ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO

| Bonus Assignment 1 |

Môn học: Lập trình cho Khoa học dữ liệu

Sinh viên thực hiện: Nguyễn Đức Minh Quân

Mã số sinh viên: 20120357

TP. Hồ Chí Minh, ngày 10 tháng 12 năm 2022

I. Giả thuyết bài toàn:

1. Sinh viên sẽ sử dụng các công cụ như request, selenium, beautiful soup, và viết một chương trình bằng python để thực hiện kiểm tra trên google xem liệu một bài báo khoa học đã có source code trên Github hay chưa. Chương trình sẽ nhận vào tên của một bài báo khoa học, và chương trình sẽ trả ra danh sách đường dẫn đến các Github tương ứng.

2. Nếu như chương trình tìm được các trang Github tương ứng với bài báo, chương trình sẽ trả về các Github đó cho người dùng. Nếu như chương trình không thể tìm được repository tương ứng, chương trình sẽ thông báo cho người dùng.

Theo yêu cầu bài toán, em hiểu và định nghĩa như sau:

- Viết chương trình bằng ngôn ngữ **python** để giải quyết.
- Ở chương trình này, em đi tìm kiếm đường dẫn Github liên quan tới bài báo, chứ không phải đi tìm bài báo chính xác với dữ liệu nhập vào.
- Dữ liệu nhập vào sẽ là **tên đầy đủ** của bài báo.
- Kết quả sẽ là danh sách đường dẫn Github tương ứng với **tên bài báo**. Tại sao phải là **tên đầy đủ** của bài báo, vì nếu không đầy đủ, kết quả trả về sẽ là những bài báo liên quan tới keyword của dữ liệu nhập vào.
- Kết quả trả về cũng **không phải chính xác 100%** rằng đường dẫn Github là của bài báo.
- Quá trình tìm kiếm sẽ dựa trên kết quả của Google, do đó sẽ dùng **tên bài báo** để thu thập dữ liệu trên Google.

II. Thư viện sử dụng:

Trong chương trình này, em sẽ dùng thư viện **BeautifulSoup** để thu thập dữ liệu tĩnh mà không cần tác động tới trang web. Cụ thể là sẽ thu thập tất cả đường dẫn từ kết quả trả về của Google sau khi nhập tên bài báo.

Trước khi truy cập vào đường dẫn, chúng ta cần phải dùng thư viện **Requests** để gửi yêu cầu lấy dữ liệu từ đường dẫn đó.

Và trong quá trình thu thập dữ liệu, có rất nhiều bài báo trả về kết quả của trang https://arxiv.org, vì thế em cũng tiến hành thu thập riêng trang này nếu nằm trong kết quả trả về. Trang này yêu cầu tương tác nên em phải dùng thư viện **Selenium.**

Webdriver em dùng trong **Selenium** là Edge(). **Microsoft Edge.**

III. Thu thập dữ liệu:

1. Dữ liệu nhập vào:

Khởi chạy main.py và tiến hành nhập tên bài báo.

Trong quá trình xây dựng, em có thử qua tên các bài báo nước ngoài và cả Việt Nam. Ở đây em sẽ báo cáo về danh sách các bài báo sau:

i. ***A Textual-Visual-Entailment-based Unsupervised Algorithm for Cheapfake Detection.*** trong Grand Challenge: Detecting CheapFakes, thuộc khuôn khổ hội nghị A* ACM MM 2022 (Lisbon, Bồ Đào Nha) của nhóm nghiên cứu đến từ fit@hcmus đã chiến thắng và được chấp nhận.
ii. ***EVA: Exploring the Limits of Masked Visual Representation Learning at Scale.*** Của Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, Yue Cao

iii. ***ACE: Cooperative Multi-agent Q-learning with Bidirectional Action-Dependency.*** Của Chuming Li, Jie Liu, Yinmin Zhang, Yuhong Wei, Yazhe Niu, Yaodong Yang, Yu Liu, Wanli Ouyang

2. Thu thập dữ liệu từ Google:

Vì là Báo khoa học nên em tiến hành thu thập tất cả đường dẫn trả về của Google và cả Google Scholar.

