LLM x 书签收藏：摘要 & 全文索引

Oct 07, 2024

背景

网上冲浪的时候，经常会遇到一些有趣的文章或者网站，让我有收藏起来以备后用的冲动（虽然绝大部分情况下都没有再用过）。然而一个人收藏未免有些太孤单了，因此自从 2021 年 5 月以来，我一直在使用一个名为 osmos::memo 的书签插件，将我的收藏直接记录到一个公开的 Github 存储库。这个插件的工作原理很简单，首先设置好 Github 的 token，然后每次点击收藏按钮都会在浏览器里临时 clone 一次，追加新收藏的条目到文件顶部，生成 commit 并提交，然后推送回 Github。但是这样简单的工作流程也十分有效，除了 token 过期的时候需要手动续期（过期前有 Github 自带的邮件提醒，所以基本上不会拖到最后），没有什么可以出错的空间，近三年半下来也已经帮我积累了 800+ 条目了。

问题

然而目前的书签收藏流程中，依然会存在一些问题。

1. 书签指向的 URL 可能不再存在（例如某个博客的主人决定不再续费域名，或者是做了链接格式的调整），导致成为悬空的死链接
2. 目前的记录项只有书签的 URL、标题和可选的标签（而且我打标签的习惯不太好，光靠标签基本上不太能找到），导致查找的时候如果对关键词记忆不清楚，很有可能找不到
3. 书签里一大部分是长文章，时间一久很有可能忘记内容，如果只是临时找些东西，通读一次又略微有些费时费事，导致查找&引用效率下降。

解决

为了解决这些问题，我建立了一个新的存储库 bookmark-summary。这个存储库可以视为现有书签存储库的辅助数据，其中包含了新增书签的 Markdown 格式全文、列表摘要、一句话总结，和现有存储库之间通过 Github Actions 联动。其工作原理如下：

1. 我通过书签插件，在现有的书签存储库中新增了一个条目
2. 代码提交到主干，触发名为 `summarize` 的 Github Actions（yml 工作流文件）
3. Github Actions 执行，首先 checkout 书签存储库和摘要存储库，然后执行 process_changes.py
    1. 首先解析书签 README.md 文件，找到最近新增的条目标题和 URL

2. 将 URL 保存到 Wayback Machine
3. 输入 URL，使用 jina reader API 获取网址的 Markdown 全文，并保存到 `YYYYMM/{title}_raw.md`
4. 输入 URL，使用 LLM 生成列表摘要（prompt 在 `summarize_text` 函数 link）
5. 输入列表摘要，使用 LLM 生成一句话总结
6. 将列表摘要和一句话总结保存到 `YYYYMM/{title}.md`（效果示例）
7. 更新摘要存储库的 README.md，增加到摘要文件的链接

4. Github Actions 提交变更到摘要存储库

这里的主要代码基本都是 Claude 和 GPT4o 写的，人肉做了一些小调整。后面随着使用又逐步发现了一些 bug，最近还用 o1-mini 修复了一个，算是真切感受到了 LLM 对生产力的巨大提升。目前摘要生成用的是深度求索的 deepseek-chat，便宜是真便宜（输入 1元/1M token，输出 2元/1M token，在这个场景下的成本基本上是每个月1元不到），效果也还算可以。

未来

最后是一些已知问题，以及未来可能的优化方向。当然和其他所有项目一样，欢迎 fork & PR。

1. 列表摘要质量：可能是 prompt 的问题，列表摘要倾向于每个大点下面只列两个小点，且没有充分合并需要合并的论点；可能需要考虑进一步优化 prompt，或者换用其他模型（不过我拿便宜的模型都试了一轮，基本上都存在类似问题）
2. 数据结构化：目前摘要存储库下有个简单的 data.json，但是核心的摘要和全文内容依然是 Markdown 存储的，而不是 JSON 这类程序友好的结构化存储。可能需要考虑在 Markdown 之外另外维护一个 JSON，以备未来的查询。
3. 代码整理和重构：目前所有逻辑都混在一个大的 Python 文件里，修改和测试起来都很烦人（实际上没有特别好的办法手动测试，目前都得靠手动注释掉部分代码）。未来一个考虑是做重构（o1-mini也给出过比较好的重构结构）+补充测试；另一个是改进书签存储库和摘要存储库的交互方式，例如通过读 git log 或者是明确传递最近书签条目的方式来触发摘要生成，而不是靠目前读文件对比的方式
4. 向量搜索：目前虽然原文和摘要都存下来了，搜索却还是只能靠基本的文本匹配；可能可以考虑接个 embedding 模型自动生成下嵌入，存到一个 SQLite 数据库（或者用各种向量数据库 as a Service）；主要是查询的时候也得生成 embedding，英文还有小模型可以搞，中文的模型都太大了，没法直接在前端跑不依赖后端服务，这里还得再仔细想想。

