ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

MẬT MÃ VÀ AN NINH MẠNG

BÀI TẬP MỞ RỘNG CRAWLER

Giảng viên hướng dẫn: Nguyễn Đức Thái
Sinh viên: Lê Thành Sơn - 1810481

Tp. Hồ Chí Minh, Tháng 6/2022

Mục lục

Danh sách hình vẽ

1 Tổng quan

Crawler, cụ thể là web crawler, giúp ta có thể thu thập thông tin một cách tự động từ website, lấy ra những thông tin mà ta muốn thu thập. Ở đây, ta sẽ thu thập các link dẫn đến file có định dạng mong muốn và tải về chúng một cách tự động vào máy.

Đối với crawler đã thực hiện, ta sẽ chỉ xem xét các link có thể truy cập được ngay từ giao diện của web. Nghĩa là các file có thể được tải về thông qua một nút bấm. Một ví dụ có thể được tìm thấy ở Hình 1. Ta có lựa chọn vào các nút `pdf` để tải về file pdf tương ứng mà không phải làm gì thêm. Do đó, các link được sinh ra bởi các script được viết bằng JavaScript sẽ bị bỏ qua. Ngoài ra, ta cũng chưa xem xét lấy các file nằm trên nhiều phân trang khác nhau.

[1] **arXiv:2206.04561** [**pdf**, **ps**, **other**]

Functional Code Building Genetic Programming
Edward Pantridge, Thomas Helmuth, Lee Spector
Subjects: **Artificial Intelligence (cs.AI)**; Neural and Evolutionary Computing (cs.NE)

[2] **arXiv:2206.04513** [**pdf**, **other**]

AAM-Gym: Artificial Intelligence Testbed for Advanced Air Mobility
Marc Brittain, Luis E. Alvarez, Kara Breeden, Ian Jessen
Comments: 10 pages, accepted for publication in 2022 IEEE/AIAA Digital Avionics Systems Conferer
Subjects: **Artificial Intelligence (cs.AI)**; Signal Processing (eess.SP)

[3] **arXiv:2206.04460** [**pdf**, **other**]

Open ERP System Data For Occupational Fraud Detection
Julian Tritscher, Fabian Gwinner, Daniel Schlör, Anna Krause, Andreas Hotho
Subjects: **Artificial Intelligence (cs.AI)**

[4] **arXiv:2206.04438** [**pdf**]

A taxonomy of explanations to support Explainability-by-Design
Niko Tsakalakis, Sophie Stalla-Bourdillon, Trung Dong Huynh, Luc Moreau
Subjects: **Artificial Intelligence (cs.AI)**; Computers and Society (cs.CY)

[5] **arXiv:2206.04269** [**pdf**, **other**]

Smart System: Joint Utility and Frequency for Pattern Classification
Qi Lin, Wensheng Gan, Yongdong Wu, Jiahui Chen, Chien-Ming Chen
Comments: ACM Transactions on Management Information Systems. 10 figures, 7 tables
Subjects: **Artificial Intelligence (cs.AI)**; Databases (cs.DB)

Hình 1: Giao diện của trang export.arixv.org

2 Thư viện sử dụng

2.1 Selenium

Selenium là thuật ngữ bao hàm số lượng lớn các công cụ và thư viện với cùng mục tiêu là tự động hóa trình duyệt. Khởi nguồn từ 2004 tại công ty ThoughtWorks với vai trò đơn giản là một "JavaScriptTestRunner" để kiểm tra một ứng dụng của công ty, nay, hệ sinh thái Selenium đã được xây dựng và phát triển một cách mạnh mẽ bởi các dự án Open Source xoay quanh Selenium WebDriver.

