

Web_document_crawling-file_format_conversion

Web legal document crawling and file format conversion tools

法律文档爬取与转换工具

一个自动化爬取香港司法机构网站法律文档并将其转换为结构化Markdown的工具。

功能特性

- ☒ 自动爬取香港司法机构网站的法律文档(PDF/DOCX)
 - ☒ 自动将DOCX转换为PDF
 - ☒ 通过MinerU API将PDF转换为结构化Markdown
 - ☒ 支持多种搜索条件(案件号、当事人、判决日期等)
 - ☒ 自动创建分类文件夹保存结果
-

快速开始

前置要求

- Python 3.8+
 - Chrome浏览器
 - ChromeDriver (与浏览器版本匹配)
 - 技术栈
 - Python + Selenium
 - Flask + Requests
 - MinerU API
-

提示:

tips1. MinerU本地化部署可以参考: <https://zhuanlan.zhihu.com/p/1908942870666282723> 、
[https://www.bilibili.com/video/BV1zCXhYbE2p/?spm_id_from=333.337.search-card.all.click&](https://www.bilibili.com/video/BV1zCXhYbE2p/?spm_id_from=333.337.search-card.all.click&vd_source=c414af4780348b902d1d9d8bf1ba7692)、 [MinerU教程第二弹 | MinerU 本地部署保姆级“喂饭”教程](#)

tips2. 本地部署后的API化需要下载MinerU官网包,克隆地址: <https://github.com/opendatalab/MinerU.git> ,API化可参考上面的csdn博客 (按照视频操作会有问题) 。

tips3. 本人只用了一个conda环境 (MinerU) ,且为了保护我的本地tourtch环境, 只用了CPU进行训练。本机CPU:笔记本版的i7-12400H+32G运行内存。

tips4. 如果不需要配置本地API环境, 只想单个转换文件格式 (或者是demo), 可以直接在cmd(管理员模式)下运行conda环境后:

```
cd “要转格式的pdf文件所在文件夹”  
magic-pdf -p 要转格式的文件.pdf -o ./output
```

tips5. MinerU框架

MinerU Readme 地址 (中文) : [github.com/opendatalab/...](https://github.com/opendatalab/MinerU)

敲黑板! MinerU 仓库中有些目录虽然常被开发者忽视,但它们对于理解项目架构与功能扩展却至关重要。以下是对部分关键文件夹的简要介绍:

```
MinerU/  
├─ demo/           # 用于运行转换演示的脚本  
├─ docker/         # 用于容器化的 Dockerfile+ 配置文件  
├─ docs/           # 存储各类说明文档  
├─ projects/       # 存放由 MinerU 衍生或相关的项目  
│   ├─ gradio_app/ # MinerU Gradio 界面+的源代码  
│   ├─ multi_gpu/  # 为 MinerU 提供多 GPU 支持+的解决方案  
│   └─ web_api/    # 提供本地 Web API+ 接口的服务端代码
```

tips6. 全文的所有操作都需要管理员权限。

安装步骤

1. 克隆仓库:

```
git clone https://github.com/yourusername/legal-doc-scraper.git  
cd legal-doc-scraper
```

2. 安装依赖:

```
pip install -r requirements.txt
```

3. 下载并配置ChromeDriver:

- 下载地址: <https://chromedriver.chromium.org/>
- 将chromedriver.exe放在项目目录或系统PATH中

使用方法

1. 运行:

需要先配置[MinerU本地运行环境](#)后, 再配置 [本地MinerU API环境](#)

1.1运行本地API环境:

[运行本地API环境](#)

