**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

****

**BÁO CÁO KĨ THUẬT**

**ĐỀ TÀI: CRAWL WEB’S DATA**

**Giảng viên hướng dẫn: Th.S Huỳnh Thị Thanh Thương**

**Sinh viên thực hiện:**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và Tên** | **MSSV** |
| **1** | **Diệp Minh Tâm** | **16521057** |

Mục lục

1. *Lời mở đầu 2*
2. *Phân tích cấu trúc web và những thành phần thông tin hữu ích 3*
3. *Kĩ thuật crawl 5*
4. *Các thành phần đã crawl được 10*
5. *Các vấn đề hiện gặp phải 10*
6. **Lời nói đầu**

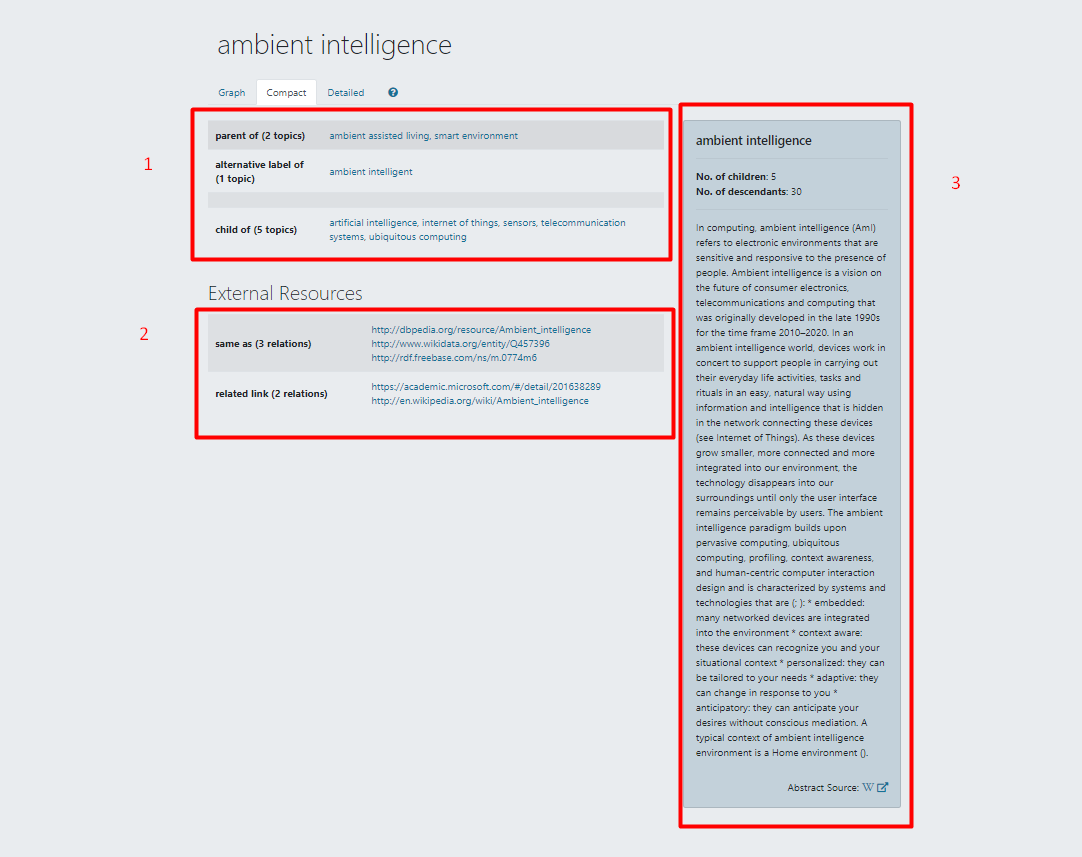
Dữ liệu ngày nay không còn đơn thuần được lưu trữ thủ công trên sách, giấy,… mà đã được mã hóa, lưu trữ trong máy tính với dung lượng rất lớn, được xuất bản rộng rãi đến mọi người qua các hình thức như website,… Vậy thay vì đọc từng website và phân tích dữ liệu, có cách nào khiến cho máy tính tự động kết nối web và lưu trữ dữ liệu mà người dùng mong muốn không ? Đó chính là tiền đề cho sự ra đời của Crawler (web bot), để thực hiện các yêu cầu kể trên hoặc.

