Học phần:

Lập trình với Python

Báo cáo bài tập lớn

Machine learning: Mô hình học máy lọc thư Spam dựa trên phương pháp xây dựng mạng Nơ-ron tuần tự với Python

Nhóm 3

Tên thành viên	Mã sinh viên	Chuẩn bị
Nguyễn Đình Sơn	B21DCCN652	Thuyết trình, tổng hợp nội dung mọi người chuẩn bị và viết báo cáo, code phương pháp KNN + Support Vector Machine + Naïve Bayes, xây dựng và huấn luyện mô hình
Nguyễn Minh Đức	B21DCCN246	Tìm hiểu cách thu thập dữ liệu + chuẩn bị phần mềm, code giao diện + phương pháp phương pháp xây dựng mạng Nơ-ron tuần tự
Phạm Thị Thanh Hằng	B21DCCN328	Tìm hiểu các tiền xử lí dữ liệu + Flask, code BE + phương pháp Logistic Regression, viết hoàn thiện báo cáo, làm slide thuyết trình
Tống Quang Nam	B21DCCN556	Tìm hiểu hiện trạng, các phương pháp phân biệt mail spam, tìm dữ liệu mail spam để huấn luyên
Nguyễn Xuân Thức	B21DCCN700	Tìm hiểu học máy học sâu trong phân loại mail spam, tìm dữ liệu mail spam để huấn luyện

Tóm tắt :
Phát hiện thư rác *là một chủ đề phổ biến trong lĩnh vực '****Học máy, học sâu****'. Trong thực tế để đạt đến độ chính xác 100% khi phân loại thư là không thể vì cả nội dung và cách dùng từ ngữ có thể xuất hiện trong một lá thư là vô cùng rộng và không thể bị giới hạn. Tuy nhiên các lập trình viên luôn cố gắng để cải thiện hệ thống này đến mức chính xác nhất có thể. Nội dung báo cáo dưới đây của nhóm nhằm trình bày mô hình lọc thư rác dựa trên phương pháp* ***Xây dựng mạng Nơ - Ron tuần tự*** *, sử dụng module* ***Keras*** *của thư viện mở* ***Tensorflow****, với phần mở rộng là áp dụng phương pháp này trong việc phân loại thư điện tử được viết bằng* ***Tiếng Việt****, một ngôn ngữ phức tạp. Sau khi trải qua các bước tiền xử lý dữ liệu, kết quả cho thấy mô hình được huấn luyện bằng phương pháp này đạt độ chính xác đến* ***92%*** *với tập dữ liệu sử dụng Tiếng Anh và độ chính xác* ***90%*** *với tập dữ liệu sử dụng Tiếng Việt trong quá trình huấn luyện Model.*

1. Giới thiệu

1.1. Hiện trạng và nhu cầu lọc thư rác

Ngày nay, nhu cầu sử dụng thư điện tử là vô cùng lớn ở rất nhiều lĩnh vực. Cũng vì tính đa dụng của thư điện tử và đa mục đích của người sử dụng mà sự xuất hiện của thư rác hay thuật ngữ tiếng anh gọi là thư spam ngày một nhiều.

Thư spam là những thư điện tử có mục đích thương mại được gửi không theo yêu cầu từ bên gửi đến một danh sách rất dài các bên nhận một cách vô tội vạ, nội dung của thư spam thường chứa các thông điệp lừa đảo, các thông điệp quảng cáo, và các nội dung không liên quan đến người nhận.

Vì gây tiêu cực cho người nhận với nội dung quảng cáo không mong muốn đặc biệt là nội dung lừa đảo, vì vậy nên nhu cầu tự động loại bỏ loại thư này của người dùng thư điện tử ngày một tăng.

1.2. Những phương pháp lọc thư rác đã được sử dụng

Nhu cầu lọc thư rác đã có từ trước khi làn sóng trí tuệ nhân tạo và học máy bùng nổ, vậy nên trước khi các mô hình học máy và học sâu hiện đại được áp dụng người ta đã sử dụng một loạt các kỹ thuật để phân loại các thư điện tử pam như là Rule-based Filtering, Bayesian Filtering, Blacklist and Whitelist,... các phương pháp này chia ra thường đi theo 2 lối chính, hoặc là trích xuất từ vựng và cấu trúc thư để đánh giá đối tượng có phải thư spam hay không hoặc các phương pháp loại bỏ từ đầu để không nhận email ví dụ như thêm CAPTCHA hoặc chặn thẳng từ địa chỉ tên miền.

Không có sự so sánh giữa 2 hướng này mà để nâng cao hiệu quả người ta thường kết hợp chúng với nhau để hạn chế càng nhiều thư spam càng tốt. Tuy nhiên chúng không thể hiện tính hiệu quả khi các thư Spam ngày càng trở nên phức tạp hơn và biểu hiện của thư Spam ngày càng trở nên đa dạng và tinh vi hơn, không còn dựa vào các quy tắc cố định.

Chính điều đấy đã làm tiền đề và khuyến khích việc sử dụng phương pháp hiện đại học máy, học sâu vào quá trình phân tích và lọc thư rác.

1.3. Áp dụng học máy, học sâu vào phân loại thư rác

Trong một thập kỷ trở lại đây, học máy học sâu đã được sử dụng và áp dụng trong rất nhiều lĩnh vực như Thị giác máy tính, Xử lý âm thanh, Xử lý ngôn ngữ tự nhiên, Tự lái xe, Y tế dược phẩm, Kinh tế tài chính, Khoa học và công nghệ vật liệu,… trong đó nhánh trực tiếp đảm nhận xử lý và phân loại thư rác chính là Xử lý ngôn ngữ tự nhiên.

Xét về mặt tác vụ công việc thì phân loại thư rác được xếp là tác vụ 'Phân loại nhị phân (Binary Classification)' trong các tác vụ mà học máy học sâu có thể xử lý như 'Dự đoán dựa vào hồi quy (Regression)', 'nhận dạng đối tượng (Object Detection)', phân đoạn hoặc tạo mô hình,…

Đi sâu vào phân loại dành cho học máy thì có rất nhiều thuật toán được sử dụng phổ biến có thể kể đến Hồi quy Logitstic (Logistic Regression), Naive Bayes, KNN (K-Nearest Neighbors), Cây quyết định (Decision Tree), SVM (Support Vector Machine). Trong bài này thuật toán được sử dụng sẽ là ANN (Artificial Neutral Network) được điều chỉnh cho việc phân loại thư rác. Kết quả so sánh với các phương pháp phổ biến sẽ được ghi rõ trong phần nội dung và được phân tích.

