PR #7193 完整报告

verl-project/verl

[ckpt, model] fix: validate model merger outputs

合并时间：2026-07-30 06:30

原文链接：http://prhub.com.cn/verl-project/verl/pull/7193

执行摘要

- 一句话：验证模型合并器输出的 Hugging Face 模型完整性
- 推荐动作：值得精读。本 PR 展示了一个好的防御性编程实践：在关键输出点添加显示后置条件检查，且设计上避免加载权重以保持轻量。对于处理模型保存和上传的开发者有参考价值。

功能与动机

模型合并器在 save_hf_model_and_tokenizer 中调用 model.save_pretrained 后直接返回，没有验证输出是否完整。如 Issue #7191 所述，如果模型实现返回了正常的 save_pretrained 但未写入权重，合并器不会报错。本 PR 添加显式后置条件检查，确保输出符合 Hugging Face 模型规范。

实现拆解

1. 新建 verl/model_merger/output_validation.py，实现 validate_hf_model_output 函数，通过文件系统检查 config.json 和权重文件的完整性，不加载张量。
2. 在 base_model_merger.py 的 save_hf_model_and_tokenizer 末尾添加调用 validate_hf_model_output(self.config.target_dir)。
3. 在 megatron_model_merger.py 的 save_hf_model_and_tokenizer 末尾（rank 0 路径）添加相同调用。
4. 新增测试文件 tests/model_merger/test_output_validation_on_cpu.py，使用 Mock 模型和真实 Transformers 模型验证接受 / 拒绝场景。
5. 验证函数包括路径安全检查，防止分片索引包含绝对路径或 ".."。

关键文件：

- verl/model_merger/output_validation.py（模块 模型合并；类别 source；类型 data-contract；符号 _is_nonempty_file, _safe_shard_path, validate_hf_model_output）：核心验证函数，检查模型输出目录是否包含完整的 Hugging Face 模型文件
- tests/model_merger/test_output_validation_on_cpu.py（模块 模型合并；类别 test；类型 test-coverage；符号 _TestModelMerger, merge_and_save, cleanup, _ConfigOnlyModel）：全面的 CPU 测试，覆盖接受和拒绝场景

- verl/model_merger/base_model_merger.py（模块 模型合并；类别 source；类型 data-contract）：在 FSDP 保存路径末尾添加验证调用
- verl/model_merger/megatron_model_merger.py（模块 模型合并；类别 source；类型 data-contract）：在 Megatron 保存路径末尾（rank 0）添加验证调用

关键符号：validate_hf_model_output, _is_nonempty_file, _safe_shard_path

关键源码片段

verl/model_merger/output_validation.py

核心验证函数，检查模型输出目录是否包含完整的 Hugging Face 模型文件

```
import json
from pathlib import Path, PurePosixPath

# 标准 Hugging Face 文件名常量
_CONFIG_NAME = "config.json"
_WEIGHT_NAMES = ("model.safetensors", "pytorch_model.bin")
_WEIGHT_INDEX_NAMES = ("model.safetensors.index.json", "pytorch_model.bin.index.json")


def _is_nonempty_file(path: Path) -> bool:
    # 文件存在且不为空
    return path.is_file() and path.stat().st_size > 0


def _safe_shard_path(target_dir: Path, shard_name: str) -> Path | None:
    # 防止分片路径穿越或绝对路径：只允许纯文件名或相对子路径
    shard_path = PurePosixPath(shard_name)
    if shard_path.is_absolute() or not shard_path.parts or ".." in shard_path.parts:
        return None
    return target_dir.joinpath(*shard_path.parts)


def validate_hf_model_output(target_dir: str | Path) -> None:
    # 确保目标目录包含完整的 Hugging Face 模型输出
    target_dir = Path(target_dir)
    errors = []

    # 验证 config.json: 必须存在、非空、可解析为 JSON 对象
    config_path = target_dir / _CONFIG_NAME
    if not _is_nonempty_file(config_path):
        errors.append(f"missing or empty {_CONFIG_NAME}")
    else:
        try:
            config = json.loads(config_path.read_text(encoding="utf-8"))
            if not isinstance(config, dict):
                errors.append(f"{_CONFIG_NAME} must contain a JSON object")
        except (OSError, UnicodeError, json.JSONDecodeError) as exc:
```

```
        errors.append(f"invalid {_CONFIG_NAME}: {exc}")

# 检查标准单权重文件 (model.safetensors 或 pytorch_model.bin)
has_complete_weights = any(
    _is_nonempty_file(target_dir / name) for name in _WEIGHT_NAMES
)

# 如果没有单文件，尝试解析分片索引
index_errors = []
for index_name in _WEIGHT_INDEX_NAMES:
    index_path = target_dir / index_name
    if not index_path.exists():
        continue
    try:
        index = json.loads(index_path.read_text(encoding="utf-8"))
    except (OSError, UnicodeError, json.JSONDecodeError) as exc:
        index_errors.append(f"invalid {index_name}: {exc}")
        continue

    weight_map = index.get("weight_map") if isinstance(index, dict) else None
    if not isinstance(weight_map, dict) or not weight_map:
        index_errors.append(f"{index_name} has no non-empty weight_map")
        continue

    shard_names = list(weight_map.values())
    if not all(isinstance(name, str) and name for name in shard_names):
        index_errors.append(f"{index_name} contains invalid shard names")
        continue

    # 去重并检查每个分片路径的安全性
    shard_names = set(shard_names)
    shard_paths = {name: _safe_shard_path(target_dir, name) for name in shard_names}
    unsafe_shards = sorted(name for name, path in shard_paths.items() if path is None)
    if unsafe_shards:
        index_errors.append(f"{index_name} contains unsafe shard paths: {unsafe_shards}")
        continue

    missing_shards = sorted(
        name for name, path in shard_paths.items() if not _is_nonempty_file(path)
    )
    if missing_shards:
        index_errors.append(f"{index_name} references missing or empty shards: {missing_
        shards}")
        continue

    has_complete_weights = True

# 汇总结果：如果没有完整权重，收集错误并抛出
if not has_complete_weights:
```

```
        errors.extend(index_errors)
        expected = ", ".join((*_WEIGHT_NAMES, *_WEIGHT_INDEX_NAMES))
        errors.append(f"no complete model weights; expected one of: {expected}")

    if errors:
        details = "; ".join(errors)
        raise RuntimeError(f"Incomplete Hugging Face model output at {target_dir}: {details}")
```

评论区精华

没有实质性的讨论评论。PR 获得了一位 reviewer 的批准。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。验证函数只进行文件存在性和 JSON 检查，不加载模型张量，对性能影响可忽略。唯一潜在风险：如果用户的自定义模型输出格式不符合标准的单文件或分片索引布局（例如使用自定义权重文件名），验证可能误判。但这种布局本就不被 Hugging Face 生态支持，提前失败反而有益。
- 影响：直接影响模型合并流程：之前可能静默接受不完整输出，现在会提前抛出 RuntimeError，防止后续上传或使用失败。影响范围限于使用 model_merger 模块的代码路径（FSDP 和 Megatron 后端）。测试覆盖了主流场景，包括单权重文件、分片索引、缺失权重、空 config 等。
- 风险标记：输入验证新增，测试覆盖充分，性能开销低

关联脉络

- PR #7068 [rollout, vllm, test] fix: normalize merged key names...: 解决了相邻但不同的模型合并输出问题（键名归一化），本 PR 补充了最终输出完整性检查
- PR #4770 [ckpt] fix: normalize PEFT keys in HF save and dump LoRA adapter with live config: 处理了 LoRA 适配器保存，但同样缺少对完整输出的终端检查