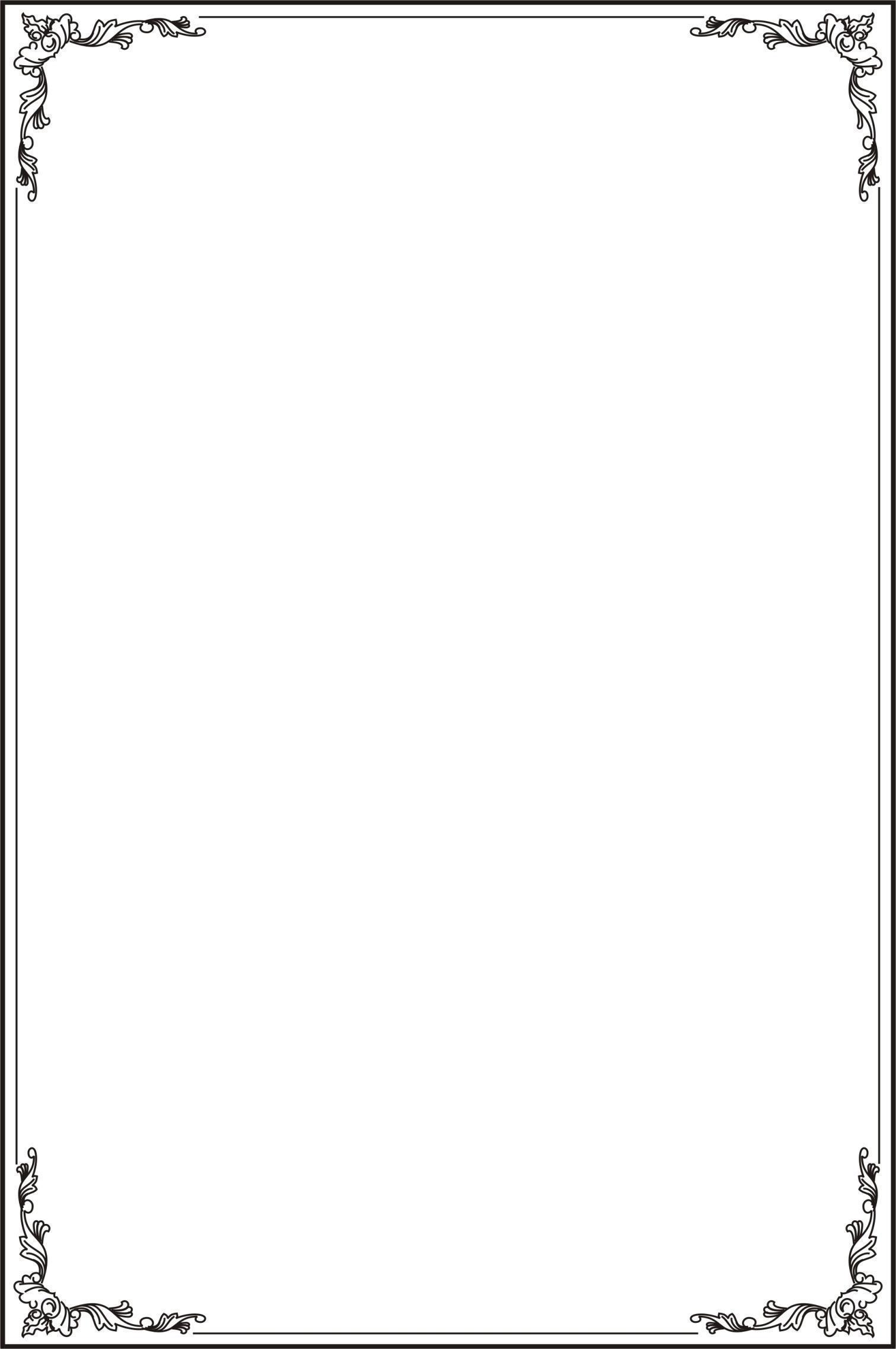
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN 2**

-----🖎🕮✍-----

****

**BÁO CÁO CUỐI KỲ**

***Đề tài:* XÂY DỰNG MÔ HÌNH DEEP LEARNING VÀ EXTENSION CHROME PHÁT HIỆN TRANG WEB LỪA ĐẢO DỰA TRÊN URL**

|  |  |  |  |
| --- | --- | --- | --- |
| **- Môn học:** | **IoT và Ứng dụng(INT14149)** | | |
| **- Giảng viên hướng dẫn :** | **ThS. Đàm Minh Lịnh** | | |
| **- Nhóm:** | **6** | | |
| **- Thành viên:** | **N20DCCN062** | **-** | **Đặng Thành Tân** |
|  | **N20DCCN073** | **-** | **Phạm Đức Thắng** |

**Thành phố Hồ Chí Minh, ngày 5 tháng 1 năm 2024.**

**Mục lục**

[**Chương I. Tổng quan.** 1](#_Toc155300746)

[**Chương II. Cơ sở lý thuyết.** 2](#_Toc155300747)

[**1. Thành phần của URL.** 2](#_Toc155300748)

[**2. Thiết kế mô hình.** 3](#_Toc155300749)

[**2.1. Character embedding.** 3](#_Toc155300750)

[**2.2. Cấu trúc mô hình.** 5](#_Toc155300751)

[**3. Trích xuất đặc trưng thủ công.** 6](#_Toc155300752)

[**4. Flask.** 8](#_Toc155300753)

[**Chương III. Xây dựng mô hình và Extension.** 9](#_Toc155300754)

[**1. Tập dữ liệu.** 9](#_Toc155300755)

[**2. Mô hình sử dụng Character embedding.** 10](#_Toc155300756)

[**3. Mô hình sử dụng các đặc trưng trích xuất thủ công.** 13](#_Toc155300757)

[**4. Đánh giá.** 16](#_Toc155300758)

[**5. Triển khai Extension.** 17](#_Toc155300759)

[**5.1. Tạo Extension.** 17](#_Toc155300760)

[**5.2. Tạo API.** 19](#_Toc155300761)

[**5.3. Gọi API và trả kết quả về cho người dùng.** 21](#_Toc155300762)

[**5.4 Kiểm thử.** 23](#_Toc155300763)

[**Chương IV. Kết luận.** 24](#_Toc155300764)

[**Tài liệu tham khảo.** 25](#_Toc155300765)

# **Chương I. Tổng quan.**

Các cuộc tấn công lừa đảo là hành vi phạm tội mà kẻ lừa đảo tìm cách lừa người dùng tiết lộ thông tin cá nhân quan trọng như chi tiết thẻ tín dụng và mật khẩu. Mục đích của những kẻ lừa đảo thực hiện cuộc tấn công lừa đảo là bán thông tin của nạn nhân, lấy tiền chuộc, khai thác lỗ hổng trong hệ thống, để đạt được nhận lợi nhuận tài chính. Một trong những hành vi phạm tội phổ biến này là thiết kế các trang web lừa đảo bắt chước các trang web lành tính (ví dụ: PayPal, eBay, v.v.) và lưu trữ chúng trong một miền bị tấn công.

Do sự bùng nổ nhanh chóng của thương mại điện tử, tội phạm lừa đảo đang gia tăng hàng năm. Kẻ lừa đảo sử dụng các công cụ như phần mềm độc hại, trang web giả mạo, và email để thực hiện các cuộc tấn công, tận dụng sự khó nhận biết giữa các trang web an toàn và trang web độc hại. Điều này đặt ra một thách thức lớn cho người dùng trong việc bảo vệ thông tin cá nhân của họ khỏi những mối đe dọa trực tuyến ngày càng phức tạp.

