

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



Đồ án I

Crawler dữ liệu bằng Python

Giảng viên hướng dẫn: **TS. Trần Ngọc Thăng**

Sinh viên thực hiện: **Lê Khánh Thành - 20195917**

HÀ NỘI-2023

Lời cảm ơn

Lời đầu tiên, chúng em xin gửi lời cảm ơn tới Thầy Trần Ngọc Thăng đã tận tình hướng dẫn, đóng góp những ý kiến bổ ích trong suốt quá trình em thực hiện đề tài nghiên cứu. Cảm ơn Thầy đã cung cấp và trang bị kiến thức, tạo điều kiện để em hoàn thiện được nghiên cứu này.

Trong quá trình thực hiện đề tài này chúng em đã tìm hiểu và vận dụng các kiến thức được Thầy trang bị và có những cố gắng nhất định, tuy nhiên do kiến thức còn hạn chế nên nghiên cứu của chúng em không tránh khỏi những thiếu sót. Em rất mong nhận được những đóng góp chỉnh sửa của Thầy để nghiên cứu của chúng em được hoàn thiện hơn.

Em xin chân thành cảm ơn!

Hà Nội, tháng 12 năm 2022

Tác giả

Lê Khánh Thành

Mục lục

Lời nói đầu	1
1 Web Crawling	2
1.1 Khái niệm về Web Crawling	2
1.2 Các phương pháp Web Crawling thường dùng	2
1.3 Thực hành crawl trang web bán hàng	3
1.4 Tổ chức lưu trữ dữ liệu trong MySQL	4
2 Thu thập file tài liệu bằng Selenium	6
2.1 Tổng quan về Selenium	6
2.2 Vì sao chọn sử dụng Selenium?	6
2.3 Lợi ích của Selenium	7
2.4 Tự động hoá việc đăng nhập và download tài liệu	8
Kết Luận	21

Lời nói đầu

Thông qua việc tiếp xúc với các trang web trên internet, bạn có thể thấy Web Crawling rất cần thiết vì nó cung cấp một cách hiệu quả và tiết kiệm thời gian để thu thập dữ liệu từ các trang web. Một số lý do khiến việc sử dụng web crawling cần thiết bao gồm:

1. Tìm kiếm thông tin: Web crawling cung cấp một cách tiết kiệm thời gian và hiệu quả để tìm kiếm các thông tin cần thiết từ các trang web.
2. Indexing các trang web cho các công cụ tìm kiếm: Web crawling cung cấp một cách tự động hóa việc indexing các trang web, giúp cho việc tìm kiếm thông tin trở nên dễ dàng hơn.
3. Tạo ra các bộ sưu tập dữ liệu: Web crawling cung cấp một cách tiết kiệm thời gian và hiệu quả để tạo ra các bộ sưu tập dữ liệu lớn.
4. Tự động hóa việc thu thập dữ liệu: Web crawling cung cấp một cách tự động hóa việc thu thập dữ liệu, giúp cho việc thu thập dữ liệu trở nên dễ dàng và nhanh chóng hơn.

Đề tài có kết cấu gồm 2 chương:

Chương 1: Web Crawling.

Chương 2: Thu thập file tài liệu bằng Selenium.

Chương 1

Web Crawling

1.1 Khái niệm về Web Crawling

Web crawling (hoặc web scrapping) là quá trình tự động thu thập dữ liệu từ các trang web. Nó sử dụng một chương trình máy tính để truy cập và tải về các trang web, sau đó phân tích và lấy nội dung cần thiết từ các trang đó. Web crawling là một phần quan trọng của việc xử lý dữ liệu trực tuyến và được sử dụng trong nhiều lĩnh vực, bao gồm tìm kiếm, phân tích dữ liệu, xử lý ngôn ngữ tự nhiên và học máy.

1.2 Các phương pháp Web Crawling thường dùng

Bạn có thể sử dụng Python để crawl dữ liệu từ web bằng cách sử dụng thư viện như BeautifulSoup, Requests hoặc Scrapy.

