ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC VÀ KĨ THUẬT THÔNG TIN

NGUYỄN ĐỨC HUY LONG – 18521034

TIÊU TỰ ĐẠT - 18520589

THÁI VĨNH ĐỨC - 18520623

NGUYỄN TUẤN KHA – 18520873

ĐỒ ÁN MÔN HỌC

MÔN HỌC: HỌC MÁY THỐNG KÊ

LỚP: DS102.L11.CNCL

PHÂN LOẠI CHỦ ĐỀ VĂN BẢN

TOPIC CLASSIFICATION

TP. HỒ CHÍ MINH, 2020

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC VÀ KĨ THUẬT THÔNG TIN

NGUYỄN ĐỨC HUY LONG – 18521034

TIÊU TỰ ĐẠT - 18520589

THÁI VĨNH ĐỨC - 18520623

NGUYỄN TUẤN KHA – 18520873

ĐỒ ÁN MÔN HỌC

MÔN HỌC: HỌC MÁY THỐNG KÊ

LỚP: DS102.L11.CNCL

PHÂN LOẠI CHỦ ĐỀ VĂN BẢN

Topic Classification

GIẢNG VIÊN HƯỚNG DẪN

TS Nguyễn Tấn Trần Minh Khang

Th.S Võ Duy Nguyên

TP. HỒ CHÍ MINH, 2020

LỜI CẢM ƠN

Đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến tập thể quý cô thầy Trường Đại học Công nghệ thông tin – Đại học Quốc gia TP.HCM đã giúp cho chúng em có những kiến thức cơ bản làm nền tảng để thực hiện nghiên cứu này.

Đặc biệt, cả nhóm xin gửi lời cảm ơn và lòng biết ơn sâu sắc nhất tới TS. Nguyễn Tấn Trần Minh Khang và ThS. Võ Duy Nguyên đã trực tiếp hướng dẫn, tận tình chỉ bảo, đã tạo điều kiện về cơ sở vật chất, kỹ thuật, đóng góp nhiều ý kiến quý báu giúp nhóm chúng em hoàn thành tốt đồ án.

Trong thời gian thực hiện đồ án, cả nhóm đã vận dụng những kiến thức nền tảng đã tích lũy đồng thời kết hợp với việc học hỏi và nghiên cứu những kiến thức mới. Từ đó, nhóm vận dụng tối đa những gì đã thu thập được để hoàn thành một báo cáo đồ án tốt nhất. Tuy nhiên, trong quá trình thực hiện, nhóm không tránh khỏi những thiếu sót. Chính vì vậy, nhóm rất mong nhận được những sự góp ý từ phía các thầy nhằm hoàn thiện những kiến thức mà nhóm đã học tập và là hành trang để nhóm chúng em thực hiện tiếp các đề tài khác trong tương lai.

Xin chân thành cảm ơn thầy.

Mục lục

Danh sách hình vẽ

Danh mục từ viết tắt

CNN	**C**onvolutional **N**eural **N**etwork
SPANet	**SP**atial **A**ttentive **N**etwork
IRNN	Identity Matrix Initialization
RB	**R**esidual **B**locks
SAB	**S**patial **A**ttentive **B**lock
SARB	**S**patial **A**ttentive **R**esidual **B**lock
SAM	**S**patial **A**ttentive **M**odule
MSE	**M**ean - **S**quare **E**rror
PSNR	**P**eak **S**ignal-to- **N**oise **R**atio
SSIM	**S**tructural **S**imilarity **I**ndex **M**easure
TP	**T**rue **P**ositive
FP	**F**alse **P**ositive
TN	**T**rue **N**egative
FN	**F**alse **N**egative
KNN	**K**-**N**earest **N**eighbors
SVM	**S**upport **V**ector **M**achine

TÓM TẮT KHOÁ LUẬN

Trong đồ án chuyên ngành này, chúng em tập trung nghiên cứu các phương pháp học máy đã học [1] và một phương pháp học sâu LSTM [2] cho việc giải quyết bài toán Topic Classification (Phân loại chủ đề văn bản).

Mục tiêu của đồ án là nghiên cứu về ý tưởng, kỹ thuật cốt lõi cũng như thực nghiệm nó trên bộ dữ liệu đã được nhận. Qua đó chúng em đánh giá kết quả thực nghiệm, phân tích thách thức và đề xuất hướng giải quyết trong tương lai.

Chương 1

Tổng quan

1.1 Giới thiệu đề tài

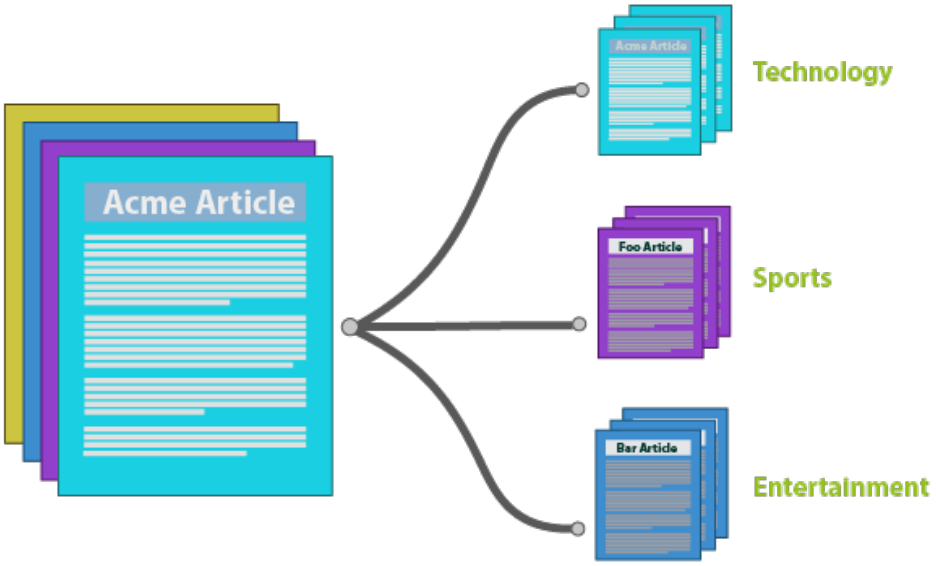
1.1.1 Tổng quan

Ngày nay, kiến thức con người ngày càng nhiều dẫn đến số lượng tài liệu, sách báo, văn bản trở nên gần như vô hạn. Điều đó đã khiến cho việc tìm kiếm trở nên khó khăn, vì thế chúng ta cần phải phân loại các văn bản, tài liệu, sách báo trên nhiều phương diện để có thể dễ dàng tìm kiếm một cách nhanh chóng. Trong đó, việc phân loại các văn bản trên chủ đề là phổ biến nhất và hiểu quả nhất.

Tuy nhiên, việc phân loại văn bản dựa trên chủ đề cũng có chung một số vấn đề là việc đọc hết nội dung của văn bản tốn khá nhiều thời gian, đồng thời số lượng văn bản quá nhiều để có thể phân loại các văn bản một cách thủ công. Bởi vì lí do đó, mà một bài toán mới đã được đặt ra, đó chính là phân loại chủ đề văn bản tự động.

1.1.2 Giới thiệu bài toán và đối tượng nghiên cứu

Tocpic Classification là bài toán có mục tiêu phân loại các văn bản dựa trên các chủ đề đã được cho sẵn. Bài toán Topic Classification có đầu vào là một văn bản và đầu ra là chủ đề của văn bản đó (Hình 1.1).



HÌNH 1.1: Ảnh minh họa phân loại văn bản

Đối tượng nghiên cứu

Đối tượng được hướng đến là những bài báo chưa được phân loại chủ đề. Nhóm hướng đến việc xây dựng một thuật tám có thể tự động phân loại những bài báo này theo 27 chủ đề vì vậy trong đồ án này các bài báo sử dụng sẽ được chia làm 27 chủ đề chính.

1.2 Mục tiêu

- Hiểu được cách thức giải quyết các bài toán NLP nói chung và bài toán phân loại chủ đề văn bản nói riêng.
- Nắm bắt ý tưởng, kỹ thuật cơ sở của các phương pháp Machine Learning và Deep Learning.
- Cài đặt, thực nghiệm các phương pháp đã được học và các phương pháp Deep Learning đã tìm hiểu .
- Thu thập kết quả thực nghiệm, đánh giá hiệu suất, thách thức của phương pháp, đưa ra đề xuất phát triển.

1.3 Nội dung đồ án chuyên ngành

- Cái nhìn tổng quan về phương pháp học máy và học sâu, tiềm năng ứng dụng trong nghiên cứu và thực nghiệm.
- Tìm hiểu cách giải quyết bài toán phân loại chủ đề văn bản tiếng Việt (topic classification).
- Nghiên cứu về phương pháp Deep Learning (LSTM).
- Qui trình cài đặt, thực nghiệm phương pháp trên bộ dữ liệu tương ứng.
- Kết luận, đánh giá kết quả đạt được, nêu hạn chế, định hướng phát triển.

1.4 Cấu trúc đồ án

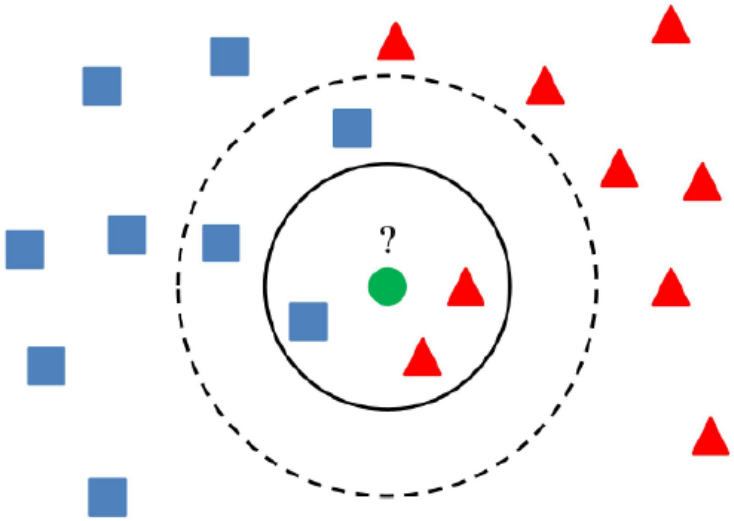
Phần còn lại của đồ án được tổ chức như sau: Ở chương 2 chúng em sẽ trình bày các cơ sở lý thuyết phục vụ cho bài toán. Chương 3 chúng em sẽ trình bày môi trường và cách thức chạy thực nghiệm, kết quả đánh giá hiệu suất và nhận xét. Chương 4 đưa ra kết luận và tổng kết về những gì đạt được, cùng với đó là đề ra các hướng nghiên cứu trong tương lai.