Selenium WebDriver (WebDriver) đóng vai trò cầu nối cung cấp người kiểm tra phần mềm một API để giao tiếp với các APIs của hầu hết các trình duyệt được dùng rộng rãi ngày nay (Firefox, Chrome/Chromium, Edge, Safari, ...). Mỗi trình duyệt được phát triển một phương thức hiện thực hóa WebDriver riêng biệt, gọi là driver. Selenium Framework tạo ra phương thức cho phép người dùng điều khiển các driver này qua chỉ một giao diện hoạt động, cross-browser, cross-platform, đồng thời, cho phép Selenium kiểm tra ứng dụng trên các trình duyệt như thể nó là một khách hàng.

Selenium bao gồm rất nhiều thành phần như WebDriver, IDE,... Đối với những máy khác nhau với phiên bản khác nhau cần những webdriver driver phù hợp. Ở đây, ta sử dụng Selenium trên nền Python, thông qua webdriver của Google Chrome phiên bản 101 để thực hiện crawler. Các phiên bản webdriver của Google Chrome có thể được tải xuống tại link.

2.2 BeautifulSoup

`Beautiful Soup` là một thư viện Python có thể được sử dụng để lấy được dữ liệu từ các files HTML và XML. Nó là một phiên bản gọn nhẹ hơn so với công cụ khác như Scrapy. Bằng cách sử dụng Beautiful Soup để trích xuất nguồn từ file HTML, parser của nó sẽ tạo ra một cấu trúc cây phức tạp. Từ đó, ta chỉ cần quan tâm đến các thành phần `Tag`, `NavigableString`, `BeautifulSoup`, và `Comment`. Ở đây ta chủ yếu quan tâm đến thành phần `Tag`.

Mỗi `Tag` tương ứng với một tag trong file HTML nguyên gốc, và nó có rất nhiều thuộc tính và phương thức. Mỗi `Tag` cũng có một tên có thể được truy cập qua thuộc tính `name`. Đối với các thuộc tính của `Tag`, chúng được truy suất thông qua thuộc tính `attrs`. Thuộc tính `attrs` có thể được xem như một từ điển trong Python và ta cũng có thể được truy cập các giá trị của từ điển thông qua các khoá, là các thuộc tính tương ứng của tag trong file HTML.

2.3 Urllib

`Urllib` là một thư viện dùng để xử lý URLs, bao gồm 4 modules lớn:

- `urllib.request`: mở và đọc các URLs.
- `urllib.error`: chứa các exceptions tạo ra bởi `urllib.request`.
- `urllib.parse`: dùng để parse các URLs.
- `urllib.robotparser`: dùng đề parse các files `robots.txt`.

Ta sẽ chỉ tập trung vào một số hàm được sử dụng khi thực hiện crawler.

2.3.1 Urlparse

`Urlparse` là một hàm trong module `urllib.parse` dùng để chia URL ra thành 6 phần tương ứng với cấu trúc tổng quát của một URL `scheme://netloc/path;parameters?query#fragment`. Các giả trị trả về có thể null. Ta sử dụng `urlparse` để lấy ra `path` trong URL để có thể cấu trúc lại tên file khi tải xuống.

