PR #30917 完整报告

sgl-project/sglang

Add return_token_ids support to completions and chat completions APIs

合并时间：2026-07-24 05:41

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30917

执行摘要

- 一句话：OpenAI 端点新增 return_token_ids，返回原始 token ids
- 推荐动作：值得精读。核心看点有三个：一是流式场景下累积 / 增量两种模式下 token ids 与文本 delta 的对齐处理（n_prev_token_ids 切片 vs 直接透传），二是顺带修复的增量流式文本二次切片 bug——这是新测试暴露存量缺陷的典型案例，三是协议层用 model_serializer 在 None 时剔除字段以保持默认响应不变的扩展方式。若你有 chat 流式 token ids 需求，可关注作者提出的 " 原始 token 流独立于解析文本流返回 " 的后续方案。

功能与动机

PR body 明确指出：客户端消费 /v1/completions 做代码补全（FIM / tab-completion）时，需要生成的精确 token ids 而不是只能拿到字符串；在客户端对返回文本重新分词会与模型实际采样结果产生漂移（retokenizing the returned text client-side can drift from what the model actually sampled）。同样，/v1/chat/completions 的 agent RL 场景也需要精确 ids。此前 SGLang 的 OpenAI 兼容端点没有该能力，只有 chat 端暴露了 prompt 侧的 return_prompt_token_ids。

实现拆解

实现拆解分五步：

1. 协议层新增字段（python/sglang/srt/entrypoints/openai/protocol.py）：在 CompletionRequest 与 ChatCompletionRequest 上新增 return_token_ids: bool = False；在 CompletionResponseChoice、CompletionResponseStreamChoice、ChatCompletionResponseChoice 上新增可选字段 token_ids 与 prompt_token_ids，并通过 _serialize（model_serializer mode=wrap）在字段为 None 时从响应中剔除，确保未请求时默认响应与 OpenAI 标准格式完全一致。这是整个功能的对外契约。
2. /v1/completions 非流式支持（serving_completions.py）：在 _convert_to_internal_request 中把 request.return_token_ids 直接映射到 GenerateReqInput.return_prompt_token_ids，复用引擎侧已有的 prompt ids 捕获逻辑；在 _build_completion_response 中按请求开关填充 ret_item["output_ids"] 与 ret_item.get("prompt_token_ids")。
3. /v1/completions 流式支持（serving_completions.py）：在 _generate_completion_stream 中新增 n_prev_token_ids 字典按 choice index 记录已发

出数量：非增量模式下 output_ids 是累积的，需要切片 output_ids[n_prev_token_id:] 得到本 chunk 新增 ids；增量模式下 output_ids 本身就是本 chunk 的新增 ids，直接透传。prompt_token_ids 只在每个 choice 的第一个 chunk 上附加。

4. /v1/chat/completions 支持（serving_chat.py）：_convert_to_internal_request 在 stream=true 时对 return_token_ids 抛出明确 ValueError；引擎开关改为 request.return_prompt_token_ids or request.return_token_ids；_build_chat_response 回填解析前原始 output_ids 作为 token_ids。
5. 顺带修复与测试配套：修复 --incremental-streaming-output 下 completions 文本 delta 被按累积偏移二次切片的问题（增量模式下 delta = text 直接使用后端增量）；测试覆盖三个文件：test_serving_completions.py 新增流式 token ids 增量拼接恰好等于完整输出的属性测试（两种流式模式、prompt_token_ids 只在首 chunk）、test_serving_chat.py 参数化流式拒绝与 token ids 回填断言、test_protocol.py 补充 token_ids 序列化省略断言。

关键文件：

- python/sglang/srt/entrypoints/openai/serving_completions.py（模块 补全服务；类别 source；类型 core-logic；符号 _convert_to_internal_request, _generate_completion_stream, _build_completion_response）：核心实现文件：同时承载流式与非流式 token ids 回填、累积 / 增量双模式 chunk delta 处理，以及增量流式文本二次切片 bug 的修复。
- python/sglang/srt/entrypoints/openai/protocol.py（模块 协议层；类别 source；类型 core-logic；符号 CompletionRequest, ChatCompletionRequest, CompletionResponseChoice, CompletionResponseStreamChoice）：对外契约文件：定义请求参数与响应字段，以及 None 即省略的序列化约定，是默认响应保持不变的保证。
- python/sglang/srt/entrypoints/openai/serving_chat.py（模块 对话服务；类别 source；类型 core-logic；符号 _convert_to_internal_request, _build_chat_response）：chat 端点实现：流式组合显式拒绝、引擎开关合并，以及原始采样 ids 回填，保证 token_ids 语义在解析前。
- test/registered/unit/entrypoints/openai/test_serving_completions.py（模块 补全测试；类别 test；类型 test-coverage；符号 test_non_streaming_response, test_streaming_token_ids_deltas_cover_output_exactly, _mock_generate, run_stream）：最关键的测试文件：新增长流式属性测试，验证两种流式模式下每 chunk token ids 增量拼接恰好等于完整输出、文本修复正确性、prompt_token_ids 仅在首 chunk。
- test/registered/unit/entrypoints/openai/test_serving_chat.py（模块 对话测试；类别 test；类型 test-coverage；符号 test_convert_to_internal_request_rejects_stream_token_ids, test_non_streaming_chat_response_returns_requested_token_ids_and_meta_info）：覆盖 chat 端点的流式拒绝（参数化两个开关）与非流式 token_ids 回填断言。
- test/registered/unit/entrypoints/openai/test_protocol.py（模块 协议测试；类别 test；类型 test-coverage；符号 test_prompt_token_ids_and_meta_info_serialization）：验证协议层默认响应不含 token_ids、设置后正确序列化，守护向后兼容约定。

