PR #32078 完整报告

sgl-project/sglang

[feat] Opt-in flat response format for prompt top logprobs

合并时间：2026-07-27 14:44

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32078

执行摘要

- 一句话：新增 opt-in 扁平 top logprobs 响应格式，序列化性能提升 1.9 倍
- 推荐动作：值得精读。该 PR 以极小侵入性实现了显著的序列化性能提升，设计决策清晰（opt-in、null_prefix 处理、缓存复用），且与后续优化形成递进路径。可作为服务端响应格式优化的参考案例，尤其适合对序列化开销敏感的团队借鉴。

功能与动机

大量使用 prompt top logprobs 的工作负载（RL teacher、蒸馏、离线评分）中，响应序列化占据请求延迟的很大比例。嵌套格式每位置每 topk 生成小列表，json.dumps 类型分派开销大，且 null 文本槽位对批量消费者无用。两个长同质数组消除了所有三个问题，客户端可直接用 np.asarray(...).reshape(shape) 消费。PR body 提供的基准测试（32768 位置、k=2）显示：嵌套格式往返 102.7ms，扁平格式 55.1ms（1.9 倍），payload 大小从 2,417,567 字节降至 1,827,898 字节。

实现拆解

1. 请求参数扩展（io_struct.py）：在 GenerateReqInput 类中添加 return_flat_raw_top_logprobs: bool = False 字段。在 normalize_batch_and_arguments 的 _validate_inputs 中增加检查：若开启扁平格式同时设置了 multi_item_delimiter_indices，则抛出 ValueError，因为多条目评分会产生不连续的 top logprobs 行，无法用 (shape, null_prefix) 表示。此外，在 __getitem__ 中传播该标志，确保 batch 拆分后每个子请求也携带该设置。
2. 状态缓存（tokenizer_manager.py）：在 ReqState 数据类中添加两个缓存字段 input_top_logprobs_flat_fields: Optional[Dict] 和 input_top_logprobs_flat_num_rows: int（默认 -1），用于跨预填充 chunk 复用扁平化结果，避免 streaming decode chunk 重复构建。
3. 扁平化构建函数（tokenizer_manager.py，新增模块级函数 _build_flat_input_top_logprobs_fields）：接收 input_top_logprobs_val、input_top_logprobs_idx 列表和 top_logprobs_num。先扫描前导 null 行（None 或空）计入 null_prefix，然后提取有效行；校验有效行的长度等于 k（第一个有效行长度或 top_logprobs_num 提供的兜底值），若出现非 null 但长度不等的行则抛出 ValueError，因为这种不规则模式无法通过 (shape, null_prefix) 映射。最终生成四个字段：input_top_logprobs_val_flat（一维浮点列表）、input_top_logprobs_idx_flat（一维整数

列表）、input_top_logprobs_shape（[rows, k]）、input_top_logprobs_null_prefix（前导无效位置数）。

4. 主路径集成（tokenizer_manager.py，修改 TokenizerManager.add_logprob_to_meta_info 方法）：在原有嵌套路径分支旁新增条件：当 state.obj.return_flat_raw_top_logprobs 为 True 时，调用 _build_flat_input_top_logprobs_fields 构建扁平字段，将其直接写入 meta_info，并 跳过对 state.input_top_logprobs（分割化的 detokenize 版本）的后续更新，因为扁平格式不需要文本解码。构建的结果缓存在 state.input_top_logprobs_flat_fields，仅当累积更多预填充 chunk 导致行数增加时才重建。

5.单元测试（test/registered/unit/managers/test_flat_raw_top_logprobs.py，新增306行）：通过 _TokenizerManagerStub 代理真实管理器的 add_logprob_to_meta_info 方法，避免全量初始化。覆盖场景包括：标志默认关闭且有效、批量传播、多条目拒绝；扁平 vs 嵌套数值等价性及索引公式重建（(i - null_prefix) * k + j）；全 null 行返回 shape [0, k]；不规则 null（非前缀空行）被 builder 拒绝；还有环境门控的序列化基准测试。

关键文件：

- python/sglang/srt/managers/tokenizer_manager.py（模块 管理器；类别 source；类型 core-logic；符号 _build_flat_input_top_logprobs_fields, add_logprob_to_meta_info）：核心实现文件：新增 _build_flat_input_top_logprobs_fields 函数负责将嵌套行拍平为同质数组并计算元信息；修改 add_logprob_to_meta_info 方法根据 opt-in 标志选择扁平或嵌套路径；在 ReqState 中添加缓存字段以复用跨 chunk 的扁平化结果。
- python/sglang/srt/managers/io_struct.py（模块 请求定义；类别 source；类型 core-logic；符号 GenerateReqInput.return_flat_raw_top_logprobs, GenerateReqInput._validate_inputs）：请求结构体定义：添加 return_flat_raw_top_logprobs 字段及其默认值，并在 _validate_inputs 中增加与多条目评分的不兼容检查，确保不合法的配置在早期即被拒绝。
- test/registered/unit/managers/test_flat_raw_top_logprobs.py（模块 单元测试；类别 test；类型 test-coverage；符号 _TokenizerManagerStub, _make_state, _add_logprob_meta_info, TestFlatRawTopLogprobsValidation）：完整的单元测试套件（306 行），通过 _TokenizerManagerStub 模拟 TokenizerManager 的 logprob 方法，覆盖标志验证、批量传播、多条目拒绝、扁平与嵌套数值等价、全 null 行处理、不规则行拒绝等场景，并包含环境门控基准测试，确保实现正确性。

