PR #26116 完整报告

sgl-project/sglang

[VLM] Reuse Qwen pretokenized ids

合并时间：2026-05-23 16:01

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26116

执行摘要

- 一句话：复用 Qwen VLM 预 tokenize 的 ids 和 MRoPE 元数据
- 推荐动作：值得精读，尤其关注 Qwen 模型前处理数据流和跨模块数据复用的设计模式。建议作者为 build_padded_input_ids 和 MRoPE 复用逻辑补充单元测试，以防止未来回归。

功能与动机

Qwen VLM 处理器（processor）在内部可能已对输入文本进行分词并生成完整的 token ids、padded ids 以及 MRoPE 位置信息，但之前流程会丢弃这些结果，自行再次分词和计算 MRoPE，造成不必要的开销。PR 旨在利用处理器已有的输出，避免重复计算，提升推理前处理效率。

实现拆解

1. 在 qwen_vl.py 中新增辅助方法：添加 _get_processor_output_value 和 _get_precomputed_mrope_from_output 两个方法，用于从处理器返回对象中安全提取 mrope_positions 和 mrope_position_delta，并校验形状以确认数据可用。
2. 修改 process_mm_data_async 逻辑：先检查 base_output.input_ids 是否与展平后的 input_ids 长度一致（数据一致），若一致则复用 input_ids 列表；接着尝试从 ret 中获取预计算的 padded_input_ids，若不存在则调用 MultimodalProcessorOutput.build_padded_input_ids 构建；最后优先使用预计算的 MRoPE 数据，仅在不存在后备回 MRotaryEmbedding.get_rope_index 计算。
3. 在 schedule_batch.py 中扩展数据模型：在 MultimodalProcessorOutput 中加入 padded_input_ids 字段并在 from_dict 中传递；新增静态方法 build_padded_input_ids，根据 mm_items 中的 offsets 和 pad_value 将 input_ids 中的图像 / 视频占位符替换为实际填充值，生成可直接用于模型的 padded input ids。
4. 同步修改 MultimodalInputs：增加 padded_input_ids 字段，并在 from_processor_output 中从 obj.padded_input_ids 赋值，确保下游调度器能接收到复用数据。

关键文件：

- python/sglang/srt/multimodal/processors/qwen_vl.py（模块 多模态处理器；类别 source；类型 core-logic；符号 _get_processor_output_value, _get_precomputed_mrope_from_output）：核心变更文件，新增辅助方法并修改

process_mm_data_async 以复用预计算数据

- python/sglang/srt/managers/schedule_batch.py（模块 调度批处理；类别 source；类型 core-logic；符号 build_padded_input_ids）：数据模型扩展，新增 padded_input_ids 字段和构建方法，支撑复用数据传递

关键符号：_get_processor_output_value, _get_precomputed_mrope_from_output, build_padded_input_ids, process_mm_data_async

关键源码片段

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

核心变更文件，新增辅助方法并修改 process_mm_data_async 以复用预计算数据

```
# python/sglang/srt/multimodal/processors/qwen_vl.py

# TODO: consider moving it to SGLangBaseProcessor
@staticmethod
def _get_processor_output_value(ret, key):
    """从 processor 返回对象中安全提取值，支持 dict 和 attribute"""
    if ret is None:
        return None
    if hasattr(ret, "get"):
        value = ret.get(key)
        if value is not None:
            return value
    return getattr(ret, key, None)

def _get_precomputed_mrope_from_output(self, ret):
    """从 processor 输出中提取 MRoPE 位置，验证形状后返回"""
    mrope_positions = self._get_processor_output_value(ret, "mrope_positions")
    mrope_position_delta = self._get_processor_output_value(ret, "mrope_position_delta")
    if mrope_positions is None or mrope_position_delta is None:
        return None
    mrope_positions = torch.as_tensor(mrope_positions)
    # 处理可能的 batch 维度 (B,1,3,L) -> (3,L)
    if mrope_positions.ndim == 3:
        if mrope_positions.shape[1] != 1:
            return None
        mrope_positions = mrope_positions.squeeze(1)
    if mrope_positions.ndim != 2 or mrope_positions.shape[0] != 3:
        return None
    mrope_position_delta = torch.as_tensor(mrope_position_delta)
    if mrope_position_delta.ndim == 0:
        mrope_position_delta = mrope_position_delta.reshape(1, 1)
    elif mrope_position_delta.ndim == 1:
        mrope_position_delta = mrope_position_delta.reshape(-1, 1)
    return mrope_positions, mrope_position_delta

# 在 process_mm_data_async 中关键复用逻辑（省略上下文）:
```

```
# 优先使用 base_output.input_ids 列表，避免 tolist() 转换
base_input_ids = getattr(base_output, "input_ids", None)
if (isinstance(base_input_ids, list) and len(base_input_ids) == input_ids.numel()):
    input_ids_list = base_input_ids
else:
    input_ids_list = input_ids.tolist()

# 尝试复用 precomputed padded_input_ids
padded_input_ids = self._get_processor_output_value(ret, "padded_input_ids")
if padded_input_ids is None:
    padded_input_ids = MultimodalProcessorOutput.build_padded_input_ids(input_ids_list, mm_
    items)
elif isinstance(padded_input_ids, torch.Tensor):
    padded_input_ids = padded_input_ids.flatten().tolist()
else:
    padded_input_ids = list(padded_input_ids)

# 优先使用预计算的 MRoPE 数据，否则回退原始计算
precomputed_mrope = self._get_precomputed_mrope_from_output(ret)
if precomputed_mrope is not None:
    mrope_positions, mrope_position_delta = precomputed_mrope
else:
    mrope_positions, mrope_position_delta = MRotaryEmbedding.get_rope_index(...)
```

