PR #41359 完整报告

vllm-project/vllm

Bump Transformers version to 5.10.4

合并时间：2026-07-07 20:13

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41359

执行摘要

- 一句话：升级 Transformers 依赖至 5.10.4 并修复多模块兼容问题
- 推荐动作：值得精读，尤其是 tokenizer 适配层（vllm/tokenizers/hf.py、vllm/tokenizers/mistral.py）和多模态处理器（vllm/model_executor/models/qwen3_vl.py）的修改，展示了如何在不破坏向后兼容的前提下应对上游 API 重构。

功能与动机

根据 PR 描述，需要将一系列上游修复（如 huggingface/transformers#46456 等）合入，以解决因Transformers版本变更导致的多个测试失败，包括EntrypointsIntegration、Pooling、Multi-Modal Models 等。具体问题涉及 tie_word_embeddings 属性、缺失的 fetch_images/fetch_audio 方法、scope_prefix 引入等。

实现拆解

1. 依赖版本更新：修改 requirements/test/cuda.in、requirements/test/rocm.in、requirements/test/xpu.in、requirements/test/nightly-torch.in 等文件，将Transformers版本从5.5.3提升至5.10.4（中间经历临时git分支引用后锁定正式版本）。
2. Tokenizer 适配：在 vllm/tokenizers/hf.py 中将类型别名 HfTokenizer 从 PreTrainedTokenizer | PreTrainedTokenizerFast 改为 PythonBackend | TokenizersBackend；maybe_make_thread_pool 的类型检查同步更新。在 get_cached_tokenizer 中为 CachedTokenizer 新增 is_fast 属性（安全获取，默认 True）以及 convert_ids_to_tokens 和 convert_tokens_to_string 方法，当底层为 MistralCommonBackend 时委托给新增的 tekken 适配函数。
3. 新增 Mistral Tekken 支持：在 vllm/tokenizers/mistral.py 中新增 mistral_common_tekkenizer 辅助函数提取底层 Tekkenizer，以及 tekken_convert_ids_to_tokens（处理 byte-fallback 乱码）和 tekken_convert_tokens_to_string。在 MistralTokenizer.convert_tokens_to_string 中原有手写逻辑被替换为对 tekken_convert_tokens_to_string 的调用。
4. 多模态处理器调整：在 vllm/model_executor/models/qwen3_vl.py 中，_replace_video_token_placeholders 支持可变长度的 target 序列（从硬编码 3-token 改为通用匹配）；新增 _expands_only_video_token 静态方法检测上游版本行为。MiniCPM-V 4.6 处理器新增 _recompute_cached_prompt_update 以处理 <image_id> 索引重写。多个处理器（如 InternVL、Step3VL、Isaac）继承新的 mixin 基类以适配上游处

理器基类变更。

5. 测试与临时补丁清理：删除 tests/models/multimodal/generation/vlm_utils/model_utils.py 中 qianfan_ocr_hf_model_kwargs 和 qianfan_ocr_patch_hf_runner 函数，因为相关修复已在上游 Transformers 中实现。更新 token 计数测试以匹配新的 tokenizer 行为。

