
















 neil-zt /
common-crawl-client





 Code  Issues  Pull requests  Actions  Projects  Wiki  Security  Insights






A Common Crawl client example for scraping specific websites.



☆ 0 stars  0 forks  2 watching  Activity

 Public repository



 main ▾



 Branches  Tags

 neil-zt ... now 

[View code](#)

 README.md 

使用 Common Crawl 的網頁資料 | Access Web Archive Data with Common Crawl Client

此 ([neil-zt/common-crawl-client](https://github.com/neil-zt/common-crawl-client)) 為一個公開的 GitHub repository，IASL 如果有人需要使用可以直接複製，有問題請聯絡 neil.lu@duke.edu。

For English instructions, go to `driver.ipynb` .
本 README 檔案內容可在 `driver-zh.ipynb` 中執行。

環境建置

安裝 Dependencies

```
pip install pandas
pip install comcrawl
pip install beautifulsoup4
```

更新 comcrawl 中的程式

Common Crawl 組織在過去幾年內更新了他們的檔案下載連結，但 `comcrawl` 似乎因沒有被更新而無法下載 Common Crawl 的資料。請前往 `comcrawl` 函式庫所謂在的資料夾（如果使用的是 VS Code，可按住 `ctrl` 或 `command`，將滑鼠移動到 `from comcrawl import IndexClient` 中的 `comcrawl` 上並點擊）。接著請開啟 `utils/download.py`，並將下列的程式更新到對應的函式/變數名稱上：

- download_multiple_results

```
def download_multiple_results(results: ResultList, threads: int = None) -> ResultList
    # multi-threaded download
    if threads:
        multithreaded_download = make_multithreaded(download_single_result, threads)
        results_with_html = multithreaded_download(results)
    # single-threaded download
    else:
        for result in results:
            success = False
            while not success:
                try:
                    result_with_html = download_single_result(result)
                    results_with_html.append(result_with_html)
                    success = True
                except Exception as e:
                    print("Library Error: download_single_result failed, retrying...")
            return results_with_html
```

- URL_TEMPLATE

```
https://data.commoncrawl.org/{filename}
```

使用方法

1. 指定資料的位置與時間

在下列設定 time_code 和 searching_uri。time_code 可從 Common Crawl 的[官網列表](#)獲得。

```
time_code = "2022-05"
searching_uri = "www.cna.com.tw/news/afe/*"
```

2. 載入需要的函式庫

```
from comcrawl import IndexClient
import pandas as pd
import os
```

3. 建立下載後存放資料的資料夾

```
searching_uri_dir = searching_uri.replace("/", "-").replace("*", "-all")
if not os.path.exists("output"):
    os.makedirs("output")
if not os.path.exists(f"output/{time_code}"):
    os.makedirs(f"output/{time_code}")
if not os.path.exists(f"output/{time_code}/{searching_uri_dir}"):
    os.makedirs(f"output/{time_code}/{searching_uri_dir}")
```

4. 獲取目標在 Common Crawl 資料庫中的 Index

需要注意的是，Common Crawl Index Server 時常有著龐大的負載，因此多數時候會發生 504 timeout 的問題。此處的解決方法是不斷嘗試呼叫其伺服器直到收到回覆為止。

```
client = IndexClient([time_code], verbose=True)
success = False
while not success:
    try:
        client.search(searching_uri)
        client.results = (pd.DataFrame(client.results)
                          .sort_values(by="timestamp")
                          .drop_duplicates("urlkey", keep="last")
                          .to_dict("records"))
        pd.DataFrame(client.results).to_csv(f"output/{time_code}/{searching_uri_dir}")
        success = True
    except:
        print("Index Server Response Timeout. Retrying...")
```

5. 下載目標檔案

這個步驟可能會要下載數千至數十萬個網頁資料，除了需要預先確認記憶體大小能夠負荷外，另請注意其可能花費數個小時。

```
success = False
while not success:
    try:
        client.download()
        success = True
    except:
        print("Download Server Error. Retrying...")
```

6. 將網頁中的文章解析出來

下列提供的為很通略的文章擷取方法。如果要爬特定的網頁，可能需要特別設計擷取的步驟。前往 `/web_parse.py` 以編輯之。

```
from web_parse import extract_main_paragraphs
for result in client.results:
    extracted_text = extract_main_paragraphs(result["html"])
    if len(extracted_text) < 100:
        continue
    with open(f"output/{time_code}/{searching_uri_dir}/{result['urlkey'].replace('/', '')}"):
        f.write(extracted_text)
```

Releases

No releases published

[Create a new release](#)

Packages

No packages published


[Publish your first package](#)

Languages

Jupyter Notebook 96.6% Python 3.4%

Suggested Workflows


Based on your tech stack



Actions Importer

Set up


Automatically convert CI/CD files to YAML for GitHub Actions.



Python application

Configure

Create and test a Python application.



Django

Configure

Build and Test a Django Project

[More workflows](#)

Dismiss suggestions