

PR #53093 完整报告

vllm-project/vllm

[MM] Remove text components from ProcessorInputs

合并时间: 2026-08-21 11:42

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53093>

执行摘要

- 一句话: 从 ProcessorInputs 移除文本字段, MM 处理全面改为 token 输入
- 推荐动作: 值得精读。对于关注 vLLM 多模态数据流演进的工程师, 建议重点阅读 `vllm/multimodal/processing/processor.py` 的 `__call__` 与 `_apply_hf_processor_text_mm`, 以及 `vllm/model_executor/models/transformers/multimodal.py` 的 `Legacy/Offsets` 两条路径差异; 同时值得学习 `claude[bot] review` 中关于“方法重复定义导致参数静默丢失”“绕过 `unpad` 破坏 `cache` 语义”这类重构陷阱的判断方式。MOSS-Audio 的占位符 token 补偿逻辑也是一个很好的边界案例。后续可跟踪作者的下一 PR (`transformers` $\geq 5.10.4$ 升级与 `token-only dummy data`)。

功能与动机

PR body 明确指出: Even for the text input path in AsyncLLM, the prompt is tokenized before being passed to MM processor, so the text path in MM processor is now dead code and can be removed。即文本 prompt 在进入多模态处理器之前就已经被 tokenize, MM 处理器里的文本处理分支已无调用方, 属于死代码; 本 PR 先移除 ProcessorInputs 中的文本 prompt 与 tok_kwargs, 后续 PR 再升级 `transformers` $\geq 5.10.4$, 以便在无需 dummy text 的情况下直接传递多模态数据。

实现拆解

本 PR 是一次跨模块的多模态处理链路重构, 按以下步骤完成:

1. 统一入口编码: `vllm/multimodal/processing/processor.py` 中 `BaseMultiModalProcessor.__call__` 的签名从 `prompt: str` 改为 `prompt: str | list[int]`, 当传入 `str` 时立即用 `self.info.get_tokenizer().encode(prompt, **self.info.default_tok_params.get_encode_kwargs())` 编码为 token id; 此后构造 `ProcessorInputs` 时不再携带文本字段, `prompt` 一律是 token 序列。这一步是全部后续删除的前提。
2. 删除文本路径死代码: 在同一个文件中移除 `_hf_processor_applies_updates` 与 `_apply_hf_processor_text_only` 两个方法, 并让 `_call_hf_processor`、`_apply_hf_processor_text_mm`、`_apply_hf_processor_mm_only` 等签名全部去掉 `tokenization_kwargs/tok_kwargs` 参数。`tokenization_kwargs` 从 `ProcessorInputs` 中移除, `vllm/multimodal/processing/inputs.py` 的 `ProcessorInputs` 数据契约同步收窄。

3. Transformers 后端同步简化: `vllm/model_executor/models/transformers/multimodal.py` 中 `_MultiModalProcessorBase._call_hf_processor` 去掉 `tok_kwargs` 参数, placeholder-only 回退分支的 `encode` 调用不再传 `add_special_tokens`; `_apply_hf_processor_main` 相应简化。同时 `OffsetsMultiModalProcessor._apply_hf_processor_main` 移除已无调用方的 `tokenization_kwargs` 形参, `LegacyMultiModalProcessor._call_hf_processor` 的重复定义在后续 commit 中合并。
4. 特殊模型补偿: 由于去掉文本路径后, MOSS-Audio 这类“占位符 token 依赖 HF processor 运行时补齐”的模型会失去稳定 token 序列, `vllm/model_executor/models/moss_audio.py` 新增 `_ensure_moss_audio_placeholder_tokens`, 在 `MossAudioProcessingInfo.__init__` 中把缺失的占位符 token 注册为 special token。CLIP/SigLIP 的 `apply` 也删除了对 str prompt 的分支判断, 因为此时输入恒为 token 序列。
5. 测试与文档配套: `tests/models/multimodal/processing/test_common.py` 删除围绕 `get_text_token_prompts`、`get_token_prompt` 的用例; `vllm/model_executor/models/gemma4_mm.py`、`llava_onevision.py` 等模型文件删除各自的 `_hf_processor_applies_updates/_apply_hf_processor_text_only` 重写; 文档同步更新 (docs 相关改动)。期间经历了 10 个 commit, 含多次合并 main、修复冲突与修复失败测试。

