PR #6120 完整报告

verl-project/verl

[sglang] feat: Patch sglang to support on-policy distillation teacher

合并时间：2026-04-23 16:51

原文链接：http://prhub.com.cn/verl-project/verl/pull/6120

执行摘要

- 一句话：SGLang 引擎支持在线策略蒸馏教师推理
- 推荐动作：建议精读此 PR，尤其关注 _extract_prompt_logprobs_sglang 函数的设计——它清晰地展示了如何将 SGLang 的 logprob 输出格式对齐 vLLM 接口，是引擎适配的典范。值得关注的设计决策：采用硬断言而非防御性填充，在测试充分的前提下可保证数据质量，但生产环境中风险较高。建议后续添加单元测试覆盖异常情况。

功能与动机

在线策略蒸馏目前仅支持 vLLM 作为教师推理引擎，为提供更多引擎选择，需要增加对 SGLang 的支持。PR body 明确说明动机："On-policy distillation currently only supports vLLM as teacher inference engine. This MR adds support for SGlang."

实现拆解

1. 配置验证扩展（verl/workers/config/distillation.py）：在 DistillationTeacherModelConfig._validate_topk_logprobs 的 match 语句中新增 case "sglang" 分支，直接 pass。原因是 SGLang 的 top_logprobs_num 是逐请求参数，无需引擎级启动上限对齐（不像 vLLM 的 max_logprobs）。
2. 新增 logprob 提取函数（verl/workers/rollout/sglang_rollout/async_sglang_server.py）：定义 _extract_prompt_logprobs_sglang 函数，将 SGLang 返回的 input_token_logprobs 和 input_top_logprobs 重塑为与 vLLMextract_prompt_logprobs 相同的输出合约——填充 result_dict 中的 prompt_ids 和 prompt_logprobs 列表，形状为 [sequence_length, max(num_prompt_logprobs, 1)]。

```
# verl/workers/rollout/sglang_rollout/async_sglang_server.py
def _extract_prompt_logprobs_sglang(
    meta_info: dict,
    num_prompt_logprobs: int,
    sequence_length: int,
    result_dict: dict[str, list],
) -> None:
    # 获取 SGLang 的 input_token_logprobs 列表，每个元素为 (logprob, token_id, _)
    input_token_logprobs = meta_info.get("input_token_logprobs") or []
    # 如果需要 top-k，则获取 input_top_logprobs
    if num_prompt_logprobs > 0:
```

```
    input_top_logprobs = meta_info.get("input_top_logprobs") or []
prompt_ids_ls: list[list[int]] = []
prompt_logprobs_ls: list[list[float]] = []
# 跳过位置 0（logprob=None，无预测上下文），与 vLLM 约定一致
for position in range(1, len(input_token_logprobs)):
    if num_prompt_logprobs == 0:
        logprob, token_id, _ = input_token_logprobs[position]
        prompt_ids_ls.append([int(token_id)])
        prompt_logprobs_ls.append([float(logprob)])
    else:
        top_entries = input_top_logprobs[position]
        # SGLang 按排名返回最佳优先，保持此顺序以匹配 vLLM 提取器的 rank-1 槽位
        ids = [int(tok_id) for _, tok_id, _ in top_entries]
        logprobs = [float(logprob) for logprob, _, _ in top_entries]
        assert len(ids) == num_prompt_logprobs, (
            f"SGLang returned {len(ids)} top logprobs at position {position}, "
            f"expected {num_prompt_logprobs}."
        )
        prompt_ids_ls.append(ids)
        prompt_logprobs_ls.append(logprobs)
# 追加一行填充数据，使总长度等于 sequence_length，匹配 vLLM 约定
pad_width = max(num_prompt_logprobs, 1)
prompt_ids_ls.append([0] * pad_width)
prompt_logprobs_ls.append([0.0] * pad_width)
# 最终长度断言，确保与消费者期望一致
assert len(prompt_ids_ls) == sequence_length, (
    f"SGLang prompt_logprobs length ({len(prompt_ids_ls)}) does not match "
    f"sequence length ({sequence_length}); check logprob_start_len=0 invariant."
)
result_dict["prompt_ids"] = prompt_ids_ls
result_dict["prompt_logprobs"] = prompt_logprobs_ls
```

