PR #49227 完整报告

vllm-project/vllm

[Bugfix][Structured Output] Mask request stop tokens in xgrammar until grammar terminates

合并时间：2026-08-11 02:29

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/49227

执行摘要

- 一句话：xgrammar 语法未终止前屏蔽请求级 stop token
- 推荐动作：值得精读。改动紧凑（7 个文件约 120 行）但切中一个真实的高频数据正确性问题，测试设计精巧——用 gpt2 的固定 token id 精确驱动 grammar 状态机，验证 stop token 在非终止 / 终止两种状态下的 bitmask 差异。值得学习的点：请求级 stop 集合（generation_config eos 列表 + 用户 stop_token_ids）与 tokenizer 默认 eos 的差异如何漏进采样掩码；xgrammar override_stop_tokens 的对接方式；以及抽象接口签名变更多后端同步的工程规范。建议跟进 guidance 后端遗留的同类问题。

功能与动机

Issue #42403 报告：Gemma 4 在 structured_outputs（JSON schema）下，模型偶尔在 grammar FSM 仍处于 JSON 字符串内部时采样 <end_of_turn>（token id 106），直接中断生成，输出为未闭合 JSON；在 NVFP4 量化的 Gemma 4 31B 上，10-20 个样本中约 20%-40% 概率复现。PR body 给出根因定位：xgrammar 只知道 tokenizer 的单一 eos（Gemma 4 为 1），而请求真实 stop 集合是 generation_config 的 eos 列表 [1, 106, 50] 加用户 stop_token_ids，106 和 50 对 grammar 不可见，因此能在非终止状态下逃逸 bitmask，把生成截断成非法 JSON。

实现拆解

第 1 步：契约层——抽象接口新增 stop_token_ids 参数

vllm/v1/structured_output/backend_types.py 的抽象方法 StructuredOutputBackend.compile_grammar 新增可选参数 stop_token_ids: set[int] | None = None，并在 docstring 说明语义（请求的 EOS 与用户 stop token 集合，来自 SamplingParams.all_stop_token_ids）。随后同步更新另外三个后端 backend_guidance.py、backend_lm_format_enforcer.py、backend_outlines.py 的 compile_grammar 签名，仅保持接口一致、不改行为，避免多态调用时签名不匹配。

第 2 步：入口层——从采样参数提取请求级 stop 集合

vllm/v1/structured_output/__init__.py 的 _create_grammar 在编译语法前读取 request.sampling_params.all_stop_token_ids（为空时传 None），随 request_type 和 grammar_spec 一起传给 compile_grammar。这一步把 " 请求的真实 stop 集合 " 从采样路径带入语法编译路径。

第 3 步：核心修复——xgrammar matcher 门控 stop token

vllm/v1/structured_output/backend_xgrammar.py 的 XgrammarBackend.compile_grammar 在构造 xgr.GrammarMatcher 时新增 override_stop_tokens=list(stop_token_ids) if stop_token_ids else None。xgrammar 据此在语法未达终止状态时屏蔽这些 token，语法完整后才允许其被采样。该参数是 matcher 级配置，不影响 GrammarCompiler 的 grammar 缓存。

第 4 步：回归测试与验证

新增 tests/v1/structured_output/test_backend_xgrammar_stop_tokens.py（82 行），用 gpt2 驱动 {"type": "string"} 语法：在字符串中间断言普通 token 可采样而 stop token 被屏蔽，闭合字符串后断言 stop token 恢复可采样、且 tokenizer 默认 eos 行为不变。PR body 还给出端到端复现：logit_bias={"106": 100} 强制 Gemma 4 采样 <end_of_turn>，修复前输出被截断为 {"name": "，修复后输出完整 JSON 且 stop_reason=106。

