PR #26555 完整报告

sgl-project/sglang

[RL+VLM] Avoid retokenization drift for pre-tokenized (token-id) VLM requests

合并时间：2026-06-02 00:58

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26555

执行摘要

- 一句话：修复 VLM token-id 请求的重分词漂移问题
- 推荐动作：值得精读。该 PR 解决了 RL 训练中一个隐蔽但严重的不一致问题，设计方案清晰：通过可覆盖方法分离计数逻辑，核心路径保留 fallback。建议关注后续对 video/audio 的扩展以及 Kimi 模型的端到端测试补充。

功能与动机

RL 训练要求推理侧与训练侧 token 序列完全一致（token-in token-out），但 VLM 路径在收到 input_ids 后会执行 *decode* → 重新 tokenize，若原始 token 序列非规范（如 'Describe' 拆为 'D'+'escribe'），重新 tokenize 的结果可能不同，导致 prompt 长度变化，进而使按 token 索引的 *expert routing* 重放错位，造成训练失败。PR body 中给出了具体示例：trainer prompt = [1, 2] 经 *decode→retokenize* 后可能变成 [3]，长度不匹配。

实现拆解

实现分为三步：

1. 新增 token 计数方法：在 BaseMultimodalProcessor 中添加 resolve_image_token_counts(self, images)，利用 transformers 处理器的 _get_num_multimodal_tokens 获取每张图像的扩展 token 数；Kimi 模型在 KimiGridMMDataMixin 中覆盖该方法，使用 media_tokens_calculator 计算。失败时抛出异常，由调用方捕获并回退到 *decode+retokenize*。
2. 新增 _expand_input_ids 静态方法：遍历原始 token 列表，将图像占位符替换为 counts[i] 个副本，其余 token 原样保留。确保占位符数与图像数匹配，否则抛 ValueError。
3. 修改 process_and_combine_mm_data 执行路径：当 SGLANG_MM_AVOID_RETOKENIZE 开启、请求为 input_ids 且包含图像（无音频 / 视频）时，调用上述方法从原始 token 重建 input_ids，替代 HF 处理器的重新 tokenize 结果。异常时记录日志并回退。
4. 环境变量注册：在 environ.py 中声明 SGLANG_MM_AVOID_RETOKENIZE = EnvBool(True)。
5. 测试配套：新增端到端测试，使用非规范 prompt（'Describe' 拆开）对比 flag 开 / 关时的 prompt_tokens 差异。

关键文件：

- python/sglang/srt/multimodal/processors/base_processor.py（模块 多模态处理；类别 source；类型 core-logic；符号 resolve_image_token_counts, _expand_input_ids）：核心变更文件，新增 resolve_image_token_counts 和 _expand_input_ids 方法，并修改 process_and_combine_mm_data 执行路径，是避免重分词漂移的主逻辑所在。
- test/registered/vlm/test_token_id_retokenize_e2e.py（模块 测试；类别 test；类型 test-coverage；符号 TestQwenVLTokenIdRetokenize, test_flag_off_drifts_flag_on_does_not）：新增端到端测试，唯一保留的测试文件，验证 Qwen2.5-VL 在 flag 开关下的 prompt_tokens 变化，确保漂移消除逻辑正确。
- python/sglang/srt/multimodal/processors/kimi_common.py（模块 多模态处理；类别 source；类型 core-logic；符号 resolve_image_token_counts）：Kimi 模型覆盖 resolve_image_token_counts，适配其私有 media_tokens_calculator，确保该模型也能受益于避免漂移逻辑。
- python/sglang/srt/environ.py（模块 环境配置；类别 source；类型 configuration）：注册 SGLANG_MM_AVOID_RETOKENIZE 环境变量，默认 True，作为功能开关。

关键符号：BaseMultimodalProcessor.resolve_image_token_counts, BaseMultimodalProcessor._expand_input_ids, BaseMultimodalProcessor.process_and_combine_mm_data, KimiGridMMDataMixin.resolve_image_token_counts