2.3.2 Urljoin

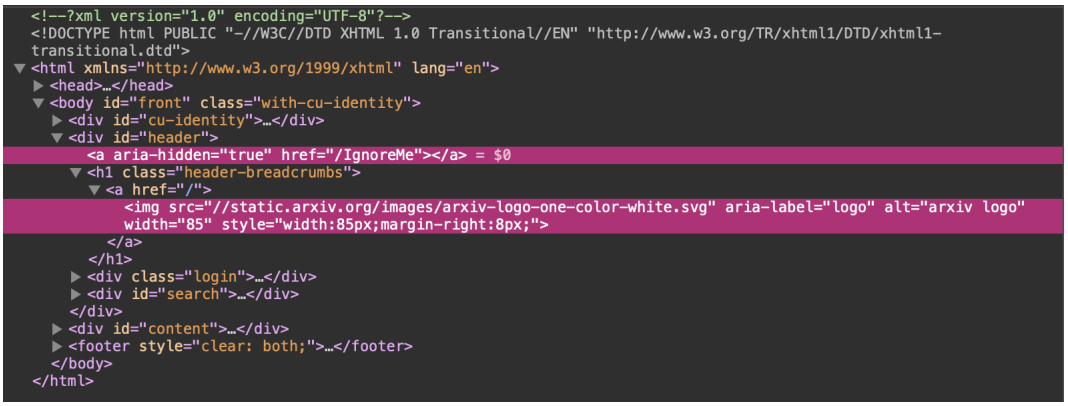
Rất nhiều files trên một trang web có thể được host trên cùng một nền tảng. Khi đó, ta có thể sử dụng URL tương đối (relative URL) để chỉ dẫn vị trí các file này trong thuộc tính `href` hoặc `src`. Các trình duyệt có thể phân giải các địa chỉ này để tìm ra file chính xác. Tuy nhiên, dưới góc độ xử lý chuỗi, ta khó có thể biết được cấu trúc nào là hợp lý. `Urljoin` là một hàm trong module `urllib.parse`. có thể tạo ra URL hoàn chỉnh từ một URL gốc (base URL) và relative URL. Ta có thể thực hiện thông qua cú pháp `urljoin(base-URL, relative-URL)`. Nếu relative URL là một URL đầy đủ, hàm này sẽ trả về kết quả của relative URL và nếu không có relative URL, base URL sẽ được trả về.

2.3.3 Urlopen

`Urlopen` là một hàm trong module `urllib.request` dùng để truyền và kiểm tra URL được nhập vào (response). Ta có thể khai thác một số thông tin của response như headers hay nội dung của trang web. Tham số được truyền vào của `urlopen` có thể là địa chỉ URL hay một đối tượng `Request`. Sử dụng đối tượng `Request`, ta có thể truyền một số thông tin khi dùng `urlopen` bên cạnh URL. Một số thông tin có thể thay đổi như data hoặc headers.

3 Cách thực hiện

Ta sẽ dùng Selenium để đi đến trang web có URL được chỉ định. Nhận thấy cái link thường được chứa trong thuộc tính `href` của thẻ `<a>` hoặc `src` của thẻ `<img>` đối với hình ảnh trên một trang web như ví dụ trong Hình 2.



Hình 2: Các link trong một trang web

Nhận thấy điều này, ta dùng thư viện `BeautifulSoup` để trích xuất html tĩnh của trang web và lọc tất cả các thẻ `<a>` cùng thuộc tính `href` và `src` của thẻ `<img>`. Sau đó dùng `urljoin` như

đã giới thiệu để có thể tạo ra các URLs hoàn chỉnh. Ta thu được một danh sách các URLs. Các URLs này bao gồm tất cả các link trên trang web, bao gồm cả link của JavaScript, một số URLs không phải dạng https truyền thống. Do đó, ta lại tiếp tục lọc ra các URLs có chứa từ khoá http hoặc https trong tên.

Sau khi đã có các URLs, ta nhận thấy chúng có thể không bao gồm phần mở rộng cụ thể. Do đó ta thực hiện request đến từng link và lấy thuộc tính `content-type` trong header. Nhiều `content-type` có thể tương ứng với cùng một định dạng file cụ thể (file extension). Do đó, dùng tham khảo tại link này, ta thiết lập được một danh sách `content-type` ứng với file extension. Dùng danh sách này, ta kiểm tra kiểu file có khớp với mong muốn hay không. Ta sẽ chỉ giữ lại các kiểu file có kiểu phù hợp với loại file ta mong muốn.