Chương 2

Cơ sở lý thuyết giải quyết bài toán

2.1 Các phương pháp Machine Learning

2.1.1 Phương pháp K-Nearest Neighbors



HÌNH 2.1: Ảnh minh họa KNN

K-nearest neighbor là một trong những thuật toán supervised-learning đơn giản nhất (mà hiệu quả trong một vài trường hợp) trong Machine Learning. Khi training, thuật toán này không học một điều gì từ dữ liệu training (đây cũng là

lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới. K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán Supervised learning là Classification và Regression. KNN còn được gọi là một thuật toán Instance-based hay Memory-based learning.

Với KNN, trong bài toán Classification, label của một điểm dữ liệu mới (hay kết quả của câu hỏi trong bài thi) được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong training set. Label của một test data có thể được quyết định bằng major voting (bầu chọn theo số phiếu) giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho mỗi trong các điểm gần nhất đó rồi suy ra label.

2.1.2 Phương pháp Logistic Regression

Logistic Regression là 1 thuật toán phân loại được dùng để gán các đối tượng cho 1 tập hợp giá trị rời rạc (như 0, 1, 2, ...). Một ví dụ điển hình là phân loại Email, gồm có email công việc, email gia đình, email spam, ... Giao dịch trực tuyến có là an toàn hay không an toàn, khối u lành tính hay ác tình. Thuật toán trên dùng hàm sigmoid logistic để đưa ra đánh giá theo xác suất.

Tuy nhiên trong thực tế dữ liệu của 01 class có thể nằm trong 1 vòng tròn, 1 class khác nằm ngoài vòng tròn, lúc đó Logistic Regression không còn đúng, không áp dụng được.

Learning là quá trình tìm, định lượng các hệ số B0, B1 trong mô hình Logistic Regression từ tập dữ liệu có sẵn (training data set). Trong Machine Learning nói riêng và AI nói chung rất khó có thể năng tìm ra đúng, chính xác tuyệt đối các hệ số này mà chỉ cố gắng định lượng giá trị có khả năng cao nhất gần đúng hay nói cách khác là giảm thiểu tối đa sai sót, chênh lệch khi lắp ghép các hệ số B0, B1 này vào model so với kết quả thực tế.

Đoán trước giá trị của model biểu diễn như sau:

$$p_{(class = 0)} = 1/ (1 - e^{output})$$

HÌNH 2.2: Công thức đoán trước giá trị Logisitic Regression

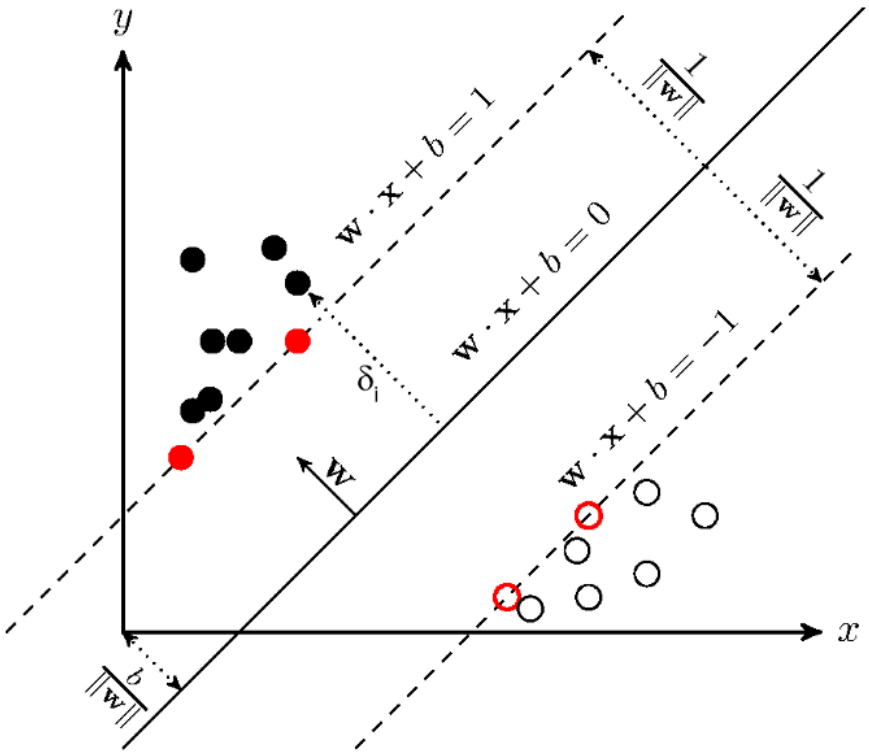
prediction = 0 nếu p(class) < 0.5

prediction = 1 nếu p(class) >= 0.5

2.1.3 Phương pháp Support Vector Machine

Thuật toán máy SVM [3] được Cortes and Vapnik (1995) giới thiệu lần đầu tiên. 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 (Chakrabarti, 2003) do tốc độ phân lớp rất nhanh và hiệu quả đối với bài toán phân lớp văn bản.

Ý tưởng của phương pháp này là cho trước một tập huấn luyện được biểu diễn trong không gian vector, 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 2.3) . 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.



HÌNH 2.3: Siêu phẳng lề cực đại trong không gian hai chiều (Cortes and Vapnik, 1995)

Phương trình mặt siêu phẳng chứa véc-tơ x trong không gian đối tượng như sau: w.x + b = 0.

Trong đó, w là véc-tơ 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: $w.x + b = 1$ và $w.x + b = -1$. 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:

$$\begin{cases} w \cdot x_i + b \geq 1, & \text{với mẫu } (+) \\ w \cdot x_i + b \leq -1, & \text{với mẫu } (-) \end{cases} \tag{2.1}$$

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 $Min\|w\|$ với w và b thỏa điều kiện sau: $\forall i \in (1, n) : y_i\,(w \cdot x_i + b) \geq 1$

2.1.4 Phương pháp Naive Bayes

Thuật toán Naïve Bayes (Mitchell, 1997) là một thuật toán phổ biến trong máy học được McCallum and Nigam (1998) và Yang and Liu (1999) đánh giá là một trong những phương pháp có hiệu năng cao nhất khi thực hiện phân lớp văn bản. Ý tưởng cơ bản của cách tiếp cận này là sử dụng xác suất có điều kiện giữa từ hoặc cụm từ và chủ đề để dự đoán xác suất chủ đề của một tài liệu cần phân loại.

Thuật toán Naïve Bayes [4] dựa trên định lý Bayes được phát biểu như sau:

$$P(Y \mid X) = \frac{P(XY)}{P(X)} = \frac{P(X \mid Y)P(Y)}{P(X)} \tag{2.2}$$

Áp dụng trong bài toán phân loại, các dữ kiện gồm có: D: tập dữ liệu huấn luyện đã được vector hóa dưới dạng $\vec{x}$=($x_1, x_2, ..., x_n$); C_i: phân lớp i, với $i = (1, 2, \ldots, m)$; các thuộc tính độc lập điều kiện đôi một với nhau. Theo định lý Bayes:

$$P\left(C_i \mid X\right) = \frac{P\left(X \mid C_i\right) P\left(C_i\right)}{P(X)} \tag{2.3}$$

Theo tính chất độc lập điều kiện:

$$P\left(X \mid C_i\right) = \prod_{k=1}^{n} P\left(x_k \mid C_i\right) \tag{2.4}$$

Trong đó, $P\left(C_i \mid X\right)$ là xác suất thuộc phân lớp i khi biết trước mẫu X; $P\left(C_i\right)$ xác suất là phân lớp i; $P\left(x_i \mid C_i\right)$ xác suất thuộc tính thứ k có giá trị x_k khi đã biết X thuộc phân lớp i.

Các phân phối thường dùng cho $P\left(x_k \mid C_i\right)$

Phân phối Gaussian Naive Bayes

Phân phối này được sử dụng khi dữ liệu cần được xử lý là các biến liên tục.

$$p\left(x_i \mid c\right) = p\left(x_i \mid \mu_{ci}, \sigma_{ci}^2\right) = \frac{1}{\sqrt{2\pi\sigma_{ci}^2}} \exp\left(-\frac{\left(x_i - \mu_{ci}\right)^2}{2\sigma_{ci}^2}\right) \tag{2.5}$$

Với mỗi chiều dữ liệu i và một class c, x_i tuân theo một phân phối chuẩn có kỳ vọng μ_{ci} và phương sai $\sigma_{c_i}^2$

Phân phối Multinomial Naive Bayes

Phân phối này thường được sử dụng trong các bài toán phân loại văn bản mà ở đó các vector được xử lý bằng Bags of Words. Lúc này, mỗi văn bản được biểu diễn bởi một vector có số chiều d chính là số từ trong từ điển. Giá trị của thành phần thứ i trong mỗi vector chính là số lần từ thứ i xuất hiện trong văn bản đó.

Khi đó $p\left(x_i \mid c\right)$ sẽ được tính theo công thức:

$$\lambda_{ci} = p\left(x_i \mid c\right) = \frac{N_{ci}}{N_c} \tag{2.6}$$

Trong đó:

N_{ci} là tổng số lần từ thứ i xuất hiện trong các văn bản của class c, nó được tính là tổng của tất cả các thành phần thứ i của các feature vectors ứng với class c.

N_c là tổng số từ (kể cả lặp) xuất hiện trong class c. Nói cách khác, nó bằng tổng độ dài của toàn bộ các văn bản thuộc vào class c.

Phân phối Bernoulli Naive Bayes

Phân phối này áp dụng cho các tập dữ liệu mà mỗi thành phần là một giá trị binary (0 hoặc 1). Ví dụ: cũng với loại văn bản nhưng thay vì đếm tổng số lần xuất hiện của 1 từ trong văn bản, ta chỉ cần quan tâm từ đó có xuất hiện hay không.