Lừa đảo trực tuyến gây ra thiệt hại lớn cho người dùng và các ngành công nghiệp liên quan như thanh toán, tài chính, email, v.v. Các kỹ thuật phòng chống lừa đảo truyền thống thường dựa trên danh sách đen, danh sách trắng, các đặc trưng dựa trên nội dung trang web hoặc các dịch vụ bên thứ ba. Tuy nhiên, các kỹ thuật này có nhược điểm là không thể phát hiện được các trang web lừa đảo mới, tốn nhiều thời gian và tài nguyên tính toán.

Qua đó, nhóm thực hiện xây dựng một mô hình phát hiện lừa đảo trực tuyến dựa trên mạng nơ-ron tích chập cấp ký tự (CNN và LSTM) từ URL của trang web. Ưu điểm của mô hình này là :

- Mô hình chỉ sử dụng các đặc trưng từ URL mà không cần truy cập nội dung của trang web, giúp tiết kiệm thời gian và tài nguyên tính toán.

- Mô hình không phụ thuộc vào ngôn ngữ của nội dung trang web, có thể hoạt động hiệu quả cho các trang web có nội dung bằng bất kỳ ngôn ngữ nào.

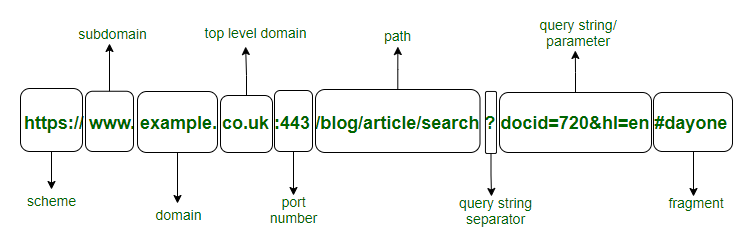
- Mô hình có thể phát hiện được các trang web lừa đảo mới chưa được phân loại trước đó, chống lại các cuộc tấn công bằng cách sử dụng các đặc trưng nhúng ở cấp ký tự.

Sau đó, nhóm tạo Extension cho ứng dụng Chrome để ứng dụng mô hình vào thực tế. Extension này hiển thị cảnh báo cho người dùng về URL mà họ muốn truy cập, nó sẽ cảnh báo URL đó có khả năng là độc hại hay không, từ đó người dùng có thể đưa ra quyết định xem có tiếp tục truy cập vào URL đó hay quay lại.

Qua thực nghiệm, nhóm đã xây dựng được mô hình phân loại trang web tốt và lừa đảo dựa trên URL đạt độ chính xác 99,76%, và tích hợp mô hình vào Extension để cảnh báo người dùng về các trang web lừa đảo.

# **Chương II. Cơ sở lý thuyết.**

## **1. Thành phần của URL.**



Hình 2.1 Thành phần URL.

Một URL bao gồm ít nhất 3 phần: scheme, domain name và top-level domain. Ngoài ra, một số URL phức tạp có thể có đến 8 hoặc 9 phần [1], bao gồm:

- Scheme: Phần giao thức hoặc scheme của URL xác định các quy tắc sẽ quyết định việc truyền và trao đổi dữ liệu. Ví dụ: HTTPS cho biết trình duyệt sẽ hiển thị trang web dưới dạng định dạng Hyper Text (HTML) cũng như mã hóa bất kỳ thông tin nào mà người dùng nhập vào trang web đó.

Subdomain: Subdomain được sử dụng để phân tách các phần khác nhau của trang web, vì nó chỉ định loại tài nguyên cần được gửi đến khách hàng. Ví dụ: subdomain “www” được sử dụng là một ký hiệu chung cho bất kỳ tài nguyên nào trên web.

- Domain: Tên miền chỉ định tổ chức hoặc thực thể mà URL thuộc về. Ví dụ: trong www.facebook.com, tên miền “facebook” cho biết tổ chức sở hữu trang web đó.

- Top-level Domain: TLD (top-level domain) chỉ định loại tổ chức mà trang web được đăng ký. Ví dụ: “.com” trong www.facebook.com cho biết một thực thể thương mại.

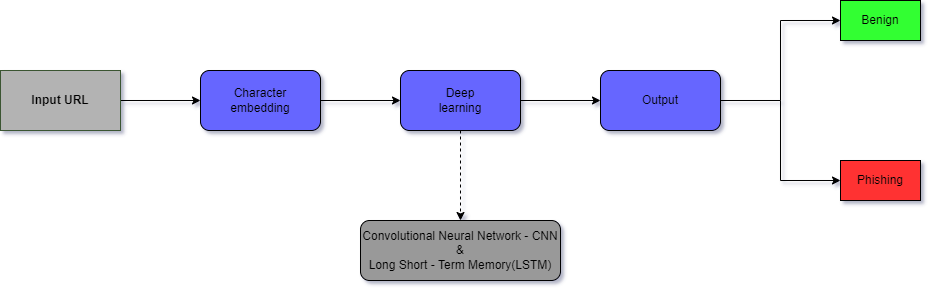
- Port Number: Số cổng chỉ định loại dịch vụ được yêu cầu bởi khách hàng, vì máy chủ thường cung cấp nhiều dịch vụ. Một số số cổng mặc định bao gồm 80 cho máy chủ HTTP và 443 cho máy chủ HTTPS.

- Path: Đường dẫn chỉ định vị trí chính xác của trang web, tệp hoặc bất kỳ tài nguyên nào mà người dùng muốn truy cập. Ví dụ: đường dẫn “/blog/article/search” chỉ định một bài viết cụ thể trong trang web blog.

- Query String Separator: Dấu chấm hỏi (?) đứng trước chuỗi truy vấn, chứa các thông số cụ thể của tìm kiếm. Dấu chấm hỏi cho trình duyệt biết rằng một truy vấn cụ thể đang được thực hiện.

- Query String: Chuỗi truy vấn chỉ định các thông số của dữ liệu đang được truy vấn từ cơ sở dữ liệu của trang web. Mỗi chuỗi truy vấn được tạo thành từ một tham số và một giá trị được nối bằng dấu bằng (=).

## **2. Thiết kế mô hình.**



Hình 2.2 Mô hình đề xuất [2].

### **2.1. Character embedding.**

Kẻ tấn công có thể làm giả các URL của các trang web chính thức bằng cách thay đổi một số ký tự không đáng chú ý. Ví dụ, họ có thể thay đổi youtube.com thành y0utube.com, thay “o” bằng “0”. Nhúng ở cấp độ ký tự giúp tìm ra thông tin bị làm giả này và cải thiện hiệu suất phát hiện URL độc hại.

Ở giai đoạn này, việc chuyển đổi được thực hiện bằng cách xác định một từ điển các chữ cái cho ngôn ngữ đầu vào. Sau đó, chuỗi ký tự được chuyển đổi thành một chuỗi vector có độ dài cố định L. Bất kỳ ký tự nào vượt quá độ dài L đều bị cắt bỏ, và bất kỳ ký tự nào không nằm trong bảng chữ cái bao gồm cả ký tự trống đều được nhúng dưới dạng vector toàn số không.

Các bước thực hiện:

- Tạo từ điển cho các chữ cái: những chữ cái được sử dụng bao gồm 95 ký tự, trong đó: 26 chữ cái tiếng Anh viết thường, 26 chữ cái tiếng Anh viết hoa, 10 số và 33 ký tự khác (ví dụ: ,;.!?: ’ /\_@#$%ˆ&\*∼`±=<>()[]{}).

- Chuyển các ký tự của chuỗi URL thành số.