Một lỗi có thể nhận thấy khi gửi request dạng này đó chính là nhiều trang web có thiết lập ngăn chặn khi request được gửi thông qua script một cách liên tục. Do đó, ta gói URL và `user-agent` trong headers để có thể qua mặt cách ngăn chặn này. Ta sử dụng hai `user-agent` của máy:

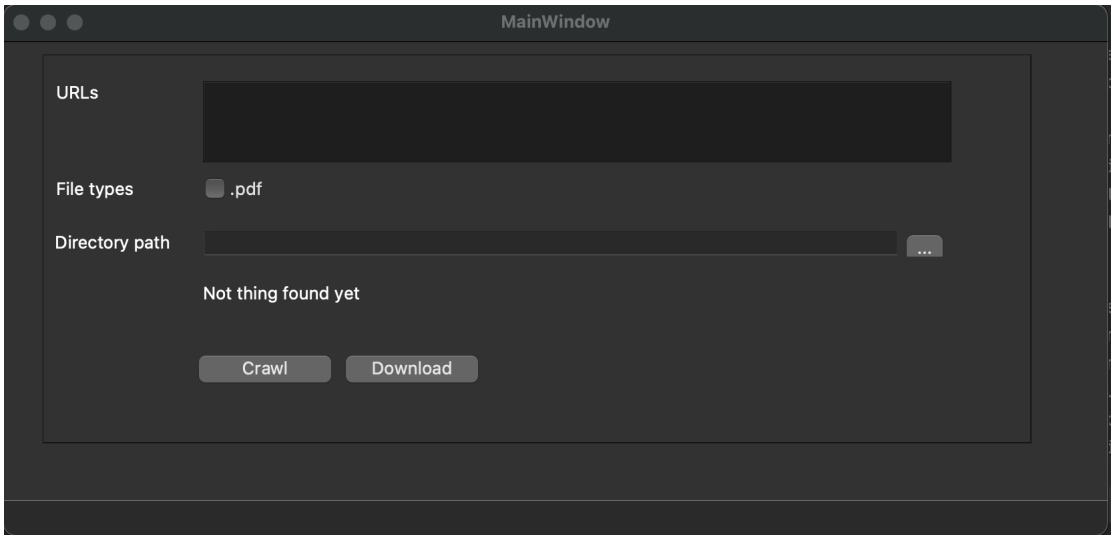
- `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/15.2 Safari/605.1.15` ứng với trình duyệt Safari.
- `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safari/537.36` ứng với trình duyệt Google Chrome.

Ta viết hàm để có thể ngẫu nhiên chọn một trong hai `user-agent` để gửi đi. Nhờ đó ta có thể qua mặt một số trang web. Tuy nhiên, khi ta sử dụng các `user-agent` này, ta có thể bị block bởi các sites, và nếu có dùng một tài khoản, nó có thể bị chặn đăng nhập tới site đó trong một khoảng thời gian. Do đó, nhiều người khuyến khích không nên đăng nhập tài khoản khi dùng crawler.

Sau khi đã có danh sách links mong muốn, ta thực hiện lấy các `path` trong URL đã phân giải để có thể tạo ra tên file phục vụ cho quá trình tải xuống. Để ý rằng, một số URL không có phần mở rộng tường minh, ví dụ như link này dẫn đến một file pdf nhưng URL không chỉ rõ phần mở rộng. Do đó, nếu URL path không có mở rộng, ta thêm phần mở rộng dựa trên danh sách đã trình bày ở phần trước. Sau khi có tên file và URL cụ thể, ta dùng `urlopen` để gửi mở file và lưu về thư mục được chỉ định.

