PR #50293 完整报告

vllm-project/vllm

[Model Runner V2] Enable encoder token classification

合并时间：2026-07-31 12:15

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50293

执行摘要

- 一句话：MRV2 支持 encoder-only 模型 token 分类
- 推荐动作：值得精读。该 PR 展示了如何在 MRV2 中按模型注意力类型开放任务，并保持向后兼容的优雅做法：通过 _get_enabled_tasks 分层过滤，并针对不可用场景给出可操作的错误提示。代码量小但契约清晰，测试覆盖了正反例和端到端精度，适合作为 MRV2 功能扩展的参考模板。

功能与动机

PR #48791 为 MRV2 引入了 sequence embedding 和 classification 支持，但 token_classify 被直接拒绝，即使模型是 encoder-only。PR #49331 提供了 encoder-only attention 的单步 prefill 路径，为 token 分类铺平了道路。本 PR 的动机正如 body 所述：让 MRV2 能够执行 encoder-only token classification，包括任务发现、校验、输出处理以及 Hugging Face 精度覆盖。

实现拆解

实现按以下步骤拆解：

1. 任务白名单扩展（pooling_runner.py）：将 _SUPPORTED_TASKS 从 {"embed", "classify"} 扩展为 {"embed", "classify", "token_classify"}。
2. 新增任务可用性判定（pooling_runner.py）：新增静态方法 _get_enabled_tasks(model_config)，当 model_config.attn_type == "encoder_only" 时返回全部任务，否则剔除 token_classify。这是因为 token 分类依赖 encoder-only attention 提供的单步 prefill 路径，decoder 及混合任务模型不支持。
3. 错误提示改进（pooling_runner.py）：__init__中的校验改用 enabled_tasks，并根据 enabled_tasks 与 model_tasks 的交集给出更精准的 fallback 提示，区分“可显式指定任务”和“必须关闭 V2 runner”。
4. 调用链同步（model_runner.py）：PoolingRunner.get_supported_tasks 签名增加 model_config 参数，model_runner.py 中调用处同步传入 self.model_config，保证 PP 第一 rank 上任务发现与校验一致。
5. 测试配套：
   - test_splade_sparse_pooler.py 新增三个单测，覆盖 encoder 支持、decoder 拒绝（错误信息不含“显式指定任务”）、decoder 混合任务过滤。

- test_token_classification.py 将原 test_bert_models 改为 test_bert_model_runner_v2，通过 monkeypatch 设置 VLLM_USE_V2_MODEL_RUNNER=1，分别验证单请求和 8-prompt 混合长度批次，并与 AutoModelForTokenClassification 输出在 atol=3.2e-2、rtol=1e-3 下对齐。

关键文件：

- vllm/v1/worker/gpu/pool/pooling_runner.py（模块 池化运行器；类别 source；类型 core-logic；符号 get_supported_tasks, _get_enabled_tasks）：核心逻辑所在：扩展任务白名单、新增 _get_enabled_tasks 判定、改进错误提示，是本次功能开启的入口。
- tests/models/language/pooling/test_splade_sparse_pooler.py（模块 池化测试；类别 test；类型 test-coverage；符号 test_pooling_runner_supports_encoder_token_classification, test_pooling_runner_rejects_decoder_token_classification, test_pooling_runner_filters_decoder_token_classification）：新增三个单测，覆盖 encoder 支持、decoder 拒绝、decoder 混合任务过滤三种关键分支，验证任务门控逻辑。
- tests/models/language/pooling/test_token_classification.py（模块 分类测试；类别 test；类型 test-coverage；符号 test_bert_models, test_bert_model_runner_v2）：将原有 BERT token 分类模型测试改造为 MRV2 专用，并添加单请求与混合长度批次的 HF 精度对比。
- vllm/v1/worker/gpu/model_runner.py（模块 模型运行器；类别 source；类型 data-contract）：调用契约随 get_supported_tasks 签名变化而同步，确保 PP 首 rank 的任务发现与 pooling_runner 配置一致。

关键符号：get_supported_tasks, _get_enabled_tasks, test_pooling_runner_supports_encoder_token_classification, test_pooling_runner_rejects_decoder_token_classification, test_pooling_runner_filters_decoder_token_classification, test_bert_model_runner_v2