1.集成到SGLang生成请求（verl/workers/rollout/sglang_rollout/async_sglang_server.py）：在 SGLangHttpServer.generate 方法中，当 sampling_params 包含 prompt_logprobs 时，将其转换为 SGLang 的 return_logprob + logprob_start_len=0 + top_logprobs_num 参数，调用生成后在 meta_info 中提取 input_token_logprobs，并调用 _extract_prompt_logprobs_sglang 填充结果字典。

```
# 在 async_sglang_server.py 的 SGLangHttpServer.generate 方法中（约第 410 行）
# 解析 prompt_logprobs 参数（来自蒸馏教师配置）
prompt_logprobs = sampling_params.pop("prompt_logprobs", None)
if prompt_logprobs is not None:
    # 设置 SGLang 的 logprob 参数
    sampling_params["return_logprob"] = True
    sampling_params["logprob_start_len"] = 0
    sampling_params["top_logprobs_num"] = prompt_logprobs
    # ... 调用生成后 ...
    # 在获取 meta_info 后，如果启用了 prompt_logprobs，则提取并填充 result_dict
    if prompt_logprobs is not None:
```

```
        _extract_prompt_logprobs_sglang(
            meta_info, prompt_logprobs, sequence_length, result_dict
        )
```

1. 测试配套：PR 说明中作者提到回测了 examples/on_policy_distillation_trainer/run_qwen_gsm8k.sh 和 run_qwen_gsm8k_megatron.sh，损失 / 奖励 / 验证分数与主分支大体一致，并附上了 FSDP 和 Megatron 的实验曲线对比图。但本次提交中未包含新的单元测试或 CI 测试文件。

关键文件：

- verl/workers/rollout/sglang_rollout/async_sglang_server.py（模块 推理引擎；类别 source；类型 core-logic；符号 _extract_prompt_logprobs_sglang）：核心变更文件，新增 _extract_prompt_logprobs_sglang 函数和 generate 方法中的 logprob 提取逻辑，是实现 SGLang 教师支持的主要入口。
- verl/workers/config/distillation.py（模块 配置层；类别 source；类型 core-logic）：在 _validate_topk_logprobs 方法中添加 SGLang 引擎分支，允许该引擎通过配置验证，是实现 SGLang 教师支持的配置侧变更。

关键符号：_extract_prompt_logprobs_sglang

关键源码片段

verl/workers/rollout/sglang_rollout/async_sglang_server.py

核心变更文件，新增 _extract_prompt_logprobs_sglang 函数和 generate 方法中的 logprob 提取逻辑，是实现 SGLang 教师支持的主要入口。

```
# verl/workers/rollout/sglang_rollout/async_sglang_server.py ( 新增函数 )

def _extract_prompt_logprobs_sglang(
    meta_info: dict,
    num_prompt_logprobs: int,
    sequence_length: int,
    result_dict: dict[str, list],
) -> None:
    # 从 meta_info 中获取 SGLang 返回的 input_token_logprobs 列表
    input_token_logprobs = meta_info.get("input_token_logprobs") or []
    # 如果请求了 top-k logprobs，则获取 input_top_logprobs
    if num_prompt_logprobs > 0:
        input_top_logprobs = meta_info.get("input_top_logprobs") or []
    prompt_ids_ls: list[list[int]] = []
    prompt_logprobs_ls: list[list[float]] = []
    # 跳过位置 0（logprob=None，无预测上下文），与 vLLM 约定一致
    for position in range(1, len(input_token_logprobs)):
        if num_prompt_logprobs == 0:
            # 仅取采样 token 的 logprob 和 token_id
            logprob, token_id, _ = input_token_logprobs[position]
            prompt_ids_ls.append([int(token_id)])
            prompt_logprobs_ls.append([float(logprob)])
```

```
            else:
                top_entries = input_top_logprobs[position]
                # SGLang 返回排名最佳优先，保持此顺序以匹配 vLLM 提取器的 rank-1 槽位
                ids = [int(tok_id) for _, tok_id, _ in top_entries]
                logprobs = [float(logprob) for logprob, _, _ in top_entries]
                # 硬断言确保数量一致（review 中建议改为填充，但作者保持断言）
                assert len(ids) == num_prompt_logprobs, (
                    f"SGLang returned {len(ids)} top logprobs at position {position}, "
                    f"expected {num_prompt_logprobs}."
                )
                prompt_ids_ls.append(ids)
                prompt_logprobs_ls.append(logprobs)
        # 追加一行填充数据，使总长度等于 sequence_length，匹配 vLLM 约定
        pad_width = max(num_prompt_logprobs, 1)
        prompt_ids_ls.append([0] * pad_width)
        prompt_logprobs_ls.append([0.0] * pad_width)
        # 最终长度断言，确保与消费者期望一致
        assert len(prompt_ids_ls) == sequence_length, (
            f"SGLang prompt_logprobs length ({len(prompt_ids_ls)}) does not match "
            f"sequence length ({sequence_length}); check logprob_start_len=0 invariant."
        )
        # 将结果填充到 result_dict 中，供蒸馏教师管理器消费
        result_dict["prompt_ids"] = prompt_ids_ls
        result_dict["prompt_logprobs"] = prompt_logprobs_ls
```

