

PR #56392 完整报告

vllm-project/vllm

[Frontend] create unified Cohere parser

合并时间: 2026-09-12 16:04

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56392>

执行摘要

- 一句话: 统一 Cohere 解析器, 旧 parser 降级为 shim 修复特殊 token 泄漏
- 推荐动作: 建议精读, 价值在于三点: 一是 DelegatingParser 统一入口 + 薄 shim 的迁移手法, 可复用于其他模型家族; 二是 convert_schema_to_structural_tags 对 xgrammar triggered_tags 的构造方式, 是结构化输出与工具调用约束生成的关键; 三是 tests/parser/cohere/utils.py 的 MockCohereTokenizer 与 token_deltas 设计, 把多字节字符拆分的 U+FFFD 缓冲、多 token chunk 流式行为都纳入测试, 值得在其它 parser 测试中借鉴。

功能与动机

PR body 的 Purpose 写明: 创建统一 `CohereCommandParser` 并将旧的 tool / reasoning parser 转为薄 shim, 采用与 mistral、gpt-oss 相同的做法; 同时修复 Cohere 模型当前解析在用户可见响应中泄漏特殊 token 的问题。代码层面, 旧的 reasoning 与 tool parser 各自持有完整的 `cohere_melody` 状态机与 `convert_schema_to_structural_tags` 实现, 重复约 300 行且容易漂移, 统一入口是消除重复与行为不一致的必要条件。

实现拆解

1. 新增统一解析器 (vllm/parser/cohere_command.py, +687 行):
`CohereCommandParser` 继承 `DelegatingParser`, 一次性带入 `cohere_melody` 的 streaming / unary 两套 `PyFilter`、citations 解析 (`_melody_sources_to_vllm`、`_melody_citations_to_vllm`)、结构化输出改写 (`adjust_request`)。
`MODEL_TO_TAG_STYLE` 以架构名为键 (`Cohere2ForCausalLM`、`Cohere2VisionForConditionalGeneration`、`Cohere2MoeForCausalLM`) 映射到各自的 `CohereTagStyle`, 统一处理 Command A 系模型的 `<ISTART_RESPONSEI>` 与 Command A+ 系的 `<ISTART_TEXTI>` 差异。
2. reasoning parser 降级为约 60 行 shim: vllm/reasoning/cohere_command_reasoning_parser.py 删掉约 660 行, 仅保留 `BaseCohereCommandReasoningParser` 的 token id 初始化、`is_reasoning_end`、`count_reasoning_tokens` 与两个子类的 `melody_preset = "cmd3" / "cmd4"`。保留 `is_reasoning_end` 是因为 structured-output 引擎仍通过它做推理结束门控。
3. tool parser 降级为 18 行 shim: vllm/tool_parsers/cohere_command_tool_parser.py 同样只保留 `melody_preset`, 类名保持不变, 保证 `ParserManager` 按名解析的兼容性。

4. 注册与调用方接线: `vllm/parser/parser_manager.py` 注册新的统一解析器;
`vllm/entrypoints/cohere/serving.py`、`cohere_chat_message.py`、
`vllm/renderers/cohere.py` 调整引用, 把 `POSITION_TO_SOURCE_KEY` 的 `citation` 位置
映射等上下文经 `chat_template_kwargs` 注入统一解析器。
5. 测试重组: 删除 `tests/reasoning/test_cohere_command_reasoning_parser.py` (796 行)
与 `tests/tool_parsers/test_cohere_command_tool_parser.py` (510 行), 在
`tests/parser/cohere/` 下按能力拆为 `test_reasoning.py`、`test_tool_calls.py`、
`test_structural_tags.py`、`test_citations.py`, 公共设施 (`MockCohereTokenizer`、
`make_parser`、`drive_parser`、`stream_parser`、`token_deltas`) 收敛到 `utils.py`。
6. 配套变更: `requirements/test/{cpu,cuda,rocm}.txt` 将 `cohere-melody` 升级到 `>=0.14.0`
; `docs/features/tool_calling.md` 同步更新。PR body 提及用
`CohereLabs/command-a-plus-05-2026-bf16` 与 `CohereLabs/North-Mini-Code-1.0` 做了
集成验证, 未出现特殊 token 泄漏。

