PR #48463 完整报告

vllm-project/vllm

[Feat] Add Support for BertForMaskedLM to vLLM

合并时间：2026-07-14 04:56

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48463

执行摘要

此 PR 为 vLLM 添加了对 Hugging Face BertForMaskedLM 模型的支持，使用户可以直接加载和运行 MLM 检查点（如 google-bert/bert-base-uncased）。实现通过复用现有 BertModel 作为编码器骨干，添加 MLM 头和 token 分类池化器，并配置了完善的权重名称映射以兼容不同检查点变体。

功能与动机

PR body 指出：vLLM currently supports BERT encoder models but does not support BertForMaskedLM, preventing Hugging Face masked language modeling checkpoints from being loaded and executed. This PR closes that gap so users can run MLM checkpoints directly on vLLM.

实现拆解

1. 定义模型类：在 vllm/model_executor/models/bert.py 中新增 BertForMaskedLM 类，使用 @attn_type('encoder_only') 和 @default_pooling_type(tok_pooling_type='ALL') 装饰。类内部实例化 BertModel 骨干、BertMLMHead 和 token 分类池化器。hf_to_vllm_mapper 处理了权重名称映射，包括丢弃 NSP head、处理 legacy LayerNorm 命名偏差和 tied decoder 权重。
2. 注册模型：在 vllm/model_executor/models/registry.py 的 _EMBEDDING_MODELS 字典中添加 'BertForMaskedLM': ('bert', 'BertForMaskedLM')，使模型可通过 Hugging Face 架构名自动加载。
3. 测试验证：在 tests/models/language/pooling/test_token_classification.py 中添加 test_bert_for_masked_lm 测试，使用 google-bert/bert-base-uncased 检查点，比较 vLLM token_classify 输出与 Hugging Face AutoModelForMaskedLM 输出的 top-1 一致性和分布相似度。同时在 tests/models/registry.py 中添加示例模型条目。

BertForMaskedLM 类实现

```
@attn_type('encoder_only')
@default_pooling_type(tok_pooling_type='ALL')
class BertForMaskedLM(nn.Module):
    '''Bert with a masked-language-modeling head on top of BertModel.'''

    is_pooling_model = True

    hf_to_vllm_mapper = WeightsMapper(
```

```
        orig_to_new_substr={
            'cls.seq_relationship': None,
            'cls.predictions.decoder.bias': None,
            'cls.predictions.transform.LayerNorm.gamma': 'mlm_head.layer_norm.weight',
            'cls.predictions.transform.LayerNorm.beta': 'mlm_head.layer_norm.bias',
            'cls.predictions.transform.LayerNorm': 'mlm_head.layer_norm',
            'cls.predictions.transform.dense': 'mlm_head.dense',
            'cls.predictions.decoder': 'mlm_head.decoder',
            'cls.predictions.bias': 'mlm_head.decoder.bias',
        }
    )

    def __init__(self, *, vllm_config: VllmConfig, prefix: str = ''):
        super().__init__()
        config = vllm_config.model_config.hf_config
        self.bert = BertModel(
            vllm_config=vllm_config,
            prefix=maybe_prefix(prefix, 'bert'),
            embedding_class=BertEmbedding,
        )
        self.mlm_head = BertMLMHead(
            hidden_size=config.hidden_size,
            vocab_size=config.vocab_size,
            layer_norm_eps=getattr(config, 'layer_norm_eps', 1e-12),
        )
        pooler_config = vllm_config.model_config.pooler_config
        assert pooler_config is not None
        self.pooler = pooler_for_token_classify(pooler_config)

    def embed_input_ids(self, input_ids):
        return self.bert.embed_input_ids(input_ids)

    def load_weights(self, weights):
        return AutoWeightsLoader(self).load_weights(weights)

    def forward(self, input_ids, positions, intermediate_tensors=None, inputs_embeds=None):
        hidden_states = self.bert(input_ids, positions, inputs_embeds, intermediate_tensors)
        logits = self.mlm_head(hidden_states)
        return self.pooler(logits)
```

测试函数

```
@pytest.mark.parametrize(
    'model',
    ['google-bert/bert-base-uncased'],
)
@pytest.mark.parametrize('dtype', ['float'])
@torch.inference_mode
def test_bert_for_masked_lm(
    hf_runner, vllm_runner, example_prompts, model: str, dtype: str
```

```
) -> None:
    with vllm_runner(model, max_model_len=None, dtype=dtype) as vllm_model:
        vllm_outputs = vllm_model.token_classify(example_prompts)

    hf_model_kwargs = {}
    if current_platform.is_rocm():
        hf_model_kwargs['attn_implementation'] = 'eager'

    with hf_runner(
        model, dtype=dtype, auto_cls=AutoModelForMaskedLM, 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]))

    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()
        assert hf_output.shape == vllm_output.shape
        assert torch.equal(hf_output.argmax(dim=-1), vllm_output.argmax(dim=-1))
        torch.testing.assert_close(hf_output, vllm_output, atol=3.2e-2, rtol=1e-3)
```

评论区精华

- DarkLight1337 建议将单独的测试文件 test_mlm.py 合并到现有的 test_token_classification.py 中，以减少新文件数量。作者 atalhens 同意并执行。该讨论体现了对测试目录结构简洁性的关注。

风险与影响

- 风险：权重映射的正确性依赖于检查点命名规范，当前仅测试了 google-bert/bert-base-uncased，对于其他变体（如中文模型、蒸馏模型）可能需要调整。此外，MLM head 的 tied embedding 处理可能在某些检查点上不适用。
- 影响：用户现在可以使用 vLLM 执行 MLM 任务，扩展了模型支持范围。影响范围局限于新增模型，现有功能不受影响。

关联脉络

此 PR 与近期历史 PR 无直接关联，但延续了 vLLM 对 BERT 系列模型的支持，进一步丰富了模型的多样性。