PR #45286 完整报告

vllm-project/vllm

[Bugfix][Rust Frontend] Return 400 for prompt-validation submit errors

合并时间：2026-06-12 15:53

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45286

执行摘要

本次 PR 修复了 Rust 前端在 prompt 验证失败时错误地返回 HTTP 500 的问题，改为返回 HTTP 400 invalid_request_error，与 Python 前端行为对齐。通过新增 text_submit_error 和 chat_submit_error 分类函数，将 PromptTooLong、EmptyPromptTokenIds 等客户端错误与服务器内部错误分离，并应用到所有三个提交入口点。新增单元测试覆盖主要变体，包括 Codex 审查中发现的嵌套 Llm 包装情况。

功能与动机

Rust 前端作为 vLLM 的新一代入口，其错误处理与 Python 前端存在不一致。当客户端发送超长 prompt（超过 max_model_len）或空 prompt 时，Python 前端返回 HTTP 400 invalid_request_error，而 Rust 前端却返回 HTTP 500 server_error。这导致客户端 SDK 无法区分“请求无效”与“服务器故障”，常因 5xx 重试逻辑而反复提交不可能成功的请求（PR body 中明确描述）。

实现拆解

1. 核心错误分类逻辑（rust/src/server/src/error.rs）：
   - 添加 text_submit_error 函数，接受 vllm_text::Error，通过 is_prompt_validation_error 判断是否属于 prompt 验证错误，是则返回 400，否则返回 500。
   - 添加 chat_submit_error 函数，处理 vllm_chat::Error，同样分类 PromptTooLong 以及包裹的文本验证错误。
   - is_prompt_validation_error 匹配三种变体：PromptTooLong、EmptyPromptTokenIds 以及通过 Llm 包装的 EmptyPromptTokenIds（最后一种在 prepare 阶段出现，初始版本遗漏，经 Codex 审查后补充）。
2. 三个入口点替换（completions.rs、chat_completions.rs、inference/generate.rs）：
   - 每个路由中原先直接构造 server_error! 并附带上下文字符串和 to_report_string() 的错误处，改为调用对应的分类函数。例如 /v1/completions 中从： 变为：
   - 错误上下文保持不变，非验证错误仍会得到 "failed to submit ..." 前缀的 500 响应。
3. 单元测试（error.rs 内嵌）：
   - 包含四个测试用例，覆盖裸 PromptTooLong、chat 包裹的 PromptTooLong、Llm 包裹的 EmptyPromptTokenIds 以及非验证错误（如 Tokenizer），确保分类正确。