1. BeautifulSoup: là một thư viện được sử dụng để phân tích và lấy dữ liệu từ HTML và XML.
2. Requests: là một thư viện được sử dụng để gửi các yêu cầu HTTP và nhận dữ liệu trả về.
3. Scrapy: là một framework để crawl dữ liệu web mạnh mẽ và dễ sử dụng. Nó cung cấp một cách dễ dàng để tạo các spiders để crawl dữ liệu từ trang web và lưu trữ

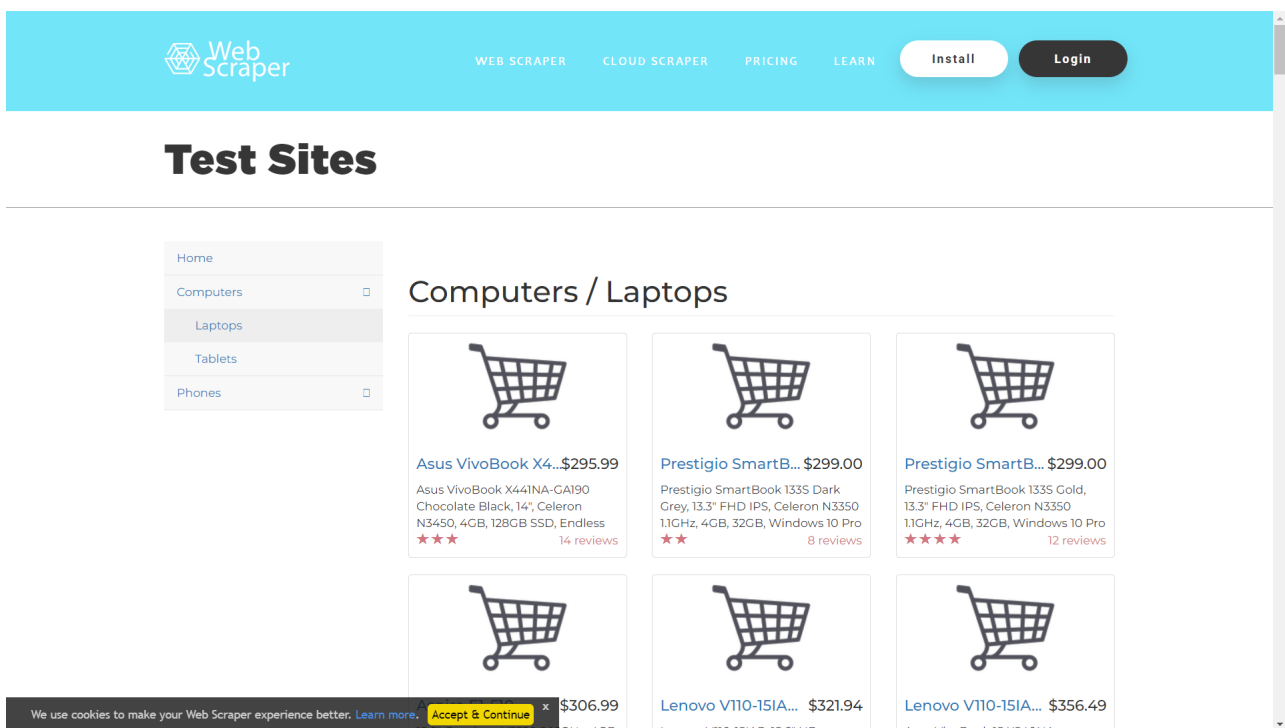
dữ liệu đó dưới dạng một định dạng nào đó.

1.3 Thực hành crawl trang web bán hàng

Link trang web cần crawl:

`https://webscraper.io/test-sites/ecommerce/allinone/computers/laptops`

Giao diện chính của trang web:



Bước đầu tiên ta cần dùng các thư viện sau:

```
import requests
from bs4 import BeautifulSoup
```

Tạo 1 biến chứa URL trang web:

```
url = 'https://webscraper.io/test-sites/e-commerce/allinone/computers/laptops'
```

Sử dụng thư viện BeautifulSoup, ta tiến hành lấy dữ liệu bao gồm: Tên sản phẩm, Giá tiền, Mô tả và Đánh giá thông qua các thẻ HTML.

```
soup = BeautifulSoup(response.content, 'html.parser')
name = [name.text for name in soup.find_all(
    'a', class_='title')] # Lay ten san pham
price = [price.text for price in soup.find_all(
    'h4', class_='pull-right price')] # Lay gia san pham
description = [desc.text for desc in soup.find_all(
    'p', class_='description')] # Lay mo ta san pham
rating = []
for rate in soup.find_all('div', class_='ratings'): # Lay danh gia san pham
    flag = 0
    for r in rate.find_all('span'):
        flag += 1
    rating.append(flag)
```