Khi đó xác suất $p\left(x_i \mid c\right)$ được hiểu là xác suất từ thứ i xuất hiện trong các văn bản của class c. Được tính theo công thức:

$$p\left(x_i \mid c\right) = p(i \mid c)^{x_i} \left(1 - p(i \mid c)^{1-x_i} \tag{2.7}$$

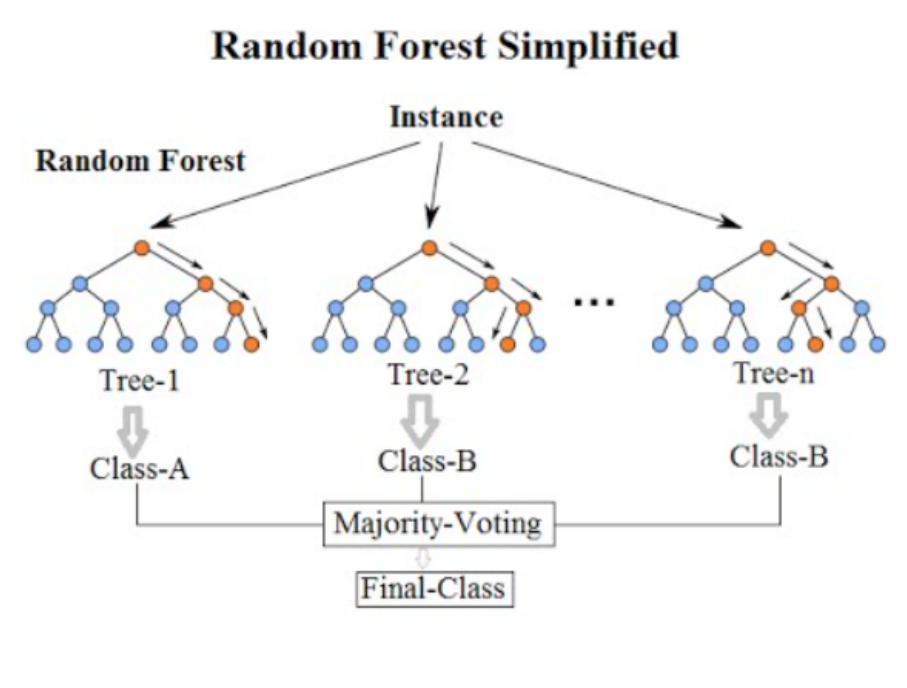
Ngoài 3 phân phối thường gặp được nêu ở trên Naive Bayes còn sử dụng hai phân phối khác là: Complement và Categorical.

Trong bài đồ án này do yêu cầu là phân loại văn bản nên nhóm quyết định sử dụng 3 phân phối thường dùng cho bài toán phân loại văn bản trong tổng số 5 phân phối của Naive Baye bao gồm: Multinomial, Complement, Bernoulli.

2.1.5 Phương pháp Random Forest Classification

Random Forests là thuật toán học có giám sát (supervised learning). Nó có thể được sử dụng cho cả phân lớp và hồi quy. Nó cũng là thuật toán linh hoạt và dễ sử dụng nhất. Một khu rừng bao gồm cây cối. Người ta nói rằng càng có nhiều cây thì rừng càng mạnh. Random forests tạo ra cây quyết định trên các mẫu dữ liệu được chọn ngẫu nhiên, được dự đoán từ mỗi cây và chọn giải pháp tốt nhất bằng cách bỏ phiếu. Nó cũng cung cấp một chỉ báo khá tốt về tầm quan trọng của tính năng. Random forests có nhiều ứng dụng, chẳng hạn như công cụ đề xuất, phân loại hình ảnh và lựa chọn tính năng. Nó có thể được sử dụng để phân loại các ứng viên cho vay trung thành, xác định hoạt động gian lận và dự đoán các

bệnh. Nó nằm ở cơ sở của thuật toán Boruta, chọn các tính năng quan trọng trong tập dữ liệu.



HÌNH 2.4: Random Forest Regression

Random Forest algorithm có thể sử dụng cho cả bài toán Classification và Regression Random Forest làm việc được với dữ liệu thiếu giá trị Khi Forest có nhiều cây hơn, chúng ta có thể tránh được việc Overfitting với tập dữ liệu Có thể tạo mô hình cho các giá trị phân loại

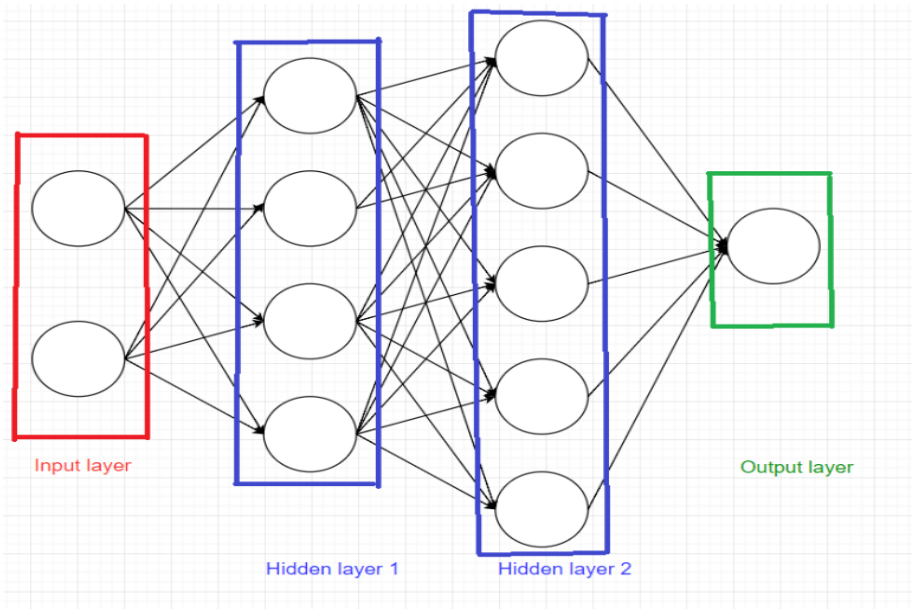
Nếu coi mỗi ý kiến của những người góp ý là một cây quyết định, thì chúng ta đã có hình dung mơ hồ về Random Forest.

Random Forest hoạt động bằng cách đánh giá nhiều Cây quyết định ngẫu nhiên, và lấy ra kết quả được đánh giá tốt nhất trong số kết quả trả về.

2.2 Phương pháp Deep Learning trong phân loại chủ đề văn bản

2.2.1 Neural Network

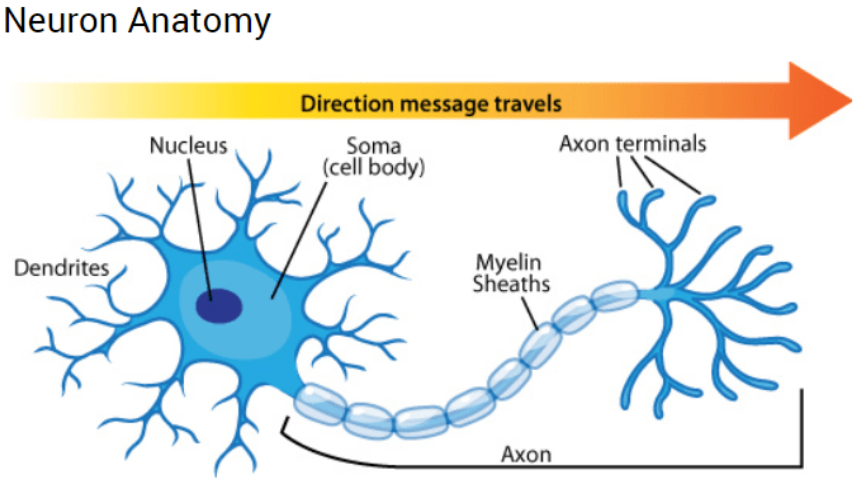
Neural Network là một phát minh quan trọng, góp phần thúc đẩy sự phát triển vượt bậc của công nghệ đặc biệt trong các lĩnh vực về khoa học máy tính. Về cơ bản, mạng neural dùng để rút trích đặc trưng mô hình chung của một bộ dữ liệu đầu vào định sẵn, dựa vào đó các nhà nghiên cứu có thể sử dụng để phân tích thông tin, phát triển các giải pháp giải quyết các vấn đề khác nhau. Kiến trúc căn bản của một mạng neural network là tổ hợp của nhiều lớp (layer) được phân ra ba nhóm chính: lớp đầu vào (Input layer), các lớp ẩn (Hidden layers), lớp đầu ra (Output layer). Hình 2.5 cho thấy, một lớp riêng biệt có nhiều nút điểm, đây là nơi diễn ra các tác vụ chính của lớp. Các nút điểm được nối với nhau thông qua các liên kết sao cho dữ liệu đầu ra của lớp này là đầu vào của lớp kế tiếp, dữ liệu sẽ tiếp tục được xử lý để thu được kết quả mong muốn ở lớp cuối cùng.



HÌNH 2.5: Minh họa mạng Neural Network

2.2.2 Neural nhân tạo (perceptron)

Hình thức cơ bản của mạng neural nhân tạo được đưa ra bởi Frank Rosenblatt tại phòng thí nghiệm Corell Aeronautical vào những năm 1950. Ông đã đưa ra mô hình bằng cách mô phỏng bộ não con người. Bộ não con người là tập hợp các tế bào thần kinh được cấu thành từ phần thân (soma) với nhân ở bên trong và xung quanh có nhiều sợi nhánh với một sợi trục chính, các tế bào thần kinh kết nối với nhau thông qua sợi nhánh và sợi trục. Các xung động thần kinh được tiếp nhận từ sợi nhánh sẽ được gửi vào phần thân xử lý, xung động lan truyền qua sợi trục, nếu chúng vượt ngưỡng, xung thần kinh được lan truyền cho các tế bào thần kinh khác (Hình 2.6).