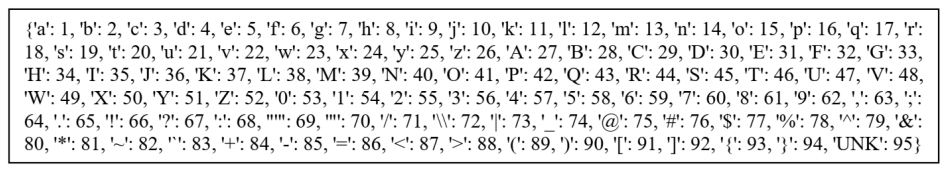
- Padding: URL có độ dài khác nhau và mạng nơ-ron chỉ có thể xử lý vector có độ dài cố định, do đó tất cả các URL phải có cùng độ dài để mô hình có thể xử lý nhiều dữ liệu cùng lúc . Ở đây, độ dài tối đa của URL được xác định là 200. Nếu độ dài của URL nhỏ hơn 200, phần còn lại sẽ được điền là 0. Nếu độ dài của URL lớn hơn 200, phần dài hơn sẽ bị cắt bỏ. Do đó, tất cả các URL sẽ giữ cùng một độ dài.

Bảng 2.1 Từ điển ký tự.

|  |  |
| --- | --- |
| Ký tự | Số chỉ mục |
| abcdefghijklmnopqrstuvwxyz | 1-26 |
| ABCDEFGHIJKLMNOPQRSTUVWXYZ | 27-52 |
| 0123456789 | 53-62 |
| ,;.!?:"’/\\|\_@#$%^&\*~`+-=<>()[]{} | 63-94 |
| UNK | 95 |
| Default | 0 |

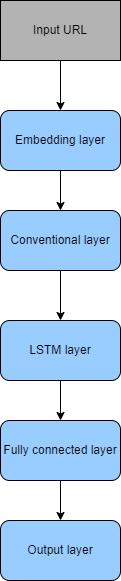


Hình 2.3 Ví dụ về biểu diễn chỉ mục URL.



Hình 2.4 Ví dụ về biểu diễn chỉ số từ vựng của ký tự.

### **2.2. Cấu trúc mô hình.**



Hình 2.5 Cấu trúc mô hình.

Embedding layer: lớp thường được sử dụng đầu tiên trong cấu trúc mô hình xử lý ngôn ngữ tự nhiên (Natural Language Processing), lớp được sử dụng để ánh xạ từng ký tự trong URL thành một vectơ dạng số.

Conventional layer: Các bộ lọc tích chập (số lượng bộ lọc, kích thước bộ lọc hoặc kernel, và kích thước pooling) được áp dụng cho mỗi lớp tích chập để trích xuất các đặc trưng quan trọng nhất và loại bỏ các đặc trưng học không cần thiết.

LSTM layer: Lớp LSTM được sử dụng để xử lý dữ liệu chuỗi và giữ lại thông tin dài hạn qua thời gian.

Fully connected layer: sử dụng để phân tích các đặc trưng từ LSTM layer mỗi lớp FC theo sau là một hàm kích hoạt ReLU, và các mô-đun dropout giữa các Fully connected layer để ngăn chặn overfitting.

Output layer: Số lượng đơn vị đầu ra cho lớp FC cuối cùng được đặt thành 1. Hàm Sigmoid được sử dụng trong lớp này để trả về kết quả nằm trong khoảng (0, 1), giá trị xác suất này xác định khả năng URL thuộc về một trang web phishing.

## **3. Trích xuất đặc trưng thủ công.**

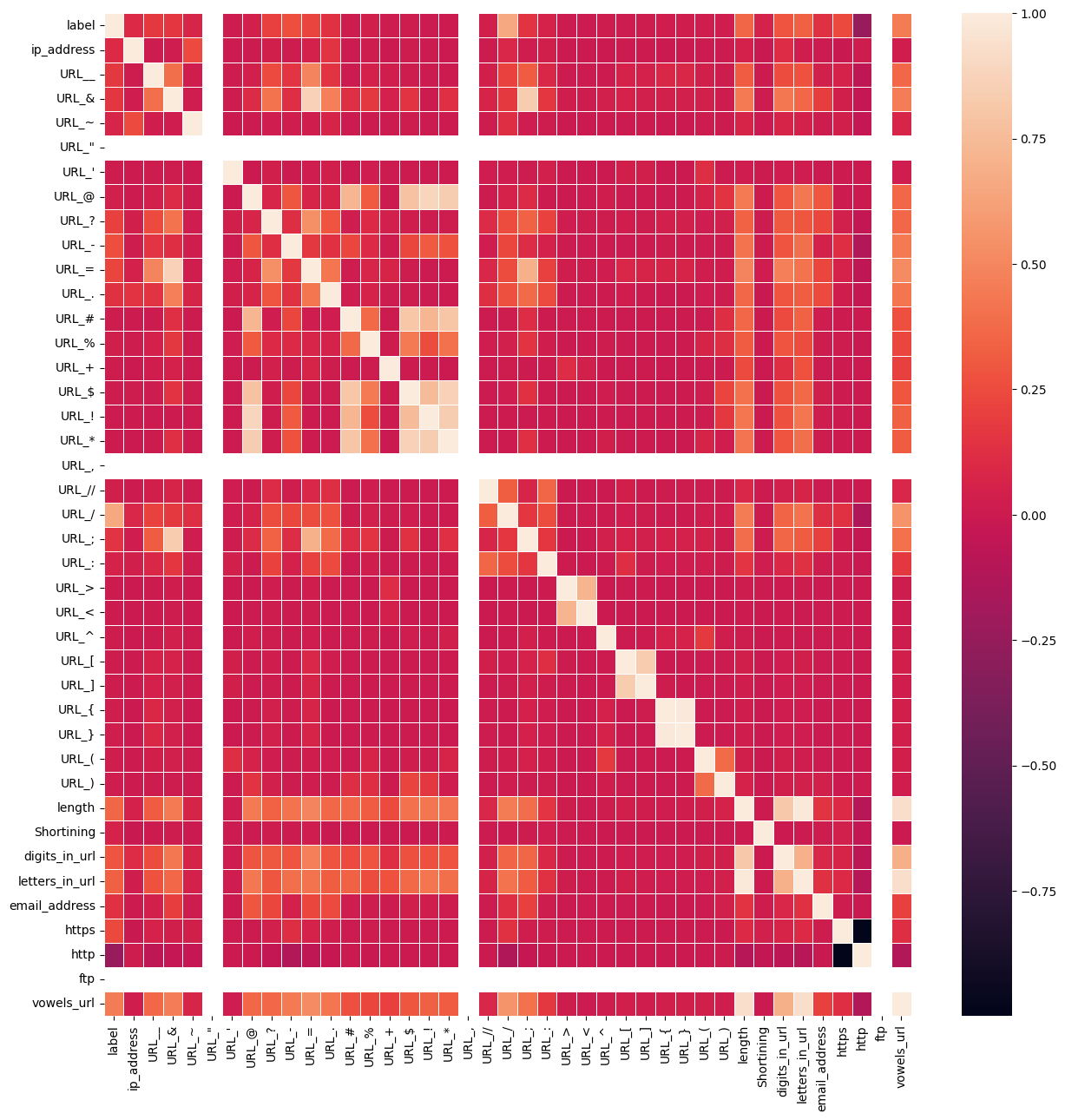
Các đặc trưng được trích xuất thủ công từ Python khi đầu vào là một URL đầy đủ. Bàng 2.2 thể hiện danh sách các đặc trưng được trích xuất thủ công từ URL [3], gồm :

- Các đặc trưng lấy số lượng các ký tự cụ thể trong URL. Các ký hiệu không thường xuyên trong các trang web benign nhưng thường xuyên trong các trang web phishing gồm nhiều ký tự đặc biệt (ví dụ: -, ., " ", /, ?, =, @, &, !, ", “,”, +, \*, #, $, %) .

