

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**



**ĐỒ ÁN
MÔN KHAI KHOÁNG DỮ LIỆU**

**Đề tài
XÂY DỰNG ỨNG DỤNG
PHÂN LOẠI VĂN BẢN CỦA 10 LĨNH VỰC**

Nhóm sinh viên thực hiện:

Nguyễn Tấn Pil	Nguyễn Thanh Tâm
Mã số: B1812295	Mã số: B1812301

Cần Thơ, 12/2021

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**



**ĐỒ ÁN
MÔN KHAI KHOÁNG DỮ LIỆU**

**Đề tài
XÂY DỰNG ỨNG DỤNG
PHÂN LOẠI VĂN BẢN 10 LĨNH VỰC**

**Giáo viên hướng dẫn
TS. Lưu Tiến Đạo**

**Nhóm sinh viên thực hiện
Nguyễn Tấn Pil – B181229
Nguyễn Thanh Tâm – B1812301**

Cần Thơ, 12/2021

NHẬN XÉT CỦA GIẢNG VIÊN

Cần Thơ, ngày 28 tháng 12 năm
(GVHD ký và ghi rõ họ tên)

LỜI CẢM ƠN

Để hoàn thành bài đồ án này, nhóm em xin được bày tỏ lòng biết ơn chân thành và sâu sắc đến Thầy Lưu Tiến Đạo – người đã trực tiếp tận tình hướng dẫn, giúp đỡ em trong suốt quá trình thực hiện đồ án, nhờ những sự chỉ bảo và hướng dẫn quý giá đó mà bài đồ án này được hoàn thành một cách tốt nhất.

Nhóm em cũng xin gửi lời cảm ơn chân thành đến các Thầy Cô Giảng viên Đại học Cần Thơ, đặc biệt là các Thầy Cô ở Khoa CNTT & TT, những người đã truyền đạt những kiến thức quý báu trong thời gian qua.

Nhóm em cũng xin chân thành cảm ơn bạn bè cùng với gia đình đã luôn động viên, khích lệ và tạo điều kiện giúp đỡ trong suốt quá trình thực hiện để chúng em có thể hoàn thành bài đồ án một cách tốt nhất.

Trong quá trình thực hiện đồ án không thể tránh khỏi những sai sót. Nhóm chúng em rất mong nhận được sự đóng góp ý kiến quý báu của quý Thầy và các bạn để bài đồ án hoàn thiện hơn.

Cần Thơ, ngày 28 tháng 12 năm 2021

Người viết

Nguyễn Tấn Pil Nguyễn Thanh Tâm

MỤC LỤC

DANH MỤC HÌNH	4
TÓM TẮT	8
PHẦN GIỚI THIỆU	10
1. Đặt vấn đề	10
2. Lịch sử giải quyết vấn đề	11
3. Mục tiêu đề tài.....	11
4. Đối tượng và phạm vi nghiên cứu.....	12
5. Kết quả đạt được	12
6. Bố cục đồ án.....	12
PHẦN NỘI DUNG	13
CHƯƠNG 1 - MÔ TẢ BÀI TOÁN	13
1. Mô tả chi tiết bài toán.....	13
2. Vấn đề liên quan đến bài toán	15
2.1 Tách từ.....	15
2.2 Vector hóa(TF-IDF)	16
CHƯƠNG 2 - THIẾT KẾ VÀ CÀI ĐẶT	18
1. Cơ sở lý thuyết	18
1.1. Các giải thuật.....	18
1.1.1 Giải thuật Naive Bayes	18
1.1.2 Giải thuật SVM (Support Vector Machine).....	19
1.1.3 Giải thuật Decision Tree	21

1.1.4 Giải thuật Logistic Regression	23
1.1.5 Mô hình PhoBERT	24
1.2 HTML và CSS	25
1.3 JavaScript	25
1.4 ReactJS	26
2. Thiết kế và cài đặt	26
2.1 Chuẩn bị dữ liệu	26
2.2 Khám phá dữ liệu	27
2.2.1 Mô tả tập dữ liệu	27
2.2.2 Xử lý dữ liệu bị thiếu	29
2.2.3 Mô hình hóa dữ liệu	30
2.3 Tiền xử lý dữ liệu	31
2.3.1 Chuyển đổi dữ liệu về dạng chữ thường	32
2.3.2 Loại bỏ các ký tự số	32
2.3.3 Loại các ký tự đơn, bắt đầu bằng ký tự đơn và khoảng trắng	33
2.3.4 Chuẩn hóa về Unicode dựng sẵn	33
2.3.5 Chuẩn hóa kiểu gõ dấu tiếng Việt	34
2.3.6 Loại bỏ những ký tự lặp	37
2.3.7 Loại bỏ các từ dừng	37
2.3.8 Tách từ tiếng Việt	38
2.3.9 Thực hiện làm sạch dữ liệu	39
2.3.10 Vector hóa văn bản	41

2.3.11 Xáo trộn dữ liệu	42
2.4 Xây dựng mô hình phân loại văn bản	43
2.4.1 Xây dựng mô hình phân loại văn bản với SVM	43
2.4.2 Xây dựng mô hình phân loại văn bản với Naive Bayes.....	44
2.4.3 Xây dựng mô hình phân loại văn bản với Decision Tree	46
2.4.4 Xây dựng mô hình phân loại văn bản với Logistic Regression	48
2.4.5 Xây dựng mô hình phân loại văn bản với PhoBERT	50
2.5 Xây dựng website dự đoán.....	58
CHƯƠNG 3 - KIỂM THỬ VÀ ĐÁNH GIÁ.....	62
1. Mục tiêu	62
2. Các thông số đánh giá giải thuật	62
3. Đánh giá	64
PHẦN KẾT LUẬN	66
1. Kết quả đạt được	66
2. Hạn chế và hướng phát triển	66
2.1 Hạn chế.....	66
2.2 Hướng phát triển	66
TÀI LIỆU THAM KHẢO.....	68

DANH MỤC HÌNH

Hình 1: Siêu phẳng lẻ cực đại trong không gian hai chiều	20
Hình 2 Mô hình cây quyết định[7]	22
Hình 3 Tạo tập tin csv từ thư mục Train_Full và Test_Full	27
Hình 4: Tập dữ liệu huấn luyện.....	28
Hình 5: Tập dữ liệu kiểm tra	29
Hình 6: Kiểm tra dữ liệu bị thiếu	30
Hình 7: Mô hình hóa số lượng mẫu tin của từng phân lớp trong tập huấn luyện	30
Hình 8: Mô hình hóa số lượng mẫu tin của từng phân lớp trong tập kiểm tra.....	31
Hình 9: Hàm chuyển đổi dữ liệu về dạng chữ thường	32
Hình 10: Hàm loại bỏ các ký tự số.....	32
Hình 11: Hàm loại các ký tự đơn, bắt đầu bằng ký tự đơn và khoảng trắng	33
Hình 12: Hàm chuẩn hóa về Unicode dựng sẵn.....	34
Hình 13: Hàm kiểm tra từ tiếng Việt hợp lệ	35
Hình 14: Hàm chuẩn hóa từ gõ dấu tiếng Việt.....	36
Hình 15: Hàm chuẩn hóa câu gõ dấu tiếng Việt	37
Hình 16: Hàm loại bỏ ký tự lặp.....	37
Hình 17: Danh sách từ dừng	38
Hình 18: Hàm lấy danh sách từ dừng.....	38
Hình 19: Hàm tách từ với pyvi.....	39
Hình 20: Hàm tách từ với vncoreNLP	39
Hình 21: Hàm làm sạch dữ liệu.....	40
Hình 22: Làm sạch dữ liệu huấn luyện và kiểm tra	40
Hình 23: Tách từ dữ liệu huấn luyện và kiểm tra bằng pyvi	41
Hình 24: Tách từ dữ liệu huấn luyện và kiểm tra bằng vncoreNLP	41

Hình 25: Dữ liệu sau quá trình làm sạch.....	41
Hình 26: Vector hóa dữ liệu văn bản	42
Hình 27: Lưu lại mô hình tf-idf.....	42
Hình 28: Xáo trộn dữ liệu	42
Hình 29: Mô hình SVM	43
Hình 30: Đánh giá trên từng phân lớp - SVM	44
Hình 31: Mô hình Naive Bayes.....	45
Hình 32: Đánh giá trên từng phân lớp - Naive Bayes.....	46
Hình 33: Mô hình cây quyết định	47
Hình 34: Đánh giá trên từng phân lớp - Decision Tree.....	48
Hình 35: Mô hình Logistic Regression	49
Hình 36: Đánh giá trên từng phân lớp - Logistic Regression	49
Hình 37: Cài đặt transformers	50
Hình 38: Cài đặt fastBPE và fairseq	50
Hình 39: Cài đặt mô hình PhoBERT	51
Hình 40: Tải mô hình bpe và từ điển	51
Hình 41: Encode nhãn	52
Hình 42: Ánh xạ subword	52
Hình 43: Kiểm tra padding của mẫu tin	53
Hình 44: Tạo DataLoader.....	53
Hình 45: Tải mô hình PhoBERT.....	54
Hình 46: Hàm đánh giá	54
Hình 47: Hàm huấn luyện	55
Hình 48: Huấn luyện mô hình PhoBERT	56
Hình 49: Kết quả sau khi huấn luyện trên 10 epochs.....	56
Hình 50: Dự đoán và đánh giá mô hình PhoBERT.....	57

Hình 51: Xóa cache cuda	57
Hình 52 Quá trình dự nhãn của website.....	58
Hình 53: Lấy dữ liệu từ client và tiền xử lý	59
Hình 54: Tải mô hình, dự đoán và trả về kết quả.....	60
Hình 55: Giao diện website dự đoán.....	61

DANH MỤC BẢNG

Bảng 1: Mô tả tập dữ liệu.....15

Bảng 2: Chỉ số đánh giá giải thuật học63

Bảng 3: Thông tin đánh giá từng giải thuật học.....65

TÓM TẮT

Bài toán phân loại (phân lớp) văn bản tự động hay **Text Classification** thuộc về lĩnh vực **xử lý ngôn ngữ tự nhiên** ở dạng văn bản, là việc phân chia một tập văn bản đầu vào thành hai hoặc nhiều lớp, trong đó mỗi lớp văn bản có thể thuộc một hoặc nhiều lớp. Công việc này nhằm mục đích gán nhãn (lớp) được định nghĩa trước cho các văn bản mới đến.

Đối với từng ngôn ngữ khác nhau bài toán có cách giải quyết khác nhau. Đề tài **“Xây dựng website phân loại văn bản của 10 lĩnh vực”** sẽ được giải quyết bằng bài toán phân loại văn bản tiếng Việt. Do từ tiếng Việt đa số là từ đơn âm tiết và không biến đổi hình thái nên việc xác định ý nghĩa phụ thuộc vào nhiều yếu tố khác nhau như ngữ cảnh và trật tự từ trong câu. Quy trình thực hiện việc phân loại văn bản được thực hiện như sau. Đầu tiên là bước tiền xử lý văn bản, đây là bước rất quan trọng, nó ảnh hưởng đến kết quả của đề tài, trong bước này ta thực hiện việc làm sạch dữ liệu, tách từ, chuẩn hóa từ và loại bỏ stopwords. Trong đó, tách từ là bước cực kỳ quan trọng trong việc xử lý ngôn ngữ tự nhiên nói chung và phân loại văn bản nói riêng, Đối với phân loại văn bản tiếng Việt thì khó khăn hơn phân loại văn bản tiếng Anh bởi vì nó có thêm các từ ghép tồn tại trong câu, chúng tôi sử dụng công cụ VnTokenizer để thực hiện tách từ. Dữ liệu trước khi được đưa vào mô hình chúng ta cần thực hiện transform dữ liệu thành các vector (vector hóa) bằng phương pháp TF-IDF để trích xuất các đặc trưng sau đó đưa vào huấn luyện. Bước tiếp theo là sử dụng mô hình để huấn luyện và dự đoán. Chúng tôi sử dụng các mô hình Decision Tree, Naive Bayes, SVM, Logistic Regression và mô hình pretraining PhoBERT huấn luyện với **33.759** mẫu tin.

Kết quả trong việc phân loại 10 lĩnh vực với **30.759** mẫu train và hơn **50.373** mẫu test cho thấy hiệu quả phân lớp của giải thuật học SVM, Logistic Regression và PhoBERT cho kết quả cao hơn các giải thuật Naive Bayes, Decision Tree,, điều đó

có nghĩa là các mô hình SVM, Logistic Regression và PhoBERT có khả năng học tốt hơn các giải thuật còn lại với tập dữ liệu đã cho. Về thời gian thì giải thuật Naive Bayes có thời gian train và predict nhanh hơn các mô hình còn lại.

PHẦN GIỚI THIỆU

1. Đặt vấn đề

Hiện nay với sự bùng nổ của Công Nghệ Thông Tin, mọi thứ đều được đưa lên Internet, do đó số lượng tài liệu trên các nền tảng website là rất lớn như các website, tạp chí khoa học, tài liệu học tập, sách điện tử, tạp chí, tin tức và nhiều thứ khác. Ngày qua ngày lượng thông tin ngày càng đa dạng và phong phú về nội dung, với lượng thông tin lớn như vậy thì yêu cầu đặt ra là làm sao tổ chức, lưu trữ và tìm kiếm thông tin một cách nhanh chóng, hiệu quả. Ngày nay có một phương pháp máy học phân loại văn bản có thể giải quyết được các yêu cầu nêu trên.

Các hệ thống phân loại văn bản có thể ứng dụng trong phân loại thư điện tử, phân loại nội dung tin tức, phân loại bình luận. Trong đề tài này chúng tôi sẽ xây dựng hệ thống phân loại tin tức với 10 lĩnh vực khác nhau.

Các lĩnh vực trong máy học được ứng dụng rộng rãi trong hầu hết các lĩnh vực, trong đó phân loại văn bản cũng đang được quan tâm. Xây dựng mô hình phân loại văn bản tiếng Việt phải thông qua nhiều bước xử lý mới thu được mô hình phù hợp với độ chính xác mong đợi. Để thu được mô hình mong đợi dữ liệu đầu vào cần phải thông qua các bước tiền xử lý như làm sạch dữ liệu, tách từ, chuẩn hóa và loại bỏ các từ dừng trong tiếng Việt. Sau đó lựa chọn mô hình phù hợp, tinh chỉnh các chỉ số của mô hình và đánh giá, kiểm thử để điều chỉnh nếu cần thiết.

Các mô hình phân loại văn bản hiện nay sử dụng mô hình không gian vector để biểu diễn, các máy tính không thể hiểu được các từ ngữ của ngôn ngữ tự nhiên nên cần phải dùng các phương pháp chuyển đổi các từ thành dữ liệu mà máy tính có thể hiểu và thực thi được. Khi xử lý ngôn ngữ tự nhiên với các ngôn ngữ tự nhiên, mỗi ngôn ngữ có những đặc điểm khác nhau, cái riêng của từng ngôn ngữ. Đối với tiếng Việt thì sử dụng công cụ VnTokenizer để tách các từ trong câu, đây là một công cụ rất hiệu quả. Quá trình tách từ giúp tách các từ có nghĩa với nhau, tức là tách các từ

đơn và các từ ghép trong câu, điều này thực sự quan trọng, nó giúp cho máy tính có thể hiểu được các từ trong câu.

Sau khi dữ liệu đã được tiền xử lý thì tiếp đến là lựa chọn mô hình để huấn luyện trên tập dữ liệu đã tiền xử lý. Sử dụng các mô hình phân lớp như SVM, Decision Tree, Naive Bayes và mô hình pretraining PhoBERT để tiến hành phân lớp.