Các lợi ích của việc áp dụng học máy học sâu là từ tập dữ liệu đầu vào mà máy tính có thể học và phát hiện những biểu hiện phức tạp của thư rác mà không bị dựa trên những quy tắc cố định, ngoài ra việc được thêm vào các cơ chế tự động điều chỉnh các thông số để cải thiện hiệu suất sẽ giúp các mô hình thích nghi tốt hơn với các biểu hiện của thư rác mà không cần phải điều chỉnh thủ công, chưa kể kết hợp với các phương pháp tiền xử lý dữ liệu và khả năng xử lý các loại dữ liệu phi cấu trúc giúp cho mô hình tăng cường hiệu suất.

Nhờ sự bùng bổ của trí tuệ nhân tạo mà nhiều phương pháp xử lý dữ liệu và tối ưu quá trình đã xuất hiện và được áp dụng, trong bài này cũng sẽ áp dụng một vài phương pháp đó như Adam cho Optimizer, các hàm Callback, và các lớp xử lý trong mô hình dạng mạng Nơ-ron như LSTM và các hàm kích hoạt xử lý các mối quan hệ phi tuyến tính.

Tuy nhiên chất lượng của mô hình xử lý vẫn bị phụ thuộc rất nhiều vào tập dữ liệu đầu vào, điều này đòi hỏi tập dữ liệu phải được cập nhập liên tục và mô hình cũng phải được chăm sóc và tinh chỉnh để đối phó với các kỹ thuật thay đổi của các đối tượng spam.

2. Nội dung

2.1. Thu nhập dữ liệu

Vì phương pháp được sử dụng cho mô hình là học máy có giám sát vậy nên yêu cầu tập dữ liệu trong bài này cần được đánh dấu dán nhãn trước.

Tập dữ liệu cũng sẽ được phân chia thành tập huấn luyện và tập kiểm tra để sử dụng cho huấn luyện mô hình và kiểm thử, vậy nên tính thực tế của dữ liệu sẽ cho thấy khả năng của mô hình khi áp dụng vào thực tế.

Dữ liệu sẽ được sử dụng để huấn luyện yêu cầu sự chính xác và đa dạng để mô hình có thể học tập chính xác và xử lý mạnh mẽ trong nhiều trường hợp. Các kiến thức, quy luật và mối quan hệ giữa các đối tượng trong tập dữ liệu, trong bài này cụ thể là giữa các từ sẽ được mô hình học máy học tập.

Để phù hợp cho việc so sánh và đánh giá mô hình trong bài, file CSV chứa nội dung các thư được đánh dấu sẵn là spam hay không được thu nhập từ Kaggle. Dữ liệu tiếng anh do nền tảng này cung cấp được sử dụng trong bài này có độ tin cậy và được áp dụng nhiều trong các mô hình trước đó.

Đối với tập dữ liệu tiếng việt cho nội dung phân biệt thư spam vì còn hạn chế nên tập dữ liệu sẽ được thu nhập từ nguồn cá nhân và sử dụng các công cụ hỗ trợ.

Dữ liệu stopword Tiếng việt được thu nhập trên github và sẽ được giải thích cụ thể ở phần tiền xử lý dữ liệu.

2.2. Chuẩn bị phần mềm.

Với ngôn ngữ lập trình được sử dụng để tạo và huấn luyện mô hình học máy lọc thư rác là Python thì nhóm quyết định lựa chọn triển khai mã nguồn dưới dạng tệp ipynb để phù hợp cho báo cáo và và dạng tệp .py cơ bản cho lập trình.

ipynb (Ipython Notebook) là một định dạng tệp tin được sử dụng trong Jupyter Notebook, một môi trường tích hợp phát triển cho ngôn ngữ lập trình Python, phù hợp để tạo và chia sẻ các tài liệu tính toán để phân tích. Loại tệp này là một tệp chứa thông tin về cấu trúc của note - book bao gồm các ô chứa mã, kết quả của mã, văn bản và hình ảnh. Việc lựa chọn và triển khai sử dụng mã nguồn của mô hình dưới dạng tệp này cho phép theo dõi sát kết quả của từng giai đoạn trong suốt quá trình huấn luyện và chạy thử mô hình, rất thuận tiện cho việc sửa chữa và tinh chỉnh mã nguồn cũng như dữ liệu đầu vào cho phù hợp với mục đích lập trình.

Có nhiều môi trường có thể sử dụng để chạy tệp ipynb, trong quá trình huấn luyện mô hình thì nhóm sử dụng Vscode (Visual Studio Code) trong cho việc chạy mã nguồn của mô hình. Phiên bản Python được sử dụng trong nhân xử lý khi Vscode chạy mã nguồn định dạng ipynb là 3.9.

Bài tập lớn sử dụng các thư viện :

Pandas : là một thư viện mã nguồn mở được sử dụng trong ngôn ngữ lập trình Python để xử lý và phân tích dữ liệu.

Flask : Framework phát triển ứng dụng web cho Python, sử dụng để giao tiếp dữ liệu với người dùng thông qua web.

nltk (Natural Language Toolkit) : một thư viện Python cung cấp các công cụ trong lĩnh vực xử lý ngôn ngữ tự nhiên

wordcloud : Thư viện cung cấp công cụ vẽ hình ảnh các đám mây từ khóa, đóng vai trò trong việc thể hiện độ ưu tiên của từng ngữ trong phân loại thư rác.

tensorflow : Thư viện mở hỗ trợ mạnh mẽ các phép toán học trong tính toán machine learning và deep learning

scikit – learn : Thư viện cung cấp các công cụ để xây dựng và huấn luyện các mô hình học máy.

matplotlib, seaborn : 2 thư viện được sử dụng để tạo biểu đồ giúp thể hiện trực quan dữ liệu

2.3. Tiền xử lý dữ liệu

Như đã nói, chất lượng của mô hình phụ thuộc rất nhiều vào tập dữ liệu đầu vào vậy nên việc tiền xử lý dữ liệu trước khi đưa vào cho mô hình áp dụng các thuật toán để học là vô cùng quan trọng.

Với 2 tập dữ liệu tiếng anh và tiếng việt được sử dụng để huấn luyện mô hình trong việc phân loại thư rác, bước đầu tiên là đọc tập dữ liệu này vào môi trường thông qua lệnh read_csv được hỗ trợ trong thư viện pandas để đọc 2 tệp data_en.csv (Tập dữ liệu huấn luyện tiếng Anh) và data_vi.csv (Tập dữ liệu huấn luyện Tiếng Việt).

