PR #6126 完整报告

verl-project/verl

[misc] refactor: re-format examples and deprecate old examples

合并时间：2026-04-30 11:37

原文链接：http://prhub.com.cn/verl-project/verl/pull/6126

执行摘要

- 一句话：重构并规范化示例目录，清理旧脚本，统一命名规范
- 推荐动作：建议在合并前对部分脚本中缺失的变量定义进行补充（如 actor_cp、actor_etp），并确保所有文档中的链接均已更新指向新路径。此外，建议尽快完成 NPU 脚本的统一对齐工作。该 PR 的设计思路（环境变量驱动的标准化脚本）值得后续新算法示例借鉴。

功能与动机

PR body 明确指出："Current examples are too heavy, some old ones may need to patch or modify verl's main-stream code and already placed in recipe, clear useless and repeated examples with feature tests." 其目标是让 examples 仅包含可直接使用主分支功能的轻量脚本，便于用户快速上手和社区维护。

实现拆解

1. 定义目录结构与命名规范：按算法名称（ppo_trainer、grpo_trainer 等）组织子目录，规范脚本文件名格式为 run__.sh，通过环境变量暴露所有可调参数，禁止在文件名中体现具体数据集或特征。
2. 重写所有示例脚本：将原有分散的、硬编码的脚本替换为规范化的模板，每组算法只保留一个 canonical 脚本，并支持通过 DEVICE、INFER_BACKEND、MACHINE 等环境切换 GPU/NPU 及后端（vLLM、SGLang、TensorRT-LLM）。
3. 删除旧例子和废弃目录：移除 fapo_trainer、split_placement、sglang_multiturn/search_r1_like 等目录，涉及的奖励函数、数据预处理、检索服务器、monkey patch 训练循环等文件一并删除，其功能要么已由主分支直接支持，要么已迁入 recipe 目录。
4. 添加命名规范检查测试：在 tests/special_sanity 下新增 check_example_naming.py，包含规范化检查函数和对应的 pytest 用例，并由 pre-commit 钩子触发，确保后续新增脚本命名合规。
5. 同步更新文档与配置：更新 docs/start/agentic_rl.rst、docs/ascend_tutorial 等文档中的脚本链接和路径，调整相关 CI 配置和 config 文件的 Hydra 参数。

关键文件：

- examples/grpo_trainer/run_qwen3_8b_fsdp.sh（模块 示例脚本；类别 config；类型 configuration）：新规范 canonical 示例脚本的代表，展示环境变量控制的标准化模式

- tests/special_sanity/check_example_naming.py（模块 命名检查；类别 test；类型 test-coverage；符号 _split_tokens, _is_ignored, check_filename, collect_scripts）：为核心命名规范提供自动化检查机制，通过 pre-commit 确保长期合规
- tests/special_sanity/test_check_example_naming.py（模块 测试工具；类别 test；类型 test-coverage；符号 _violations, test_canonical_name_passes, test_pre_backend_suffix_passes, test_all_train_backends_accepted）：为命名检查工具提供单元测试，确保规则逻辑正确
- examples/fapo_trainer/reward_fn.py（模块 奖励函数；类别 source；类型 deletion；符号 verify, compute_score_baseline, post_request, compute_score_fapo）：典型被删除的旧例子，包含自定义奖励函数和 GenRM 模板，说明清理范围
- examples/sglang_multiturn/search_r1_like/local_dense_retriever/retrieval_server.py（模块 检索服务器；类别 source；类型 deletion；符号 load_corpus, load_docs, load_model, pooling）：删除的多轮检索示例核心文件，体现对本地检索流水线的清理
- examples/split_placement/main_ppo_split.py（模块 训练脚本；类别 source；类型 deletion；符号 _select_rm_score_fn, RewardManager, init, call）：被删除的 split placement 训练入口，包含自定义 RewardManager 和 monkey patch，体现清理范围
- docs/start/agentic_rl.rst（模块 文档；类别 docs；类型 documentation）：文档更新同步脚本路径和命名，反映本次重构对文档的影响