1.4 Tổ chức lưu trữ dữ liệu trong MySQL

Tạo Cơ sở dữ liệu và kết nối với MySQL bằng hàm `Store_to_mysql()`

```
def Store_to_mysql():
    connection = mysql.connector.connect(host='127.0.0.1',
                                         user='root',
                                         password='sieunhan1234')

    cursor = connection.cursor()
    cursor.execute("CREATE DATABASE data_db")
    print("Connection to MySQL Established!")
    insert_data()
```

Sau khi đã kết nối Cơ sở dữ liệu đến MySQL thành công, ta tạo hàm `insert_data()`. Hàm này sẽ tạo 1 bảng để chứa dữ liệu crawl được, bảng này gồm 4 cột: title, price, description và rating.

Kết quả thu được

Query 1 x

Limit to 1000 rows

```
1 select * from data_table
2
```

Result Grid

	title	price	description	rating
▶	Asus VivoBook X4...	\$295.99	Asus VivoBook X441NA-GA190 Chocolate Black, 14", Cel...	3
	Prestigio SmartB...	\$299.00	Prestigio SmartBook 133S Dark Grey, 13.3" FHD IPS, Cel...	2
	Prestigio SmartB...	\$299.00	Prestigio SmartBook 133S Gold, 13.3" FHD IPS, Celeron ...	4
	Aspire E1-510	\$306.99	15.6", Pentium N3520 2.16GHz, 4GB, 500GB, Linux	3
	Lenovo V110-15IA...	\$321.94	Lenovo V110-15IAP, 15.6" HD, Celeron N3350 1.1GHz, ...	3
	Lenovo V110-15IA...	\$356.49	Asus VivoBook 15 X540NA-GQ008T Chocolate Black, 15....	2
	Hewlett Packard...	\$364.46	Hewlett Packard 250 G6 Dark Ash Silver, 15.6" HD, Celer...	1
	Acer Aspire 3 A3...	\$372.70	Acer Aspire 3 A315-31 Black, 15.6" HD, Celeron N3350 1...	2
	Acer Aspire A315...	\$379.94	Acer Aspire A315-31-C33J Black 15.6", HD, Celeron N33...	2
	Acer Aspire ES1-...	\$379.95	Acer Aspire ES1-572 Black, 15.6" HD, Core i3-6006U, 4G...	4
	Acer Aspire 3 A3...	\$391.48	Acer Aspire 3 A315-31 Black, 15.6" HD, Pentium N4200 1...	4
	Acer Aspire 3 A3...	\$393.88	Acer Aspire 3 A315-21, 15.6", AMD A4-9120. 4GB. 128G...	3
	Asus VivoBook Ma...	\$399.00	Asus VivoBook Max X541NA-GQ041 Black Chocolate, 15....	1
	Asus VivoBook E5...	\$399.99	Asus VivoBook E502NA-GO022T Dark Blue, 15.6" HD, Pe...	4
	Lenovo ThinkPad...	\$404.23	Lenovo ThinkPad E31-80, 13.3" HD, Celeron 3855U 1.6G...	1

data_table 1 x

Chương 2

Thu thập file tài liệu bằng Selenium

2.1 Tổng quan về Selenium

Selenium là một công cụ tự động hóa testing cho trang web. Nó cho phép bạn ghi lại và chạy các bước tương tác với trang web như một người dùng thực sự, giúp kiểm tra tính đúng đắn và tính năng của trang web. Selenium hỗ trợ nhiều ngôn ngữ lập trình như Java, Python, C# và Ruby, có thể chạy trên nhiều nền tảng khác nhau và cung cấp một giao diện để tạo và chạy các tập lệnh kiểm tra trang web.

2.2 Vì sao chọn sử dụng Selenium?

Có nhiều lý do tại sao cần sử dụng Selenium cho việc tự động hóa testing trang web:

1. Tự động hóa: Selenium có thể giúp tự động hóa quá trình kiểm tra và giảm thiểu sự nhầm lẫn của con người.
2. Nhiều ngôn ngữ hỗ trợ: Selenium hỗ trợ nhiều ngôn ngữ lập trình như Java, Python, C# và Ruby, cho phép bạn lựa chọn ngôn ngữ phù hợp nhất với nhu cầu của dự án.
3. Chạy trên nhiều nền tảng: Selenium có thể chạy trên nhiều hệ điều hành và trình duyệt khác nhau, giúp bạn kiểm tra tính năng của trang web trên nhiều môi trường

khác nhau.

