PR #20835 完整报告

sgl-project/sglang

Fix multimodal /v1/embeddings Jinja chat template handling

合并时间：2026-04-29 04:05

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/20835

执行摘要

- 一句话：修复 multimodal embedding Jinja 模板处理
- 推荐动作：该 PR 修复了已上报 bug，解决方案稳健，测试全面，建议快速合并。值得关注的设计决策是异常处理策略和 multimodal content 构建顺序，对后续模块有参考价值。

功能与动机

Issue #20811 报告：使用 --chat-template 启动 Qwen3-VL-Embedding 时，multimodal embedding 请求未应用 Jinja 对话模板，导致输入文本未经渲染直接传入，影响嵌入质量。该问题在 PR 描述中也被归因为 chat_template_name 为 None 时 embed 路径跳跃了模板处理。

实现拆解

1. 修改 multimodal 分支（serving_embedding.py）：在 _convert_to_internal_request 中原有 chat_template_name 检查之后，增加 elif 分支，检查 self.tokenizer_manager.tokenizer 是否包含 chat_template 属性；若存在则调用新方法 _apply_jinja_template_to_embedding_inputs，否则保留原有 fallback（纯文本）。同时移除原先 item.text if text is not None else "padding" 的填充，改为直接保留 None，避免图像 / 视频仅输入时产生多余文本。
2. 新增 _apply_jinja_template_to_embedding_inputs 方法（serving_embedding.py）：对每个输入项，按 图像 / 视频 / 文本 顺序构建 content_parts 列表，组装消息字典，调用 tokenizer.apply_chat_template 渲染。使用 jinja_template_content_format 参数调整内容格式。捕获 (jinja2.TemplateError, TypeError, KeyError, AttributeError) 并重抛为 ValueError，让上层返回 400 而非 500。
3. 更新测试文件（test_serving_embedding.py）：新增 8 个测试方法，覆盖 Jinja 模板生效、仅图像 / 视频输入、模板回退、无 tokenizer 降级、模板错误抛出等场景。同时调整 _MockTemplateManager 的 jinja_template_content_format 默认值为 "openai" 以匹配新逻辑。
4. 补充导入与异常处理：在 serving_embedding.py 中添加 import jinja2 和 from sglang.srt.parser.jinja_template_utils import process_content_for_template_format。异常捕获范围参考 serving_chat.py 中的已有做法，确保一致性。
5. 合并主分支：共 3 次 merge commit 解决冲突，保持与最新 main 兼容。

关键文件：

- python/sglang/srt/entrypoints/openai/serving_embedding.py（模块 嵌入端点；类别 source；类型 core-logic；符号 _apply_jinja_template_to_embedding_inputs, _convert_to_internal_request）：核心修复文件：新增 _apply_jinja_template_to_embedding_inputs 方法并修改 _convert_to_internal_request 中的 multimodal 分支，使 Jinja 模板能够正确应用于多模态 embedding 输入。
- test/registered/unit/entrypoints/openai/test_serving_embedding.py（模块 测试；类别 test；类型 test-coverage；符号 test_convert_multimodal_request_with_jinja_chat_template, test_convert_image_only_multimodal_request_with_jinja_chat_template, test_convert_video_multimodal_request_with_jinja_chat_template, test_multimodal_request_falls_back_when_no_chat_template）：全面的回归测试覆盖，包括 8 个新测试方法，确保 Jinja 模板在多模态 embedding 中正确应用，以及各种边界和错误场景。