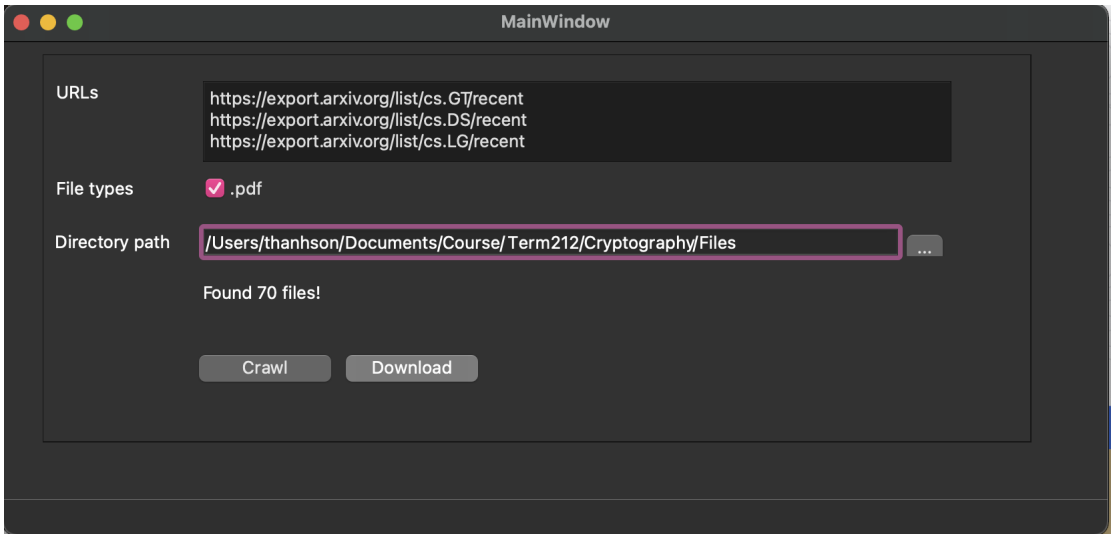
4 Demo

Với các bước đã trình bày ở phần trước, ta tiến hành hiện thực trên Python và viết được một ứng dụng web crawler đơn giản. Phần code cho bài báo cáo này có thể được tìm thấy ở GitHub. Giao diện của crawler như được trình bày ở Hình 3.



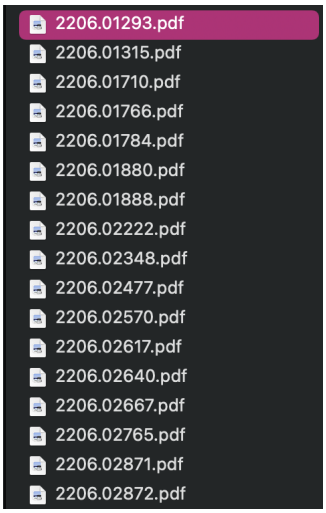
Hình 3: Giao diện của crawler

Ta sẽ tiến hành thử lấy tất cả các file pdf trên các link: link 1, link 2, và link 3. Các link này đều thuộc nền tảng arixv, do đó, giao diện và cấu trúc link của trang web tương tự như trong các Hình 1 và 2. Sau khi đã nhập xong tất cả các nội dung đường dẫn, loại file và nơi lưu file, ta bấm nút `Crawl` để bắt đầu crawl và kết quả đạt được như ở Hình 4. Có thể thấy ta thu thập được 70 links file.



Hình 4: Caption

Cuối cùng, ta thực hiện tải các file này về thông qua nút `Download` và kết quả như ở hình 5.



Hình 5: Các files tải về

5 Tổng kết

Bài báo cáo đã trình bày về cách thức thực hiện một webcrawler đơn giản. Ta đã giới thiệu về webcrawler, các thư viện có sử dụng, cách thực hiện và thực hiện một ứng dụng web-crawler cơ bản. Web crawler có thể được sử dụng để tải files một cách tự động trong một số điều kiện nhất định.

Tuy nhiên, web crawler hiện tại chỉ mới được thực hiện ở mức độ đơn giản, link file được nhúng trực tiếp vào các thành phần trên web. Crawler hiện tại đã hỗ trợ một số loại files nhưng chưa hiện thực lên UI mà chỉ mới nhận ở dạnh lệnh. Ngoài ra, vẫn chưa giải quyết được vấn đề bị block khi crawl ở một số site. Do đó, ta có thể cải thiện crawler bằng cách tự động quét các nút bấm để sinh toàn bộ link, tự động chuyển phân trang và có thể tự đăng nhập ở các trang cần đăng nhập. Ngoài ra, UI cần thêm nhiều loại file và hướng dẫn cụ thể hơn để tránh các lỗi xảy ra khi thao tác không chính xác.