

PR #56415 完整报告

vllm-project/vllm

[Bugfix][Pooling] Restore token limits for offline Jina scoring

合并时间: 2026-09-11 18:26

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56415>

执行摘要

- 一句话: 恢复离线 Jina 评分的 token 截断, 避免超长输入报错
- 推荐动作: 值得精读, 但重点不在 7 行源码, 而在三处: 一是「必须先截断文本再格式化 prompt」的因果 (结构标记与文档索引的绑定关系, 与 #55200 是同一问题的两面); 二是 `_get_token_limits` 的校验契约 (None/0 不截断、负数与超过 `max_model_len` 抛 `ValueError`); 三是新增单测对 1:N 与 N:N 两种输入形态的断言方式, 可直接作为后续改动离线 scoring 请求构造的回归基线。若要继续推进, 建议补一个 CI 可跑的模式级回归 (或至少覆盖在线 / 离线截断结果一致性的对拍), 并把 #55200 的标记校验与本 PR 一起评估合并顺序。

功能与动机

PR body 明确指出: `LLM.score()` 搭配 `jinaai/jina-reranker-v3` 时, 通过 `PoolingParams.extra_kwargs` 传入的 `max_tokens_per_query / max_tokens_per_doc` 被静默忽略, 长输入不截断, 即使设置了限制也会触发模型 `input-length` 检查失败。作者定位到根因是 #47699 把离线预处理迁移到 `request factories` 时漏掉了 Jina 原本从 `bi-encoder` 继承的截断逻辑, 因此「Restore it by reusing the existing token-limit validation and truncation helpers before formatting the query and documents, matching Jina's online path」。

实现拆解

1. 定位入口: 离线评分路径是 `vllm/entrypoints/pooling/scoring/io_processor.py` 中 `JinaRankingIOProcessor.get_request_factory_offline`。在线路径 `get_request_factory_online` 此前已通过代理 `PoolingCompletionRequest` 转发 `truncate_prompt_tokens / truncation_side` (同文件内已有注释说明), 而离线路径只读取了 `chat_template_kwargs`, 完全没碰 token 限制。
2. 恢复限制读取与截断: 在 `scoring_data = ctx.scoring_data` 之前插入 `self._get_token_limits(pooling_params=ctx.pooling_params)`, 拿到 `max_tokens_per_query / max_tokens_per_doc`; 随后 `if max_tokens_per_query > 0 or max_tokens_per_doc > 0`: 才调用 `self._truncate_scoring_data(scoring_data, ...)` 覆盖 `scoring_data`, 再经 `ensure_str` 取出 `queries/docs` 交给 `format_docs_prompts_func`。两个 helper 都来自基类 `PoolingIOProcessor`, 因此非法值 (负数、超过 `max_model_len`) 由既有校验统一抛 `ValueError`, 错误信息带字段名。

3. 保持语义边界：未设置 (None) 或显式 0 的 limit 不触发截断，完全保留现状；截断只作用于 `scoring_data`, `chat_template_kwargs.instruction` 从原有 `extra_kwargs` 读取，因此自定义指令不会被裁掉。1:N (单 query 对多 doc) 与 N:N (query/doc 一一配对) 两种输入形态的格式化分支未改动，只吃到了截断后的文本。
4. 测试配套： `tests/entrypoints/pooling/scoring/test_jina_ranking_io_processor_unit.py` 新增 `offline_processor_and_context` fixture (tokenizers 的 WordLevel + WhitespaceSplit 构造最小 PreTrainedTokenizerFast, SimpleNamespace 伪造 `model_config / renderer`) , 并新增两个用例： `test_offline_truncates_text_before_formatting` 用 6 组参数 (unset / zero / query / doc / both-1-to-n / both-n-to-n) 断言截断后的 `<query>` 与 `<passage id="j">` 文本、 `<embed_token>` 与 `<rerank_token>` 标记计数、以及 instruction 保留； `test_offline_rejects_invalid_token_limits` 对两个字段 `× {-1, 1024}` 断言 `ValueError`。模块 docstring 也从「online request building」改为「request building」。
5. 验证口径：作者在 macOS CPU 上从「8 failed, 3 passed」变为「11 passed」，并在单卡 A10 上用真实权重跑离线端到端 12 组合，长 1:1 输入在双限制为 10 时 prompt 从 567 token 降到 177 token, `max_model_len=256` 下由报错变为成功；pre-commit 与 mypy-3.12 均通过。仓库侧 CI 由维护者 `/ci run` 触发 Buildkite #88292。