关键文件:

- `vllm/multimodal/processing/processor.py` (模块 MM 处理; 类别 source; 类型 core-logic; 符号 `_hf_processor_applies_updates`, `_apply_hf_processor_text_only`, `call`) : MM 处理主路径: `__call__` 统一 tokenize 文本 prompt, 移除 `tok_kwargs` 与文本死代码 (`_hf_processor_applies_updates`、`_apply_hf_processor_text_only`), 是本 PR 的变更中枢。
- `vllm/model_executor/models/transformers/multimodal.py` (模块 HF 适配; 类别 source; 类型 data-contract; 符号 `_apply_hf_processor_main`, `_call_hf_processor`, `_unpad_audios`) : Transformers 后端 MM 处理器的统一下游: `_call_hf_processor` 与 `_apply_hf_processor_main` 去掉 `tok_kwargs`, Legacy 路径在 review 中被发现重复定义并最终合并。
- `vllm/model_executor/models/moss_audio.py` (模块 音频模型; 类别 source; 类型 data-contract; 符号 `_ensure_moss_audio_placeholder_tokens`, `init`) : MOSS-Audio 因移除文本路径而需要补偿: 新增 `_ensure_moss_audio_placeholder_tokens` 在处理器初始化时注册缺失的占位符 token, 防止 BPE 合并破坏 prompt update 匹配。
- `vllm/model_executor/models/qwen2_5_omni_thinker.py` (模块 Qwen 模型; 类别 source; 类型 data-contract; 符号 `_apply_hf_processor_main`, `_apply_hf_processor_mm_only`) : Qwen2.5-Omni 此前专门为文本路径重写了 `_apply_hf_processor_main`, 本次整个文本分支被删除, 是死代码清理的代表性样本。
- `vllm/model_executor/models/clip.py` (模块 视觉模型; 类别 source; 类型 data-contract; 符号 `_hf_processor_applies_updates`, `apply`) : CLIP 的 `apply` 不再需要区分 str/token prompt 两种输入, 逻辑简化为只判断 token 是否全为 special token, 是本次契约收窄的直接受益者。

- vllm/model_executor/models/siglip.py (模块 视觉模型; 类别 source; 类型 data-contract; 符号 _hf_processor_applies_updates, apply) : 与 CLIP 同样的简化模式, SigLIP 处理器的 apply 与 _hf_processor_applies_updates 被清理。
- tests/models/multimodal/processing/test_common.py (模块 MM 测试; 类别 test; 类型 test-coverage; 符号 get_text_token_prompts, get_token_prompt) : 测试配套主文件: 移除围绕文本 prompt 的 get_text_token_prompts/get_token_prompt 用例, 与其他测试文件一起验证纯 token 输入路径。

关键符号: BaseMultiModalProcessor.call, BaseMultiModalProcessor._apply_hf_processor_text_mm, BaseMultiModalProcessor._call_hf_processor, _ensure_moss_audio_placeholder_tokens, MossAudioProcessingInfo.init, _MultiModalProcessorBase._call_hf_processor, LegacyMultiModalProcessor._call_hf_processor, OffsetsMultiModalProcessor._apply_hf_processor_main, Qwen2_5OmniThinkerMultiModalProcessor._apply_hf_processor_mm_only, BaseDummyInputsBuilder.get_dummy_processor_inputs

关键源码片段

vllm/multimodal/processing/processor.py

MM 处理主路径: __call__ 统一 tokenize 文本 prompt, 移除 tok_kwargs 与文本死代码 (_hf_processor_applies_updates、_apply_hf_processor_text_only), 是本 PR 的变更中枢。

```
def __call__(
    self,
    prompt: str | list[int],
    mm_items: MultiModalDataItems,
    mm_uuid_items: MultiModalUUIDItems | None = None,
    hf_processor_mm_kwargs: Mapping[str, object] | None = None,
) -> MultiModalInput:
    # 统一在入口把文本 prompt 转成 token id 序列。
    # AsyncLLM 在调用 MM processor 之前已完成 tokenize,
    # 因此这里接受 str 只是为了兼容直接调用方;
    # 转换后 ProcessorInputs 内只保留 token 形式, 不再有文本字段。
    if isinstance(prompt, str):
        tokenizer = self.info.get_tokenizer()
        prompt = tokenizer.encode(
            prompt,
            **self.info.default_tok_params.get_encode_kwargs(),
        )

    # 构造纯 token 形式的 ProcessorInputs,
    # hf_processor_mm_kwargs 作为唯一剩余的 MM 参数入口。
    processor_inputs = ProcessorInputs(
        prompt,
        mm_items,
        mm_uuid_items,
```

```

        hf_processor_mm_kwargs=hf_processor_mm_kwargs or {},
    )