Mỗi tập dữ liệu bao gồm 2 cột : Cột thứ nhất ‘spam’ chứa nội dung dán nhãn phân loại nhị phân với giá trị 0 biểu thị không phải thư rác và giá trị 1 biểu thị là thư rác, tương ứng với các giá trị được dán nhãn ở cột 1 và nội dung lá thư ở cột 2 ‘text’.

```
data_en.head()
```
✓ 0.0s

	spam	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

```
data_vi.head()
```
✓ 0.0s

	spam	text
0	0.0	Hãy đến cho đến Jurong Point, điên rồ .. Chỉ c...
1	0.0	Ok lar ... nói đùa wif u oni ...
2	1.0	Nhập viện miễn phí trong 2 WKLY Comp để giành ...
3	0.0	Bạn không nói quá sớm ... bạn đã nói ...
4	0.0	Không, tôi không nghĩ anh ấy đến USF, anh ấy s...

Sau khi hoàn thành đọc dữ liệu. Mỗi mô hình huấn luyện sẽ được nhận dữ liệu đầu

vào tương ứng với ngôn ngữ của thư rác mà mô hình được đảm nhận để phân loại.

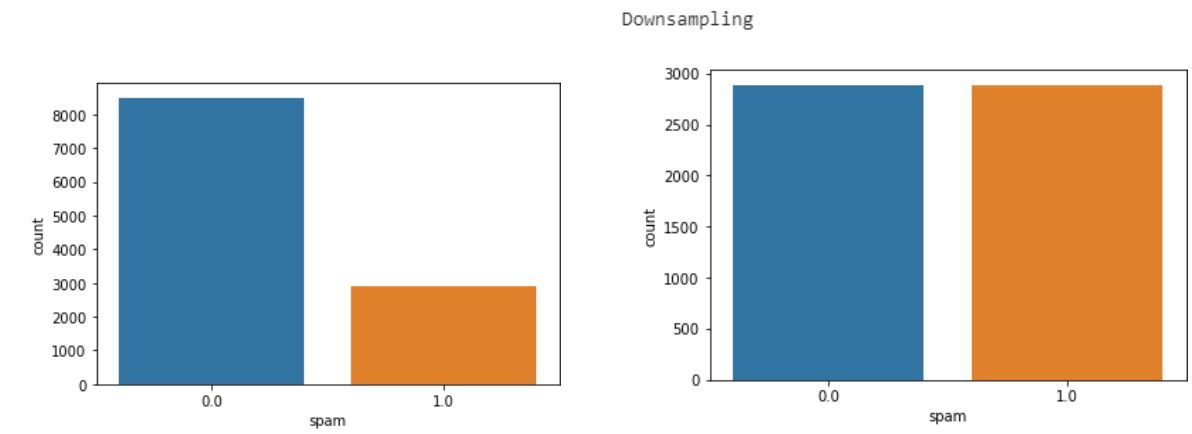
Từ phần này trở đi các bước tiền xử lý có sự tương đồng ở cả 2 mô hình huấn luyện để lọc thư rác tiếng anh và tiếng việt.

2.3.1. Cân bằng tập dữ liệu

Trong thực tế số lượng mail ham nhiều hơn rất nhiều so với số lượng mail spam, vậy nên nếu cứ để mô hình nhận tập dữ liệu và tiến hành học thông qua thuật toán sẽ dễ xảy ra hiện tượng mô hình học lệch về phần dữ liệu đa số là thư bình thường và thể hiện kết quả nghèo nàn với kết quả về thư spam.

Do đó với phương pháp cân bằng tập dữ liệu sẽ ngăn chặn tình trạng này và đồng thời giảm tỉ lệ dự đoán sai cả thư spam và thư không spam.

Dưới đây là biểu đồ thể hiện tập dữ liệu trước và sau khi cân bằng.



2.3.2. Xử lý văn bản

Vì là một chủ đề trong xử lý ngôn ngữ tự nhiên nên việc xử lý ngôn ngữ tự nhiên dưới dạng văn bản trong tập dữ liệu đầu vào đóng một vai trò quan trọng, các bước của xử lý văn bản bao gồm :

Loại bỏ Stopword: Stopword là những từ xuất hiện thường xuyên trong văn bản và không mang nhiều ý nghĩa về mặt nội dung của văn bản đó, vốn dĩ sự xuất hiện của những từ này không mang giá trị khi phân tích trong trường hợp này vì chúng thường xuất hiện ở cả thư spam và thư ham.

So với tiếng anh, Stopword trong tiếng việt có sự xuất hiện của các từ ghép từ nhiều chữ, phức tạp hơn so với danh sách các Stopword trong tiếng anh, vậy nên phương pháp phổ biến để loại bỏ các Stopword trong tiếng anh là chuyển đổi nội dung text thành 1 list các word và duyệt toàn bộ để loại bỏ từng chữ là không khả thi khi áp dụng với Stopword tiếng việt.

Thay vào đó vì phần lớn nguyên nhân đến từ sự xuất hiện của các cụm từ Stopword trong tiếng việt vậy nên người lập trình có thể sử dụng mảng đánh dấu kí tự để loại bỏ

các stopword. Trước hết chuyển đầu vào thành chữ thường do tập các stopword là chữ thường, hơn nữa là để thống nhất dữ liệu trong việc xử lí. Sau đó cần loại bỏ các khoảng trống thừa giữa các từ, mỗi từ chỉ cách nhau 1 khoảng trống. Đầu và cuối đầu vào được đệm thêm 1 khoảng trống. Do đó stopword bị loại bỏ sẽ cách nhau 1 khoảng trống.

Duy trì 1 mảng có độ dài bằng đầu vào, khởi tạo giá trị = 0. Duyệt các stopword trong tập stopword-vi, đệm thêm khoảng trống ở đầu và cuối mỗi lần duyệt. Tìm kiếm trong xâu đầu vào, nếu xuất hiện stopword thì tiến hành đánh dấu loại bỏ. Đánh dấu được đánh từ vị trí tìm thấy +1 đến chiều dài -1(do đã thêm khoảng trống ở đầu và cuối).

Kết thúc duyệt, nối tất cả các kí tự trong xâu đầu vào chưa bị đánh dấu, xoá bỏ các khoảng trống thừa thu được xâu đã chuẩn hoá.

Để phục vụ hiệu quả cho việc loại bỏ những từ vô nghĩa thì trước đấy phải loại bỏ dấu câu và loại bỏ những ký tự tin hoa để thuận tiện cho việc so sánh và tìm kiếm các từ hoặc cụm từ Stopword thông qua danh sách các từ này được chuẩn bị sẵn.