关键文件：

- `vllm/entrypoints/pooling/scoring/io_processor.py` (模块 池化评分；类别 source；类型 core-logic；符号 `JinaRankingIOProcessor.get_request_factory_offline`, `_get_token_limits`, `_truncate_scoring_data`, `format_docs_prompts_func`)：核心修复点： `JinaRankingIOProcessor.get_request_factory_offline` 在格式化 prompt 之前恢复 `max_tokens_per_query / max_tokens_per_doc` 的读取与截断，是唯一源码改动。
- `tests/entrypoints/pooling/scoring/test_jina_ranking_io_processor_unit.py` (模块 评分测试；类别 test；类型 test-coverage；符号 `offline_processor_and_context`, `test_offline_truncates_text_before_formatting`, `test_offline_rejects_invalid_token_limits`)：回归防线：新增离线 fixture 与两个参数化用例，锁住「先截断后格式化」的语义、限制组合、输入形态与非法值报错。

关键符号： `JinaRankingIOProcessor.get_request_factory_offline`, `_get_token_limits`, `_truncate_scoring_data`, `format_docs_prompts_func`, `ensure_str`, `offline_processor_and_context`, `test_offline_truncates_text_before_formatting`, `test_offline_rejects_invalid_token_limits`

关键源码片段

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

核心修复点： `JinaRankingIOProcessor.get_request_factory_offline` 在格式化 prompt 之前恢复 `max_tokens_per_query / max_tokens_per_doc` 的读取与截断，是唯一源码改动。

```
def get_request_factory_offline(
    self, ctx: AnyOfflineInputsContext
) -> tuple[RequestFactory, int]:
    assert isinstance(ctx, OfflineScoringInputsContext)
```

```

# 从 PoolingParams.extra_kwargs 读取 max_tokens_per_query /
# max_tokens_per_doc: 未设置按 0 处理, 负数或超过 max_model_len 会
# 在这里直接抛 ValueError (错误信息带字段名), 本次修复复用了既有校验。
max_tokens_per_query, max_tokens_per_doc = self._get_token_limits(
    pooling_params=ctx.pooling_params
)
scoring_data = ctx.scoring_data
# 关键顺序: 必须在 format_docs_prompts_func 之前截断原始文本。
# Jina 的 prompt 由 <query>、<passage id="i">、<lrerank_token!>、
# <lembed_token!> 等结构标记拼接而成, 先格式化再截断有可能切掉标记,
# 导致分数与文档索引错位 (这也是后续 PR#55200 要加校验的动机)。
if max_tokens_per_query > 0 or max_tokens_per_doc > 0:
    scoring_data = self._truncate_scoring_data(
        scoring_data, max_tokens_per_query, max_tokens_per_doc
    )
prompt_extras = ctx.pooling_params.extra_kwargs

queries = self.ensure_str(scoring_data.data_1)
docs = self.ensure_str(scoring_data.data_2)
chat_template_kwargs = (
    prompt_extras.get("chat_template_kwargs") if prompt_extras else None
)
# instruction 不属于被截断的 scoring_data, 因此自定义指令始终保留
instruction = (
    chat_template_kwargs.get("instruction") if chat_template_kwargs else None
)

# 1:N (单 query 对多 doc) 与 N:N (query/doc 一一配对) 两种输入形态
if len(queries) == 1:
    prompts = [
        self.format_docs_prompts_func(
            query=queries[0], docs=docs, instruction=instruction
        )
    ]
else:
    prompts = [
        self.format_docs_prompts_func(
            query=q, docs=[d], instruction=instruction
        )
        for q, d in zip(queries, docs)
    ]

# 其余字段 (chat_template、lora_request、priorities 等) 原样透传 ctx,
# 由基类工厂完成 tokenize 与请求构造。
return PoolingIOProcessor.get_request_factory_offline(
    self,
    OfflineEncodeInputsContext(
        pooling_task=self.pooling_task,
        prompts=prompts,