关键源码片段

python/sglang/srt/multimodal/processors/base_processor.py

核心变更文件，新增 resolve_image_token_counts 和 _expand_input_ids 方法，并修改 process_and_combine_mm_data 执行路径，是避免重分词漂移的主逻辑所在。

```
# python/sglang/srt/multimodal/processors/base_processor.py
# ( 新增方法，位于 _wrap_tensor_for_cuda_ipc 之后 )

    def resolve_image_token_counts(self, images: List) -> List[int]:
        """计算每张图像扩展后的 token 数，避免 re-tokenize。

        默认实现使用 transformers 的约定方法
        ``_get_num_multimodal_tokens(image_sizes=...)``,
        Kimi 等模型会覆盖此方法。
        """
        assert images is not None
        image_sizes = [(image.height, image.width) for image in images]
        num_image_tokens = self._processor._get_num_multimodal_tokens(
            image_sizes=image_sizes
        ).num_image_tokens
        return [int(count) for count in num_image_tokens]

    @staticmethod
    def _expand_input_ids(
        original_ids: List[int],
```

```
        counts: List[int],
        placeholder_token_id: Optional[int],
    ) -> List[int]:
        """从原始 token 列表重建最终 input_ids。

        将第 i 个图像占位符扩展为 ``counts[i]`` 个副本，
        非媒体 token 保持不变，从而避免重分词漂移。
        """
        if placeholder_token_id is None:
            raise ValueError("placeholder_token_id is not set for this processor")

        # 验证占位符数量与图像数一致
        num_placeholders = sum(
            1 for token_id in original_ids if token_id == placeholder_token_id
        )
        if num_placeholders != len(counts):
            raise ValueError(
                f"prompt has {num_placeholders} image placeholder token(s) but "
                f"{len(counts)} image(s) were provided"
            )

        rebuilt: List[int] = []
        next_image_idx = 0
        for token_id in original_ids:
            if token_id == placeholder_token_id:
                rebuilt.extend([placeholder_token_id] * counts[next_image_idx])
                next_image_idx += 1
            else:
                rebuilt.append(token_id)
        return rebuilt
```

test/registered/vlm/test_token_id_retokenize_e2e.py

新增端到端测试，唯一保留的测试文件，验证 Qwen2.5-VL 在 flag 开关下的 prompt_tokens 变化，确保漂移消除逻辑正确。

```
# test/registered/vlm/test_token_id_retokenize_e2e.py ( 完整文件 )

import base64
import io
import unittest
import requests
from PIL import Image
from transformers import AutoProcessor
from sglang.srt.utils import kill_process_tree
from sglang.test.ci.ci_register import register_cuda_ci
from sglang.test.test_utils import (
    DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH,
    DEFAULT_URL_FOR_TEST,
    CustomTestCase,
```

```
    popen_launch_server,
)

register_cuda_ci(est_time=300, suite="base-b-test-1-gpu-large")


def _data_uri():
    """生成一张 64x64 灰色 PNG 的 data URI"""
    img = Image.new("RGB", (64, 64), (128, 128, 128))
    buf = io.BytesIO()
    img.save(buf, format="PNG")
    return "" + base64.b64encode(buf.getvalue()).decode()


def _build_drift_prompt(model, image_token):
    """构造非规范 prompt: "Describe" 拆为 "D"+"escribe",
    并计算漂移量 (原始长度 - 规范重编码长度)。"""
    tok = AutoProcessor.from_pretrained(
        model, trust_remote_code=True, use_fast=True
    ).tokenizer

    def enc(text):
        return tok.encode(text, add_special_tokens=False)

    input_ids = enc("D") + enc("escribe") + enc(" the picture: ") + enc(image_token)
    canonical = enc(tok.decode(input_ids))
    drift_delta = len(input_ids) - len(canonical)
    return input_ids, drift_delta


def _prompt_tokens(base_url, input_ids, image):
    """发送生成请求并返回 prompt_tokens"""
    resp = requests.post(
        base_url + "/generate",
        json={
            "input_ids": input_ids,
            "image_data": [image],
            "sampling_params": {"temperature": 0.0, "max_new_tokens": 1},
        },
        timeout=300,
    )
    resp.raise_for_status()
    return resp.json()["meta_info"]["prompt_tokens"]


class TestQwenVLTokenIdRetokenize(CustomTestCase):
    model = "Qwen/Qwen2.5-VL-3B-Instruct"
    image_token = "<|vision_start|><|image_pad|><|vision_end|>"
    other_args = ["--trust-remote-code", "--mem-fraction-static", "0.7"]
```

```
    def test_flag_off_drifts_flag_on_does_not(self):
        input_ids, drift_delta = _build_drift_prompt(self.model, self.image_token)
        self.assertGreater(drift_delta, 0, "prompt is canonical; no drift to exercise")
        image = _data_uri()

        prompt_tokens = {}
        for flag in ("0", "1"):
            process = popen_launch_server(
                self.model,
                DEFAULT_URL_FOR_TEST,
                timeout=DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH,
                other_args=self.other_args,
                env={"SGLANG_MM_AVOID_RETOKENIZE": flag},
            )
            try:
                prompt_tokens[flag] = _prompt_tokens(
                    DEFAULT_URL_FOR_TEST, input_ids, image
                )
            finally:
                kill_process_tree(process.pid)

        # flag=ON 应当保留原始长度，flag=OFF 失去漂移量
        pt_off, pt_on = prompt_tokens["0"], prompt_tokens["1"]
        self.assertEqual(pt_on - pt_off, drift_delta, f"on={pt_on}, off={pt_off}")


if __name__ == "__main__":
    unittest.main()
```