Dù vậy trong tiếng Anh tồn tại một vấn đề là có nhiều thể của một từ tùy ngữ cảnh, vậy nên có 2 cách để xử lý và đưa một từ tiếng anh về thể gốc của nó là Stemming và Lemmatization, kỹ thuật Stemming chỉ đưa 1 từ về dạng gốc của nó bằng cách loại bỏ hậu tố, Lemmatization sẽ xem xét cả ngữ cảnh của câu để đưa 1 từ về dạng gốc. Tóm lại Lemmatization thể hiện tốt hơn trong những trường hợp cần xem xét về ngữ nghĩa.

Dưới đây là kết quả trước và sau khi tiền xử lý dữ liệu đối với tập dữ liệu sử dụng Tiếng Việt.

	spam	text
0	0.0	Chủ đề: Ước tính sản xuất tháng 7\ndaren,\nFYI...
1	0.0	Tôi thức dậy từ lâu rồi ... Dunno, điều gì khác?
2	0.0	Takin một vòi hoa sen bây giờ nhưng vâng, tôi ...
3	0.0	Hahaha.. Sử dụng bộ não của bạn thân yêu
4	0.0	K Tôi sẽ chăm sóc nó
...	...	...
5771	1.0	Bạn là người chiến thắng may mắn của rút thăm ...
5772	1.0	Gặp gỡ người bạn tâm giao của bạn trực tuyến n...
5773	1.0	Làm việc tại nhà - Kiếm tiền lớn nhanh chóng
5774	1.0	Cơ hội việc làm - mở cửa ngay lập tức
5775	1.0	Thay đổi mật khẩu Facebook của bạn ngay bây giờ!

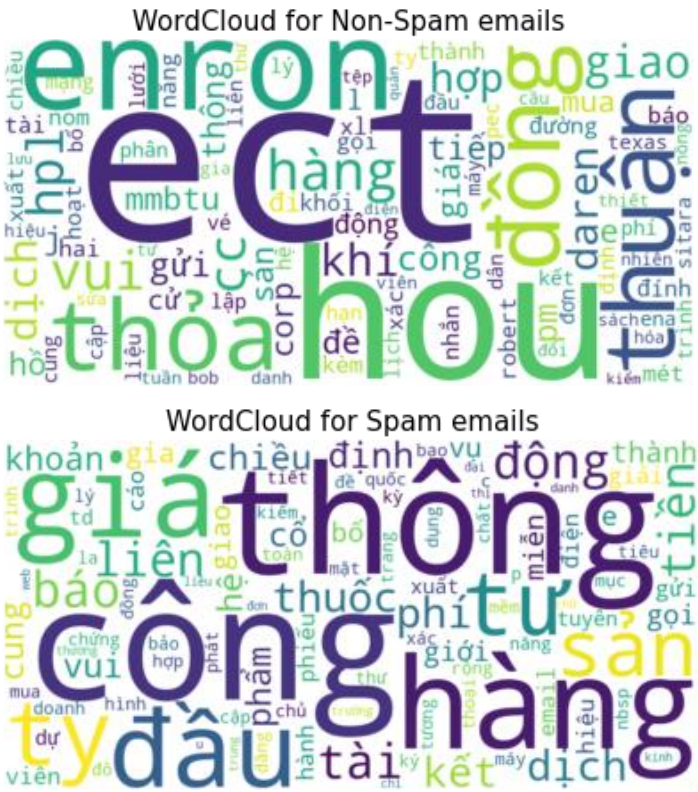
5776 rows × 2 columns

	spam	text
0	0.0	ước sản xuất 7 daren fyi bob tiếp robert cotte...
1	0.0	thức dậy dunno
2	0.0	takin vòi hoa sen rời đi hoàn thành
3	0.0	hahaha não thân yêu
4	0.0	k chăm sóc
...	...	...
5771	1.0	chiến thắng may mắn rút thăm trúng thưởng coca...
5772	1.0	gỡ tâm giao trực tuyến hôm
5773	1.0	kiếm tiền chóng
5774	1.0	cửa
5775	1.0	mật khẩu facebook

5776 rows × 2 columns

Sử dụng công cụ WordCloud từ thư viện wordcould, đây là một công cụ cho phép người sử dụng trực quan hóa dữ liệu đưa vào, trong trường hợp này công cụ hỗ trợ cho lập trình viên lấy được thông tin về những từ ngữ có tầng suất xuất hiện cao nhất trong mỗi phân loại thư điện tử Spam hoặc là Ham.

Kết quả mô hình được đưa ra như dưới hình :



2.3.3. Chia tập, số hóa dữ liệu và Padding

Chia tập :

Chia tập huấn luyện và tập kiểm tra là bước đầu tiên.

Sử dụng hàm train_test_split từ thư viện sklearn ta chia tập dữ liệu thành 4 bao gồm : Tập train_x, test_x, và train_y, test_y.

Tham số test_size nhập vào trong hàm có ý nghĩa thể hiện tỷ lệ dữ liệu kiểm tra, với giá trị 0,2 mang ý nghĩa 20% tập dữ liệu sẽ được dùng để kiểm tra trong khi 80% còn lại sẽ được dùng để huấn luyện.

Việc chia dữ liệu thành các tập nhằm kiểm tra liệu mô hình có hoạt động tốt sau khi được train kể cả với tập dữ liệu chưa được gặp trước đó hay không. Tránh trường hợp mô hình bị Overlifting khi hoạt động rất tốt trên tập dữ liệu huấn luyện và thể hiện kém trên tập dữ liệu hoàn toàn khác so với dữ liệu mô hình đã được học.

Số hóa dữ liệu :

Vì các mô hình học máy chỉ hoạt động với dạng dữ liệu số nên ta không thể trực tiếp đưa những từ vựng từ tập dữ liệu vào thẳng cho mô hình mà phải trải qua bước chuyển các dữ liệu ngôn ngữ tự nhiên sang kiểu dữ liệu số học mà mô hình có thể sử dụng.

Phương pháp được sử dụng để giải quyết vấn đề trong trường hợp này là lớp Tokenizer và lớp text_to_sequences trong module Keras của thư viện TensorFlow.

Trước khi có thể số hóa một từ, sử dụng lớp Tokenizer và hàm fit_on_texts, ta có thể tạo ra một thư viện bao gồm mỗi một số nguyên duy nhất được gán cho mỗi từ trong từ điển, chẳng hạn từ điển sẽ đánh dấu và mã hóa các từ thông dụng trong email spam như giá, công, thông hàng, thành các số nguyên lần lượt là 1 2 3 4.

Sau khi đã tạo ra một thư viện bao gồm các ký tự đã được chuyển đổi thành mã số nguyên duy nhất, mô hình sử dụng thư viện cùng hàm fit_on_texts để tiến hành số hóa dữ liệu tập dữ liệu được đưa vào trong hàm, trong trường hợp này chính là tập ký tự được xử lý văn bản từ nội dung thư từ cả tập huấn luyện và tập kiểm tra là train_X và test_X.