```

```

        tokenization_kwargs=ctx.tokenization_kwargs,
        # 其余 ctx 字段继续透传
    ),
)

```

tests/entrypoints/pooling/scoring/test_jina_ranking_io_processor_unit.py

回归防线：新增离线 fixture 与两个参数化用例，锁住「先截断后格式化」的语义、限制组合、输入形态与非法值报错。

```

@pytest.fixture
def offline_processor_and_context():
    # 用最小 WordLevel 分词器 + WhitespaceSplit 复现按词切分,
    # 让 max_tokens_per_query / max_tokens_per_doc 的截断结果可逐词断言。
    backend = Tokenizer(models.WordLevel({"[UNK]": 0}, unk_token="[UNK]"))
    backend.pre_tokenizer = pre_tokenizers.WhitespaceSplit()
    proc = JinaRankingIOProcessor.__new__(JinaRankingIOProcessor)
    proc.tokenizer = PreTrainedTokenizerFast(tokenizer_object=backend)
    proc.model_config = SimpleNamespace(max_model_len=1024, is_encoder_decoder=False)
    proc.renderer = SimpleNamespace(
        default_cmpl_tok_params=TokenizeParams(max_total_tokens=1024)
    )
    ctx = OfflineScoringInputsContext(
        pooling_task="token_embed",
        scoring_data=ScoringData(
            data_1=["query alpha beta", "query gamma delta"],
            data_2=["document one two", "document three four"],
        ),
        pooling_params=PoolingParams(
            extra_kwargs={
                "chat_template_kwargs": {"instruction": "Keep this instruction"}
            }
        ),
        tokenization_kwargs=None,
        chat_template=None,
        lora_request=None,
        priorities=None,
    )
    return proc, ctx

```

```

@pytest.mark.parametrize(
    ("query_limit", "doc_limit", "n_queries"),
    [(None, None, 1), (0, 0, 2), (1, None, 1), (None, 2, 1), (1, 2, 1), (1, 2, 2)],
    ids=["unset", "zero", "query", "doc", "both-1-to-n", "both-n-to-n"],
)
def test_offline_truncates_text_before_formatting(
    offline_processor_and_context, query_limit, doc_limit, n_queries
):
    proc, ctx = offline_processor_and_context

```

```

queries = ctx.scoring_data.data_1[:n_queries]
docs = ctx.scoring_data.data_2
ctx.scoring_data = ScoringData(data_1=queries, data_2=docs)
# 只写入非 None 的 limit, 用来验证「未设置」与「显式 0」都等价于不截断
ctx.pooling_params.extra_kwargs.update(
    {
        name: limit
        for name, limit in (
            ("max_tokens_per_query", query_limit),
            ("max_tokens_per_doc", doc_limit),
        )
        if limit is not None
    }
)

factory, num_requests = proc.get_request_factory_offline(ctx)
requests = list(factory())
assert len(requests) == num_requests == n_queries
for i, request in enumerate(requests):
    prompt = request["prompts"]["prompt"]
    query = " ".join(queries[i].split()[: query_limit or None])
    # 标记仍完整 + 文本已裁到 limit 个词, 说明截断发生在格式化之前
    assert f"<query>\n{query}</rerank_token>\n</query>" in prompt
    prompt_docs = docs if n_queries == 1 else [docs[i]]
    for j, doc in enumerate(prompt_docs):
        doc = " ".join(doc.split()[: doc_limit or None])
        assert f"<passage id='{j}'>\n{doc}</lembded_token>\n</passage>" in prompt
    # 标记计数保证 1:N / N:N 下文档索引与分数一一对应
    assert prompt.count("<lembded_token>") == len(prompt_docs)
    assert prompt.count("</rerank_token>") == 1
    assert "<instruct>\nKeep this instruction\n</instruct>" in prompt

```

评论区精华

本次 PR 几乎没有技术讨论: [claude\[bot\]](#) 因 PR 来自 fork 而禁用自动审查, 只留下「A repository maintainer can comment [@claude review](#) to run a one-time review」; 维护者 [noooop](#) 直接 APPROVED 且 review body 为空, inline review comments 为 0; 仓库侧评论仅有 [noooop](#) 的 [/ci run](#) 与 CI 机器人的 Buildkite 触发回执。也就是说, 改动虽然涉及评分正确性, 但截断语义、与 prompt 级 [truncate_prompt_tokens](#) 的先后关系都没有经过第三方公开讨论, 作者是用异常详尽的 PR body (含 before/after 失败用例数与 A10 端到端对照) 来替代 review 论证的。

