项目简介

该项目是一个网络爬虫，设计用来从各种网站（包括汽车和社交网站）抓取数据，如作者信息、互动数据等。数据将被存储为JSON格式，用于后续的数据分析和处理。

所需库和工具

在开始运行项目之前，你需要安装一些Python库和工具。这些库和工具可以在终端通过 pip 工具安装，或者在 IDE 中安装。

Python 库：

- requests: 用于发送HTTP请求
- beautifulsoup4: 用于解析HTML页面并提取所需信息
- concurrent.futures: 用于并行处理多个任务
- tqdm: 用于显示进度条
- selenium: 用于模拟浏览器行为抓取动态网页
- re: 正则表达式库，用于处理和匹配字符串
- json: 用于处理JSON数据
- asyncio: 用于处理异步I/O操作
- mysql: 用于储存数据
- mysql-connector-python: mysql 驱动
- 安装示例：pip install beautifulsoup4

工具：

- Python 3: 我们的代码是用Python 3编写的，因此需要在你的机器上安装Python 3。
- Firefox: 我们的代码使用了Firefox浏览器的自动化工具，因此你需要安装Firefox浏览器。
- Geckodriver: 用于驱动Firefox浏览器。

```
wget https://github.com/mozilla/geckodriver/releases/download/v0.23.0/geckodriver-v0.23.0-linux64.tar.gz
```

- 具体命令请根据使用的版本以及操作系统修改。
- 参考：https://github.com/mozilla/geckodriver/releases

- MySQL: 用于储存爬取的数据。

IDE：

你可以选择任何你喜欢的Python IDE，例如PyCharm、VS Code、Jupyter Notebook等。

运行方法

在安装了所有必要的库和工具后，你可以按照以下步骤来运行项目：

- 解压压缩包
- 在终端中导航到你的 crawling目录下
- 根据环境安装 修改 runall.sh 文件 “python3 "$script" &
- ”代码，（pip 安装的环境就改为 python，pip3 就改为 python3）
- 使用“bash runall.sh”命令运行
- 等待程序运行完成。完成后，你可以在各个平台的文件路径下找到输出的JSON文件。
- 数据会保存到mysql 中。

数据存储

1. 我们的程序将抓取的数据保存为JSON格式，并存储在一个文本文件中。每个数据项（例如一个网页的信息）被保存为一个JSON对象，每个对象占一行。这样的格式方便后续的数据处理和分析。注意：如果某一行的结果为 null，可以过几分钟重新运行一次程序。如果某一列为的结果为 null，常见原因为平台网页数据储存方式更新或者帖子被删除，具体需要人工分析原因并维护代码。

2. 数据会保存到 mysql 中的 crawling_data的数据库中。之后会把数据写入平台对应表格。如果无该数据库，或者平台表格，脚本会自动创建。数据的入表时间也会记录在数据库中。

3. 如果需要增减表格保存的数据内容，需要先删除原有表格或者新建表格名字，否则会报错。

检查操作：

登录 mysql：mysql -uroot -p

输入密码 （dasheng202307）

use crawling_data;

show tables;

可以使用 mysql 命令来调取。

具体爬取的数据包括：

- 平台：　平台的名称，例如“小红书”、“易车”等。
- 浏览数：　文章的浏览数量（只适用于某些平台）。
- 点赞数：　文章的点赞数量。
- 收藏数：　文章的收藏数量（只适用于某些平台）。
- 分享数：　文章的分享数量（只适用于某些平台）。
- 评论数：　文章的评论数量。
- 文章：　文章的URL链接。
- 作者：　作者的名字（平台昵称）。
- 作者ID：　作者的ID（只适用于某些平台）。
- 作者主页：作者的主页URL（只适用于某些平台）。

如需增加爬取数据需根据平台的网页规则进行扩展。

注意事项

- 确保你的机器上安装了所有必要的库和工具。
- 在运行项目时，确保你的机器上有足够的网络连接。
- 由于网络问题或目标网站的防爬机制，爬虫可能无法获取所有的信息。如果遇到这种情况，你可以尝试调整爬虫的行为，例如增加延迟，改变请求头等。

