PR #26864 完整报告

sgl-project/sglang

Fix multimodal synthetic benchmark prompt generation to exclude special tokens

合并时间：2026-06-05 06:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26864

执行摘要

- 一句话：修复多模态基准测试提示生成中特殊标记污染问题
- 推荐动作：建议合并。此 PR 修复了真实用户发现的多模态基准测试数据生成正确性问题，代码变更简洁，有单元测试覆盖，且通过了 review 的讨论和验证。

功能与动机

在运行多模态合成基准测试（--dataset-name image）时，gen_mm_prompt 生成的提示文本可能包含如 <|vidéo_padl|> 等特殊标记，而这些标记在没有对应 payload 的情况下会导致服务器返回 No data iterator found for token 错误，使得基准测试结果不可靠。该问题在 Qwen3-VL 等模型中复现，且通过 payload 审计确认 8/12900 条提示包含被禁止的特殊标记。此 PR 旨在从生成侧剔除所有特殊标记，确保基准测试数据的合法性。

实现拆解

1. 提取辅助函数并添加缓存：在 python/sglang/benchmark/datasets/common.py 中新增 get_available_multimodal_text_tokens(tokenizer, image_pad_id)，使用 @lru_cache(maxsize=1) 装饰以避免重复词汇表扫描。该函数从 tokenizer.all_special_ids 构建排除集，并额外将 image_pad_id 加入排除集（若不为 None）。最终返回 get_available_tokens(tokenizer) 中不在排除集中的 token id 列表。
2. 更新 gen_mm_prompt：将原本直接调用 tokenizer.get_vocab().values() 并逐一移除 image_pad_id 的逻辑，改为调用新的 get_available_multimodal_text_tokens 函数，简化实现并统一过滤逻辑。同时修复了 image_pad_id 检查条件，从 if image_pad_id: 改为 if image_pad_id is not None: 以正确处理 token id 为 0 的情况。
3. 添加单元测试：在 test/registered/bench_fn/test_benchmark_datasets_api.py 中新增 test_gen_mm_prompt_excludes_special_tokens 方法。使用 create_lightweight_tokenizer 构造包含多模态特殊标记的轻量 tokenizer，通过 mockrandom.choices 捕获候选池，断言候选池中不包含任何特殊标记的 token id，并确保候选池非空。

关键文件：

- python/sglang/benchmark/datasets/common.py（模块 基准生成；类别 source；类型 core-logic；符号 get_available_multimodal_text_tokens）：核心变更，新增过滤辅助函数并修改 gen_mm_prompt 以正确排除所有特殊标记。

- test/registered/bench_fn/test_benchmark_datasets_api.py（模块 基准测试；类别 test；类型 test-coverage；符号 test_gen_mm_prompt_excludes_special_tokens, fake_choices）：新增确定性单元测试，验证特殊标记排除逻辑。

关键符号：get_available_multimodal_text_tokens, gen_mm_prompt

关键源码片段

python/sglang/benchmark/datasets/common.py

核心变更，新增过滤辅助函数并修改 gen_mm_prompt 以正确排除所有特殊标记。

```
import random
from functools import lru_cache

# 复用已缓存的通用有效 Token 获取函数
from sglang.benchmark.datasets.common import get_available_tokens

# 缓存装饰器确保多模态文本 Token 池只计算一次
@lru_cache(maxsize=1)
def get_available_multimodal_text_tokens(tokenizer, image_pad_id):
    """Get valid token ids for synthetic multimodal text prompts."""
    # 收集所有特殊 Token 的 ID，防御性地使用 or [] 避免 None
    excluded_token_ids = set(getattr(tokenizer, "all_special_ids", []) or [])
    if image_pad_id is not None:
        excluded_token_ids.add(image_pad_id)
    # 仅从已缓存的通用池中过滤，避免重复扫描完整词表
    return [
        token_id
        for token_id in get_available_tokens(tokenizer)
        if token_id not in excluded_token_ids
    ]


def gen_mm_prompt(tokenizer, image_pad_id, token_num):
    """Generate a random prompt of specified token length using tokenizer vocabulary."""
    # 直接使用过滤后的多模态安全池
    all_available_tokens = get_available_multimodal_text_tokens(tokenizer, image_pad_id)
    selected_tokens = random.choices(all_available_tokens, k=token_num)
    return tokenizer.decode(selected_tokens)
```