Nhu cầu to lớn trên cũng khiến các ngôn ngữ lập trình liên tục cập nhật và cải thiện các tính năng, thư viện hỗ trợ các crawler như Scrapy, Beautiful Soup, Phantom Js, Selenium,… Việc chọn ngôn ngữ và thư viện hỗ trợ cũng tùy vào trang cần crawl, kiểu sắp xếp dữ liệu.

1. **PHÂN TÍCH CẤU TRÚC WEB VÀ NHỮNG THÀNH PHẦN THÔNG TIN HỮU ÍCH**
2. **Web cần crawl**

<https://cso.kmi.open.ac.uk/topics/computer_science> và các children, decendents của web trên.

1. **Dữ liệu cần crawl**
2. **Cấu trúc trang web:**



Cấu trúc trang web gồm các thành phần mang thông tin như sau:

1 – **Relationship**: bảng đầu tiên, thể hiện các mối quan hệ các topic con (child) của label đang được nhắc dến, các label có nghĩa tương đương, label hiện tại là con của các label nào

2 – **External Resources**: các định nghĩa theo các nguồn dữ liệu khác

3 – **Content:** Chứa nội dung được trích dẫn từ Wikipedia, link Wikipedia, thông tin về số children, số decendants.

4 – **Cấu trúc đường link:** [**https://cso.kmi.open.ac.uk/topics/computer\_science**](https://cso.kmi.open.ac.uk/topics/computer_science)

**https://cso.kmi.open.ac.uk/topics/ +** tên topic có dấu gạch dưới thay cho khoảng trắng

1. **Thành phần cần crawl:**

Tất cả các thành phần trên, bao gồm cả các đường link, label tương ứng với các đường link, content,…

1. **Tổ chức và lưu dữ liệu:**

Kiểu tổ chức và lưu dữ liệu đơn giản nhất là qua file text (.txt), kế đó là lưu bằng database, hiện tại module đang thực hiện việc lưu trữ dữ liệu qua (.txt), sẽ nghiên cứu và cải tiến phát triển đến database sau.

1. **Kĩ thuật crawl**

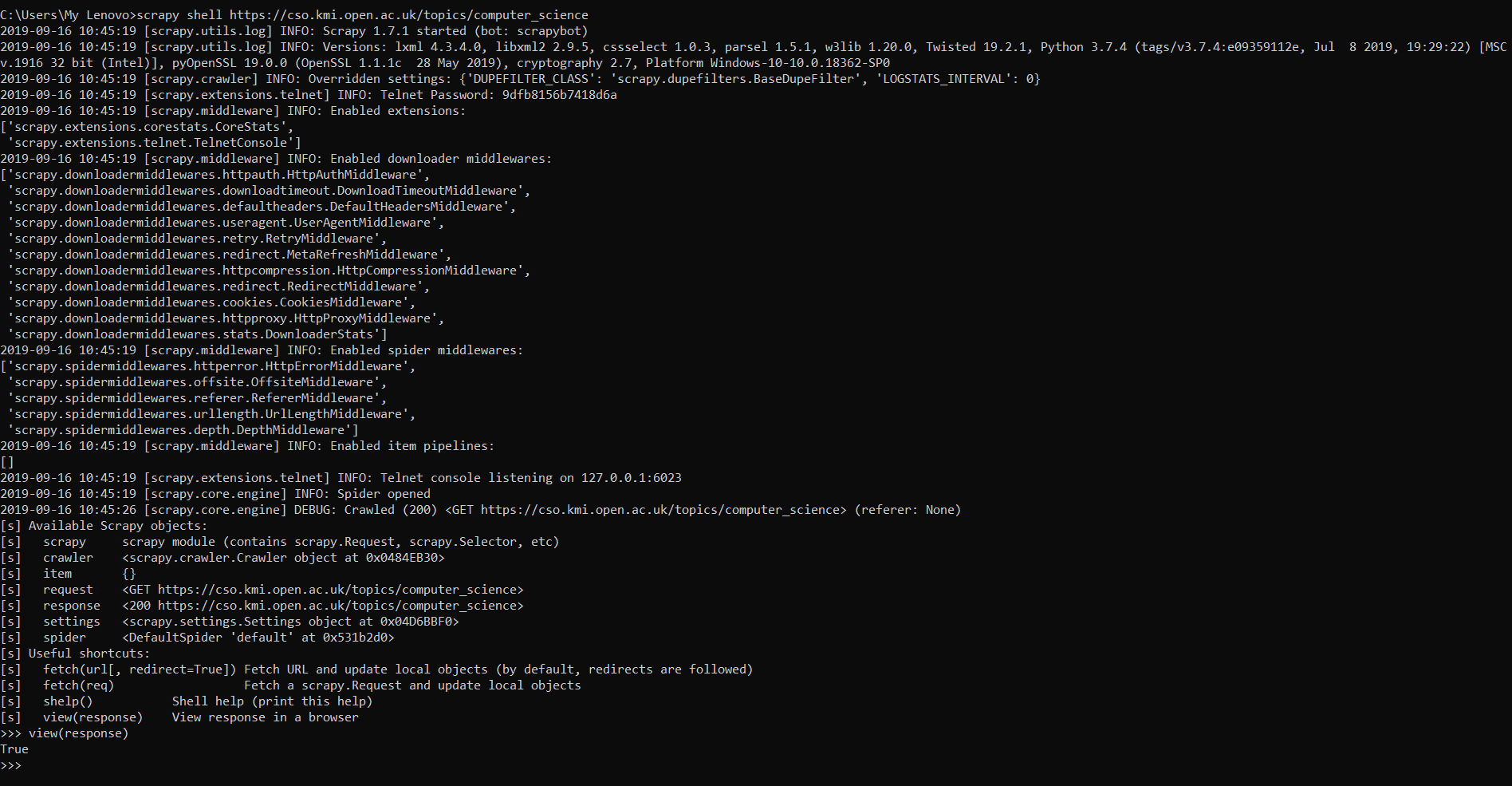
Vì số lượng dữ liệu cần crawl rất lớn (12382 pages), yêu cầu đặt ra là cần dùng 1 thư viện hỗ trợ crawl nhẹ, nhanh. Từ yêu cầu trên dẫn đến việc chọn thư viện scrapy hoạt động trên python làm công cụ chính. Với ưu điểm là tốc độ nhanh do không truy cập web thông qua driver.

