PR #6631 完整报告

verl-project/verl

[data] fix: support path multimodal placeholders

合并时间：2026-06-08 11:40

原文链接：http://prhub.com.cn/verl-project/verl/pull/6631

执行摘要

- 一句话：支持多模态路径占位符
- 推荐动作：此 PR 设计简洁且测试充分，适合作为多模态数据集扩展的参考实现。值得关注的地方：如何在保留后向兼容的前提下扩展输入类型，以及视频帧列表的路径统一转换策略。

功能与动机

关联 Issue #6623 报告了 maybe_filter_out_long_prompts 过滤 prompt 时无法处理图片路径，会触发 TypeError。PR Body 说明用户直接传入文件路径字符串时 _build_messages 会失败。此项修改使路径输入能被正确转换为 processor 要求的 content 格式。

实现拆解

1. 扩展 image 处理分支：在 verl/utils/dataset/rl_dataset.py 的 _build_messages 方法中，为 <image> 占位符新增 elif isinstance(image, str | os.PathLike) 分支，调用 os.fspath(image) 将路径转换为字符串，构建 {'type': 'image', 'image': path} 格式的 content 元素。
2. 扩展 video 处理分支：类似地，为 <video> 占位符增加三种 case：当 video 为 dict 时保留原有逻辑；为 str | os.PathLike 时转换为 {'type': 'video', 'video': os.fspath(video)}；为 list 时逐个将 os.PathLike 元素转为字符串，再构建 {'type': 'video', 'video': video_list}。
3. 保留后向兼容：所有现有逻辑（dict、PIL.Image）保持不变，仅增加新分支，确保已有数据集不受影响。
4. 新增 CPU 级别单元测试：在 tests/utils/dataset/test_rl_dataset_on_cpu.py 中新增辅助函数 _mock_rlhf_dataset 和三个测试函数，分别验证 image 路径、video 路径、video 帧列表在 _build_messages 中的正确输出。
5. 集成过滤流程测试：通过 test_maybe_filter_out_long_prompts_accepts_image_path 使用 monkeypatch 模拟全链路，验证路径输入在 maybe_filter_out_long_prompts 中能正确通过 processor 处理。

关键文件：

- verl/utils/dataset/rl_dataset.py（模块 数据处理；类别 source；类型 core-logic；符号 _build_messages）：核心逻辑修改，在 _build_messages 中新增 image 和 video 的路径类型支持

- tests/utils/dataset/test_rl_dataset_on_cpu.py（模块 单元测试；类别 test；类型 test-coverage；符号 _mock_rlhf_dataset, test_build_messages_accepts_image_path, test_build_messages_accepts_video_path_or_frame_list, test_maybe_filter_out_long_prompts_accepts_image_path）：新增 CPU 测试覆盖，验证路径输入在 _build_messages 和 maybe_filter_out_long_prompts 中的正确处理

关键符号：_build_messages, _mock_rlhf_dataset, test_build_messages_accepts_image_path, test_build_messages_accepts_video_path_or_frame_list, test_maybe_filter_out_long_prompts_accepts_image_path

关键源码片段

verl/utils/dataset/rl_dataset.py

核心逻辑修改，在 _build_messages 中新增 image 和 video 的路径类型支持

```
def _build_messages(self, example: dict, key: str):
    '''Replace multimodal placeholders in messages with structured content.'''
    messages: list = example[key]
    images = example.get(self.image_key, None) or []
    videos = example.get(self.video_key, None) or []
    audios = example.get(self.audio_key, None) or []

    image_offset, video_offset, audio_offset = 0, 0, 0
    for message in messages:
        if not images and not videos and not audios:
            continue
        assert self.processor is not None

        content = message['content']
        if not isinstance(content, str):
            continue

        content_list = []
        segments = re.split('(<image>|<video>|<audio>)', content)
        segments = [item for item in segments if item != '']
        for segment in segments:
            if segment == '<image>':
                assert image_offset < len(images)
                image = images[image_offset]
                if isinstance(image, Image.Image):
                    image = image.convert('RGB')
                    content_list.append({'type': 'image', 'image': image})
                elif isinstance(image, dict):
                    if 'bytes' in image:
                        image['image'] = Image.open(BytesIO(image['bytes']))
                    content_list.append({'type': 'image', **image})
                elif isinstance(image, str | os.PathLike):
                    # 新分支：支持路径字符串或 PathLike
                    content_list.append({'type': 'image', 'image': os.fspath(image)})
```

```
                else:
                    raise TypeError(...)
                image_offset += 1
            elif segment == '<video>':
                assert video_offset < len(videos)
                video = videos[video_offset]
                if isinstance(video, dict):
                    content_list.append({'type': 'video', **video})
                elif isinstance(video, str | os.PathLike):
                    # 新分支：视频路径
                    content_list.append({'type': 'video', 'video': os.fspath(video)})
                elif isinstance(video, list):
                    # 新分支：视频帧列表，统一转换内部 PathLike
                    video = [os.fspath(frame) if isinstance(frame, os.PathLike) else frame for frame
                    in video]
                    content_list.append({'type': 'video', 'video': video})
                else:
                    raise TypeError(...)
                video_offset += 1
            elif segment == '<audio>':
                # audio 逻辑不变
                ...
            else:
                content_list.append({'type': 'text', 'text': segment})
        message['content'] = content_list
    ...
    return messages
```

