

# PR #54196 完整报告

vllm-project/vllm

[Bugfix][Frontend] Validate stop\_token\_ids against vocab size

合并时间: 2026-08-30 00:25

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/54196>

## 执行摘要

- 一句话: 为 stop\_token\_ids 补词表校验, 杜绝越界击穿引擎
- 推荐动作: 值得精读。虽然代码改动只有单文件 46 行, 但 review 过程质量很高: 关于 len(tokenizer) 与 model\_config.get\_vocab\_size() 差异的论证 (多模态占位 token、模型侧新增 token、gap id 越界路径)、max 方案的过度修正与回退、以及逐字段审计“谁消费索引、张量尺寸由谁决定”的方法论, 对理解 vLLM 的词表边界约定和前端参数校验架构都很有借鉴价值。关注点: 无自动化测试的取舍是否符合团队长期预期, 以及 bad\_words 字段的边界问题是否会作为后续跟进。

## 功能与动机

PR body 与关联 issue #54195 明确说明: stop\_token\_ids 是唯一未做词表校验的用户可见 token-id 参数, 越界 id 会经 MinTokensLogitsProcessor 的 logits.index\_put\_((rows, all\_stop\_token\_ids), -inf) 落到设备端索引路径。在 Ascend CANN 9.1 上单请求即可触发 IndexCheck 内核断言, 导致全部 8 个 TP rank 同时死亡、预填充实例对所有后续请求返回 HTTP 500; 在无边界检查的后端则静默污染同 batch 相邻请求的 logits。151645 是旧版 DeepSeek V3 tokenizer 的 <lim\_endl>, 客户端复用陈旧 token id 即可轻易触发, 且 PD 代理示例脚本会把请求改写为 max\_tokens=1, min\_tokens=1, 使每个代理请求都激活该路径。logit\_bias 与 allowed\_token\_ids 已有校验, stop\_token\_ids 是唯一漏网入口。

## 实现拆解

本 PR 围绕 vllm/sampling\_params.py 单文件展开, 分四步完成:

1. 在 verify() 中接入新校验: 在 SamplingParams.verify() 的 \_validate\_logit\_bias 之后、\_validate\_trace\_replay 之前插入 self.\_validate\_stop\_token\_ids(model\_config) 调用。位置选择与 \_validate\_logit\_bias 紧邻, 因为二者索引的是同一张 logits 张量。同时把 \_validate\_allowed\_token\_ids 的调用参数从 tokenizer 换成 model\_config。
2. 新增 \_validate\_stop\_token\_ids 方法: 完全镜像 \_validate\_logit\_bias 的既有模式——stop\_token\_ids 为空即返回; 否则收集所有 < 0 或 >= model\_config.get\_vocab\_size() 的非法 id, 命中即抛 VLLMValidationError (parameter="stop\_token\_ids"), 该异常在前端转换为 HTTP 400, 请求不会进入 engine。关键设计决策是边界取模型词表而非 len(tokenizer), 因为 stop\_token\_ids 是 logits 张量的列索引, 而 logits 宽度由模型词表决定。

3. 修正 `_validate_allowed_token_ids` 的边界来源：签名从 `tokenizer: TokenizerLike | None` 改为 `model_config: ModelConfig`，移除 `tokenizer is None` 的早退分支（保证 `skip_tokenizer_init` 场景下仍能校验），并把 `len(tokenizer)` 换成 `model_config.get_vocab_size()`。原因经 review 确认：`allowed_token_ids` 同样由客户端直接以 `list[int]` 传入，且设备端掩码张量（`InputBatch` 中的 `mask[req_index][allowed_token_ids]`）按模型词表尺寸分配。作者还验证了内部调用者不受影响：`beam search` 的 `id` 来自按模型词表尺寸的 `bitmask`，`Whisper` 语言检测 `id` 是模型词表内的特殊 `token`。
4. 测试与配套取舍：最初新增 `tests/entrypoints/openai/chat_completion/test_chat_stop_token_ids_validation.py`（沿用 `test_chat_logit_bias_validation.py` 模式，覆盖合法 `id` 通过、越界与负 `id` 返回 400），但按 `DarkLight1337`“CI 昂贵、不值得加测试”的意见在提交 `d0ec419` 中移除。作者改为离线对 `Qwen/Qwen2.5-1.5B-Instruct` 验证（`[vocab_size-1, 0]` 通过，`[vocab_size, vocab_size+5]` 与 `[-2]` 报错），并在复现环境端到端确认崩溃请求现在返回 400 且引擎存活。最终合入版本为纯源码单文件变更，无测试、配置或部署配套改动。

