PR #47772 完整报告

vllm-project/vllm

[Bugfix][Pooling] Align CrossEncoder token type ids after truncation

合并时间：2026-07-08 11:59

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47772

执行摘要

- 一句话：修复 CrossEncoder 左截断后 token type ids 错位导致分数异常
- 推荐动作：值得精读，特别是需要处理 CrossEncoder token type ids 对齐的开发者。_apply_post_tokenization_to_token_type_ids 函数的实现可以作为参考模式，用于其他需要与 prompt 同步后分词处理的场景。

功能与动机

PR 描述明确指出：当使用 truncate_prompt_tokens 且 truncation_side="left" 时，left-truncated prompt 可能全部来自 document 段，但 compress_token_type_ids 基于原始未截断的 token_type_ids 计算的分割点仍然将截断后的 token 标记为 query 类型，导致 CrossEncoder 模型（如 cross-encoder/ms-marco-MiniLM-L6-v2）返回错误分数。PR 背景提到最近合并的 #47082 保留了 extra_kwargs 但未更新截断后的分割边界。

实现拆解

1. 新增独立函数 _apply_post_tokenization_to_token_type_ids（vllm/entrypoints/pooling/scoring/io_processor.py）：对 token_type_ids 列表应用与 engine prompt 相同的后分词处理逻辑，包括填充（pad）和截断（truncate），处理负数 pad_length/max_length 表示使用 max_input_tokens、默认截断方向（优先 tok_params 后 tokenizer），并区分 left 和 right 截断方向。
2. 调整 ScoringIOProcessor._pre_process 方法：将 token_type_ids 的弹出和 apply_post_tokenization 的调用顺序重构——先弹出 token type ids，再对 engine prompt 应用后分词处理，最后将 token type ids 通过新函数处理后，传递给 compress_token_type_ids。这样保证 token_type_ids 与截断后的 prompt 长度和内容对齐。
3. 保留现有 extra_kwargs：cache_salt 等已有额外参数在重构后的代码中仍然通过 params.extra_kwargs 传递。
4. 新增单元测试 test_token_type_ids_follow_post_tokenization（tests/entrypoints/pooling/scoring/test_cross_encoder_offline.py）：使用模拟的 CrossEncoderIOProcessor 实例测试两种场景：左截断 16 tokens 后 compressed_token_type_ids 应为 0（全部来自 document）；填充到 40 tokens 后 compressed_token_type_ids 仍为 16（原有分割点）。

关键文件：

- vllm/entrypoints/pooling/scoring/io_processor.py（模块 入口处理；类别 source；类型 core-logic；符号 _apply_post_tokenization_to_token_type_ids）：核心逻辑所在文件。新增 _apply_post_tokenization_to_token_type_ids 函数，并重构 _pre_process 中 token_type_ids 的处理顺序。
- tests/entrypoints/pooling/scoring/test_cross_encoder_offline.py（模块 测试；类别 test；类型 test-coverage；符号 test_token_type_ids_follow_post_tokenization）：新增 test_token_type_ids_follow_post_tokenization 单元测试，通过模拟 CrossEncoderIOProcessor 验证左截断和填充场景下 compressed_token_type_ids 的正确性。

关键符号：_apply_post_tokenization_to_token_type_ids, ScoringIOProcessor._pre_process

关键源码片段

vllm/entrypoints/pooling/scoring/io_processor.py

核心逻辑所在文件。新增 _apply_post_tokenization_to_token_type_ids 函数，并重构 _pre_process 中 token_type_ids 的处理顺序。

```
# vllm/entrypoints/pooling/scoring/io_processor.py

def _apply_post_tokenization_to_token_type_ids(
    tokenizer: Any,
    tok_params: TokenizeParams,
    token_type_ids: list[int],
) -> list[int]:
    # 处理填充逻辑
    pad_length = tok_params.pad_prompt_tokens
    if pad_length is not None and pad_length < 0:
        pad_length = tok_params.max_input_tokens
    if pad_length is not None and pad_length > len(token_type_ids):
        pad_token_type_id = token_type_ids[-1] if token_type_ids else 0
        token_type_ids = token_type_ids + [pad_token_type_id] * (
            pad_length - len(token_type_ids)
        )

    # 处理截断逻辑
    max_length = tok_params.truncate_prompt_tokens
    if max_length is not None and max_length < 0:
        max_length = tok_params.max_input_tokens

    if max_length is None or max_length >= len(token_type_ids):
        return token_type_ids
    if max_length == 0:
        return token_type_ids[:0]

    # 确定截断方向：优先使用 tok_params，其次 tokenizer 默认值
```

```
    side = tok_params.truncation_side or (
        tokenizer.truncation_side if tokenizer is not None else None
    )
    if side == 'left':
        return token_type_ids[-max_length:]
    return token_type_ids[:max_length]

# ... 在 _pre_process 方法中的使用（核心变更部分）：
# 先保留 token_type_ids 再 apply_post_tokenization，确保二者同步
token_type_ids = engine_prompt.pop('token_type_ids', None)
tok_params.apply_post_tokenization(self.tokenizer, engine_prompt)

if token_type_ids is not None:
    params = pooling_params.clone()
    # 传入截断后的 token_type_ids 进行压缩
    compressed = compress_token_type_ids(
        _apply_post_tokenization_to_token_type_ids(
            self.tokenizer, tok_params, token_type_ids
        )
    )
    params.extra_kwargs = {
        **(params.extra_kwargs or {}),
        'compressed_token_type_ids': compressed,
    }
    pooling_params_list.append(params)
else:
    pooling_params_list.append(pooling_params)
```