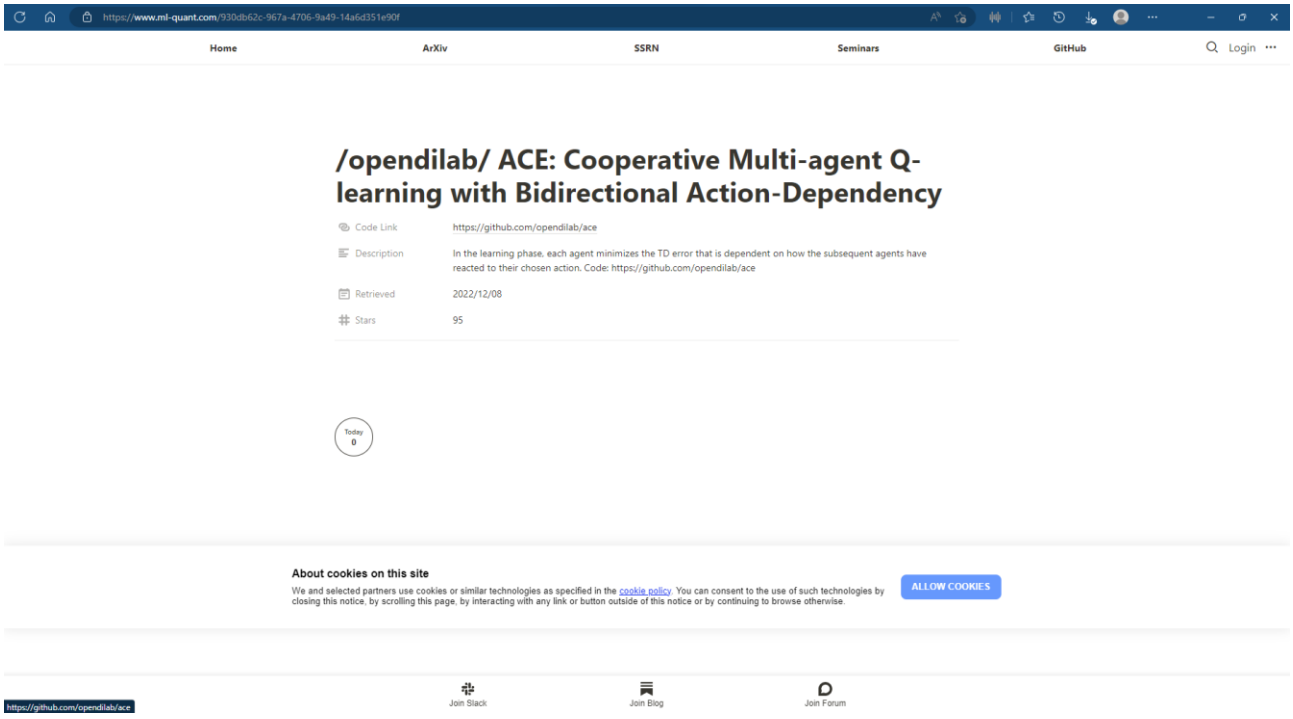
Sau khi có tất cả đường dẫn, em kiểm tra xem có đường dẫn nào của Github hay không, sau đó lưu vào 1 mảng để kiểm tra sau.

Quá trình thu thập được xây dựng trong **Get_Google.py**

3. Thu thập dữ liệu từ kết quả trả về từ Google:

Với một bài báo khoa học, có thể có tới vài trăm kết quả, nếu ta vào từng trang một để tìm kiếm thì rất mất thời gian.

Và việc vào tất cả đường dẫn là không cần thiết, vì theo thuật toán của Google thì tất cả đường dẫn liên quan thì sẽ năm ở những vị trí đầu của kết quả. Vì thế em chỉ tiến hành thu thập từ **top 10** kết quả tìm kiếm được ở cả Google và Google Scholar. Vậy nhiều nhất ta sẽ truy cập vào tối đa 20 đường dẫn để tìm kiếm tiếp.