关键符号：check_filename, collect_scripts, _split_tokens, _is_ignored, verify, compute_score_baseline, compute_score_fapo, post_request, _select_rm_score_fn, RewardManager.call, Encoder.encode, BaseRetriever, toolcall_shaping_reward, compute_score

关键源码片段

examples/sglang_multiturn/search_r1_like/local_dense_retriever/retrieval_server.py

删除的多轮检索示例核心文件，体现对本地检索流水线的清理

```
# 以下代码来自 examples/sglang_multiturn/search_r1_like/local_dense_retriever/retrieval_server.py,
# 该文件在本 PR 中被删除，因为其功能已被更通用的架构替代。
# 这里展示 Encoder 类实现，供理解旧检索流程参考。

class Encoder:
    """检索编码器，支持 e5、bge 等模型，提供查询/文档编码及池化。"""
    def __init__(self, model_name, model_path, pooling_method, max_length, use_fp16):
        self.model_name = model_name
        self.model_path = model_path
        self.pooling_method = pooling_method
        self.max_length = max_length
        self.use_fp16 = use_fp16
        self.model, self.tokenizer = load_model(model_path=model_path, use_fp16=use_fp16)
        self.model.eval()
```

```
    @torch.no_grad()
    def encode(self, query_list: list[str], is_query=True) -> np.ndarray:
        if isinstance(query_list, str):
            query_list = [query_list]
        # e5 模型使用 query/passage 前缀
        if "e5" in self.model_name.lower():
            prefix = "query: " if is_query else "passage: "
            query_list = [prefix + q for q in query_list]
        # bge 模型使用指令前缀
        if "bge" in self.model_name.lower() and is_query:
            query_list = [
                f"Represent this sentence for searching relevant passages: {q}" for q in query_list
            ]
        inputs = self.tokenizer(
            query_list, max_length=self.max_length, padding=True, truncation=True, return_
            tensors="pt"
        )
        inputs = {k: v.cuda() for k, v in inputs.items()}
        if "T5" in type(self.model).__name__:
            decoder_input_ids = torch.zeros((inputs["input_ids"].shape[0], 1), dtype=torch.long).to(
                inputs["input_ids"].device
            )
            output = self.model(**inputs, decoder_input_ids=decoder_input_ids, return_dict=True)
            query_emb = output.last_hidden_state[:, 0, :]
        else:
            output = self.model(**inputs, return_dict=True)
            query_emb = pooling(
                output.pooler_output, output.last_hidden_state, inputs["attention_mask"], self.
                pooling_method
            )
            if "dpr" not in self.model_name.lower():
                query_emb = torch.nn.functional.normalize(query_emb, dim=-1)
        query_emb = query_emb.detach().cpu().numpy().astype(np.float32, order="C")
        del inputs, output
        torch.cuda.empty_cache()
        return query_emb
```

examples/split_placement/main_ppo_split.py

被删除的 split placement 训练入口，包含自定义 RewardManager 和 monkey patch，体现清理范围

```
# 以下代码来自 examples/split_placement/main_ppo_split.py，该文件在本 PR 中被删除。
# 展示其自定义 RewardManager 实现，该功能现可由标准 reward 配置替代。

class RewardManager:
    def __init__(self, tokenizer, num_examine) -> None:
        self.tokenizer = tokenizer
        self.num_examine = num_examine
```

```
    def __call__(self, data: DataProto, return_dict: bool = False):
        if "rm_scores" in data.batch.keys():
            return data.batch["rm_scores"]
        reward_tensor = torch.zeros_like(data.batch["responses"], dtype=torch.float32)
        for i in range(len(data)):
            data_item = data[i]
            prompt_ids = data_item.batch["prompts"]
            prompt_length = prompt_ids.shape[-1]
            valid_prompt_length = data_item.batch["attention_mask"][:prompt_length].sum()
            valid_prompt_ids = prompt_ids[-valid_prompt_length:]
            response_ids = data_item.batch["responses"]
            valid_response_length = data_item.batch["attention_mask"][prompt_length:].sum()
            valid_response_ids = response_ids[:valid_response_length]
            sequences = torch.cat((valid_prompt_ids, valid_response_ids))
            sequences_str = self.tokenizer.decode(sequences)
            ground_truth = data_item.non_tensor_batch["reward_model"]["ground_truth"]
            data_source = data_item.non_tensor_batch["data_source"]
            compute_score_fn = _select_rm_score_fn(data_source)
            score = compute_score_fn(solution_str=sequences_str, ground_truth=ground_truth)
            reward_tensor[i, valid_response_length - 1] = score
        if return_dict:
            return {"reward_tensor": reward_tensor}
        else:
            return reward_tensor
```