Padding:

Mô hình mạng Nơ – ron tuần tự hoạt động dựa trên dữ liệu đầu vào cố định, tuy nhiên dữ liệu kể cả sau khi được số hóa thì độ dài của mỗi chuỗi là không thống nhất nên cần một bước gọi là padding để xử lý vấn đề này.

Pad_sequences là một hàm được gọi từ module Keras của thư viện Tensorflow.

Lựa chọn một giá trị tối đa cho độ dài của chuỗi số, nếu sau quá trình số hóa, chuỗi số có độ dài chưa đạt đủ thì padding sẽ làm nhiệm vụ thêm vào một chuỗi, với giá trị padding thêm vào mỗi chuỗi số chưa đủ độ dài là ký tự 0 dựa vào giá trị tham số padding = post trong hàm. Đối với những chuỗi sau số hóa có độ dài lớn hơn độ dài cố định thì padding làm nhiệm vụ xóa bớt những ký tự phía sau đi thông qua giá trị truncating = ‘post’ được nhập vào hàm. Trong trường hợp bài tập của nhóm chọn giá trị tối đa cho padding bằng 100.

Tổng kết dữ liệu sau tiền xử lý : 80% Train 20% Test

```
print(train_sequences)
train_sequences.shape
```
✓ 0.0s Python

```
[[  485    72   566 ...  4572   301   105]
 [ 7073   547     9 ...     0     0     0]
 [ 4579  1200 11158 ...     0     0     0]
 ...
 [  376    21   153 ...     0     0     0]
 [   25    60   836 ...     0     0     0]
 [  525  2469    24 ...     0     0     0]]

(4620, 100)
```

```
print(test_sequences)
test_sequences.shape
```
✓ 0.0s Python

```
[[  662   731   271 ...     0     0     0]
 [   70   327   416 ...     0     0     0]
 [  945   341  3769 ...  6221  1068 28658]
 ...
 [  562   115   771 ...  2115     6     1]
 [   90   232   103 ...     1    61   259]
 [   11   285    63 ...     0     0     0]]

(1156, 100)
```

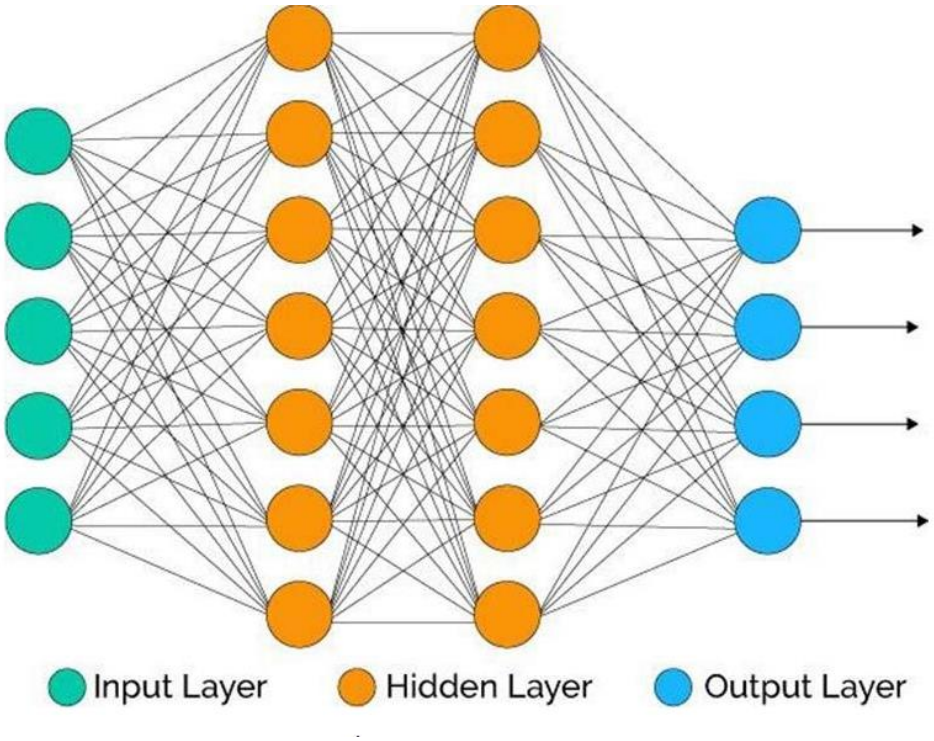
2.4. Xây dựng và huấn luyện mô hình

2.4.1. Xây dựng:

Như đã ở trên có nhiều phương pháp để huấn luyện mô hình học máy phân biệt thư rác. Trong bài này nhóm sẽ huấn luyện phương pháp chính là xây dựng mạng Nơ – ron tuần tự, đồng thời sẽ so sánh với các phương pháp phổ biến khác đã được dùng trong phân loại thư rác.

Xây dựng mạng Nơ – ron nhân tạo (ANN) là một phương pháp xây dựng kiến trúc Nơ – ron cho mô hình học máy dựa trên cấu tạo của mạng Nơ -ron trong não người, với phương pháp xây dựng mạng Nơ – ron cụ thể được nhóm tham khảo và sử dụng trong bài là Xây dựng mạng Nơ – ron tuần tự (SNN) Cấu tạo của cấu trúc này là tuyến tính các lớp Nơ- ron được xếp chồng lên nhau trình tự, và dữ liệu được truyền qua mạng từ đầu đến cuối theo một hướng.

Dưới đây cấu trúc của một mạng Nơ- ron nhân tạo mẫu.



Input Layer : Lớp dữ liệu đầu vào.
Hideen Layer : Lớp ẩn.
Output Layer : Lớp dữ liệu đầu ra.

Một mạng nơ-ron nhân tạo có 1 lớp dữ liệu đầu vào, 1 lớp dữ liệu đầu ra nhưng có thể có thể nhiều lớp ẩn.

Trong bài này, lớp đầu tiên hoạt động dưới mô hình một mạng nơ-ron tiêu chuẩn để xử lý và đưa dữ liệu cho các lớp sau là lớp nhúng :

Lớp đầu tiên : Embedding Layer.