- 缺少实质性 review 讨论 (other): 无技术异议并被合并, 但截断顺序、截断侧语义以及与 prompt 级 [truncate_prompt_tokens](#) 的关系没有经过第三方公开审查, 作者以详尽 PR body 与 A10 端到端对照替代了 review 论证。

风险与影响

- 风险:

1. 行为变更（可控但需知会）：修复前设置 limit 等同于无效，修复后限制真正生效，依赖旧「设了也不截断」行为的调用方会拿到不同 prompt 与不同分数。作者用「手工截断对照组」验证分数差为 0，说明语义正确，但这是行为契约变化。
2. 依赖基类截断语义：_get_token_limits / _truncate_scoring_data 定义在 PoolingIOProcessor，本 PR 默认其计数与截断侧和 Jina 旧 bi-encoder 预处理一致；而在线路径走的是 truncate_prompt_tokens + truncation_side 的 prompt 级截断，两条路径在极端边界（截断到 1 个 token、子词边界）上是否严格等价，PR 中没有讨论，也没有对拍在线 / 离线输出的测试。
3. 与 #55200 的交互：本 PR 只把截断提前到格式化之前，保证 <query> / <passage id="i"> / <lrerank_tokenl> / <lembed_tokenl> 标记不被破坏；但 prompt 级 truncate_prompt_tokens 仍可能在格式化之后切掉标记，#55200 (open、needs-rebase) 才负责校验这一情况。两者可独立合并，但只合本 PR 时「超大 max_tokens_per_*」组合仍会产生错位评分的可能。
4. 测试 tokenizer 过于简化：单测用 WordLevel + WhitespaceSplit 逐词计数，无法覆盖真实 jina-reranker-v3 tokenizer 的 BPE 子词与特殊 token 边界；作者的 A10 端到端验证没有进 CI，属于一次性人工证据。
5. CI 覆盖缺口：改动没有新增可在 CI 中复现的 GPU/ 模型级回归用例，后续若有人再动 get_request_factory_offline，只有 CPU 单测这一层防护。
6. 低风险面：未设置或 0 时完全不进入截断分支，且改动只在离线 scoring 的请求构造层，不触及模型算术、调度器或 KV 缓存，对其他 pooling 任务与在线 rerank 无影响。 - 影响：用户侧：使用 LLM.score() / 离线 rerank 搭配 jinaai/jina-reranker-v3 并显式设置 max_tokens_per_query / max_tokens_per_doc 的场景由「限制无效 + 可能直接报 input-length 错」变为「按限制截断并可正常完成」；未设置限制的用户无感。系统侧：只影响离线 pooling/scoring 的请求构造，不涉及 GPU 内核、调度与显存，性能影响可忽略（截断发生在 tokenization 之前，通常还会减少后续 tokenize/ 前向的开销）。团队侧：这是 #47699 离线预处理迁移的收尾补丁，再次说明该类「路径迁移时漏掉子类特化逻辑」的重构需要按子类逐项核对；同时也提示 JinaRankingIOProcessor 在线与离线两条路径的截断策略存在长期分叉，值得后续统一。 - 风险标记：核心路径变更，缺少独立 review，依赖基类截断语义，单测 tokenizer 过于简化，模型级验证未进 CI

关联脉络

- PR #47699 (PR body 所述) 将离线预处理迁移到 request factories: 本次修复的直接诱因：该重构把离线预处理移到 request factory，期间遗漏了 Jina 原有的 max_tokens_per_query / max_tokens_per_doc 截断，本 PR 属于其回归补丁。
- PR #55200 [Bugfix][Pooling] Validate truncated Jina ranking prompts: 同一功能线的姊妹修复：本 PR 保证 query/doc 截断发生在格式化之前，#55200 则校验 prompt 级截断后是否仍保留 <lrerank_tokenl> / <lembed_tokenl> 等排名标记，PR body 明确说明两者「can be merged independently」。