5. 自动生成每周周报：~~既然现在书签有时间信息，可以考虑每周新增的书签+原文+摘要全部往 LLM 扔，自动生成一个每周摘要，放在 Github Release 里（不过不知道有没有人愿意看就是了）~~ (已完成，参见 Releases，实现见 build_weekly_release.py，代码主要由 o1-mini 实现)
6. 改用更现代的工具链：例如 uv，以及把依赖写在 Python 代码头部（PEP 723 Inline Script Metadata）？

我也想要

可以参考以下步骤，在自己的 Github 账户下部署一套类似的系统。（根据回忆写的，所以可能不太详尽）

1. 参考 osmos::memo 的指引，初始化书签存储库（我的叫做 bookmark-collections），安装浏览器插件，并连接到 Github
2. 新建一个摘要存储库（我的叫做 bookmark-summary）
3. 将 process_changes.py 添加到摘要存储库，用实际的存储库名修改 `BOOKMARK_COLLECTION_REPO_NAME` 和 `BOOKMARK_SUMMARY_REPO_NAME`；如果需要的话，可以调整 `summarize_text` 和 `one_sentence_summary` 中的 prompt
4. 回到书签存储库，将 bookmark_summary.yml 添加到 `.github/workflows/bookmark_summary.yml`，用 `Github账号/摘要存储库名` 替换 27 行 `jerrylususu/bookmark-summary`
5. 新建一个 PAT（Personal Access Token）
    - 入口：Github 主页 - 右上角 Settings - 左侧列表底部 Developer Settings - 左侧列表 Personal Access Token / Fine-grained Tokens - 右侧 Generate New Token - 验证密码
    - Token Name: 随便写
    - Expiration：可以长一些，但是不能超过 1 年
    - Repository access：选 Only select repositories，然后在下面选中自己的摘要存储库
    - Permissions：点开 Repository Permissions，找到 Contents，选择 Read and write；其他不用动
    - 点击底部 Generate Token；Token 只会显示一次，复制下来保存好
6. 回到书签存储库，添加密钥到环境变量
    - 入口：摘要存储库 - 顶部 Settings - 左侧 Secrets & Variables / Actions - Repository secrets - New Repository Secret

- 需要添加 4 个（其实有的可以放在 Environments 里，不过这里我为了方便先全放到 Secrets 里了）；冒号前面的是名字，冒号后面的是内容

1. PAT： 第 5 步生成的 token
2. OPENAI_API_MODEL： 模型名，如 gpt-4o-mini；如果像我一样用 deepseek 则填写 deepseek-chat
3. OPENAI_API_KEY： API key，通常以 sk- 开头
4. OPENAI_API_ENDPOINT： 模型 API 地址，留空默认用 OpenAI 官方；可以填中转站；用 deepseek 则填写 `https://api.deepseek.com/chat/completions`

7. 至此应该配置完成了。可以用 osmos::memo 扩展添加一个书签试试，观察书签存储库中工作流是否正常运行，摘要存储库中是否生成了对应的摘要。

另请参阅

- 2024 年了，我最近在用什么工具
- 我的 AI Prompt
- 用 mitmproxy 让 ChatGLM 适配 OpenAI 接口

0 个表情

2 条评论 · 1 条回复 – *由 giscus 提供支持*

最早 最新

输入 预览 Aa

登录后可发表评论

使用 GitHub 登录

VmzXv 22 小时前

感谢博主的分享！按照步骤试了一下，发现 http://web.archive.org 暂停服务，导致重试时间很长。

```
2024-10-11 07:53:49 - process_changes.py:29 - wrapper - Entering process_bookmark_file
2024-10-11 07:53:49 - process_changes.py:29 - wrapper - Entering submit_to_wayback_machine
2024-10-11 07:56:04 - connectionpool.py:870 - urlopen - Retrying (Retry(total=4, connect=None, r
2024-10-11 07:58:18 - connectionpool.py:870 - urlopen - Retrying (Retry(total=3, connect=None, r
2024-10-11 08:00:34 - connectionpool.py:870 - urlopen - Retrying (Retry(total=2, connect=None, r
2024-10-11 08:02:53 - connectionpool.py:870 - urlopen - Retrying (Retry(total=1, connect=None, r
2024-10-11 08:05:16 - connectionpool.py:870 - urlopen - Retrying (Retry(total=0, connect=None, r
```

↑ 1

1 条回复

jerrylususu 16 小时前 所有者

的确，最近有黑客在攻击 wayback machine，可以先注释掉对 `submit_to_wayback_machine` 方法的调用