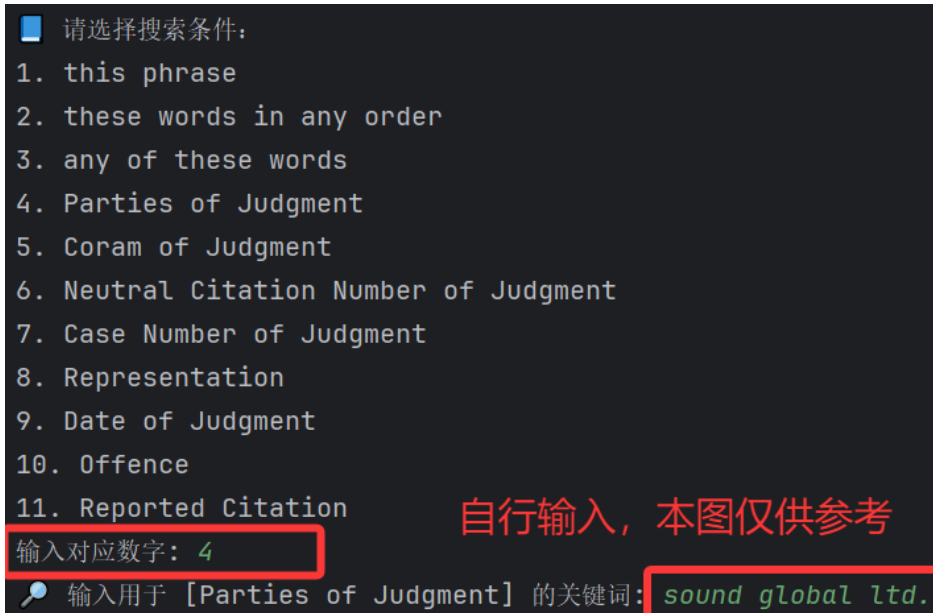
1.2运行主程序

方法1: 在管理员模式下打开pycharm, 在里面直接按F5运行main函数。(我是这个)

方法2: 在终端运行python代码:

```
python main.py
```

2. 按照提示选择搜索类型和输入关键词



■ 请选择搜索条件:

1. this phrase
2. these words in any order
3. any of these words
4. Parties of Judgment
5. Coram of Judgment
6. Neutral Citation Number of Judgment
7. Case Number of Judgment
8. Representation
9. Date of Judgment
10. Offence
11. Reported Citation

输入对应数字: 4

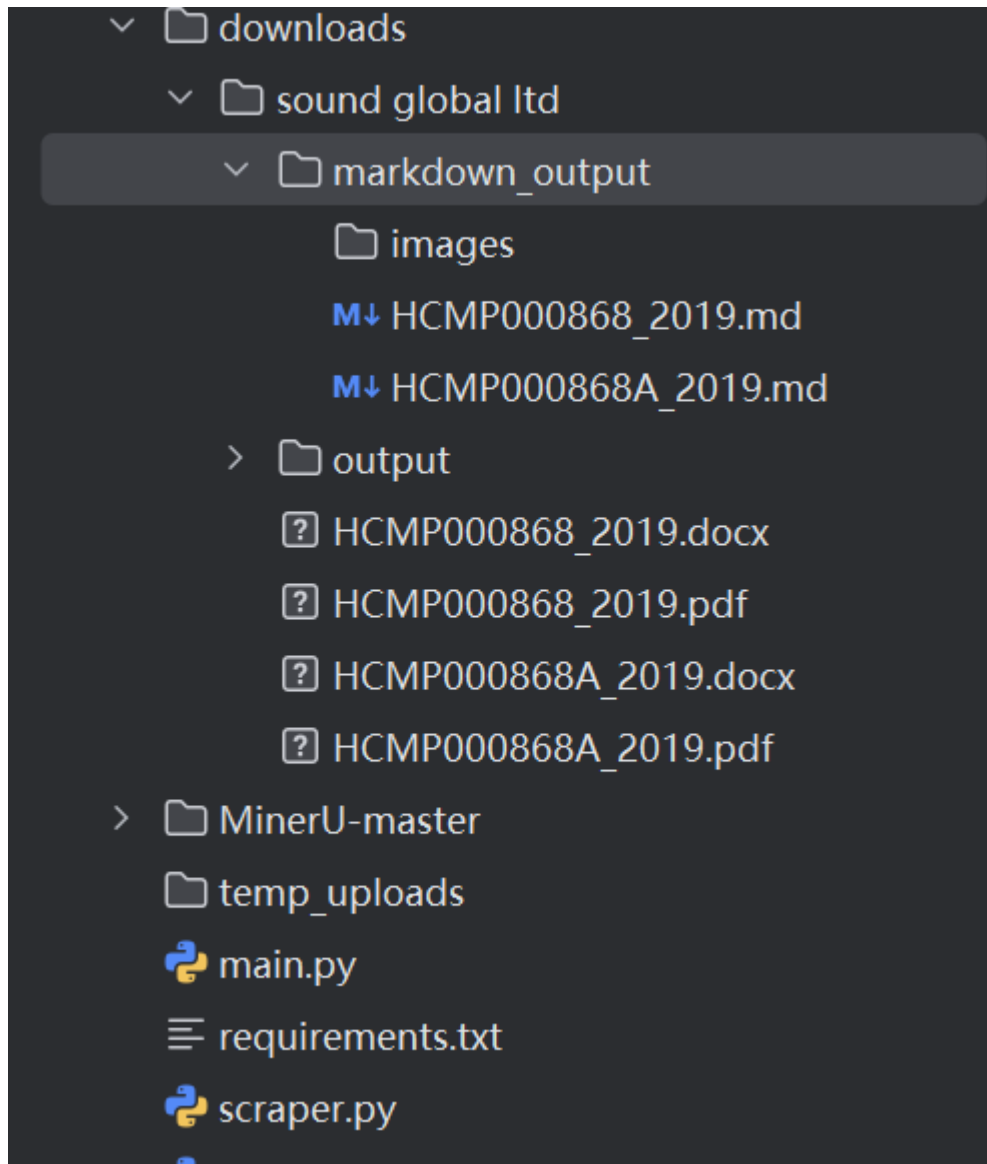
输入用于 [Parties of Judgment] 的关键词: sound global ltd.