关键源码片段

vllm/v1/worker/gpu/pool/pooling_runner.py

核心逻辑所在：扩展任务白名单、新增 _get_enabled_tasks 判定、改进错误提示，是本次功能开启的入口。

```
# vllm/v1/worker/gpu/pool/pooling_runner.py
# 全局支持的任务集合：在 embed/classify 基础上新增 token_classify
_SUPPORTED_TASKS: frozenset[PoolingTask] = frozenset(
    {"embed", "classify", "token_classify"}
)
```

```
class PoolingRunner:
    def __init__(self, model: nn.Module, vllm_config: VllmConfig):
        self.model = cast(VllmModelForPooling, model)
        self.model_config = vllm_config.model_config
        self.max_num_reqs = vllm_config.scheduler_config.max_num_seqs
        model_tasks = tuple(sorted(self.model.pooler.get_supported_tasks()))
```

```
        # 根据模型注意力类型计算本 runner 实际可用的任务集
        enabled_tasks = self._get_enabled_tasks(self.model_config)
        selected_task = self.model_config.get_pooling_task(model_tasks)
        if selected_task not in enabled_tasks:
            # 区分“可显式指定任务”与“必须关闭 V2 runner”两类错误提示
            hint = (
                "Set an explicitly supported task or VLLM_USE_V2_MODEL_RUNNER=0."
                if enabled_tasks.intersection(model_tasks)
                else "Set VLLM_USE_V2_MODEL_RUNNER=0 to use this model."
            )
            raise ValueError(
                "Model Runner V2 supports pooling tasks "
                f"{sorted(enabled_tasks)}, but this model selects "
                f"{selected_task!r} from {list(model_tasks)}. {hint}"
            )
        self.supported_tasks = frozenset(
            self.get_supported_tasks(model, self.model_config)
        )
        if not self.supported_tasks:
            raise ValueError(
                "Model Runner V2 supports pooling tasks "
                f"{sorted(enabled_tasks)}, but this model supports "
                f"{list(model_tasks)}. "
                "Set VLLM_USE_V2_MODEL_RUNNER=0 to use this model."
            )
        self.pooling_params: dict[int, PoolingParams] = {}
        self.pooling_states: dict[int, PoolingStates] = {}
        self.prompt_token_ids: dict[int, torch.Tensor] = {}

    @staticmethod
    def _get_enabled_tasks(model_config: ModelConfig) -> frozenset[PoolingTask]:
        # Token classification 只在无分块的 encoder-only 预填充路径下有效
        if model_config.attn_type == "encoder_only":
            return _SUPPORTED_TASKS
        # decoder 或混合注意力模型不支持 token 级分类
        return _SUPPORTED_TASKS - {"token_classify"}

    @staticmethod
    def get_supported_tasks(
        model: nn.Module, model_config: ModelConfig
    ) -> list[PoolingTask]:
        if not is_pooling_model(model):
            return []
        enabled_tasks = PoolingRunner._get_enabled_tasks(model_config)
        # 取模型声明任务与全局启用任务的交集，保证任务发现一致
        return sorted(model.pooler.get_supported_tasks() & enabled_tasks)
```

tests/models/language/pooling/test_splade_sparse_pooler.py

新增三个单测，覆盖 encoder 支持、decoder 拒绝、decoder 混合任务过滤三种关键分支，验证任务门控逻辑。

```
# tests/models/language/pooling/test_splade_sparse_pooler.py
def test_pooling_runner_supports_encoder_token_classification() -> None:
    # encoder_only 模型应完整支持 token_classify
    model = MagicMock()
    model.pooler.get_supported_tasks.return_value = {"token_classify"}
    vllm_config = MagicMock()
    vllm_config.scheduler_config.max_num_seqs = 2
    vllm_config.model_config.attn_type = "encoder_only"
    vllm_config.model_config.get_pooling_task.return_value = "token_classify"

    runner = PoolingRunner(model, vllm_config)

    assert runner.supported_tasks == {"token_classify"}


def test_pooling_runner_rejects_decoder_token_classification() -> None:
    # decoder 模型即使声明 token_classify，选中后也应报错
    model = MagicMock()
    model.pooler.get_supported_tasks.return_value = {"token_classify"}
    vllm_config = MagicMock()
    vllm_config.scheduler_config.max_num_seqs = 2
    vllm_config.model_config.attn_type = "decoder"
    vllm_config.model_config.get_pooling_task.return_value = "token_classify"

    with pytest.raises(ValueError, match="selects 'token_classify'") as exc_info:
        PoolingRunner(model, vllm_config)

    # decoder 模型没有可显式支持的替代任务，因此不应出现该提示
    assert "Set an explicitly supported task" not in str(exc_info.value)


def test_pooling_runner_filters_decoder_token_classification() -> None:
    # decoder 模型若同时声明 embed 和 token_classify，token_classify 应被过滤
    model = MagicMock()
    model.pooler.get_supported_tasks.return_value = {"embed", "token_classify"}
    vllm_config = MagicMock()
    vllm_config.scheduler_config.max_num_seqs = 2
    vllm_config.model_config.attn_type = "decoder"
    vllm_config.model_config.get_pooling_task.return_value = "embed"

    runner = PoolingRunner(model, vllm_config)

    assert runner.supported_tasks == {"embed"}
```