python/sglang/srt/managers/schedule_batch.py

数据模型扩展，新增 padded_input_ids 字段和构建方法，支撑复用数据传递

```
# python/sglang/srt/managers/schedule_batch.py

@dataclasses.dataclass
class MultimodalProcessorOutput:
    """多模态处理器原始输出，尚未计算 padding 和 hash"""
    mm_items: List[MultimodalDataItem]
    input_ids: Optional[List[int]] = None
    padded_input_ids: Optional[List[int]] = None # 新增：预先填充的 input ids
    # ... 其余字段不变

    @staticmethod
    def build_padded_input_ids(input_ids, mm_items: List[MultimodalDataItem]):
        """根据 mm_items 中的偏移和 pad 值，将 input_ids 中的占位符替换为实际值"""
        if input_ids is None or not mm_items:
            return None
        # 确保所有 item 都有必要的 pad 信息
        for item in mm_items:
            if item.pad_value is None or item.offsets is None:
                return None
        if isinstance(input_ids, torch.Tensor):
            padded_input_ids = input_ids.flatten().tolist()
        else:
```

```
            padded_input_ids = list(input_ids)
        for item in mm_items:
            for start, end in item.offsets:
                padded_input_ids[start : end + 1] = [item.pad_value] * (end - start + 1)
        return padded_input_ids

@dataclasses.dataclass
class MultimodalInputs:
    """多模态输入数据，用于调度"""
    mm_items: List[MultimodalDataItem]
    padded_input_ids: Optional[List[int]] = None # 新增字段，接收复用值
    # ... 其余字段不变

    @staticmethod
    def from_processor_output(obj: MultimodalProcessorOutput):
        # ...
        ret = MultimodalInputs(
            mm_items=mm_items,
            padded_input_ids=obj.padded_input_ids, # 传递复用值
        )
```

评论区精华

PR 无 reviewer 讨论，所有评论为 CI 重跑指令。代码中的 `# TODO: consider moving it to SGLangBaseProcessor` 提示未来可将通用辅助方法提升到基类，说明作者已考虑跨模型复用性。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 数据一致性风险：process_mm_data_async 中比较 base_output.input_ids 长度与展平 input_ids 的元素数，若长度一致但实际 token 不同（如微调后 prompt 格式变更但长度未变），会导致 token 错乱。该检查不够严格，但实际场景中长度相同的不同序列概率较低。
  2. MRoPE 形状校验过于宽松：_get_precomputed_mrope_from_output 仅检查 ndim 和 shape[0]==3，未验证序列长度维度是否匹配，可能传递错误尺寸。
  3. 缺少测试覆盖：改动涉及两个关键模块，未添加单元测试或集成测试，回归风险依赖人工验证。 - 影响：影响范围：仅影响 Qwen 系列 VLM 模型（qwen_vl.py 对应模型），覆盖所有启用 multimodal processor 的请求。影响程度：中度提升前处理阶段性能（减少分词和 MRoPE 计算），对长上下文或多图片场景效果更明显。不改变模型计算或解码逻辑，输出结果应与之前一致（当预计算可用时）。团队影响：低，代码变更集中在两个文件，易于审查和维护。
- 风险标记：缺少测试覆盖，数据一致性检查较宽松，MRoPE 形状校验不严格

关联脉络

- PR #24144 [BugFix][EPD] adapt for qwen3.5-mtp & del duplicated logs: 相同文件 qwen_vl.py 的近期改动，涉及 Qwen 模型适配
- PR #26121 [diffusion] Auto-select VAE channels_last_3d: 同属多模态生成模块的性能优化 PR，复用预计算思路相似