```

```

    return self.apply(processor_inputs, TimingContext(enabled=False))

```

vllm/model_executor/models/transformers/multimodal.py

Transformers 后端 MM 处理器的统一下游: `_call_hf_processor` 与 `_apply_hf_processor_main` 去掉 `tok_kwargs`, Legacy 路径在 review 中被发现重复定义并最终合并。

```

def _call_hf_processor(
    self,
    prompt: str,
    mm_data: Mapping[str, object],
    mm_kwargs: Mapping[str, object],
) -> "BatchFeature":
    """运行 HF processor 并对输出做 unpad 处理。"""
    try:
        # tok_kwargs 已从调用链移除: prompt 在进入 MM 处理之前
        # 由入口统一编码为 token id, 文本语义不再需要二次传递。
        hf_inputs = super()._call_hf_processor(prompt, mm_data, mm_kwargs)
    except ValueError:
        if any(mm_data.values()):
            raise
        # 部分 processor 拒绝“只含占位符但没有数据”的 prompt,
        # 此时直接用 tokenizer 编码, 不经过 HF processor。
        prompt_ids = self.info.get_tokenizer().encode(prompt)
        hf_inputs = transformers.BatchFeature(
            dict(input_ids=[prompt_ids]), tensor_type="pt"
        )

    # 去掉 processor 按 batch 最大尺寸填充的冗余部分,
    # 保证 MM processor cache 中每个 item 的结果只依赖自身数据。
    self._unpad_images(hf_inputs)
    self._unpad_audios(hf_inputs, mm_data, mm_kwargs)
    return hf_inputs

```

评论区精华

claude[bot] 的自动化 review 发现了多处阻塞性问题, 作者通过后续 commit 逐一修复:

- LegacyMultiModalProcessor 双重定义 `_call_hf_processor`: 第二个定义覆盖第一个, 导致 `return_mm_token_type_ids=True` 永远传不到 HF processor, 进而使 `_apply_vision` 拿不到 `mm_token_type_ids` 定位图片占位符。最终合并为单一定义。
- Legacy 路径绕过 unpad: 直接调用 `ctx.call_hf_processor` 会跳过 `_unpad_images/_unpad_audios`, 导致 padded 后的 `pixel_values` 被错误地按 `flat_from_sizes` 拆分。最终改回 `super()._call_hf_processor`。

- `__call__` 无条件 tokenize 破坏 fake tokenizer: `test_moss_audio.py` 的 fake tokenizer 只接受 `add_special_tokens`, 新代码传入 `truncation` 等 kwargs 会抛 `TypeError`。
- dummy input 对 `skip_tokenizer_init=True` 崩溃: embed-only 模型 (如 `PrithviGeoSpatialMAE`) 在没有 tokenizer 时调用 `get_tokenizer()` 会抛错, 最终版本在 `get_dummy_processor_inputs` 中增加 tokenizer is None 的保护分支。

另外 `Isotr0py` 提出了两个前瞻性设计问题:

- 是否应弃用 `get_dummy_text`、直接用 token id 构建 dummy data? 作者回答“planned in the next PR”。
- 是否应放弃整体调用 HF processor, 改为按需逐个调用 image/video/audio processor? 作者认可方向, 但深入调查后发现 `Qwen2.5-VL` 的 `seconds_per_grid_ts` 只在顶层 processor 中设置 (v5.15.1 源码第 138 行), 逐个调用并不 straightforward, 同样留待下一 PR。
- 入口无条件 tokenize 破坏 fake tokenizer (testing): 作者通过后续 commit (Fix failing tests) 更新 fake tokenizer 或调整调用方式解决。
- dummy input 在 `skip_tokenizer_init=True` 时崩溃 (correctness): 最终版本改为先取 `self.info.ctx.tokenizer`, 为 None 时走 tokenizer-less 分支构造 dummy prompt。
- `LegacyMultiModalProcessor` 重复定义 `_call_hf_processor` (correctness): 作者合并两个定义为单一定义, 保留 `return_mm_token_type_ids` 与 `unpad` 行为 (Address comment 提交)。
- Legacy 路径绕过 `unpad` 处理 (correctness): 最终改为调用 `super()._call_hf_processor`, 保留 `unpad` 逻辑。
- `OffsetsMultiModalProcessor` 仍传递 `enable_hf_prompt_update` (correctness): 作者同步移除该参数, 保证 `Offsets` 路径与基类契约一致。
- placeholder-only fallback 丢失 `add_special_tokens=False` (correctness): 属于窄路径风险, 最终代码未显式传 `add_special_tokens`, 依赖上游 `decode/encode` 的对称性; 风险已记录但仍存在。
- 是否弃用 `get_dummy_text` 改用 token id 构造 dummy data (design): `DarkLight1337` 确认已在下一 PR 计划中实现。
- 是否逐个调用各模态 processor 取代整体 HF processor (design): 方向可行但实现复杂, 留待下一 PR 继续探索; 同时可兼顾 HF Hub 上 transformers<5.10 自定义代码的兼容。

风险与影响

- 风险: 本 PR 是 MM 处理主链路的跨模块重构, 风险集中在以下方面:
- 回归风险 (高): 92 个文件、净删 806 行, 涉及 `vllm/multimodal/processing/processor.py` 与 `vllm/model_executor/models/transformers/multimodal.py` 这两条所有 Transformers 后端 MM 模型的必经路径。review 期间暴露的 Legacy 双重定义、`unpad` 绕过等问题说明该区域极易在重构中静默丢失行为, 需依赖完整的多模态模型测试矩阵兜底。
- tokenizer 兼容性: 入口处新增 `tokenizer.encode(prompt, **default_tok_params.get_encode_kwargs())` 调用, 要求调用方 tokenizer 必须接受 `truncation/max_length` 等 kwargs; 任何只实现部分签名的自定义 tokenizer (或测试

fake) 都会在运行时抛 `TypeError`。

- `skip_tokenizer_init=True` 边界: embed-only 模型在 memory profiling 阶段构建 dummy processor inputs 时不能访问 tokenizer, 最终版本虽加了 tokenizer is None 保护, 但仍需确认所有 embed-only 模型 (PrithviGeoSpatialMAE 等) 的 dummy 路径不依赖文本。
- 特殊 token 重复: placeholder-only fallback 中 `encode(prompt)` 不再显式传 `add_special_tokens=False`, 而 `LegacyMultiModalProcessor.apply` 解码出的文本已含特殊 token, 窄路径下可能重复添加 BOS/EOS。
- 行为变更: `MossAudioProcessingInfo.__init__` 新增对 tokenizer 的写操作 (`add_tokens`), 会改变 tokenizer 状态, 需确认与 cache、多租户共用 tokenizer 的场景兼容。
- 影响: 对用户与模型侧基本透明: 对外 API 与请求格式没有变化, MM 模型的推理结果预期保持不变, 但任何自定义 MM processor 或依赖 `ProcessorInputs` 文本字段的扩展点都会因数据契约收窄而需要同步迁移。对系统而言, 这是多模态输入处理向纯 token 形态演进的关键一步: 删除文本路径后, `ProcessorInputs` 更简单、可控, 为后续升级 `transformers>=5.10.4`、移除 dummy text、甚至按需逐个调用各模态 processor 铺平道路。对团队而言, 影响面较大: 一次触及大量模型文件 (qwen、llama、deepseek、mistral、glm、minimax、inkling、cohere、kimi 等), 后续需在这些模型上回归验证, 且 claude[bot] review 暴露的问题表明该区域需要更强的自动化检查。
- 风险标记: 92 文件跨模块重构, MM 处理主链路变更, tokenizer 兼容性风险, `skip_tokenizer_init` 边界, 特殊 token 重复风险, 多处 review 阻塞问题已修复

关联脉络

- PR #53064 (前置 PR, 未在此次历史数据中提供标题): PR body 明确声明本 PR 是 #53064 的 follow-up: 文本 prompt 在进入 MM processor 前已 tokenize, 本 PR 基于该前提移除文本路径。