

PR #55288 完整报告

vllm-project/vllm

[Bugfix] Fix double BOS in LLM.chat() for multimodal models

合并时间: 2026-09-04 22:23

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/55288>

执行摘要

- 一句话: 修复离线 LLM.chat() 为多模态模型默认添加特殊令牌导致的双重 BOS 问题。
- 推荐动作: 该 PR 值得精读。它清晰地解决了一个影响多模态模型离线 / 在线一致性的实际 bug。核心修复简洁有效, 位于一个关键的集中点 `_preprocess_chat`。最值得关注的设计决策是: 1) 选择在调用入口 (离线 LLM 方法) 而非基础设施层 (渲染器) 调整行为, 以对齐在线 API; 2) 利用字典合并的“覆盖”语义, 在提供安全默认值的同时保持灵活性。全面的测试用例 (包括 mock 验证和真实模型行为比对) 是 bugfix 的典范。

功能与动机

修复 Issue #55197 描述的 Bug: 离线 `LLM.chat()` 对多模态模型渲染 chat 模板后, 使用分词器进行 tokenize 时, `add_special_tokens` 参数默认为 `True` (继承自处理器默认值), 导致任何包含 `bos_token` 输出的模板 (如 Gemma 3、`template_deepseek_vl2.jinja`) 会产生两个 BOS token。而在线聊天 API (`ChatCompletionRequest.add_special_tokens`) 默认值为 `False`, 使得同一对话在离线和在线路径产生不同的 token 序列, 造成结果差异。

实现拆解

1. 入口修改与默认值设置: 在核心预处理函数 `vllm/entrypoints/offline_utils.py` 的 `_preprocess_chat` 方法中, 在构建 `tok_params` 之前, 新增逻辑。首先, 将 `tokenization_kwargs` 字典与 `{'add_special_tokens': False}` 合并, 确保 `add_special_tokens` 默认为 `False`, 但允许调用者传入的 `tokenization_kwargs` (若包含此键) 覆盖默认值。
2. 传递给分词参数: 合并后的 `tokenization_kwargs` 通过 `renderer.default_chat_tok_params.with_kwargs(**tokenization_kwargs)` 传递给分词器参数对象 `tok_params`。此方法是所有离线聊天路径 (`LLM.chat()`, `LLM.enqueue_chat()`) 的集中点, 确保行为一致。
3. 配套测试: 在 `tests/entrypoints/multimodal/llm/test_mm_processor_kwargs.py` 中, 新增了大量测试用例。包括:
 - Mock 级测试: 验证 `_preprocess_chat` 默认调用 `with_kwargs(add_special_tokens=False)`, 以及调用者显式提供 `tokenization_kwargs` 时的覆盖行为。
 - 行为级测试 (`TestChatAddSpecialTokensDefault` 类): 使用真实的 `llava-hf/llava-1.5-7b-hf` 模型配置和 `HfRenderer`, 验证: a) 一个输出 BOS 的模板不再产生双重 BOS; b) 离线 `LLM.chat()` 产生的 token ID 序列与在线 API (`add_special_tokens=False`) 完全一致; c) 显式传入

tokenization_kwargs={'add_special_tokens': True} 能覆盖默认值并产生双重 BOS（符合预期）；d) 原始提示词的 LLM.generate() 路径（_preprocess_cmpl）仍从分词器获取 BOS（保持不变）。

- 所有测试均通过。

关键文件：

- vllm/entrypoints/offline_utils.py（模块 离线入口；类别 source；类型 core-logic；符号 _preprocess_chat）：核心修复所在的源文件，包含离线聊天预处理的入口点 _preprocess_chat。
- tests/entrypoints/multimodal/llm/test_mm_processor_kwargs.py（模块 多模态测试；类别 test；类型 test-coverage；符号 test_preprocess_chat_defaults_add_special_tokens_to_false, test_preprocess_chat_tokenization_kwargs_override_add_special_tokens, llama_llm, TestChatAddSpecialTokensDefault）：新增全面的测试用例，覆盖了修复的核心逻辑和端到端行为，是验证修复有效性和防止回归的关键。