tests/utils/dataset/test_rl_dataset_on_cpu.py

新增 CPU 测试覆盖，验证路径输入在 _build_messages 和 maybe_filter_out_long_prompts 中的正确处理

```
def _mock_rlhf_dataset():
    '''使用 __new__ 创建轻量 RLHFDataset 实例，避免完整初始化'''
    dataset = RLHFDataset.__new__(RLHFDataset)
    dataset.prompt_key = 'prompt'
    dataset.image_key = 'images'
    dataset.video_key = 'videos'
    dataset.audio_key = 'audios'
    dataset.processor = object() # 任意对象，仅用于跳过 None 检查
    return dataset


def test_build_messages_accepts_image_path():
    '''验证图片路径字符串能正确转换为 processor content 格式'''
    dataset = _mock_rlhf_dataset()
    image_path = 'file:///tmp/image.jpg'
    example = {
        'prompt': [{'role': 'user', 'content': 'Describe <image>'}],
```

```
        'images': [image_path],
    }
    messages = dataset._build_messages(example, key=dataset.prompt_key)
    # 断言 content 结构：图片被包装为 {'type': 'image', 'image': path}
    assert messages[0]['content'] == [
        {'type': 'text', 'text': 'Describe '},
        {'type': 'image', 'image': image_path},
    ]


@pytest.mark.parametrize(
    ('videos', 'expected_video'),
    [
        (['file:///tmp/video.mp4'], 'file:///tmp/video.mp4'),
        ([['file:///tmp/frame1.jpg', 'file:///tmp/frame2.jpg']], ['file:///tmp/frame1.jpg', 'file:///tmp/
        frame2.jpg']),
        ([[Path('/tmp/frame1.jpg'), Path('/tmp/frame2.jpg')]], ['/tmp/frame1.jpg', '/tmp/frame2.
        jpg']),
    ],
)
def test_build_messages_accepts_video_path_or_frame_list(videos, expected_video):
    '''参数化测试：验证视频路径、帧列表、包含 Path 的帧列表均能正确转换'''
    dataset = _mock_rlhf_dataset()
    example = {
        'prompt': [{'role': 'user', 'content': 'Describe <video>'}],
        'videos': videos,
    }
    messages = dataset._build_messages(example, key=dataset.prompt_key)
    assert messages[0]['content'] == [
        {'type': 'text', 'text': 'Describe '},
        {'type': 'video', 'video': expected_video},
    ]
```

评论区精华

Gemini Code Assist Bot 在 review 中指出：当 video 为 list 时，若列表内包含 os.PathLike 元素（如 pathlib.Path），下游可能因序列化或类型不兼容而出错。建议对这些元素也调用 os.fspath 统一转换为字符串。最终实现已采纳，在 video 分支中通过列表理解 [os.fspath(frame) if isinstance(frame, os.PathLike) else frame for frame in video] 处理。

- Video list 中的 PathLike 归一化 (correctness): 已采纳。最终实现中在视频帧列表分支增加了列表理解：video = [os.fspath(frame) if isinstance(frame, os.PathLike) else frame for frame in video]。

风险与影响

- 风险：风险较低。新增的路径判断分支与原有分支互斥，不影响现有 dict 和 PIL.Image 输入。测试覆盖了 image 路径、video 路径、video 帧列表以及完整过滤流程，回归风险可控。

唯一潜在风险是路径字符串可能包含非 UTF-8 字符，但 os.fspath 会正确处理，且在后续 processor 中通常会被识别为 URI。

- 影响：对用户而言，现在可以直接传入文件路径字符串（如 'file:///tmp/image.jpg'）作为 images 或 videos 字段的值，而无需先转换为 dict 或 PIL.Image。对系统而言，_build_messages 的输入输出格式保持一致，不影响下游 __getitem__和 collate_fn。对团队而言，代码可维护性提升，路径与 dict 两种输入格式统一管理。
- 风险标记：核心逻辑调整

关联脉络

- PR #6595 [data] fix: default image_patch_size to processor's real patch_size in filter_overlong_prompts: 同一文件同一功能域（filter_overlong_prompts 中的多模态处理）的先前修复，本次 PR 进一步扩展了输入类型的支持