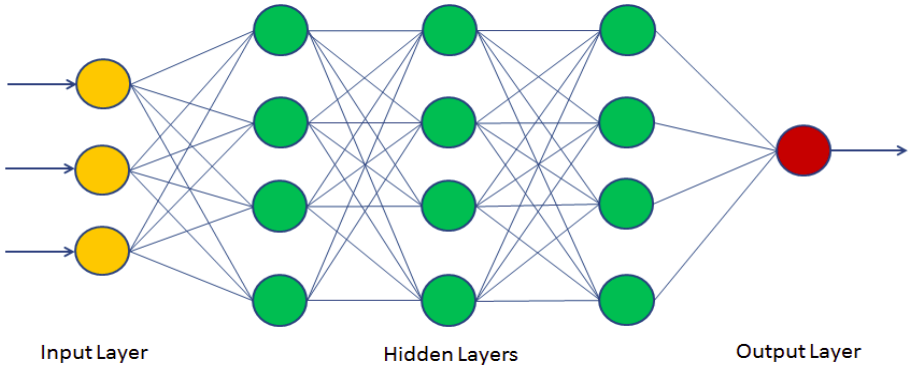


HÌNH 2.6: Tế bào thần kinh sinh học

Dựa trên cấu trúc và nguyên lý lan truyền xung thần kinh của bộ não người, neural nhân tạo lấy các giá trị đầu vào $x_1, x_2, ..., x_n$ từ các neural khác, cùng với $w_1, w_2, ..., w_n$ đại diện cho tín hiệu đầu ra. Thay thế cho ngưỡng lan truyền xung thần kinh chính là tập các hàm kích hoạt (activation).

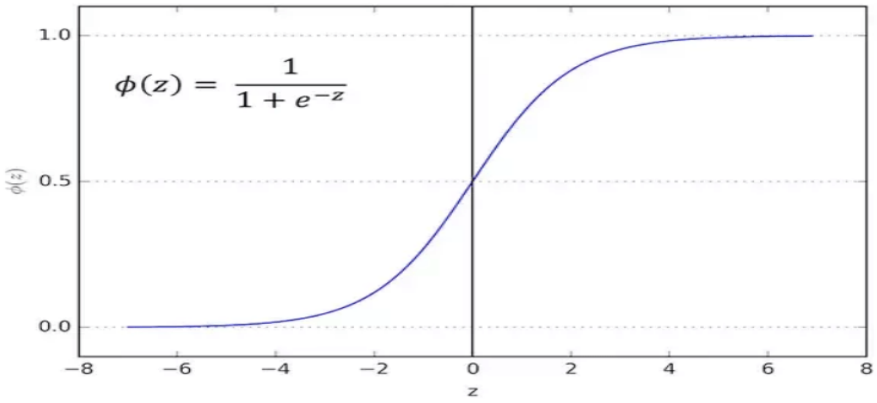
2.2.3 Mạng neural đa lớp

Xét đến việc giải quyết một vấn đề phức tạp, một vài neural nhân tạo hiển nhiên là chưa đủ, do đó cần phải tổ chức lại các neural thành một cấu trúc mạng thống nhất với quy mô lớn về số lượng neural cũng như là kết cấu phức tạp hơn. Hình 2.7 mô tả kiến trúc mạng điển hình, với một lớp đầu vào theo sau đó là tập n lớp ẩn liên tục nhận tín hiệu từ lớp trước nó liên kết để tạo thông tin trừu tượng cho lớp tiếp theo. Quá trình này sẽ lặp lại cho đến lớp đầu ra cuối cùng, nơi mà quyết định được đưa ra.

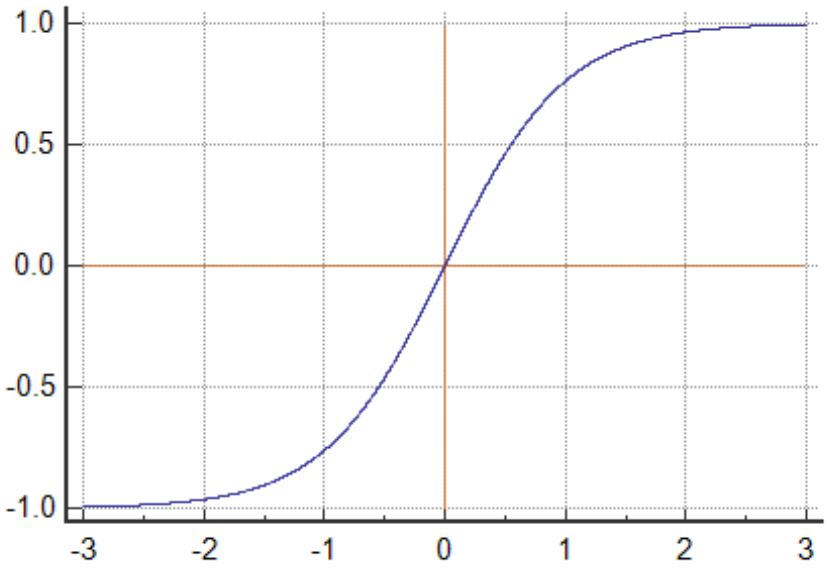


HÌNH 2.7: Kiến trúc mạng neural nhân tạo với 3 lớp ẩn

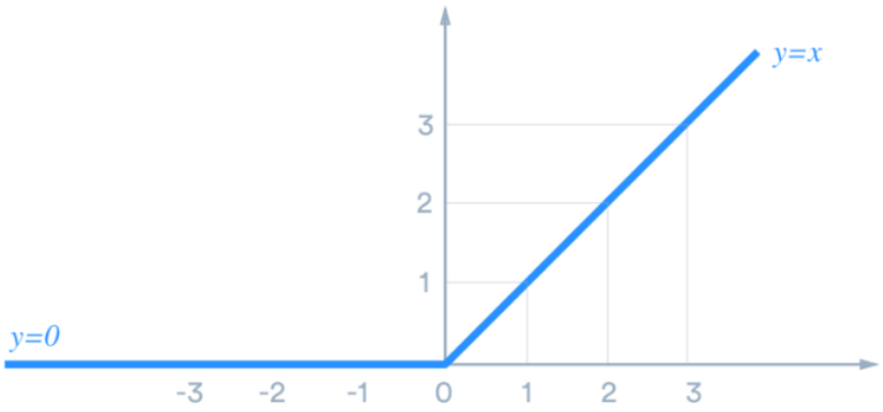
2.2.4 Hàm kích hoạt (Activation function)



HÌNH 2.8: Đồ thị hàm Sigmoid



HÌNH 2.9: Đồ thị hàm Tanh



HÌNH 2.10: So sánh đồ thị Sigmoid và ReLU

Trong quá trình học, trọng số và bias được tinh chỉnh dần theo thời gian để đưa kết quả tiến gần đến giá trị thực. Ngoài một hàm mất mát (loss function) được thiết kết để ước tính lỗi và truyền lại tín hiệu cho neural điều chỉnh tham số thì còn có một hàm kích hoạt. Hàm kích hoạt được thiết lập ở mỗi neural với nhiệm vụ ánh xạ đầu ra từ neural đến miền giá trị mong muốn trước khi lan

truyền kết quả ấy đến những neural khác đảm bảo duy trì việc học của mạng được tiếp diễn. Hàm kích hoạt phải là hàm phi tuyến tính để dữ liệu đầu ra cũng có tính chất phi tuyến, đều này rất quan trọng nếu không muốn phá vỡ phân cấp kiến trúc mạng và biến đổi nó thành một tổ hợp tuyến tính.

- Hàm sigmoid (Hình 2.8): Đây là hàm được sử dụng phổ biến vì độ phức tạp thấp, mô phỏng tốc độ neural sinh học và đưa các giá trị về miền [0,1].Công thức của hàm sigmoid:

$$\sigma(z) \equiv \frac{1}{1+e^{-z}} \tag{2.8}$$

- Hàm Tanh: Miền giá trị hàm này thuộc đoạn [-1, 1]. Đồ thị hàm Tanh được biểu diễn như (Hình 2.9).

- Hàm ReLU (Rectified Linear Unit) (Hình 2.10): Hàm kích hoạt được sử dụng nhiều nhất trên thế giới hiện nay, ReLU cho phép quá trình học nhanh hơn so với hàm Sigmoid và Tanh. Vì vậy, nó được sử dụng trong hầu hết các mạng thần kinh tích chập hoặc học sâu. Có dạng đồ thị tương đồng với Sigmoid, nhưng đối với ReLU, f(z) = 0 khi z < 0, f(z) = z khi z lớn hơn hoặc bằng 0, miền giá trị được xác định trong khoảng [0 đến infinity).

$$f(x) = \max(x, 0) \tag{2.9}$$

- Hàm Leaky ReLU: Với thay đổi nhỏ, Leaky ReLU hoàn toàn có thể ngăn chặn việc kích hoạt bão hòa khi giá trị nhỏ hơn 0. Bên cạnh đó, cũng có một phiên bản khác được gọi là Parametric ReLU. Công thức của Leaky ReLU:

$$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \tag{2.10}$$

- Softmax: Hàm kích hoạt này rất thú vị bởi vì nó không chỉ ánh xạ đầu ra đến phạm vi [0,1] mà còn ánh xạ từng đầu ra theo cách sao cho tổng của chúng bằng 1. Do đó, đầu ra của Softmax là phân phối xác suất. Hàm softmax thường được sử dụng trong lớp cuối cùng của phân loại dựa trên mạng thần kinh. Các mạng như vậy thường được đào tạo theo chế độ log-loss (hoặc cross-entropy), tạo ra một biến thể phi tuyến tính của hồi quy logistic đa phương.

