PR #7038 完整报告

verl-project/verl

[rollout, vllm] fix: stop the policy from sampling vision placeholder tokens

合并时间：2026-07-14 16:30

原文链接：http://prhub.com.cn/verl-project/verl/pull/7038

执行摘要

- 一句话：在 vLLM 引擎 logits 中屏蔽视觉占位符 token，防止训练崩溃
- 推荐动作：该 PR 的修复方案（engine 层 logits mask 取代 per-request bad_words）是处理此类问题的正确模式，值得相关后端开发人员深入理解。讨论中涉及的安全性权衡（setdefault 漏洞、MTP drafter 覆盖、兼容性）也很有参考价值。建议合并并尽快推动其余后端的适配。

功能与动机

多模态模型（如 Qwen3-VL）在 GRPO 训练中，policy 可能无意中采样到 <image_padl> 或 <lvideo_padl> 占位符 token，而这些 token 必须与实际多模态输入一一对应。未对齐的占位符会导致下游 get_rope_index（StopIteration）或 merge_multimodal_embeddings（masked_scatter_ shape mismatch）崩溃，训练数小时后非确定性地失败（issue #5817）。PR 作者通过一次 8 小时 Qwen3-VL-8B GRPO 运行在 step 44 复现了该问题。

实现拆解

1. 提取占位符 token ID：在 verl/workers/rollout/utils.py 中新增 get_vision_placeholder_token_ids(processor) 函数，利用 get_processor_token_id 从 processor 的 image_token_id/video_token_id（或对应的字符串 token）解析出需屏蔽的 token ID 列表；纯文本模型返回空列表。
2. 扩展 logits 屏蔽函数：在 verl/workers/rollout/vllm_rollout/utils.py 中修改 monkey_patch_compute_logits，新增 banned_token_ids 参数；闭包内部的 compute_logits 在已有的 OOV mask（logits[..., vocab_size:] = -inf）之外，对 banned_token_ids 指定的索引也设为 -inf。同时更新 monkey_patch_model 以透传该参数。
3. 引擎初始化时注入屏蔽：在 verl/workers/rollout/vllm_rollout/vllm_async_server.py 的 run_server 中，调用 collective_rpc(method='monkey_patch_model', kwargs={...}) 时新增 banned_token_ids 字段，其值来自 get_vision_placeholder_token_ids(self.model_config.processor)。至此所有请求（含 MTP drafter）在 engine 内部即被保护。
4. 新增单元测试：新建 tests/workers/rollout/test_vision_placeholder_tokens_on_cpu.py，包含两类测试：TestGetVisionPlaceholderTokenIds（验证 ID 解析正确性：来自 int ID、字符串、仅图像、纯文本等）；TestMonkeyPatchComputeLogits（验证 logits mask 生效：禁止 token 和 OOV 尾部均被屏蔽为 -inf，不传 banned_token_ids 时行为不变）。所有测试无需模型下载，在 CPU 中运行。

关键文件：

- verl/workers/rollout/vllm_rollout/vllm_async_server.py（模块 vLLM 引擎；类别 source；类型 dependency-wiring）：引擎初始化入口：在 collective_rpc 中传递 banned_token_ids，使得所有请求在 engine 内部即被保护。
- verl/workers/rollout/vllm_rollout/utils.py（模块 vLLM 工具；类别 source；类型 core-logic；符号 monkey_patch_compute_logits, monkey_patch_model）：核心补丁：monkey_patch_compute_logits 新增 banned_token_ids 参数，将占位符 token 与 OOV 一并 mask；monkey_patch_model 透传该参数。
- verl/workers/rollout/utils.py（模块 Rollout 工具；类别 source；类型 core-logic；符号 get_vision_placeholder_token_ids）：新增 get_vision_placeholder_token_ids 函数，从 processor 中提取 image/video 占位符 token ID，供 engine 和测试使用。
- tests/workers/rollout/test_vision_placeholder_tokens_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 make_processor, TestGetVisionPlaceholderTokenIds, test_resolves_both_placeholders_from_token_ids, test_resolves_placeholders_from_token_strings）：新增单元测试覆盖核心函数：ID 解析（四种场景）和 logits 屏蔽（三种场景），确保功能正确且无回归。