关键符号：_convert_to_internal_request, _apply_jinja_template_to_embedding_inputs

关键源码片段

python/sglang/srt/entrypoints/openai/serving_embedding.py

核心修复文件：新增 _apply_jinja_template_to_embedding_inputs 方法并修改 _convert_to_internal_request 中的 multimodal 分支，使 Jinja 模板能够正确应用于多模态 embedding 输入。

```
def _convert_to_internal_request(self, request, raw_request=None):
    # ... 省略开头 ...
    if isinstance(prompt, list):
        if len(prompt) > 0 and isinstance(prompt[0], MultimodalEmbeddingInput):
            texts, images, videos = [], [], []
            for item in prompt:
                texts.append(item.text) # 保留 None，不再填充 "padding"
                images.append(item.image if item.image is not None else None)
                videos.append(item.video if item.video is not None else None)

            generate_prompts = []
            # Precedence: SGLang-registered conversation template 优先于 tokenizer 自己的 HF
            Jinja template
            if self.template_manager.chat_template_name is not None:
                convs = generate_embedding_convs(
                    texts, images, videos, self.template_manager.chat_template_name
                )
                for conv in convs:
                    generate_prompts.append(conv.get_prompt())
            elif (
                self.tokenizer_manager.tokenizer is not None
                and getattr(self.tokenizer_manager.tokenizer, "chat_template", None) is not None
            ):
                # 使用 tokenizer 的 apply_chat_template 渲染 Jinja 模板
                generate_prompts = self._apply_jinja_template_to_embedding_inputs(
```

```python
                texts, images, videos
            )
        else:
            # 无模板时，text 为 None 则填充 "padding"（兼容原有行为）
            generate_prompts = [
                text if text is not None else "padding" for text in texts
            ]
        # ... 后续组装 prompt_kwargs ...
    # ... 其余逻辑 ...


def _apply_jinja_template_to_embedding_inputs(
    self,
    texts: List[Optional[str]],
    images: List[Optional[str]],
    videos: List[Optional[str]],
) -> List[str]:
    """Render each multimodal embedding input through the tokenizer's Jinja chat template.

    Image/video bytes are threaded to the engine separately via
    ``EmbeddingReqInput.image_data``/``video_data``; this method only produces
    the prompt string. ``text=None`` emits no text chunk (no ``"padding"``
    literal). Jinja failures are re-raised as ``ValueError`` so the caller
    returns HTTP 400 instead of 500.
    """
    prompts: List[str] = []
    template_content_format = self.template_manager.jinja_template_content_format

    for text, image, video in zip(texts, images, videos):
        content_parts = []
        if image is not None:
            content_parts.append({"type": "image_url", "image_url": {"url": image}})
        if video is not None:
            content_parts.append({"type": "video_url", "video_url": {"url": video}})
        if text is not None:
            content_parts.append({"type": "text", "text": text})

        msg_dict = {
            "role": "user",
            "content": content_parts if content_parts else "",
        }
        # 应用 content format 处理（如 openai 格式）
        if template_content_format:
            msg_dict = process_content_for_template_format(msg_dict, template_content_format)

        try:
            rendered = self.tokenizer_manager.tokenizer.apply_chat_template(
                [msg_dict],
                tokenize=False,
```

```python
            add_generation_prompt=True,
        )
    except (jinja2.TemplateError, TypeError, KeyError, AttributeError) as e:
        # 转换为 ValueError，使上层返回 400
        if hasattr(e, "lineno"):
            raise ValueError(f"Chat template error at line {e.lineno}: {e}") from e
        raise ValueError(f"Chat template error: {e}") from e

    prompts.append(rendered)

return prompts
```