关键符号：_convert_to_internal_request, _generate_completion_stream, _build_completion_response, _build_chat_response, _serialize

关键源码片段

python/sglang/srt/entrypoints/openai/serving_completions.py

核心实现文件：同时承载流式与非流式 token ids 回填、累积 / 增量双模式 chunk delta 处理，以及增量流式文本二次切片 bug 的修复。

```
        # ---- 流式 chunk 的 token_ids 提取 ----
        chunk_token_ids = None
        chunk_prompt_token_ids = None
        if request.return_token_ids:
            output_ids = content["output_ids"]
            if not self.tokenizer_manager.server_args.incremental_streaming_output:
                # 非增量模式下 output_ids 是累积的（例如 [5, 5, 6, 5, 6, 7]），
                # 需要用 n_prev_token_ids 记录上一 chunk 已发出的数量再做切片，
                # 保证每个 chunk 只携带本段新增的 token ids。
                n_prev_token_id = n_prev_token_ids.get(index, 0)
                chunk_token_ids = output_ids[n_prev_token_id:]
                n_prev_token_ids[index] = len(output_ids)
            else:
                # 增量模式下 output_ids 本身就是本 chunk 的新增 ids，直接透传
                chunk_token_ids = output_ids
            if is_first_chunk:
                # prompt_token_ids 只挂在每个 choice 的第一个 chunk 上
                chunk_prompt_token_ids = content.get("prompt_token_ids")

        # ---- 文本 delta 生成（含增量流式修复） ----
        if self.tokenizer_manager.server_args.incremental_streaming_output:
            # 后端在增量模式下返回的 text 已是增量，若仍按累积偏移
            # text[offset:] 二次切片，会把文本重复截断导致损坏（本 PR 顺带修复）
            delta = text
        else:
            delta = text[offset:]
        stream_offsets[index] = len(content["text"])
```

python/sglang/srt/entrypoints/openai/protocol.py

对外契约文件：定义请求参数与响应字段，以及 None 即省略的序列化约定，是默认响应保持不变的保证。

```
class CompletionResponseChoice(BaseModel):
    index: int
    text: str
    logprobs: Optional[LogProbs] = None
    finish_reason: Optional[Literal["stop", "length", "content_filter", "abort"]] = None
    matched_stop: Union[None, int, str] = None
    hidden_states: Optional[object] = None
    # return_token_ids=true 时才填充，否则保持 None 并在序列化时剔除
    token_ids: Optional[List[int]] = None
    prompt_token_ids: Optional[List[int]] = None
```

```
@model_serializer(mode="wrap")
def _serialize(self, handler):
    data = handler(self)
    # 所有可选扩展字段在 None 时从响应中移除，
    # 保证默认响应与 OpenAI 标准格式保持一致
    if self.hidden_states is None:
        data.pop("hidden_states", None)
    if self.token_ids is None:
        data.pop("token_ids", None)
    if self.prompt_token_ids is None:
        data.pop("prompt_token_ids", None)
    return data
```