- Các đặc trưng khác như kiểm tra xem trong URL có hay không có giao thức, địa chỉ email, địa chỉ IP,….

Bảng 2.2 Danh sách các đặc trưng được trích xuất thủ công.

|  |  |  |  |
| --- | --- | --- | --- |
| U1 | Có địa chỉ IP | U23 | Số lượng dấu ‘:’ |
| U2 | Độ dài của URL | U24 | Số lượng dấu ‘>’ |
| U3 | Số lượng dấu ‘**\_**’ | U25 | Số lượng dấu ‘<’ |
| U4 | Số lượng dấu ‘&’ | U26 | Số lượng dấu ‘^’ |
| U5 | Số lượng dấu ‘~’ | U27 | Số lượng dấu ‘[’ |
| U6 | Số lượng dấu ‘”’ | U28 | Số lượng dấu ‘]’ |
| U7 | Số lượng dấu ‘’’ | U29 | Số lượng dấu ‘{’ |
| U8 | Số lượng dấu ‘@’ | U30 | Số lượng dấu ‘}’ |
| U9 | Số lượng dấu ‘?’ | U31 | Số lượng dấu ‘(’ |
| U10 | Số lượng dấu ‘-’ | U32 | Số lượng dấu ‘)’ |
| U11 | Số lượng dấu ‘=’ | U33 | Có rút ngắn |
| U12 | Số lượng dấu ‘.’ | U34 | Số lượng chữ số |
| U13 | Số lượng dấu ‘#’ | U35 | Số lượng chữ cái |
| U14 | Số lượng dấu ‘%’ | U36 | Có địa chỉ email |
| U15 | Số lượng dấu ‘+’ | U37 | Số lượng dấu ‘+’ |
| U16 | Số lượng dấu ‘$’ | U38 | Có giao thức ‘https’ |
| U17 | Số lượng dấu ‘!’ | U39 | Có giao thức ‘http’ |
| U18 | Số lượng dấu ‘\*’ | U40 | Có giao thức ‘ftp’ |
| U19 | Số lượng dấu ‘,’ | U41 | Số lượng kí tự nguyên âm |
| U20 | Số lượng dấu ‘//’ |  |  |
| U21 | Số lượng dấu ‘/’ |  |  |
| U22 | Số lượng dấu ‘;’ |  |  |



Hình 2.6 Ma trận tương quan giữa các đặc trưng trích xuất.

## **4. Flask.**

Python Flask là một micro-framework được viết bằng ngôn ngữ lập trình Python dùng cho các nhà phát triển web. Nó được phát triển bởi Armin Ronacher, người dẫn đầu một nhóm những người đam mê Python quốc tế có tên là Poocco. Flask dựa trên bộ công cụ Werkzeg WSGI và template engine Jinja2. Cả hai đều là các dự án của Pocco. Micro ở đây không có nghĩa là framework này thiếu các chức năng mà thể hiện ở việc nó sẽ cung cấp những chức năng “cốt lõi” nhất cho các ứng dụng web và có khả năng mở rộng, người dùng cũng có thể mở rộng bất cứ lúc nào vì Flask hỗ trợ rất nhiều các tiện ích mở rộng như tích hợp CSDL, hệ thống upload, xác thực, template, email… Việc là một micro-framework cũng giúp cho flask có một môi trường xử lý độc lập và ít phải sử dụng các thư viện bên ngoài, điều này giúp nó nhẹ và ít gặp phải các lỗi hơn, việc phát hiện và xử lý các lỗi cũng dễ dàng và đơn giản hơn.

Ưu điểm:

- Siêu nhỏ nhẹ, là một công cụ tối giản.

- Tốc độ hoạt động cực nhanh.

- Tương đối đơn giản (so với các framework có cùng chức năng khác).

- Việc tạo 1 trang web được thực hiện nhanh chóng.

Nhược điểm: Chính vì siêu nhỏ nhẹ và tối giản, Flask không phải là một lựa chọn tốt nếu lập trình viên muốn một framework có đầy đủ các tính năng. Thay vào đó, lập trình viên sẽ phải tự gọi các tiện ích mà mình có nhu cầu sử dụng vì nó không được tích hợp sẵn trong framework

Do đó trong khi tạo extension sẽ dùng Flask để làm server để mỗi khi người dùng truy cập vào một URL thì extension sẽ gửi yêu cầu về server thông qua API để biết được xem URL này có khả năng độc hại hay không.

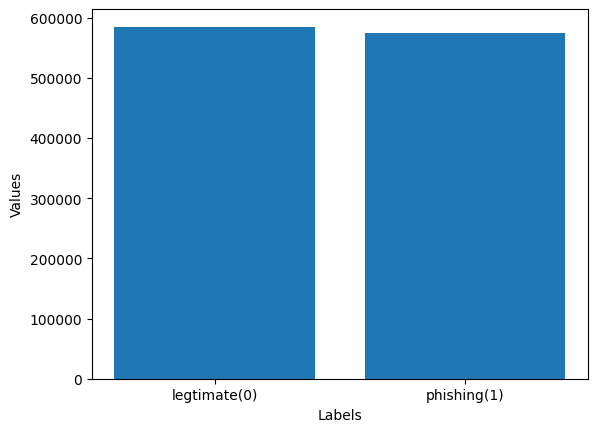
# **Chương III. Xây dựng mô hình và Extension.**

## **1. Tập dữ liệu.**

Nhóm sử dụng tập dữ liệu “1 Million Phishing Detection Dataset (1M-PD)” từ bài báo “URL2Vec: URL Modeling with Character Embeddings for Fast and Accurate Phishing Website Detection” của tác giả Huaping Yuan [4].

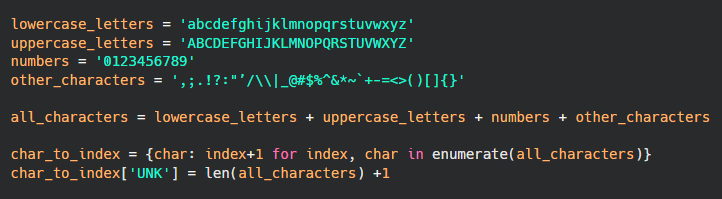
“1 Million Phishing Detection Dataset (1M-PD)” là một tập dữ liệu quy mô lớn có thể được sử dụng để phát hiện lừa đảo. Nó chứa 1 triệu URL và đã được phát hành để sử dụng công khai. Tập dữ liệu bao gồm 584909 URL của trang web tốt(benign) và 573964 URL của trang web lừa đảo(phishing).

Biểu đồ 3.1 Biểu đồ thể hiện số lượng các URL trong tập dữ liệu.

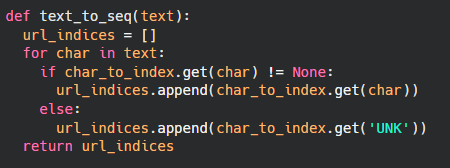


## **2. Mô hình sử dụng Character embedding.**

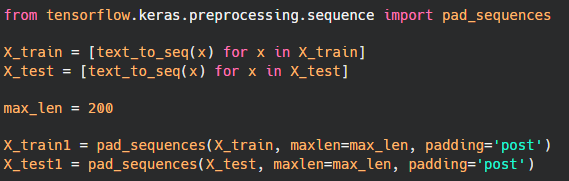
Tạo từ điển ký tự.



