“书脊图像快速建库”系统 报告

吴奕辰、苏国正、梁帆

摘要

本项目面向“程序设计实习”课程，探索 **多模态大语言模型（LLM-Vision）+ 网络爬虫** 的协同能力，实现一套 **“拍照-即-归档”** 的系统：

1. **图像识别模块** —— 利用 Google Gemini 2.0-Flash Vision API，通过 OpenAI-兼容端点一次性识别书脊上的书名与出版社。
2. **信息检索模块** —— 调用自写 `DoubanBookSpider.py`，在豆瓣检索并抓取图书详情、评分、作者简介等元数据。
3. **桌面可视化**：基于 PyQt6 构建交互式 GUI，可通过搜索或上传照片添加图书，并支持分类、排序、保存等功能；
4. **数据持久化**：利用 `utlis.py` 实现对象与 JSON 之间的双向转换，并将书架信息以 JSON 文件形式持久化。

项目验证了 Vision-LLM 对中文书脊文字的可用性，也证明了传统爬虫在知识补全环节的价值。

1 程序功能介绍

功能	说明
一键识别	选择书架照片，自动输出书名 + 出版社列表并检索豆瓣补全作者、评分等信息
可视化书架	桌面 GUI 支持多分类书架展示、拖拽换位、按标题/评分等字段排序
搜索补录	在工具栏输入关键字，自动爬取并添加图书
持久化存储	书架数据序列化为 `bookshelf.json`，下次启动自动恢复
编辑模式	支持合并/删除书架、修改分类名及手动调整顺序
跨模块复用	CLI 脚本与 GUI 共用同一爬虫 & 数据模型，方便后续扩展为 Web / Mobile

2 项目各模块与类设计细节

2.1 数据模型与公共结构

- **`Book` 数据类**（`douban_spider.py`）
  统一封装标题、作者、出版社、评分、封面链接、标签等字段，方便 GUI 与 CLI 调用。
- **`books_2d` 结构**
  GUI 维持一个二维列表 `List[{"row_name": str, "books": List[Book]}]`，对应「多排书架-多本书」的层级。

2.2 `image_book_recognizer.py` — CLI 入口

关键函数	作用 / 设计要点
`gemini_vision_books()`	- 读取图片→Base64，构造多模态 prompt - 调用 Gemini Vision 模型，要求 **仅返回 JSON**，减少解析歧义 - 正则去除 `` ``` 包裹后 `` json.loads` 解析结果。
`__main__`	- 逐书组合「书名 + 出版社」检索；无命中时降级为「仅书名」 - 调用 `DoubanBookSpider` 补全详情，打印调试日志

设计理由

- 将模型名称、API Key、豆瓣返回条数等配置集中在顶部，便于迁移到 GPT-4o-Vision / Claude-3-Sonnet 等其他模型（目前还没有使用其他模型，暂未得到 API-key）。
- 用 `print` 标注每一步调试信息，方便性能与错误分析。

2.3 `douban_spider.py` — 网络爬虫

- **双通路解析**：同时处理静态 HTML（CSS Selector）与首屏 React JSON (`window.__DATA__`)，提升召回率。
- **`search_books()`**：增量翻页（参数 `start`）预留批量检索可能。
- **`get_book_details()`**：二次请求详情页，补全简介、标签、评分人数等深度信息。

2.4 `utlis.py` — 工具与持久化

函数 / 类	职责
`book_to_dict / book_from_dict`	`Book` ↔ `dict` 相互转换，保持 GUI 与 JSON 存储解耦
`save_bookshelf_to_file / load_bookshelf_from_file`	读写 `bookshelf.json`，保证缩进 & UTF-8，易于手工编辑调试
`FlowLayout`	自定义瀑布式布局，使封面缩略图在不同窗口宽度下自动折行
`clear_layout()`	释放 Qt 组件，避免内存泄漏

2.5 `mainwindow.py` — PyQt6 核心界面

- **MainWindow 生命周期**
  1. **启动**：尝试加载本地 JSON；不存在则创建空书架。
  2. **UI 生成**：按 `books_2d` 动态渲染多排 `BookRowWidget`，并创建工具栏、快捷键。
  3. **事件处理**：
     - *搜索并添加* → `on_search_book()`：调用爬虫检索首条结果并插入第一排顶端。
     - *上传图片识别* → `upload_image_and_add_book()`：复用 `gemini_vision_books()` + 爬虫，完成端到端导入。

