# PR #25364 完整报告

sgl-project/sglang

Add Accuracy Benchmark for OCR models

合并时间：2026-07-06 16:18

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25364

## 执行摘要

- 一句话：为 DeepSeek-OCR-2 新增 olmOCR-bench 准确度基准测试与 HTML 报告
- 推荐动作：建议合并。该 PR 提供了有价值的基准测试框架和必要的 bug 修复，设计上考虑了可扩展性（from_cli_args 模式、异步并发、灵活报告生成）。review 中的改进已全部落实。

## 功能与动机

提供一个可复现的、标准化的 OCR 模型准确度评估工具，使用 DeepSeek-OCR-2 官方引用的 olmOCR-bench 数据集。在开发该基准测试过程中，发现并修复了 bench_serving 中未能正确加载 DeepSeek-OCR-2 处理器（DeepseekOCRProcessor）的 bug，该 bug 导致服务器在 token 计数时因传入 None 而崩溃。

## 实现拆解

1. 新增基准测试框架：添加 benchmark/ocr/bench_sglang.py，定义 BenchArgs 数据类，支持 CLI 参数，使用 aiohttp 异步并发向 sglang 服务器发送请求，渲染 PDF 页面为 base64 PNG 后发送给模型，并收集原始输出。
2. 实现评估逻辑：添加 benchmark/ocr/eval_utils.py，实现 olmOCR-bench 的 5 种测试类型：text_presence（精确和模糊匹配）、text_absence、natural_reading_order（词级别顺序匹配）、table_accuracy（支持 Markdown 和 HTML 表格）、math_formula_accuracy（基于 LaTeX 关键令牌重叠的简化评估）。还提供 normalized_edit_distance 用于额外文本质量测量。
3. 生成 HTML 报告：添加 benchmark/ocr/generate_report.py，从原始输出构建自包含的 HTML 报告，支持 MathJax 渲染 LaTeX 公式，可交互展开 / 折叠样本。
4. 修复 bench_serving 处理器加载：修改 benchmark/utils.py 中的 get_processor，调用 hf_transformers_utils.get_processor 以使用模型特定的处理器检测，而非直接 AutoProcessor.from_pretrained。此变更对所有模型安全，仅对 DeepSeek-OCR 系列生效，其他模型回退到原行为。
5. 配置更新：在 pyproject.toml 及平台变体（pyproject_cpu.toml、pyproject_npu.toml、pyproject_other.toml、pyproject_xpu.toml）的 test 依赖组中添加 pymupdf，用于 PDF 渲染。

6. 文档编写：添加 benchmark/ocr/README.md，包含完整设置、使用示例、参数说明和参考分数。

关键文件：

- benchmark/ocr/bench_sglang.py（模块 OCR 基准；类别 source；类型 core-logic；符号 BenchArgs, add_cli_args, from_cli_args, preflight_check）：新增 OCR 基准测试入口，定义 BenchArgs 和异步请求核心逻辑。
- benchmark/ocr/eval_utils.py（模块 评估逻辑；类别 source；类型 core-logic；符号 normalize_text, strip_markdown, fuzzy_contains, exact_contains）：实现 olmOCR-bench 所有测试类型的评估逻辑，包括文本存在性、阅读顺序、表格和数学公式准确度。
- benchmark/ocr/generate_report.py（模块 报告生成；类别 source；类型 core-logic；符号 _ocr_to_html, _latex_to_display, _render_sample, generate_report）：根据基准测试原始输出生成包含 MathJax 的 HTML 验证报告。
- benchmark/ocr/README.md（模块 文档；类别 docs；类型 documentation）：文档说明，包含设置、用法、参数和参考分数。
- python/pyproject.toml（模块 项目配置；类别 config；类型 configuration）：在 test 依赖组中添加 pymupdf 用于 PDF 渲染。

关键符号：BenchArgs.add_cli_args, BenchArgs.from_cli_args, run_ocr_request, normalize_text, fuzzy_contains, eval_text_presence, eval_text_absence, eval_reading_order, generate_report, _ocr_to_html