2. Lịch sử giải quyết vấn đề

Trong máy học, phân loại văn bản đã có nhiều nghiên cứu, cải tiến và đã đạt được nhiều thành công mong đợi. Các giải thuật ngày càng chạy nhanh hơn và độ chính xác cũng được cải thiện. Phân loại văn bản tiếng Anh dễ dàng hơn phân loại văn bản tiếng Việt, do đặc điểm ngôn ngữ. Tiếng Việt bị hạn chế ở bước tách từ, có thể các công cụ tách từ còn chưa chính xác dẫn đến kết quả của quá trình phân lớp bị ảnh hưởng khá nhiều.

Có một số công trình đã và đang nghiên cứu về các phương pháp tách từ, cũng như là mô hình huấn luyện. Ví dụ như mô hình PhoBERT của VinAI chỉ dành riêng cho tiếng Việt giúp tăng đáng kể hiệu quả của quá trình phân lớp.

3. Mục tiêu đề tài

Tìm hiểu về quá trình phân lớp dữ liệu nói chung và phân lớp dữ liệu với tiếng Việt nói riêng. Tìm hiểu quá trình tiền làm sạch dữ liệu, tách từ, chuẩn hóa từ và các mô hình cho bài toán phân lớp. Đặt biệt tìm hiểu cách sử dụng mô hình pretraining (PhoBERT). Nâng cao khả năng tìm kiếm tài liệu, tự học và khả năng làm việc nhóm một cách hiệu quả hơn. Xây dựng một trang web để thực hiện quá trình dự đoán lĩnh vực (nhân) của các bài báo.

4. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu là các phương pháp tách từ, chuẩn hóa từ và các bước tiền xử lý dữ liệu. Các mô hình máy học phù hợp với quá trình phân lớp văn bản, các phương pháp và chỉ số đánh giá hiệu quả giải thuật học.

Phạm vi nghiên cứu là các kiến thức của Nguyên Lý Máy Học, Khai Khoáng Dữ Liệu, Lập Trình Web và các kiến thức hỗ trợ liên quan đến đề tài.

5. Kết quả đạt được

Hiểu được các bước trong quá trình phân loại văn bản tiếng Việt, học được cách sử dụng các mô hình máy học như SVM, Naive Bayes, PhoBERT ..., xây dựng một trang Web với Machine Learning và đưa trang Web lên production. Học được cách phân công và làm việc nhóm một cách hiệu quả.

6. Bố cục đồ án

Phần giới thiệu

Giới thiệu tổng quát về đề tài.

Phần nội dung

Chương 1 : Mô tả bài toán.

Chương 2 : Thiết kế và cài đặt giải thuật.

Chương 3 : Kiểm thử và đánh giá hệ thống.

Phần kết luận

Trình bày kết quả đạt được và hướng phát triển hệ thống.

PHẦN NỘI DUNG

CHƯƠNG 1 - MÔ TẢ BÀI TOÁN

1. Mô tả chi tiết bài toán

Phân lớp hay phân loại văn bản là một bài toán cổ điển và cơ bản trong khai phá dữ liệu văn bản, bài toán phân lớp thực hiện quá trình gán nhãn cho các phần tử mới đến từ những kinh nghiệm đã học được từ mô hình đã được đào tạo trước đó.

Đề tài phân lớp văn bản cụ thể là phân lớp các bài báo, tin tức có 10 lĩnh vực khác nhau như Chính Trị Xa Hoi, Doi Song, Vi Tinh ..., những mẫu tin đã được gán nhãn chính xác với nội dung mà mẫu tin miêu tả. Chúng tôi thực hiện việc train mô hình và sau đó xây dựng website để dự đoán các tin tức, bài báo mới chưa có nhãn để dự đoán. Với đề tài này chúng tôi sử dụng các giải thuật cơ bản của bài toán phân lớp như SVM, Naive Bayes, Decision Tree, Logistic Regression và mô hình PhoBERT để thực hiện huấn luyện với hơn 30 nghìn mẫu tin.

Chúng tôi giải quyết bài toán phân lớp theo các bước sau:

- **Khám phá dữ liệu:** sử dụng biểu đồ để xem tỉ lệ các nhãn trong tập dữ liệu có cân bằng hay không.
- **Tiền xử lý dữ liệu:** để cho mô hình máy học có thể học và đạt được hiệu quả tốt nhất chúng ta cần làm sạch dữ liệu, tách từ, chuẩn hóa từ, loại bỏ những từ dừng có trong tập từ dừng có sẵn và xáo trộn dữ liệu. Với mô hình PhoBERT chúng ta cần phải chuyển tất cả các nhãn thành số (mã hóa nhãn).

- **Huấn luyện mô hình:** sử dụng các giải thuật học như SVM, Naive Bayes, Decision Tree, Logistic Regression và PhoBERT để thực hiện quá trình huấn luyện trên tập dữ liệu học.
- **Dự đoán và đánh giá mô hình:** từ những mô hình đã được huấn luyện, ta thực hiện dự đoán trên tập dữ liệu kiểm tra và sử dụng các chỉ số đánh giá để đánh giá hiệu quả của các mô hình.

Mô tả tập dữ liệu train và test:

STT	TÊN LỚP (LĨNH VỰC)	SỐ LƯỢNG MẪU TIN TRAIN	SỐ LƯỢNG MẪU TIN TEST	SỐ LƯỢNG MẪU TIN TỪNG PHÂN LỚP
1	The Thao	5298	6667	11965
2	Chinh Tri Xa Hoi	5219	7567	12786
3	Phap Luat	3868	3788	7656
4	Suc Khoe	3384	5147	8531
5	Doi Song	3159	2036	5195
6	Van Hoa	3080	6250	9330
7	The Gioi	2898	6716	9614
8	Kinh Doanh	2552	5276	7828
9	Vi Tinh	2481	4560	7041
10	Khoa Hoc	1820	2096	3376

TỔNG	30.759	50.373	81.132
-------------	---------------	---------------	---------------

Bảng 1: Mô tả tập dữ liệu

Tập dữ liệu train và test có 10 nhãn tương ứng với 10 lĩnh vực và tổng cộng có **81.132** mẫu tin, trong đó có **30.759** mẫu train và **50.373** mẫu test. Với tập dữ liệu lớn như vậy, thời gian huấn luyện và dự đoán tốn khá nhiều thời gian.

2. Vấn đề liên quan đến bài toán

2.1 Tách từ

Tách từ hay còn gọi là **Tokenization** là một bước quan trọng trong việc tiền xử lý văn bản, đây là một bước không thể bỏ qua cho các bài toán phân loại văn bản hiện nay. Tách từ được hiểu là phân chia (phân tách) một câu, một đoạn văn thành các đơn vị nhỏ hơn như từ hoặc ký tự. Các đơn vị này được gọi là các tokens, sau đó sử dụng để dựng nên từ điển, các tokens này sẽ tương ứng với một ID riêng biệt.

Trong xử lý ngôn ngữ tự nhiên có một số kiểu tách từ như sau[5]:

- **Tách từ theo word-level:** Các câu được phân tách thành các tokens được ngăn cách bởi khoảng trắng hoặc dấu câu. Khi đó mỗi token là một từ đơn âm tiết. Đây là phương pháp được sử dụng trong các thuật toán truyền thống như GloVe, word2vec.
- **Tách từ theo multi-word-level:** Không phải ngôn ngữ nào cũng chỉ có đơn âm tiết mà có những ngôn ngữ tồn tại cả đơn âm tiết (từ đơn) và đa âm tiết (từ ghép), tiếng Việt là một trường hợp. Do vậy các câu được tách thành các từ đơn âm tiết sẽ làm nghĩa của câu bị sai đi. Ví dụ cụm từ **không xác định** nếu được chia thành **không**, **xác** và **định** sẽ làm mất đi ý nghĩa phủ định vốn có của cụm từ, vì thế chúng ta cần sử dụng thêm các từ điển hỗ trợ đơn âm tiết và đa âm tiết để các câu được phân tách đúng ý nghĩa của chúng. Trong tiếng Việt có các module hỗ trợ nổi bật như VnCoreNLP, pyvivn.

- **Tách từ theo character-level:** Việc tách từ theo **word-level** thường sinh ra một từ điển có kích thước rất lớn, điều này làm tăng chi phí tính toán. Hơn nữa, nếu tách từ theo **word level** thì cần có từ điển rất lớn thì mới hạn chế được những từ nằm ngoài từ điển trước đó. Tuy nhiên, nếu phân tích các từ ta sẽ thấy hầu hết các từ được hình thành từ các nhóm ký tự như chữ cái, số và dấu câu.. Như vậy, chỉ cần sử dụng không gian từ của ngôn ngữ thì có thể bao quát được tất cả các từ của ngôn ngữ. Quá trình tách từ dựa trên level ký tự làm giảm kích thước từ điển mà có thể biểu diễn các từ nằm ngoài từ điển. Đây là phương pháp được áp dụng trong mô hình fasttext.
- **Phương pháp BPE (SOTA):** Tách từ theo **character-level** có nhược điểm đó là các tokens không có ý nghĩa khi chúng đứng đơn lẻ. Do đó, với các bài toán phân tích cảm xúc nếu áp dụng tách từ theo **character-level** sẽ không mang lại hiệu quả. Còn tách từ theo **word-level** cũng có một số hạn chế đó là không đạt được hiệu quả với các từ không nằm trong từ điển trước đó. Là phương pháp có khả năng tách từ theo level nhỏ hơn từ và lớn hơn ký tự được gọi là subword. Khi áp dụng tách từ theo phương pháp mới này đã cải thiện được độ chính xác trên nhiều tác vụ dịch máy, phân loại văn bản, dự báo câu tiếp theo, hỏi đáp, dự báo mối quan hệ văn bản.

2.2 Vector hóa(TF-IDF)

TF-IDF(Term Frequency – Inverse Document Frequency) là một kỹ thuật sử dụng trong quá trình khai phá dữ liệu văn bản. Trọng số này được sử dụng ước lượng tần suất xuất hiện và mức độ quan trọng của một từ trong văn bản[6].

TF(term frequency - tần suất xuất hiện của một từ) là số lần xuất hiện của một từ trong văn bản. Vì các văn bản không phải lúc nào cũng có cùng độ dài nên có thể từ đó xuất hiện nhiều hơn do nằm trong đoạn văn có độ dài lớn hơn. Do vậy, số lần xuất hiện của từ được chia cho tổng các từ của văn bản đó.

$$tf(t, d) = \frac{f(t, d)}{\max\{f(w, d) : w \in d\}}$$

Trong đó[6]:

- $tf(t, d)$: tần suất xuất hiện của từ t trong văn bản d
- $f(t, d)$: Số lần xuất hiện của từ t trong văn bản d
- $\max(\{f(w, d) : w \in d\})$: Số lần xuất hiện của từ có số lần xuất hiện nhiều nhất trong văn bản d

IDF(Inverse Document Frequency - tần suất nghịch đảo văn bản) là mức độ quan trọng của của một từ trong một văn bản. Khi tính tf thì mọi từ có mức độ quan trọng như nhau, một số từ có tần suất rất nhiều nhưng mức độ quan trọng như các từ dừng. Vì thế, ta cần giảm mức độ quan trọng của các từ này xuống.

$$idf(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Trong đó[6]:

- $idf(t, D)$: giá trị idf của từ t trong tập văn bản
- $|D|$: Tổng số văn bản trong tập D
- $|\{d \in D : t \in d\}|$: thể hiện số văn bản trong tập D có chứa từ t

Từ đó ta có: **$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$**

Những từ có giá trị **$tfidf$** cao là những từ xuất hiện nhiều trong văn bản này và xuất hiện ít trong các văn bản khác. Việc này giúp lọc ra những từ xuất hiện phổ biến và giữ lại những từ có giá trị cao để sử dụng[6].

CHƯƠNG 2 - THIẾT KẾ VÀ CÀI ĐẶT

1. Cơ sở lý thuyết

1.1. Các giải thuật

1.1.1 Giải thuật Naive Bayes

Naive Bayes là một thuật toán phổ biến trong máy học được đánh giá là một trong những phương pháp phân loại văn bản có hiệu năng cao. Phương pháp này là một giải thuật học có giám sát và xây dựng mô hình dựa trên tập dữ liệu đã được gán nhãn. Naive Bayes là một thuật toán phân lớp được mô hình hóa dựa trên định lý Bayes về lý thuyết xác suất để đưa ra các dự đoán cũng như phân loại phân tử mới đến dựa trên các thông tin đã được thống kê trong quá trình huấn luyện.

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

Trong đó:

- **$P(y|X)$** (posterior probability): xác suất của mục tiêu **y** với điều kiện có đặc trưng **X**
- **$P(X|y)$** (likelihood): xác suất của đặc trưng **X** khi đã biết mục tiêu **y**
- **$P(y)$** (prior probability): xác suất tiên nghiệm của mục tiêu **y**
- **$P(X)$** (prior probability): xác suất tiên nghiệm của đặc trưng **X**
- **X** là vector các đặc trưng, có thể viết dưới dạng:

$$X = (X_1, X_2, X_3, \dots, X_n)$$

Khi đó, đẳng thức Bayes sẽ được xây dựng:

$$P(y|x_1, x_2, \dots, x_n) = \frac{P(x_1|y)P(x_2|y) \dots P(x_n|y)P(y)}{P(x_1)P(x_2) \dots P(x_n)}$$

Trong mô hình Naïve Bayes, có hai giả thuyết được đặt ra:

- Các đặc trưng đưa vào mô hình được xem là độc lập với nhau. Tức là sự thay đổi giá trị của một đặc trưng không ảnh hưởng đến các đặc trưng còn lại.
- Các đặc trưng đưa vào mô hình có độ quan trọng ngang nhau đối với đầu ra mục tiêu.

Khí đó, kết quả mục tiêu $P(y|X)$ đạt cực đại trở thành:

$$y = \underset{y}{\operatorname{argmax}} P(y) \prod_{i=1}^n P(x_i|y)$$

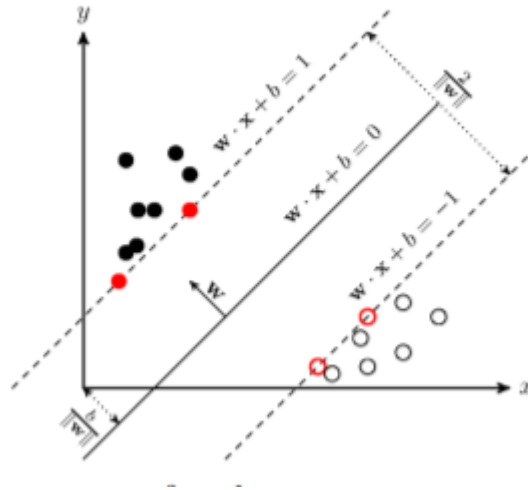
Do hai giả thuyết đã được nêu trên, mô hình này được gọi là **Naive(Thơ Ngây)**. Tuy nhiên, chính sự đơn giản của nó với việc huấn luyện cũng như dự đoán rất nhanh kết quả đầu ra khiến nó được sử dụng rất nhiều trong thực tế trên những bộ dữ liệu lớn, đem lại kết quả khá khả quan.

1.1.2 Giải thuật SVM (Support Vector Machine)

SVM (Support Vector Machine) là một thuật toán học máy có giám sát được sử dụng rất phổ biến ngày nay trong các bài toán phân lớp (classification). SVM rất hiệu quả để giải quyết các bài toán với dữ liệu có số chiều lớn như các véc-tơ biểu diễn văn bản. SVM được xem là bộ phân lớp chính xác nhất cho bài toán phân lớp văn bản do tốc độ phân lớp rất nhanh và hiệu quả đối với bài toán phân lớp văn bản[2].