4. Tích hợp với các công cụ khác: Selenium có thể tích hợp với các công cụ khác như Jenkins, TestNG hoặc JUnit để tự động hóa quá trình kiểm tra và tăng hiệu suất.
5. Mã nguồn mở: Selenium là mã nguồn mở, cho phép bạn tùy chỉnh và mở rộng công cụ theo nhu cầu của dự án.

2.3 Lợi ích của Selenium

Có nhiều lợi ích cho con người khi sử dụng Selenium cho việc tự động hóa testing trang web:

1. Giảm thời gian kiểm tra: Selenium có thể giúp tự động hóa quá trình kiểm tra, giảm thời gian và nỗ lực cần thiết cho việc thực hiện các bước kiểm tra thủ công.
2. Giảm tỉ lệ lỗi: Selenium có thể giúp giảm tỉ lệ lỗi do con người gây ra trong quá trình kiểm tra.
3. Tăng tính nhất quán và tính bảo mật: Selenium có thể giúp tăng tính nhất quán và tính bảo mật của trang web bằng cách kiểm tra tính năng và tính đúng đắn của nó theo một cách tự động và lặp đi lặp lại.
4. Tăng hiệu suất làm việc: Selenium có thể giúp tăng hiệu suất làm việc bằng cách tự động hóa các bước kiểm tra, giúp người dùng tập trung vào các công việc khác.
5. Mở rộng kiến thức về testing: Sử dụng Selenium có thể giúp người dùng mở rộng kiến thức về testing trang web và nâng cao kỹ năng sử dụng công cụ tự động hóa.

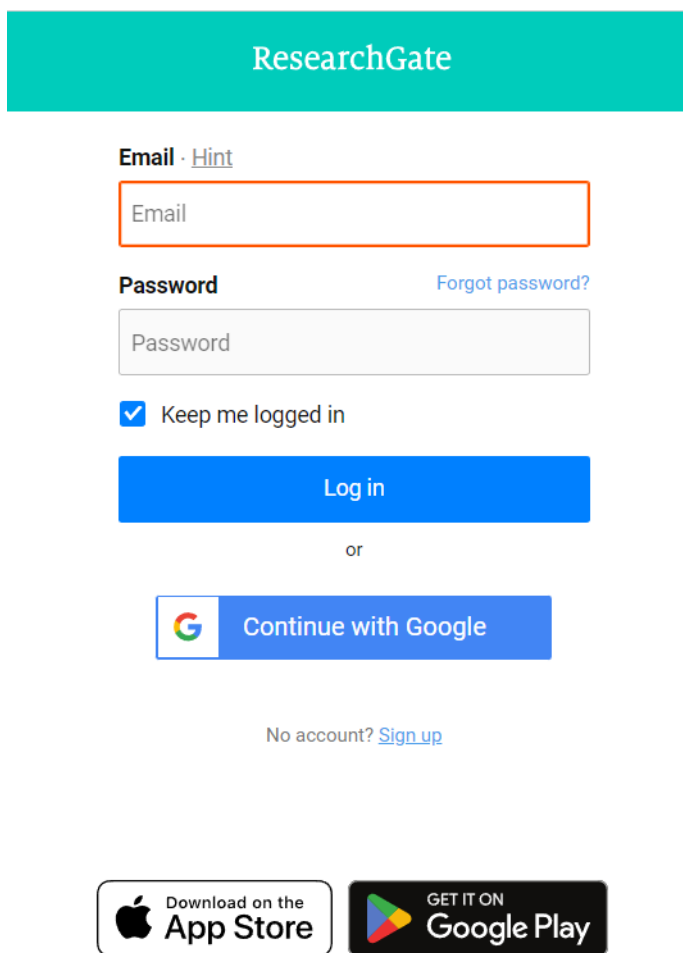
2.4 Tự động hoá việc đăng nhập và download tài liệu

Để tự động hoá việc đăng nhập và tải tài liệu, ta có thể sử dụng công cụ Selenium cùng với Chrome WebDriver.