关键符号：_preprocess_chat

关键源码片段

vllm/entrypoints/offline_utils.py

核心修复所在的源文件，包含离线聊天预处理的入口点 _preprocess_chat。

来自 vllm/entrypoints/offline_utils.py 中 _preprocess_chat 方法的核心修复逻辑

该修改确保离线聊天路径的默认分词行为与在线 API 一致

```
chat_params = ChatParams(
    chat_template=chat_template,
    chat_template_content_format=chat_template_content_format,
    chat_template_kwargs=merge_kwargs(
        chat_template_kwargs,
        dict(
            add_generation_prompt=add_generation_prompt,
            continue_final_message=continue_final_message,
            tools=tools,
            tokenize=(
                is_mistral_tokenizer(renderer.tokenizer)
                or self.model_config.enable_prompt_embeds
            ),
        ),
    ),
    mm_processor_kwargs=mm_processor_kwargs,
)
```

注释：原实现直接使用 renderer.default_chat_tok_params.with_kwargs(**(tokenization_kwargs or {}))

这会继承多模态处理器默认的 add_special_tokens=True，导致模板输出的 BOS 被分词器再次添加。

新实现：显式将 add_special_tokens 默认设为 False，仅在调用者显式覆盖时才改变。

此修改只影响 _preprocess_chat，是 LLM.chat() 和 LLM.enqueue_chat() 的共同路径，

确保离线 / 在线 API 行为对齐，且不改变渲染器语义。

```

tokenization_kwargs = {
    "add_special_tokens": False,
    **(tokenization_kwargs or {}), # 调用者传入的参数覆盖默认值
}
tok_params = renderer.default_chat_tok_params.with_kwargs(**tokenization_kwargs)

```

tests/entrypoints/multimodal/llm/test_mm_processor_kwargs.py

新增全面的测试用例，覆盖了修复的核心逻辑和端到端行为，是验证修复有效性和防止回归的关键。

```

# 来自 tests/entrypoints/multimodal/llm/test_mm_processor_kwargs.py 的关键测试类
# 使用真实多模态模型 llava-1.5-7b-hf 验证修复后的行为
@pytest.fixture(scope="module")
def llava_llm():
    # 设置真实模型配置，其处理器默认 add_special_tokens=True
    model_config = ModelConfig(model="llava-hf/llava-1.5-7b-hf", max_model_len=128)
    renderer = HfRenderer(
        VllmConfig(model_config=model_config),
        cached_tokenizer_from_config(model_config),
    )
    # 确保测试基线：处理器默认是 True，这正是导致双重 BOS 的原因
    assert renderer.default_chat_tok_params.add_special_tokens is True
    llm = _make_mock_llm()
    llm.model_config = model_config
    llm.renderer = renderer
    return llm

class TestChatAddSpecialTokensDefault:
    """验证修复后，离线 LLM.chat() 不再为模板输出的 BOS 添加第二个 BOS。"""
    # 模拟一个会输出 BOS 的 chat 模板（如 Gemma 3, DeepSeek VL2 模板）
    BOS_TEMPLATE = "{{ bos_token }}{% for m in messages %}{{ m['content'] }}{% endfor %}"
    MESSAGES = [{"role": "user", "content": "hi"}]

    def _chat_token_ids(self, llm, **kwargs):
        # 调用修复后的 _preprocess_chat 方法
        (engine_input,) = llm._preprocess_chat(
            self.MESSAGES,
            chat_template=self.BOS_TEMPLATE,
            add_generation_prompt=False,
            **kwargs,
        )
        return engine_input["prompt_token_ids"]

    def test_chat_does_not_duplicate_template_bos(self, llava_llm):
        """核心断言：修复后，BOS 应只出现一次。"""
        bos = llava_llm.renderer.tokenizer.bos_token_id
        prompt_token_ids = self._chat_token_ids(llava_llm)
        assert prompt_token_ids[0] == bos # 确保有 BOS
        assert prompt_token_ids.count(bos) == 1 # 确保没有重复

```

```

def test_chat_matches_online_chat_api(self, llava_llm):
    """验证离线 chat 输出与在线 API (add_special_tokens=False) 输出完全相同。"""
    from vllm.renderers.params import ChatParams, TokenizeParams
    online_tok_params = TokenizeParams(
        max_total_tokens=llava_llm.renderer.model_config.max_model_len,
        add_special_tokens=False, # 在线 API 的默认值
    )
    chat_params = ChatParams(
        chat_template=self.BOS_TEMPLATE,
        chat_template_kwargs=dict(tokenize=False, add_generation_prompt=False),
    )
    _, (online_input,) = llava_llm.renderer.render_chat(
        self.MESSAGES, chat_params, online_tok_params
    )
    assert self._chat_token_ids(llava_llm) == online_input["prompt_token_ids"]

def test_explicit_tokenization_kwargs_override_default(self, llava_llm):
    """验证调用者可以显式覆盖默认值以恢复旧行为。"""
    bos = llava_llm.renderer.tokenizer.bos_token_id
    prompt_token_ids = self._chat_token_ids(
        llava_llm, tokenization_kwargs={"add_special_tokens": True}
    )
    assert prompt_token_ids.count(bos) == 2 # 显式覆盖导致两个 BOS