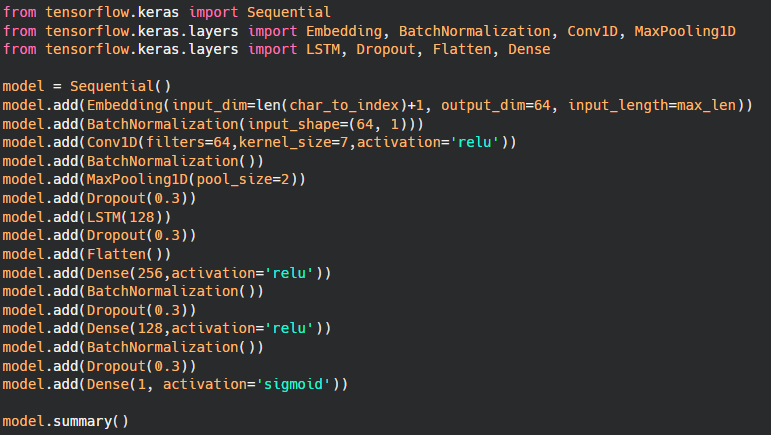
Hàm chuyển URL từ chuỗi ký tự thành chuỗi số theo chỉ mục.

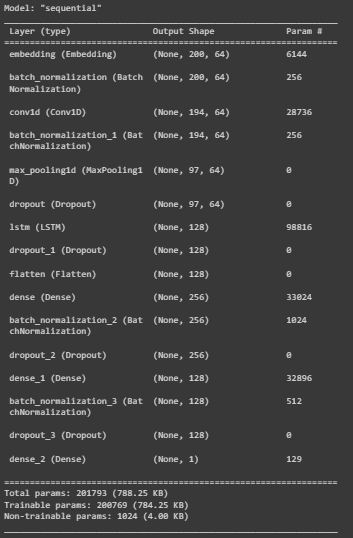


Sau đó sử dụng pad\_sequences để chuyển các chuỗi URL được số hoá về độ dài 200. Nếu độ dài của URL nhỏ hơn 200, phần còn lại sẽ được điền là 0. Nếu độ dài của URL lớn hơn 200, phần dài hơn sẽ bị cắt bỏ.



Xây dựng và huấn luyện mô hình [5].



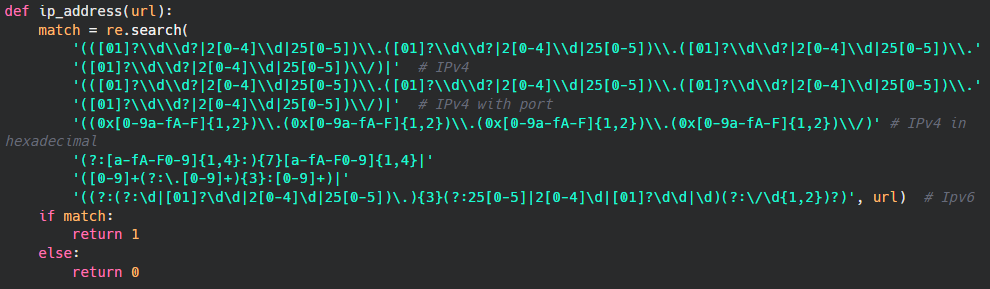




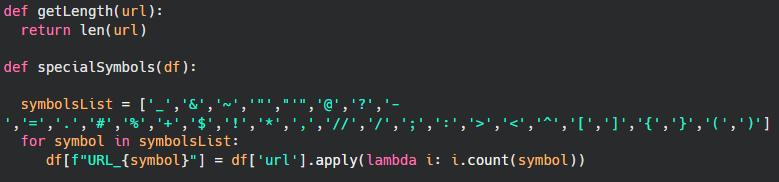
## **3. Mô hình sử dụng các đặc trưng trích xuất thủ công.**

Tạo các hàm để trích xuất các đặc trưng trong bảng 2.2.

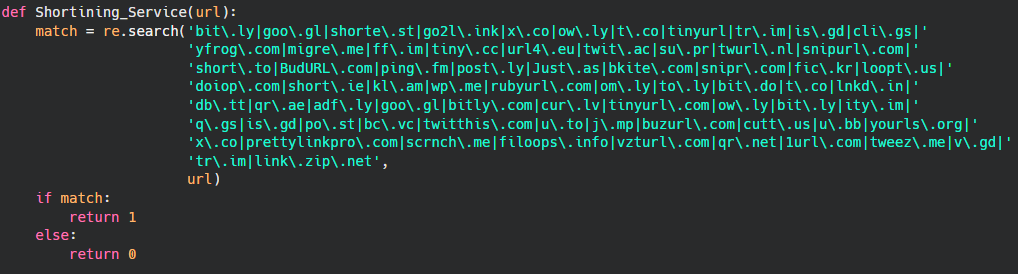
- Hàm kiểm tra có tồn tại địa chỉ IP trong URL.



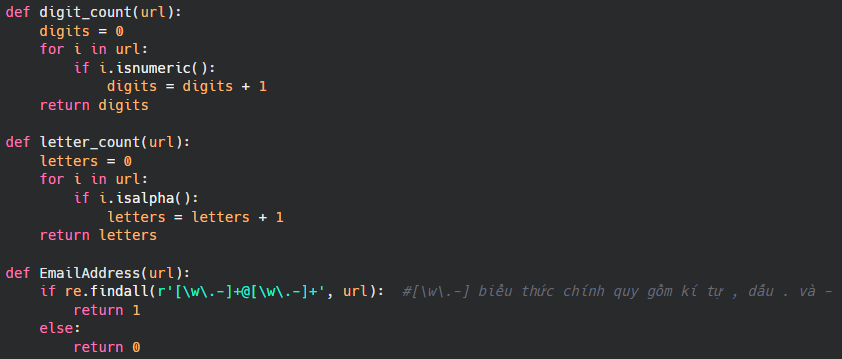
- Hàm lấy ra độ dài URL và hàm đếm số lượng các kí tự đặt biệt xuất hiện trong URL.



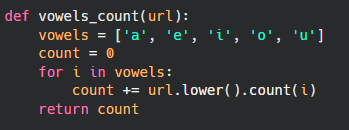
- Hàm kiểm tra URL có bị rút ngắn.



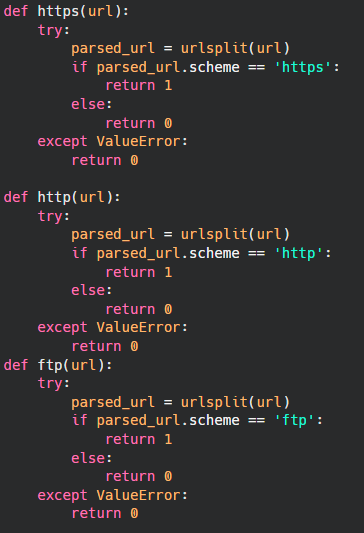
- Hàm đếm số lượng ký tự số, chữ cái, và hàm kiểm tra có tồn tại email.



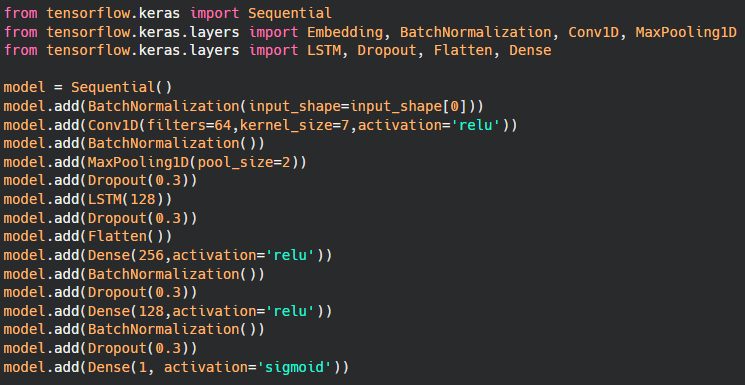
- Hàm đếm số lượng các kí tự nguyên âm.