Ý tưởng của phương pháp này là cho trước một tập huấn luyện được biểu diễn trong không gian véc-tơ, trong đó mỗi văn bản được xem là một điểm trong không

gian này. Phương pháp này tìm ra một mặt siêu phẳng h quyết định tốt nhất có thể chia các điểm trên không gian này thành hai lớp riêng biệt tương ứng, gọi là lớp dương (+) và lớp âm (-). Như vậy, bộ phân loại **SVM** là một mặt siêu phẳng tách các mẫu thuộc lớp dương ra khỏi các mẫu thuộc lớp âm với độ chênh lệch lớn nhất. Độ chênh lệch này hay còn gọi là khoảng cách biên được xác định bằng khoảng cách giữa mẫu (+) và mẫu (-) gần mặt siêu phẳng nhất (Hình 1). Khoảng cách này càng lớn thì các mẫu thuộc hai lớp càng được phân chia rõ ràng, nghĩa là sẽ đạt được kết quả phân loại tốt. Mục tiêu của thuật toán SVM là tìm được khoảng cách biên lớn nhất để tạo được kết quả phân loại tốt[2].



Hình 1: Siêu phẳng lề cực đại trong không gian hai chiều

Phương trình mặt siêu phẳng chứa vector x trong không gian đối tượng là:

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Trong đó,

- \mathbf{w} : là vector trọng số
- b : là độ lệch/thiên vị (bias).

Hướng và khoảng cách từ gốc tọa độ đến mặt siêu phẳng thay đổi khi thay đổi w và b .

Bộ phân lớp SVM được định nghĩa như sau:

$$f(x) = \text{sign}(w \cdot x + b)$$

Trong đó:
$$\begin{cases} f(x) = +1, w \cdot x + b \geq 0 \\ f(x) = -1, w \cdot x + b < 0 \end{cases}$$

Gọi y_i mang giá trị +1 hoặc -1. Nếu $y_i = +1$ thì x thuộc về lớp (+), ngược lại $y_i = -1$ thì x thuộc về lớp (-). Hai mặt siêu phẳng tách các mẫu thành hai phần được mô tả bởi các phương trình: $w \cdot x + b = 1$ và $w \cdot x + b = -1$. Bằng hình học có thể tính khoảng cách giữa hai mặt siêu phẳng này là: $\frac{2}{\|w\|}$

Để khoảng cách biên là lớn nhất cần phải tìm giá trị nhỏ nhất của $\|w\|$; đồng thời, ngăn chặn các điểm dữ liệu rơi vào vùng bên trong biên, cần thêm ràng buộc sau:

$$\begin{cases} w \cdot x_i + b \geq 1, \text{ với mẫu } (+) \\ w \cdot x_i + b \leq -1, \text{ với mẫu } (-) \end{cases}$$

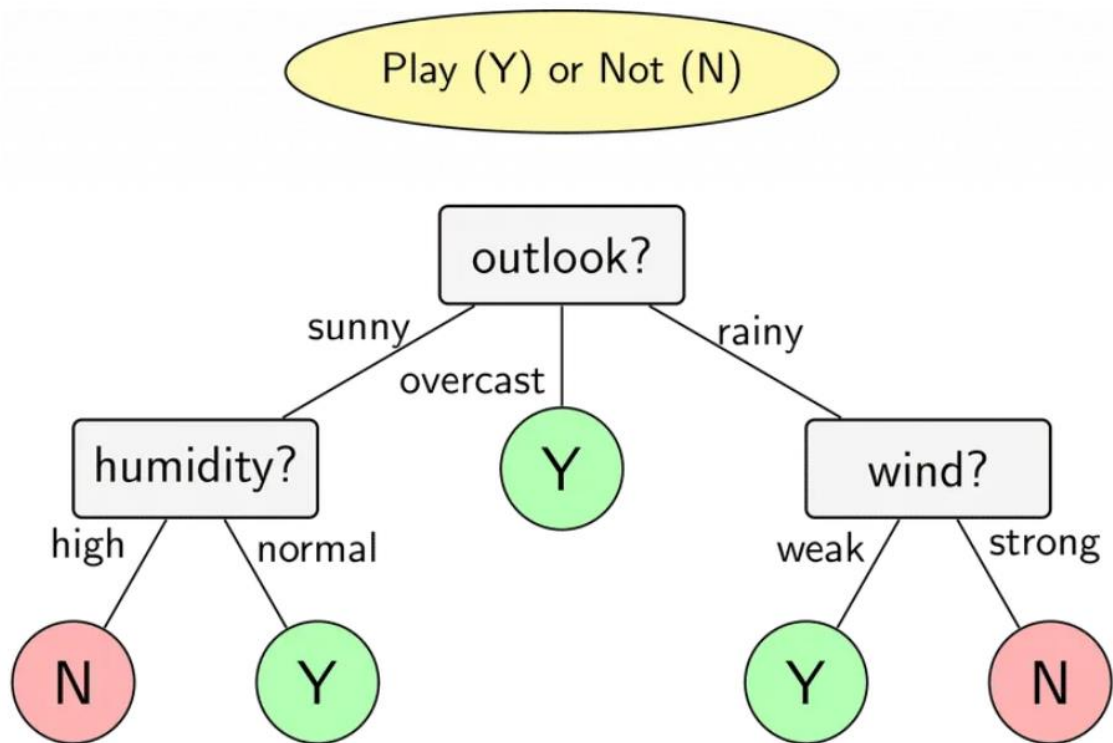
Có thể viết lại như sau: $y_i(w \cdot x_i + b) \geq 1$, với $i \in (1, n)$

Khi đó, việc tìm siêu phẳng h tương đương giải bài toán tìm $\text{Min} \|w\|$ với w và b thỏa điều kiện sau: $\forall i \in (1, n): y_i(w \cdot x_i + b) \geq 1$

1.1.3 Giải thuật Decision Tree

Cây quyết định (Decision Tree) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Các thuộc tính của đối tượng có thể thuộc các kiểu dữ liệu khác nhau như Nhị phân (Binary), Định danh (Nominal), Thứ tự (Ordinal), Số lượng (Quantitative) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal. Với dữ liệu về các đối tượng gồm các thuộc tính cùng

với lớp(classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các dữ liệu chưa biết[7].



Hình 2 Mô hình cây quyết định[7]

Từ cây quyết định ta có thể thấy, nếu trời mưa và gió mạnh thì không thể đi chơi. Ngược lại, nếu trời nắng và nhiệt độ bình thường thì có thể đi chơi.

Xây dựng cây theo từ trên xuống, bắt đầu nút gốc, nếu dữ liệu tại một nút thuần nhất thì nút đó là nút lá. Ngược lại, nếu tại một nút mà các phần tử có nhiều nhãn khác nhau (không thuần nhất) thì chọn một thuộc tính tốt nhất để tiếp tục phân hoạch cho đến khi thuần nhất, với **C4.5/ID3** chọn thuộc tính có chỉ số độ lợi thông tin lớn nhất, với **CART** chọn thuộc tính có chỉ số gini nhỏ nhất. Cuối cùng thu được cây quyết định nhỏ nhất.

Với cây quyết định được dựng sẵn thì khi một phần tử mới đến sẽ dễ dàng dự đoán được nhãn thông qua các giá trị của thuộc tính cần dự đoán.

Cây quyết định là giải thuật phổ biến bởi tính đơn giản và dễ đọc của nó. Với một dữ liệu mới đến thì chỉ cần dựa theo các nút trong để dự đoán nhãn, có thể sử dụng cho bài toán phân lớp và hồi quy. Ngoài ra, cây quyết định có thể bị overfitting, tức là đạt độ chính xác cao với tập dữ liệu huấn luyện, nhưng lại thấp với tập dữ liệu kiểm tra.

1.1.4 Giải thuật Logistic Regression

Logistic Regression là một kỹ thuật phân loại được máy học mượn từ lĩnh vực thống kê. Logistic Regression là một phương pháp thống kê để phân tích một tập dữ liệu trong đó có một hoặc nhiều biến độc lập xác định một kết quả. Mục đích đằng sau việc sử dụng Logistic Regression là tìm ra mô hình phù hợp nhất để mô tả mối quan hệ giữa biến phụ thuộc và biến độc lập[8].

Logistic Regression là một kỹ thuật phân lớp được sử dụng trong các mô hình học máy. Sử dụng một hàm Logistic để mô hình hóa biến phụ thuộc. Biến phụ thuộc có tính chất phân đôi, tức là chỉ một trong hai trường hợp xảy ra (ví dụ: email có spam hay không). Kỹ thuật này thường được sử dụng trong các bài toán phân lớp khi dữ liệu có 2 nhãn hay còn gọi là phân lớp nhị phân.

Mặc dù thường được dùng để phân loại nhị phân, nhưng có thể mở rộng thành nhiều biến thể như:

- **Nhị phân:** dữ liệu được phân thành hai nhãn, ví dụ như email có spam hay không.
- **Đa nhãn:** dữ liệu được có thể có ba nhãn hoặc nhiều hơn, ví dụ như phân loại các lĩnh vực tin tức.
- **Nguyên mẫu:** dữ liệu nhãn là các danh mục được sắp xếp, ví dụ như phân loại đánh giá từ 1 đến 5 sao.

Trong **Logistic Regression** để ánh xạ các giá trị dự đoán với xác suất, hàm sigmoid được sử dụng. Hàm này ánh xạ bất kỳ giá trị thực nào thành một giá trị khác trong khoảng từ 0 đến 1. Hàm này có đạo hàm không âm tại mỗi điểm và chính xác một điểm uốn cong[8].

Hàm chi phí là một hàm được sử dụng để tính toán khoảng sai số giữa giá trị thực tế và giá trị dự đoán. Nói một cách đơn giản, hàm chi phí là một phép đo mức độ sai khác của mô hình về khả năng ước tính mối quan hệ giữa giá trị thực và giá trị dự đoán. Giá trị được trả về từ hàm chi phí được gọi là chi phí hoặc tổn thất hoặc đơn giản là lỗi. Đối với Logistic Regression giá trị lỗi càng thấp thì mô hình càng tốt hay càng chính xác, hàm chi phí được cho bởi phương trình:

$$-\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

Hàm âm này là do khi huấn luyện, chúng ta cần giảm tối đa lỗi. Giảm hàm chi phí sẽ làm tăng lỗi xảy ra, giả sử rằng các mẫu được lấy từ một phân phối chuẩn.

1.1.5 Mô hình PhoBERT

PhoBERT là một mô hình pre-trained được huấn luyện chỉ dành cho tiếng Việt, quá trình huấn luyện tương tự như RoBERTa tối ưu hóa quá trình đào tạo của BERT để đạt được hiệu suất cao hơn.

Kiến trúc PhoBERT có 2 phiên bản khác nhau, đó là PhoBERTbase và PhoBERTlarge dựa trên kiến trúc của BERT tương ứng là BERTbase và BERTlarge. Mô hình sử dụng 20GB tập dữ liệu để thực hiện quá trình huấn luyện, trong đó xấp xỉ 1GB từ kho dữ liệu của Wikipedia tiếng Việt và dữ liệu còn lại lấy từ kho dữ liệu Vietnamese News sau đó xóa những dữ liệu trùng lặp trong 50GB dữ liệu.

Sử dụng RDRSegmenter của VnCoreNLP thực hiện việc tách từ của văn bản trước khi đưa vào huấn luyện. Mô hình sử dụng fastBFE để phân đoạn các câu với các subwords, sử dụng kỹ thuật Adam để tối ưu hóa mô hình.

1.2 HTML và CSS

HTML (viết tắt của từ Hypertext Markup Language, hay là “Ngôn ngữ đánh dấu siêu văn bản”) là một ngôn ngữ đánh dấu được thiết kế ra để tạo nên cấu trúc hay bộ khung các trang web trên World Wide Web. Cùng với CSS và JavaScript, HTML là một trong những ngôn ngữ quan trọng trong lĩnh vực thiết kế website. HTML được định nghĩa như là một ứng dụng đơn giản của SGML và được sử dụng trong các tổ chức cần đến các yêu cầu xuất bản phức tạp. HTML đã trở thành một phần chuẩn mực của Internet do tổ chức World Wide Web Consortium (W3C) duy trì.

Phiên bản chính thức mới nhất của HTML là HTML 4.01 (1999). Sau đó, các nhà phát triển đã thay thế nó bằng XHTML. Hiện nay, phiên bản mới nhất của ngôn ngữ này là HTML5.

Khi làm việc với HTML, chúng ta sẽ sử dụng cấu trúc code đơn giản (tags và attributes) để đánh dấu lên trang web. Ví dụ, chúng ta có thể tạo một đoạn văn bằng cách đặt văn bản vào trong cặp tag mở và đóng văn bản `<p>` và `</p>`.

CSS là ngôn ngữ tạo phong cách cho trang web – **Cascading Style Sheet** language. Nó dùng để tạo styles và định kiểu cho những thẻ được viết dưới dạng ngôn ngữ đánh dấu, như là HTML. Nó có thể định dạng của nhiều trang web cùng lúc để tiết kiệm công sức cho người viết web. Nó phân biệt cách hiển thị của trang web với nội dung chính của trang bằng cách điều khiển bố cục, màu sắc, và font chữ.

1.3 JavaScript

JavaScript (viết tắt là JS) là một ngôn ngữ lập trình kịch bản phía máy khách, mã lệnh được thực thi bởi trình duyệt của người dùng. JavaScript được sử dụng rộng

rãi trong việc kết hợp với HTML/CSS để thiết kế trang web sinh động và tương tác với người dùng.

JavaScript là một ngôn ngữ lập trình phổ biến bậc nhất thế giới và là một trong ba ngôn ngữ không thể thiếu đối với một lập trình viên web.

1.4 ReactJS

ReactJS là một thư viện JavaScript xây dựng giao diện người dùng có tính hiệu quả và linh hoạt có thể tái sử dụng lại thông qua các thành phần gọi là components. ReactJS xây dựng các thành phần với tư tưởng phân chia, từ các thành phần lớn, phức tạp chuyển đổi thành các thành phần nhỏ hơn, dễ quản lý và tái sử dụng hơn. React được tạo ra bởi Jordan Walke, một kỹ sư phần mềm tại Facebook, phát triển và duy trì bởi Facebook và các ứng dụng nổi tiếng như WhatsApp & Instagram sử dụng công nghệ này.

ReactJS được dùng để xây dựng các ứng dụng [Single Page Application] (SPA). Một trong những điểm hấp dẫn của ReactJS là nó không chỉ được xây dựng bên phía clients mà còn sử dụng được bên phía server[4]

2. Thiết kế và cài đặt

2.1 Chuẩn bị dữ liệu

Dữ liệu của đề tài là các tin tức được phân loại thành 10 lĩnh vực riêng biệt được chứa trong 2 thư mục Train_Full(tập train) và Test_Full(tập test). Dữ liệu là các tập tin txt thuần, chưa phải là file csv, nên chúng ta cần chuyển các dữ liệu trong 2 thư mục thành 2 file csv. Chúng ta sẽ đọc qua từng tập tin trong thư mục, rồi thêm vào một DataFrame và lưu vào file csv để thực hiện việc huấn luyện cũng như đánh giá.

```
1
2 train_dir = './Train_Full'
3 test_dir = './Test_Full'
4
5 LABELS = ['Chinh tri Xa hoi', 'Doi song', 'Khoa hoc', 'Kinh doanh', 'Phap luat', 'Suc khoe', 'The gioi', 'The thao', 'Van hoa', 'Vi tinh']
6
7 def create_dataset(file_name_output, directory):
8     labels = []
9     text = []
10    for label in LABELS:
11        dir_folder = '%s/%s' % (directory, label)
12        for filename in listdir(dir_folder):
13            full_filename = '%s/%s' % (dir_folder, filename)
14            with open(full_filename, "r", encoding='utf-16') as f:
15                get_text = f.read().replace('\n', ' ')
16                text.append(get_text)
17                labels.append(label)
18
19    df = pd.DataFrame([text, labels], ['text', 'label']).T
20    df.to_csv(file_name_output, encoding='utf-8', index=False)
21
22
23 # Train dataset
24 create_dataset('train_data.csv', train_dir)
25 # Test dataset
26 create_dataset('test_data.csv', test_dir)
```

Hình 3 Tạo tập tin csv từ thư mục Train_Full và Test_Full

2.2 Khám phá dữ liệu

Phân tích khám phá dữ liệu (EDA) là bước đầu tiên trong quy trình phân tích dữ liệu của bạn do "John Tukey" phát triển vào những năm 1970. Trong thống kê, phân tích dữ liệu khám phá là một cách tiếp cận để phân tích các tập dữ liệu để tóm tắt các đặc điểm chính của chúng, thường bằng các phương pháp trực quan. Với chính cái tên, chúng ta có thể biết rằng đó là một bước mà chúng ta cần khám phá tập dữ liệu[9].