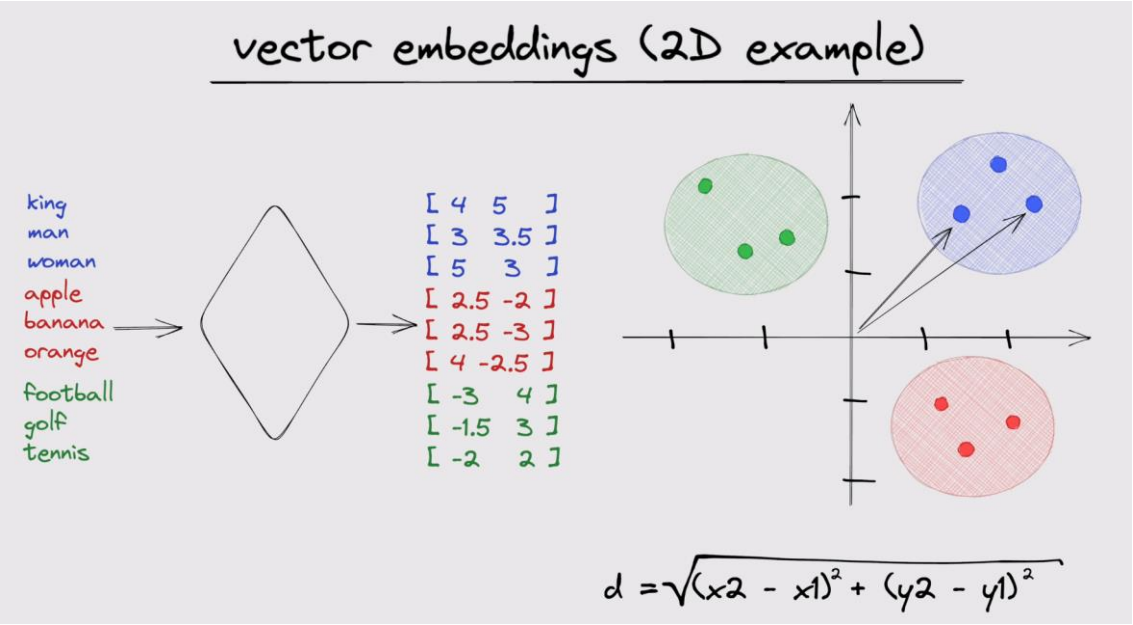
Thông qua câu lệnh:
model.add(tf.keras.layers.Embedding(input_dim=len(tokenizer.word_index+ 1, output_dim=32, input_length=max_len))

Câu lệnh trên khởi tạo 1 lớp đầu tiên trong mô hình có tác dụng biểu diễn từng chuỗi số input thành các chuỗi vector nhúng.

Dưới đây là ví dụ giải thích về Vector nhúng.

Mỗi số trong chuỗi số tồn tại trong thư viện ban đầu (input_dim = len(tokenizer.word_index +1)) sẽ ánh xạ đến 1 điểm trên không gian đa chiều, ở đây lựa chọn 32 chiều. Đường thẳng đi từ gốc đến điểm đó chính là vector nhúng của từ đó sau khi được chuyển đổi. Mô hình sẽ chuyển đổi sao cho khoảng cách của những vector biểu thị

cho các từ có nghĩa gần nhau cũng sẽ có khoảng cách vector gần nhau nhất tính theo công thức khoảng cách Vector trên không gian Eculid.



Lớp thứ 2 : LSTM (Long Short Term Memory).

Thông qua câu lệnh dưới để thêm lớp LSTM.

model.add(tf.keras.layers.LSTM(16))

Có tác dụng học ngữ nghĩa thông qua các chuỗi vector nhúng sau lớp 1, đồng thời nhờ cơ chế lưu trạng thái ẩn mà có thể lưu trữ thông tin ngữ nghĩa qua các giai đoạn, thông số 16 nhập vào là số Unit mà LSTM dùng để học dữ liệu, số này quá lớn sẽ gia tăng khả năng mô hình có thể hiểu những ngữ nghĩa phức tạp nhưng đồng thời cũng khiến cho thời gian chạy code tăng lên và có nguy cơ dẫn đến overlifting.

So với SNN (Mạng Nơ ron tuần tự) thì (RNN) mạng nơ ron hồi quy có khả năng học các mối liên hệ giữa các đối tượng hiệu quả hơn vì có cơ chế lưu lại các trạng thái trong quá khứ, LSTM được áp dụng vào code ở trên là một dạng RNN phức tạp giúp tăng hiệu suất của mô hình.

Lớp thứ 3 : Lớp kết nối toàn diện.

Vì dữ liệu đưa cho máy tính xử lý là các con số vậy nên mối quan hệ giữa chúng sẽ được mô tả bằng các hàm tuyến tính, người ta gọi là các mối quan hệ tuyến tính, tuy nhiên trong việc xử lý ngôn ngữ tự nhiên sẽ xuất hiện các mối quan hệ phi tuyến tính như tăng giảm dần,…

LSTM làm tốt vai trò học các mối quan hệ ngữ nghĩa tuyến tính tuy nhiên cần thêm 1 lớp làm nhiệm vụ học các mối quan hệ phi tuyến tính, lúc này lớp thứ 3 sẽ làm nhiệm vụ này :

Được thêm vào bằng câu lệnh:

model.add(tf.keras.layer.Dense(32, activation = ‘relu’))

Với 32 Unit No-ron ở lớp ẩn được kết nối với tất cả Unit Nơ-ron đầu vào và ra gọi là mạng Nơ-ron dày đặc (Dense), sau quá trình học, hàm kích hoạt Relu được sử dụng để giúp mô hình học các đặc trưng phi tuyến tính trong dữ liệu.

Lớp thứ 4 : Lớp dự đoán.

Được thêm vào bằng câu lệnh.

model.add(tf.keras.layers.Dense(1, activation = ‘sigmod’))

Nhận vào kết quả từ lớp trước, lớp cuối cùng chỉ tạo duy nhất 1 giá trị, với hàm kích hoạt sigmod.

Trong mô hình này tác vụ yêu cầu là phân loại nhị phân nên ngưỡng (threshold) là 0.5 và hàm sẽ so sánh giá trị đưa ra từ Unit với giá trị ngưỡng, nếu lớn hơn thì giá trị output từ lớp này là 1, ngược lại là 0.

Cấu hình huấn luyện :

Sử dụng hàm model.compile để cấu hình:

```
model.compile(loss = tf.keras.losses.BinaryCrossentropy(from_logits = True),
              metrics = ['accuracy'],
              optimizer = 'adam')
```

Optimizer là thuật toán tối ưu được sử dụng cho mô hình, được sử dụng trong bài là ‘Adam’, một kỹ thuật hiệu quả và phổ biến để mô hình tự điều chỉnh trọng số khi học.

Metrics: Cho phép đánh giá mô hình thông qua thông số được lựa chọn ở đây là accuracy.

Loss: Hàm mất mát, tính toán tỉ lệ mất mát, cụ thể là độ lệch của kết quả mô hình đưa ra với kết quả trong tập dữ liệu.

