PR #27866 完整报告

sgl-project/sglang

examples: add Chain-of-Verification (CoVe) hallucination reduction demo

合并时间：2026-06-12 16:57

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27866

执行摘要

- 一句话：添加 CoVe 幻觉减少示例
- 推荐动作：对于想了解或使用 CoVe 模式的研究者和工程师，建议精读该示例，尤其是验证会话隔离的设计。对普通用户，可以作为脚本直接运行。

功能与动机

提供一个开箱即用的演示，展示如何通过 Factored CoVe 模式减少 LLM 幻觉。PR body 强调验证步骤使用全新会话（无共享 KV-cache）以避免模型自我确认，从而提升真实性。

实现拆解

1. 入口与参数解析(parse_args / main)：支持 --prompt、--base-url、--model、--max-tokens、--temperature、--summarize 等参数，默认连接本地 http://127.0.0.1:30000/v1。
2. 模型发现(resolve_model)：若未指定模型 ID，自动从 /v1/models 获取第一个可用模型。
3. 通用 chat 函数(chat)：封装 OpenAI 兼容 API 调用，兼容推理模型（如 Kimi-K2.5）的 reasoning_content fallback 逻辑。
4. 核心 CoVe 管线(chain_of_verification)：
   - Step 1 Draft：用用户查询生成初始回答。
   - Step 2 Verify：用全新会话（无历史）构造验证提示，调用严格的 fact-checker system prompt，要求返回 PASS/FAIL。
   - Step 3 Refine：若验证失败，用 REFINE_INSTRUCTION 引导模型修正。
   - Step 4 Summarize（可选）：压缩最终答案为一段。
5. 文档更新：在 README.md 示例列表中加入新文件链接和一句描述。

关键文件：

- examples/runtime/chain_of_verification.py（模块 示例；类别 source；类型 core-logic；符号 resolve_model, chat, chain_of_verification, _log）：新增的 CoVe 示例脚本，实现了完整的 Factored Chain-of-Verification 流程，是本 PR 的核心内容。
- examples/runtime/README.md（模块 文档；类别 docs；类型 documentation）：文档文件，新增了一行指向新 CoVe 示例的链接和简要说明，便于用户发现。

关键符号：resolve_model, chat, chain_of_verification, main

关键源码片段

examples/runtime/chain_of_verification.py

新增的 CoVe 示例脚本，实现了完整的 Factored Chain-of-Verification 流程，是本 PR 的核心内容。

```
# 验证步骤的系统提示：要求严格检查答案是否准确和相关
VERIFY_SYSTEM_PROMPT = (
    "You are a strict fact-checker. "
    "You will be given a user question and a candidate answer. "
    "Decide whether the answer is accurate and directly addresses the question. "
    "Reply with exactly one of: PASS or FAIL, followed by a brief reason."
)

# 修正指令：告知模型前次回答被标记为不准确
REFINE_INSTRUCTION = (
    "The previous answer was flagged as inaccurate or off-topic. "
    "Please provide a corrected, accurate answer to the original question."
)

# 可选总结指令
SUMMARIZE_INSTRUCTION = (
    "Please give a concise, one-paragraph version of the verified answer above."
)

def chat(
    client: OpenAI,
    model: str,
    messages: list[dict[str, Any]],
    max_tokens: int,
    temperature: float,
) -> str:
    """调用 OpenAI 兼容 API 并返回响应文本。
    处理推理模型可能将结果放在 reasoning_content 的情况。
    """
    response = client.chat.completions.create(
        model=model,
        messages=messages,
        max_tokens=max_tokens,
        temperature=temperature,
    )
    msg = response.choices[0].message
    content = msg.content or ""
    if not content.strip():
        content = getattr(msg, "reasoning_content", None) or ""
    return content
```

评论区精华

PR 无实质性 Review 讨论，仅有一条机器人自动提示配额已满。维护者直接批准合并。

- 暂无高价值评论线程

风险与影响

- 风险：该变更仅涉及 *examples/runtime* 目录下的示例和文档，不影响核心框架或生产逻辑。风险极低，但需注意：
  - 示例依赖 *openai* 包，若用户未安装会显示友好错误并退出。
  - 示例假定本地已有运行的 SGLang 服务器，用户需预先启动。
  - 影响：对用户：提供了可复现的 *CoVe* 实现，便于学习和实验；通过 README 新增条目提升发现性。 对系统：无影响，示例代码不参与 SRT 运行。 对团队：可作为官方推荐的反幻觉模式参考。影响范围小（2 个文件），程度低。
- 风险标记：外部依赖 *openai*，示例需先启动服务器

关联脉络

- 暂无明显关联 PR