Selenium là một framework open source cho phép bạn giả lập hành động của người dùng trên web và gửi các yêu cầu đến trình duyệt. Chrome WebDriver là một công cụ cho phép Selenium giao tiếp với trình duyệt Chrome.

Tự động hoá việc đăng nhập và tải tài liệu bằng Selenium và Chrome WebDriver giúp cho việc tải dữ liệu trở nên nhanh chóng và hiệu quả hơn, giúp tiết kiệm thời gian và nỗ lực của người dùng.

Ta có thể lấy ví dụ từ trang web: <https://www.researchgate.net/>. Sau đây là giao diện phần đăng nhập của trang web **researchgate.net**:



ResearchGate

Email · [Hint](#)

Email


Password [Forgot password?](#)

Password


☒ Keep me logged in


Log in

or

 Continue with Google

No account? [Sign up](#)

 Download on the App Store

 GET IT ON Google Play

Dưới đây là mã nguồn của class Bot dùng để tạo một bot tải về các tài liệu trên trang web Researchgate.net. Mã nguồn sử dụng các thư viện selenium và BeautifulSoup để hoạt động.

Hàm `__init__` sẽ khởi tạo một instance của Bot với tham số truyền vào là URL của trang cần tải, tham số headless (mặc định là True) để chỉ định xem cần mở trình duyệt hay không, và tham số credential (mặc định là None) để chứa thông tin đăng nhập. Sau đó, nó sẽ gọi hàm `_create_folder()` để tạo mới một thư mục chứa dữ liệu tải về và gọi hàm `_init_driver()` để khởi tạo driver:

```
def __init__(self, url, headless=True, credential=None):
    self.url = url
    self._create_folder()
    self.driver = self._init_driver(headless)
    self.credential = credential
    self.files = []
```

Hàm `_create_folder()` sẽ tạo mới một thư mục với tên là tên profile trên ResearchGate.

```
def _create_folder(self):
    folder_name = self.url.split('/profile/')[1]
    self.folder = folder_name.split('/')[0]
    if not os.path.exists(self.folder):
        os.makedirs(self.folder)
```

Hàm `_init_driver` sẽ khởi tạo trình duyệt Chrome với các tùy chọn đã chỉ định (nếu có), bao gồm mở trình duyệt ẩn (headless), tắt thông báo, tắt chặn popup, mở trình duyệt toàn màn hình và sử dụng chế độ incognito (trình duyệt ẩn danh). Hàm này sẽ cấu hình chế độ tải về với đường dẫn là thư mục vừa tạo.

```
def _init_driver(self, headless):
    options = uc.ChromeOptions()
    options.add_argument("--disable-notifications")
    options.add_argument("--disable-popup-blocking")
    options.add_argument("--start-maximized")
    options.add_argument("--incognito")
    if headless:
        options.add_argument("--headless")
    driver = uc.Chrome(options=options, version_main=chrome_version)
    params = {
        "behavior": "allow",
        "downloadPath": os.path.join(os.getcwd(), self.folder)
    }
    driver.execute_cdp_cmd("Page.setDownloadBehavior", params)
    return driver
    self.driver.get(self.url)
    a_elements = self.driver.find_elements(
        'xpath', '//*[@class="nova-legacy-o-stack__item"]/div/a')
    category_urls = [a.get_attribute('href') for a in a_elements]
    with open('category.txt', 'w') as f:
        f.writelines([url.split('/')[-2] + '\n' for url in category_urls])
    for url in category_urls:
        self.crawl_page_by_category(url)
```

Hàm `scroll_down()` sẽ cuộn trang web xuống dưới để load thêm nội dung trang.

```
def scroll_down(self):
    try:
        # Get scroll height
        last_height = self.driver.execute_script(
            "return document.body.scrollHeight")

        while True:
            # Scroll down to bottom
            self.driver.execute_script(
                "window.scrollTo(0, document.body.scrollHeight);")

            # Wait to load page
            time.sleep(random.randint(2, 3))

            # Calculate new scroll height and compare with last scroll height
            new_height = self.driver.execute_script(
                "return document.body.scrollHeight")
            if new_height == last_height:
                break
            last_height = new_height
    except:
        pass
```

`get_filename()` là một phương thức trong class **Bot** để lấy tên file pdf mới nhất trong thư mục. Các bước và chức năng của hàm như sau:

1. Sử dụng hàm **glob** để lấy tất cả các file có đuôi `'pdf'` trong thư mục tương ứng với tên thư mục lưu trữ dữ liệu là **self.folder**.
2. Duyệt qua từng file trong danh sách các file đã tìm được.
3. Nếu một file chưa tồn tại trong danh sách **self.files**, thì thêm nó vào danh sách **self.files** và trả về tên file đó.
4. Nếu toàn bộ danh sách đã được duyệt xong mà vẫn chưa tìm thấy file mới, trả về **None**.

```
def get_filename(self):  
    pdf_files = glob(f'{self.folder}/*.pdf')  
    for file in pdf_files:  
        if file not in self.files:  
            self.files.append(file)  
            return file  
    return None
```

`extract_title_and_abstract()` là một hàm để trích xuất tiêu đề và tóm tắt từ một trang web. Cụ thể, hàm này sử dụng thư viện BeautifulSoup để phân tích mã HTML của trang và lấy ra các thông tin cần thiết. Các bước thực hiện của hàm như sau:

1. Khởi tạo các biến **title** và **abstract** để lưu trữ kết quả trích xuất.
2. Sử dụng BeautifulSoup để phân tích mã HTML của trang và tạo một đối tượng `soup`.
3. Khởi tạo một danh sách `strings` để lưu trữ các chuỗi đã được xử lý trước.
4. Thực hiện việc xử lý trước bằng cách loại bỏ ký tự xuống dòng và khoảng trắng trên hai đầu của từng chuỗi trong trang HTML, rồi thêm chuỗi đó vào danh sách `strings`.
5. Tìm kiếm tiêu đề trong danh sách `strings` bằng cách duyệt từng phần tử trong danh sách và kiểm tra nếu chuỗi đó có chứa từ "title" hoặc "Abstract". Nếu tìm thấy, gán giá trị của phần tử đó cho biến `title` hoặc `abstract` tương ứng.
6. Trả về giá trị của `title` và `abstract`.

```
def extract_title_and_abstract(self):  
    title = ''  
    abstract = ''  
    soup = BeautifulSoup(self.driver.page_source, 'html.parser')  
    strings = []  
    # Preprocessing  
    for string in soup.strings:  
        string = string.replace('\n', '')  
        string = string.strip()  
        if string:  
            strings.append(string)  
    # Title
```



```
for string in strings:
    if 'PDF' in string:
        title = string
        break

for i in range(len(strings)):
    if 'Descriptions' in strings[i]:
        continue

    if 'Abstract' in strings[i] or 'Description' in strings[i]:
        abstract = strings[i+1]
        break

return title, abstract
```

Hàm `download_paper()` dùng để tải về các tài liệu từ một trang web. Các bước và chức năng của hàm `download_paper()` như sau:

1. Sử dụng Selenium để mở trang web chứa tài liệu cần tải.
2. Tìm và kích hoạt nút tải về tài liệu.
3. Chờ cho tài liệu được tải xuống và lưu vào thư mục.
4. Sử dụng BeautifulSoup để trích xuất tiêu đề và tóm tắt của tài liệu.
5. Trả về tiêu đề và tóm tắt của tài liệu.

```
def download_paper(self, paper_url):
    result = {'status': False, 'title': '', 'abstract': '', 'file': ''}
    self.driver.get(paper_url)
    titles = self.driver.find_elements(
        'xpath', '//div[contains(@class, "nova-legacy-e-text--size-xl")]')
    if len(titles):
        result['title'] = titles[0].text
    try:
        result['abstract'] = self.driver.find_element(
            'xpath', '//div[@itemprop="description"]').text
    except Exception as ex:
        result['title'], result['abstract'] = self.extract_title_and_abstract()
        text = BeautifulSoup(self.driver.page_source, 'html.parser').text
        if 'Request full-text' in text:
            result['file'] = 'Request full-text'
            print(
                f'{result["title"]} FAILURE (contact with authors to download)')
            return result

    papers = self.driver.find_elements(
        'xpath', '//a[contains(@href, ".pdf")]')
    if len(papers) == 0:
```

```
print(
    f'{paper_url.split("/")[-1]} FAILURE (contact with authors to
        download)')

return result

# Click download
ActionChains(self.driver).click(papers[0]).perform()
time.sleep(random.randint(2, 3))
filename = self.get_filename()
retry = 0
while not filename:
    if retry == 15:
        break

    filename = self.get_filename()
    print('Waiting download . . . . .')
    time.sleep(2)
    retry += 1
result['file'] = filename
result['status'] = True if filename else False
print(f'{result["title"]} {result["status"]}')
return result
```