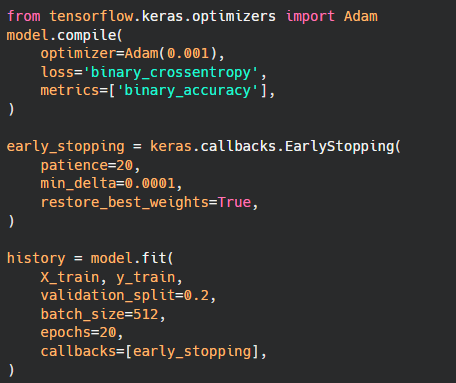


- Hàm kiểm tra có tồn tại các giao thức.



- Xây dựng và huấn luyện mô hình [5].





## **4. Đánh giá.**

Thông qua quá trình kiểm thử cho thấy mô hình sử dụng Character embedding đạt được độ chính xác cao nhất là 99,76%, mô hình sử dụng các đặc trưng trích xuất bằng tay có độ chính xác cao nhất là 97,84%.

|  |  |
| --- | --- |
| Training và validation accuracy. | Training và validation loss. |
| Hình 3.1 Đánh giá mô hình sử dụng Character embedding. | |
| Training và validation accuracy. | Training và validation loss. |
| Hình 3.2 Đánh giá mô hình sử dụng các đặc trưng trích xuất thủ công. | |

## **5. Triển khai Extension.**

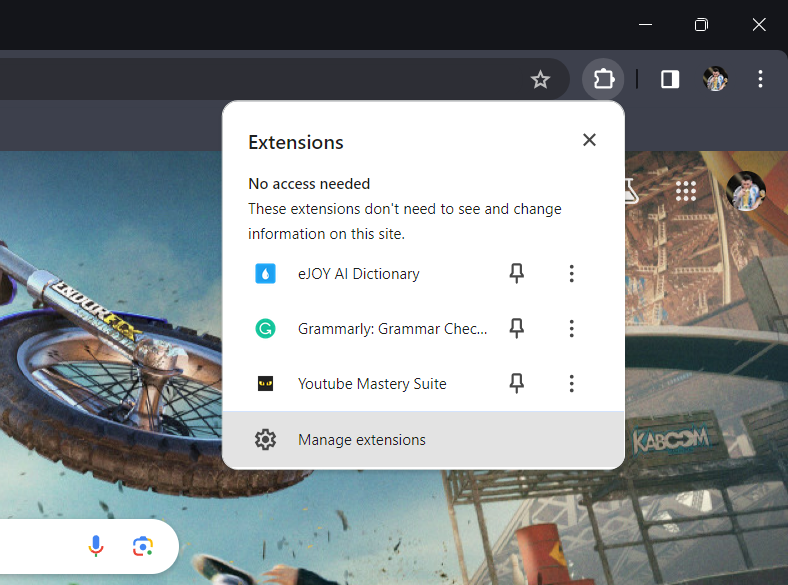
### **5.1. Tạo Extension.**

Tạo file manifest.json, file này có chức năng tạo ra 1 extension trên Chrome. Các thành phần chính trong file này :

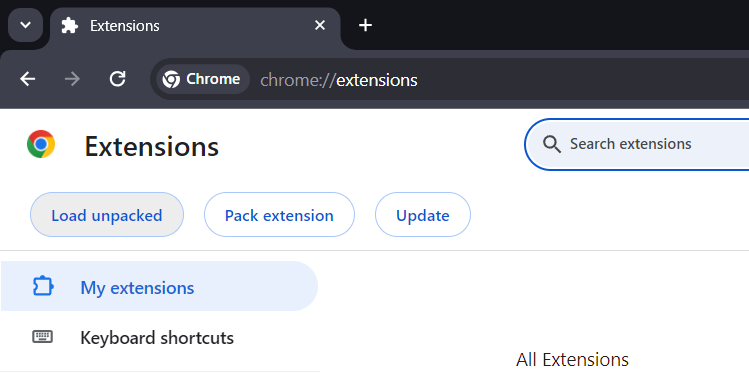


Sau khi tạo file manifest.json xong thì ta tiến hành thêm extension vào Chrome, việc thêm được thực hiện như sau :

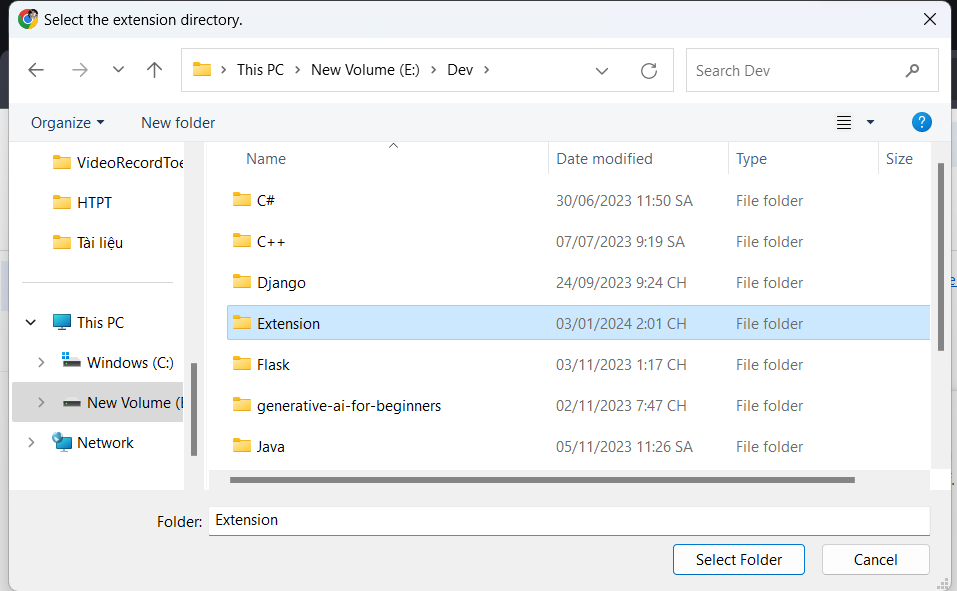
- Bước 1 : Mở Chrome và chọn Manage extensions.