关键文件：

- vllm/tokenizers/hf.py（模块 分词器；类别 source；类型 dependency-wiring；符号 is_fast, convert_ids_to_tokens, convert_tokens_to_string）：核心 tokenizer 适配层：HfTokenizer 类型别名从 PreTrainedTokenizerFast 改为 TokenizersBackend，maybe_make_thread_pool 类型检查同步更新；CachedTokenizer 新增 is_fast 属性和 convert_ids_to_tokens/convert_tokens_to_string 方法以支持 MistralCommonBackend。
- vllm/tokenizers/mistral.py（模块 分词器；类别 source；类型 core-logic；符号 mistral_common_tekkenizer, tekken_convert_ids_to_tokens, tekken_convert_tokens_to_string）：新增 Mistral Tekken 适配函数：mistral_common_tekkenizer 提取底层 Tekkenizer，tekken_convert_ids_to_tokens 和 tekken_convert_tokens_to_string 处理 byte-fallback 转换。原 MistralTokenizer.convert_tokens_to_string 被简化。
- vllm/model_executor/models/qwen3_vl.py（模块 多模态模型；类别 source；类型 data-contract；符号 _expands_only_video_token）：多模态核心模型：placeholder 展开逻辑从硬编码 3-token 改为通用序列匹配，新增 _expands_only_video_token 检测上游版本行为以确保正确展开。
- tests/models/multimodal/generation/vlm_utils/model_utils.py（模块 测试工具；类别 test；类型 test-coverage；符号 qianfan_ocr_hf_model_kwargs, qianfan_ocr_patch_hf_runner）：测试工具：移除了 qianfan_ocr_hf_model_kwargs 和 qianfan_ocr_patch_hf_runner 等临时补丁，因为这些功能已在上游实现。
- vllm/model_executor/models/minicpmv4_6.py（模块 多模态模型；类别 source；类型 data-contract；符号 _recompute_cached_prompt_update）：模型适配：新增 _recompute_cached_prompt_update 方法处理 <image_id> 索引重写，以适配下游多模态处理器的变化。
- requirements/test/cuda.in（模块 依赖配置；类别 config；类型 dependency-upgrade）：依赖入口：从 transformers==5.5.3 改为指向 git 分支（临时）或最终版本 5.10.4，是供应链风险讨论的焦点。

关键符号：maybe_make_thread_pool, get_cached_tokenizer, mistral_common_tekkenizer, tekken_convert_ids_to_tokens, tekken_convert_tokens_to_string, _expands_only_video_token, _replace_video_token_placeholders, _recompute_cached_prompt_update, speech_tokenizer, get_speech_tokenizer

关键源码片段

vllm/tokenizers/hf.py

核心 tokenizer 适配层：HfTokenizer 类型别名从 PreTrainedTokenizerFast 改为 TokenizersBackend，maybe_make_thread_pool 类型检查同步更新；CachedTokenizer 新增 is_fast 属性和 convert_ids_to_tokens/convert_tokens_to_string 方法以支持 MistralCommonBackend。

```
class CachedTokenizer(tokenizer.__class__): # 继承原 tokenizer 类以保留所有原始方法
    @property
    def all_special_ids(self) -> list[int]:
        return tokenizer_all_special_ids

    @property
    def all_special_tokens(self) -> list[str]:
        return tokenizer_all_special_tokens

    @property
    def max_token_id(self) -> int:
        return max_token_id

    @property
    def max_chars_per_token(self) -> int:
        return max_chars_per_token

    @property
    def is_fast(self) -> bool:
        # MistralCommonBackend 不实现 is_fast，所以安全获取并默认 True
        return tokenizer_is_fast

    def convert_ids_to_tokens(self, ids, skip_special_tokens: bool = False):
        # 如果底层是 Mistral Tekken，使用 byte-fallback 感知的转换（返回可能含 bytes 的列表）
        if mistral_tekkenizer is not None:
            from vllm.tokenizers.mistral import tekken_convert_ids_to_tokens
            return tekken_convert_ids_to_tokens(mistral_tekkenizer, ids)
        return super().convert_ids_to_tokens(ids, skip_special_tokens=skip_special_tokens)

    def convert_tokens_to_string(self, tokens: list[str]) -> str:
        if mistral_tekkenizer is not None:
            from vllm.tokenizers.mistral import tekken_convert_tokens_to_string
            return tekken_convert_tokens_to_string(mistral_tekkenizer, tokens)
        return super().convert_tokens_to_string(tokens)
```