Phân tích khám phá dữ liệu cho chúng ta phân tích được điểm chung của dữ liệu, xem các dữ liệu có tốt hay không? Có cần phải xử lý gì không? Có cần sử dụng tất cả các thuộc tính hay chỉ sử dụng những thuộc tính thật sự quan trọng hay không? Và còn rất nhiều vấn đề cần được giải quyết sau khi phân tích khám phá dữ liệu. Phân tích dữ liệu có thể cho ta cái nhìn trực quan hơn về dữ liệu, sử dụng các biểu đồ để dễ dàng nhìn thấy và đánh giá dữ liệu.

2.2.1 Mô tả tập dữ liệu

Có 2 tập dữ liệu là huấn luyện và kiểm tra, tập huấn luyện có 33.757 mẫu tin và tập kiểm tra có 50.374 mẫu tin. Mỗi tập dữ liệu có 2 cột, cột đầu tiên(text) là các tin

tức được đọc từ tập tin txt, cột tiếp theo là các lĩnh vực(label) ứng với tin tức ở cột text. Tập dữ liệu có số lượng mẫu tin khá lớn nên thời gian mất khá nhiều thời gian để huấn luyện cũng như dự đoán và đánh giá, tất cả các giá trị đều có kiểu dữ liệu là chuỗi

	text	label
0	Thành lập dự án POLICY phòng chống HIV/AIDS ở...	Chính trị Xa hoi
1	Hơn 16.000 khách đến vịnh Nha Trang Theo trực...	Chính trị Xa hoi
2	TPHCM: Khai trương dịch vụ lặn biển săn cá mậ...	Chính trị Xa hoi
3	Du lịch VN sẽ có tư vấn nước ngoài Ông Phạm T...	Chính trị Xa hoi
4	Quy chế tuyển sinh 2006: Không làm tròn điểm ...	Chính trị Xa hoi
...
33754	Điện thoại di động tương lai trông như thế nào...	Ví tính
33755	Internet sẽ tăng tốc 1.000 lần Trong tương la...	Ví tính
33756	Phần lớn thế giới thứ 3 thất bại với chính phủ...	Ví tính
33757	'Doom 3' giành chiến thắng kép Trò chơi hành ...	Ví tính
33758	Lỗi về Window Media Player (1) Tôi đang dùng W...	Ví tính
33759 rows × 2 columns		

Hình 4: Tập dữ liệu huấn luyện

	text	label
0	Mạo hiểm rừng Đa Mi Cuộc hành quân khám phá t...	Chính trị Xa hoi
1	Tàu du lịch cao tốc Cần Thơ - Phnom Penh Công...	Chính trị Xa hoi
2	Miền Trung được mùa khách Thái Đoàn du khách ...	Chính trị Xa hoi
3	7 kỳ quan mới của thế giới Cầu Akashi - Kaiky...	Chính trị Xa hoi
4	Khối A thi được mấy trường? Thi khối A vào ĐH...	Chính trị Xa hoi
...
50368	Phần mềm chống tải nhạc bất hợp pháp Nhóm nghi...	Vĩ tinh
50369	Linux tăng ảnh hưởng trên thị trường máy chủ T...	Vĩ tinh
50370	Napster phát không máy nghe nhạc Công ty phát ...	Vĩ tinh
50371	Intel giới thiệu chip và chipset chuyên biệt h...	Vĩ tinh
50372	Yahoo lại bị trì trệ Lần thứ 2 trong chưa đầy ...	Vĩ tinh
50373 rows × 2 columns		

Hình 5: Tập dữ liệu kiểm tra

2.2.2 Xử lý dữ liệu bị thiếu

Dữ liệu bị thiếu hay bị mất mát là hiện tượng khá phổ biến khi thu thập dữ liệu, có nhiều nguyên nhân nhưng phổ biến là lỗi nhập liệu, người khảo sát không điền thông tin và nhiều nguyên nhân khác. Dữ liệu bị thiếu dẫn đến quá trình huấn luyện bị giảm độ chính xác không mong muốn và có thể dẫn đến quá trình dự đoán không chính xác vì thế cần phải xử lý cẩn thận các giá trị bị thiếu.

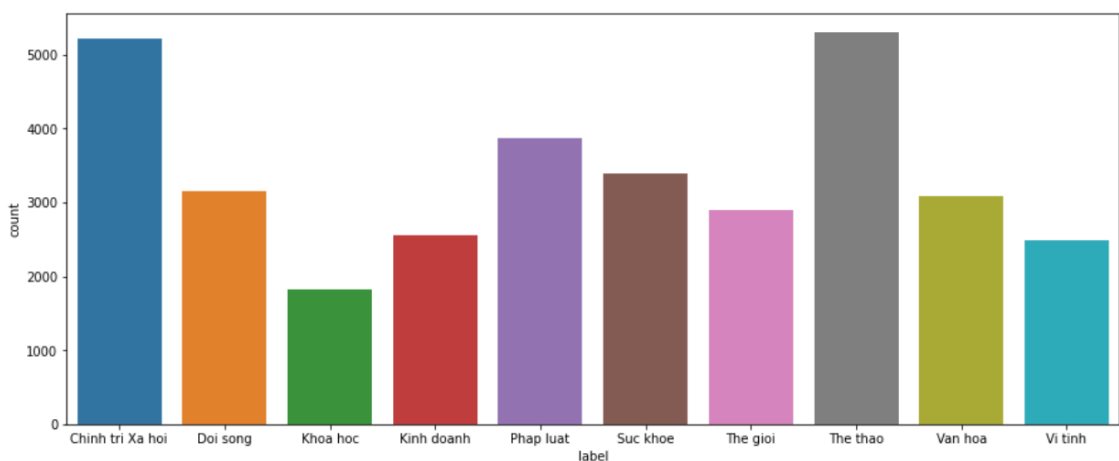
Dữ liệu đã cho không có dấu hiệu bị mất mát dữ liệu, nên chúng ta sẽ bỏ qua bước này. Lưu ý nếu dữ liệu bị mất mát thì phải dùng các phương pháp để xử lý dữ liệu bị thiếu, trước khi thực hiện huấn luyện.

```
Train data
text      0
label     0
dtype: int64
Test data
text      0
label     0
dtype: int64
```

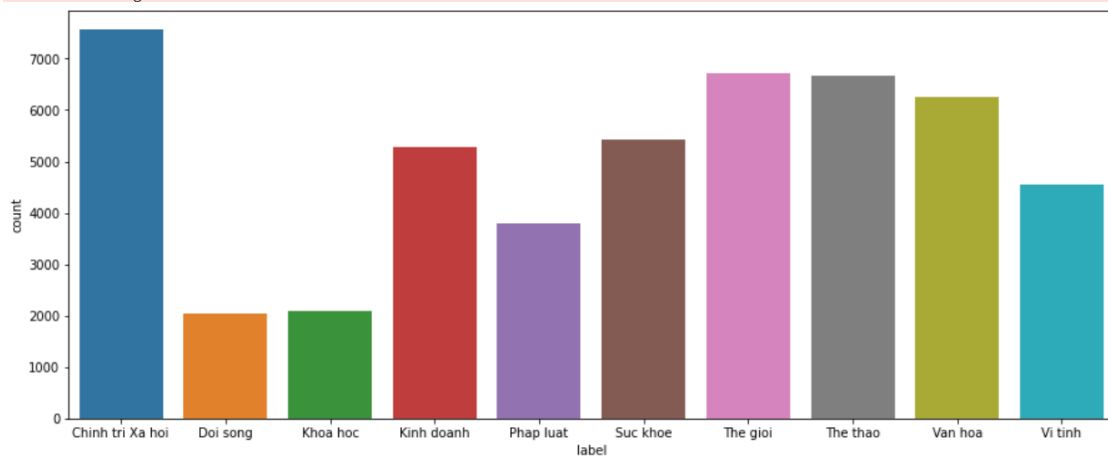
Hình 6: Kiểm tra dữ liệu bị thiếu

2.2.3 Mô hình hóa dữ liệu

Để có một cái nhìn trực quan về dữ liệu thì các biểu đồ, hình ảnh sẽ chúng ta có thể dễ dàng hình dung, đưa ra nhận xét về sự phân bố, tương quan của dữ liệu với nhau. Mô hình hóa không chỉ giúp nhà phát triển có thể hiểu mà những người ngoài ngành có thể hiểu được, đó là điều rất quan trọng trong việc đánh giá, ước lượng từ chuyên gia trong lĩnh vực nghiên cứu. Một biểu đồ có giá trị hơn nhiều so với một bảng tính, một danh sách liệt kê.



Hình 7: Mô hình hóa số lượng mẫu tin của từng phân lớp trong tập huấn luyện



Hình 8: Mô hình hóa số lượng mẫu tin của từng phân lớp trong tập kiểm tra

2.3 Tiền xử lý dữ liệu

Đối với dữ liệu mới thu thập là văn bản thô không thích hợp để đưa vào mô hình máy học huấn luyện được, dữ liệu được đưa vào mô hình phải qua bước tiền xử lý, loại bỏ các thành phần không cần thiết, chuẩn hóa dữ liệu. Quá trình tiền xử lý dữ liệu là một bước quan trọng trong phân loại dữ liệu văn bản, nó ảnh hưởng trực tiếp đến quá trình huấn luyện và dự đoán, vì vậy đây là bước thật sự cần thiết.

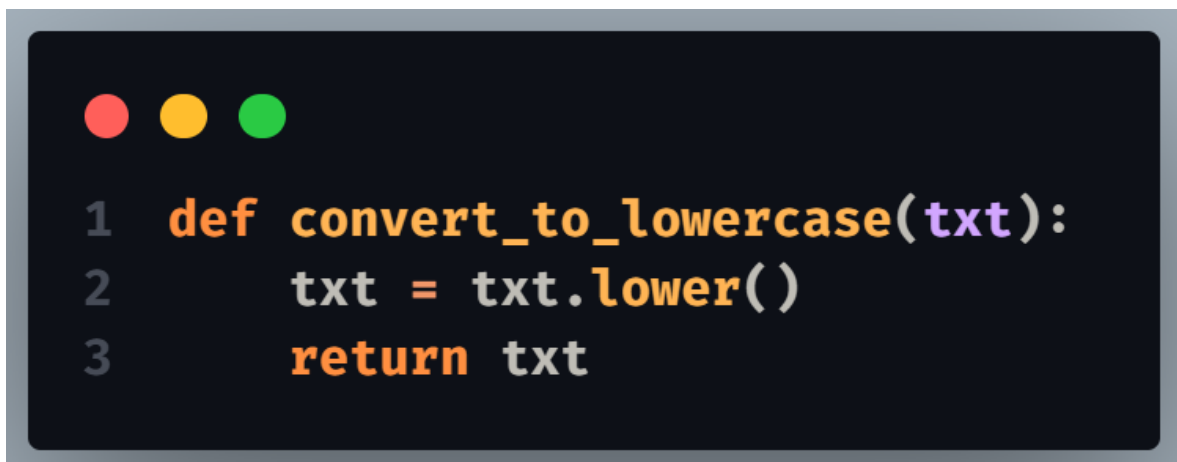
Đối với bài toán phân loại văn bản trong đề tài này chúng tôi thực hiện các bước sau để tiền xử lý dữ liệu:

- Chuyển đổi dữ liệu về dạng chữ thường
- Loại bỏ các ký tự số
- Loại các ký tự đơn, bắt đầu bằng ký tự đơn và khoảng trắng
- Chuẩn hóa về Unicode dạng sẵn
- Chuẩn hóa kiểu gõ dấu tiếng Việt
- Loại bỏ những ký tự lặp
- Loại bỏ các từ dừng

- Tách từ tiếng Việt (sử dụng thư viện pyvi và vncorenlp)

2.3.1 Chuyển đổi dữ liệu về dạng chữ thường

Việc đưa dữ liệu về chữ viết thường là rất cần thiết. Bởi vì đặc trưng này không có tác dụng ở bài toán phân loại văn bản. Đưa về chữ viết thường giúp giảm số lượng đặc trưng (vì máy tính hiểu hoa thường là 2 từ khác nhau) và tăng độ chính xác hơn cho mô hình[10].

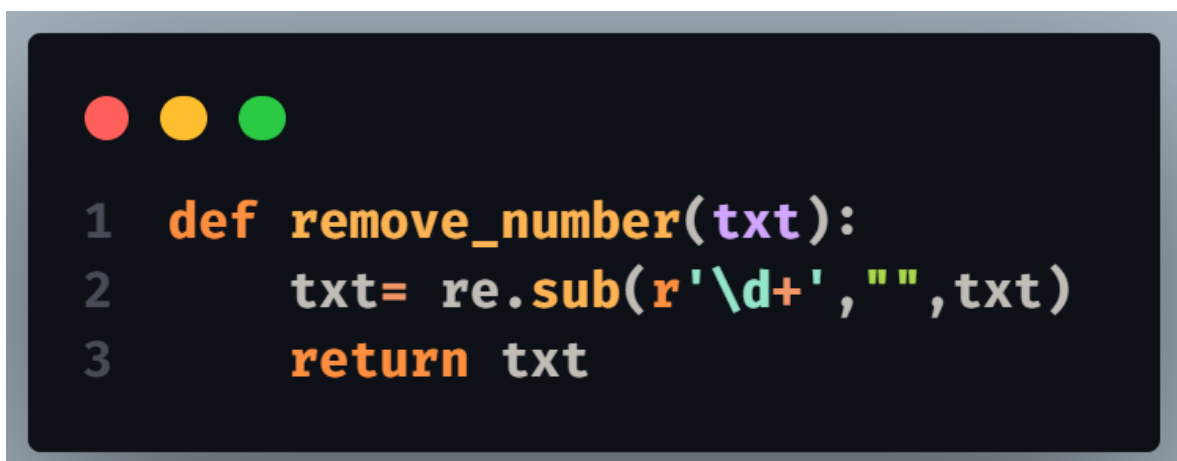
A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The code is written in Python and defines a function named `convert_to_lowercase` that takes a parameter `txt`. The function body consists of two lines: `txt = txt.lower()` and `return txt`.

```
1 def convert_to_lowercase(txt):  
2     txt = txt.lower()  
3     return txt
```

Hình 9: Hàm chuyển đổi dữ liệu về dạng chữ thường

2.3.2 Loại bỏ các ký tự số

Loại bỏ các ký tự số không cần thiết trong văn bản.