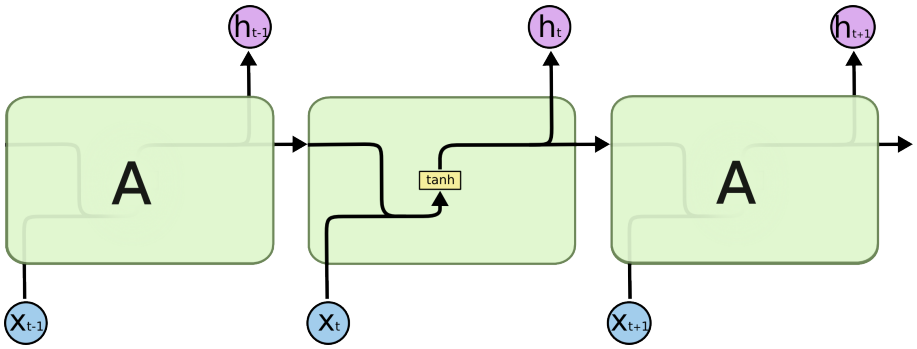
2.2.5 Recurrent neural network (RNN)

Mạng neural hồi quy - Recurrent Neural Network (RNNs) là một hệ thống rất mạnh mẽ trong xử lý ngôn ngữ tự nhiên nhận một chuỗi đầu vào và đầu ra là một chuỗi khác. RNN được ứng dụng trong Speech to text (Chuyển giọng nói thành văn bản), Machine Translation (Dịch máy) hay sử dụng để dự đoán từ tiếp theo trong một câu. Ý tưởng chính của RNN (Recurrent Neural Network) là sử dụng chuỗi các thông tin. Trong các mạng nơ-ron truyền thống tất cả các đầu vào và cả đầu ra là độc lập với nhau. Tức là chúng không liên kết thành chuỗi với nhau. Nhưng các mô hình này không phù hợp trong rất nhiều bài toán như dự đoán câu tiếp theo hoặc phân loại hành động của một người trong video, thì input của hai bài toán này là một chuỗi input có thứ tự của từ hoặc ảnh và Neural Network truyền thống không thể liên kết các chuỗi này lại một cách có thứ tự được. Đó chính là điểm mạnh của Recurrent neural networ kbởi lẽ chúng thực hiện cùng một tác vụ cho tất cả các phần tử của một chuỗi với đầu ra phụ thuộc vào cả các phép tính trước đó. Nói cách khác, RNN có khả năng nhớ các thông tin được tính toán trước đó.

2.2.6 Long short term memory (LSTM)

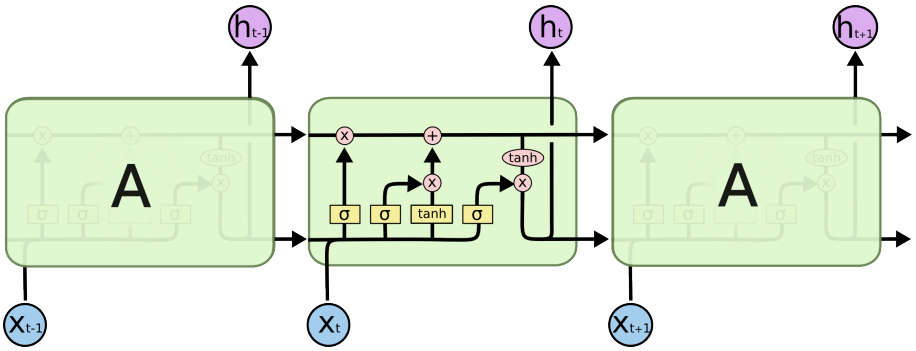
Tuy nhiên, do vấn đề vanishing gradient, RNN rất khó để train và không ổn định. Về lý thuyết là RNN có thể mang thông tin từ các layer trước đến các layer sau, nhưng thực tế là thông tin chỉ mang được qua một số lượng state nhất định, sau đó thì sẽ bị vanishing gradient, hay nói cách khác là model chỉ học được từ các state gần nó (short term memory). LSTM (Long Short Term Memory) với các cổng và memory cells đã khắc phục được nhược điểm trên. Thành công đầu tiên khi sử dụng LSTM là trong lĩnh vực image recognition. Các nghiên cứu trên deep forward network đã chỉ ra rằng ReLUs dễ train hơn là tanh hay logistic.

LSTM được giới thiệu bởi Hochreite và Schmidhuber (1997), và sau đó đã được cải tiến và phổ biến bởi rất nhiều người trong ngành. Chúng hoạt động cực kì hiệu quả trên nhiều bài toán khác nhau nên dần đã trở nên phổ biến như hiện nay. LSTM được thiết kế để tránh được vấn đề phụ thuộc xa (long-term dependency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của chúng, chứ ta không cần phải huấn luyện nó để có thể nhớ được. Tức là ngay nội tại của nó đã có thể ghi nhớ được mà không cần bất kì can thiệp nào. Mọi mạng hồi quy đều có dạng là một chuỗi các mô-đun lặp đi lặp lại của mạng nơ-ron. Với mạng RNN chuẩn, các mô-dun này có cấu trúc rất đơn giản, thường là một tầng tanh (Hình 2.11).



HÌNH 2.11: Mạng RNN chuẩn

LSTM cũng có kiến trúc dạng chuỗi như vậy, nhưng các mô-đun trong nó có cấu trúc khác với mạng RNN chuẩn. Thay vì chỉ có một tầng mạng nơ-ron, chúng có tới 4 tầng tương tác với nhau một cách rất đặc biệt (Hình 2.12).



HÌNH 2.12: Mạng LSTM

Chìa khóa của LSTM là trạng thái tế bào (cell state) - chính đường chạy thông ngang phía trên của sơ đồ hình vẽ.Trạng thái tế bào là một dạng giống như băng truyền. Nó chạy xuyên suốt tất cả các mắt xích (các nút mạng) và chỉ tương tác tuyến tính đôi chút. Vì vậy mà các thông tin có thể dễ dàng truyền đi thông suốt mà không sợ bị thay đổi. Bước đầu tiên của LSTM là quyết định xem thông tin nào cần bỏ đi từ trạng thái tế bào. Quyết định này được đưa ra bởi tầng sigmoid - gọi là "tầng cổng quên" (forget gate layer). Nó sẽ lấy đầu vào là h_{t-1} và x_t rồi đưa ra kết quả là một số trong khoảng [0, 1] cho mỗi số trong trạng thái tế bào C_{t-1}. Đầu ra là 1 thể hiện rằng nó giữ toàn bộ thông tin lại, còn 0 chỉ rằng toàn bộ thông tin sẽ bị bỏ đi.

Chương 3

Thực nghiệm

3.1 Dataset và Preprocessing

3.1.1 Nguồn gốc bộ dữ liệu

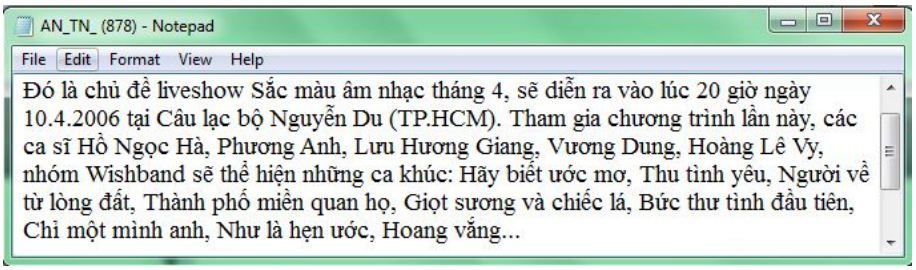
Bộ dữ liệu sử dụng bao gồm 26451 đoạn văn bản đã được tổng hợp và phân loại theo 27 chủ đề lớn (Hình 3.1) là các chủ đề thường thấy trong đời sống như:Am nhac, Am thuc ,Bat dong san, Bong da ,...

HÌNH 3.1: Các chủ đề được sử dụng trong bộ dữ liệu.

Bộ dữ liệu được tổ chức lưu trữ theo theo dạng cây thư mục với tên thư mục chính là nhãn của các đoạn văn bản có cùng chủ đề được lưu trữ vên trong nó.

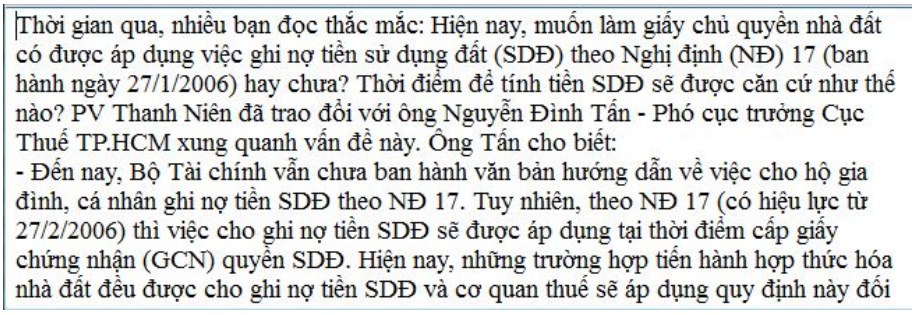
3.1.2 Mô tả chi tiết

Trong bộ dữ liệu được sử dụng mỗi một tập dữ liệu sẽ là một đoạn văn bản được lấy ngẫu nhiên từ một bài báo có chung chủ đề.



HÌNH 3.2: Một tập dữ liệu lấy từ bộ dữ liệu có chủ đề Âm nhạc

Các tập dữ liệu như trên được lấy một cách ngẫu nhiên vì vậy có thể chứa các ký tự không có nghĩa như: dấu ( ), dấu " ", những chữ số hoặc ngày tháng,...


Thời gian qua, nhiều bạn đọc thắc mắc: Hiện nay, muốn làm giấy chủ quyền nhà đất có được áp dụng việc ghi nợ tiền sử dụng đất (SDĐ) theo Nghị định (NĐ) 17 (ban hành ngày 27/1/2006) hay chưa? Thời điểm để tính tiền SDĐ sẽ được căn cứ như thế nào? PV Thanh Niên đã trao đổi với ông Nguyễn Đình Tấn - Phó cục trưởng Cục Thuế TP.HCM xung quanh vấn đề này. Ông Tấn cho biết:
- Đến nay, Bộ Tài chính vẫn chưa ban hành văn bản hướng dẫn về việc cho hộ gia đình, cá nhân ghi nợ tiền SDĐ theo NĐ 17. Tuy nhiên, theo NĐ 17 (có hiệu lực từ 27/2/2006) thì việc cho ghi nợ tiền SDĐ sẽ được áp dụng tại thời điểm cấp giấy chứng nhận (GCN) quyền SDĐ. Hiện nay, những trường hợp tiến hành hợp thức hóa nhà đất đều được cho ghi nợ tiền SDĐ và cơ quan thuế sẽ áp dụng quy định này đối

HÌNH 3.3: Một đoạn văn bản có chứa các ký tự không có ý nghĩa.

