PR #47991 完整报告

vllm-project/vllm

[Model] Add RobertaForTokenClassification / XLMRobertaForTokenClassification

合并时间：2026-07-15 22:30

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47991

执行摘要

- 一句话：新增 RoBERTa/XLM-RoBERTa 的 Token 分类支持
- 推荐动作：值得查阅以了解如何添加新的 pooling 模型。设计决策包括：通过装饰器指定 attention 类型和池化类型，复用 BertModel 并替换 embedding_class，利用 AutoWeightsLoader 自动映射权重。这展示了一种低代码复用的模型注册模式。

功能与动机

当前 vLLM 的 /pooling 路由支持 BertForTokenClassification 和 ModernBertForTokenClassification，但 RoBERTa-family checkpoints（如 roberta-large-ner-english、xlm-roberta-base-ner-hrl）没有对应的 token-classification 入口，导致无法通过 vLLM 的 pooling runner 正确服务于这些模型。此 PR 填补该空白。

实现拆解

1. 在 vllm/model_executor/models/roberta.py 中新增 RobertaForTokenClassification 类：使用 @attn_type('encoder_only') 和 @default_pooling_type(tok_pooling_type='ALL') 装饰器标识为 encoder-only 模型和 ALL token 池化。构造函数初始化一个 BertModel（指定 embedding_class=RobertaEmbedding）和线性分类器，池化层使用 pooler_for_token_classify。load_weights 通过 AutoWeightsLoader 自动加载。forward 方法调用 roberta 子模块后调用分类器。
2. 在 vllm/model_executor/models/registry.py 注册模型：在 _TOKEN_CLASSIFICATION_MODELS 字典中添加 RobertaForTokenClassification 和 XLMRobertaForTokenClassification，均指向 roberta 模块的 RobertaForTokenClassification 类。
3. 测试配套：在 tests/models/language/pooling/test_token_classification.py 中新增 test_xlm_roberta_models 测试函数，使用 Davlan/xlm-roberta-base-ner-hrl 模型，比较 vLLM 输出与 HF Transformers 输出的 softmax logits，容忍度 3.2e-2 atol / 1e-3 rtol。同时更新 tests/models/registry.py，注册两个测试模型。
4. 文档更新：在 docs/models/pooling_models/token_classify.md 的支持模型表中添加两行。

关键文件：

- vllm/model_executor/models/roberta.py（模块 模型层；类别 source；类型 core-logic；符号 RobertaForTokenClassification, init, embed_input_ids, load_weights）：核心实现

文件，新增 RobertaForTokenClassification 类及其 forward/load_weights/embed_input_ids 方法，通过装饰器和复用 BertModel 实现 token 分类逻辑。

- vllm/model_executor/models/registry.py（模块 模型注册；类别 source；类型 data-contract）：模型注册表，将两个架构名映射到 RobertaForTokenClassification 类，是模型发现的关键环节。
- tests/models/language/pooling/test_token_classification.py（模块 测试；类别 test；类型 test-coverage；符号 test_xlm_roberta_models）：端到端测试验证新模型在 vLLM 与 HF Transformers 上输出一致，是质量保证的重要环节。
- tests/models/registry.py（模块 测试；类别 test；类型 test-coverage）：注册测试用模型示例，确保新架构在注册表一致性测试中被覆盖。
- docs/models/pooling_models/token_classify.md（模块 文档；类别 docs；类型 documentation）：更新支持模型列表文档，使用户能发现新功能。

关键符号：RobertaForTokenClassification.init, RobertaForTokenClassification.forward, RobertaForTokenClassification.load_weights, RobertaForTokenClassification.embed_input_ids, test_xlm_roberta_models