Phương thức (method) **run(self)** của class **Bot** chứa một số bước thực hiện như sau:

1. Gọi phương thức **driver.get(self.url)** để đến URL của trang mong muốn.
2. Tìm tất cả các thẻ **<a>** trong phần tử có class **"nova-legacy-o-stack__item"** và lưu vào danh sách **a_elements**.
3. Tạo danh sách **category_urls** chứa các URL liên kết tới các danh mục từ các thẻ **<a>** trong danh sách **a_elements**.
4. Lưu danh sách **category_urls** vào file văn bản **category.txt**, mỗi URL trên một dòng.
5. Duyệt qua từng URL trong **category_urls** và gọi phương thức **crawl_page_by_category** để thu thập dữ liệu từ các trang trong danh mục.

```
def run(self):
    if credential:
        if not self.login(credential):
            print('Login Fail')
            return False
        else:
            research_url = os.path.join(self.url, 'research')
            self.driver.get(research_url)
            self.scroll_down()
    else:
        self.driver.get(self.url)
        # Get all paper url
        a_elements = self.driver.find_elements(
            'xpath',
            '//*[@class="nova-legacy-v-publication-item__stack-item"]/div/a')
        paper_urls = [a.get_attribute('href') for a in a_elements if type(
```

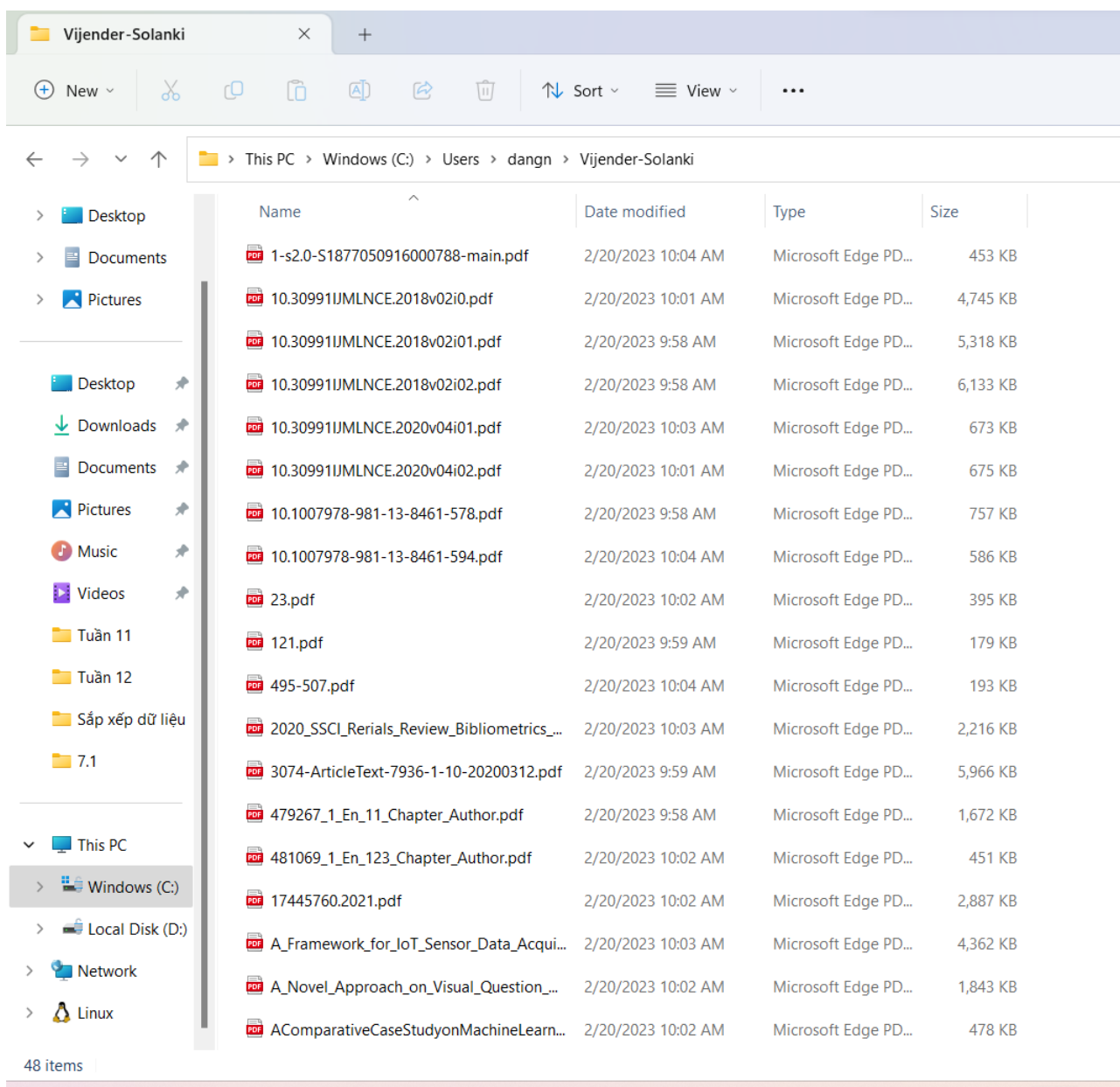
```
        a.get_attribute('href')) == str and '/publication/' in
            a.get_attribute('href')]]
data = {'title': [], 'abstract': [], 'filename': []}
print('Starting crawler . . . . .')
# Download paper
for paper_url in set(paper_urls):
    result = self.download_paper(paper_url)
    data['title'].append(result['title'])
    data['abstract'].append(result['abstract'])
    data['filename'].append(result['file'])
df = pd.DataFrame(data)
return df.to_csv(f'{self.folder}_papers.csv', index=False)
```