评论区精华

> Reviewer: gemini-code-assist[bot]指出 _extract_prompt_logprobs_sglang 中 input_top_logprobs 可能短于 input_token_logprobs 导致 IndexError，并建议将对 len(ids) == num_prompt_logprobs 的硬断言改为填充或截断，以避免因 SGLang 返回数量不一致时 worker 崩溃。
>
> Author: mingruimingrui回应边缘情况处理正确，如果 input_token_logprobs 为空，内层循环会被跳过。但对第二个关于断言严格性的评论未直接回复。

讨论集中在对 logprob 数据健壮性的处理上：reviewer 建议防御性编程（填充 / 截断），而作者维持了硬断言方案，仅在主线逻辑中确保一致性。该讨论未完全解决，但 PR 仍被合并。

- logprob 数据健壮性：空列表和 IndexError 风险 (correctness): 作者回复边缘情况处理正确（空列表时循环跳过），但未对 IndexError 场景给出方案。PR 合并时未修改代码，风险保留。
- 硬断言 vs 填充 / 截断的权衡 (design): 作者未直接回应，代码维持硬断言。可能认为在正确配置下不应出现不一致，但生产环境存在风险。

风险与影响

- 风险：

1. 生产环境健壮性风险：_extract_prompt_logprobs_sglang 中的 assert 语句如果 SGLang 返回的 top-k 数量与 num_prompt_logprobs 不一致，或最终列表长度与 sequence_length 不匹配，将导致进程崩溃。虽在测试中未触发，但在不同模型或极端情况下可能暴露。 - 涉及文件：verl/workers/rollout/sglang_rollout/async_sglang_server.py
2. 回退风险：该修改影响 SGLang rollouter 的 generate 接口，当未启用蒸馏（prompt_logprobs 为 None）时，行为应与之前完全一致，回归风险较低。
3. 缺失测试覆盖：目前无针对 _extract_prompt_logprobs_sglang 的单元测试，也未集成到 CI 中，边角情况可能不会被及时发现。

- 影响：
  - 用户影响：使用者只需在教师配置中将 distillation.teacher_models.teacher_model.inference.name 设为 sglang 即可使用 SGLang 作为蒸馏推理引擎，无需其他改动。
  - 系统影响：改动仅作用于蒸馏训练场景下的 SGLang 推理路径，不影响常规 rollout 或非蒸馏训练。
  - 团队影响：提供了独立于 vLLM 的教师引擎选项，有助于降低对单一推理引擎的依赖，并可能在多教师蒸馏场景中利用 SGLang 的性能优势。
  - 风险标记：缺乏测试覆盖，断言可能在生产环境崩溃

关联脉络

- PR #6072 [veomni] feat: enable VeOmni engine for on-policy distillation: 同为在线策略蒸馏添加新推理引擎支持，属于同一功能线（on-policy distillation teacher engine 扩展）。