test/registered/bench_fn/test_benchmark_datasets_api.py

新增确定性单元测试，验证特殊标记排除逻辑。

```
def test_gen_mm_prompt_excludes_special_tokens(self):
    # 创建轻量 tokenizer，包含标准词汇
    tokenizer = create_lightweight_tokenizer()
    # 注入一组多模态特殊标记（模拟 Qwen3-VL 等模型）
    multimodal_special_tokens = [
        "<|image_pad|>",
```

```
        "<|video_pad|>",
        "<|vision_start|>",
        "<|vision_end|>",
        "<|vision_pad|>",
    ]
    tokenizer.add_special_tokens(
        {"additional_special_tokens": multimodal_special_tokens}
    )
    special_token_ids = set(
        tokenizer.convert_tokens_to_ids(multimodal_special_tokens)
    )
    image_pad_id = tokenizer.convert_tokens_to_ids("<|image_pad|>")

    # 使用字典捕获 random.choices 被调用时的候选池
    captured_population = {}

    def fake_choices(population, k):
        captured_population["tokens"] = population
        return population[:k]

    # 替换 random.choices 以拦截候选池，避免真正随机采样
    with patch(
        "sglang.benchmark.datasets.common.random.choices",
        side_effect=fake_choices,
    ):
        gen_mm_prompt(tokenizer, image_pad_id, token_num=8)

    sampled_pool = set(captured_population["tokens"])
    # 断言候选池中没有任何特殊标记
    self.assertFalse(special_token_ids & sampled_pool)
    # 断言候选池非空
    self.assertTrue(sampled_pool)
```

评论区精华

Review 过程中有两个核心讨论：

- 性能优化：Gemini Code Assist 建议给新增函数添加 @lru_cache(maxsize=1)，避免每次 gen_mm_prompt 都重复扫描词汇表。作者采纳并实现，与现有 get_available_tokens 模式一致。
- 设计简化：JustinTong 指出新函数可以直接复用已有的 get_available_tokens（已缓存）而无需重新调用 tokenizer.get_vocab()，避免冗余扫描和 isinstance 守卫检查。最终实现改为基于 get_available_tokens 构建过滤，代码更简洁。
  - 添加 lru_cache 避免重复词汇表扫描 (performance): 作者采纳建议，在函数定义前添加了 @lru_cache(maxsize=1) 装饰器，与同文件中 get_available_tokens 的模式一致。
  - 复用 get_available_tokens 避免重复扫描 (design): 最终实现改为遍历 get_available_tokens(tokenizer) 的结果，将 isinstance 检查委托给通用函数，代码更

简洁且避免重复。

风险与影响

- 风险：此变更仅影响基准测试数据生成路径，不涉及模型前向、解码或服务性能。主要风险在于 tokenizer.all_special_ids 可能在不兼容的 tokenizer 中返回非预期值（如空列表），但代码中已有防御性取值 getattr(tokenizer, "all_special_ids", []) or []。此外，缓存机制假设 tokenizer 和 image_pad_id 在单次基准测试运行中不会变化，这符合现有使用场景。单元测试覆盖了典型的 Qwen3-VL 特殊标记集合，但仍需注意其他 tokenizer 引入的新特殊标记类型。
- 影响：直接用户：执行 python -m sglang.bench_serving --dataset-name image 的基准测试用户将不再因特殊标记导致 HTTP 400 错误，基准测试结果更加稳定。间接用户：依赖 SGLang 基准测试结果进行模型评估的开发者和 CI 系统将获得更可靠的数据。影响范围限定在基准测试模块，不影响生产推理路径。团队维护：新增的辅助函数与现有 get_available_tokens 风格一致，易于理解和扩展。
- 风险标记：数据生成路径变更

关联脉络

- 暂无明显关联 PR