PR #6488 完整报告

verl-project/verl

[veomni] fix: VeOmniEngineWithValueHead loads ForTokenClassification

合并时间：2026-05-27 10:16

原文链接：http://prhub.com.cn/verl-project/verl/pull/6488

执行摘要

- 一句话：修复 VeOmni critic 加载错误模型类型
- 推荐动作：本次 PR 虽小但具有较高价值，解决了 VeOmni 后端下 PPO 训练的核心计算错误。值得关注的设计决策是 _get_model_config_path 钩子模式：它避免了在 _build_model_optimizer 中硬编码配置分支，保持了父类逻辑的完整性，让子类通过重写来定制模型配置。建议后续将 hidden_dropout = "0" 改为 0.0 以消除类型风险。

功能与动机

verl 的 PPO 训练使用 critic/value 模型，需要 ForTokenClassification(num_labels=1) 输出每个 token 的标量值。VeOmni 的 build_foundation_model 根据 config.architectures 始终分发 ForCausalLM，导致 critic 输出 (B, seq_len, vocab_size) 的 logits 而非 (B, seq_len) 的标量值，破坏了 PPO 训练的正确性。PR body 中明确指出："Result: the critic/value model outputs (B, seq_len, vocab_size) logits instead of (B, seq_len) scalar values, breaking PPO training with VeOmni backend."

实现拆解

1. 新增 _get_model_config_path 钩子：在 VeOmniEngine 类（verl/workers/engine/veomni/transformer_impl.py）中添加 _get_model_config_path 方法，默认返回 self.model_config.local_hf_config_path，即原始配置路径，不改变原有行为。
2. 修改 _build_model_optimizer 调用点：将原本直接传入 config_path=self.model_config.local_hf_config_path 改为 config_path=self._get_model_config_path()，使得子类可通过重写钩子传递修改后的配置对象。
3. VeOmniEngineWithValueHead 重写钩子：该子类重写 _get_model_config_path，使用 transformers.AutoModelForTokenClassification._model_mapping 获取当前模型族对应的 ForTokenClassification 类名，创建并修改 HuggingFace 配置对象：设置 num_labels=1、classifier_dropout=0.0、tie_word_embeddings=False，并将 architectures 改为 [token_cls.__name__]，最后返回配置对象，使 VeOmni 的 MODELING_REGISTRY 能分派到正确的模型类。
4. 配套上游依赖：PR body 指出此修复依赖于 ByteDance-Seed/VeOmni#795（已合入），该 PR 在 VeOmni 的模型注册表中注册了 ForTokenClassification 类，否则 VeOmni 仍然

无法正确分派。

关键文件：

- verl/workers/engine/veomni/transformer_impl.py（模块 引擎；类别 source；类型 dependency-wiring；符号 _get_model_config_path）：唯一变更文件，通过新增 _get_model_config_path 钩子及子类重写，修复 VeOmni critic 加载错误模型类型的问题。