1. **Xác định kết cấu trang web**

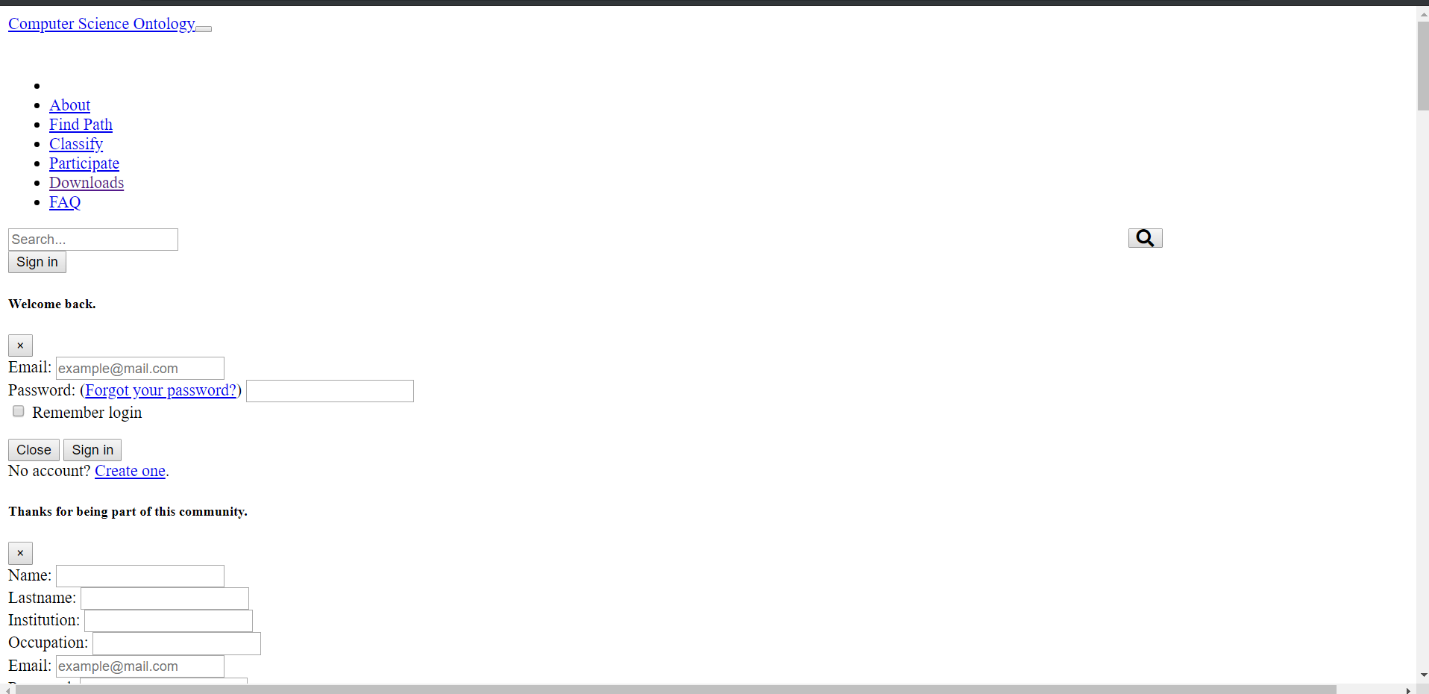
* Vì các web hiện nay đa phần đều là các dynamic web (các thành phần của web sẽ được các file Js hiển thị lên) nên việc xác định được crawler có thể “nhìn thấy” những thành phần nào là rất quan trọng trước khi crawl để tránh hiện tượng không thể crawl những nội dung như mong muốn mặt dù web hiển thị trên trình duyệt như thế.
* Áp dụng Scrapy Shell để kiểm tra các thành phần mà Crawler nhìn được trước khi thực hiện crawl.

