

PR #40136 完整报告

vllm-project/vllm

[Model] support Qianfan-OCR model

合并时间: 2026-05-05 18:51

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/40136>

执行摘要

- 一句话: 新增 QianfanOCR 多模态模型支持
- 推荐动作: 此 PR 适合作为多模态新模型集成的参考范例, 特别是如何基于现有架构 (InternVL) 快速实现。Review 中关于 `trust_remote_code` 的设计权衡也值得学习。建议开发者关注其配置管理和测试组织方式。

功能与动机

根据 PR 描述, 目的是为 QianfanOCR 模型添加支持。在 Review 讨论中, 作者表示希望将配置直接包含在 vLLM 中, 使用户无需指定 `trust_remote_code` 即可加载模型, 与其他开源模型的使用体验保持一致。

实现拆解

1. 配置文件定义 (`vllm/transformers_utils/configs/qianfan_ocr.py`): 新增 `QianfanOCRVisionConfig` 和 `QianfanOCRConfig`, 分别对应视觉编码器和全模型配置。`QianfanOCRConfig` 兼容 `dict` 或 `PretrainedConfig` 输入, 并自动使用 `Qwen3` 作为默认文本配置。
2. 模型执行器 (`vllm/model_executor/models/qianfan_ocr.py`): 定义 `QianfanOCRProcessingInfo` (仅图像处理) 和 `QianfanOCRForConditionalGeneration`, 后者直接继承 `InternVLChatModel`, 仅重写 `_patch_quant_config` 以在 FP8 量化时排除视觉层。通过 `MULTIMODAL_REGISTRY` 注册了处理器和 `DummyInputs` 生成器。
3. 模型和配置注册: 在 `vllm/model_executor/models/registry.py` 中添加架构映射; 在 `vllm/transformers_utils/configs/__init__.py` 和 `vllm/transformers_utils/config.py` 中添加配置类映射, 使 vLLM 能自动识别并加载。
4. 测试支持: 在 `tests/models/multimodal/generation/vlm_utils/model_utils.py` 中添加辅助函数; 在 `test_common.py` 中添加 `VLMTestInfo` 条目; 处理器测试因与 InternVL 共享, 经 review 后删除; 由于需要 `transformers 5.6.0`, 设置了 `min_transformers_version` 跳过测试。
5. 文档更新: 在 `docs/models/supported_models.md` 中新增条目。

关键文件:

- `vllm/model_executor/models/qianfan_ocr.py` (模块 模型执行; 类别 `source`; 类型 `data-contract`; 符号 `QianfanOCRProcessingInfo`, `get_hf_processor`,

QianfanOCRForConditionalGeneration, _patch_quant_config) : 核心模型执行器文件, 定义 QianfanOCRProcessingInfo 和 QianfanOCRForConditionalGeneration, 重用 InternVL 的模型结构和多模态处理逻辑, 仅覆盖量化配置补丁和处理器参数。

- vllm/transformers_utils/configs/qianfan_ocr.py (模块 配置类; 类别 source; 类型 core-logic; 符号 QianfanOCRVisionConfig, QianfanOCRConfig, init) : 定义 QianfanOCRVisionConfig 和 QianfanOCRConfig, 处理视觉配置和文本配置, 被模型执行器和测试工具引用。
- tests/models/multimodal/generation/vlm_utils/model_utils.py (模块 测试工具; 类别 test; 类型 test-coverage; 符号 qianfan_ocr_hf_model_kwargs, qianfan_ocr_patch_hf_runner, QianfanOCRProcessor) : 添加了测试辅助函数 qianfan_ocr_hf_model_kwargs 和 qianfan_ocr_patch_hf_runner, 用于生成测试所需的 HF 模型参数并打补丁以执行推理验证。
- vllm/model_executor/models/registry.py (模块 模型注册; 类别 source; 类型 data-contract) : 注册 QianfanOCRForConditionalGeneration 到模型映射, 使 vLLM 能根据 architecture 名称找到对应模型实现。
- vllm/transformers_utils/configs/__init__.py (模块 配置注册; 类别 source; 类型 core-logic) : 注册 QianfanOCRConfig 和 QianfanOCRVisionConfig 到配置映射, 使 vLLM 能自动加载正确的配置类。

关键符号: QianfanOCRProcessingInfo.get_hf_processor,
QianfanOCRForConditionalGeneration._patch_quant_config,
QianfanOCRVisionConfig.init, QianfanOCRConfig.init, qianfan_ocr_hf_model_kwargs,
qianfan_ocr_patch_hf_runner, QianfanOCRProcessor.init, QianfanOCRProcessor.call

关键源码片段

vllm/model_executor/models/qianfan_ocr.py

核心模型执行器文件, 定义 QianfanOCRProcessingInfo 和 QianfanOCRForConditionalGeneration, 重用 InternVL 的模型结构和多模态处理逻辑, 仅覆盖量化配置补丁和处理器参数。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project

# QianfanOCR 基于 InternVL 构建, 使用 Qwen3 作为语言骨干。
# 模型架构和权重与 InternVLChatModel 完全兼容, 仅 config.model_type 和 architectures 不同。
```