关键文件:

- `vllm/parser/cohere_command.py` (模块 解析器; 类别 `source`; 类型 `core-logic`; 符号
`CohereCommandParser`, `CohereTagRegistry`, `CohereTagStyle`, `CohereNormalizedTool`)
: 新增的统一解析器核心文件, 687 行, 收拢全部 Cohere 解析逻辑与 `xgrammar`
`structural_tag` 构造
- `vllm/reasoning/cohere_command_reasoning_parser.py` (模块 推理解析; 类别 `source`;
类型 `dependency-wiring`; 符号 `BaseCohereCommandReasoningParser`,
`CohereCommand3ReasoningParser`, `CohereCommand4ReasoningParser`,
`is_reasoning_end`) : 从 670 行精简到约 60 行, 保留 `is_reasoning_end` 与
`count_reasoning_tokens` 供 `structured-output` 引擎使用
- `vllm/tool_parsers/cohere_command_tool_parser.py` (模块 工具解析; 类别 `source`; 类型
`dependency-wiring`; 符号 `BaseCohereCommandToolParser`,
`CohereCommand3ToolParser`, `CohereCommand4ToolParser`, `melody_preset`) : 从
148 行精简到 18 行, 仅保留 `melody_preset` 声明, 体现 `registry` 兼容的 `shim` 模式
- `vllm/parser/parser_manager.py` (模块 解析注册; 类别 `source`; 类型 `entrypoint`; 符号
`ParserManager`, `_CohereCommandParser`) : 注册新的统一 `CohereCommandParser`,
是入口路由的关键接线点
- `tests/parser/cohere/test_tool_calls.py` (模块 工具测试; 类别 `test`; 类型 `test-coverage`
; 符号 `TestExtractToolCalls`, `TestSpeculativeDecodingMultiTokenDelta`,
`run_streaming`, `run_nonstreaming`) : 新增工具调用测试, 含多 token chunk 下特殊
token 泄漏的回归用例
- `tests/parser/cohere/test_citations.py` (模块 引用测试; 类别 `test`; 类型 `test-coverage`;
符号 `TestMelodySourceResolution`, `TestParserStreamingEndToEnd`,
`test_full_wire_stream_shape`) : 新增 `citation` 解析与 `wire shape` 断言, 覆盖位置映射、
多索引扇出与引文偏移锚定

关键符号: `CohereCommandParser.parse`, `CohereCommandParser.parse_delta`,
`CohereCommandParser.adjust_request`, `convert_schema_to_structural_tags`,
`collect_tool_schema`, `_tool_definitions_to_schema_list`, `_has_effective_tools`, `_add_tag`,

BaseCohereCommandReasoningParser.is_reasoning_end, BaseCohereCommandReasoningParser.count_reasoning_tokens

