PR #23503 完整报告

sgl-project/sglang

fix retrive -> retrieve typo

合并时间：2026-04-23 07:35

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23503

执行摘要

- 一句话：修复 Python 侧 'retrive' 拼写错误为 'retrieve'，保持内核兼容性。
- 推荐动作：该 PR 值得快速浏览以了解拼写标准化和内核兼容性设计；对于新贡献者，可关注如何在保持向后兼容的前提下进行重命名。

功能与动机

修复拼写错误以提高代码可读性和一致性，避免未来混淆。PR body 明确指出：'Rename retrive -> retrieve on the Python side. Kernel schema (sgl-kernel) is untouched; at the kernel-call boundary the kwarg LHS is kept as retrive_* to match the existing op signature.'

实现拆解

1. 识别并修改 Python 侧拼写错误：在 16 个 Python 源码文件中，将所有变量名、字段名从 'retrive'（如 retrive_index）重命名为 'retrieve'（如 retrieve_index），涉及推测解码、批处理重叠和模型执行器等模块。
2. 保持内核兼容性：在调用内核操作（如 sgl_build_tree_kernel_efficient）时，关键字参数的左侧（LHS）仍使用 'retrive_*' 以匹配现有内核 schema，例如在 eagle_utils.py 中，调用时传递 retrive_index=retrieve_index。
3. 修改关键数据结构：更新了多个数据类（如 EagleVerifyInput）和函数中的字段名，确保内部逻辑一致。
4. 无测试或配置配套改动：本次变更纯属重命名，未添加或修改测试文件，也未影响配置或部署。

关键文件：

- python/sglang/srt/batch_overlap/two_batch_overlap.py（模块 批处理重叠；类别 source；类型 core-logic；符号 split_spec_info）：修改了 split_spec_info 函数中的变量名，影响批处理重叠逻辑，是核心变更之一。
- python/sglang/srt/speculative/eagle_utils.py（模块 推测解码；类别 source；类型 core-logic；符号 build_tree_kernel_efficient, verify_tree_greedy_func）：修改了 build_tree_kernel_efficient 和 verify_tree_greedy_func 函数，涉及推测解码树构建和验证逻辑。

- python/sglang/srt/speculative/ngram_worker.py（模块 推测解码；类别 source；类型 core-logic；符号 _init_preallocated_tensors, _prepare_for_speculative_decoding）：修改了类属性和方法中的变量名，影响 N-gram 推测解码工作器。
- python/sglang/srt/speculative/eagle_info.py（模块 推测解码；类别 source；类型 core-logic；符号 EagleVerifyInput）：修改了 EagleVerifyInput 数据类的字段名，是推测解码输入结构的关键变更。
- python/sglang/srt/model_executor/cuda_graph_runner.py（模块 模型执行器；类别 source；类型 data-contract；符号 get_spec_info）：修改了数据契约中的字段名，影响 CUDA 图运行器的 spec info 处理。

关键符号：split_spec_info, build_tree_kernel_efficient, verify_tree_greedy_func, EagleVerifyInput.create_idle_input, get_spec_info

关键源码片段

python/sglang/srt/batch_overlap/two_batch_overlap.py

修改了 split_spec_info 函数中的变量名，影响批处理重叠逻辑，是核心变更之一。

```
def split_spec_info(
    spec_info: SpecInfo,
    start_seq_index: int,
    end_seq_index: int,
    start_token_index: int,
    end_token_index: int,
) -> SpecInfo:
    # ... 其他代码 ...
    if spec_info.retrieve_index is not None: # 将 retrive_index 改为 retrieve_index
        retrieve_index = spec_info.retrieve_index[start_seq_index:end_seq_index]
    else:
        retrieve_index = None
    if spec_info.retrieve_next_token is not None: # 类似重命名
        retrieve_next_token = spec_info.retrieve_next_token[start_seq_index:end_seq_index]
    else:
        retrieve_next_token = None
    if spec_info.retrieve_next_sibling is not None:
        retrieve_next_sibling = spec_info.retrieve_next_sibling[start_seq_index:end_seq_index]
    else:
        retrieve_next_sibling = None
    if spec_info.retrieve_cum_len is not None:
        retrieve_cum_len = spec_info.retrieve_cum_len[start_seq_index:end_seq_index]
    else:
        retrieve_cum_len = None
    # ... 其他代码 ...
    output_spec_info = replace(
        spec_info,
        custom_mask=custom_mask,
        draft_token=draft_token,
        positions=positions,
```

```
            retrieve_index=retrieve_index, # 更新字段名
            retrieve_next_token=retrieve_next_token,
            retrieve_next_sibling=retrieve_next_sibling,
            retrieve_cum_len=retrieve_cum_len,
            seq_lens_cpu=seq_lens_cpu,
            seq_lens_sum=seq_lens_sum,
        )
        return output_spec_info
```

python/sglang/srt/speculative/eagle_utils.py

修改了 build_tree_kernel_efficient 和 verify_tree_greedy_func 函数，涉及推测解码树构建和验证逻辑。

```
def build_tree_kernel_efficient(
    parent_list: torch.Tensor,
    top_scores_index: torch.Tensor,
    seq_lens: torch.Tensor,
    tree_mask_mode: TreeMaskMode,
    device: torch.device,
    position_buf: Optional[torch.Tensor] = None,
) -> Tuple[torch.Tensor, ...]:
    # ... 其他代码 ...
    retrieve_buf = torch.full( # 将 retrive_buf 改为 retrieve_buf
        (3, bs, num_verify_tokens), -1, device=device, dtype=torch.long
    )
    retrieve_index, retrieve_next_token, retrieve_next_sibling = retrieve_buf # 重命名变量
    # ... 其他代码 ...
    if _is_npu:
        torch.ops.npu.build_tree_kernel_efficient(
            parent_list.to(dtype=torch.int64),
            top_scores_index,
            seq_lens,
            tree_mask,
            positions,
            retrieve_index, # 传递重命名后的变量
            retrieve_next_token,
            retrieve_next_sibling,
            topk,
            spec_steps,
            num_verify_tokens,
            tree_mask_mode,
        )
    else:
        sgl_build_tree_kernel_efficient(
            parent_list,
            top_scores_index,
            seq_lens,
            tree_mask,
            positions,
```

```
            retrieve_index, # 类似传递
            retrieve_next_token,
            retrieve_next_sibling,
            topk,
            spec_steps,
            num_verify_tokens,
            tree_mask_mode,
        )
    return (
        tree_mask,
        positions,
        retrieve_index, # 返回重命名后的变量
        retrieve_next_token,
        retrieve_next_sibling,
        draft_tokens,
    )
```

评论区精华

review 评论为空，无讨论。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低：
  - 回归风险：拼写重命名不影响核心逻辑，但需确保所有使用处均已更新，避免遗漏导致运行时 AttributeError 或 KeyError。例如，在 two_batch_overlap.py 中，若遗漏 retrive_index 的引用，可能引发错误。
  - 兼容性风险：内核边界已处理，保持向后兼容，无 breaking change。
  - 性能与安全风险：无影响。
- 影响：影响范围有限：
  - 用户影响：无直接影响，系统行为不变。
  - 系统影响：代码可读性提升，便于维护和未来开发。
  - 团队影响：开发者需注意新命名约定，但变更机械简单，易于适应。
  - 风险标记：拼写错误修复，兼容性保持，无测试覆盖

关联脉络

- PR #23465 Add 'allready' to ignore words list in .codespellrc: 同为拼写相关修复，涉及代码拼写检查配置，可视为拼写标准化工作的一部分。