Với một trang web như trên, đường dẫn của Github có thể xuất hiện ở dạng Text hoặc là Link. Vì thế ta tiến hành cào hết tất cả các text có chứa ‘https://github.com’ về. Đồng thời thu thập hết các đường dẫn từ name tag <a[@href]> có chứa ‘https://github.com’

Trong quá trình tìm kiếm, em có khảo sát được 2 trang uy tín và có sẵn đường dẫn xuất hiện nhiều trong Google chính là *arxiv.org* và *paperswithcode.com.*

Vì vậy em có viết thư viện riêng để cào 2 trang này là **Get_arxiv.py** và **Get_pwc.py**

IV. Xử lý dữ liệu:

Trong quá trình thu thập dữ liệu, chắc chắn sẽ có những đường dẫn Github không liên quan tới bài báo, hoặc là đường dẫn Github sai.

Vì thế ta cần xử lý danh sách đường dẫn Github thu thập được.

Ở đây, nếu kết quả Github được tìm từ 2 trang arxiv và paperswithcode thì ta sẽ không cần xử lý.

1. Xử lý đường dẫn:

Đường dẫn Github có dạng: https://github.com/[Tên người dùng]/[Tên repository]/…. Với … dẫn tới các dữ liệu có trong repository.

Vì vậy đơn giản nhất là ta có kiểm tra xem đường dẫn nào đáp ứng được dạng trên, nếu không thì xoá.

Điểm yếu: Đôi khi đường dẫn chính xác dẫn tới dữ liệu trong repository nhưng ta lại xoá nó vì ta chỉ lấy các đường dẫn tới [Tên repository].

2. Xử lý nội dung mô tả của người dùng.

Với tên bài báo nhập vào, ta bắt đầu đối chiếu với nội dung có trong đường dẫn Github. Thông thường người dũng sẽ mô tả qua tệp **Readme.**

Vì vậy ta chỉ tiến hành đối chiếu tên bài báo với tệp readme của người dùng để xử lý.

- **Bước 1:** Tên bài báo sẽ được tách ra thành các từ. Danh sách sẽ bao gồm: Danh sách 1 từ, danh sách 2 từ liền nhau, …, danh sách **n** từ liền nhau. Với **n** là số từ trong tên bài báo

  Ví dụ: Bài báo “Bài báo khoa học” sẽ được tách thành:

  + Danh sách 1 từ gồm: “Bài”, “báo”, “khoa”, “học”.

  + Danh sách 2 từ liền nhau gồm: “Bài báo”, “báo khoa”, “khoa học”.