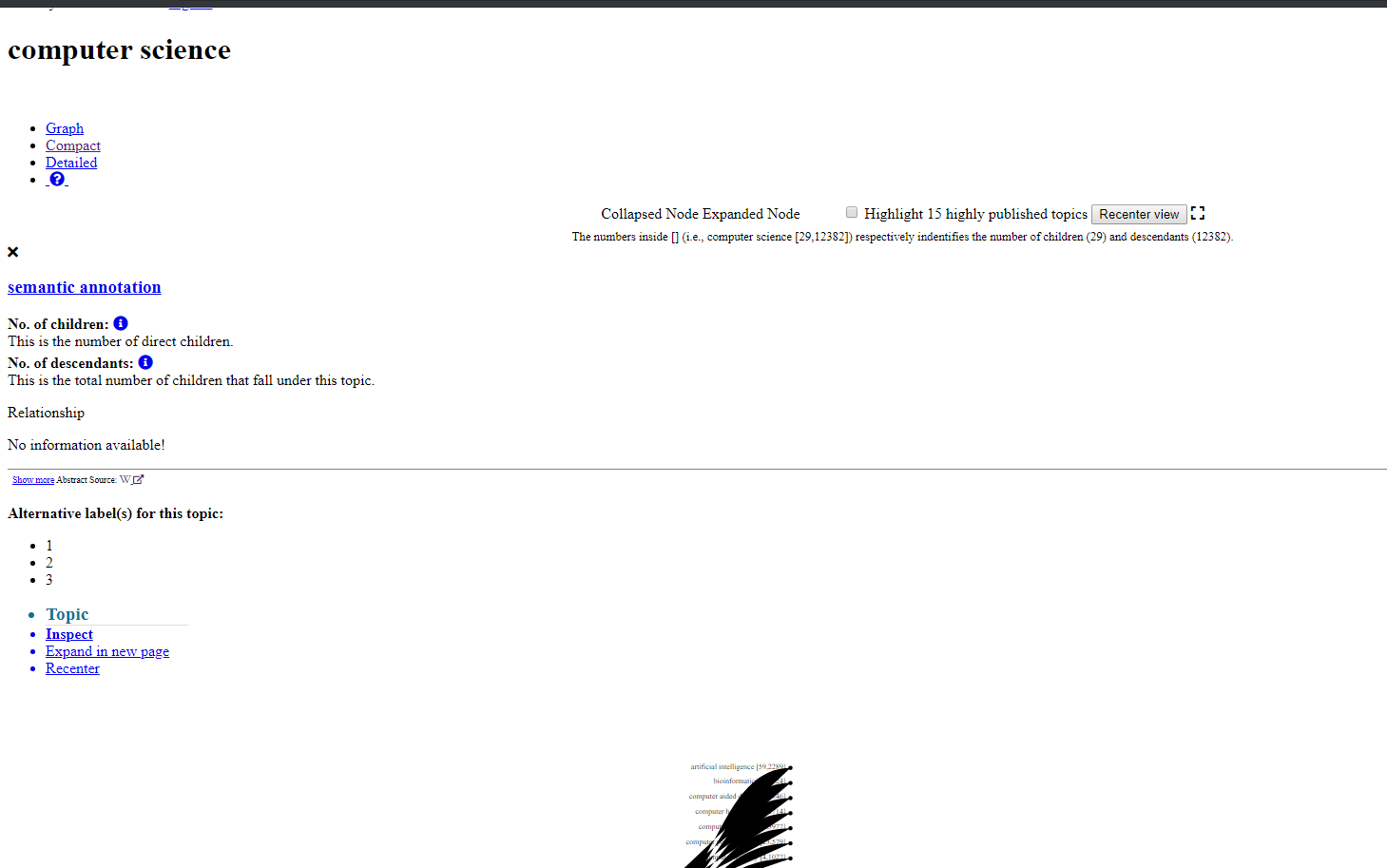
+ Mở cmd và nhập lệnh sau : scrapy shell + url