评论区精华

Review 中核心讨论包括：

- 未定义变量问题：gemini-code-assist 指出新脚本中多处使用了 actor_cp、actor_etp 等变量但未在用户可调整部分定义，执行时可能失败（文件：run_qwen3_30b_a3b_megatron.sh 等）。
- 命名规范未完全对齐：sglang_multiturn 目录下的脚本名仍包含特征词（如 tool_agent、mlflow），与规范冲突，后通过删除该目录解决。
- NPU脚本暂不修改：wucong25提议暂时不调整NPU定制脚本，作者同意待后续统一对齐。
- 整体测试覆盖：tardis-key 询问是否测试了所有修改脚本，作者回应需社区协助验证配置可用性。
- DEVICE 自动检测：wuxibin89 建议用 pip3 show torch-npu 自动识别设备类型，避免用户手动设置。
- 弃用参数清理：wuxibin89 指出应移除 actor_rollout_ref.rollout.mode=async 等已弃用的 Hydra 配置项。
  - 脚本中未定义变量 actor_cp / actor_etp 导致运行时失败 (correctness): 作者需补充变量定义或调整脚本使其不依赖这些变量。后续提交中可能已部分修复，但 review 时仍存在。
  - sglang_multiturn 目录脚本命名未遵循新规范 (design): 后续提交中删除了 sglang_multiturn 目录，该问题自动解决。

- NPU 脚本是否暂不修改以对齐 GPU (question): 作者同意保持现有 NPU 脚本功能，待后续统一对齐。
- 测试所有修改脚本的可用性 (testing): 作者回应需要社区帮助在不同机器上验证配置，并强调脚本仅为可运行示例，不保证最佳性能。
- DEVICE 应自动检测而非用户手动设置 (design): 作者可能已采用建议，在最终脚本中改为自动检测方式。
- 移除已弃用的 Hydra 参数（async、hybrid_engine） (correctness): 作者需清理这些配置项，后续提交中可能已处理。

风险与影响

- 风险：主要风险包括：
  1. 脚本变量未定义风险：部分新脚本中引用的变量（如 actor_cp）在缺失定义时可能被 shell 的 set -u 设为空或报错，导致运行时失败。
  2. 文档链接失效：大量旧文档中的示例链接指向已删除 / 重命名的文件，需逐一确认更新。
  3. 删除的示例可能仍有用户依赖：虽然功能已由主分支或 recipe 覆盖，但用户现有工作流若直接引用旧路径会中断。
  4. 命名检查可能遗漏：check_example_naming.py 的规则集可能无法覆盖所有异常情况（如未来新增的后缀 token）。
  5. NPU 脚本兼容性：暂时保留的 NPU 脚本与新规范的统一调整工作尚未完成，可能出现两套风格并存的时期。 - 影响：用户影响：所有依赖旧 examples 目录结构或脚本名称的用户需迁移至新规范，但新脚本提供更灵活的环境变量控制方式，降低配置成本。团队影响：维护者获得更清晰的示例组织方式和自动化命名检查，减少后续 review 负担。系统影响：无直接核心代码变更，CI 中新增命名检查作业，不影响训练 / 推理流程。 - 风险标记：删除大量示例，新脚本变量定义风险，文档链接需更新，NPU 脚本待后续统一，命名检查可能遗漏

关联脉络

- 暂无明显关联 PR