2.4.2. Huấn luyện:

Trước khi bắt đầu huấn luyện 2 hàm Callback sẽ được thêm vào để làm giảm thời gian chạy của Code.

```
es = EarlyStopping(patience=3,
                   monitor = 'val_accuracy',
                   restore_best_weights = True)

lr = ReduceLROnPlateau(patience = 2,
                       monitor = 'val_loss',
                       factor = 0.5,
                       verbose = 0)
```

Trong đó làm ‘es’ sẽ đánh giá nếu như giá trị độ chính xác chạy trên tập kiểm thử sau 3 epoch (3 vòng lặp) không tốt hơn sẽ tự đưa về giá trị tốt nhất trước đó.

Hàm ‘lr’ còn lại sẽ kiểm tra nếu như giá trị mất mát vượt quá 1 nửa, tức là mô hình có độ sai lớn hơn 50% sau 2 vòng lặp thì sẽ giảm tốc độ học đi dựa theo giá trị factor ở đây là 0,5 tức là giảm tốc độ học một nửa.

Chạy thử mô hình :

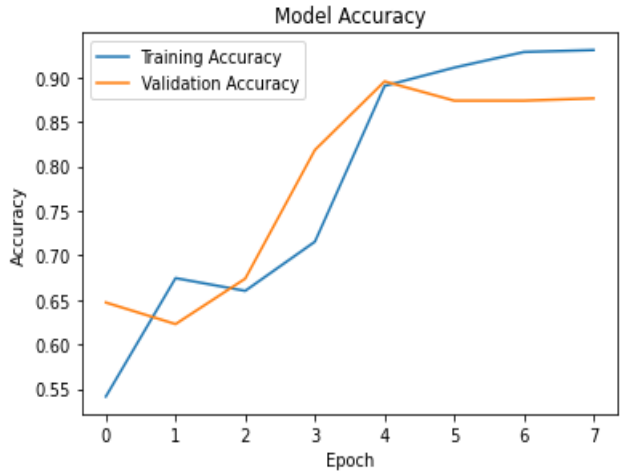
Hàm đưới dây sẽ ghi lại các kết quả bao gồm độ chính xác và mất mát trên cả tập dữ liệu huấn luyện và tập dữ liệu kiểm tra, quá trình huấn luyện sẽ chạy 20 epchos (20 vòng lặp) và có áp dụng 2 hàm callback.

```
history = model.fit(train_sequences, train_Y,
                    validation_data=(test_sequences, test_Y),
                    epochs=20,
                    batch_size=32,
                    callbacks = [lr, es]
                   )
```

Sau đó sẽ in ra kết quả cuối cùng khi kiểm tra trên tập dữ liệu kiểm tra lần cuối.

Đối với tập dữ liệu tiếng việt :

```
Epoch 8/20
145/145 [==============================] - 6s 38ms/step - loss: 0.2355 - accuracy: 0.9305
37/37 [==============================] - 0s 8ms/step - loss: 0.3304 - accuracy: 0.8953
Test Loss : 0.3303581774234772
Test Accuracy : 0.89532870054245
```

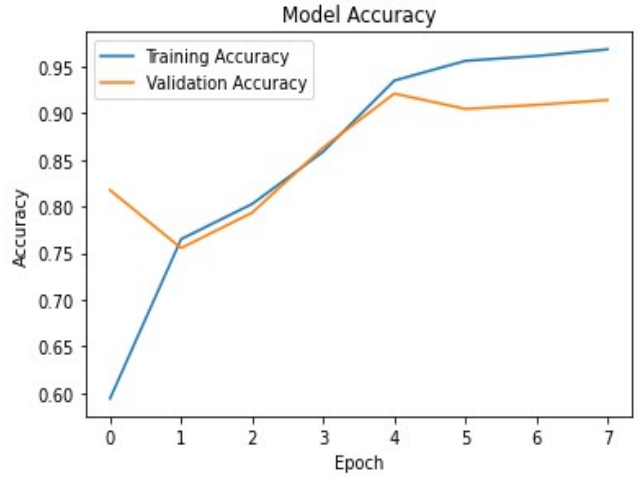


Kết quả cuối cùng trên tập huấn luyện có độ mất mát là 0,2355 và độ chính xác 0.9305.

Kết quả tốt nhất trên tập kiểm tra có độ mất mát 0.3304 và độ chính xác là 0,8953.

Đối với tập dữ liệu tiếng anh.

```
Đánh giá
37/37 [==============================] - 1s 13ms/step - loss: 0.2733 - accuracy: 0.9208
Test Loss : 0.27328068017959595
Test Accuracy : 0.9207579493522644
```



Kết quả cuối cùng trên tập huấn luyện có độ mất mát là 0,272 và độ chính xác 0,9208.

Kết quả tốt nhất trên tập kiểm tra có độ mất mát 0.2355 và độ chính xác là 0.92075.

2.4.3. So sánh với các mô hình phổ biến khác

4 Mô hình học máy KNN, Logistic Regression, SVM, Naïve Bayes sử dụng chung phương pháp TF-idf (Term Frequency – Inverse Document Frequency) để chuyển đổi dữ liệu trước khi đưa vào mô hình.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=3)
```
✓ 0.0s Python

```
feature_extraction = TfidfVectorizer(min_df = 1, stop_words='english', lowercase=True)
X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)
Y_train = Y_train.astype('str')
Y_test = Y_test.astype('str')
```
✓ 1.5s Python

Kết quả huấn luyện các mô hình phổ biến khác:

KNN (K – Nearest Neighbor):

```
model2 = KNeighborsClassifier()
model2.fit(X_train_features, Y_train)
prediction_on_training_data_knn = model2.predict(X_train_features)
accuracy_on_training_data_knn = accuracy_score(Y_train, prediction_on_training_data_knn)
prediction_on_test_data_knn = model2.predict(X_test_features)
accuracy_on_test_data_knn = accuracy_score(Y_test, prediction_on_test_data_knn)
```
✓ 5.8s Python

```
print('Accuracy on training data_knn : ', accuracy_on_training_data_knn)
print('Accuracy on test data_knn : ', accuracy_on_test_data_knn)
```
✓ 0.0s Python

```
Accuracy on training data_knn :  0.8502961175696425
Accuracy on test data_knn :  0.849561403508772
```

Logistic Regression:

```
model1 = LogisticRegression()
model1.fit(X_train_features, Y_train)
prediction_on_training_data_LgR = model1.predict(X_train_features)
accuracy_on_training_data_LgR = accuracy_score(Y_train, prediction_on_training_data_LgR)
prediction_on_test_data_LgR = model1.predict(X_test_features)
accuracy_on_test_data_LgR = accuracy_score(Y_test, prediction_on_test_data_LgR)
```
✓ 0.8s Python