vllm/tokenizers/mistral.py

新增 Mistral Tekken 适配函数：mistral_common_tekkenizer 提取底层 Tekkenizer，tekken_convert_ids_to_tokens 和 tekken_convert_tokens_to_string 处理 byte-fallback 转换。原 MistralTokenizer.convert_tokens_to_string 被简化。

```
def mistral_common_tekkenizer(tokenizer: object) -> "Tekkenizer | None":
    """获取 MistralCommonBackend 底层的 `Tekkenizer`，如果存在的话。"""
    mistral = getattr(tokenizer, "tokenizer", None)
    instruct = getattr(mistral, "instruct_tokenizer", None)
```

```
    tekken = getattr(instruct, "tokenizer", None)
    return tekken if isinstance(tekken, Tekkenizer) else None


def tekken_convert_ids_to_tokens(
    tokenizer: "Tekkenizer", ids: Sequence[int]
) -> list[str | bytes]:
    """
    将 IDs 转换为 tokens。对于 byte-fallback 替换导致的乱码（包含 ），
    改用 `id_to_byte_piece` 返回原始 bytes，避免信息丢失。
    """
    tokens: list[str | bytes] = [tokenizer.id_to_piece(i) for i in ids]
    if any("" in t for t in tokens):
        # 遇到替换字符（）时，对非特殊 token 使用 id_to_byte_piece
        tokens = [
            tokenizer.id_to_byte_piece(i, SpecialTokenPolicy.KEEP)
            if i >= tokenizer.num_special_tokens
            else tokenizer.decode([i], SpecialTokenPolicy.KEEP)
            for i in ids
        ]
    return tokens


def tekken_convert_tokens_to_string(
    tokenizer: "Tekkenizer", tokens: Sequence[str | bytes]
) -> str:
    """将 `tekken_convert_ids_to_tokens` 返回的 token 列表重新组装为字符串。"""
    if any(isinstance(t, bytes) for t in tokens):
        # 如果有 bytes 元素，需要 id 来回 decode
        ids = [_tekken_token_to_id(tokenizer, t) for t in tokens]
        return tokenizer.decode(ids, SpecialTokenPolicy.KEEP)
    return "".join(cast(Sequence[str], tokens))
```

评论区精华

依赖安全性：depthfirst-app[bot] 指出 requirements/test/cuda.in 中指向可变 git 分支 v5.10-release 存在供应链风险（强制推送会静默改变安装代码）。作者 hmellor 回应这是临时措施，会在正式发布前改为具体版本。

合并要求：Isotr0py 要求先运行完整 CI 并更新至 transformers==5.10.3 再合并。最终标题显示升级至 5.10.4。

- 使用可变 git 分支作为依赖的供应链风险 (security): 作者确认临时使用，正式发布时改为具体版本。
- 合并前需运行完整 CI 并更新版本 (other): 最终 PR 标题显示升级至 5.10.4 且已合并，说明要求已满足。

风险与影响

- 风险：核心路径变更：vllm/tokenizers/hf.py 中 HfTokenizer 类型别名变更影响整个 tokenizer 使用链；maybe_make_thread_pool 的类型判断从 PreTrainedTokenizerFast 改为 TokenizersBackend，可能导致其他第三方 tokenizer 后端未被正确识别。

多模态处理：Qwen3VL 等模型的 placeholder 展开逻辑依赖上游版本行为，如果版本判断有误可能导致视频 token 展开错误或断言失败。

测试覆盖：虽改动大量测试，但 qianfan_ocr 临时补丁被直接删除，对应测试是否充分需验证。

依赖版本跃升：Transformers 从 5.5.3 到 5.10.4 跨越多个主版本，可能存在未发现的 API 不兼容。

- 影响：用户：无直接功能变化，但修复了部分模型在最新 Transformers 下的兼容性问题，提升稳定性。 系统：Transformers 版本大幅跳跃（5.5.3 -> 5.10.4），未来升级需关注类似 API 变更。 团队：合并后需确保 CI 全绿，后续维护需跟进 Transformers 版本演进。
- 风险标记：核心路径变更，缺少测试覆盖，依赖版本跃升

关联脉络

- 暂无明显关联 PR