tests/entrypoints/pooling/scoring/test_cross_encoder_offline.py

新增 test_token_type_ids_follow_post_tokenization 单元测试，通过模拟 CrossEncoderIOProcessor 验证左截断和填充场景下 compressed_token_type_ids 的正确性。

```
# tests/entrypoints/pooling/scoring/test_cross_encoder_offline.py

def test_token_type_ids_follow_post_tokenization():
    # 使用 object.__new__ 绕过 __init__ 来构造模拟处理器
    processor = object.__new__(CrossEncoderIOProcessor)
    processor.tokenizer = SimpleNamespace(truncation_side='right', pad_token_id=-1)
    processor.renderer = SimpleNamespace(process_for_engine=lambda prompt, _: prompt)
    processor.model_config = None
    # mock get_score_prompt: 返回 32 个 token，前 16 个类型 0(query)，后 16 个类型
    1(document)
    processor.get_score_prompt = lambda **_: (
        '',
        {
            'prompt_token_ids': list(range(32)),
            'token_type_ids': [0] * 16 + [1] * 16,
        },
    )
```

```
# 场景 1: left-truncation 到 16 tokens -> 只保留 document 段
engine_inputs, pooling_params = processor._pre_process(
    ScoringData(data_1=['query'], data_2=['document']),
    TokenizeParams(
        max_total_tokens=None,
        truncate_prompt_tokens=16,
        truncation_side='left',
    ),
    PoolingParams(task='classify', extra_kwargs={'cache_salt': 'salt'}),
)

assert engine_inputs[0]['prompt_token_ids'] == list(range(16, 32))
# compressed 值应为 0 ( 全部为 document 类型 )
assert pooling_params[0].extra_kwargs == {
    'cache_salt': 'salt',
    'compressed_token_type_ids': 0,
}

# 场景 2: padding 到 40 tokens ( 原有 32 tokens)
engine_inputs, pooling_params = processor._pre_process(
    ScoringData(data_1=['query'], data_2=['document']),
    TokenizeParams(max_total_tokens=None, pad_prompt_tokens=40),
    PoolingParams(task='classify'),
)

assert engine_inputs[0]['prompt_token_ids'] == list(range(32)) + [-1] * 8
# compressed 值应保持原有分割点 16
assert pooling_params[0].extra_kwargs == {'compressed_token_type_ids': 16}
```

评论区精华

该 PR 没有 review 评论或讨论。仅有的 review 是 claude[bot] 的自动评论（指出 fork 的 PR 自动审查被禁用）和 noooop 的批准。

- 暂无高价值评论线程

风险与影响

- 风险：低风险。变更集中在 CrossEncoder 评分路径的 token_type_ids 处理逻辑，不影响其他模型或非 CrossEncoder 的 pooling 请求。测试覆盖了左截断和填充两种场景，并与 HF 参考分数进行了 E2E 对比。_apply_post_tokenization_to_token_type_ids 是新增辅助函数，不修改原有函数签名或行为；_pre_process 中仅调整了 token_type_ids 的处理时机（先 pop 后 apply_post_tokenization），对于没有 token_type_ids 的请求走原路径。
- 影响：
  - 用户影响：使用 CrossEncoder 模型且设置了 truncation_side="left" 的用户将获得正确的评分结果；无截断或默认 right 截断的用户不受影响。
  - 系统影响：无性能影响；额外函数调用很小。

- 团队影响：为后续类似的 post-tokenization 对齐需求提供了可复用的模式（_apply_post_tokenization_to_token_type_ids）。
- 风险标记：核心路径变更（CrossEncoder token type ids 处理）

关联脉络

- PR #47082 [Bugfix][Pooling] Preserve cross encoder extra_kwargs in offline scoring: PR body 说明本 PR 修复了 #47082 未解决的问题——#47082 保留了 extra_kwargs 但未更新截断后的分割边界。