关键文件：

- vllm/v1/structured_output/backend_xgrammar.py（模块 结构化输出；类别 source；类型 core-logic；符号 compile_grammar）：核心修复所在：compile_grammar 新增 stop_token_ids 参数并透传给 xgr.GrammarMatcher 的 override_stop_tokens，是 stop token 门控到语法终止状态的关键实现。
- vllm/v1/structured_output/__init__.py（模块 结构化输出；类别 source；类型 core-logic；符号 _create_grammar）：语法编译入口：_create_grammar 从 request.sampling_params.all_stop_token_ids 提取请求真实 stop 集合并传入后端，是数据链路打通的一步。
- tests/v1/structured_output/test_backend_xgrammar_stop_tokens.py（模块 结构化输出；类别 test；类型 test-coverage；符号 _token_allowed, backend, test_request_stop_tokens_gated_to_grammar_terminal）：新增回归测试（issue #42403），用 gpt2 固定 token id 精确驱动 grammar 状态机，验证 stop token 在非终止态被屏蔽、终止态放行，且 tokenizer 默认 eos 行为不变。
- vllm/v1/structured_output/backend_types.py（模块 结构化输出；类别 source；类型 interface-contract；符号 compile_grammar）：抽象接口 compile_grammar 增加 stop_token_ids 参数并补充 docstring，是所有后端签名同步的契约来源。
- vllm/v1/structured_output/backend_guidance.py（模块 结构化输出；类别 source；类型 signature-sync；符号 compile_grammar）：signature 同步：guidance 后端 compile_grammar 接受 stop_token_ids 但未实现屏蔽逻辑，yzong-rh 指出需单独修复，是遗留缺口记录点。
- vllm/v1/structured_output/backend_lm_format_enforcer.py（模块 结构化输出；类别 source；类型 signature-sync；符号 compile_grammar）：签名同步：lmfe 后端 compile_grammar 接受 stop_token_ids 参数保持接口一致，行为未变。
- vllm/v1/structured_output/backend_outlines.py（模块 结构化输出；类别 source；类型 signature-sync；符号 compile_grammar）：签名同步：outlines 后端 compile_grammar 接受 stop_token_ids 参数保持接口一致，行为未变。

关键符号：compile_grammar, _create_grammar, test_request_stop_tokens_gated_to_grammar_terminal, _token_allowed

关键源码片段

vllm/v1/structured_output/backend_xgrammar.py

核心修复所在：compile_grammar 新增 stop_token_ids 参数并透传给 xgr.GrammarMatcher 的 override_stop_tokens，是 stop token 门控到语法终止状态的关键实现。

```
# vllm/v1/structured_output/backend_xgrammar.py
# compile_grammar 在原有语法编译逻辑不变的前提下，
# 新增 stop_token_ids 参数并把它透传给底层 matcher。

def compile_grammar(
    self,
    request_type: StructuredOutputOptions,
    grammar_spec: str,
    stop_token_ids: set[int] | None = None, # 请求级 stop 集合（eos 列表 + 用户 stop_token_ids）
) -> StructuredOutputGrammar:
    # 按请求类型编译 grammar，JSON / GRAMMAR / REGEX / STRUCTURAL_TAG 分支从略，
    # 这里以 JSON schema 为例。
    ctx = self.compiler.compile_json_schema(
        grammar_spec, any_whitespace=not self.disable_any_whitespace
    )

    # 关键修复：override_stop_tokens 让 xgrammar 在语法未终止时屏蔽这些 token，
    # 只有 grammar 到达终止状态后才允许它们被采样，从而避免 mid-object 截断。
    return XgrammarGrammar(
        matcher=xgr.GrammarMatcher(
            ctx,
            override_stop_tokens=list(stop_token_ids) if stop_token_ids else None,
            max_rollback_tokens=self.num_speculative_tokens,
        ),
        vocab_size=self.vocab_size,
        ctx=ctx,
    )
```

vllm/v1/structured_output/__init__.py

语法编译入口：_create_grammar 从 request.sampling_params.all_stop_token_ids 提取请求真实 stop 集合并传入后端，是数据链路打通的一步。

```
# vllm/v1/structured_output/__init__.py
# 语法编译入口：把请求的真实 stop 集合传给后端。

def _create_grammar(self, request: "Request") -> StructuredOutputGrammar:
    struct_request = request.structured_output_request
    assert struct_request is not None
    try:
```

```
            request_type, grammar_spec = struct_request.structured_output_key
            assert self.backend is not None
            # generation_config 的 eos 列表与用户 stop_token_ids 都汇总在
            # all_stop_token_ids 中；xgrammar 默认只认识 tokenizer 的单一 eos，
            # 必须显式透传，否则 106/50 这类额外 stop token 会逃逸 bitmask。
            stop_token_ids = (
                request.sampling_params.all_stop_token_ids
                if request.sampling_params is not None
                else None
            )
            return self.backend.compile_grammar(
                request_type, grammar_spec, stop_token_ids=stop_token_ids
            )
        except Exception:
            logger.exception(
                "Failed to compile grammar for request %s", request.request_id
            )
            raise
```