Tập dữ liệu train bao gồm 27 chủ đề với 14375 tập dữ liệu được lưu dưới dạng tên.txt có dung lượng 94MB.

Tập dữ liệu test bao gồm 27 chủ đề với 12076 tập dữ liệu được lưu dưới dạng tên.txt có dung lượng 74.4MB.

3.1.3 Tiền xử lý dữ liệu (Preprocessing)

Văn bản trước khi được vector hóa, trước khi sử dụng cần phải được tiến hành tiền xử lý. Quá trình tiền xử lý sẽ nâng cao độ chính xác trong quá trình phân loại và giảm sự phức tạp của các thuật tán huấn luyện. Từ đó giảm thời gian thực hiện đồng thời nâng cao hiệu suất. Quá trình tiền xử lý dữ liệu tiến hành càng tốt, dữ liệu được xử lý tốt sẽ nâng cao dộ chính xác trong quá trình phân loại vì vậy đây là một quá trình quan trọng cần phải được thực hiện trước khi tiến hành train dữ liệu.

Tùy từng trường hợp sẽ có những phương thức tiền xử lý khác nhau để đạt được kết quả cao nhất. Trong trường hợp này nhóm sử tiến hành tiền xử lý như sau:

- Xóa các ký tự đặc biệt không phải chữ hoặc số.
- Xóa ký tự là số.
- Xóa từ có số và chữ, vd a1, a2,...
- Chuẩn hóa để mỗi từ cách nhau một khoảng trắng.
- Ghép các từ riêng lẻ thành cụm từ có nghĩa.
- xóa stopword[5].
- Xóa các chữ đứng riêng lẻ.

Những ký tự được xáo trên trong văn bản tiếng việt nó không mang ý nghĩa giúp phân loại văn bản tiếng việt vì vậy cần xóa đi để nội dung văn bản được cô động hơn.

Tất cả quá trình xử lý được nhóm làm thành một hàm để thuận tiện cho việc sử dụng.

```
# hàm dùng tiền xử lý dữ liệu
def Preprocessing(path):
  files = []
  ids = []
  stopword = []
  with open("/content/drive/MyDrive/CoLab/Đồ án Machine Learning/data/vietnamese-stopwords-dash.txt", "r") as f:
    stopword = f.read().splitlines()
# Lấy các file .txt
  # r=root, d=directories, f = files
  for r, d, f in os.walk(path):
      for folder in d:
          id = PhanLoai(folder)
          for r1, d1, f in os.walk(os.path.join(r, folder)):
            for file in f:
              files.append(os.path.join(r1, file))
              ids.append(id)
# tiến hành đọc từng file
  X = []
  for file in tqdm(files, desc='Processing', position=0):
    with open(file, 'r', encoding='utf-16') as f:
      # Xóa các ký tự đặc biệt không phải chữ hoặc số
      txt = strip_non_alphanum(txt).lower().strip()
      # Xóa ký tự là số
      txt = strip_numeric(txt)
      # Xóa từ có số và chữ, vd a1 a2
      txt = split_alphanum(txt)
      # Chuẩn hóa để mỗi từ cách nhau một khoảng trắng
```

HÌNH 3.4: Hàm dùng cho quá trình tiền xử lý dữ liệu.

3.1.4 Xử lý dữ liệu

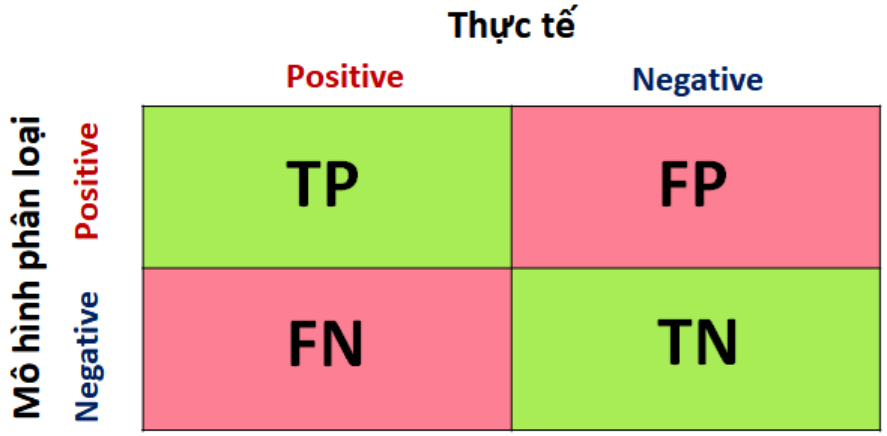
Tập dữ liệu sau khi qua quá trình tiền xử lý vẫn chưa thể đưa vào xử dụng được. Muốn các thuật toán có thể xử lý được dữ liệu để "học" thì dữ liệu cần phải được vector hóa và transformer.

Sau khi khảo sát qua các phương pháp vector và transformer khác nhau nhóm đưa ra phương pháp vector hóa và transformer thích hợp nhất và cho kết quả cao nhất là: CountVectorizer dùng để vector hóa và TfidfTransformer là phương pháp dùng để transformer. Tuy nhiên để quá trình này diễn ra tốt hơn nhóm đã dùng thêm một phương pháp để kết hợp quá trình này lại là Pipeline [6, 7].

```
Pipeline([('vect', CountVectorizer(ngram_range=(1,1),max_df=0.8, max_features=None)),
          ('tfidf', TfidfTransformer()),('clf',ComplementNB())])
```

HÌNH 3.5: Sử dụng Pipeline trong quá trình xử lý dữ liệu.

3.2 Các độ đo [8]



HÌNH 3.6: Minh họa confusion matrix

TP : True Positive

TN : True Negative

FP : False Positive

FN : False Negative

3.2.1 Độ đo Recall

Chỉ số này còn được gọi là độ bao phủ tức là xem xét xem mô hình tìm được có khả năng tổng quát hóa như thế nào.

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

3.2.2 Độ đo Precision

Precision (độ chính xác) là tỉ lệ thực sự positive trên tổng số các trường hợp được mô hình dán nhãn "Positive".

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

3.2.3 Độ đo Accuracy

Có thể tạm dịch thuật ngữ Accuracy như "độ chính xác tổng quát", vì nó đơn giản là tỉ lệ của tất cả trường hợp phân loại Đúng (không phân biệt negative/positive) trên toàn bộ trường hợp trong mẫu kiểm định.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Đây là tiêu chí phổ biến nhất (thường được nghĩ đến đầu tiên) khi kiểm định hiệu năng của mô hình phân loại, tuy nhiên giá trị thực dụng của nó thường kém vì nó không đặc hiệu cho một mục tiêu nào cả.

3.2.4 Độ đo F1

Đây được gọi là một trung bình điều hòa(harmonic mean) của các tiêu chí Precision và Recall. Nó có xu hướng lấy giá trị gần với giá trị nào nhỏ hơn giữa 2 giá trị Precision và Recall và đồng thời nó có giá trị lớn nếu cả 2 giá trị Precision và Recall đều lớn. Chính vì thế F1-Score thể hiện được một cách khách quan hơn performance của một mô hình học máy.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

3.3 Các phương pháp máy học (Machine Learning)

3.3.1 Phương pháp K-Nearest Neighbors

Trong quá trình huấn luyện mô hình bằng thuật toán KNN, nhóm đã sử dụng các tham số như: n_neighbors, metric và weights. Các độ đo khoảng cách được dùng là euclidean, cosine, manhattan, minkowski.

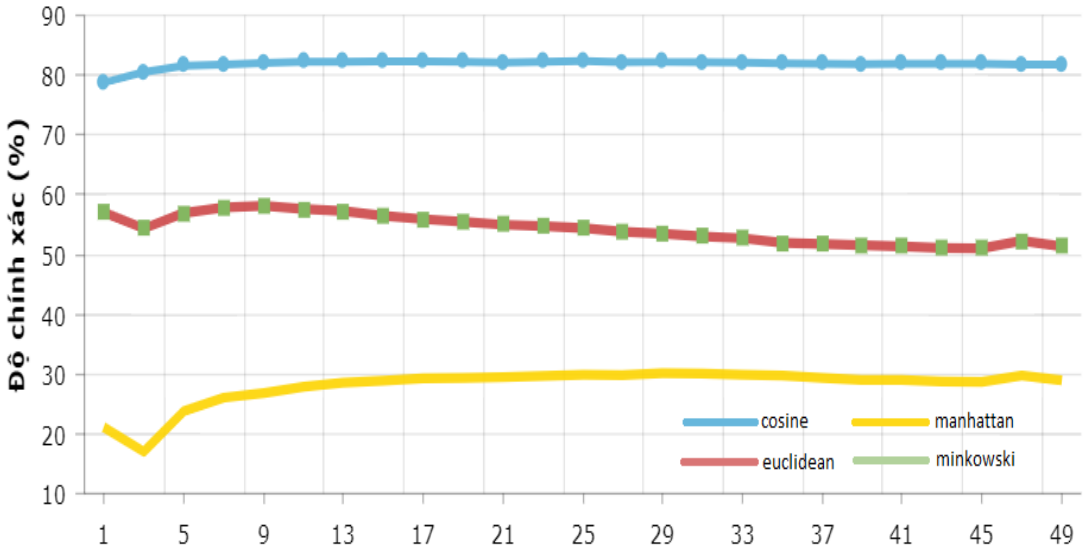
```
a = [1, 3, 5, 7, 9, 11, 13, 15 ,17 ,19,
     21, 23, 25, 27, 29, 31, 33 , 37, 39,
     41, 43, 47, 49]
b = ["euclidean", "cosine","manhattan", "minkowski"]
c = ["uniform", "distance"]
df = pd.DataFrame(columns=('time', 'n_neibors', 'metric',
                           'weights','f1_score', 'accuracy_score',
                           'precision_score', 'recall_score'))
for i in a:
  for j in b:
    for l in c:
      t = time.time()
      model = KNeighborsClassifier(n_neighbors= i, metric= j, weights= l)
      model.fit(X_train, Y_train)
      Y_pred = model.predict(X_test)
      for k in ["micro", "macro", "weighted"]:
        #Tính các độ chính xác
        d1 = f1_score(Y_test, Y_pred, average=k)
        d2 = accuracy_score(Y_test, Y_pred)
        d3 = precision_score(Y_test, Y_pred, average = k)
        d4 = recall_score(Y_test, Y_pred, average = k)
        #Lưu lại kết quả
        row = pd.Series({'time':time.time()-t, 'n_neibors':i, 'metric':j,
                         'average':k, 'weights':l, 'f1_score':d1, 'accuracy_score':d2,
                         'precision_score':d3, 'recall_score':d4})
        df = df.append(row, ignore_index=True)
```