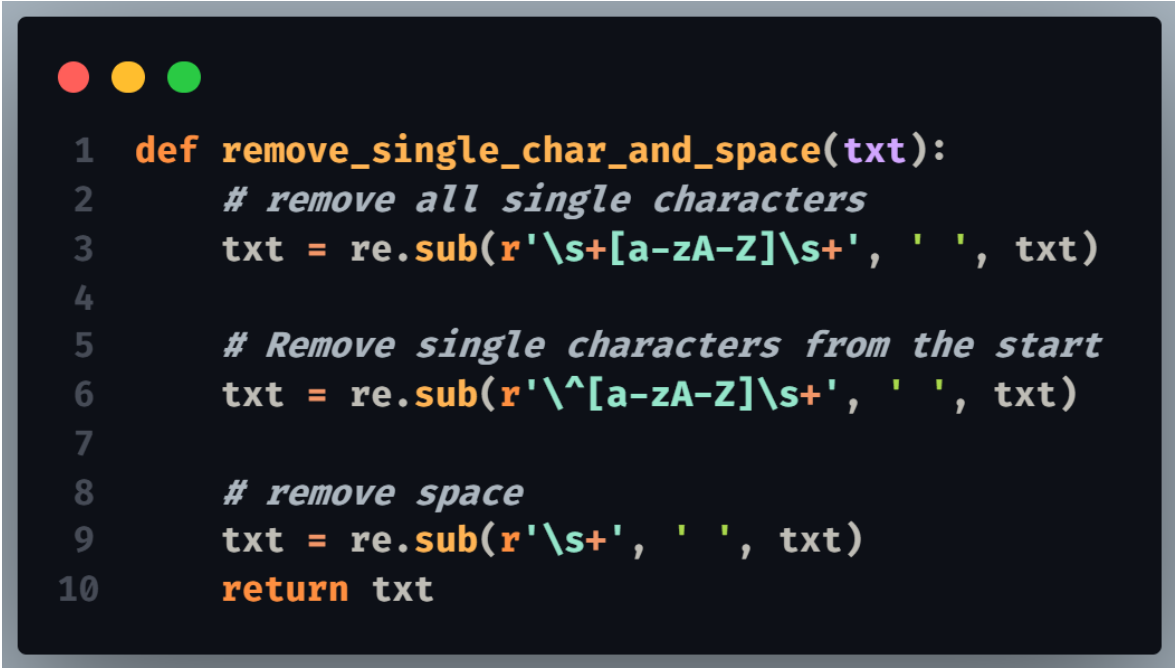
A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The code is written in Python and defines a function named `remove_number` that takes a parameter `txt`. The function body consists of two lines: `txt = re.sub(r'\d+', '', txt)` and `return txt`.

```
1 def remove_number(txt):  
2     txt = re.sub(r'\d+', '', txt)  
3     return txt
```

Hình 10: Hàm loại bỏ các ký tự số

2.3.3 Loại các ký tự đơn, bắt đầu bằng ký tự đơn và khoảng trắng

Các ký tự đơn trong văn bản không mang nhiều ý nghĩa, một số trường hợp các ký tự này đứng đầu câu thì nên loại bỏ. Các khoảng trắng dư thừa trong câu cần phải xóa chỉ để lại khoảng trắng phân tách các từ.



```
1 def remove_single_char_and_space(txt):
2     # remove all single characters
3     txt = re.sub(r'\s+[a-zA-Z]\s+', ' ', txt)
4
5     # Remove single characters from the start
6     txt = re.sub(r'^[a-zA-Z]\s+', ' ', txt)
7
8     # remove space
9     txt = re.sub(r'\s+', ' ', txt)
10    return txt
```

Hình 11: Hàm loại các ký tự đơn, bắt đầu bằng ký tự đơn và khoảng trắng

2.3.4 Chuẩn hóa về Unicode dựng sẵn

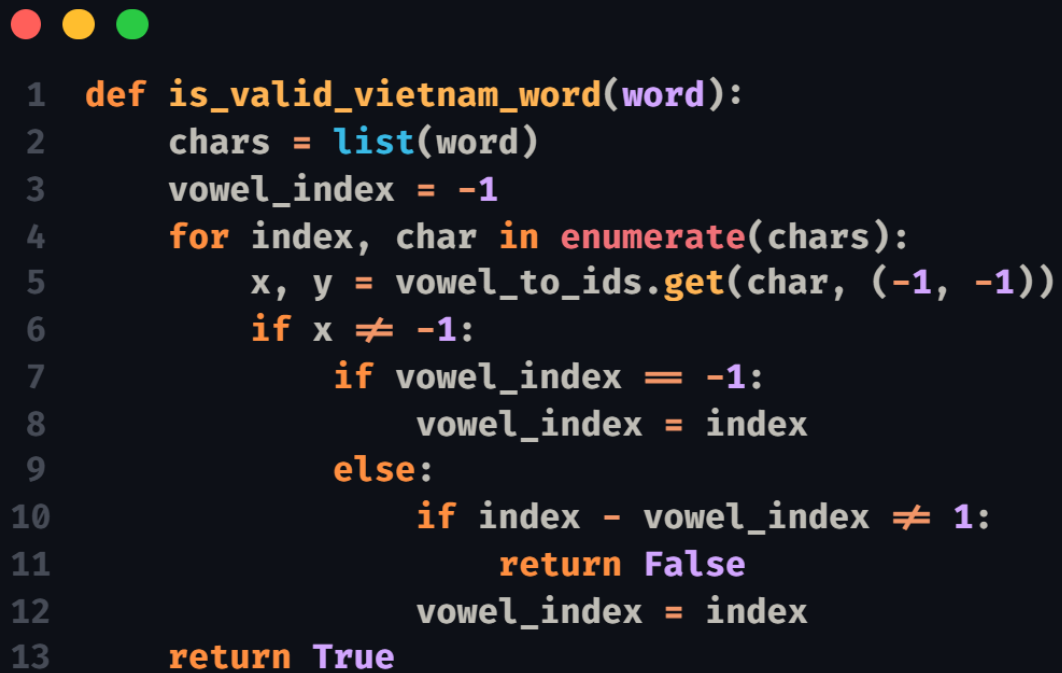
Có 2 kiểu Unicode được dùng trong tiếng Việt, để đồng bộ trong quá trình xử lý, chúng ta cần chuyển đổi thành một kiểu Unicode.

[illegible]

Hình 12: Hàm chuẩn hóa về Unicode định sẵn

2.3.5 Chuẩn hóa kiểu gõ dấu tiếng Việt

Hiện tại đa số đã sử dụng kiểu gõ dấu hiện đại nhưng đôi khi các văn bản còn sử dụng kiểu gõ cổ điển nên chúng tôi sẽ thực hiện việc chuẩn hóa kiểu gõ dấu về một kiểu gõ, để thuận tiện trong quá trình xử lý.

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The code is a Python function named `is_valid_vietnam_word` that takes a `word` as input. It converts the word into a list of characters and iterates through them using `enumerate`. For each character, it looks up its index in a dictionary `vowel_to_ids`. If the character is a vowel (index $\neq -1$), it checks if the difference between the current index and the vowel index is 1. If not, it returns `False`. If the character is not a vowel, it updates the `vowel_index` to the current index. If the loop completes without returning `False`, it returns `True`.

```
1 def is_valid_vietnam_word(word):
2     chars = list(word)
3     vowel_index = -1
4     for index, char in enumerate(chars):
5         x, y = vowel_to_ids.get(char, (-1, -1))
6         if x  $\neq$  -1:
7             if vowel_index == -1:
8                 vowel_index = index
9             else:
10                 if index - vowel_index  $\neq$  1:
11                     return False
12                 vowel_index = index
13     return True
```

Hình 13: Hàm kiểm tra từ tiếng Việt hợp lệ

```
1 def standard_vietnam_word_marks(word):
2     if not is_valid_vietnam_word(word):
3         return word
4
5     chars = list(word)
6     dau_cau = 0
7     vowel_index = []
8     qu_or_gi = False
9     for index, char in enumerate(chars):
10        x, y = vowel_to_ids.get(char, (-1, -1))
11        if x == -1:
12            continue
13        elif x == 9: # check qu
14            if index != 0 and chars[index - 1] == 'q':
15                chars[index] = 'u'
16                qu_or_gi = True
17        elif x == 5: # check gi
18            if index != 0 and chars[index - 1] == 'g':
19                chars[index] = 'i'
20                qu_or_gi = True
21        if y != 0:
22            dau_cau = y
23            chars[index] = vowel_table[x][0]
24        if not qu_or_gi or index != 1:
25            vowel_index.append(index)
26    if len(vowel_index) < 2:
27        if qu_or_gi:
28            if len(chars) == 2:
29                x, y = vowel_to_ids.get(chars[1])
30                chars[1] = vowel_table[x][dau_cau]
31            else:
32                x, y = vowel_to_ids.get(chars[2], (-1, -1))
33                if x != -1:
34                    chars[2] = vowel_table[x][dau_cau]
35                else:
36                    chars[1] = vowel_table[5][dau_cau] if chars[1] == 'i' else vowel_table[9][dau_cau]
37        return ''.join(chars)
38    return word
39
40    for index in vowel_index:
41        x, y = vowel_to_ids[chars[index]]
42        if x == 4 or x == 8: # ê, ô
43            chars[index] = vowel_table[x][dau_cau]
44        return ''.join(chars)
45
46    if len(vowel_index) == 2:
47        if vowel_index[-1] == len(chars) - 1:
48            x, y = vowel_to_ids[chars[vowel_index[0]]]
49            chars[vowel_index[0]] = vowel_table[x][dau_cau]
50        else:
51            x, y = vowel_to_ids[chars[vowel_index[1]]]
52            chars[vowel_index[1]] = vowel_table[x][dau_cau]
53    else:
54        x, y = vowel_to_ids[chars[vowel_index[1]]]
55        chars[vowel_index[1]] = vowel_table[x][dau_cau]
56    return ''.join(chars)
```

Hình 14: Hàm chuẩn hóa từ gõ dấu tiếng Việt


```

1 def standard_vietnam_sentence_marks(sentence):
2     sentence = sentence.lower()
3     words = sentence.split()
4     for index, word in enumerate(words):
5         cw = re.sub(r'(^\\p{P}*)([p{L}.]*\\p{L}+)(\\p{P}*$)', r'\1/\2/\3', word).split('/')
6         # print(cw)
7         if len(cw) == 3:
8             cw[1] = standard_vietnam_word_marks(cw[1])
9             words[index] = ''.join(cw)
10    return ' '.join(words)

```

Hình 15: Hàm chuẩn hóa câu gõ dấu tiếng Việt

2.3.6 Loại bỏ những ký tự lặp

Những ký tự lặp trong một từ có thể bị hiểu không đúng với nghĩa thực sự của từ đó, xóa bỏ những ký tự lặp giúp cải thiện độ chính xác mô hình.

```
1 def remove_loop_char(txt):  
2     txt = re.sub(r'([A-Z])\1+', lambda m: m.group(1).upper(), str(txt), flags=re.IGNORECASE)  
3     txt = re.sub(r'^\s\\wáàáâãäåäääåäåäåäåäéëêëëëëëëëëëëöóôõöôôôôôôôôôôðþíîïñúûüýÿýÿýÿd_]', ' ',txt)  
4     return txt
```

Hình 16: Hàm loại bỏ ký tự lặp

2.3.7 Loại bỏ các từ dừng

Từ dừng là những từ xuất hiện trong hầu hết các chủ đề cần phân loại, do đó các từ này thường không mang ý nghĩa cho một chủ đề cụ thể nào, không có tác dụng trong việc phân loại.

Các từ dừng thường là các từ nối (của, là, có, được, những,...) và các từ đặc trưng của dữ liệu (ví dụ như các từ “máy bay, tiếp viên” là các stopwords nếu làm bài phân loại đánh giá khách hàng của doanh nghiệp vận tải hàng không[10])

Một số từ dùng trong danh sách từ dùng:

```
a lô  
a ha  
ai  
ai ai  
ai nấy  
ai đó  
alô  
amen  
anh  
anh ấy  
ba  
ba ba  
ba bản  
ba cùng  
ba họ
```

Hình 17: Danh sách từ dừng

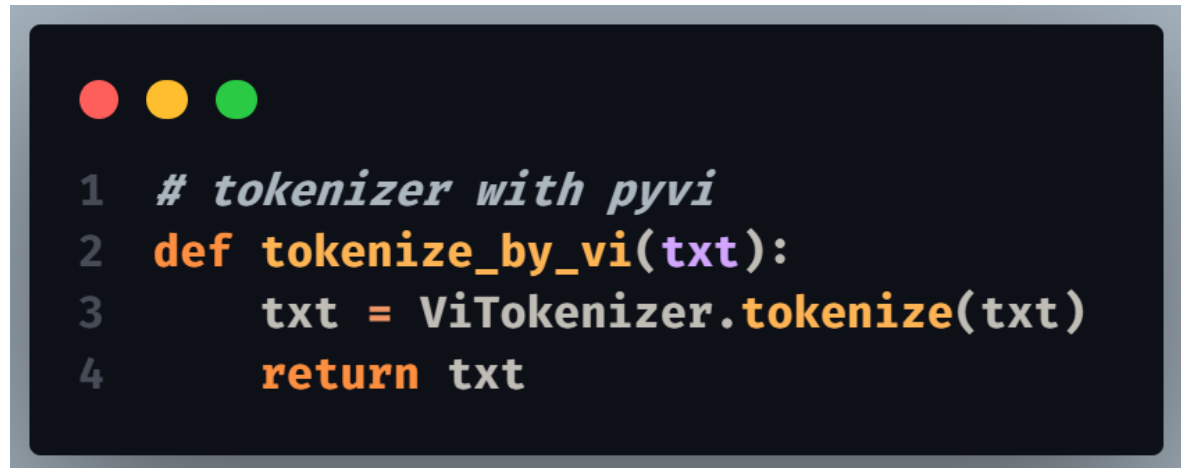
```
1 def get_stopwords_to_list(path):  
2     stopwords = []  
3     with open(path, 'r', encoding="utf-8") as f:  
4         stopwords_list = list(f)  
5         stopwords = [re.sub('\n', '', word) for word in stopwords_list]  
6     return stopwords
```

Hình 18: Hàm lấy danh sách từ dừng

2.3.8 Tách từ tiếng Việt

Tách từ là một việc làm cần thiết trong quá trình tiền xử lý dữ liệu với các bài toán phân loại văn bản, tách từ tức là từ một câu, văn bản ban đầu ta sẽ phân chia thành các đơn vị từ nhỏ hơn. Ngôn ngữ tiếng Việt là ngôn ngữ đa âm tiếng, có 2 đơn vị từ là từ đơn và từ ghép vì vậy chúng ta cần sử dụng công cụ phù hợp dành cho

tiếng Việt để thực hiện việc tách từ. Trong đề tài này chúng tôi sử dụng pyvi và vncoreNLP.



```
1 # tokenizer with pyvi
2 def tokenize_by_vi(txt):
3     txt = ViTokenizer.tokenize(txt)
4     return txt
```

Hình 19: Hàm tách từ với pyvi



```
1 # tokenizer with vncorenlp
2 def tokenize_by_vn(txt):
3     txt = rdrsegmenter.tokenize(txt)
4     txt = ' '.join([' '.join(x) for x in txt])
5     return txt
```

Hình 20: Hàm tách từ với vncoreNLP

2.3.9 Thực hiện làm sạch dữ liệu

```
1 def clean_data(txt):
2     # convert txt to lowercase
3     txt = convert_to_lowercase(txt)
4     # remove number
5     txt = remove_number(txt)
6     # remove single char and space
7     txt = remove_single_char_and_space(txt)
8     # convert txt from unicode
9     txt = convert_unicode(txt)
10    # standard vietnam sentence marks
11    txt = standard_vietnam_sentence_marks(txt)
12    # remove loop char
13    txt = remove_loop_char(txt)
14    # remove stopwords
15    txt = remove_stopwords(txt)
16    return txt
```

Hình 21: Hàm làm sạch dữ liệu

```
1 # preprocess
2 tqdm.pandas()
3 train_data['preprocess_text'] = train_data['text'].progress_apply(lambda s : clean_data(s))
4 tqdm.pandas()
5 test_data['preprocess_text'] = test_data['text'].progress_apply(lambda s : clean_data(s))
```

Hình 22: Làm sạch dữ liệu huấn luyện và kiểm tra

```
1 # tokenizer with pyvi
2 tqdm.pandas()
3 train_data['preprocess_text_by_vi'] = train_data['preprocess_text'].progress_apply(lambda s : tokenize_by_vi(s))
4 tqdm.pandas()
5 test_data['preprocess_text_by_vi'] = test_data['preprocess_text'].progress_apply(lambda s : tokenize_by_vi(s))
```

Hình 23: Tách từ dữ liệu huấn luyện và kiểm tra bằng pyvi

```
1 # tokenizer with vncoreNLP
2 tqdm.pandas()
3 train_data['preprocess_text_by_vn'] = train_data['preprocess_text'].progress_apply(lambda s : tokenize_by_vn(s))
4 tqdm.pandas()
5 test_data['preprocess_text_by_vn'] = test_data['preprocess_text'].progress_apply(lambda s : tokenize_by_vn(s))
```

Hình 24: Tách từ dữ liệu huấn luyện và kiểm tra bằng vncoreNLP

	text	label	preprocess_text	preprocess_text_by_vi	preprocess_text_by_vn
0	Thành lập dự án POLICY phòng chống HIV/AIDS ở...	Chính trị Xã hội	thành lập dự án policy phòng chống hivaid... ...	thành_lập dự_án policy phòng_chống hivaid... ...	thành_lập dự_án policy phòng_chống hivaid... ...
1	Hơn 16.000 khách đến vịnh Nha Trang Theo tr...	Chính trị Xã hội	vịnh nha trang trực ban đội biên phòng cắ... du...	vịnh nha trang trực_ban đội biên_phòng cắ... du...	vịnh nha_trang trực_ban đội biên_phòng cắ... du...
2	TPHCM: Khai trương dịch vụ lặn biển sản cá mậ...	Chính trị Xã hội	tpchm khai trương dịch vụ lặn biển sản cá mậ... ...	tpchm khai_trương dịch_vụ lặn biển sản cá_mậ... ...	tpchm khai_trương dịch_vụ lặn biển sản cá_mậ... ...
3	Du lịch VN sẽ có tư vấn nước ngoài Ông Phạm T...	Chính trị Xã hội	du lịch vn tư vấn phạm phó tổng cục trưởng... tôn...	du_lịch vn tư_vấn phạm phó_tổng_cục trưởng... tôn...	du_lịch vn tư_vấn phạm phó_tổng_cục trưởng... tôn...

