PR #34627 完整报告

sgl-project/sglang

fix: preserve output logprobs without input logprobs

合并时间：2026-08-19 01:07

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34627

执行摘要

- 一句话：修复 output-only logprobs 被误丢弃的问题
- 推荐动作：值得精读。核心设计决策是“把 input 与 output 两条 logprob 数据流彻底解耦”，并统一增加 None 防御，这种模式可推广到其他按请求维度切分的元数据组装逻辑；同时可对照 Dynamo 侧 PR#12820 理解端到端闭环。

功能与动机

PR body 明确说明：convert_logprob_style() 目前只要 input_token_logprobs_val 为 None 就直接返回，即使存在有效的 output logprobs 也会被一并丢弃，导致 logprobs=true、top_logprobs=0 这类 output-only 请求在 Dynamo 侧拿不到任何 logprob；下游上下文为 ai-dynamo/dynamo 的 PR#12820，Dynamo 需要把 token ID 与 logprob 组合补全 OpenAI 响应的 token 文本与 bytes 字段。

实现拆解

1. 定位根因：convert_logprob_style（python/sglang/srt/managers/tokenizer_manager.py）开头存在 if recv_obj.input_token_logprobs_val is None: return，把“输入侧缺失”当成整条 logprob 处理路径的终止条件，output 侧数据被连带丢弃。
2. 解耦两条数据流：将 input 侧处理改为三重条件（is not None、len > 0、当前请求位非 None）才累积；output 侧独立成分支，仅当 output_token_logprobs_val 及其当前请求位均非 None 时才累积到 state.output_token_logprobs_val 与 state.output_token_logprobs_idx。
3. 保持其余分支不动：top_logprobs_num > 0 时的 top-k 候选累积、input_top_logprobs_val_flat 的 scheduler-flat 三元组缓存以及 token_ids_logprob 分支均保持原逻辑，最小化回归面。
4. 测试配套：在 test/registered/unit/managers/test_flat_raw_top_logprobs.py 中为 _TokenizerManagerStub 补充 convert_logprob_style 引用，并新增 TestTokenizerManagerLogprobs.test_output_logprobs_without_input_logprobs，断言 input 为空、output 为 [(-0.25, 42, None)]、长度 1。

关键文件：

- python/sglang/srt/managers/tokenizer_manager.py（模块 请求处理；类别 source；类型 core-logic；符号 convert_logprob_style）：核心修复文件：convert_logprob_style 移

除提前 return，input 与 output logprobs 各自独立判断，是本次 bug 的根因所在。

- test/registered/unit/managers/test_flat_raw_top_logprobs.py（模块 对数概率；类别 test；类型 test-coverage；符号 TestTokenizerManagerLogprobs, test_output_logprobs_without_input_logprobs）：回归测试载体：新增 TestTokenizerManagerLogprobs 覆盖“只有 output logprobs、没有 input logprobs”的关键场景，并让 Stub 暴露 convert_logprob_style 方法。

关键符号：convert_logprob_style, test_output_logprobs_without_input_logprobs

关键源码片段

python/sglang/srt/managers/tokenizer_manager.py

核心修复文件：convert_logprob_style 移除提前 return，input 与 output logprobs 各自独立判断，是本次 bug 的根因所在。

```
def convert_logprob_style(
    self,
    meta_info: dict,
    state: ReqState,
    top_logprobs_num: int,
    token_ids_logprob: List[int],
    return_text_in_logprobs: bool,
    recv_obj: BatchStrOutput,
    recv_obj_index: int,
):
    # 修复核心：旧实现先判断 input_token_logprobs_val 为 None 就提前 return，
    # 导致 output 侧 logprobs 在 logprobs=true、top_logprobs=0 这类请求中被整体丢弃。
    # 现在 input 与 output 各走独立分支，互不阻塞。
    if (
        recv_obj.input_token_logprobs_val is not None
        and len(recv_obj.input_token_logprobs_val) > 0
        and recv_obj.input_token_logprobs_val[recv_obj_index] is not None
    ):
        # 仅当本请求确实带输入 logprob 数据时才累积到 state。
        state.input_token_logprobs_val.extend(
            recv_obj.input_token_logprobs_val[recv_obj_index]
        )
        state.input_token_logprobs_idx.extend(
            recv_obj.input_token_logprobs_idx[recv_obj_index]
        )
    if (
        recv_obj.output_token_logprobs_val is not None
        and recv_obj.output_token_logprobs_val[recv_obj_index] is not None
    ):
        # 输出 logprob 独立保留：即使输入侧没有数据，生成 token 的概率
        # 仍可透传给下游（如 Dynamo），由其按 token_id 补全文本与 bytes。
        state.output_token_logprobs_val.extend(
            recv_obj.output_token_logprobs_val[recv_obj_index]
```

```
            )
            state.output_token_logprobs_idx.extend(
                recv_obj.output_token_logprobs_idx[recv_obj_index]
            )

        # 以下分支保持原逻辑：仅在请求 top-k 候选时处理 top logprobs。
        if top_logprobs_num > 0:
            if len(recv_obj.input_top_logprobs_val) > 0:
                state.input_top_logprobs_val.extend(
                    recv_obj.input_top_logprobs_val[recv_obj_index]
                )
                state.input_top_logprobs_idx.extend(
                    recv_obj.input_top_logprobs_idx[recv_obj_index]
                )
            if (
                recv_obj.input_top_logprobs_val_flat is not None
                and recv_obj.input_top_logprobs_val_flat[recv_obj_index] is not None
            ):
                # scheduler-flat 路径：直接缓存数组三元组，避免嵌套行开销。
                state.input_top_logprobs_scheduler_flat = (
                    recv_obj.input_top_logprobs_val_flat[recv_obj_index],
                    recv_obj.input_top_logprobs_idx_flat[recv_obj_index],
                    recv_obj.input_top_logprobs_flat_null_prefix[recv_obj_index],
                )
            state.output_top_logprobs_val.extend(
                recv_obj.output_top_logprobs_val[recv_obj_index]
            )
            state.output_top_logprobs_idx.extend(
                recv_obj.output_top_logprobs_idx[recv_obj_index]
            )

        if token_ids_logprob is not None:
            # 后续 token_ids 分支保持原实现不变，与本次修复点无关。
            pass
```