关键符号：_build_flat_input_top_logprobs_fields, TokenizerManager.add_logprob_to_meta_info, GenerateReqInput._validate_inputs

关键源码片段

python/sglang/srt/managers/tokenizer_manager.py

核心实现文件：新增 _build_flat_input_top_logprobs_fields 函数负责将嵌套行拍平为同质数组并计算元信息；修改 add_logprob_to_meta_info 方法根据 opt-in 标志选择扁平或嵌套路径；在 ReqState 中添加缓存字段以复用跨 chunk 的扁平化结果。

```
def _build_flat_input_top_logprobs_fields(     input_top_logprobs_val: List[Optional[List[float]]],     input_top_logprobs_idx: List[Optional[List[int]]],
```

top_logprobs_num: int,) -> Dict[str, Any]: """构建扁平化 prompt top logprob 响应字段。 `input_top_logprobs_shape` 是字面量 `[rows, k]` 数组形状。 `input_top_logprobs_null_prefix` 计数前导无效位置（无 top logprobs 的位置）， 这些位置 在数组之前，有效位置 i 的 entry j 位于 `flat[(i - null_prefix) * k + j]`。 """ num_rows = len(input_top_logprobs_val) null_prefix = 0 # 扫描前导空行（None 或空列表），计 入 null_prefix while null_prefix < num_rows and not input_top_logprobs_val[null_prefix]: null_prefix += 1 val_rows = input_top_logprobs_val[null_prefix:] idx_rows = input_top_logprobs_idx[null_prefix:] # k 取第一个有效行长度，若全空则 fallback 到 top_logprobs_num k = len(val_rows[0]) if val_rows else top_logprobs_num for offset, row in enumerate(val_rows): if row is None or len(row) != k: # 不规则行（非前导空行、长度不一致）无法用 (shape, null_prefix) 表示 raise ValueError("return_flat_raw_top_logprobs requires rectangular top logprob " f"rows with nulls only in the leading prefix; row{null_prefix+offset}" f"has{NoneifrowisNoneelselen(row)}entries (expected{k}).") fields: Dict[str, Any] = {} fields["input_top_logprobs_val_flat"] = [v for row in val_rows for v in row] fields["input_top_logprobs_idx_flat"] = [i for row in idx_rows for i in row] fields["input_top_logprobs_shape"] = [len(val_rows), k] fields["input_top_logprobs_null_prefix"] = null_prefix return fields # 在 add_logprob_to_meta_info 中，约第 2264 行附近 if top_logprobs_num > 0: use_flat = state.obj.return_flat_raw_top_logprobs # type: ignore[union-attr] if use_flat: # 扁平路径：跳过 detokenize，直接构建扁平字段并缓存 if state.input_top_logprobs_flat_fields is None or \ len(state.input_top_logprobs_val) != state.input_top_logprobs_flat_num_rows: state.input_top_logprobs_flat_fields = _build_flat_input_top_logprobs_fields(state.input_top_logprobs_val, state.input_top_logprobs_idx, top_logprobs_num,) state.input_top_logprobs_flat_num_rows = len(state.input_top_logprobs_val) meta_info.update(state.input_top_logprobs_flat_f ields) else: # 嵌套路径（原逻辑）：detokenize 后写入 input_top_logprobs if len(state.input_top_logprobs_val) > len(state.input_top_logprobs): state.input_top_logprobs.extend(self.detokenize_top_logprobs_tokens(state.input_top_logprobs_val[len(state.input_top_logprobs):], state.input_top_logprobs_idx[len(state.input_top_logprobs):], return_text_in_logprobs,)) meta_info["input_top_logprobs"] = state.input_top_logprobs