test/registered/unit/entrypoints/openai/test_serving_embedding.py

全面的回归测试覆盖，包括 8 个新测试方法，确保 Jinja 模板在多模态 embedding 中正确应用，以及各种边界和错误场景。

```python
def test_convert_multimodal_request_with_jinja_chat_template(self):
    """Multimodal embeddings should apply explicit/HF Jinja chat templates."""
    # 模拟 tokenizer 具有 chat_template 属性
    self.tokenizer_manager.tokenizer.chat_template = "mock-template"
    self.tokenizer_manager.tokenizer.apply_chat_template = Mock(
        side_effect=[
            "<prompt>Hello<image></prompt>",
            "<prompt>World</prompt>",
        ]
    )

    adapted_request, _ = self.serving_embedding._convert_to_internal_request(
        self.multimodal_req
    )

    # 验证生成的 prompt 正确应用了模板
    self.assertEqual(
        adapted_request.text,
        ["<prompt>Hello<image></prompt>", "<prompt>World</prompt>"],
    )
    self.assertEqual(adapted_request.image_data[0], "base64_image_data")
    self.assertIsNone(adapted_request.image_data[1])

    # 验证 apply_chat_template 被正确调用：每条输入一次
    self.assertEqual(
        self.tokenizer_manager.tokenizer.apply_chat_template.call_count, 2
    )
    first_call = self.tokenizer_manager.tokenizer.apply_chat_template.call_args_list[0]
    first_messages = first_call.args[0]
    # 验证消息结构：图像在前，文本在后
    self.assertEqual(first_messages[0]["role"], "user")
    self.assertEqual(first_messages[0]["content"][0]["type"], "image")
    self.assertEqual(first_messages[0]["content"][1]["type"], "text")
```

```
    self.assertEqual(first_messages[0]["content"][1]["text"], "Hello")
    self.assertEqual(first_call.kwargs["tokenize"], False)
    self.assertEqual(first_call.kwargs["add_generation_prompt"], True)

    second_call = self.tokenizer_manager.tokenizer.apply_chat_template.call_args_list[1]
    second_messages = second_call.args[0]
    # 第二项无图像，只有文本
    self.assertEqual(len(second_messages[0]["content"]), 1)
    self.assertEqual(second_messages[0]["content"][0]["type"], "text")
    self.assertEqual(second_messages[0]["content"][0]["text"], "World")
```

评论区精华

Review 中 chatgpt-codex-connector 提出了两个关键问题：

- P1 图像 / 视频仅输入时填充 padding：原实现中 Missing MultimodalEmbeddingInput.text 被替换为 "padding"，经 Jinja 模板渲染后生成了包含 padding 字样的 prompt，影响嵌入准确性。开发者已在后续 commit 中修复，不再填充 None，而是直接保留 None 并在渲染时跳过。
- P2 模板渲染失败应返回 400：若 tokenizer.apply_chat_template 抛出异常（如用户自定义 Jinja 调用 raise_exception），会逃逸为 500 错误。开发者随后采用宽异常捕获并重抛 ValueError，确保返回 400。 这两个问题均在 commit 5bf1e0（Address review: broader exception catch, ...）中得到解决。
- 图像 / 视频仅输入时填充 padding 导致模板渲染异常 (correctness): 开发者已修复，将 texts 列表填充逻辑改为保留 None，并在 _apply_jinja_template_to_embedding_inputs 中跳过 text 为 None 的项。
- 模板渲染失败应返回 400 而非 500 (correctness): 开发者通过宽异常捕获 (jinja2.TemplateError, TypeError, KeyError, AttributeError) 并重抛 ValueError，使上层返回 400。

风险与影响

- 风险：风险较低。新增代码路径仅在显式使用 Jinja 模板时激活，不影响既有行为。主要风险：1）若 tokenizer 的 apply_chat_template 对 content_parts 格式要求严格，可能导致渲染错误，但测试已覆盖常见格式；2）宽异常捕获可能误吞非模板错误，但仅限于 _apply_jinja_template_to_embedding_inputs 内部，且最终仍抛出 ValueError；3）jinja_template_content_format 默认值变更可能影响其他调用方，但测试已同步调整。
- 影响：影响范围限缩于 multimodal embedding 请求（/v1/embeddings 的多模态输入）。对于无模板的部署无影响。用户若依赖 --chat-template 参数，此修复将正确渲染对话模板，提高嵌入质量和一致性。团队收益：增加测试覆盖，减少类似回归可能。
- 风险标记：异常处理依赖模板实现，兼容已有部署无 Jinja 模板

关联脉络

- 暂无明显关联 PR