test/registered/unit/managers/test_flat_raw_top_logprobs.py

回归测试载体：新增 TestTokenizerManagerLogprobs 覆盖“只有 output logprobs、没有 input logprobs”的关键场景，并让 Stub 暴露 convert_logprob_style 方法。

```
class TestTokenizerManagerLogprobs(CustomTestCase):
    # 回归测试：只请求输出 logprobs、不请求输入 logprobs 的场景。
    # 当 recv_obj.input_token_logprobs_val 为 None 时，旧实现会提前 return，
    # 导致 output 侧 (-0.25, 42) 这一条 logprob 丢失。
    def test_output_logprobs_without_input_logprobs(self):
        state = _make_state(return_logprob=True, top_logprobs_num=0)
        recv_obj = SimpleNamespace(
            input_token_logprobs_val=None,
            input_token_logprobs_idx=None,
            output_token_logprobs_val=[[-0.25]],
```

```
        output_token_logprobs_idx=[[42]],
    )
    meta_info = {}

    _TokenizerManagerStub().convert_logprob_style(
        meta_info,
        state,
        top_logprobs_num=0,
        token_ids_logprob=None,
        return_text_in_logprobs=False,
        recv_obj=recv_obj,
        recv_obj_index=0,
    )

    self.assertEqual(meta_info["input_token_logprobs"], [])
    # 关键断言：输出 logprob 必须被保留为 (logprob, token_id, 文本) 三元组。
    self.assertEqual(meta_info["output_token_logprobs"], [(-0.25, 42, None)])
    self.assertEqual(meta_info["output_token_logprobs_length"], 1)
```

评论区精华

唯一的 review 线程集中在测试文件归属上：审阅者 ishandhanani 认为这个回归测试可能很重要，询问是否应放在 tokenizer 相关测试文件里（原先是新增的 test_tokenizer_manager_logprobs.py）；作者 jain-ria 随即回复已移动到 test/registered/unit/managers/test_flat_raw_top_logprobs.py。讨论已解决，无未决技术争议。

- 回归测试文件的放置位置 (testing): 作者 jain-ria 回复已移动到 test/registered/unit/managers/test_flat_raw_top_logprobs.py，与现有 logprobs 测试聚合。

风险与影响

- 风险：
  1. 核心路径变更：convert_logprob_style 是 TokenizerManager 组装 logprob 元数据的必经路径，控制流从“提前 return”改为“逐项条件判断”，虽然对 input 侧语义等价，仍需关注既有 input-only 请求的行为是否完全不变。
  2. None 分支行为变化：旧代码在 output_token_logprobs_val 为 None 时会抛 AttributeError，新代码改为静默跳过，可能掩盖上游数据异常，需要观察日志或增加告警。
  3. 覆盖范围有限：本仓库只增加了单元级回归，缺少对完整生成链路（如 OpenAI API 嵌套 logprobs 结构、流式输出）的端到端验证，主要依赖 Dynamo 侧的外部 e2e 矩阵兜底。 - 影响：影响面集中在 logprob 返回链路：Dynamo 等下游消费方在 output-only 请求时能获得完整输出 logprobs 元数据；对模型 forward、采样、KV cache 等计算路径零改动。对团队而言，这是一个低风险高价值的契约修复，改动量小（源码 +12/-10），但补上了 Python Engine API 与下游推理框架之间的关键断点。 - 风险标记：核心服

务路径变更，None 分支行为变化，缺少端到端 logprobs 回归

关联脉络

- PR #35225 refactor: rename chat response token IDs: 同一 API 响应契约演进线：该 PR 调整 OpenAI chat 响应的 token_ids 字段，本 PR 修复 logprobs 字段的完整性，两者共同影响下游对生成结果元数据的解析。