```
from transformers import PretrainedConfig

from vllm.model_executor.layers.quantization import QuantizationConfig
from vllm.model_executor.layers.quantization.fp8 import Fp8Config
from vllm.multimodal import MULTIMODAL_REGISTRY
from vllm.transformers_utils.processors.internvl import (
    InternVLImageProcessor,
    InternVLProcessor,
)
```

```

from .internvl import (
    BaseInternVLDummyInputsBuilder,
    BaseInternVLMultiModalProcessor,
    BaseInternVLProcessingInfo,
    InternVLChatModel,
)

class QianfanOCRProcessingInfo(BaseInternVLProcessingInfo):
    """仅支持图片的 ProcessingInfo, 不支持视频。"""

    def get_hf_processor(self, **kwargs: object) -> InternVLProcessor:
        config = self.get_hf_config()
        vision_config = config.vision_config

        # 合并多模态上下文中的额外参数, 并从配置中提取默认值
        kwargs = self.ctx.get_merged_mm_kwargs(kwargs)
        kwargs.setdefault("image_size", vision_config.image_size)
        kwargs.setdefault("min_dynamic_patch", config.min_dynamic_patch)
        kwargs.setdefault("max_dynamic_patch", config.max_dynamic_patch)
        kwargs.setdefault("dynamic_image_size", config.dynamic_image_size)
        kwargs.setdefault("use_thumbnail", config.use_thumbnail)

        image_processor = InternVLImageProcessor(**kwargs)
        image_size = image_processor.image_size
        patch_size = vision_config.patch_size
        downsample_ratio = config.downsample_ratio
        # 计算每个图像对应的 token 数
        image_seq_length = int((image_size // patch_size) ** 2 * (downsample_ratio**2))

        return InternVLProcessor(
            tokenizer=self.get_tokenizer(),
            image_processor=image_processor,
            video_processor=None, # 不支持视频
            image_seq_length=image_seq_length,
            ctx_video_token=None,
        )

@MULTIMODAL_REGISTRY.register_processor(
    BaseInternVLMultiModalProcessor,
    info=QianfanOCRProcessingInfo,
    dummy_inputs=BaseInternVLDummyInputsBuilder,
)

class QianfanOCRForConditionalGeneration(InternVLChatModel):
    """
    QianfanOCR 多模态模型。

    结构与 InternVLChatModel 完全相同 (InternViT 视觉编码器 +

```

pixel-shuffle MLP 连接器 + Qwen3 语言模型)。此类仅用于注册出现在模型 config.json 中的架构名称。

"""

```
def _patch_quant_config(
    self, config: PretrainedConfig, quant_config: QuantizationConfig
) -> None:
    super()._patch_quant_config(config, quant_config)
    # 在 FP8 量化时忽略视觉层以保持模型性能
    if isinstance(quant_config, Fp8Config):
        # 动态获取 vision encoder 层数，支持模型变体
        _FP8_IGNORED_LAYERS = [
            *(
                layer
                for i in range(config.vision_config.num_hidden_layers)
                for layer in [
                    f"vision_model.encoder.layers.{i}.attn.qkv",
                    f"vision_model.encoder.layers.{i}.attn.proj",
                    f"vision_model.encoder.layers.{i}.mlp.fc1",
                    f"vision_model.encoder.layers.{i}.mlp.fc2",
                ]
            ),
            "language_model.lm_head",
            "mlp1.1",
            "mlp1.3",
        ]
        for layer in _FP8_IGNORED_LAYERS:
            if layer not in quant_config.ignored_layers:
                quant_config.ignored_layers.append(layer)
```

评论区精华

1. trust_remote_code 权衡: DarkLight1337 建议使用 trust_remote_code 以减少维护负担，作者表示愿意承担维护责任，最终保留内置配置。
 2. 测试冗余性: DarkLight1337 指出单独的处理器测试与 InternVL 测试重叠，且模型初始化测试已覆盖配置加载，作者同意后移除了 test_qianfan_ocr.py。
 3. 排序问题: DarkLight1337 要求将新增条目按字母顺序排列，作者已修正。
 4. 硬编码层数: gemini-code-assist[bot] 建议将视觉层数硬编码 24 改为动态读取配置，已被采纳（最终代码使用 config.vision_config.num_hidden_layers）。
- 是否将配置文件内置到 vLLM 还是使用 trust_remote_code (design): 双方达成一致，保留配置文件并作者负责维护。
 - 测试文件冗余 (testing): 作者移除了处理器测试和冗余配置测试文件。
 - 代码组织规范性 (style): 作者更新了排序。
 - 硬编码 vision encoder 层数 (correctness): 已采纳，最终代码使用 config.vision_config.num_hidden_layers。

风险与影响

- 风险:

1. 配置类需要持续维护，若上游模型配置变更可能产生不一致，但作者承诺维护。
2. 依赖 transformers 5.6.0, vLLM 当前默认 5.5.3, 测试已通过 min_transformers_version 跳过，但用户需要手动升级 transformers。
3. 模型完全继承 InternVL，若 InternVL 出现 bug 修复，QianfanOCR 也需要同步更新。
 - 影响：用户可以直接使用 baidu/QianfanOCR 模型，无需 trust_remote_code，降低使用门槛。新增约 300 行源码，对系统影响局部化。团队维护负担由作者承担，但减少了其他用户的接入成本。
 - 风险标记：新模型集成，外部维护依赖，transformers 版本兼容，继承架构同步风险

关联脉络

- 暂无明显关联 PR