关键文件：

- `vllm/sampling_params.py`（模块 采样参数；类别 `source`；类型 `core-logic`；符号 `_validate_stop_token_ids`, `_validate_allowed_token_ids`, `verify`）：核心修复文件。新增 `_validate_stop_token_ids` 并按 `model_config.get_vocab_size()` 校验 `stop_token_ids`，同时将 `_validate_allowed_token_ids` 的边界来源从 `len(tokenizer)` 统一为模型词表，修复跨后端的越界索引崩溃 / 数据污染问题。

关键符号：`_validate_stop_token_ids`, `_validate_allowed_token_ids`, `verify`

## 关键源码片段

### `vllm/sampling_params.py`

核心修复文件。新增 `_validate_stop_token_ids` 并按 `model_config.get_vocab_size()` 校验 `stop_token_ids`，同时将 `_validate_allowed_token_ids` 的边界来源从 `len(tokenizer)` 统一为模型词表，修复跨后端的越界索引崩溃 / 数据污染问题。

```
def verify(
    self,
    model_config: ModelConfig,
    speculative_config: SpeculativeConfig | None,
    structured_outputs_config: StructuredOutputsConfig | None,
    tokenizer: TokenizerLike | None,
) -> None:
    self._validate_logprobs(model_config)
    self._validate_logit_bias(model_config)
    self._validate_trace_replay(model_config, speculative_config)
    # stop_token_ids 是此前唯一未校验词表边界的客户端直传 token-id 参数,
    # 越界 id 会进入 all_stop_token_ids 并在设备端做 logits.index_put_,
    # 必须在到达 engine 前以 400 拒绝
    self._validate_stop_token_ids(model_config)
```

```

self._validate_logits_processors(model_config)
# allowed_token_ids 同为客户端直传 list[int], 索引 model vocab 宽度的
# 掩码张量, 故与 stop_token_ids 统一改用 model_config 校验
self._validate_allowed_token_ids(model_config)
self._validate_spec_decode(speculative_config)
self._validate_diffusion(model_config)
self._validate_structured_outputs(
    model_config, structured_outputs_config, tokenizer
)

```

```

def _validate_stop_token_ids(self, model_config: ModelConfig) -> None:
    """Validate stop_token_ids are within vocabulary range."""
    if not self.stop_token_ids:
        return

    # stop_token_ids 会被用作 logits 张量的列索引 (如 MinTokensLogitsProcessor
    # 的 logits.index_put_((rows, all_stop_token_ids), -inf)) , 而 logits 宽度
    # 由 model_config.get_vocab_size() 决定; 不能用 len(tokenizer) 作边界,
    # 因为多模态模型的 tokenizer 可能比模型词表更大, gap id 会漏过校验后
    # 依旧越界 (CANN 9.1 上直接触发 IndexCheck 内核崩溃)
    vocab_size = model_config.get_vocab_size()
    invalid_token_ids = [
        token_id
        for token_id in self.stop_token_ids
        if token_id < 0 or token_id >= vocab_size
    ]

    if invalid_token_ids:
        raise VLLMValidationError(
            f"token_id(s) {invalid_token_ids} in stop_token_ids contain "
            f"out-of-vocab token ids. Vocabulary size: {vocab_size}",
            parameter="stop_token_ids",
            value=invalid_token_ids,
        )

def _validate_allowed_token_ids(self, model_config: ModelConfig) -> None:
    """按模型词表校验 allowed_token_ids, 与 stop_token_ids 保持一致。"""
    allowed_token_ids = self.allowed_token_ids
    if allowed_token_ids is None:
        return

    if len(allowed_token_ids) == 0:
        raise VLLMValidationError(
            "allowed_token_ids is not None and empty!",
            parameter="allowed_token_ids",
            value=allowed_token_ids,
        )

```

```

# allowed_token_ids 由客户端直接以 list[int] 传入，随后作为列索引写入
# InputBatch 中按 model_config.get_vocab_size() 尺寸分配的掩码张量
# (mask[req_index][allowed_token_ids])，因此同样必须按模型词表校验；
# 与旧版 len(tokenizer) 相比，这一改动同时保证了 skip_tokenizer_init
# 场景下校验仍然生效
vocab_size = model_config.get_vocab_size()
invalid_token_ids = [
    token_id
    for token_id in allowed_token_ids
    if token_id < 0 or token_id >= vocab_size
]
if invalid_token_ids:
    raise VLLMValidationError(
        "allowed_token_ids contains out-of-vocab token id!",
        parameter="allowed_token_ids",
        value=invalid_token_ids,
    )