关键源码片段

vllm/parser/cohere_command.py

新增的统一解析器核心文件，687 行，收拢全部 Cohere 解析逻辑与 xgrammar structural_tag 构造

```
# MODEL_TO_TAG_STYLE 定义每个 Cohere 架构要使用的结构性标签：
# json_tags 是 JSON schema 的包裹对，tools 是工具调用数组的包裹对。
# Command A 系用 <ISTART_RESPONSEI> / <IEND_RESPONSEI>，Command A+ 系
# （Vision / MOE）额外接受 <ISTART_TEXTI> / <IEND_TEXTI>。
COMMAND_A_TOOLS_TAG = CohereTagRegistry(
    trigger="<ISTART_ACTIONI>",
    end="<IEND_ACTIONI>",
)
COMMAND_A_JSON_TAG = CohereTagRegistry(
    trigger="<ISTART_RESPONSEI>",
    end="<IEND_RESPONSEI>",
)
COMMAND_A_PLUS_JSON_TAG = CohereTagRegistry(
    trigger="<ISTART_TEXTI>",
    end="<IEND_TEXTI>",
)

MODEL_TO_TAG_STYLE: dict[str, CohereTagStyle] = {
    "Cohere2ForCausalLM": CohereTagStyle(
        json_tags=(COMMAND_A_JSON_TAG,),
        tools=COMMAND_A_TOOLS_TAG,
    ),
    "Cohere2VisionForConditionalGeneration": CohereTagStyle(
        json_tags=(COMMAND_A_JSON_TAG, COMMAND_A_PLUS_JSON_TAG),
        tools=COMMAND_A_TOOLS_TAG,
    ),
    "Cohere2MoeForCausalLM": CohereTagStyle(
        json_tags=(COMMAND_A_JSON_TAG, COMMAND_A_PLUS_JSON_TAG),
        tools=COMMAND_A_TOOLS_TAG,
    ),
}

def convert_schema_to_structural_tags(
    schema: dict | None = None,
    tools: str | list[Any] | None = None,
    model_architecture: str | None = None,
) -> str | None:
    # 返回 xgrammar 认可的 structural_tag response_format；未知架构或
    # 缺少 model_config 时返回 None，让调用方走原路径。
```

```

if model_architecture is None or model_architecture not in MODEL_TO_TAG_STYLE:
    return None
style = MODEL_TO_TAG_STYLE[model_architecture]

tags: list[dict] = []
triggers: list[str] = []

def _add_tag(tag: CohereTagRegistry, content: dict) -> None:
    # 每个 tag 用 begin / end 标记触发范围, content 限定范围内生成约束;
    # triggers 汇总所有 begin, 供 xgrammar 触发匹配。
    tags.append({"begin": tag.trigger, "content": content, "end": tag.end})
    triggers.append(tag.trigger)

if schema is not None:
    # JSON schema 需要逐个包裹; MOE 同时输出 RESPONSE 与 TEXT 两套
    # 分隔符, 因此 json_tags 是元组, 逐个注册。
    for jt in style.json_tags:
        _add_tag(jt, {"type": "json_schema", "json_schema": schema})

if _has_effective_tools(tools):
    # 工具定义可能来自 JSON 字符串 (poseidon 场景) 或请求体列表
    # (OpenAI Chat / Responses 格式), 先归一化再生成 EBNF grammar。
    tool_schema_list = _tool_definitions_to_schema_list(tools)
    if not tool_schema_list:
        raise ValueError(
            "No valid tool definitions could be parsed from the request for "
            "structural tag conversion."
        )
    tool_grammar = collect_tool_schema(tool_schema_list)
    _add_tag(style.tools, {"type": "grammar", "grammar": tool_grammar})

if not tags:
    return None
return json.dumps(
    {
        "type": "structural_tag",
        "format": {
            "type": "triggered_tags",
            "triggers": triggers,
            "tags": tags,
        },
    }
)

```

[vllm/reasoning/cohere_command_reasoning_parser.py](#)

从 670 行精简到约 60 行, 保留 is_reasoning_end 与 count_reasoning_tokens 供 structured-output 引擎使用

SPDX-License-Identifier: Apache-2.0

```

# SPDX-FileCopyrightText: Copyright contributors to the vLLM project

from __future__ import annotations

from collections.abc import Sequence
from typing import TYPE_CHECKING, ClassVar

from vllm.reasoning import ReasoningParser

if TYPE_CHECKING:
    from vllm.tokenizers import TokenizerLike

class BaseCohereCommandReasoningParser(ReasoningParser):
    """Expose is_reasoning_end for the structured-output engine.
    Everything else runs in
    :class:`vllm.parser.cohere_command.CohereCommandParser`.
    """

    melody_preset: ClassVar[str]

    def __init__(self, tokenizer: TokenizerLike, *args, **kwargs):
        super().__init__(tokenizer, *args, **kwargs)
        # 只缓存推理边界 token 的 id; 真正的解析状态机已上移到
        # CohereCommandParser, 这个类退化为 structured-output 引擎的钩子。
        self.start_token_id = tokenizer.convert_tokens_to_ids("<ISTART_THINKINGI>")
        self.end_token_id = tokenizer.convert_tokens_to_ids("<IEND_THINKINGI>")
        self.chatbot_token_id = tokenizer.convert_tokens_to_ids("<ICHATBOT_TOKENI>")

    @property
    def reasoning_start_str(self) -> str | None:
        return "<ISTART_THINKINGI>"

    @property
    def reasoning_end_str(self) -> str | None:
        return "<IEND_THINKINGI>"

    def is_reasoning_end(self, input_ids: Sequence[int]) -> bool:
        # 从后向前扫描: 遇到 START 时返回是否已见 END; 先遇到 CHATBOT
        # 说明这段历史属于先前轮次, 忽略。这样对完整 prompt / history
        # 也能把推理结束判定限定在最后一个对话轮次内。
        chatbot = self.chatbot_token_id
        start = self.start_token_id
        end = self.end_token_id
        has_end_token = False

        for i in reversed(range(len(input_ids))):
            tid = input_ids[i]
            if tid == start:

```

```

        return has_end_token
    if tid == chatbot:
        return False
    if tid == end:
        has_end_token = True

    return has_end_token

```

```

class CohereCommand3ReasoningParser(BaseCohereCommandReasoningParser):
    melody_preset = "cmd3"

```

```

class CohereCommand4ReasoningParser(BaseCohereCommandReasoningParser):
    melody_preset = "cmd4"

```

vllm/tool_parsers/cohere_command_tool_parser.py

从 148 行精简到 18 行，仅保留 melody_preset 声明，体现 registry 兼容的 shim 模式

```

# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project

```

```

from typing import ClassVar

```

```

from vllm.tool_parsers import ToolParser

```

```

class BaseCohereCommandToolParser(ToolParser):
    # 旧实现在这里维护 melody PyFilter 与流式增量解析；统一后仅保留
    # preset 声明，实例化与解析均由 CohereCommandParser 完成。
    melody_preset: ClassVar[str]

```

```

class CohereCommand3ToolParser(BaseCohereCommandToolParser):
    melody_preset = "cmd3"

```

```

class CohereCommand4ToolParser(BaseCohereCommandToolParser):
    melody_preset = "cmd4"

```

评论区精华

整个 PR 没有任何 review comments，唯一的审核动作是维护者 sfeng33 的快速批准：

“Thank you for the making the migration, LGTM!”。另外 claude[bot] 提示本 PR 来自 fork，自动化 review 被禁用，需维护者用 [@claude review](#) 手动触发；Issue 评论区 5 条均为自动化消息（mergify 文档预览、[/ci run](#)、[/ci retry](#) 与两次 Buildkite 触发），说明合入前 CI 实际跑了两轮。由于没有实质技术交锋，讨论信息量有限，也侧面说明迁移方案本身争议不大。

- 无实质技术讨论，维护者直接批准 (other): 无未解决的技术疑虑；迁移方案获得维护者认可。

风险与影响

- 风险：
 - 解析行为回归（高）：所有 Cohere 前端解析由两套旧实现切换到一个新实现，流式 delta 形状、tool_call_id 挂起缓存、citations 偏移等细节的细微差异都会影响线上输出；缓解靠用例整体迁移与新增 chunk_size 参数化断言。
 - 外部实例化兼容性（中）：CohereCommand3ToolParser / CohereCommand4ToolParser 类名保留但不再具备解析能力，仓库外直接实例化的调用方会拿到只有 melody_preset 的空壳。
 - 依赖门槛（中）：cohere-melody 从 $\geq 0.11.1$ 提升到 $\geq 0.14.0$ ，旧环境 import 即失败；三平台测试 requirements 已同步，生产环境需留意。
 - 架构映射维护（低）：MODEL_TO_TAG_STYLE 是硬编码表，未来新增 Cohere 架构变体时容易遗漏，导致 convert_schema_to_structural_tags 静默返回 None。
- 影响：
 - 用户：Cohere Command A / A+ / MOE / Vision 系列模型的 tool calling、reasoning、citations 与结构化输出行为统一，特殊 token 不再泄漏。
 - 系统：解析逻辑收敛到 vllm/parser/cohere_command.py 单文件，后续修 bug 或加能力只需改一处。
 - 团队：测试目录从按解析器类型组织改为按能力组织；与 mistral / gpt-oss 模式一致，可作为其他模型家族迁移模板。
 - CI：tests/parser/cohere 新旧目录切换，cohere-melody 版本约束三平台同步升级。
 - 风险标记：核心解析路径变更，大量旧解析逻辑删除，cohere-melody 版本门槛提升，外部直接实例化兼容性，架构映射表需持续维护

关联脉络

- 暂无明显关联 PR