PR #52384 完整报告

vllm-project/vllm

[Rust Frontend][gRPC] Preserve skip_special_tokens decoding option

合并时间：2026-08-15 16:54

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52384

执行摘要

PR #52384 为 Rust gRPC 原生 Generate 端点补上 skip_special_tokens 解码选项：在 ResponseOptions 中新增 optional 布尔字段，并在请求转换时映射到 TextDecodeOptions，使 gRPC 调用方与 Python 各端点（chat completions、completions、responses、generate）保持一致。改动仅 5 行，影响面限定在 rust/proto 与 rust/server，省略字段时行为不变，显式 false 可保留推理 / 工具解析所需的 special tokens。

功能与动机

PR body 指出，Python 侧的 /v1/chat/completions、/v1/completions、/v1/responses 与 /inference/v1/generate 全部默认 skip_special_tokens = true，而 gRPC Generate 请求此前未暴露该选项，调用方无法为 reasoning 与 tool 解析保留 tokenizer 定义的 special markers。本 PR 正是为对齐这些 API 的默认语义而设计，并将字段设为 optional，以便服务端区分“省略”与“显式 false”。

实现拆解

1. 协议扩展（rust/proto/inference.proto）：在 ResponseOptions 消息中新增 optional bool skip_special_tokens = 8，注释标明省略时默认 true。选择 optional 而非普通 bool，是为了保留“未提供”与“显式 false”两种语义，与 Python 默认 true 对齐。早期版本把字段放在 StoppingCriteria，review 后移入 ResponseOptions，因为它属于输出解码行为而非停止条件。
2. 转换映射（rust/src/server/src/grpc/convert.rs）：在 to_text_request 中，TextDecodeOptions 的 skip_special_tokens 从硬编码 true 改为 response.and_then(|options| options.skip_special_tokens).unwrap_or(true)。and_then 消费 Option<ResponseOptions>，再取内部 optional 字段，unwrap_or(true) 保证缺省时沿用 Python 默认值。
3. 测试与验证：作者最初在 StoppingCriteria 中新增字段并附带两个单测（缺省 true / 显式 false 保留），review 后字段移入 ResponseOptions，单测按评审意见移除；最终通过 cargo fmt --check 与既有 grpc::convert::tests 13 个用例。由于 proto 变更无新增直接测试，回归保护依赖转换层既有用例。

rust/src/server/src/grpc/convert.rs

将硬编码的 skip_special_tokens: true 改为从 ResponseOptions 读取并用 unwrap_or(true) 保持 Python 默认值，是功能生效的核心转换逻辑。

```
// to_text_request 内部：将 gRPC 的 ResponseOptions 映射为 v1 引擎的 TextDecodeOptions。
// 关键点：skip_special_tokens 用 optional 字段区分“未提供”和“显式 false”。
let response = req.response.as_ref();

let decode_options = TextDecodeOptions {
    // 与 Python 各端点默认值保持一致：请求未携带该字段时默认跳过 special tokens；
    // 调用方显式传入 Some(false) 时则保留 special tokens（部分 reasoning / tool parser 依赖）。
    skip_special_tokens: response
        .and_then(|options| options.skip_special_tokens)
        .unwrap_or(true),
    include_stop_str_in_output: stopping.is_some_and(|s| s.include_stop_strings),
    stop_strings: stopping.map(|s| &s.stop_strings).filter(|ss| !ss.is_empty()).cloned(),
    min_tokens: stopping.map_or(0, |s| s.min_new_tokens),
};
```

评论区精华

> connorcarpenter15（proto 字段归属）：“This should probably be in ResponseOptions, not StoppingCriteria.”
>
> biswapanda：“good point, fixed it.”
>
> connorcarpenter15（测试取舍）：“I don't think these tests are strictly necessary.”
>
> njhill（合并者）：“Thanks @biswapanda”

风险与影响

- 协议契约变化：ResponseOptions 是公开 gRPC 接口，新增 optional 字段不影响旧客户端（缺省行为不变）；但新客户端显式设置 false 后，输出文本可能包含 special tokens，依赖输出的下游解析逻辑需感知此行为变化。
- 覆盖不足：最终版本没有针对该映射的直接单测，回归保护依赖既有 13 个转换测试。
- 兼容性：Rust 侧 TextDecodeOptions 语义与 Python 保持一致，不会改变采样与 EOS 行为，风险面被 PR body 明确限定在解码文本。

关联脉络

本次改动与 PR #51316（[Rust Frontend][gRPC] Add RL lifecycle control）同属 Rust gRPC 前端功能线，后者扩展了 control.proto 与 grpc 控制面 RPC，本 PR 则继续在 Generate 请求侧补齐与 Python API 的语义对齐；两者共同表明 Rust gRPC 前端正在逐步覆盖 v1 引擎的请求能力。