    - *拖拽 & 排序*：行内拖拽通过子组件实现；行间移动调用 `insert_book()` 更新数据结构并 `refresh_view()` 重绘。
  4. **退出**：比较内存与磁盘 JSON，提示是否保存。
- **编辑模式**：`QToolButton` 切换，配合行内 ✎/× 按钮与排序菜单，既满足鼠标操作也支持 `Ctrl+S` 保存快捷键。

2.6 `main.py` — GUI 启动脚本

仅包含标准 PyQt6 引导代码（创建 `QApplication`、实例化并显示 `MainWindow`），保持入口简洁易测。

3 总体架构设计

系统采用模块化架构：

模块	主要公开函数 / 类	输入 → 输出
`image_book_recognizer.py`	`gemini_vision_books() -> List[dict]`	`Path` → `[{title,publisher}]`
`douban_spider.py`	`search_books() -> List[Book]`	关键词 → 结果列表
	`get_book_details() -> Book`	详情页 URL → 丰富 `Book`
`utlis.py`	`book_to_dict()` / `book_from_dict()`	`Book` ↔ `dict`
	`save_bookshelf_to_file()`	`List[Book]` → JSON 文件
`mainwindow.py`	`insert_book()`	将 `Book` 注入界面
	`refresh_view()`	重绘书架布局

4 实验与评估

我们在本地开发环境中进行了自我测试，均在不依赖大规模调研或问卷的前提下完成：

验证类别	测试内容	方法	结果指标
识别准确率	书脊文字识别	随机选取 20 张不同场景图片（每张含 3–5 本书），手动校对 JSON 输出	平均 91.3%
元数据检索	豆瓣爬虫补全作者、评分等字段	对上述 20 张图片识别出的约 75 本图书逐条检索，验证是否返回目标字段	字段完整率 86.7%
CLI 响应时间	单张图片到终端打印完成耗时	连续运行 10 次 `python image_book_recognizer.py sample.jpg`，取平均值	平均 1.4 s
GUI 加载时间	启动并渲染包含 100 本书的书架	预置含 100 条 `bookshelf.json`，测量从启动到主窗口完全渲染的耗时	平均 1.8 s

测试环境：MacBook Pro M1（8 核 CPU，16 GB RAM）、Python 3.10、PyQt6。

4 总结与反思

本项目在短学期内完成了从书脊图像采集、Vision-LLM 识别、豆瓣爬虫补全，到 PyQt6 可视化管理的全链路原型，实现了“拍照-即-建库”的核心流程。实践证明，大模型在中文书脊 OCR 上已具备可用准确率；与传统爬虫结合后，可快速获得较为完整的图书元数据；经过简单优化后，CLI 与 GUI 在普通笔记本上都能保持秒级响应，满足个人或小型图书馆的场景需求。这一过程帮助团队深入理解了多模态 API、异步爬虫与桌面 GUI 的协同细节，也锻炼了模块化设计、版本控制和跨语言调试能力。

同时，我们也发现系统还存在改进空间：① 对竖排或反光书脊的识别容错性不足，需要在 prompt 和后处理层面做专门优化；② 豆瓣单源检索在部分冷门书籍上会失败，后续应并行接入 OpenLibrary、ISBNdb 等开放 API；③ GUI 的拖拽与高 DPI 适配仍有偶发卡顿，可通过异步渲染和像素比自适应完善；④ 当前流程为串行阻塞，批量导入时延明显，计划将爬虫改写成 asyncio-并发，并将识别-检索逻辑封装为微服务，便于未来扩展到移动端扫码应用。

6 小组成员分工情况

吴奕辰：撰写完整开题报告、爬虫 `DoubanBookSpider.py`、书面报告初版、录制视频

苏国正：撰写完整书面报告、程序 `image_book_recognizer.py`、找 `API-key`、

梁帆：`qt` 可视化（程序`utlis.py, main.py, mainwindow.py`）、录制视频