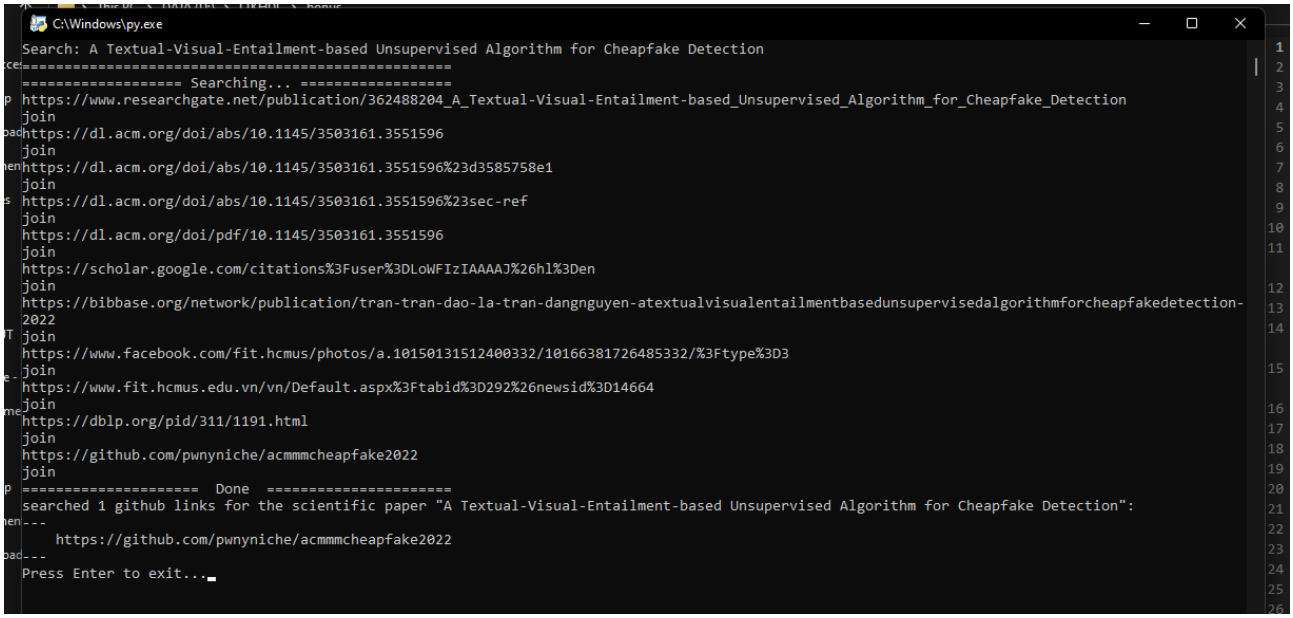
  …

  + Danh sách **n** từ liền nhau gồm: “Bài báo khoa học”.

- **Bước 2:** Ta kiểm tra xem các từ khoá đã tách như trên có xuất hiện trong mô tả hay không. Với từng danh sách, ta kiểm tra mật độ số từ xuất hiện và trả về kết quả phần trăm (%). Ví dụ: Nếu cả cụm “Bài báo khoa học” xuất hiện thì trả về 100%, nếu chỉ có “khoa học” xuất hiện thì danh sách 1 từ sẽ là 50% và 2 từ sẽ là 33,33%. Ta lấy kết quả lớn nhất.
- **Bước 3:** Dựa trên kết quả của mật độ trên ta kết luận được là đường dẫn Github đó có liên quan tới tên bài báo được tìm kiếm hay không. Đúng thì ta giữ lại.

V. Kiểm thử:

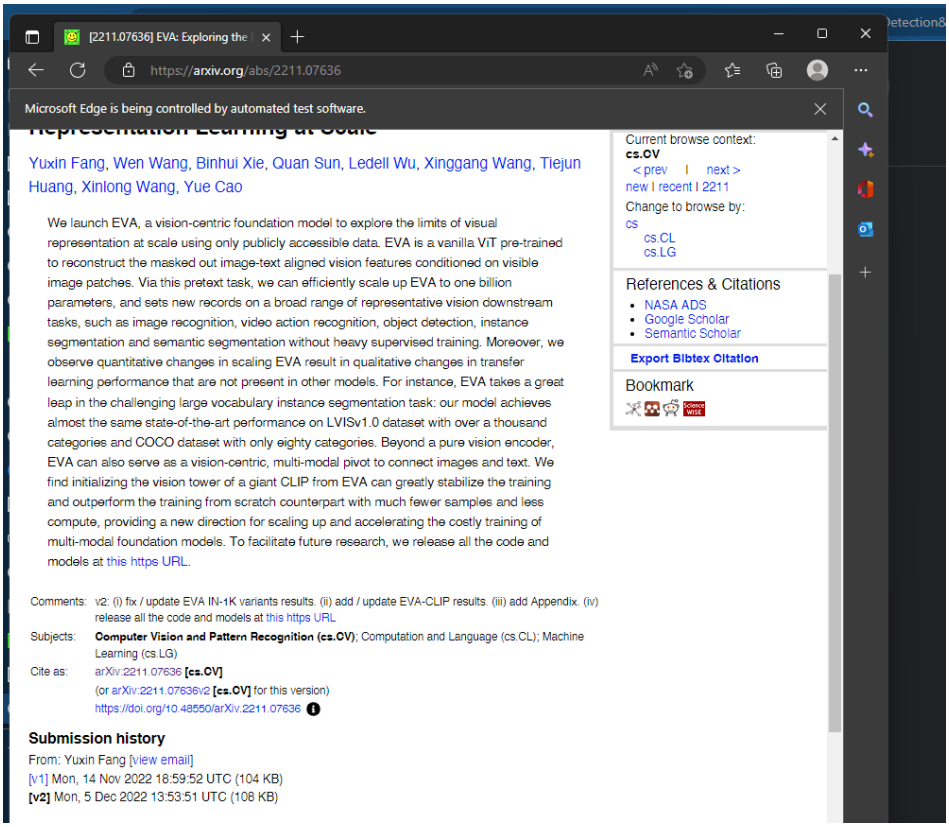
1. *A Textual-Visual-Entailment-based Unsupervised Algorithm for Cheapfake Detection.*