## 关键源码片段

### benchmark/ocr/bench_sglang.py

新增 OCR 基准测试入口，定义 BenchArgs 和异步请求核心逻辑。

```
# benchmark/ocr/bench_sglang.py

@dataclass
class BenchArgs:
    # 配置参数：服务器地址、模型名称、并发数、输出目录等
    port: int = 30000
    host: str = "127.0.0.1"
    model: str = "deepseek-ai/DeepSeek-OCR-2"
    split: str = "all"
    concurrency: int = 8
    output_dir: str = "./ocr_bench_results"
    max_samples: int = -1
    prompt_mode: str = "markdown"
    bench_dir: str = "./olmOCR-bench/bench_data"
    request_timeout: int = 300
    save_raw_outputs: bool = False
    render_dpi: int = 150
    debug: bool = False
```

```
    debug_accuracy: bool = False

    @staticmethod
    def add_cli_args(parser: argparse.ArgumentParser) -> None:
        # 将所有字段注册为 CLI 参数
        parser.add_argument("--port", type=int, default=BenchArgs.port)
        parser.add_argument("--host", type=str, default=BenchArgs.host)
        parser.add_argument("--model", type=str, default=BenchArgs.model)
        parser.add_argument("--split", type=str, default=BenchArgs.split,
                            choices=OLMOCR_BENCH_SPLITS + ["all"])
        parser.add_argument("--concurrency", type=int, default=BenchArgs.concurrency)
        parser.add_argument("--output-dir", type=str, default=BenchArgs.output_dir)
        parser.add_argument("--max-samples", type=int, default=BenchArgs.max_samples)
        parser.add_argument("--prompt-mode", default="markdown", choices=["markdown", "free_
        ocr"])
        parser.add_argument("--bench-dir", type=str, default=BenchArgs.bench_dir)
        parser.add_argument("--request-timeout", type=int, default=BenchArgs.request_timeout)
        parser.add_argument("--save-raw-outputs", action="store_true")
        parser.add_argument("--render-dpi", type=int, default=BenchArgs.render_dpi)
        parser.add_argument("--debug", action="store_true")
        parser.add_argument("--debug-accuracy", action="store_true")

    @classmethod
    def from_cli_args(cls) -> "BenchArgs":
        # 从 CLI 解析并返回实例（简化维护，新增参数只需在类中声明并注册 CLI）
        parser = argparse.ArgumentParser(description="OCR Benchmark (olmOCR-bench)")
        cls.add_cli_args(parser)
        args = parser.parse_args()
        return cls(**{k: v for k, v in vars(args).items() if k in cls.__dataclass_fields__})

async def run_ocr_request(
    session: aiohttp.ClientSession,
    base_url: str,
    base64_image: str,
    prompt: str,
    timeout: int,
) -> str:
    # 向 sglang 服务器发送单次 OCR 请求（使用 chat/completions 接口）
    payload = {
        "model": "default",
        "messages": [
            {
                "role": "user",
                "content": [
                    {"type": "image_url", "image_url": {"url": f"{base64_image}"}
                    },
                    {"type": "text", "text": prompt},
                ],
            }
```

```
            ],
        }
        async with session.post(
            f"{base_url}/v1/chat/completions",
            json=payload,
            timeout=aiohttp.ClientTimeout(total=timeout),
        ) as resp:
            resp.raise_for_status()
            data = await resp.json()
            return data["choices"][0]["message"]["content"]
```