关键符号：get_vision_placeholder_token_ids, monkey_patch_compute_logits, monkey_patch_model, make_processor, compute_logits (FakeModel)

关键源码片段

verl/workers/rollout/vllm_rollout/utils.py

核心补丁：monkey_patch_compute_logits 新增 banned_token_ids 参数，将占位符 token 与 OOV 一并 mask；monkey_patch_model 透传该参数。

```
def monkey_patch_compute_logits(model, vocab_size: int, banned_token_ids: Optional[list[int]] = None):
    """Mask the tokens the sampler must never pick.

    Beyond the out-of-vocabulary tail, `banned_token_ids` covers tokens that live *inside* the
    vocabulary yet are still illegal to generate: the vision placeholders, which are meaningless
    unless a real image or video sits behind them. See `get_vision_placeholder_token_ids`.
    """
    original_compute_logits = model.compute_logits

    def compute_logits(self, *args, **kwargs) -> torch.Tensor:
        logits = original_compute_logits(*args, **kwargs)
        # 屏蔽 OOV 尾部（已存在的逻辑）
        logits[..., vocab_size:] = float("-inf")
        # 屏蔽禁止采样的 token（如视觉占位符）
        if banned_token_ids:
            logits[..., banned_token_ids] = float("-inf")
        return logits

    model.compute_logits = MethodType(compute_logits, model)
```

```
class vLLMColocateWorkerExtension:
    # ...
    def monkey_patch_model(self, vocab_size: int, banned_token_ids: Optional[list[int]] = None):
        for model in self._iter_all_models():
            # 同时屏蔽 OOV 和禁止 token
            monkey_patch_compute_logits(model, vocab_size, banned_token_ids)
            patch_vllm_moe_model_weight_loader(model)
```

tests/workers/rollout/test_vision_placeholder_tokens_on_cpu.py

新增单元测试覆盖核心函数：ID 解析（四种场景）和 logits 屏蔽（三种场景），确保功能正确且无回归。

```
# 测试辅助：构造假 processor（简化版）
def make_processor(**attrs):
    tokenizer = SimpleNamespace(convert_ids_to_tokens=TOKENS.get)
    return SimpleNamespace(tokenizer=tokenizer, **attrs)


class TestGetVisionPlaceholderTokenIds:
    """验证 get_vision_placeholder_token_ids 能正确解析各种 processor 配置。"""

    def test_resolves_both_placeholders_from_token_ids(self):
        # 标准情况：processor 提供 image_token_id 和 video_token_id
        processor = make_processor(image_token_id=151655, video_token_id=151656)
        assert get_vision_placeholder_token_ids(processor) == [151655, 151656]

    def test_resolves_placeholders_from_token_strings(self):
        # 新式 processor：用字符串 token 代替 ID
        processor = make_processor(image_token="<|image_pad|>", video_token="<|video_pad|>")
        processor.tokenizer.convert_tokens_to_ids = {v: k for k, v in TOKENS.items()}.get
        assert get_vision_placeholder_token_ids(processor) == [151655, 151656]

    def test_text_only_model_leaves_sampling_untouched(self):
        # 纯文本模型：processor 为 None，应返回空列表
        assert get_vision_placeholder_token_ids(None) == []


class TestMonkeyPatchComputeLogits:
    """验证 monkey_patch_compute_logits 正确屏蔽禁止 token。"""

    def test_banned_tokens_and_oov_tail_are_masked(self):
        model = FakeModel() # 固定 logits 为 [[0,1,2,3,4,5]]*2, vocab=6, pad=8
        monkey_patch_compute_logits(model, VOCAB_SIZE, banned_token_ids=[2, 4])
        logits = model.compute_logits()
        # 禁止 token 位置为 -inf
        assert (logits[:, [2, 4]] == float("-inf")).all()
        # OOV 尾部（>=6）为 -inf
        assert (logits[:, VOCAB_SIZE:] == float("-inf")).all()
        # 合法 token 保持不变
```

```
assert torch.equal(logits[:, [0, 1, 3, 5]], UNMASKED[:, [0, 1, 3, 5]])
```