python/sglang/srt/multimodal/processors/kimi_common.py

Kimi 模型覆盖 resolve_image_token_counts，适配其私有 media_tokens_calculator，确保该模型也能受益于避免漂移逻辑。

```
# python/sglang/srt/multimodal/processors/kimi_common.py

class KimiGridMMDataMixin:
    # ... 其他方法

    def resolve_image_token_counts(self, images):
        """Kimi 的处理器基于 remote code，未实现 transformers
        ``_get_num_multimodal_tokens`` 约定，故通过 media_tokens_calculator 计算。"""
        assert images is not None
        media_tokens_calculator = (
            self._processor.media_processor.media_tokens_calculator
        )
        return [
            int(media_tokens_calculator({"type": "image", "image": image}))
            for image in images
        ]
```

评论区精华

- gemini-code-assist建议在早期 kimi_token_ids.py（后删除）中利用 image_offsets 直接计算 token 数量，避免重复调用 media_tokens_calculator；该文件在后续迭代中被移除，优化未采用。
- mickqian询问是否需同时处理 video/audio 占位符，ByronHsu回复目前无 RL 用例，可后续扩展。
- 优化建议：从 image_offsets 直接计算 token 数 (performance): 未采纳，文件移除后该优化不再适用。
- 是否同时处理 video/audio placeholder (question): 暂不处理，保留扩展点。

风险与影响

- 风险：
  1. 核心路径变更：修改了所有 VLM 的 process_and_combine_mm_data，若新逻辑有误可能影响所有多模态请求。但通过 try-except 回退机制降低风险。
  2. 覆盖不全：resolve_image_token_counts 仅对图像生效，视频与音频未处理；SGLANG_MM_AVOID_RETOKENIZE 开关条件中明确排除了音频 / 视频，若后续扩展需同步修改条件。
  3. 测试覆盖单一：仅测试 Qwen2.5-VL，未覆盖 Kimi 或其他模型，Kimi 的覆盖方法 media_tokens_calculator 的正确性依赖外部代码。
  4. 潜在性能损耗：对预 token 化请求新增了一次 resolve_image_token_counts 调用（通常图片数 < 10），影响可忽略。
  5. 环境变量默认开启：可能改变依赖于重新 tokenize 行为（如特殊预处理）的用户，但原则上应无此类依赖。 - 影响：对用户：RL 训练用户将获得 token 一致性保障，避免因漂移导致的训练失败；普通 VLM 用户无感知（默认行为兼容性良好，失败时自动 fallback）。对系统：增加约 100 行核心逻辑，环境变量控制开关，无新依赖。对团队：提供可扩展的架构（resolve_image_token_counts 可供其他模型覆盖），为后续 video/audio 支持奠定基础。
- 风险标记：核心路径变更，测试覆盖单一模型，默认开启可能影响隐式依赖

关联脉络

- 暂无明显关联 PR