HÌNH 3.7: Train model KNN.

n_neibors	metric	weights	f1	accuracy	precision	recall
25	cosine	distance	0.8235	0.8235	0.8235	0.8235
17	cosine	distance	0.8231	0.8231	0.8231	0.8231
15	cosine	distance	0.8228	0.8228	0.8228	0.8228
13	cosine	distance	0.8226	0.8226	0.8226	0.8226
19	cosine	distance	0.8226	0.8226	0.8226	0.8226
11	cosine	distance	0.8222	0.8222	0.8222	0.8222
23	cosine	distance	0.8221	0.8221	0.8221	0.8221
29	cosine	distance	0.822	0.822	0.822	0.822

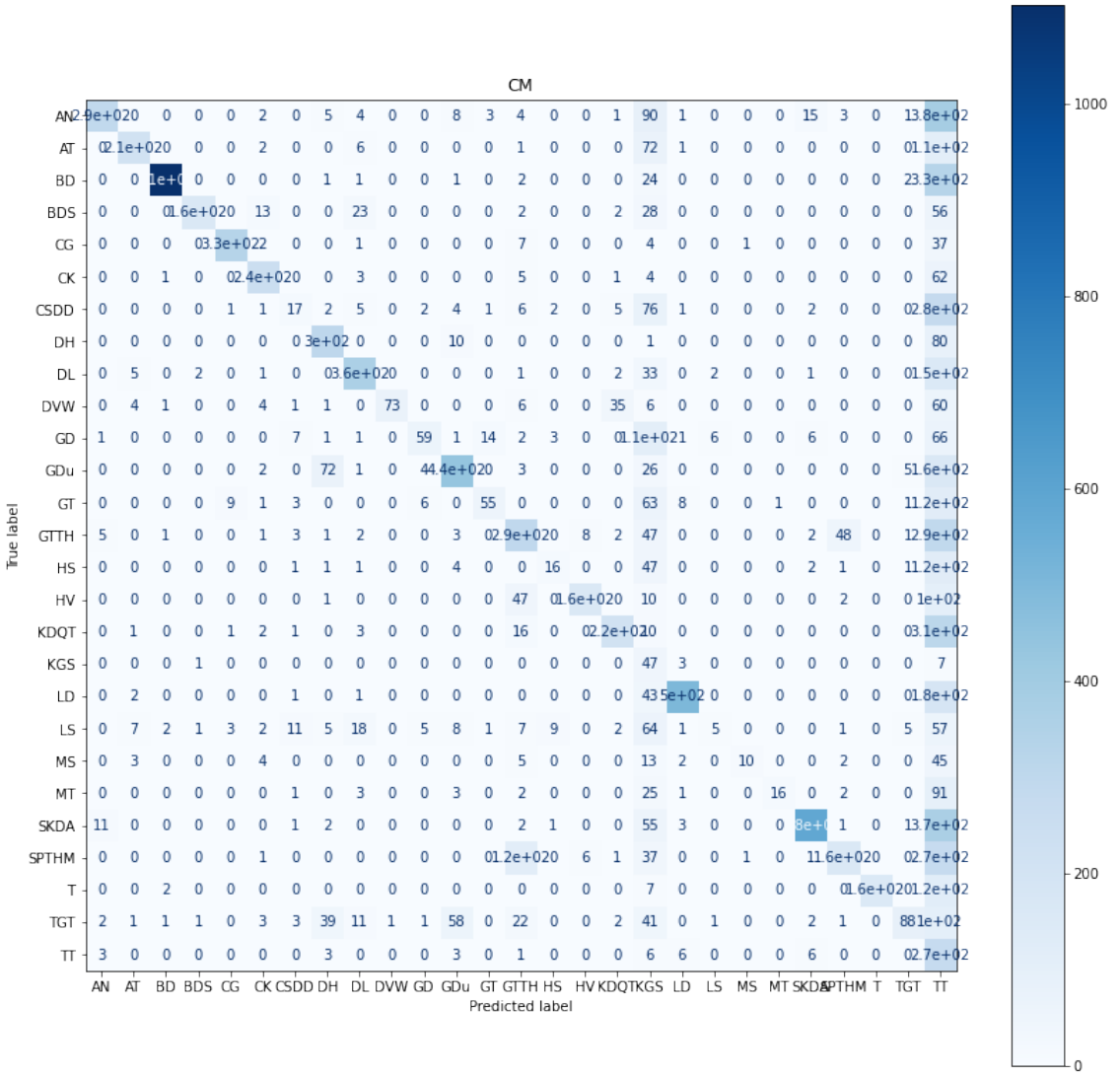
HÌNH 3.8: Các model cho ra kết quả cao nhất.

Tất cả các model cho ra kết quả cao nhất đều sử dụng độ đo khoảng cách cosine. Do đó ta thấy được sử dụng độ đo khoảng cách cosine cho bài toán phân loại chủ đề là rất phù hợp.



HÌNH 3.9: Biểu đồ sự ảnh hưởng của n_neighbors đối với độ chính xác.

Theo kết quả trong hình, Khi sử dụng độ đo khoảng cách là "cosine", ta sẽ được kết quả cao nhất trong 4 độ đo, lên đến hơn 80%, trong khi đó, minkowski và euclidean chỉ khoảng 50% đến 60%, và manhattan cho kết quả thấp nhất khoảng 30%. Trong bài toán phân loại văn bản này thì n_neighbors không quan trọng, độ chính xác không thay đổi nhiều khi tăng n_neighbors.



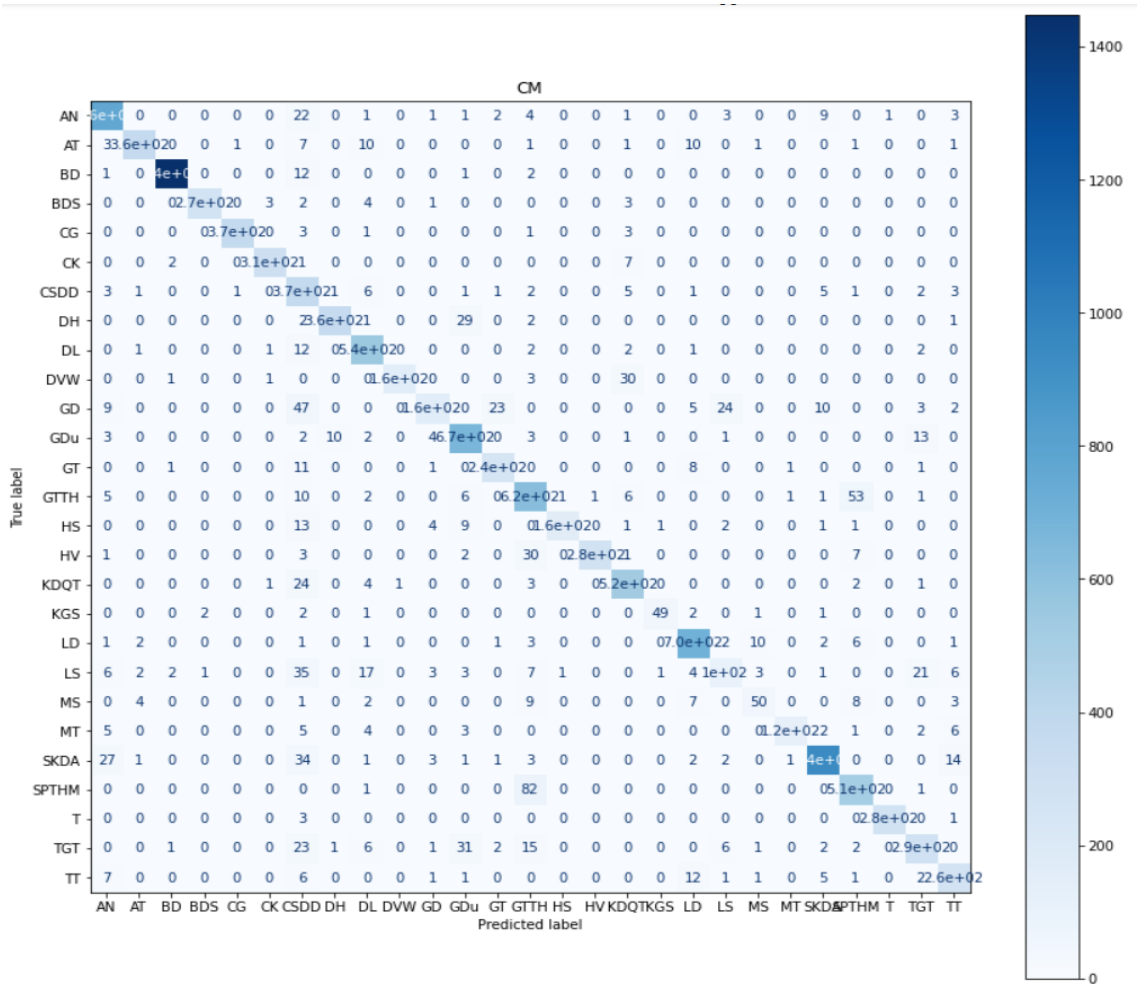
HÌNH 3.10: Confusion Matrix của mô hình KNN cho độ chính xác lớn nhất

"Thời trang" và "Không gian sống: là 2 chủ đề dễ phân loại sai nhất. Thời gian để huấn luyện mô hình với metric là manhattan tốn nhiều thời gian nhất, trung bình 220 giây cho một mô hình, 3 metric còn chỉ mấy từ 10-12 giây.

