PR #31417 完整报告

sgl-project/sglang

Return 400 instead of 500 for unfetchable or unparseable multimodal inputs

合并时间：2026-07-28 20:23

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31417

执行摘要

- 一句话：多模态输入获取 / 解码失败时返回 400 而非 500
- 推荐动作：该 PR 是教科书式的错误分类修复，值得精读：通过集中异常元组 CLIENT_MEDIA_EXCEPTIONS 实现跨组件错误分类一致性，并通过 mock 测试全面覆盖非功能性路径。设计思路适用于其他需要区分客户端错误和服务端错误的场景。

功能与动机

当前多模态请求中无法获取或解码的 media URL 或字节会引发 500 InternalServerError，这不符合 OpenAI API 语义（应为 invalid_request_error，400），且污染服务器错误监控。生产环境遇到媒体主机间歇性 404/500 导致大量 SGLang 500，无法与真实服务器故障区分。PR body 指出：'These are client-payload errors, not server faults: by OpenAI API semantics they should be invalid_request_error (400).'

实现拆解

1. 定义集中异常元组：在 python/sglang/srt/utils/common.py 中导入 UnidentifiedImageError，新增 CLIENT_MEDIA_EXCEPTIONS 元组，包含 ValueError、UnidentifiedImageError、requests.exceptions.RequestException。同时修改 load_audio 和 load_video 函数，将特定解码异常（如 sf.LibsndfileError、torchcodec 解码错误）捕获并重新抛出为 ValueError，确保它们被纳入客户端错误分类。
2. 改造 BaseMultimodalProcessor._load_single_item：在 python/sglang/srt/multimodal/processors/base_processor.py 中，将原有的 except ValueError 替换为 except CLIENT_MEDIA_EXCEPTIONS，使不可获取或不可解码的媒体数据统一抛出 ValueError，其他异常仍抛出 RuntimeError。
3. 改造编码服务器 _load_single_item：在 python/sglang/srt/disaggregation/encode_server.py 中新增 except CLIENT_MEDIA_EXCEPTIONS 分支，将其转换为 BadRequestError（而非 ValueError），以适应 DP 环境中的分类机制。
4. 新增单元测试：创建 test/registered/unit/multimodal/test_base_processor_bad_input.py，使用 _StubProcessor 和 mock 全面覆盖：不可获取的 URL（HTTPError/ConnectionError/Timeout）、无效 base64、不可解码的图像 / 音频 / 视频字节；同时验证真正的服务器错误（ImportError/MemoryError）仍保持 RuntimeError，并测试 CLIENT_MEDIA_EXCEPTIONS 元组覆盖正确。

关键文件：

- python/sglang/srt/multimodal/processors/base_processor.py（模块 多模态处理器；类别 source；类型 core-logic；符号 _load_single_item）：核心异常分类逻辑：将 except ValueError 替换为 except CLIENT_MEDIA_EXCEPTIONS，统一客户端错误处理。
- python/sglang/srt/utils/common.py（模块 基础工具；类别 source；类型 dependency-wiring；符号 CLIENT_MEDIA_EXCEPTIONS, load_audio, load_video）：定义 CLIENT_MEDIA_EXCEPTIONS 元组，并修改 load_audio/load_video 将解码异常转为 ValueError。
- python/sglang/srt/disaggregation/encode_server.py（模块 编码服务器；类别 source；类型 core-logic；符号 _load_single_item）：编码服务器中的 _load_single_item 新增 CLIENT_MEDIA_EXCEPTIONS 捕获，转为 BadRequestError。
- test/registered/unit/multimodal/test_base_processor_bad_input.py（模块 测试；类别 test；类型 test-coverage；符号 _StubProcessor, _session_raising, TestBadInputIsClientError, _assert_client_error）：新增全面的单元测试，验证所有客户端错误场景返回 ValueError，服务器错误保持 RuntimeError。

关键符号：BaseMultimodalProcessor._load_single_item, EncodeServer._load_single_item, load_audio, load_video, CLIENT_MEDIA_EXCEPTIONS