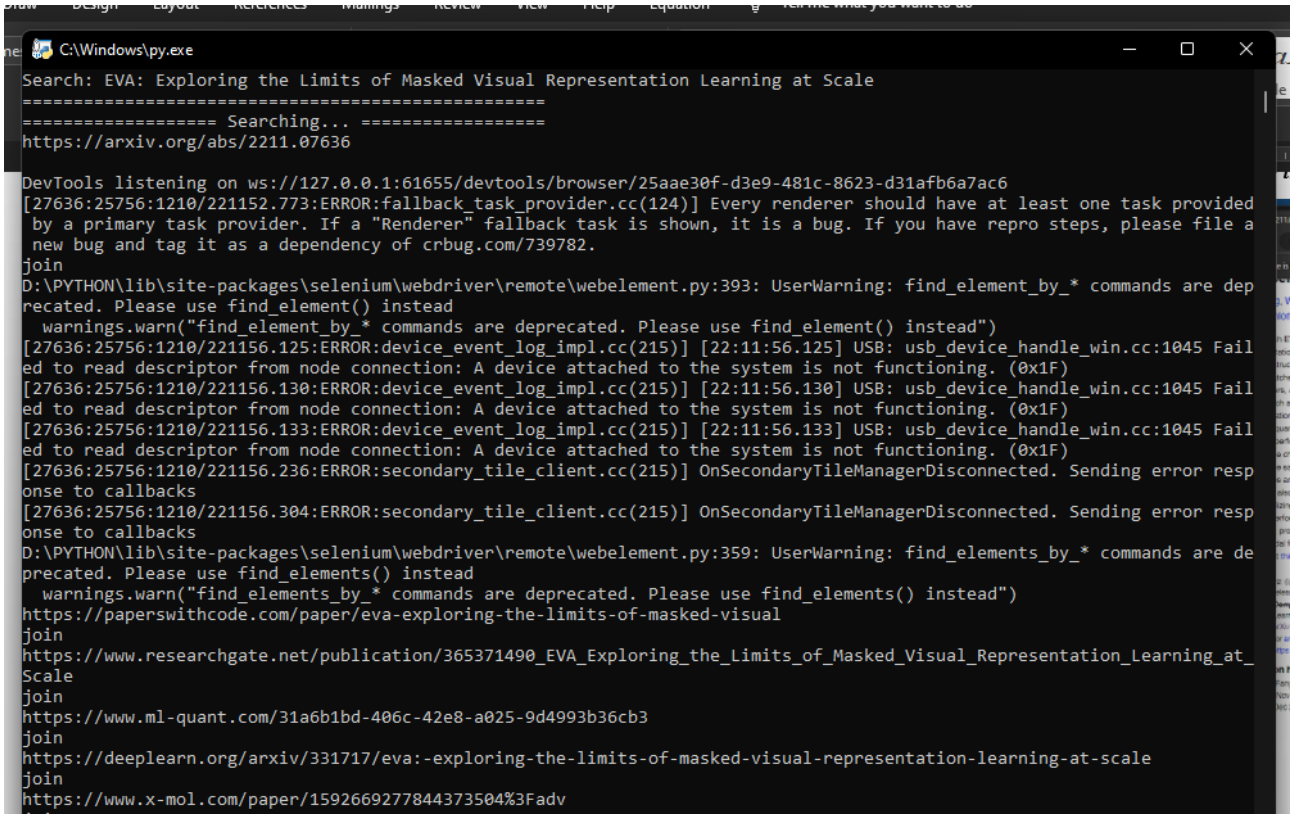
Trong quá trình tìm kiếm sẽ hiện tên các trang web là kết quả của Google mà chương trình tiến hành vào để tìm kiếm.

