

AI Agent Function Calling

智能体函数调用

2026年5月24日 | AI Agent 知识分享 | 第4期

一、一句话概括

函数调用 (Function

Calling) 让AI模型通过声明式API描述调用外部工具的能力，变「只会说」为「还能做」——Agent通过Function Calling与大千世界交互，完成API调用、数据库查询、文件操作等真实任务，是Agent「知行合一」的关键桥梁。

二、核心原理

>> 声明式工具定义 (Declarative Tool Schema)

开发者以JSON

Schema格式定义函数的名称、参数和描述，模型在生成回复时选择「是否调用」及「传入什么参数」。核心是将工具的能力边界和调用方式以结构化方式告知模型，而非像传统prompt工程那样写在自然语言中。

>> 调用-执行-返回闭环 (Tool Call Loop)

模型返回函数调用请求（含函数名和参数）→ 应用执行实际函数（API调用、数据库查询等）→ 执行结果返回给模型 → 模型基于结果生成最终答复。这个3步循环是Agent执行真实操作的基本单元，每次调用都是Agent与外部世界的一次交互。

>> 并行与链式调用 (Parallel & Chained Calls)

支持一次调用多个独立函数（如同时查天气和日程），也支持依赖链式调用（第一步的结果作为第二步的输入参数）。OpenAI和Anthropic均原生支持并行调用，极大提升了Agent执行多步骤任务时的效率。

核心要点：JSON Schema 的质量直接决定 Agent 选择工具的准确率

工具描述精度每提升10%，工具选择准确率约提升3-5个百分点

三、测试实战

测试一：工具选择准确性测试

操作：定义2-3个功能相似的工具有如 `search_user_by_name` 和 `search_user_by_id`，构造故意模糊的查询（如「帮我找一下张三」），观察Agent能否选对工具

工具：OpenAI Playground (<https://platform.openai.com/playground>) 或 LangSmith Hub (<https://smith.langchain.com/hub>)

指标：工具选择准确率 (Accuracy)、Top-3召回率、误选率 (False Positive Rate)

测试二：边界参数与异常注入测试

操作：构造空字符串("")、超长字符(10万token)、SQL注入语法、特殊Unicode等异常参数传入工具，观察是否导致Agent崩溃、泄露信息或执行异常操作

工具：Burp Suite (<https://portswigger.net/burp>) 或 Postman 做参数模糊测试

指标：异常参数处理率、崩溃率 (Crash Rate)、错误信息泄露率

测试三：返回结果幻觉测试

操作：让工具返回预设的「查无结果」「数据不存在」等空结果，观察Agent是如实告知用户「未找到相关信息」，还是编造虚构的数据填充回答

工具：自定义Mock Server (Python Flask即可) 控制工具返回值

指标：幻觉率 (Hallucination Rate) = 编造数据次数 ÷ 空结果返回总次数

测试四：多工具并行调用稳定性测试

操作：让Agent同时调用3-5个工具（部分正常返回、部分超时、部分报错），验证Agent能否正确合并多路返回结果，不遗漏、不混淆、不重复

工具：K6 (<https://k6.io/>) 或 Locust 模拟并发调用场景

指标：结果合并完整率、调用顺序正确率、超时处理正确率

四、常见坑点

坑点1：工具描述不清导致Agent选错工具

JSON

Schema中的description字段质量直接影响Agent的决策。模糊的描述（如「搜索用户」）会让模型选错工具。正确做法是写清楚什么场景用什么工具，如「当用户提供手机号/邮箱时用search_user_by_id，提供姓名时用search_user_by_name」。建议每个工具描述包含：使用场景、参数说明、返回格式示例。

坑点2：参数类型校验缺失

模型有时会传字符串给期望整数的参数，或缺失必填参数（由于模型输出概率性）。必须在调用层加严格的输入校验层（Validation Layer），使用Pydantic/Zod做严格的参数类型检查，否则生产环境会出现奇怪的运行时错误。

坑点3：无限循环调用陷阱

Agent可能在工具返回值中提取到新的可执行信息后陷入无限调用。例如Agent调用了一个列表查询工具，返回了100条记录，Agent可能为了处理每条记录而不断调用处理函数，导致无限循环。必须在Agent循环中设置最大调用次数（max_turns=10~25）、超时限制和递归深度限制。

五、延伸阅读

- OpenAI Function Calling 官方指南: <https://platform.openai.com/docs/guides/function-calling>
- Anthropic Tool Use 文档: <https://docs.anthropic.com/en/docs/build-with-claude/tool-use>
- Toolformer (Meta) 论文: <https://arxiv.org/abs/2302.04761>
- OpenAI Parallel Function Calling: <https://platform.openai.com/docs/guides/function-calling#parallel-function-calling>