防御循环 schema 导致的崩溃。
3. 完整测试套件：新增 test/registered/unit/function_call/test_normalize_json_schema_types.py，包含 14 个测试用例，覆盖精确别名、前缀匹配边界、递归关键字（anyOf、$defs 等）、type 列表形式、大小写不敏感、括号参数剥离、未知类型保留，以及循环 schema 保护。所有测试均通过 CPU CI。

关键文件：

- test/registered/unit/function_call/test_normalize_json_schema_types.py（模块 测试覆盖；类别 test；类型 test-coverage；符号 TestNormalizeJsonSchemaTypes, _assert_accepts, test_enum_alias_becomes_string, test_varchar_alias_becomes_string）：新增 391 行测试，覆盖所有别名映射、前缀匹配

边界、递归关键字、type 列表、循环 schema 保护，确保核心逻辑正确性。

- python/sglang/srt/function_call/utils.py（模块 工具调用；类别 source；类型 core-logic；符号 _matches_type_prefix, _normalize_single_type, normalize_json_schema_types）：核心兼容层，新增 normalize_json_schema_types 和辅助函数，负责将非标准类型映射到标准 JSON Schema 类型。
- python/sglang/srt/entrypoints/openai/serving_chat.py（模块 请求入口；类别 source；类型 dependency-wiring）：在请求验证入口插入兼容层并捕获递归循环。

关键符号：normalize_json_schema_types, _normalize_single_type, _matches_type_prefix

关键源码片段

test/registered/unit/function_call/test_normalize_json_schema_types.py

新增 391 行测试，覆盖所有别名映射、前缀匹配边界、递归关键字、type 列表、循环 schema 保护，确保核心逻辑正确性。

```
"""工具参数 schema 别名归一化单元测试。"""
import json
import unittest
from jsonschema import Draft202012Validator, SchemaError
from sglang.srt.function_call.utils import normalize_json_schema_types
from sglang.test.ci.ci_register import register_cpu_ci
from sglang.test.test_utils import CustomTestCase

register_cpu_ci(1.0, "stage-a-test-cpu")


class TestNormalizeJsonSchemaTypes(CustomTestCase):
    def _assert_accepts(self, schema: dict) -> None:
        # 验证归一化后的 schema 能通过 Draft 2020-12 校验
        Draft202012Validator.check_schema(schema)

    def test_enum_alias_becomes_string(self):
        schema = {
            "type": "object",
            "properties": {
                "color": {"type": "enum", "enum": ["red", "green", "blue"]}
            },
        }
        # ``enum`` 应被归一化为 ``string``
        normalize_json_schema_types(schema)
        self.assertEqual(schema["properties"]["color"]["type"], "string")
        self._assert_accepts(schema)

    def test_varchar_alias_becomes_string(self):
        schema = {
            "type": "object",
            "properties": {
```

```
            "name": {"type": "varchar"},
            "short_name": {"type": "VARCHAR(255)"}, # 括号参数应被剥离
        },
    }
    normalize_json_schema_types(schema)
    self.assertEqual(schema["properties"]["name"]["type"], "string")
    self.assertEqual(schema["properties"]["short_name"]["type"], "string")
    self._assert_accepts(schema)

def test_prefix_matched_numeric_types(self):
    # 前缀匹配需处理 ``int32``、``float64``、``long long`` 等参数化名称
    schema = {
        "type": "object",
        "properties": {
            "a": {"type": "int32"},
            "b": {"type": "numeric"},
        },
    }
    normalize_json_schema_types(schema)
    self.assertEqual(schema["properties"]["a"]["type"], "integer")
    self.assertEqual(schema["properties"]["b"]["type"], "number")
    self._assert_accepts(schema)

def test_word_boundary_prevents_false_positives(self):
    # 自定义类型 ``internal`` 不应被误判为 ``integer``
    schema = {
        "type": "object",
        "properties": {"a": {"type": "internal"}},
    }
    normalize_json_schema_types(schema)
    self.assertEqual(schema["properties"]["a"]["type"], "internal")
```

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

在请求验证入口插入兼容层并捕获递归循环。

```
  def _validate_request(self, request: ChatCompletionRequest) -> Optional[str]:
    ...
    # 验证工具定义
    for i, tool in enumerate(request.tools or []):
        if tool.function.parameters is None:
            continue
        try:
            # 调用 normalize 将非标准类型转换为标准类型
            normalize_json_schema_types(tool.function.parameters)
            Draft202012Validator.check_schema(tool.function.parameters)
        except SchemaError as e:
            return f"Tool {i} function has invalid 'parameters' schema: {str(e)}"
        except RecursionError:
            # 防御手工构造的循环 schema 导致崩溃
```

```
            return (
                f"Tool {i} function 'parameters' schema is too deeply nested "
                "or contains a cycle."
            )
    ...
```

评论区精华

审查中 gemini-code-assist[bot] 提出两点关键改进：

- 前缀匹配误判：使用 startswith 过于宽泛，如 internal 会被误映射为 integer。建议要求前缀后紧跟非字母字符（数字、括号、空格）。作者采纳后引入 _PREFIX_BOUNDARY_CHARS 并在 _matches_type_prefix 中检查 base[len(p)] in _PREFIX_BOUNDARY_CHARS。
- JSON Schema 关键字完整性：缺少对 dependentSchemas、unevaluatedItems、propertyNames 等关键字的遍历。作者在提交中添加了对 dependentSchemas 的支持，并涵盖更多嵌套关键字。
  - 前缀匹配边界字符检查 (correctness): 作者在后续提交中引入 _PREFIX_BOUNDARY_CHARS 集合并在 _matches_type_prefix 中检查 base[len(p)] in _PREFIX_BOUNDARY_CHARS，解决此问题。
  - 增加更多 JSON Schema 2020-12 关键字支持 (correctness): 作者在提交中添加了 dependentSchemas 到字典遍历列表，并扩充了直接 schema 遍历的关键字集合。

风险与影响

- 风险：
  - 前缀匹配仍可能误判：虽然已加入边界字符检查，但自定义类型如 list123 仍会被前缀匹配捕获（因为数字是边界字符），这符合预期。但若用户定义了 list_price 作为独立类型，且其含义不是数组，则会被错误归一化。然而根据 JSON Schema 惯例，list 前缀暗示数组，所以风险可控。
  - 递归深度：schema 可能很深（如包含 $defs 递归），normalize_json_schema_types 的递归实现可能导致栈溢出。已通过 RecursionError 捕获返回 400，但深度过大时仍可能触发系统栈溢出导致进程崩溃。可考虑增加最大深度限制。
  - 原地修改：直接修改传入的 dict 对象，如果同一 schema 被多处引用，可能导致其他部分意外更改。但请求对象通常是新建的，影响有限。
- 影响：
  - 用户：之前因非标准类型被拒绝的请求现在正常执行，零配置即可受益。
  - 系统：请求验证路径增加 O(N) 递归遍历，N 为 schema 节点数，通常很小，不影响推理吞吐。
  - 团队：降低了工具函数定义的兼容性门槛，用户可从主流框架导出 schema 直接使用。
  - 风险标记：前缀匹配误判（已缓解），递归循环保护已添加，原地修改可能影响引用者

关联脉络

- 暂无明显关联 PR