自行输入, 本图仅供参考

3. 结果将保存在downloads/[关键词]目录下, 包括:

- 原始文档(PDF/DOCX)
- 转换后的Markdown文件
- API返回的JSON数据(可选)

4. 代码文件夹示例:



项目结构

```

Web_document_crawling-file_format_conversion/
├── downloads/                # 下载文件存储目录
│   └── /                    # 文件夹名称为查询的关键词，按关键词搜索之后会自动保存网页
里有的文件
│   └── markdown_output/    # 文件转化为Md文件之后所储存的位置
├── scraper.py               # 主爬虫逻辑
├── main.py                  # 用户交互入口
├── requirements.txt         # Python依赖
├── README.md                # 项目文档
├── MinerU-master            # 从github上克隆下来的MinerU官网包、克隆地址：
https://github.com/opendatalab/MinerU.git
└── examples/                # 示例输出

```

API配置

参考链接: [MinerU教程第二弹 | MinerU 本地部署保姆级“喂饭”教程](#)

1. 配置本地MinerU API环境:

创建conda环境, 或者跟我一样使用现有的MinerU环境。

二选一:

```
conda create -p J:\Anaconda3\envs\MinerU 'python=3.12' -y # 使用指定地址建立环境
```

```
conda create -n MinerU 'python=3.12' -y # 使用默认地址建立环境
```

因为我的C盘空间不多, 所以在建立conda环境的时候, 使用了指定地址。将"J:\Anaconda3\envs"更换为你自己的地址即可,

2. 安装依赖

```
# 进入从github clone的MuierU-master的api目录
```

```
cd projects/web_api
```

```
#安装依赖
```

```
pip install -r requirements.txt
```

3. 启动服务

```
python app.py
```

使用MinerU API转换PDF, 请确保:

- 1. MinerU服务运行在<http://localhost:8888>
- 2. 或修改`scraper.py`中的`mineru_api_url`变量
- 3. 确保文件格式是pdf。如果是word格式, 请先转为pdf格式。

成功范本

[成功输出结果示例](#)

如果你不小心关闭了 MinerU 的 API 服务 (例如你按了 **Ctrl+C** 或关了窗口), 别担心, **只需要重新启动服务即可**。

✅ 重新启动方法（用的是 CPU 模型，Windows 系统）：

启动服务

1. 在管理员模式下打开 Anaconda Prompt 或 CMD
2. 激活之前用于 MinerU API 的环境（如我的是 MinerU）：

```
conda activate MinerU
```

3. 进入 MinerU 的 API 目录：

```
cd F:\Web_document_crawling-file_format_conversion\MinerU-master\projects\web_api
```

4. 启动 API 服务：

```
python app.py
```

你会看到：

```
Uvicorn running on http://0.0.0.0:8888
```

说明服务又启动成功了 ✅

常见问题

Q: ChromeDriver版本不匹配 A: 请确保ChromeDriver版本与已安装的Chrome浏览器版本一致

Q: DOCX转换PDF失败 A: 确保已安装Microsoft Word或LibreOffice

Q: API请求超时 A: 增大scraper.py中的timeout值(CPU训练默认900秒，GPU训练时间会更短可自行设定)

贡献指南

欢迎提交Pull Request！请确保：

1. 代码符合PEP8规范
2. 添加适当的单元测试
3. 更新相关文档

许可证

MIT License