关键源码片段

vllm/model_executor/models/roberta.py

核心实现文件，新增 RobertaForTokenClassification 类及其 forward/load_weights/embed_input_ids 方法，通过装饰器和复用 BertModel 实现 token 分类逻辑。

```
# vllm/model_executor/models/roberta.py

# 新增的 RobertaForTokenClassification 类
@attn_type("encoder_only")
@default_pooling_type(tok_pooling_type="ALL")
class RobertaForTokenClassification(nn.Module):
    """基于 RoBERTa 的 Token 分类模型，直接复用 BertForTokenClassification 的结构，
    替换为 RobertaEmbedding 和 roberta 权重前缀。该别名也用于 XLM-RoBERTa。"""

    is_pooling_model = True

    def __init__(self, *, vllm_config: VllmConfig, prefix: str = ""):
        super().__init__()
        config = vllm_config.model_config.hf_config
        self.head_dtype = vllm_config.model_config.head_dtype
        self.num_labels = config.num_labels
        # 使用 BertModel 并指定 embedding_class=RobertaEmbedding
        self.roberta = BertModel(
            vllm_config=vllm_config,
            prefix=maybe_prefix(prefix, "roberta"),
            embedding_class=RobertaEmbedding,
```

```
        )
        # 线性分类器，输出 num_labels 个 logits
        self.classifier = nn.Linear(
            config.hidden_size, config.num_labels, dtype=self.head_dtype
        )

        pooler_config = vllm_config.model_config.pooler_config
        assert pooler_config is not None
        # 使用 token 分类池化器（ALL pooling）
        self.pooler = pooler_for_token_classify(pooler_config)

    def embed_input_ids(self, input_ids: torch.Tensor) -> torch.Tensor:
        # 委托给 roberta 子模型
        return self.roberta.embed_input_ids(input_ids)

    def load_weights(self, weights: Iterable[tuple[str, torch.Tensor]]):
        # 使用 AutoWeightsLoader 自动处理权重映射
        loader = AutoWeightsLoader(self)
        return loader.load_weights(weights)

    def forward(
        self,
        input_ids: torch.Tensor | None,
        positions: torch.Tensor,
        intermediate_tensors: IntermediateTensors | None = None,
        inputs_embeds: torch.Tensor | None = None,
        token_type_ids: torch.Tensor | None = None,
    ) -> torch.Tensor:
        # 处理 token_type_ids 编码（与 Bert 相同）
        if token_type_ids is not None:
            assert self.roberta.config.vocab_size < (1 << TOKEN_TYPE_SHIFT)
            assert input_ids is not None
            _encode_token_type_ids(input_ids, token_type_ids)

        # 通过 RoBERTa backbone 获取 hidden states
        hidden_states = self.roberta(
            input_ids=input_ids,
            positions=positions,
            inputs_embeds=inputs_embeds,
            intermediate_tensors=intermediate_tensors,
        )
        # 转换到 head_dtype 后通过分类器
        hidden_states = hidden_states.to(self.head_dtype)
        return self.classifier(hidden_states)
```

tests/models/language/pooling/test_token_classification.py

端到端测试验证新模型在 vLLM 与 HF Transformers 上输出一致，是质量保证的重要环节。

```
# tests/models/language/pooling/test_token_classification.py
```

```
@pytest.mark.parametrize("model", ["Davlan/xlm-roberta-base-ner-hrl"])
@pytest.mark.parametrize("dtype", ["float"])
@torch.inference_mode
def test_xlm_roberta_models(hf_runner, vllm_runner, example_prompts, model, dtype):
    # 使用 vLLM runner 进行 token 分类，与 HF Transformers 对比
    with vllm_runner(model, max_model_len=None, dtype=dtype) as vllm_model:
        vllm_outputs = vllm_model.token_classify(example_prompts)

    # ROCm 上使用 eager attention 避免 flash attention 精度差异
    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]))

    # 比较每个 token 的 softmax logits 差异，误差容忍度与现有模型测试一致
    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

该 PR 来自 fork，Claude 自动 review 未开启。审核者 noooop 直接批准，无额外讨论。

- 暂无高价值评论线程

风险与影响

- 风险：风险极低。新类完全模仿已存在的 BertForTokenClassification 结构，仅替换 embedding 层和权重前缀。RoBERTa 权重前缀（roberta.）与 vLLM 现有 RobertaForSequenceClassification 的处理方式一致，已在生产中使用。测试覆盖了代表性的 XLM-RoBERTa 模型。潜在风险包括：未来 HuggingFace 可能修改 RoBERTa 配置，但 vLLM 会持续适配。建议监控 RoBERTa 系列模型的 regression。
- 影响：影响有限。仅扩展 pooling runner 的能力，不影响系统性能。用户现在可以使用 token-classify 模式加载 RoBERTa 和 XLM-RoBERTa 模型。团队在后续模型扩展时可以借鉴此模式。文档已更新，用户可发现新支持。
- 风险标记：暂无

关联脉络

- 暂无明显关联 PR