评论区精华

核心讨论围绕屏蔽方案的安全性展开：

- bad_words 方案缺陷：最初 PR 尝试在 sampling_params 中通过 setdefault 添加 bad_words，但 gemini-code-assist[bot] 指出若用户已传入 bad_words，setdefault 不会覆盖，导致保护被静默绕过。
- 采纳 compute_logits 屏蔽：维护者 wuxibin89 建议参考 release/v0.6.1 中在 compute_logits 直接 mask 的做法。作者 zhshj0110 采纳并指出该方案更强：无法被 per-request 参数绕过、覆盖 MTP drafter、且与已有的 OOV mask 逻辑一致。
- agent_loop fallback 移除：随 engine 层屏蔽生效，代理循环中的 _drop_ungrounded_vision_tokens 保护不再触发 vLLM 路径，因此删除相关代码及测试。
- 测试文件合并：wuxibin89 建议将两个测试文件合并，zhshj0110 在最终 commit 中合并为一个文件，涵盖 ID 解析和 mask 验证。
  - bad_words setdefault 安全漏洞 (security): 作者接受建议，后续改用 engine 层 logits mask 方案，完全移除 bad_words 路径。
  - 改用 compute_logits 屏蔽替代 bad_words (design): 采用 compute_logits mask 方案，agent_loop 中的 fallback 因不再需要而删除。
  - 测试文件合并建议 (testing): 作者在最终 commit 中合并为 test_vision_placeholder_tokens_on_cpu.py，覆盖所有用例。

风险与影响

- 风险：
  1. 仅覆盖 vLLM 后端：PR 仅针对 vLLM（vllm_async_server.py 中的 engine 屏蔽），sglang（不使用 compute_logits patch）和 trtllm（无等效机制）仍可能采样占位符 token 而崩溃，需后续同步修复。
  2. 托管 token ID 解析失败：get_processor_token_id 若对特定 processor 返回 None，占位符将不被屏蔽；但纯文本模型不受影响，多模态模型若解析失败则是更早的问题。
  3. 重要性比率轻微偏差：Masking 两个 logit 值会使 rollout 分布与 actor 略有差异，但重要性比率仍基于原始 logits（mask 仅作用于 compute_logits，重要性比率使用另一份 logits），实际无影响。具体见 PR body 讨论。
  4. 误屏蔽危险性低：占位符 token ID 是特定的（如 Qwen3-VL 的 151655/151656），正常 token 不会被波及。 - 影响：对用户：使用 vLLM 进行多模态 RL 训练的用户不会再遇到因占位符 token 导致的非确定性崩溃，无需额外配置。纯文本模型不受影响。对系统：改动集中在 engine 初始化阶段，对推理性能无影响；不引入新配置项。对团队：需为 sglang、trtllm 及其他可能的 rollout 后端实现等效屏蔽，以保证多模态训练的稳定性。
- 风险标记：仅修复 vLLM 后端，依赖 get_processor_token_id 正确性，潜在误屏蔽（极低）

关联脉络

- PR #5817 [Bug] Full Async vLLM: Model generates video_token_id during RL rollout, causing vLLM MaskedScatter AICPU crash: 此 issue 报告了相同问题的崩溃现象（MaskedScatter 形状不匹配），PR 中的 fix 直接解决了该 issue 描述的根本原因。