读取方式

每个平台的文件夹下的脚本会读取各自平台的 url 文件，请根据平台分类储存要爬取的网页。

Selenium与Requests的使用及其不同点

Requests + Headers

使用requests库是发送HTTP请求的一种常见方式。在这个项目中，我们使用requests库发送GET请求来获取网页的HTML内容。为了模拟浏览器行为，我们还在请求头中添加了一些信息（例如User-Agent）。这种方法的优点是简单快速，因为它直接获取了网页的源代码。然而，这种方法可能无法处理一些复杂的情况，例如JavaScript生成的动态内容。

使用requests库发送请求时，需要注意以下几点：

- 有些网站可能会检查请求头中的User-Agent，如果它看起来像是由机器生成的，网站可能会拒绝请求。因此，你应该在请求头中设置一个看起来像是由浏览器生成的User-Agent。
- 有些网站可能会使用cookies来追踪用户，如果你的请求中没有正确的cookies，网站可能会拒绝你的请求。在这种情况下，你需要使用requests库的session对象来处理cookies。
- 可以使用https://curlconverter.com/ 一键生成 header 和 cookie，具体使用方式参考：https://zhuanlan.zhihu.com/p/518788491

Selenium

Selenium是一种自动化测试工具，它可以模拟真实的浏览器行为，例如点击按钮，滚动页面，填写表单等。在这个项目中，我们使用Selenium来处理那些不能通过直接发送HTTP请求获取到的动态内容。Selenium会启动一个真实的浏览器，并按照我们的指示操作这个浏览器。这种方法的优点是可以处理任何复杂的网页，无论它是静态的还是动态的。然而，这种方法的缺点是速度较慢，因为它需要启动和操作真实的浏览器。

使用Selenium时，需要注意以下几点：

- 由于Selenium会启动一个真实的浏览器，因此它需要更多的资源，包括CPU和内存。如果你需要同时处理很多任务，你可能需要考虑资源的使用。

- 有些网站可能会检查用户的行为，如果它看起来像是由机器生成的，网站可能会封锁你的IP。因此，你应该尽可能地让你的行为看起来像是由人类生成的，例如增加随机的延迟，不连续地滚动页面等。
- 在关闭Selenium的浏览器或驱动程序时，你应该确保所有的资源都被正确地清理掉，否则它们可能会占用大量的系统资源。

asyncio

Asyncio 是 Python 用于编写单线程并发代码的库，使用事件循环驱动的协程。Asyncio 可以用于IO-bound任务，例如网络请求，其中大部分时间都在等待。在这个项目中，Asyncio 用于管理和调度各种 URL 的爬取任务。

concurrent.futures

concurrent.futures 库提供了一个高级接口，用于异步执行可调用对象。concurrent.futures 支持多进程和多线程，这在处理 CPU-bound 任务时非常有用。在这个项目中，我们使用 ThreadPoolExecutor 创建一个线程池，然后使用 map 或 submit 方法来提交任务到线程池。

asyncio 和 concurrent.futures 的区别

Asyncio 是基于单线程的异步IO模型，适用于 IO-bound 任务，如网络请求，文件读写等，可以通过并发的方式提高程序的执行效率。

Concurrent.futures 是多线程/多进程的模型，适用于 CPU-bound 任务，如计算密集型任务。多线程/多进程可以利用多核CPU的优势，提高程序的执行效率。

注意事项

当你使用 asyncio 和 concurrent.futures 时，要注意管理好你的资源。例如，你应该确保在任务完成后关闭你的事件循环或线程池，否则它们可能会占用大量的系统资源。

由于 Python 的全局解释器锁（GIL）的限制，Python 的多线程并不能真正意义上地并行执行，而是在同一时间内只允许一个线程执行。这意味着，对于 CPU-bound 任务，使用多线程可能并不会带来太大的性能提升。在这种情况下，你可能需要使用多进程或其他方式来提高你的程序性能。

在使用 asyncio 时，你应该尽可能地使用非阻塞的操作。例如，你应该使用 aiohttp 而不是 requests 来发送 HTTP 请求，使用 asyncio.sleep 而不是 time.sleep 等等。否则，你的阻塞操作可能会阻塞整个事件循环，导致你的程序性能下降。

当你使用 concurrent.futures 的 ThreadPoolExecutor 时，你需要注意线程的数量。创建太多的线程可能会导致大量的上下文切换，降低你的程序性能。此外，线程的数量也不应该超过你的 CPU 核心数量，否则可能会造成 CPU 的过度使用。