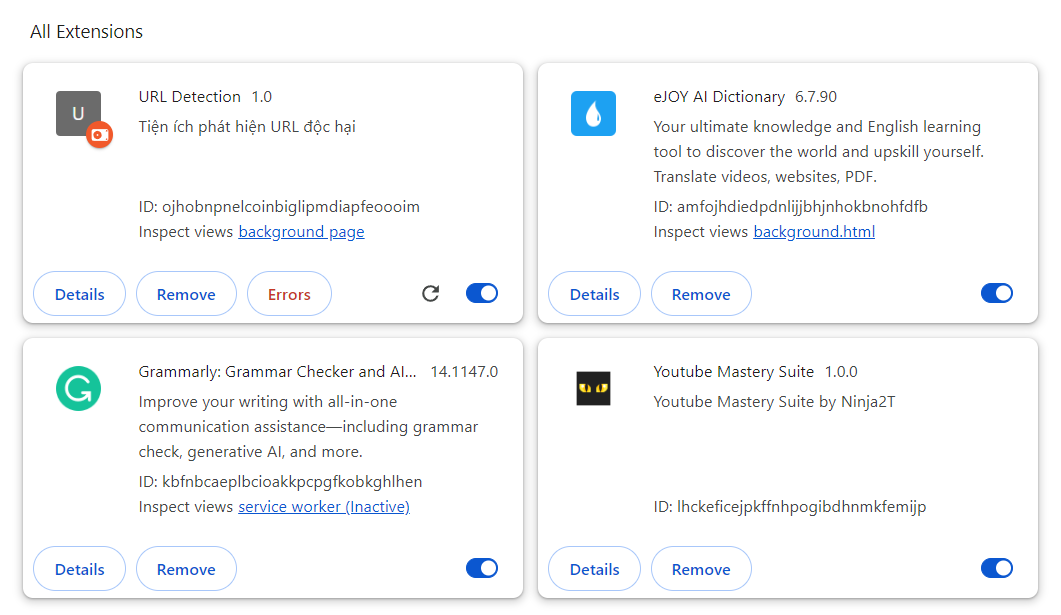
- Bước 2 : Chọn Load unpacked để tải code của extension này lên.



- Bước 3 : Chọn đường dẫn tới folder lưu chữ code của extension này.

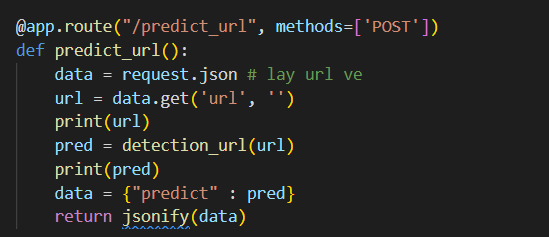


- Bước 4 : Sau khi hoàn thành, nếu thành công thì extension đã được thêm vào Chrome.

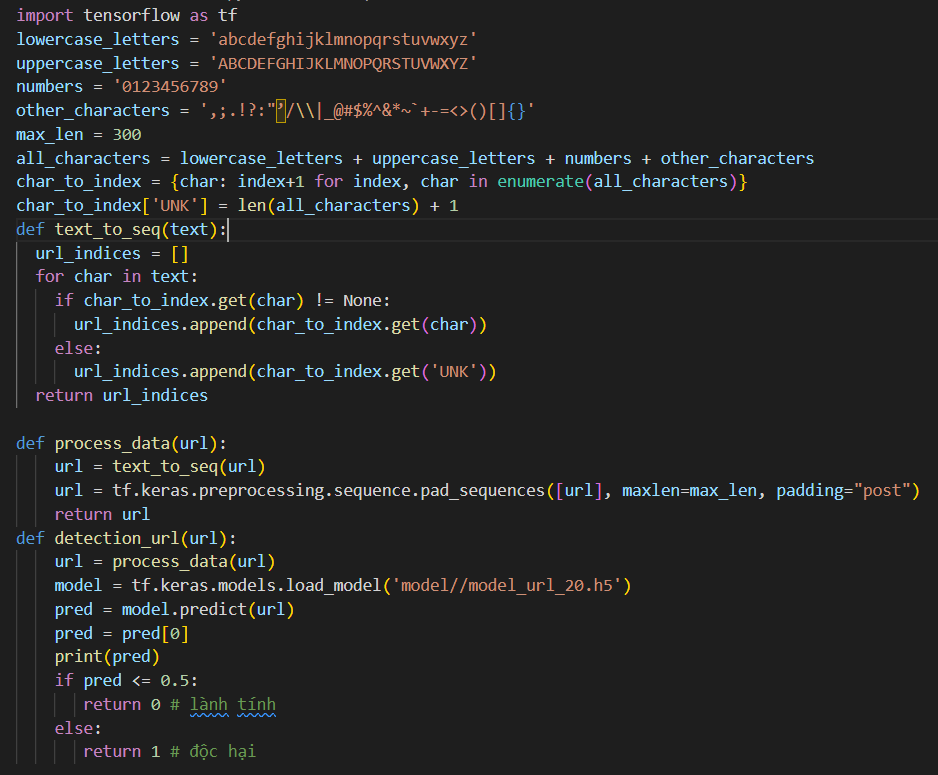


### **5.2. Tạo API.**

Việc tạo 1 API trong Flask rất đơn giản, ta sẽ tạo 1 API có tên là predict\_url với method là POST. Trong API này ta sẽ lấy url từ extension gửi về và thực hiện gọi hàm deetection\_url để đưa url qua model đã training.



Trước khi đưa url qua model thì ta cần đi xử lý dữ liệu gửi về. Hàm process\_data sẽ nhận vào là 1 url sau đó trả về là 1 ma trận sau khi đã thực hiện xử lý dữ liệu. Ta sẽ load model đã lưu và cho model đi thực hiện dự đoán xem url này có khả năng độc hại hay không.

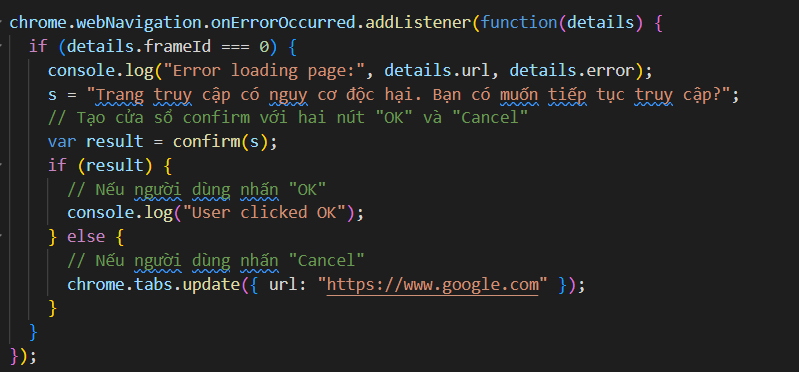


### **5.3. Gọi API và trả kết quả về cho người dùng.**

Trong file background.js ta đi lấy url khi người dùng nhập trên trình duyệt Chrome và sẽ gọi API đã viết ở trên. Ở đây ta sẽ dùng API của Chrome, cụ thể là chrome.webNavigation.onCompleted để lấy ra đường dẫn của trang người dùng đã truy cập thành công. Sau đó ta sẽ gọi API từ http://127.0.0.1:3001/predict\_url để model dự đoán kết quả.



Đối với các trang bị lỗi không thể tải hoàn tất trang, ta sẽ dùng chrome.webNavigation.onErrorOccurred để lấy lỗi và hiển thị thông báo.