tests/v1/structured_output/test_backend_xgrammar_stop_tokens.py

新增回归测试（issue #42403），用 gpt2 固定 token id 精确驱动 grammar 状态机，验证 stop token 在非终止态被屏蔽、终止态放行，且 tokenizer 默认 eos 行为不变。

```
# tests/v1/structured_output/test_backend_xgrammar_stop_tokens.py
# 回归测试（issue #42403）：验证 stop token 被门控到语法终止状态。

def test_request_stop_tokens_gated_to_grammar_terminal(backend: XgrammarBackend):
    schema = '{"type": "string"}'
    default = backend.compile_grammar(StructuredOutputOptions.JSON, schema)
    override = backend.compile_grammar(
        StructuredOutputOptions.JSON, schema, stop_token_ids={EOS, LETTER}
    )

    # 打开引号：两个 grammar 都进入非终止状态（字符串内部）。
    for grammar in (default, override):
        assert grammar.accept_tokens("req", [QUOTE])

    bm_default = backend.allocate_token_bitmask(1)
    bm_override = backend.allocate_token_bitmask(1)
    default.fill_bitmask(bm_default, 0)
    override.fill_bitmask(bm_override, 0)

    # 字符串中间：默认 grammar 下普通 token 仍可采样（泄漏点）；
    # 把它注册为 stop token 后，override grammar 将其屏蔽。
    assert _token_allowed(bm_default[0], LETTER)
    assert not _token_allowed(bm_override[0], LETTER)

    # 闭合引号 -> 语法完整但尚未终止。
    for grammar in (default, override):
```

```
        assert grammar.accept_tokens("req", [QUOTE])
        assert not grammar.is_terminated()

    default.fill_bitmask(bm_default, 0)
    override.fill_bitmask(bm_override, 0)

    # 额外 stop token 现在只允许在 override grammar 下终止；
    # tokenizer 默认 eos 在两种 grammar 下仍可终止，默认行为被保留。
    assert not _token_allowed(bm_default[0], LETTER)
    assert _token_allowed(bm_override[0], LETTER)
    assert _token_allowed(bm_default[0], EOS)
    assert _token_allowed(bm_override[0], EOS)
```

评论区精华

三位 reviewer 的交互中，最有价值的是 yzong-rh 的批准意见："The issue occur in guidance as well, but it doesn't seem mask out stop tokens even if we override eos_token in llguidance_hf.from_tokenizer. Separate fix required."（同类问题在 guidance 后端也存在，即使 override eos_token 也不会屏蔽 stop token，需要单独修复）。这确认了本修复的正确性，同时暴露了多后端覆盖不一致的遗留缺口。bbrowning 本地拉取验证："Pulled locally, reviewed, and confirmed the new test properly catches this regression."。另外 mergify 曾提示存在合并冲突需 rebase，作者 rebase 后触发 Buildkite CI（#83175）。

- guidance 后端存在同源 stop token 泄漏 (design): 本 PR 仅修复 xgrammar 后端，guidance 后端问题记录为后续单独修复项。
- 回归测试有效性确认 (testing): 测试被两位 reviewer 验证有效，可防止该 bug 回归。
- 合并冲突与 CI 重跑 (other): rebase 完成，CI 已触发并通过。

风险与影响

- 风险：
  1. 行为变化：修复前能从非终止状态采样到的 stop token 现在被屏蔽，对依赖旧行为的用户（极少数把 stop token 当普通 token 用的场景）有感知，但这正是修复目标。
  2. 多后端不一致：guidance 后端只改了签名、未实现屏蔽逻辑，同一请求在 xgrammar 与 guidance 后端下行为不同，需后续单独修复（yzong-rh 已确认）。
  3. 接口契约变更：compile_grammar 是所有结构化输出后端的公共抽象方法，签名变更会影响仓库外的自定义后端实现（TypeError），仓库内四个后端已同步。
  4. 终止风险：若 stop token 全被屏蔽且语法因与输入冲突永远无法到达终止态，理论上生成可能无法按时停止；但 override_stop_tokens 仅在非终止状态屏蔽、语法完成即放行，实际风险低。
  5. 性能：每请求多一次属性读取、set 传递与 list 转换，相对 grammar 编译与 bitmask 填充的开销可忽略。 - 影响：影响面集中于 v1 结构化输出路径：所有使用 xgrammar 后端的 JSON / grammar / regex 请求都会把请求级 stop token 纳入 bitmask 门控。对 Gemma 4 等 generation_config 含多个 eos 的模型是直接受益者——结构化输出不再

被 mid-object 的 stop 截断成非法 JSON；对单 eos 模型行为基本不变（默认 eos 原本就在 tokenizer 内）。系统层面，改动发生在 grammar 编译期，matcher 参数不影响 grammar compiler 缓存，采样热路径只增加一次 all_stop_token_ids 读取。团队层面，override_stop_tokens 的用法为其它后端（guidance、outlines、lmfe）修复同类问题提供了参照模板。 - 风险标记：核心采样路径变更，多后端接口同步，guidance 后端缺口遗留，行为变化

关联脉络

- 暂无明显关联 PR