tests/models/language/pooling/test_token_classification.py

将原有 BERT token 分类模型测试改造为 MRV2 专用，并添加单请求与混合长度批次的 HF 精度对比。

```
# tests/models/language/pooling/test_token_classification.py
# 该测试强制使用 MRV2 并对比 HF 输出
@pytest.mark.parametrize("model", ["boltuix/NeuroBERT-NER"])
@pytest.mark.parametrize("dtype", ["float"])
@pytest.mark.core_model
@torch.inference_mode
def test_bert_model_runner_v2(
    hf_runner,
    vllm_runner,
    example_prompts,
    monkeypatch,
    model: str,
    dtype: str,
) -> None:
    # 构造单请求与整批两种 batch，覆盖混合长度场景
    prompt_batches = [[example_prompts[0]], example_prompts]

    monkeypatch.setenv("VLLM_USE_V2_MODEL_RUNNER", "1")
    with vllm_runner(model, max_model_len=None, dtype=dtype) as vllm_model:
        assert vllm_model.llm.llm_engine.vllm_config.use_v2_model_runner
        vllm_output_batches = [
            vllm_model.token_classify(prompts) for prompts in prompt_batches
        ]

    # ROCm 上 HF 侧使用 eager attention 规避精度问题
    hf_model_kwargs = {}
    if current_platform.is_rocm():
        hf_model_kwargs["attn_implementation"] = "eager"

    with hf_runner(
        model, dtype=dtype,
        auto_cls=AutoModelForTokenClassification,
        model_kwargs=hf_model_kwargs,
    ) as hf_model:
        tokenizer = hf_model.tokenizer
        hf_outputs = []
        for prompt in example_prompts:
            inputs = tokenizer([prompt], return_tensors="pt")
            inputs = hf_model.wrap_device(inputs)
            output = hf_model.model(**inputs)
            hf_outputs.append(softmax(output.logits[0]))
        hf_output_batches = [[hf_outputs[0]], hf_outputs]

    for hf_outputs, vllm_outputs in zip(hf_output_batches, vllm_output_batches):
        for hf_output, vllm_output in zip(hf_outputs, vllm_outputs):
            hf_output = hf_output.detach().clone().cpu().float()
```

```
vllm_output = vllm_output.detach().clone().cpu().float()
torch.testing.assert_close(
    hf_output, vllm_output, atol=3.2e-2, rtol=1e-3
)
```

评论区精华

本 PR 没有实质性的 review 讨论评论。两位维护者 yewentao256 和 njhill 均直接批准（LGTM）。唯一可提取的设计决策来自代码注释与 PR body：token_classify 仅对 encoder_only 注意力开放，原因是 encoder-only attention 提供单步 prefill 路径，decoder 和混合任务模型缺少该路径，因此需要按 attn_type 过滤。

- token_classify 仅对 encoder-only 开放的判定依据 (design): 采用 attn_type == "encoder_only" 作为唯一判据，decoder 模型统一剔除 token_classify。

风险与影响

- 风险：风险点如下：
  - 判定条件单一：仅凭 model_config.attn_type == "encoder_only" 来决定是否开放 token_classify。若未来出现支持单步 prefill 的 decoder 或混合注意力模型，该模型会被误拒，需要扩展判定逻辑。
  - 隐式行为变更：decoder 模型原本在 MRV2 中会因选中 token_classify 报错，现在若模型同时声明 embed 和 token_classify，token_classify 会被静默过滤，仅保留 embed；如果用户未显式指定任务，get_pooling_task 的选择结果可能变化。
  - 测试覆盖有限：端到端精度测试仅覆盖 boltuix/NeuroBERT-NER（BERT 架构），未覆盖其他 encoder-only 模型（如 ModernBERT 仍走 MRV1 路径），潜在回归风险未被完全暴露。
- 影响：影响范围集中在 MRV2 的 pooling 工作流：
  - 用户侧：使用 encoder-only 模型（如 BERT 系）执行 token 分类时，无需 VLLM_USE_V2_MODEL_RUNNER=0 回退即可使用 V2 引擎。
  - 系统侧：PoolingRunner 的任务发现逻辑调整为配置驱动，model_runner.py 的 get_supported_tasks 调用契约变化，所有 MRV2 下运行的 pooling 模型都会经过新判定。
  - 团队侧：该 PR 补全了 MRV2 与 MRV1 在 pooling 任务上的能力差距，是 MRV2 走向完全兼容的一步。
  - 风险标记：任务门控依赖单一 attn_type 判定，decoder 混合任务静默过滤行为变化，端到端测试仅覆盖单一 BERT 模型

关联脉络

- PR #48791 Model Runner V2 sequence embedding and classification support: 为 MRV2 引入 embed/classify 的基础支持，本 PR 在其之上扩展 token_classify。
- PR #49331 Encoder-only attention support for Model Runner V2: 提供 encoder-only 单步 prefill 路径，是本 PR 启用 token 分类的前置条件。