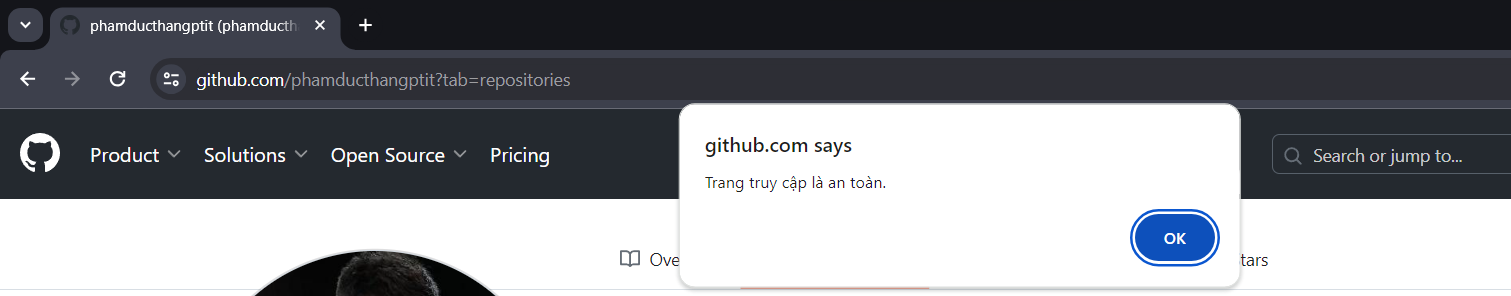


Sau khi model đã dự đoán xong ta sẽ gửi data để thông báo cho người dùng, việc hiển thị thông báo cho người dùng sẽ được thực thi trong file content.js. Ở đây ta sẽ lấy dữ liệu gửi lên, nếu action là checkURL thì ta sẽ đi thông báo ra cho người dùng. Nếu dữ liệu gửi lên là 0 thì thông báo cho người dùng url họ đang truy cập là an toàn, còn nếu dữ liệu gửi lên là 1 thì thông báo cho người dùng url họ đang truy cập có nguy cơ độc hại, và cho người dùng lựa chọn xem có tiếp tục truy cập vào trang đó hay không.

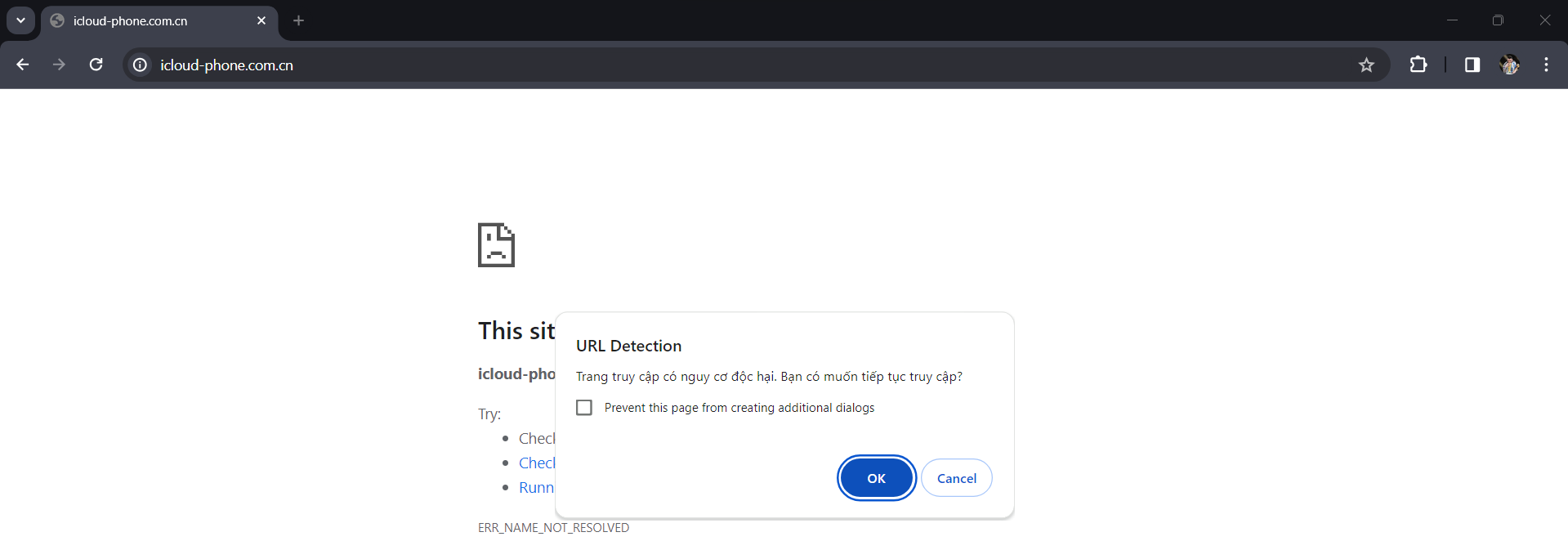


### **5.4 Kiểm thử.**

Đối với url an toàn: Extension sẽ hiện thông báo ra cho người dùng biết mình đang truy cập vào đường link an toàn.



Đối với url có nguy cơ độc hại: Extension sẽ hiện thông báo ra cho người dùng nguy cơ link này độc hại, nếu người dùng vẫn muốn tiếp tục truy cập thì nhấn Ok, còn khi người dùng ấn Cancel thì sẽ tự động chuyển về google.com.



# **Chương IV. Kết luận.**

Qua bài nghiên cứu này, nhóm đã xây dựng ra được mô hình dự đoán URL của trang web lừa đảo với độ chính xác cao. Mô hình chỉ sử dụng các đặc trưng từ URL mà không trích xuất nội dung của trang web để cải thiện khả năng tính toán và tài nguyên. Mô hình cũng không phụ thuộc vào ngôn ngữ của trang web, dù trang web có viết bằng bất cứ ngôn ngữ gì thì mô hình vẫn hoạt động tốt. Bên cạnh đó mô hình còn phát hiện được các trang web lừa đảo mới chưa được phân loại trước đó. Mô hình được ứng dụng vào thực tế thông qua Extension của Chrome, nó sẽ tự động nhận biết URL mà người dùng đang truy cập an toàn hoặc có khả năng là độc hại hay không, từ đó người dùng sẽ đưa ra quyết định xem có muốn tiếp tục truy cập vào URL đó hay không.

Bên cạnh đó, mô hình còn cũng còn một số hạn chế, khi URL mà người dùng muốn truy cập có nhiều query thì mô hình vẫn chưa đạt được mức độ chính xác cao, vẫn còn xảy ra nhầm lẫn ở một số URL lành tính. Lí do là vì tập dữ liệu sử dụng cho huấn luyện mô hình chưa được đa dạng để thể hiện rõ các đặc trưng của URL.

Trong tương lai, nhóm cố gắng cải thiện khả năng dự đoán của mô hình phù hợp với nhiều trường hợp trong thực tế bằng nhiều phương pháp khác. Thay vì chỉ sử dụng mô hình deep learning phát hiện trang web lừa đảo thì có thể kết hợp machine learning và deep learning hoặc trích xuất thêm các đặc trưng khác chỉ xuất hiện ở trang web lừa đảo mà trên URL không thể hiện được.

# **Tài liệu tham khảo.**

|  |  |
| --- | --- |
| [1] | nafisa\_anjum, "GeeksforGeeks," GeeksforGeeks, 29 June 2021. [Online]. Available: https://www.geeksforgeeks.org/components-of-a-url/. [Accessed 20 December 2023]. |
| [2] | A. Aljofey, "An Effective Phishing Detection Model Based on Character Level Convolutional Neural Network," Electronics 2020, 9, 1514, Basel, 2020. |
| [3] | Aws\_kyoma, "Kaggle," Kaggle, 21 October 2023. [Online]. Available: https://www.kaggle.com/code/awskyoma/malicious-url-detection-accuracy-95-61-ml-dl/notebook. [Accessed 21 December 2023]. |
| [4] | H. Yuan, "Github," Github Company, 15 May 2018. [Online]. Available: https://github.com/huapingz/ChracterEmbedding/tree/master. [Accessed 23 December 2023]. |
| [5] | Y. Zhang, "Kaggle," Kaggle, 21 April 2022. [Online]. Available: https://www.kaggle.com/code/yaleee/phishing-c2gw7fy2j4-3-lstm-cnn?fbclid=IwAR1rtNZRI82kN6PdEjeU0VzwL0-L1LQVGrxdrIfi9DM4j9T\_Q0Cf0feKEU4. [Accessed 5 December 2023]. |