关键源码片段

python/sglang/srt/multimodal/processors/base_processor.py

核心异常分类逻辑：将 except ValueError 替换为 except CLIENT_MEDIA_EXCEPTIONS，统一客户端错误处理。

```
# python/sglang/srt/multimodal/processors/base_processor.py (partial)
from sglang.srt.utils import CLIENT_MEDIA_EXCEPTIONS # 新增导入，定义见 common.py

@classmethod
def _load_single_item(cls, data, modality, frame_count_limit=None,
                      audio_sample_rate=None, discard_alpha_channel=True):
    """加载单个多模态数据，若为预计算结果则直接返回。类方法，支持 pickle 用于多进程。"""
    if cls._is_preprocessed_input(data):
        return data
    try:
        if modality == Modality.IMAGE:
            img, _ = load_image(data, cls.gpu_image_decode)
            if isinstance(img, torch.Tensor):
                return img
            if discard_alpha_channel and img.mode != "RGB":
                img = img.convert("RGB")
            img.load() # 在 I/O worker 中立即解码
            return img
        elif modality == Modality.VIDEO:
            return load_video(data, frame_count_limit)
        elif modality == Modality.AUDIO:
```

```
            return load_audio(data, audio_sample_rate)
        except CLIENT_MEDIA_EXCEPTIONS as e:
            # 客户端错误（无法获取 URL / 无法解码媒体）-> 抛出 ValueError，服务层返回 400
            data_str = str(data)
            if len(data_str) > 100:
                data_str = data_str[:100] + "..."
            raise ValueError(f"Error while loading data {data_str}: {e}") from e
        except Exception as e:
            # 真正的服务器端故障 -> 保持 RuntimeError，最终返回 500
            data_str = str(data)
            if len(data_str) > 100:
                data_str = data_str[:100] + "..."
            raise RuntimeError(f"Error while loading data {data_str}: {e}") from e
```

test/registered/unit/multimodal/test_base_processor_bad_input.py

新增全面的单元测试，验证所有客户端错误场景返回 ValueError，服务器错误保持 RuntimeError。

```
# test/registered/unit/multimodal/test_base_processor_bad_input.py

class _StubProcessor(BaseMultimodalProcessor):
    # 使用 CPU 解码，无需 GPU，仅测试 _load_single_item classmethod
    gpu_image_decode = False

class TestBadInputIsClientError(CustomTestCase):
    def _assert_client_error(self, data, modality):
        # 断言客户端错误引发 ValueError
        with self.assertRaises(ValueError):
            _StubProcessor._load_single_item(data, modality)

    def test_unfetchable_url_every_modality(self):
        # 模拟 HTTPError / ConnectionError / Timeout，它们都是 RequestException 子类
        for exc in (
            requests.exceptions.HTTPError("404 from media host"),
            requests.exceptions.ConnectionError("dns failure"),
            requests.exceptions.Timeout("read timed out"),
        ):
            for modality in MODALITIES: # IMAGE / AUDIO / VIDEO
                with self.subTest(exc=type(exc).__name__, modality=modality):
                    with patch("sglang.srt.utils.common.get_mm_http_session",
                               return_value=_session_raising(exc)):
                        self._assert_client_error("https://media.host/clip", modality)

    def test_invalid_base64(self):
        self._assert_client_error("!!!not-base64!!!", Modality.IMAGE)

    def test_undecodable_image_bytes(self):
        # PIL 抛出 UnidentifiedImageError（OSError 子类），不是 ValueError
        self._assert_client_error(b"definitely not an image", Modality.IMAGE)
```

```
    def test_undecodable_audio_bytes(self):
        # soundfile 抛出 LibsndfileError（RuntimeError 子类），不是 ValueError
        self._assert_client_error(b"definitely not audio", Modality.AUDIO)

    def test_undecodable_video_bytes(self):
        # 模拟解码器抛出 RuntimeError（如 "invalid data found"）
        with patch("sglang.srt.utils.common.VideoDecoderWrapper",
                   side_effect=RuntimeError("invalid data found when processing input")):
            self._assert_client_error(b"definitely not a video", Modality.VIDEO)
```

评论区精华

无代码审查讨论。合并者通过 CI 触发测试 test_base_processor_bad_input.py 和 test_base_processor_image_decode.py，两次运行均通过，验证了变更正确性。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。主要风险在于 CLIENT_MEDIA_EXCEPTIONS 可能遗漏某些客户端错误类型，但 RequestException 包含了 HTTPError、ConnectionError、Timeout 等常见子类，UnidentifiedImageError 覆盖 PIL 解码失败，ValueError 覆盖无效 base64；load_audio 和 load_video 中新增的转换排除了 ImportError/MemoryError，避免掩盖服务器故障。兼容性方面，依赖 500 状态码的客户端需调整，但此次变更符合 API 规范，属于正确行为修正。
- 影响：影响所有多模态推理请求的输入校验。用户：无效输入将收到 400 状态码和更清晰的错误信息，而非 500。系统：减少错误监控中的误报，提升可观测性。团队：异常分类更清晰，便于快速定位问题。测试覆盖全面，回归风险低。
- 风险标记：客户端兼容性，异常分类完整性

关联脉络

- PR #30260 [Fix] --mm-process-config crash when video config contains: 同属多模态错误处理改进，修复视频配置参数传递错误。
- PR #32498 [mm] Handle per-item embeddings in cache misses: 同属多模态错误处理改进，处理 cache miss 时的 per-item embeddings。