Các file pdf bài báo khoa học sẽ được download từ trang cá nhân của GS TS. Vijender Kumar Solanki.

```
def crawl_papers(url, credential):  
    print('Inting crawler . . . . .')  
    bot = Bot(url=url, credential=credential, headless=False)  
    bot.run()  
    bot.driver.quit()  
  
if __name__ == '__main__':  
    credential = {  
        'email': 'than*****@sis.hust.edu.vn',  
        'password': '*****'  
    }  
    crawl_papers(  
        url='https://www.researchgate.net/profile/Vijender-Solanki',  
        credential=credential)
```

Kết quả thu được:

Kết quả thu được sau khi tự động download file tài liệu sẽ là một thư mục chứa các file có định dạng PDF như hình dưới:



Kết luận

Trong thế giới của big data và thông tin, việc crawl dữ liệu trên web trở nên rất quan trọng để cung cấp nguồn dữ liệu cho nhiều mục đích, bao gồm phân tích thị trường, xây dựng các hệ thống tìm kiếm, vv.

Python và Selenium là hai công cụ mạnh mẽ và phổ biến để crawl dữ liệu trên web. Python cung cấp rất nhiều thư viện hỗ trợ việc crawl dữ liệu như BeautifulSoup, Requests, Scrapy vv. Còn Selenium là một công cụ mã nguồn mở cho phép bạn tự động hóa các hoạt động trên trình duyệt web, giúp bạn crawl dữ liệu từ các trang web đòi hỏi đăng nhập hoặc các trang web có sử dụng AJAX.

Tuy nhiên, việc crawl dữ liệu trên web cũng cần phải tuân thủ các quy định về bảo mật và quyền riêng tư, bạn cần chú ý đến việc sử dụng các gói dữ liệu một cách hợp lý và chỉ crawl dữ liệu từ các nguồn có sẵn với sự cho phép của chủ sở hữu.

Tổng kết, crawl dữ liệu web sử dụng Python và Selenium là một công cụ rất hữu ích và tiên tiến cho việc phân tích dữ liệu web, nhưng cần phải tuân thủ các quy tắc và điều lệ của các trang web để tránh vi phạm bản quyền hoặc gây ảnh hưởng đến hiệu suất của trang web. Khi sử dụng Python và Selenium, người dùng cần phải cẩn thận với việc tự động hóa quá nhiều hoạt động, vì nó có thể gây tắc nghẽn trang web và gây ra sự phản ứng của trang web. Tuy nhiên, khi sử dụng một cách cẩn thận và đúng cách, Python và Selenium có thể giúp cho việc crawl dữ liệu web trở nên dễ dàng và hiệu quả hơn.