关键符号：_get_model_config_path

关键源码片段

verl/workers/engine/veomni/transformer_impl.py

唯一变更文件，通过新增 _get_model_config_path 钩子及子类重写，修复 VeOmni critic 加载错误模型类型的问题。

```
# verl/workers/engine/veomni/transformer_impl.py

class VeOmniEngine:
    # ... ( 其他方法 )

    def _get_model_config_path(self):
        """返回给 build_foundation_model 的 config 路径（或 PretrainedConfig 对象）。

        子类可重写此方法以在模型构建前修改 HF 配置
        （例如 VeOmniEngineWithValueHead 将 architectures 重写为 ForTokenClassification）。
        """
        return self.model_config.local_hf_config_path

    def _build_model_optimizer(self):
        # ... (ops_implementation 设置 )
        # 原本直接使用 self.model_config.local_hf_config_path，现在通过钩子方法获取
        module = build_foundation_model(
            config_path=self._get_model_config_path(), # <-- 关键改动
            weights_path=self.model_config.local_path,
            # ... ( 其他参数 )
        )
        # ... ( 后续并行化、优化器等 )


@EngineRegistry.register(model_type="language_model", backend=["veomni"], device=["cuda",
"npu"])
class VeOmniEngineWithValueHead(VeOmniEngine, FSDPEngineWithValueHead):
    """
    用于 value model 的 VeOmni 引擎，继承自 VeOmniEngine 和 FSDPEngineWithValueHead。
    重写 _get_model_config_path 以确保加载 ForTokenClassification(num_labels=1) 模型。
    """

    def _get_model_config_path(self):
        """返回修改后的 HF 配置，强制加载 ForTokenClassification(num_labels=1)。
```

```
    使用 transformers 的 AutoModelForTokenClassification._model_mapping
    获取当前模型族对应的 ForTokenClassification 类名，
    并设置 config.architectures，使 VeOmni 的 MODELING_REGISTRY 分派到正确的类。
    """
    from transformers import AutoModelForTokenClassification
    from veomni.models.auto import build_config

    # 1. 加载原始配置
    config = build_config(self.model_config.local_hf_config_path)

    # 2. 修改为 TokenClassification 所需设置
    config.num_labels = 1
    config.classifier_dropout = 0.0
    config.hidden_dropout = "0" # 注意：此处应为 0.0，string 有类型风险
    config.summary_dropout_prob = 0.0
    config.tie_word_embeddings = False

    # 3. 获取 ForTokenClassification 类名
    token_cls = AutoModelForTokenClassification._model_mapping.get(type(config))
    if token_cls is None:
        raise ValueError(
            f"No ForTokenClassification class in transformers for {type(config).__name__}."
        )
    # 4. 重写 architectures，使 VeOmni 分派到正确的模型类
    config.architectures = [token_cls.__name__]

    return config
```

评论区精华

Review 中只有一个实质性评论：gemini-code-assist[bot] 指出 config.hidden_dropout = "0" 应为 0.0（float 类型），否则会在 PyTorch 的 nn.Dropout 初始化时触发 TypeError。作者 Luosuu 回复“same as fsdp”，推测 FSDP 路径中也使用了相同的字符串写法（未在当前 diff 中验证）。该评论未被解决（not resolved），但 wuxibin89 已批准合并，说明 reviewer 可能认为该问题不影响功能（dropout=0 时无论 string 还是 float 都能正确执行？但严格来说确实不符合类型契约）。

- hidden_dropout 类型应为 float 而非 string (correctness): 未正式解决，但 reviewer 已批准合并。该写法在 dropout=0 时可能不会触发运行时错误，但不符合类型契约。

风险与影响

- 风险：
  1. 类型风险：config.hidden_dropout = "0" 使用字符串而非浮点数，虽在 dropout=0 时可能不会触发错误，但不符合 HuggingFace 配置类型规范，未来若变更采样逻辑可能导致异常。

2. 上游依赖风险：此修复依赖 VeOmni#795（ForTokenClassification 注册），若用户未升级对应 VeOmni 版本，MODELING_REGISTRY 无法解析 ForTokenClassification，会回退到 ForCausalLM，造成静默错误。
3. 回归风险：_get_model_config_path 默认路径未变，不影响原有 VeOmniEngine 行为；但 VeOmniEngineWithValueHead 的配置修改（如 tie_word_embeddings=False）可能影响权重加载兼容性。 - 影响：对用户：修复了 VeOmni 后端下 PPO critic 训练的核心 bug，使 VeOmni 用户能正确运行 PPO（含 GAE）。对系统：仅修改 verl/workers/engine/veomni/transformer_impl.py 一个文件，增加 32 行代码，无其他模块影响。对团队：解锁了 VeOmni 后端的完整 PPO 训练能力，是 #6453（VeOmni-native critic support）的关键 bugfix。 - 风险标记：类型不兼容 (hidden_dropout=string)，依赖上游 VeOmni#795

关联脉络

- PR #6453 [veomni] feat: add VeOmni-native critic support: 本 PR 是对 #6453 功能的 bugfix，#6453 添加了 VeOmniEngineWithValueHead 类但未正确处理模型类型加载。
- PR #795 [model] feat: register ForTokenClassification in model registries: Verl 中 VeOmni 的 ForTokenClassification 分派依赖于 ByteDance-Seed/VeOmni#795 在 VeOmni 模型注册表中注册相关类。