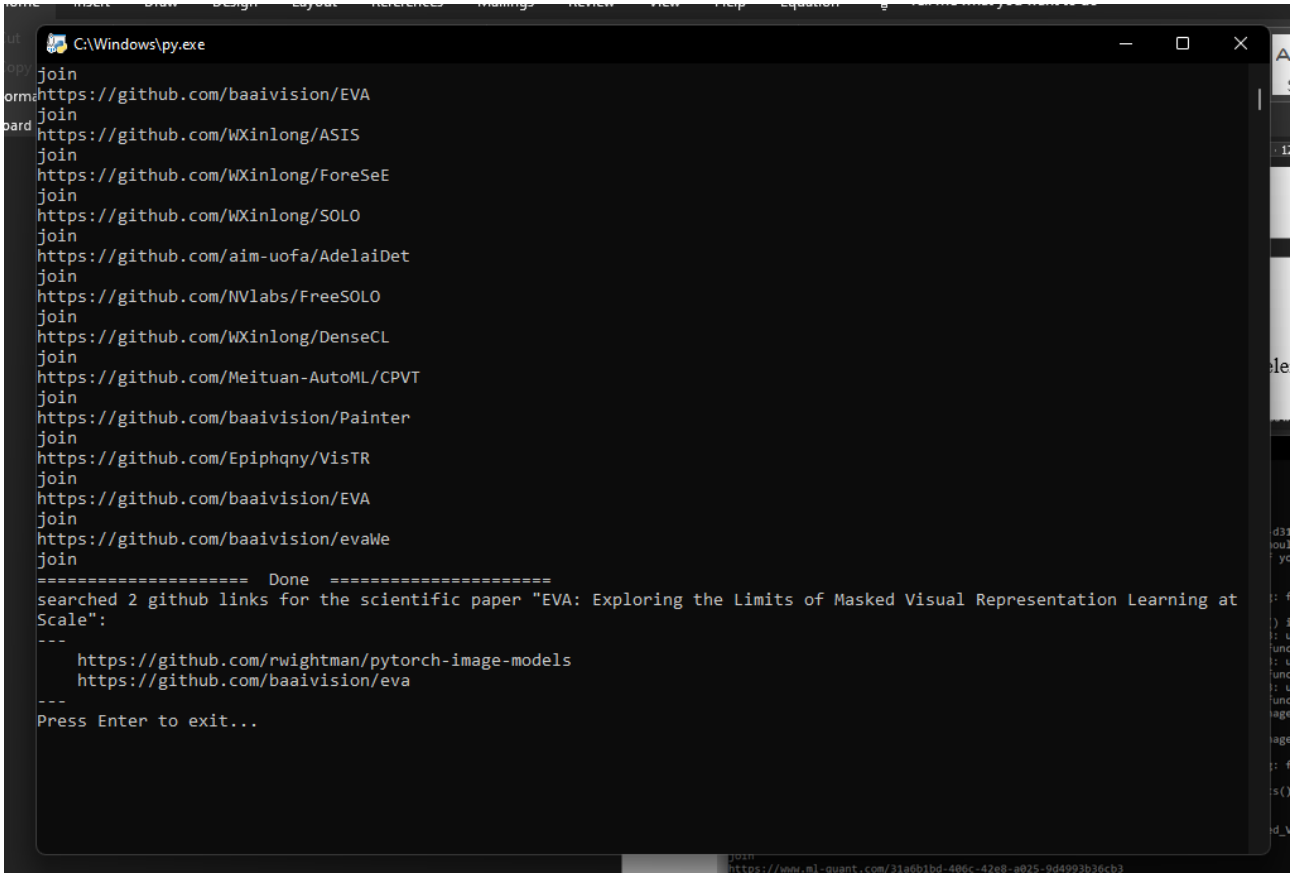
Nếu vào được thì hiện *"join"*, còn nếu mất nhiều thời gian thì sẽ báo lỗi Timeout Error và nếu lỗi kết nối thì báo lỗi Connection Error.

2. *EVA: Exploring the Limits of Masked Visual Representation Learning at Scale*



Trong quá trình tìm kiếm có xuất hiện trang arxiv.org, vì thế Selenium sẽ get driver của Edge để truy cập vào trang web cào dữ liệu và sẽ tắt sau đó.





Các trang github xuất hiện sau có nghĩa là chương trình đang truy cập vào để xử lý kiểm tra xem đường dẫn có liên quan tới bài báo hay không.

3. *ACE: Cooperative Multi-agent Q-learning with Bidirectional Action-Dependency.*