```

## 评论区精华

review 的核心交锋围绕“校验边界该取 tokenizer 还是模型词表”展开，经历了三轮迭代：

- depthfirst-app[bot] 首先指出 len(tokenizer) 不是安全边界：stop\_token\_ids 索引的 logits 张量由 model\_config.get\_vocab\_size() 决定，二者在多模态模型上可能不一致，建议改用模型词表。
- QwertyJack 曾尝试取 max(model\_config.get\_vocab\_size(), len(tokenizer))，理由是避免误伤 Qwen3 这类 tokenizer 比模型词表小的模型上“模型侧新增 token”，但被 depthfirst-app[bot] 再次否决：该方案在多模态模型（tokenizer 更大）的 gap 区间仍会越界。
- 作者最终承认“I over-corrected in de34963”，回退到纯 model\_config.get\_vocab\_size() 边界，并指出该取舍的代价是 tokenizer 可编码但模型不可采样的 stop\_token\_id 会被拒绝——这正是保守且正确的行为。
- DarkLight1337 在追问中发现了更大的漏洞：\_validate\_allowed\_token\_ids 沿用 len(tokenizer) 且同样由客户端直传 list[int]，最终拍板“Let's update allowed\_token\_ids then”，使本 PR 从单字段修复扩展为两个字段的边界统一。
- 测试文件方面，DarkLight1337 明确表示“our CI is expensive enough as it is, let's just remove them”，作者接受并移除。
- stop\_token\_ids 校验边界：len(tokenizer) 还是 model\_config.get\_vocab\_size() (correctness): 采用 model\_config.get\_vocab\_size()，与 \_validate\_logit\_bias 一致（索引同一张 logits 张量）。
- max(vocab\_size, len(tokenizer)) 方案的过度修正与回退 (correctness): 作者承认 over-corrected，回退为纯 model\_config.get\_vocab\_size() 边界，并记录取舍：tokenizer 可编码但模型不可采样的 id 被拒绝是正确行为。
- allowed\_token\_ids 是否同步改用模型词表边界 (correctness): 将 \_validate\_allowed\_token\_ids 签名从 tokenizer 改为 model\_config，移除 None-guard,

保证 skip\_tokenizer\_init 场景仍校验；bad\_words 因不可自由构造保持现状。

- 新增测试文件是否值得保留 (testing): 测试文件被移除，合入版本无自动化测试覆盖。

## 风险与影响

- 风险:

1. 行为收窄的兼容性风险：对 tokenizer 词表大于模型词表的多模态模型，位于 gap 区间的 stop\_token\_id 此前可通过 (tokenizer 边界) 校验，现在会被 400 拒绝。作者论证该 id 永远不会被采样命中、只会走越界路径，因此是正确取舍，但对依赖此类 id 的存量客户端属于破坏性变更，升级时需留意。
2. 同类风险残留：review 审计发现 bad\_words 仍用 tokenizer.max\_token\_id 做边界，但作者论证其 id 由 tokenize 产生、不可自由构造，风险不可达故未处理。未来若开放客户端直传 bad words ids 的 API，需要按本次约定补校验。
3. 无自动化测试覆盖：最终合入版本不含任何测试，该防线完全依赖 SamplingParams.verify() 被正确调用。若未来出现绕过 verify() 的内部构造路径（如直接实例化 SamplingParams 的代码），回归风险上升。
4. 核心请求路径变更：verify() 是前端所有请求的必经路径，新增校验为纯 Python 列表扫描，复杂度  $O(n)$ ，且 model\_config.get\_vocab\_size() 已有缓存，无性能影响。
  - 影响：对用户：所有通过 OpenAI 兼容 API 传 stop\_token\_ids 的用户行为发生变化——越界 id 从“可能崩溃整个引擎”或“静默污染相邻请求”变为确定性的 HTTP 400 拒绝；对 Ascend CANN 9.1 后端用户是直接的单请求 DoS 修复，对 CUDA 等无边界检查后端则是把不确定行为变为确定行为。对系统：消除了 TP 全 rank 同时死亡、预填充实例对后续所有请求返回 500 的故障模式，PD 代理改写 min\_tokens=1 后的每请求崩溃风险也被消除。对团队：确立了“设备端索引型 token-id 参数必须按 model\_config 词表校验”的工程约定，并顺带修正了 allowed\_token\_ids 的同方向漏洞，为后续同类字段审计提供了模板。
  - 风险标记：单请求 DoS 高危修复，无自动化测试覆盖，核心请求路径变更，行为边界收窄 (gap id 被拒)

## 关联脉络

- PR #54509 [Bugfix][Frontend] Truncate prompt\_is\_token\_ids with the prompt: 同为前端请求参数边界修复（截断 prompt 时同步截断掩码），与本次 stop\_token\_ids 校验同属 vLLM 近期持续补漏的“前端参数硬化”系列。
- PR #54407 [Bugfix][Frontend] Truncate prompt\_token\_offsets with the prompt: 修复 prompt\_token\_offsets 未随截断同步导致的并行数组长度漂移，同一批前端参数一致性 bugfix，反映团队正在系统性地收敛请求参数边界漏洞。
- PR #54364 [Bugfix][Frontend] Truncate pooling prompts before padding them: 修复 pooling 先补位后截断导致 prompt 全为 pad，同为请求参数处理顺序 / 边界问题，属于同一功能线的演进。