Hình 25: Dữ liệu sau quá trình làm sạch

2.3.10 Vector hóa văn bản

Vector hóa văn bản là một bước quan trọng để huấn luyện mô hình, máy tính không thể hiểu được ngôn ngữ tự nhiên của con người, vì vậy phải chuyển về dữ liệu có thể hiểu và học được. Chúng ta sẽ sử dụng phương pháp tf-idf để chuyển văn bản thành các vector số học. Sau khi vector hóa ta sẽ lưu lại mô hình vector để vector hóa dữ liệu mới đến khi dự đoán ở phía website.

```
1 from sklearn.feature_extraction.text import TfidfVectorizer
2
3 count_vect = TfidfVectorizer()
4 X_train = count_vect.fit_transform(train_data['preprocess_text_by_vi'])
5 X_test = count_vect.transform(test_data['preprocess_text_by_vi'])
```

Hình 26: Vector hóa dữ liệu văn bản

```
1 with open('tfidf.pkl', 'wb') as file:
2     pickle.dump(count_vect, file)
```

Hình 27: Lưu lại mô hình tf-idf

2.3.11 Xáo trộn dữ liệu

Để dữ liệu được huấn luyện một cách ngẫu nhiên, chúng ta tiến hành xáo trộn tập dữ liệu

```
1 from sklearn.utils import shuffle
2 train_data = shuffle(train_data)
3 test_data = shuffle(test_data)
```

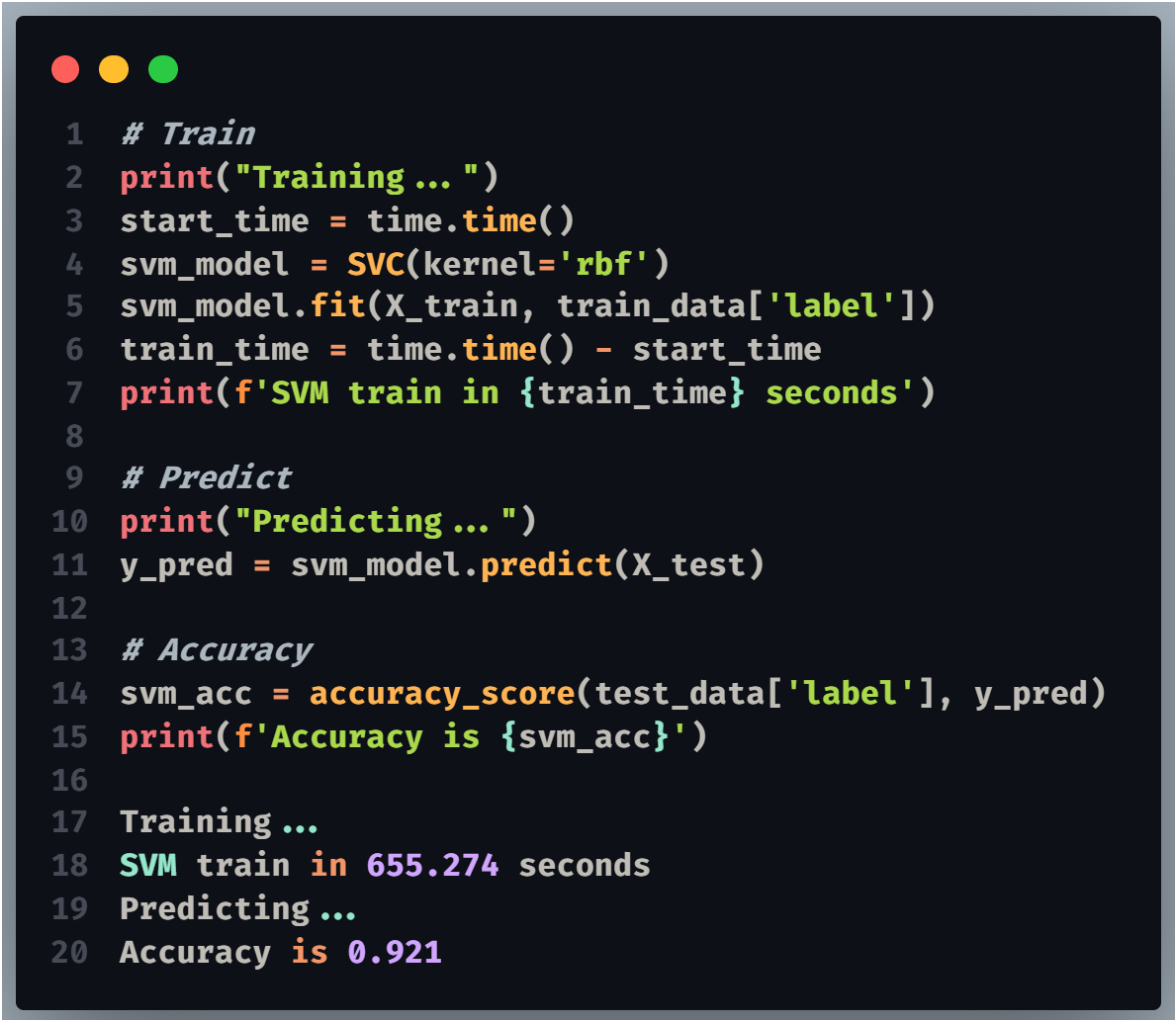
Hình 28: Xáo trộn dữ liệu

2.4 Xây dựng mô hình phân loại văn bản

2.4.1 Xây dựng mô hình phân loại văn bản với SVM

SVM (Support Vector Machine) là một thuật toán học máy có giám sát được sử dụng rất phổ biến ngày nay trong các bài toán phân lớp (classification). SVM rất hiệu quả để giải quyết các bài toán với dữ liệu có số chiều lớn như các véc-tơ biểu diễn văn bản. [2].

Chúng tôi sử dụng mô hình **SVM** từ thư viện **sklearn** với phương pháp tách từ bằng **pyvi**:



```
1  # Train
2  print("Training ... ")
3  start_time = time.time()
4  svm_model = SVC(kernel='rbf')
5  svm_model.fit(X_train, train_data['label'])
6  train_time = time.time() - start_time
7  print(f'SVM train in {train_time} seconds')
8
9  # Predict
10 print("Predicting ... ")
11 y_pred = svm_model.predict(X_test)
12
13 # Accuracy
14 svm_acc = accuracy_score(test_data['label'], y_pred)
15 print(f'Accuracy is {svm_acc}')
16
17 Training ...
18 SVM train in 655.274 seconds
19 Predicting ...
20 Accuracy is 0.921
```

Hình 29: Mô hình SVM

	precision	recall	f1-score	support
Suc khoe	0.84	0.92	0.88	7567
The gioi	0.78	0.72	0.75	2036
Van hoa	0.87	0.78	0.82	2096
The thao	0.94	0.88	0.91	5276
Phap luat	0.93	0.91	0.92	3788
Kinh doanh	0.93	0.95	0.94	5417
Chinh tri Xa hoi	0.96	0.93	0.95	6716
Vi tinh	0.98	0.98	0.98	6667
Khoa hoc	0.94	0.95	0.94	6250
Doi song	0.93	0.96	0.95	4560
accuracy			0.92	50373
macro avg	0.91	0.90	0.90	50373
weighted avg	0.92	0.92	0.92	50373

Hình 30: Đánh giá trên từng phân lớp - SVM

2.4.2 Xây dựng mô hình phân loại văn bản với Naive Bayes

Naive Bayes là một thuật toán phổ biến trong máy học được đánh giá là một trong những phương pháp có hiệu năng cao nhất khi thực hiện phân lớp văn bản. Naive Bayes là một thuật toán phân lớp được mô hình hóa dựa trên định lý Bayes về lý thuyết xác suất để đưa ra các phán đoán cũng như phân loại dữ liệu dựa trên các dữ liệu được sát và thống kê.

Chúng tôi sử dụng mô hình **MultinomialNB** từ thư viện **sklearn** với phương pháp tách từ bằng **pyvi**:


```
1  # Train
2  print("Training... ")
3  start_time = time.time()
4  mul_model = MultinomialNB()
5  mul_model.fit(X_train, train_data['label'])
6  train_time = time.time() - start_time
7  print(f'Naive Bayes train in {train_time} seconds')
8
9  # Predict
10 print("Predicting... ")
11 y_mul_pred = mul_model.predict(X_test)
12
13 # Accuracy
14 mul_acc = accuracy_score(test_data['label'], y_mul_pred)
15 print(f'Accuracy is {mul_acc}')
16
17 Training ...
18 Naive Bayes train in 0.23 seconds
19 Predicting ...
20 Accuracy is 0.813
```

Hình 31: Mô hình Naive Bayes

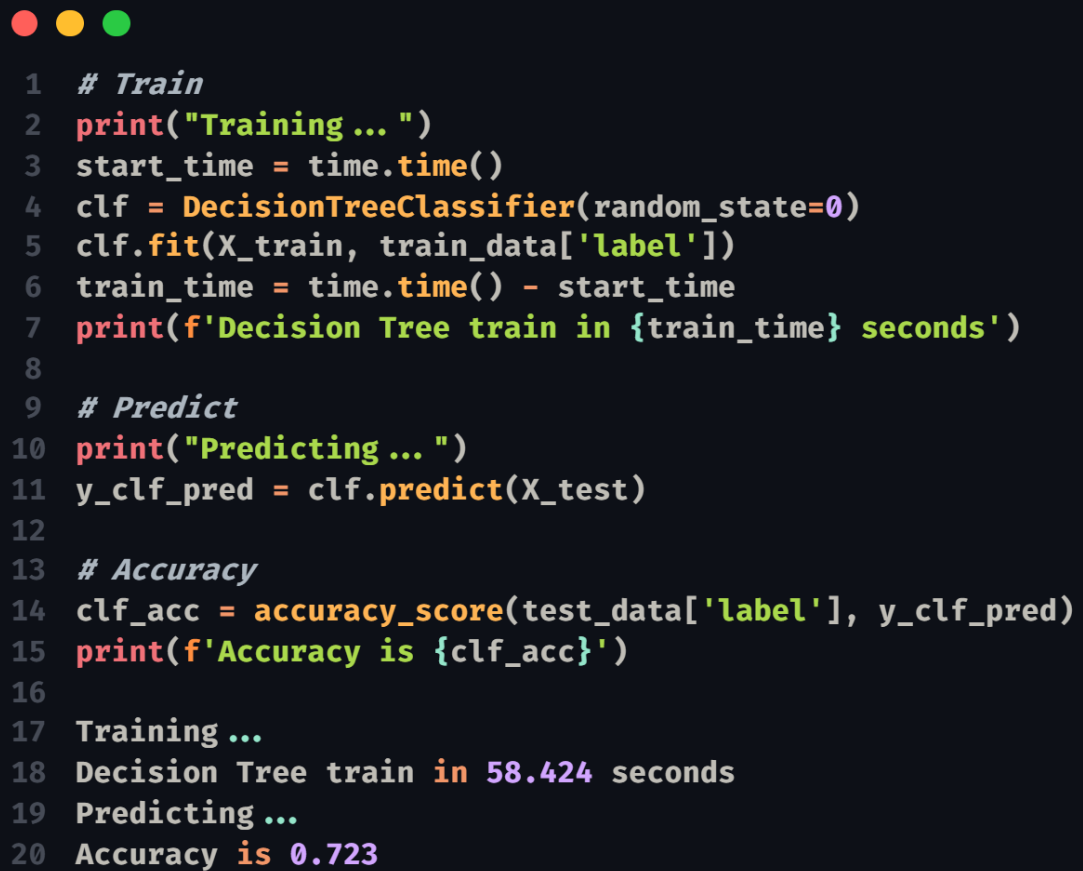
	precision	recall	f1-score	support
Suc khoe	0.52	0.97	0.68	7567
The gioi	0.69	0.56	0.62	2036
Van hoa	0.97	0.20	0.33	2096
The thao	0.97	0.53	0.69	5276
Phap luat	0.88	0.86	0.87	3788
Kinh doanh	0.89	0.91	0.90	5417
Chinh tri Xa hoi	0.97	0.79	0.87	6716
Vi tinh	0.95	0.98	0.97	6667
Khoa hoc	0.94	0.88	0.91	6250
Doi song	0.96	0.79	0.87	4560
accuracy			0.81	50373
macro avg	0.88	0.75	0.77	50373
weighted avg	0.87	0.81	0.81	50373

Hình 32: Đánh giá trên từng phân lớp - Naive Bayes

2.4.3 Xây dựng mô hình phân loại văn bản với Decision Tree

Cây quyết định (Decision Tree) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật. Các thuộc tính của đối tượng có thể thuộc các kiểu dữ liệu khác nhau như Nhị phân (Binary) , Định danh (Nominal), Thứ tự (Ordinal), Số lượng (Quantitative) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal. Với dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các dữ liệu chưa biết[7].