test/registered/unit/managers/test_flat_raw_top_logprobs.py

完整的单元测试套件（306 行），通过 _TokenizerManagerStub 模拟 TokenizerManager 的 logprob 方法，覆盖标志验证、批量传播、多条目拒绝、扁平与嵌套数值等价、全 null 行处理、不规则行拒绝等场景，并包含环境门控基准测试，确保实现正确性。

```
class TestFlatAssembly(CustomTestCase):
    """验证 _build_flat_input_top_logprobs_fields 的正确性。"""

    def test_flat_matches_nested_rows(self):
```

```
        # 模拟 4 个 prompt 位置，前 1 个为空，后 3 个各含 k=2 个 top logprobs
        val_rows = [None, [-0.1, -2.5], [-0.3, -1.5], [-0.05, -4.0]]
        idx_rows = [None, [11, 22], [33, 44], [55, 66]]
        fields = _build_flat_input_top_logprobs_fields(val_rows, idx_rows, top_logprobs_num=2)

        # 验证元信息
        self.assertEqual(fields["input_top_logprobs_shape"], [3, 2]) # 3 个有效行
        self.assertEqual(fields["input_top_logprobs_null_prefix"], 1) # 前 1 个空行

        # 验证扁平数组内容：跳过前导空行，依次连接有效行
        expected_vals = [-0.1, -2.5, -0.3, -1.5, -0.05, -4.0]
        expected_idxs = [11, 22, 33, 44, 55, 66]
        self.assertEqual(fields["input_top_logprobs_val_flat"], expected_vals)
        self.assertEqual(fields["input_top_logprobs_idx_flat"], expected_idxs)

        # 验证索引公式：位置 i 的 entry j 位于 flat[(i - null_prefix) * k + j]
        rows, k = fields["input_top_logprobs_shape"]
        null_prefix = fields["input_top_logprobs_null_prefix"]
        flat_val = fields["input_top_logprobs_val_flat"]
        self.assertEqual(len(flat_val), rows * k)
        for i in range(null_prefix, null_prefix + rows):
            start = (i - null_prefix) * k
            self.assertEqual(flat_val[start: start + k], val_rows[i])

    def test_all_null_rows(self):
        # 所有行均为 null，应返回 shape [0, k] 和空数组
        fields = _build_flat_input_top_logprobs_fields([None], [None], top_logprobs_num=2)
        self.assertEqual(fields["input_top_logprobs_shape"], [0, 2])
        self.assertEqual(fields["input_top_logprobs_null_prefix"], 1)
        self.assertEqual(fields["input_top_logprobs_val_flat"], [])
        self.assertEqual(fields["input_top_logprobs_idx_flat"], [])

    def test_rejects_null_row_after_prefix(self):
        # 非前导空行（如第 3 行为 None）应该被拒绝
        with self.assertRaisesRegex(ValueError, "leading prefix"):
            _build_flat_input_top_logprobs_fields(
                [None, [-0.1, -2.5], None, [-0.3, -1.5]],
                [None, [11, 22], None, [33, 44]],
                top_logprobs_num=2,
            )
```

评论区精华

该 PR 没有收到实质性 review comments，仅由维护者 Qiaolin-Yu 直接批准合并。主要设计决策（opt-in 扁平格式、null_prefix 表示、不支持多条目、后续 base64 编码）由作者在 PR body 中详细阐述，未引发额外争论。

- 无实质讨论 (other): 直接合并

风险与影响

- 风险：
  1. 仅覆盖 prompt 侧：新标志仅影响 input_top_logprobs（prompt 侧），output 侧保持嵌套格式，可能让用户误以为全面扁平化。需在文档中明确。
  2. 多条目评分不兼容：若用户同时启用 return_flat_raw_top_logprobs 和多条目，会在验证阶段抛出 ValueError，不会静默产生错误输出。
  3. 消费者依赖：扁平字段不包含文本（text），要求消费者已具备 token 解码能力或不需要文本。若消费者需要文本，需继续使用嵌套格式。
  4. 默认行为不变：标志默认 False，旧客户端不受影响，升级安全。
  5. 测试覆盖完善：单元测试覆盖了正常路径、边界条件和不合法输入，回归风险较低。但缺少端到端集成测试（需后端运行）。 - 影响：用户：启用 return_flat_raw_top_logprobs 的请求，响应序列化速度提升约 2 倍（扁平）乃至 7.8 倍（配合 base64），payload 减少 30%–65%，对于高频 top logprobs 消费工作负载（RL 训练、离线评估）显著改善吞吐和延迟。 系统：默认无影响；新增字段仅在一次函数调用中计算，逻辑轻量。 团队：代码增量小（3 文件 +407/-9），维护成本低；设计与后续 base64 编码（#31960）直接兼容，为渐进优化留出空间。
- 风险标记：opt-in 特性，多条目不兼容，输入侧仅 prompt

关联脉络

- PR #31960 [feat] Add base64 binary encoding for flat top logprobs: stacked follow-up：在扁平格式基础上进一步减少 payload 大小，使往返时间从 55ms 降至 13ms。PR body 中明确提及。
- PR #31958 [perf] Companion performance improvement for prompt top logprobs: 配套性能优化 PR，与本次扁平化协同提升整体效率。PR body 中提及。