python/sglang/srt/entrypoints/openai/serving_chat.py

chat 端点实现：流式组合显式拒绝、引擎开关合并，以及原始采样 ids 回填，保证 token_ids 语义在解析前。

```
    if request.stream:
        if request.return_prompt_token_ids:
            raise ValueError(
                "return_prompt_token_ids is not supported with streaming. "
                "Please set stream=false when using return_prompt_token_ids=true."
            )
        if request.return_token_ids:
            # 流式 chat 的 delta 会经过 reasoning/tool-call 解析器，
            # 文本可能被抑制、拆分或改写，raw token ids 无法与解析后的
            # 文本块对齐，因此只在非流式下开放该能力；作者建议后续以
            # 独立的原始 token 流补充（token 流与解析文本流并行返回）
            raise ValueError(
                "return_token_ids is not supported with streaming on "
                "/v1/chat/completions. Please set stream=false when using "
                "return_token_ids=true."
            )

    # ---- _build_chat_response 中回填响应 ----
    # 两个开关任一开启即回传 prompt_token_ids
    choice_prompt_token_ids = (
        ret_item.get("prompt_token_ids")
        if request.return_prompt_token_ids or request.return_token_ids
        else None
    )
    # token_ids 是未经 reasoning/tool-call 解析的原始采样 ids，
    # 调用方需自行处理与解析后 message.content 的对应关系
    choice_token_ids = ret_item["output_ids"] if request.return_token_ids else None
```

评论区精华

核心讨论有两条：

1. shadeMe 询问为什么 chat 端点流式时不能返回 prompt/output token ids（"Out of curiosity - what prevents the chat completions endpoint from returning the prompt/output token IDs when streaming?"）。Jiminator 解释：chat 流式把 delta 送入 reasoning 与 tool-call 解析器，这些解析器会抑制、拆分或改写文本，导致 token ids 无法与每个解析后的文本块可靠对齐；干净的做法是后续把原始 token ids 作为独立于解析文本的流返回，但本 PR 刻意控制在非流式。shadeMe 确认 /v1/completions 流式已覆盖其主场景，无需阻塞。
2. zyzshishui 指出已存在变通方案：non-streaming chat 下用 logprobs=True + return_prompt_token_ids + return_meta_info 可从 meta_info.output_token_logprobs 中提取精确 ids。Jiminator 与评论者都认可该变通方案不使本 PR 冗余：变通方案强制开启 logprob 元数据、不覆盖 /v1/completions、且这些开关在流式下被拒绝。

- chat 流式端点为何不支持 token ids 返回 (design): 本 PR 刻意将 chat 限定为非流式；作者表达了做独立 token 流 follow-up 的意愿，shadeMe 确认 /v1/completions 流式已满足其主场景。
- logprobs+meta_info 变通方案是否使本 PR 冗余 (design): 社区认可变通方案的局限性与本 PR 的增量价值，无代码层面修改。

风险与影响

- 风险：
  1. 依赖引擎响应字段存在：serving_completions.py 与 serving_chat.py 在 return_token_ids=true 时直接读取 content["output_ids"] / ret_item["output_ids"]，若某些引擎路径（如多模态、特殊模型或旧版本 tokenizer-manager）的响应未携带 output_ids 字段，会抛 KeyError；现有测试均用 mock 数据，未覆盖真实引擎全路径。
  2. 增量流式文本行为修复：--incremental-streaming-output 下 /v1/completions 的文本从损坏变为正确，属于修复，但已按旧（错误）行为做基线对齐的上游客户端可能感知到输出变化。
  3. chat token_ids 语义易误用：非流式 chat 的 token_ids 是经过 reasoning/tool-call 解析之前的原始 ids，与 message.content / reasoning_content 不对齐；调用方若按 ids 去逐 token 拼接解析后的文本会得到错误关联，PR body 已说明但缺少面向用户的文档。
  4. 多 choice 流式边界：n_prev_token_ids 按 index 独立跟踪，但测试未覆盖 n>1、prompt 列表等并发多 choice 场景下 chunk 乱序或 index 复用的切片正确性。 - 影响：影响集中在 OpenAI 兼容适配层（protocol.py、serving_completions.py、serving_chat.py），不涉及引擎、调度或 tokenizer 核心路径，因此无推理性能影响。对用户侧，FIM/ 代码补全与 agent RL 客户端可以在不额外开启 logprobs 的情况下拿到精确 token ids，避免二次分词漂移；默认响应零变化，向后兼容。对团队侧，协议层 " None 即省略 " 的序列化模式进一步确立了 SGLang 扩展字段的标准写法，流式下 " 每 chunk 返回增量 ids、prompt ids 只在首 chunk" 的语义可以作为后续端点扩展的参考契约。 - 风险标记：依赖引擎 output_ids 字段存在，增量流式文本行为修复，chat 流式限制需文档化，多 choice 流式边界未覆盖

关联脉络

- PR #34458 [Fix] Make DeepSeek-V4 reasoning and tool-call streaming parsing chunk-invariant: 同属 " 流式输出数据一致性 " 主题：该 PR 让 DSV4 流式解析与 chunk 切分无关，本 PR 则要求流式 token ids 增量与文本 delta 在累积 / 增量两种模式下精确对齐，两者都关注流式场景下文本 /token 层语义的稳定性；本 PR 讨论中提到的 reasoning/tool-call 解析器对文本的改写，正是该主题的延伸。