Chúng tôi sử dụng mô hình **DecisionTreeClassifier** từ thư viện **sklearn** với phương pháp tách từ bằng pyvi:



```
1  # Train
2  print("Training... ")
3  start_time = time.time()
4  clf = DecisionTreeClassifier(random_state=0)
5  clf.fit(X_train, train_data['label'])
6  train_time = time.time() - start_time
7  print(f'Decision Tree train in {train_time} seconds')
8
9  # Predict
10 print("Predicting... ")
11 y_clf_pred = clf.predict(X_test)
12
13 # Accuracy
14 clf_acc = accuracy_score(test_data['label'], y_clf_pred)
15 print(f'Accuracy is {clf_acc}')
16
17 Training ...
18 Decision Tree train in 58.424 seconds
19 Predicting ...
20 Accuracy is 0.723
```

Hình 33: Mô hình cây quyết định

	precision	recall	f1-score	support
Suc khoe	0.60	0.63	0.61	7567
The gioi	0.35	0.34	0.35	2036
Van hoa	0.46	0.51	0.48	2096
The thao	0.71	0.67	0.69	5276
Phap luat	0.70	0.79	0.74	3788
Kinh doanh	0.77	0.77	0.77	5417
Chinh tri Xa hoi	0.80	0.76	0.78	6716
Vi tinh	0.91	0.92	0.91	6667
Khoa hoc	0.80	0.76	0.78	6250
Doi song	0.73	0.71	0.72	4560
accuracy			0.72	50373
macro avg	0.68	0.69	0.68	50373
weighted avg	0.73	0.72	0.72	50373

Hình 34: Đánh giá trên từng phân lớp - Decision Tree

2.4.4 Xây dựng mô hình phân loại văn bản với Logistic Regression

Logistic Regression là một kỹ thuật phân loại được máy học mượn từ lĩnh vực thống kê. Logistic Regression là một phương pháp thống kê để phân tích một tập dữ liệu trong đó có một hoặc nhiều biến độc lập xác định một kết quả. Mục đích đằng sau việc sử dụng Logistic Regression là tìm ra mô hình phù hợp nhất để mô tả mối quan hệ giữa biến phụ thuộc và biến độc lập[8].

Chúng tôi sử dụng mô hình **LogisticRegression** từ thư viện **sklearn** với phương pháp tách từ bằng pyvi:

```
1 # Train
2 print("Training... ")
3 start_time = time.time()
4 scikit_log_reg = LogisticRegression(verbose=1, solver='liblinear', random_state=0,
5                                     C=5, penalty='l2', max_iter=1000)
6 scikit_log_reg.fit(X_train, train_data['label'])
7 train_time = time.time() - start_time
8 print(f'Logistic Regression train in {train_time} seconds')
9
10 # Predict
11 print("Predicting... ")
12 y_log_reg_pred = scikit_log_reg.predict(X_test)
13
14 # Accuracy
15 scikit_log_reg_acc = accuracy_score(test_data['label'], y_log_reg_pred)
16 print(f'Accuracy is {scikit_log_reg_acc}')
17
18 Training ...
19 Logistic Regression train in 24.88 seconds
20 Predicting ...
21 Accuracy is 0.922
```

Hình 35: Mô hình Logistic Regression

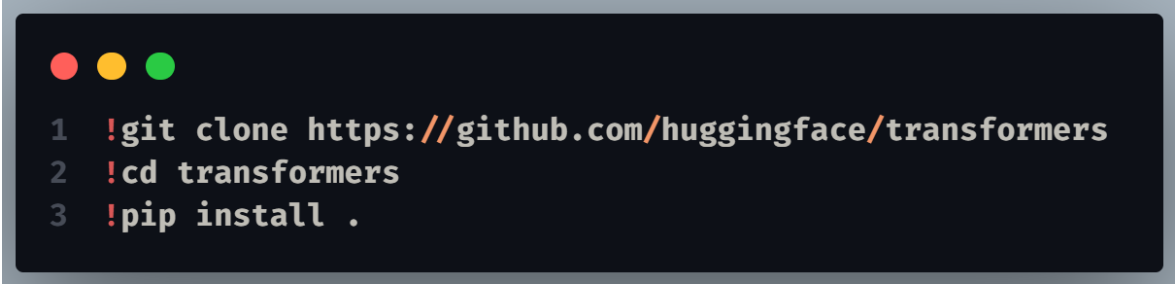
	precision	recall	f1-score	support
Suc khoe	0.86	0.91	0.88	7567
The gioi	0.81	0.69	0.74	2036
Van hoa	0.86	0.78	0.82	2096
The thao	0.94	0.89	0.91	5276
Phap luat	0.91	0.92	0.91	3788
Kinh doanh	0.93	0.96	0.94	5417
Chinh tri Xa hoi	0.96	0.93	0.95	6716
Vi tinh	0.98	0.98	0.98	6667
Khoa hoc	0.93	0.95	0.94	6250
Doi song	0.94	0.96	0.95	4560
accuracy			0.92	50373
macro avg	0.91	0.90	0.90	50373
weighted avg	0.92	0.92	0.92	50373

Hình 36: Đánh giá trên từng phân lớp - Logistic Regression

2.4.5 Xây dựng mô hình phân loại văn bản với PhoBERT

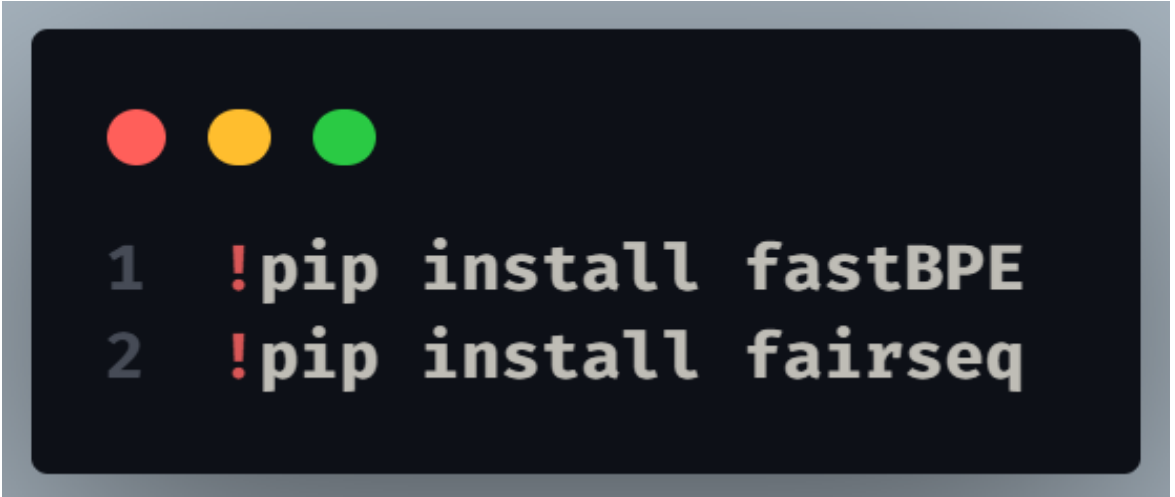
PhoBERT là một mô hình pre-trained được huấn luyện chỉ dành cho tiếng Việt, quá trình huấn luyện tương tự như RoBERTa tối ưu hóa quá trình đào tạo của BERT để đạt được hiệu suất cao hơn.

Trước khi huấn luyện ta cần cài đặt các thư viện và model cần thiết:



```
1 !git clone https://github.com/huggingface/transformers
2 !cd transformers
3 !pip install .
```

Hình 37: Cài đặt transformers



```
1 !pip install fastBPE
2 !pip install fairseq
```

Hình 38: Cài đặt fastBPE và fairseq

```
1 !wget https://public.vinai.io/PhoBERT_base_fairseq.tar.gz
2 !tar -xzf PhoBERT_base_fairseq.tar.gz
3 !wget https://public.vinai.io/PhoBERT_base_transformers.tar.gz
4 !tar -xzf PhoBERT_base_transformers.tar.gz
```

Hình 39: Cài đặt mô hình PhoBERT

Tải mô hình PhoBERT, bpe và tải từ điển từ mô hình PhoBERT:

```
1 from fairseq.data.encoders.fastbpe import fastBPE
2 from fairseq.data import Dictionary
3 import argparse
4
5 parser = argparse.ArgumentParser()
6 parser.add_argument('--bpe-codes',
7                     default="/kaggle/working/PhoBERT_base_transformers/bpe.codes",
8                     required=False,
9                     type=str,
10                    help='path to fastBPE BPE'
11 )
12 args, unknown = parser.parse_known_args()
13 bpe = fastBPE(args)
14
15 # Load the dictionary
16 vocab = Dictionary()
17 vocab.add_from_file("/kaggle/working/PhoBERT_base_transformers/dict.txt")
```

Hình 40: Tải mô hình bpe và từ điển

Do nhãn của các lĩnh vực ở dạng chuỗi, nên chúng ta cần chuyển các nhãn đó thành các số để đưa vào mô hình PhoBERT. Chúng ta sẽ sử dụng **Labels Encoder**.

```
1 from sklearn.preprocessing import LabelEncoder
2 lb = LabelEncoder()
3 lb.fit(train_data['label'])
4 train_labels_lb = lb.fit_transform(train_data['label'])
5 test_labels_lb = lb.fit_transform(test_data['label'])
6 print(lb.classes_)
7 print('Top 5 classes indices: ', train_labels_lb)
8 print('Top 5 classes indices: ', test_labels_lb)
```

Hình 41: Encode nhãn

Tiếp theo, từ dữ liệu thô này, chúng ta sử dụng **bpe** đã load ở trên để đưa text đầu vào dưới dạng subword và ánh xạ các subword này về dạng index trong từ điển[11]:

```
1 from tensorflow.keras.preprocessing.sequence import pad_sequences
2 MAX_LEN = 256
3
4 train_ids = []
5 for sent in train_sents:
6     subwords = '<s> ' + bpe.encode(sent) + ' </s>'
7     encoded_sent = vocab.encode_line(subwords, append_eos=True, add_if_not_exist=False).long().tolist()
8     train_ids.append(encoded_sent)
9
10 val_ids = []
11 for sent in val_sents:
12     subwords = '<s> ' + bpe.encode(sent) + ' </s>'
13     encoded_sent = vocab.encode_line(subwords, append_eos=True, add_if_not_exist=False).long().tolist()
14     val_ids.append(encoded_sent)
15
16 train_ids = pad_sequences(train_ids, maxlen=MAX_LEN, dtype="long", value=0, truncating="post", padding="post")
17 val_ids = pad_sequences(val_ids, maxlen=MAX_LEN, dtype="long", value=0, truncating="post", padding="post")
```

Hình 42: Ánh xạ subword

Bây giờ, **train_ids** và **test_ids** trở thành một danh sách các id của các subwords trong từ điển. Chuẩn hóa câu về cùng một độ dài, những câu có độ dài lớn hơn 256 sẽ được cắt bớt, còn những câu ngắn hơn 256 sẽ thêm padding ở cuối câu.

Chúng ta sẽ tạo một danh sách một mask gồm các giá trị 0, 1 các trị này cho biết rằng mẫu tin đó đã được padding hay chưa. Kiểm tra cho tập train và val.

```
1 train_masks = []
2 for sent in train_ids:
3     mask = [int(token_id > 0) for token_id in sent]
4     train_masks.append(mask)
5
6 val_masks = []
7 for sent in val_ids:
8     mask = [int(token_id > 0) for token_id in sent]
9     val_masks.append(mask)
```

Hình 43: Kiểm tra padding của mẫu tin

Dữ liệu đầu vào cần chuyển về tensor và tạo dataloader bằng cách sử dụng DataLoader của torch với batch-size là 32.

```
1 from torch.utils.data import TensorDataset, DataLoader, RandomSampler, SequentialSampler
2 import torch
3
4 train_inputs = torch.tensor(train_ids)
5 val_inputs = torch.tensor(val_ids)
6 train_labels = torch.tensor(train_labels)
7 val_labels = torch.tensor(val_labels)
8 train_masks = torch.tensor(train_masks)
9 val_masks = torch.tensor(val_masks)
10
11 train_data = TensorDataset(train_inputs, train_masks, train_labels)
12 train_sampler = SequentialSampler(train_data)
13 train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=32)
14
15 val_data = TensorDataset(val_inputs, val_masks, val_labels)
16 val_sampler = SequentialSampler(val_data)
17 val_dataloader = DataLoader(val_data, sampler=val_sampler, batch_size=32)
```

Hình 44: Tạo DataLoader

Tải model PhoBERT để tiến hành huấn luyện:

```
1 from transformers import RobertaForSequenceClassification, RobertaConfig, AdamW
2
3 config = RobertaConfig.from_pretrained(
4     "/kaggle/working/PhoBERT_base_transformers/config.json", from_tf=False, num_labels = 10, output_hidden_states=False,
5 )
6 BERT_SA = RobertaForSequenceClassification.from_pretrained(
7     "/kaggle/working/PhoBERT_base_transformers/model.bin",
8     config=config
9 )
10 BERT_SA.cuda()
```

Hình 45: Tải mô hình PhoBERT

Định nghĩa hàm huấn luyện và đánh giá từ mô hình PhoBERT:

```
1 def flat_accuracy(logits, labels):
2     preds = []
3     for line in logits:
4         preds.append(torch.argmax(line).cpu().numpy())
5     return accuracy_score(preds, labels.cpu().numpy())
```

Hình 46: Hàm đánh giá

```
1 def train_model(model):
2     model.train()
3     train_loss = 0
4     acc = []
5     for batch in train_dataloader:
6         b_input_ids = batch[0].to(device)
7         b_input_mask = batch[1].to(device)
8         b_labels = batch[2].to(device)
9         BERT_SA.zero_grad()
10        outputs = model(b_input_ids,
11                        token_type_ids=None,
12                        attention_mask=b_input_mask,
13                        labels=b_labels)
14
15        loss = outputs[0]
16        logits = outputs[1]
17        acc.append(flat_accuracy(logits, b_labels))
18        loss.backward()
19        torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
20        optimizer.step()
21        train_loss += loss.item()
22
23    return train_loss, round(np.mean(acc), 3)
```

Hình 47: Hàm huấn luyện

Bắt đầu huấn luyện mô hình trên **10 epochs** với device cuda:

```
1 import random
2 from tqdm import tqdm_notebook
3 device = 'cuda'
4 epochs = 10
5 import time
6
7 param_optimizer = list(BERT_SA.named_parameters())
8 no_decay = ['bias', 'LayerNorm.bias', 'LayerNorm.weight']
9 optimizer_grouped_parameters = [
10     {'params': [p for n, p in param_optimizer if not any(nd in n for nd in no_decay)], 'weight_decay': 0.01},
11     {'params': [p for n, p in param_optimizer if any(nd in n for nd in no_decay)], 'weight_decay': 0.0}
12 ]
13
14 optimizer = AdamW(optimizer_grouped_parameters, lr=1e-5, correct_bias=False)
15 train_loss_ = []
16 test_loss_ = []
17
18 print("Training start!")
19 for epoch in range(0, epochs):
20     begin_time = time.time()
21     loss, acc = train_model(BERT_SA)
22     done_time = time.time() - begin_time
23     print(f"Epoch {epoch + 1}: {round(done_time)} seconds - loss {round(loss, 3)} - accuracy {acc}")
24     train_loss_.append([loss, acc])
25
26 print("Training complete!")
```

Hình 48: Huấn luyện mô hình PhoBERT

```
Training start!
Epoch 1: 825 seconds - loss 424.347 - accuracy 0.887
Epoch 2: 825 seconds - loss 223.691 - accuracy 0.937
Epoch 3: 826 seconds - loss 164.774 - accuracy 0.955
Epoch 4: 825 seconds - loss 121.492 - accuracy 0.968
Epoch 5: 824 seconds - loss 94.633 - accuracy 0.977
Epoch 6: 825 seconds - loss 74.207 - accuracy 0.983
Epoch 7: 826 seconds - loss 59.443 - accuracy 0.987
Epoch 8: 826 seconds - loss 49.499 - accuracy 0.99
Epoch 9: 825 seconds - loss 41.559 - accuracy 0.991
Epoch 10: 825 seconds - loss 36.284 - accuracy 0.992
Training complete!
```

Hình 49: Kết quả sau khi huấn luyện trên 10 epochs

Sau khi huấn luyện mô hình chúng ta tiến hành dự đoán và tính độ chính xác:

```
1 preds = []
2 for batch in val_dataloader:
3     b_input_ids = batch[0].to(device)
4     b_input_mask = batch[1].to(device)
5     with torch.no_grad():
6         pred = BERT_SA(b_input_ids, token_type_ids=None, attention_mask=b_input_mask)
7         preds.append(pred[0])
8
9 lab = []
10 for line in preds:
11     for i in line:
12         predicts = i.to('cpu').detach().numpy().copy()
13         lab.append(np.argmax(predicts))
14
15 print(f'Accuracy BERT is {accuracy_score(val_labels.tolist(), lab)}')
```

Hình 50: Dự đoán và đánh giá mô hình PhoBERT

Xóa cache sau khi huấn luyện mô hình PhoBERT:

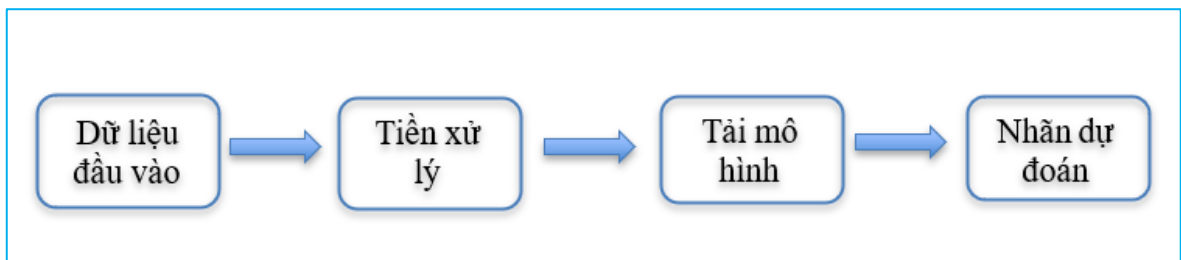
```
1 import gc
2
3 gc.collect()
4 torch.cuda.empty_cache()
5 del BERT_SA
```

Hình 51: Xóa cache cuda

2.5 Xây dựng website dự đoán


Để thực hiện việc dự đoán các tin tức mới đến, chúng tôi xây dựng website dự đoán tin tức với 10 lĩnh vực. Mô hình sau khi huấn luyện sẽ được lưu lại và sử dụng cho quá trình dự đoán. Website được xây dựng gồm 2 phần là client và server.