rust/src/server/src/error.rs

核心变更文件，新增错误分类函数和完整单元测试，是本次 PR 的逻辑主体。

```
// use thiserror_ext::AsReport as _; // 新导入，用于获取错误的可读字符串表示

/// Classify a text-pipeline submit failure: tokenized-prompt validation
/// failures (the prompt is too long for the model, or empty after
/// tokenization) are the client's fault and map to HTTP 400, mirroring the
/// Python frontend. Everything else stays an internal 500.
pub fn text_submit_error(context: &'static str, error: vllm_text::Error) -> ApiError {
    if is_prompt_validation_error(&error) {
        return invalid_request!("{error}");
    }
    server_error!("{}: {}", context, error.to_report_string())
}

/// Like [`text_submit_error`], for the chat pipeline (which both wraps the
/// text errors and raises its own prompt-length variant).
pub fn chat_submit_error(context: &'static str, error: vllm_chat::Error) -> ApiError {
    match &error {
        // chat 层直接抛出的 PromptTooLong
        vllm_chat::Error::PromptTooLong { .. } => invalid_request!("{error}"),
        // chat 层包裹的文本错误，若为验证错误也返回 400
        vllm_chat::Error::Text(text_error) if is_prompt_validation_error(text_error) => {
            invalid_request!("{error}")
        }
        _ => server_error!("{}: {}", context, error.to_report_string()),
    }
}

/// Returns true if the text error indicates a prompt validation failure that
/// should be the client's responsibility.
fn is_prompt_validation_error(error: &vllm_text::Error) -> bool {
    matches!(
        error,
        vllm_text::Error::PromptTooLong { .. }
            | vllm_text::Error::EmptyPromptTokenIds { .. }
            // An empty tokenized prompt detected later, at request prepare
            // time, surfaces through the transparent Llm wrapper.
            | vllm_text::Error::Llm(vllm_llm::Error::EmptyPromptTokenIds { .. })
    )
}

#[cfg(test)]
mod tests {
    use super::*;

    #[test]
    fn prompt_too_long_maps_to_invalid_request() {
```

```
        // 裸的 PromptTooLong 应映射到 400 并包含 token 数量信息
        let error = vllm_text::Error::PromptTooLong {
            max_model_len: 8192,
            prompt_len: 9000,
        };
        let api_error = text_submit_error("failed to submit completion request", error);
        assert_eq!(api_error.status_code(), StatusCode::BAD_REQUEST);
        let response = api_error.to_error_response();
        assert_eq!(response.error.error_type, "invalid_request_error");
        assert!(response.error.message.contains("8192"));
        assert!(response.error.message.contains("9000"));
    }

    #[test]
    fn chat_wrapped_prompt_too_long_maps_to_invalid_request() {
        // chat 层包裹的 PromptTooLong 也应返回 400
        let error = vllm_chat::Error::Text(vllm_text::Error::PromptTooLong {
            max_model_len: 8192,
            prompt_len: 9000,
        });
        let api_error = chat_submit_error("failed to submit chat request", error);
        assert_eq!(api_error.status_code(), StatusCode::BAD_REQUEST);
    }

    #[test]
    fn llm_wrapped_empty_prompt_maps_to_invalid_request() {
        // 通过 Llm 包装的 EmptyPromptTokenIds（在 prepare 阶段出现）也应返回 400
        let error = vllm_text::Error::Llm(vllm_llm::Error::EmptyPromptTokenIds {
            request_id: "req-1".to_string(),
        });
        let api_error = text_submit_error("failed to submit completion request", error);
        assert_eq!(api_error.status_code(), StatusCode::BAD_REQUEST);
    }

    #[test]
    fn other_submit_errors_stay_internal() {
        // 非验证错误（如 tokenizer 失败）仍返回 500
        let error = vllm_text::Error::Tokenizer { message: "broken model".to_string() };
        let api_error = text_submit_error("failed to submit completion request", error);
        assert_eq!(api_error.status_code(), StatusCode::INTERNAL_SERVER_ERROR);
    }
}
```

评论区精华

- Codex 审查发现初始版本缺失对 Llm 包装的 EmptyPromptTokenIds 的处理（P2 级别），作者确认后修复并补充测试。这表明即使简单的错误分类也要考虑错误类型的嵌套传递路径。
- BugenZhao表示“The fix looks nice. Thanks!”，并批准合并。

风险与影响

- 风险：
  - 非验证错误的行为完全不变（仍为 500），因此无降级风险。
  - 最大的风险是模式匹配遗漏了未来新增的验证错误变体，但现有测试已覆盖已知变体，且 _ 分支保证安全回退。
- 影响：
  - 客户端：之前收到 500 的无效请求现在收到 400，避免重试浪费。
  - 系统：与 Python 前端错误响应保持一致，降低 API 使用者迁移成本。
  - 范围：仅影响 Rust 前端三个 POST 端点，不涉及 Python 后端或推理引擎。

关联脉络

- 与历史 PR #45300 同属 Rust 前端 bugfix 系列，但功能独立。
- 本次修复使 Rust 前端在错误响应方面更接近 Python 前端，是 Rust 前端成熟度提升的一个步骤。
- 未来可能期望进一步对齐其他错误码（如认证、限流等）。