+ Nhập tiếp: view(response)



Trình duyệt sẽ tự mở và hiển thị những gì mà Crawler có thể “nhìn được” từ trang web:









1. **Sử dụng các class, id, attribute nhìn được để crawl**

Từ trang response mà Crawler “nhìn được”, ta thiết lập các trường crawl thông tin như sau:

set\_selector=response.xpath("//table[1]")

        j=0

        for title in set\_selector.css('.compact-topic'):

            child = title.css('a ::text').get()

            if j==0:

                child\_list=child

            else:

                child\_list=child\_list+", "+child

            if child\_txt.count(link\_process(child))==0:  #if not exist, add to  queue

                child\_txt.append(link\_process(child))

                child\_queue.append(link\_process(child)+'#compact')

            j=j+1

        #crawl value

        title\_page = process\_NoneType(response.css(".display-5::text").get())

no\_decendent = process\_NoneType(response.css(".card-body>p::text").getall())

content = process\_NoneType(response.css(".card-text::text").get())

        #childof\_ &\_label

        """

        link\_father = process\_NoneType(set\_selector.css("tr:last-child").getall())

        print(link\_father)

        """

        link = response.url

        #External Resources

        link\_resource=" "

link\_wiki = " "

        set\_selector=response.xpath("//table[2]")

link\_external=process\_NoneType(set\_selector.css("span>a::attr(href)").getall()) #response.xpath("//a[contain(@id, wikipedia)]/")

try:

            link\_wiki = process\_NoneType(response.xpath('//a[contains(@id, "wikipedia")]/@href')[2].get())   #link wiki

        except:

            pass

        if link\_external != " ":

            for i in range(0,len(link\_external)-1):

                link\_resource = link\_resource +", " + link\_external[i]

Các Children sẽ chọn từ trong table đầu tiên, với các tag <span> có class = compact-topic, từ đó lọc ra các attribute “href” của tag <a> sẽ có tên của các Children, dùng các tên này để thay thế và tạo đường link mới như mục **II)-2)-a)-4**

Các title page, Number\_Decentdent, Content, và link wiki, link external Resource đều lấy được chính xác lần lượt nhờ các dòng lệnh sau:

*title\_page = process\_NoneType(response.css(".display-5::text").get())*

*no\_decendent = process\_NoneType(response.css(".card-body>p::text").getall())*

*content = process\_NoneType(response.css(".card-text::text").get())*

*link\_wiki = " "*

*try:*

*link\_wiki = process\_NoneType(response.xpath('//a[contains(@id, "wikipedia")]/@href')[2].get())   #link wiki*