Phần client chúng tôi sử dụng kiến thức HTML, CSS, JS và thư viện ReactJS để xây dựng giao diện người dùng. Còn phần server chúng tôi sử dụng ngôn ngữ python với thư viện flask để xây dựng API dự đoán.




Hình 52 Quá trình dự nhận của website

Đầu tiên người dùng sẽ nhập dữ liệu tin tức vào ô textarea và chọn một mô hình có sẵn, sau submit dữ liệu tin tức sẽ được tiền xử lý giống như khi huấn luyện mô hình, tiếp đến tải mô hình được người dùng chọn và đưa vào dự đoán. Mô hình sẽ thực hiện quá trình dự đoán và trả về một lĩnh vực(nhãn) tương ứng với tin tức đã nhập. Dữ liệu trả về từ mô hình sẽ được client nhận và render ra giao diện người dùng để biết kết quả.



```
1  # get data
2  data = json.loads(request.data)
3  input = data['input']
4  model_choose = data['model']
5
6  # preprocess input
7  pre = Preprocess()
8  input_pre = pre.fit(input)
9
10 # load tfidf
11 tfidf = Models.load_tfidf()
12 input_tf = tfidf.transform([input_pre])
```

Hình 53: Lấy dữ liệu từ client và tiền xử lý



```
1  # predict by model
2  if model_choose == 'svm':
3      model = Models.load_svm_model()
4  elif model_choose == 'decision_tree':
5      model = Models.load_decision_tree()
6  elif model_choose == 'multinomiaNB':
7      model = Models.load_multinomiaNB()
8  else:
9      model = Models.load_logistic_regression()
10
11 # prediction
12 prediction = model.predict(input_tf)
13 output = ''.join(prediction)
14
15 return {
16     "result": output
17 }
```

Hình 54: Tải mô hình, dự đoán và trả về kết quả

Website dự đoán tin tức được xây dựng với giao diện đơn giản, thân thiện với người dùng và dễ sử dụng. Website được đưa lên Internet trên nền tảng heroku, đường dẫn website đánh giá tại <http://pred-news-topic.herokuapp.com/>

Đề tài: Xây dựng ứng dụng phân loại văn bản của 10 lĩnh vực

The screenshot displays the 'News Topic Predictor' web application. The title 'News Topic Predictor' is centered at the top in a large blue font, with the subtitle 'Data Mining Project' in a smaller grey font below it. The interface is divided into two main white panels on a light grey background. The left panel, titled 'News Input', contains a large text area with the placeholder 'Enter your news...'. The right panel contains several sections: 'Labels' with a 2x4 grid of topic names (Chính Trị Xã Hội, Đời Sống, Khoa Học, Kinh Doanh, Pháp Luật, Sức Khỏe, Thể Giới, Thể Thao, Văn Hóa, and Vì Tinh); 'Choose A Model' with four radio button options (Decision Tree is selected, Logistic Regression, Naive Bayes, and SVM); 'Actions' with 'PREDICT' and 'CLEAN' buttons; and 'Result' which shows 'Model is Decision Tree' and 'This news is ... Topic'. A small blue square button with an upward arrow is located in the bottom right corner of the interface.

Hình 55: Giao diện website dự đoán

CHƯƠNG 3 - KIỂM THỬ VÀ ĐÁNH GIÁ

1. Mục tiêu

Kiểm thử là một trong những công việc quan trọng để đánh giá chất lượng của hệ thống. Khi đánh giá về chất lượng của một hệ thống, yêu cầu về chức năng, khả năng sử dụng, độ hiệu quả là những tiêu chí cần phải đảm bảo.

Trong thực tế, chúng ta cần áp dụng nhiều giải thuật máy học để chọn ra mô hình phù hợp nhất cho bài toán của đặt ra. Thì với vấn đề trên câu hỏi đặt ra là mô hình này có tốt không? Và có tốt hơn mô hình khác không? Ngoài thuật toán máy học, độ chính xác của mô hình còn phụ thuộc vào các yếu tố như sự phân bố của các lớp, kích thước của tập huấn luyện, chúng tôi sẽ đánh dựa trên thời gian và độ chính xác của mô hình dự đoán với tập kiểm tra có sẵn.

Yêu cầu chức năng: các chức năng thích hợp trong hệ thống phải đáp ứng những yêu cầu đã được xác định, bao gồm cả tính phù hợp, chính xác và khả năng tương tác.

Một mô hình tốt sẽ cho ra kết quả chính xác khi dự đoán kết quả với dữ liệu mới. Nên việc đánh giá mô hình là một bước rất quan trọng để có thể xác định mô hình đó có phù hợp với bài toán đặt ra hay không? Tất nhiên không có mô hình nào là tốt nhất với tất cả các hoàn cảnh, nó phụ thuộc vào đặc trưng của mô hình, đặc trưng của dữ liệu, nên việc huấn luyện và dự đoán dữ liệu trên nhiều loại mô hình khác nhau là việc cần thiết.

Yêu cầu về khả năng sử dụng: ứng dụng tạo ra cần dễ dàng sử dụng, hiệu quả.

2. Các thông số đánh giá giải thuật

Để kiểm tra một mô hình có hiệu quả đạt độ chính xác và để so sánh khả năng học của các mô hình thì cần có một thông số để thực hiện việc đánh giá. Trong các bài toán phân lớp thì việc định nghĩa lớp dữ liệu quan trọng hơn cần được xác định

đúng là lớp dương (Positive), lớp còn lại được gọi là âm (Negative). Giả sử, để đánh giá một bộ phân loại hai lớp tạm gọi là (+) và (-), khi đó:

Predict/Actual		Actual	
		(+)(Positive)	(-)(Negative)
Predict	(+)(Positive)	TP - True Positive	FP - False Positive
	(-)(Negative)	FN - False Negative	TN - True Negative

Bảng 2: Chỉ số đánh giá giải thuật học

Trong đó:

- **TP (True positive)** là số phần tử dương được phân loại dương
- **FN (False negative)** là số phần tử dương được phân loại âm
- **TN (True negative)** là số phần tử âm được phân loại âm
- **FP (False positive)** là số phần tử âm được phân loại dương

Độ chính xác (Precision) được định nghĩa là tỷ lệ số phần tử TP trong số những phần tử được phân loại là positive (TP + FP):

$$Precision = \frac{TP}{TP + FP}$$

Độ bao phủ (Recall) được định nghĩa là tỷ lệ số phần tử TP trong số những phần tử thực sự là positive (TP + FN):

$$Recall = \frac{TP}{TP + FN}$$

Precision cao đồng nghĩa với việc độ chính xác của các phần tử tìm được là cao. Recall cao đồng nghĩa với việc tỷ lệ bỏ sót các phần tử thực sự positive là thấp.

Độ đo F_1 : chỉ số cân bằng giữa độ chính xác và độ bao phủ. Nếu độ chính xác và độ bao phủ cao và cân bằng thì độ đo F_1 lớn, còn độ chính xác và độ bao phủ nhỏ, không cân bằng thì độ đo F_1 nhỏ. Như vậy, F_1 càng cao thì bộ phân lớp càng tốt. Khi cả Recall và Precision đều bằng 1 (tốt nhất có thể) thì $F_1 = 1$. Khi cả Recall và Precision đều thấp thì phân lớp không tốt.

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Độ chính xác (Accuracy): là tỷ lệ các mẫu tin được dự báo đúng trên tổng số các mẫu tin là đúng. Độ chính xác giúp ta đánh giá hiệu quả dự báo của mô hình trên một bộ dữ liệu. Độ chính xác càng cao thì mô hình của chúng ta càng chuẩn xác.

$$Accuracy = \frac{TP + TN}{total\ sample}$$

3. Đánh giá

Sau khi huấn luyện mô hình và dự đoán thì ta thấy giải thuật SVM, Logistic Regression và PhoBERT có độ chính xác cao hơn giải thuật Naive Bayes và Decision Tree. Ta có thể thấy mô hình Naive Bayes và Decision Tree có khả năng học thấp hơn các mô hình còn lại.

Một số phân lớp như The Gioi, Van Hoa có độ chính xác khá thấp so với các phân lớp còn lại. Có thể các nội dung của các phân lớp này không rõ ràng, dễ nhầm lẫn với phân lớp khác.

Có thể tăng số lượng các phân lớp có độ chính xác thấp để tăng khả năng dự đoán của mô hình với phân lớp.

Các giải thuật được thực hiện trong đề tài có độ chính xác khá cao, độ chính xác các giải thuật và thời gian huấn luyện được mô tả dưới đây:

STT	PHƯƠNG PHÁP	THỜI GIAN HUẤN LUYỆN(S)	ĐỘ CHÍNH XÁC(%)
1	Decision Tree	58.424	72.3
2	Logistic Regression	24.88	92.2
3	Naive Bayes	0.23	81.3
4	PhoBERT	8252	90.5
5	SVM	655.274	92.1

Bảng 3: Thông tin đánh giá từng giải thuật học

Với giải thuật PhoBERT chúng tôi sử dụng phương pháp tách từ được gợi ý từ trang chủ của PhoBERT là `vncorenlp`, còn các giải thuật còn lại chúng tôi sử dụng bằng thư viện `pyvi`.

Trong các giải thuật được sử dụng, Logistic Regression có thời gian huấn luyện nhanh chóng và độ chính xác cao nhất, giải thuật SVM có độ chính xác cũng rất cao chính thấp hơn Logistic Regression không đáng kể. Giải thuật Naive Bayes có thời gian huấn luyện ngắn nhất nhưng độ chính xác không cao lắm. Giải thuật Decision Tree là giải thuật có độ chính xác thấp nhất trong các giải thuật nên trên, mô hình có độ chính xác rất cao với tập huấn luyện nhưng đạt kết quả thấp với tập kiểm tra, mô hình bị *overfitting*. PhoBERT là phương pháp DeepLearning nên thời gian huấn luyện khá lâu, vì phải huấn luyện qua 10 epochs, trung bình mỗi epoch huấn luyện khoảng 852s.

PHẦN KẾT LUẬN

1. Kết quả đạt được

Đề tài sử dụng 2 phương pháp tách từ bởi vncorenlp và pyvi giải quyết vấn đề tách từ tiếng Việt trong bài toán phân loại, áp dụng các giải thuật Decision Tree, Logistic Regression, Naive Bayes, SVM và PhoBERT. Sau khi dự đoán và đánh giá thì mô hình Logistic Regression, SVM, PhoBERT có độ chính xác trên **90%** với tập kiểm tra. Các giải thuật còn lại đạt độ chính xác tương đối và thấp hơn 3 mô hình nêu trên.

Dựa vào các mô hình đã được huấn luyện, xây dựng trang web dự đoán tin tức mới đến theo 10 lĩnh vực. Trong đó, đầu vào là một tin tức tiếng Việt thô chưa được xử lý, dữ liệu đầu vào được xử lý và đưa vào mô hình dự đoán rồi gán nhãn.

Trang web được xây dựng có giao diện khá bắt mắt, dễ sử dụng và tăng trải nghiệm người dùng. Trang web cung cấp các lựa chọn về mô hình để dự đoán, với mỗi mô hình có thể dự đoán nhãn khác nhau, do khả năng học của mô hình.

2. Hạn chế và hướng phát triển

2.1 Hạn chế

Trong quá trình phát triển và triển khai đề tài có một số hạn chế như:

- Một số mô hình học có độ chính xác chưa cao, dẫn đến việc dự đoán tin tức mới đến còn chưa chính xác.
- Mô hình Decision Tree có độ chính xác cao trên tập huấn luyện, nhưng đạt kết quả khá thấp với tập kiểm tra, đó là overfitting.
- Thời gian dự đoán nhãn trên website còn chậm.

2.2 Hướng phát triển

Sau khi thực hiện xong đề tài chúng tôi thấy có các hướng phát triển sau:

- Đưa mô hình PhoBERT lên production.

- Thêm tính năng xác nhận nhãn sau khi dự đoán, nếu nhãn dự đoán và nhãn thực tế không chính xác thì có thể thêm vào tập dữ liệu để bổ xung dữ liệu.
- Thu thập thêm dữ liệu của các phân lớp ít dữ liệu hơn.

TÀI LIỆU THAM KHẢO

- [1] W3 Schools, <https://www.w3schools.com/>.
- [2] Trần Thanh Điện, Thái Nhựt Thanh, Nguyễn Thái Nghe, Giải pháp phân loại bài báo khoa học bằng kỹ thuật máy học, Tạp chí khoa học - Trường Đại học Cần Thơ
- [3] PGS.TS.Đỗ Thanh Nghị, Giáo trình môn Lập trình web, Khoa Công nghệ thông tin & truyền thông - Trường Đại học Cần Thơ.
- [4] Freetuts, <https://freetuts.net/>
- [5] Khoa học dữ liệu - Khanh's blog, Thực hành ứng dụng BERT, https://phamdinhkhanh.github.io/2020/06/04/PhoBERT_Fairseq.html
- [6] Lập Trình Không Khó, <https://nguyenvanhieu.vn/tf-idf-la-gi/>
- [7] Trí tuệ nhân tạo, <https://trituenhantao.io/kien-thuc/decision-tree/>
- [8] Towards Data Science, <https://towardsdatascience.com/the-perfect-recipe-for-classification-using-logistic-regression-f8648e267592>
- [9] Edureka blog, Exploratory Data Analysis In Python <https://www.edureka.co/blog/exploratory-data-analysis-in-python/>
- [10] Nguyễn Văn Hiếu, Phân loại văn bản tiếng Việt sử dụng machine learning, <https://nguyenvanhieu.vn/phan-loai-van-ban-tieng-viet/>
- [11] Viblo, BERT, RoBERTa, PhoBERT, BERTweet: Ứng dụng state-of-the-art pre-trained model cho bài toán phân loại văn bản, <https://viblo.asia/p/bert-roberta-phobert-bertweet-ung-dung-state-of-the-art-pre-trained-model-cho-bai-toan-phan-loai-van-ban-4P856PEWZY3>