## benchmark/ocr/eval_utils.py

实现 olmOCR-bench 所有测试类型的评估逻辑，包括文本存在性、阅读顺序、表格和数学公式准确度。

```
# benchmark/ocr/eval_utils.py

import re
import unicodedata
from difflib import SequenceMatcher

# 正则：统一不同 Unicode 字形的连字符、双引号、单引号
_HYPHEN_RE = re.compile(r"[\u2010-\u2015\u2212\uFE58\uFE63\uFF0D]")
_DQUOTE_RE = re.compile(r"[\u00AB\u00BB\u201C-
\u201F\u2033\u2036\u276E\u276F\u3003\uFF02]")
_SQUOTE_RE = re.compile(r"[\u2018-\u201B\u2032\u2035\u2039\u203A\u2C8D\uFF07]")
_MARKDOWN_RE = re.compile(r"(\*{1,3}|_{1,3}|`{1,3}|~~|#{1,6}\s?)")

def normalize_text(text: str) -> str:
    # 应用 NFC 归一化并统一常见标点符号
    text = unicodedata.normalize("NFC", text)
    text = _HYPHEN_RE.sub("-", text)
    text = _DQUOTE_RE.sub('"', text)
    text = _SQUOTE_RE.sub("'", text)
    return text

def strip_markdown(text: str) -> str:
    # 移除 Markdown 标记以便进行软匹配
    return _MARKDOWN_RE.sub("", text)

def fuzzy_contains(needle: str, haystack: str, threshold: float = 0.85) -> bool:
    # 先尝试精确不区分大小写匹配，若失败则使用滑动窗口 + SequenceMatcher
    needle = normalize_text(strip_markdown(needle).strip())
    haystack = normalize_text(strip_markdown(haystack))

    if needle.lower() in haystack.lower():
        return True

    n = len(needle)
```

```
    if n == 0:
        return True

    step = max(1, n // 4)
    # 步长为 1/4 长度的滑动窗口，平衡速度与精度
    for i in range(0, max(1, len(haystack) - n + 1), step):
        window = haystack[i : i + n]
        ratio = SequenceMatcher(None, needle.lower(), window.lower()).ratio()
        if ratio >= threshold:
            return True
    return False

def eval_text_presence(test: dict, ocr_output: str) -> bool:
    # 评估目标文本是否出现在 OCR 输出中，支持精确 / 模糊匹配及切片
    needle = test.get("text", "")
    max_diffs = test.get("max_diffs", 0)
    case_sensitive = test.get("case_sensitive", True)
    first_n = test.get("first_n")
    last_n = test.get("last_n")

    # 限制 OCR 输出的前 / 后 N 个词（若测试要求）
    haystack = _get_words_slice(ocr_output, first_n, last_n)

    if max_diffs == 0:
        return exact_contains(needle, haystack, case_sensitive=case_sensitive)
    # 模糊：从 max_diffs 计算相似度阈值
    n = max(1, len(needle))
    threshold = max(0.6, (n - max_diffs) / n)
    return fuzzy_contains(needle, haystack, threshold=threshold)
```

# 评论区精华

依赖位置争议：reviewer mickqian 建议将 pymupdf 移出核心依赖，作者将其移至 test 组。

命令和路径调整：polisettyvarma 要求移除 HF_HOME 硬编码，并使用 hf download 而非 huggingface-cli；yanbing-j 建议使用 from_cli_args 模式减少维护开销。

文档增强：yanbing-j 建议在 README 中增加 generate_report.py 用法说明。

代码重复：yanbing-j 指出 generate_report.py 中存在重复代码。

- pymupdf 依赖位置 (design): 作者将其移至 pyproject.toml 的 test 组下。
- 移除 HF_HOME 硬编码 (style): 作者已按要求移除。
- 使用 hf download 替代 huggingface-cli (style): 作者已更新文档。
- 采用 from_cli_args 模式 (design): 作者采纳，BenchArgs 实现了 from_cli_args 方法。
- 文档中增加 generate_report.py 说明 (documentation): 作者已补充。
- generate_report.py 代码重复 (style): 作者已修复。

# 风险与影响

- 风险：回归风险 (bench_serving)：benchmark/utils.py 中 get_processor 的修改可能影响其他模型的处理器加载，但由于其回退到 AutoProcessor.from_pretrained，风险较低；需关注 _CUSTOMIZED_MM_PROCESSOR 检查的覆盖范围。

依赖风险：新增 pymupdf 仅在 test 组，对生产环境无影响。

基准测试代码安全：新增代码不进入核心服务路径，仅作为离线基准工具使用。

- 影响：用户：获得 OCR 模型的标准评估工具，可量化改进。

开发者：便于在 CI 中集成 OCR 准确度回归测试，减少手动验证。

系统：新增的文件独立，不影响现有服务逻辑。

- 风险标记：bench_serving 回归风险较低，新增依赖仅 test 组，核心路径未受影响

# 关联脉络

- 暂无明显关联 PR