*except:*

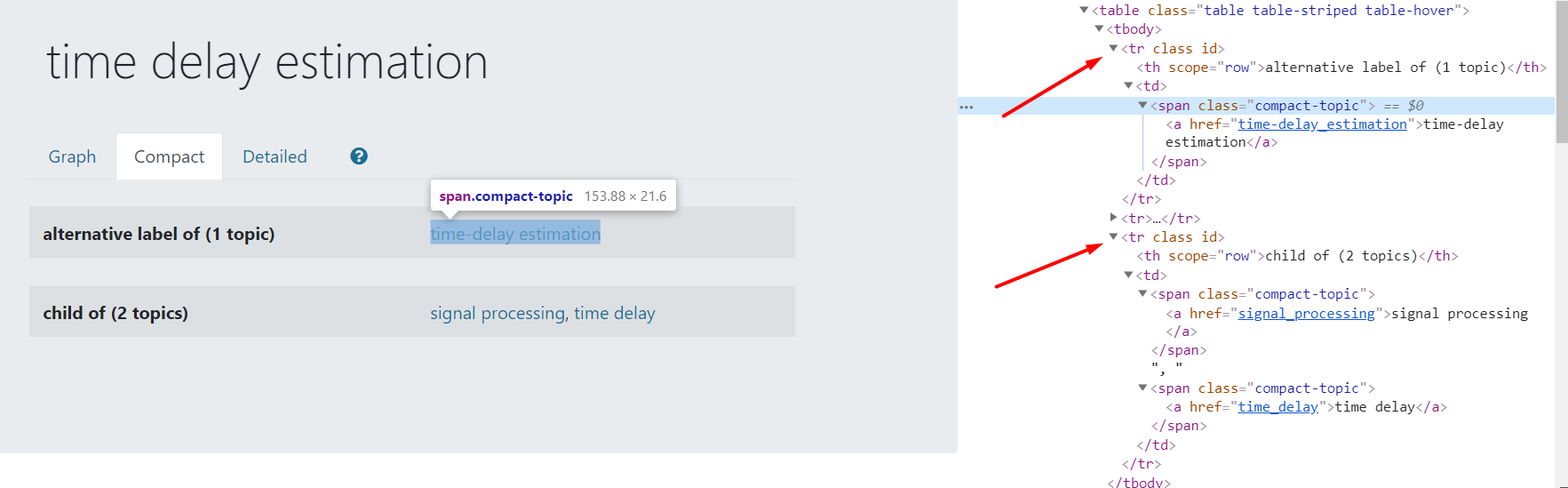
*pass*

*link\_external=process\_NoneType(set\_selector.css("span>a::attr(href)").getall())*

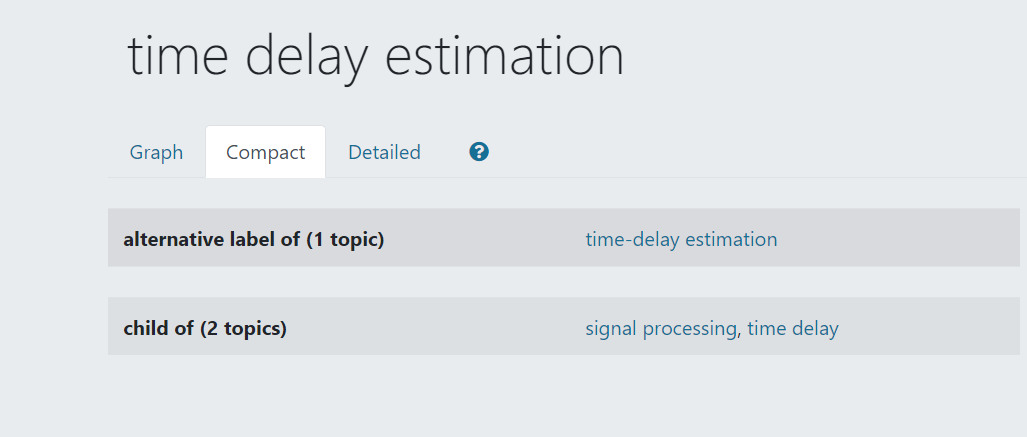
1. **Các thành phần crawl được**

Các thành phần crawl được chính xác đã kể ở trên bao gồm *title page, Number\_Decentdent, Content, và link wiki, link external resource, link response*

1. **Các vấn đề đang gặp phải**

Các thành phần child – parentTopic – alternative label of đều có chung class là compact-topic, không thể nhận biết được dựa vào class, đồng thời cũng không phân biệt được dựa vào quan hệ cha con của các tag.

Các Class cha không hề được định nghĩa

Sự bất đồng về cấu trúc của 1 số trang cần crawl: 1 số child là lá (sẽ bị mất đi thành phần child), 1 số lại khuyết thành phần parent hoặc alternative label, khiến cho việc phân biệt trở nên rất khó khăn