```
print('Accuracy on training data_LgR : ', accuracy_on_training_data_LgR)
print('Accuracy on test data_LgR : ', accuracy_on_test_data_LgR)
```
✓ 0.0s Python

```
Accuracy on training data_LgR :  0.9644658916429042
Accuracy on test data_LgR :  0.9508771929824561
```

Naïve bayes:

```
model3 = MultinomialNB()
model3.fit(X_train_features, Y_train)
prediction_on_training_data_nb = model3.predict(X_train_features)
accuracy_on_training_data_nb = accuracy_score(Y_train, prediction_on_training_data_nb)
prediction_on_test_data_nb = model3.predict(X_test_features)
accuracy_on_test_data_nb = accuracy_score(Y_test, prediction_on_test_data_nb)
```

✓ 0.1s Python

```
print('Accuracy on training data_nb : ', accuracy_on_training_data_nb)
print('Accuracy on test data_nb : ', accuracy_on_test_data_nb)
```

✓ 0.0s Python

```
Accuracy on training data_nb :  0.9520728229874973
Accuracy on test data_nb :  0.9236842105263158
```

SVM (Support Vector Machine):

```
model4 = svm.SVC()
model4.fit(X_train_features, Y_train)
prediction_on_training_data_svm = model4.predict(X_train_features)
accuracy_on_training_data_svm = accuracy_score(Y_train, prediction_on_training_data_svm)
prediction_on_test_data_svm = model4.predict(X_test_features)
accuracy_on_test_data_svm = accuracy_score(Y_test, prediction_on_test_data_svm)
```

✓ 41.5s Python

```
print('Accuracy on training data_svm : ', accuracy_on_training_data_svm)
print('Accuracy on test data_svm : ', accuracy_on_test_data_svm)
```

✓ 0.0s Python

```
Accuracy on training data_svm :  0.9992322877824085
Accuracy on test data_svm :  0.9679824561403508
```

2.4.4. Flask

Flask là 1 framework lập trình ứng dụng web viết bằng ngôn ngữ lập trình Python, cụ thể trong bài được sử dụng để xây dựng một giao diện web, xử lý các yêu cầu HTTP từ người dùng. Trong đó Flask trong ứng dụng trên hỗ trợ:

Giao diện người dùng: Flask giúp tạo một giao diện người dùng (UI) đơn giản, được hiển thị qua trang HTML ('index.html'). Người dùng thông qua trình duyệt web để nhập nội dung email, sử dụng ngôn ngữ và xem kết quả

```
@app.route( rule: '/', methods=['GET', 'POST'])
def index():
    return render_template('index.html')
```

Xử lý yêu cầu HTTP: Flask cho phép xác định các đường dẫn và xử lý các loại yêu cầu HTTP khác nhau (ví dụ: GET và POST). Đường dẫn gốc ('/') dùng để hiển thị trang nhập dữ liệu, '/process_mail' dùng để xử lý và phân loại email sau khi đã nhập nội dung.

Trả kết quả cho người dùng: Flask trả về kết quả phân loại dưới dạng phản hồi JSON sau khi email đã được phân loại bởi mô hình. Kết quả "Mail spam" hoặc "Mail ham" được hiển thị cho người dùng thông qua giao diện web.

```python
@app.route( rule: '/process_mail', methods=['POST'])
def process_mail():
    result = None
    text = request.form.get('text')
    language = request.form.get('language')
    result = process_text(text, language)

    return jsonify({'result': result})
```

2.4.5. Áp dụng thực tế.

Giải quyết nhiều mục tiêu, ví dụ như:

Bảo vệ hòm thư điện tử của người dùng: giúp người dùng tránh việc mất thời gian và sức lực khi xóa thư rác bằng tay. Từ đó sẽ tránh việc xóa nhầm thư hàng ngày sử dụng

An toàn và bảo mật: ngăn chặn các cuộc tấn công email độc hại, như email chứa mã độc hại hoặc các email lừa đảo, từ đó bảo vệ dữ liệu và thông tin cá nhân của người dùng.

Tối ưu hóa tài nguyên: Thư rác mất nhiều dung lượng, mà tốn dung lượng lại mất cả chi phí

Ứng dụng trong doanh nghiệp: Doanh nghiệp phân loại thư rác để bảo vệ thư trong tổ chức, đảm bảo rằng thư điện tử liên quan đến công việc được gửi và nhận, còn thư rác được loại trừ, tách biệt

Phát triển sản phẩm và dịch vụ: Tích hợp vào sản phẩm và dịch vụ. Ví dụ, một ứng dụng di động có thể sử dụng mô hình để lọc thư rác cho người dùng di động.

Nghiên cứu và phát triển: nâng cao hiểu biết về cách hoạt động của các mô hình học máy và mạng nơ-ron, từ đó cải thiện chất lượng và hiệu suất

Phục vụ cộng đồng trực tuyến: được sử dụng bởi các dự án mã nguồn mở hoặc tổ chức phi lợi nhuận để giúp lọc thư rác cho cộng đồng trực tuyến một cách miễn phí và làm cho internet trở nên an toàn hơn.

3. Kết luận

Bài báo cáo chủ đề học máy này nội dung chính đề cập đến việc áp dụng xây dựng mạng Nơ – ron tuần tự kết hợp cùng các phương pháp xử lý dữ liệu và tối ưu mô hình để xử lý bài toán phân loại thư rác mở rộng với ngôn ngữ phức tạp như tiếng việt.

Kết quả cho thấy sau khi trải qua các bước tiền xử lý, mô hình nhận dữ liệu và trả lại kết quả sau huấn luyện ở mức độ đáng tin cậy với nhiệm vụ phân loại thư rác, kể cả nội dung thư là Tiếng Việt.

Mô hình học máy xây dựng bằng các lớp mạng nơ-ron tuần tự cũng mang lại kết quả hiệu quả hơn KNN, tương đương Naïve Bayes, và tiệm cận với Logistic Regression và Support Vector Machine.

4. Tài liệu liên quan

Practical Statistics for Data Scientists 50 Essential Concepts Using Rand Python by Peter Bruce Andrew Bruce Peter Gedeck
https://drive.google.com/file/d/1MleA9Cxb1eoAMwRn_RWaUxhKcQ7DI4Ih/view?usp=sharing

Tìm hiểu về Tensorflow
https://web.stanford.edu/class/cs20si/syllabus.html?fbclid=IwAR3QYjLeBryI8q9zcdsJzNE-ZLFhyUulJOoL0l_CNb07nqBWWhuyBqvQUrc