```

评论区精华

核心讨论围绕修复位置展开：

1. 初始方案与反馈：作者最初考虑修改渲染器 `default_chat_tok_params` 的默认值。维护者 DarkLight1337 指出，这假设了所有模型的 chat 模板都会自行输出特殊令牌，建议改为按模型覆盖或自动检测。这促使作者重新思考。
 2. 最终方案决策：DarkLight1337 建议在 `LLM.chat()` 内部设置默认值以匹配在线 API 行为。作者表示同意，并详细解释了为什么选择在更底层的 `_preprocess_chat` 方法中实现：它是所有离线聊天路径（包括 `enqueue_chat`）的集中点，可以一次性修复所有入口，且不会改变渲染器语义或引入按模型维护的成本。
 3. 代码清晰度：DarkLight1337 要求将 `tokenization_kwargs` 的赋值语句拆分以提高可读性，作者进行了相应调整。结论：最终方案被接受，即在 `_preprocess_chat` 中为离线聊天路径默认设置 `add_special_tokens=False`。这被理解为实现离线 / 在线 API 的行为对齐，而非改变渲染器的通用默认值。
- 修复位置：在渲染器层 vs 在 LLM 入口层 (design): 最终决定在 `_preprocess_chat` 中设置 `add_special_tokens` 默认值为 `False`，以实现离线与在线 API 的行为对齐，而不是改变渲染器的通用默认值。
 - 代码可读性优化 (style): 将默认值合并操作拆分为独立的赋值语句，提高了代码可读性。

风险与影响

- 风险：行为变更风险：此修复改变了 `LLM.chat()` 的默认分词行为。对于使用处理器默认 `add_special_tokens=True` 且其 chat 模板不输出 BOS 的模型，离线 `LLM.chat()` 将不再从分词器添加 BOS。这符合在线 API 和 HuggingFace `apply_chat_template` 的标准行为，但可能影响依赖旧行为的用户脚本。PR 作者通过提供 `tokenization_kwargs={'add_special_tokens': True}` 覆盖路径来缓解此风险，并在 PR body 中明确说明了行为变化。回归风险：变更范围集中，仅影响 `_preprocess_chat` 函数。测试覆盖了核心逻辑（默认值、覆盖）和端到端场景（真实模型、与在线 API 比较），能有效防止回归。兼容性风险：对 `LLM.generate()` 等其他入口点无影响。与在线 API 的行为差异得以消除，提升了整体一致性。
- 影响：用户影响：影响所有使用离线 `LLM.chat()` 和 `LLM.enqueue_chat()` 接口的用户，特别是那些使用多模态模型（如 Gemma 3、DeepSeek VL2/OCR）且其 chat 模板包含 `bos_token` 的用户。他们的输出 token 序列将从包含两个 BOS 变为一个 BOS，与在线 API 结果一致，从而修复了生成质量或格式错误。系统影响：统一了离线和在线聊天 API 的分词行为，减少了跨路径的不一致性。变更位于模型推理流水线的前端，影响的是输入 token 化，而非核心计算或调度，因此对性能或系统稳定性无负面影响。团队影响：解决了一个长期存在的、涉及多个模型架构的细微 bug（参见关联的 Issue #38826, #39842, #52304），通过一处集中修改替代了分散的、不完美的按模型覆盖方案，降低了未来的维护负担。
- 风险标记：核心路径变更，离线 / 在线行为一致性

关联脉络

- PR #54886 [Bugfix] Reject tokenizer-less Qwen VL processor init: 同属多模态处理路径的修复，涉及处理器（processor）的初始化验证。本 PR 修复的是处理器默认值在分词阶段导致的行为问题。
- PR #52304 [XPU] Fix BOS added twice by Gemma 4 model after chat template rendering: 直接相关的前期修复。该 PR 尝试为 Gemma 4 模型在 `generate()` 路径中修复类似双重 BOS 问题，但采用的是按模型覆盖 `get_default_tok_params` 的方式，无法解决 `chat()` 路径的问题。本 PR 从入口点统一解决了该类问题。
- PR #16081 Fix `add_special_tokens` not being passed to tokenizer: 该历史 PR 是在在线 API 中引入 `add_special_tokens` 参数处理的原始提交，本 PR 的修复是为了在离线路径上实现与之对齐的行为。