3.3.2 Phương pháp Logistic Regression

Phương pháp hồi quy logistic là một mô hình hồi quy nhằm dự đoán giá trị đầu ra rời rạc (discrete target variable) y ứng với một véc-tơ đầu vào x. Việc này tương đương với chuyện phân loại các đầu vào x vào các nhóm y tương ứng.

Về phương pháp của Logistic Regression, nhóm dùng các tham số: newton-cg, lbfgs, liblinear, sag, saga. Các thang đo bao gồm F1, precision, recall, accuracy. Với thang đo F1 và tham số truyền vào là micro, ta có kết quả thu được như sau:



HÌNH 3.11: Confusion Matrix trên thang đo F1 của Logistic Regression

Thời gian thực hiện đánh giá bằng thang đo F1 của Logistic Regression là nhanh nhất với chỉ 116,72s.

```
b = ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
for k in b:
  model = LogisticRegression(solver=k)
  model.fit(X_train_trans, Y_train)
  f1 = f1_score(Y_test, model.predict(X_test_trans), average="micro")
  print(k, f1)
title = "CM"
fig, ax = plt.subplots(figsize=(14,14))
disp = plot_confusion_matrix(model, X_test, Y_test,cmap=plt.cm.Blues,ax=ax)
disp.ax_.set_title(title)
print(title)
print(disp.confusion_matrix)
```

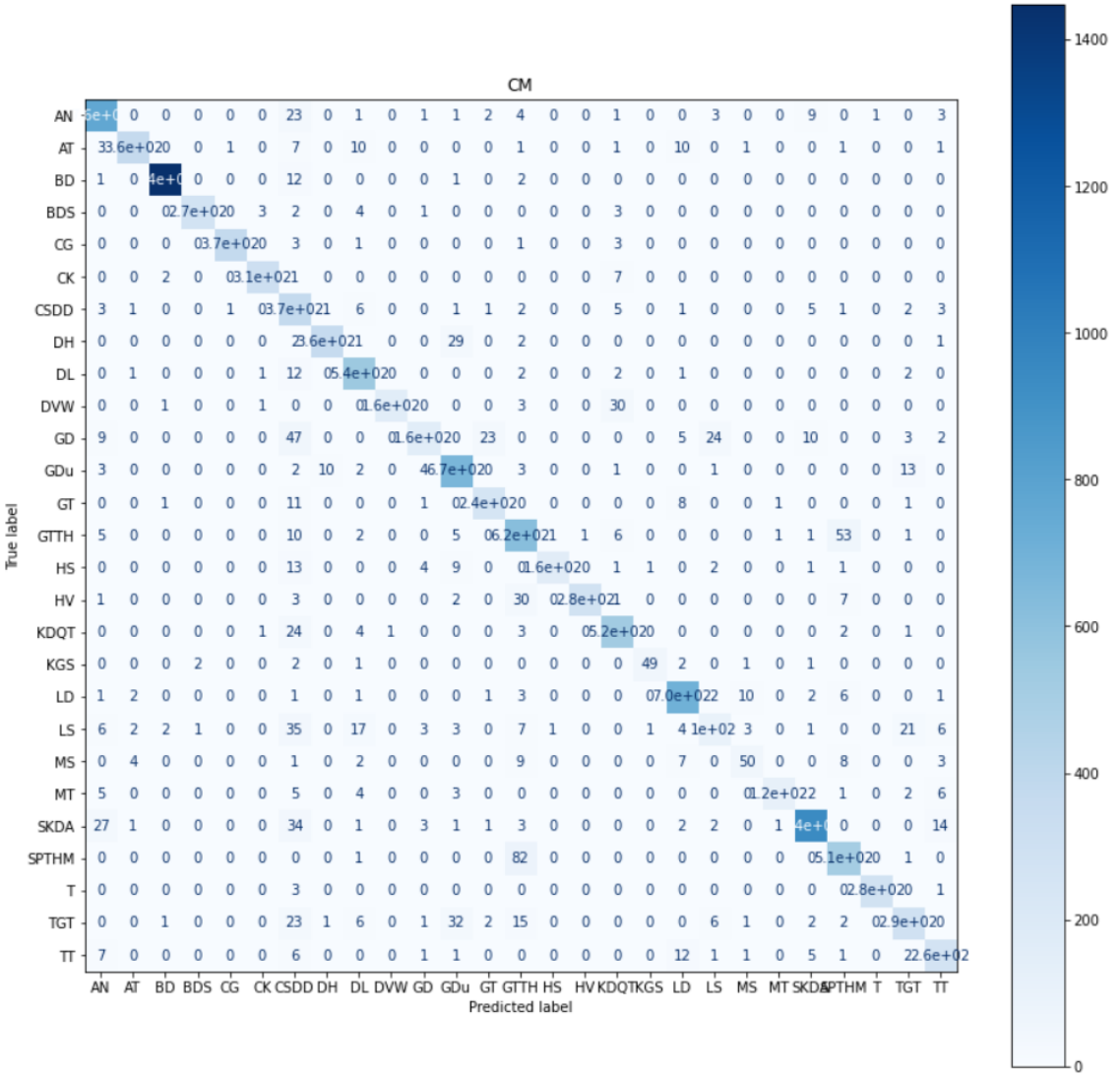
HÌNH 3.12: Train model Logistic Regression và thực hiện đánh giá bằng thang đo F1

Thời gian thực thi đánh giá bằng thang đo recall của Logistic Regression là 332,69s.

```
b = ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
for k in b:
  model = LogisticRegression(solver=k)
  model.fit(X_train_trans, Y_train)
  recall = recall_score(Y_test, model.predict(X_test_trans), average="micro")
  print(k, recall)
title = "CM"
fig, ax = plt.subplots(figsize=(14,14))
disp = plot_confusion_matrix(model, X_test, Y_test,cmap=plt.cm.Blues,ax=ax)
disp.ax_.set_title(title)
print(title)
print(disp.confusion_matrix)
```

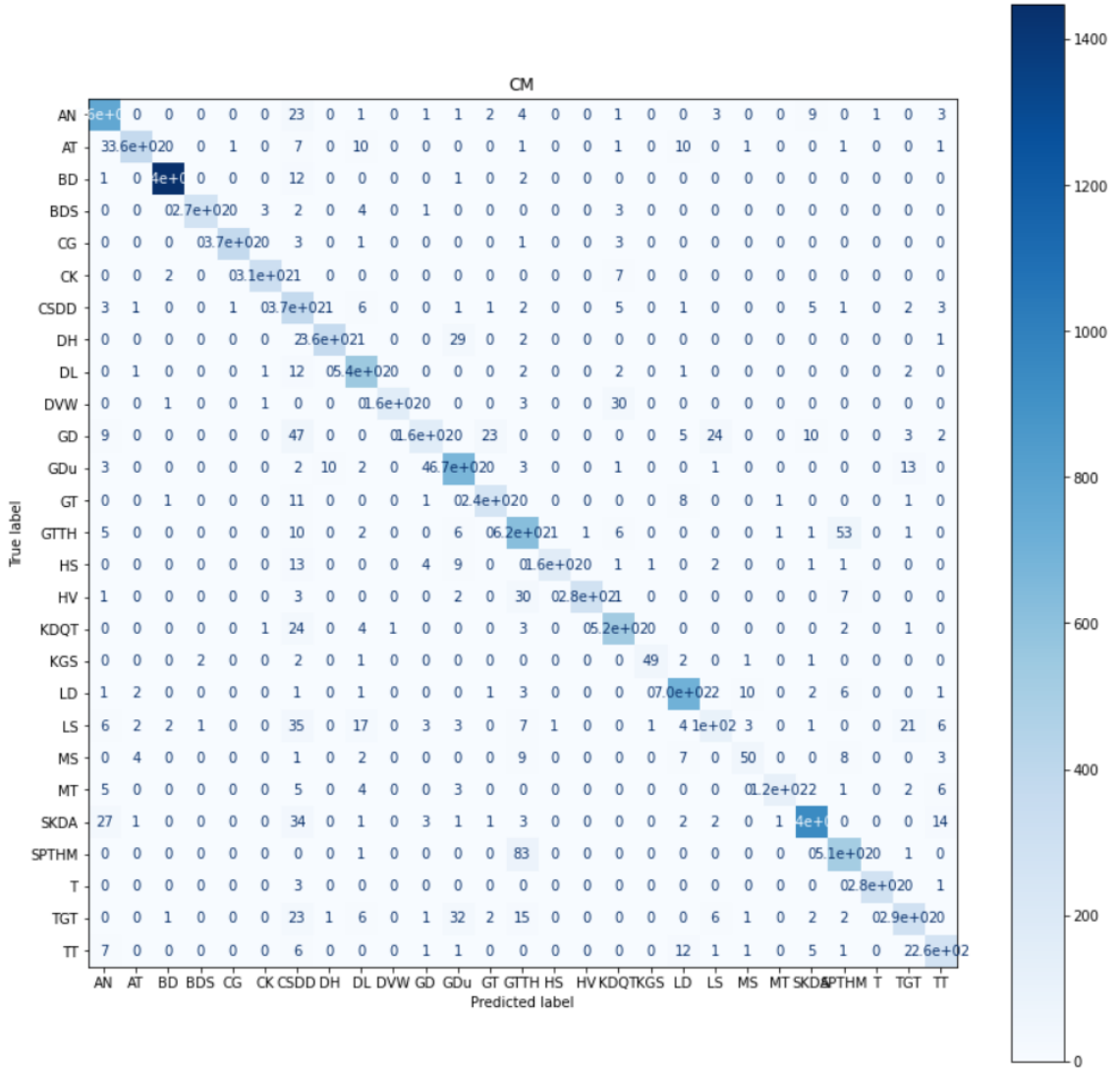
HÌNH 3.13: Train model Logistic Regression và thực hiện đánh giá bằng thang đo recall

Với thang đo Recall và tham số truyền vào là micro, ta có kết quả thu được như sau:



HÌNH 3.14: Confusion Matrix trên thang đo Recall của Logistic Regression

Với thang đo precision và tham số truyền vào là micro, ta có kết quả thu được như sau:



HÌNH 3.15: Confusion Matrix trên thang đo precision của Logistic Regression

Thời gian thực thi đánh giá bằng thang đo precision của Logistic Regression là 213,92s.

```
b = ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
for k in b:
  model = LogisticRegression(solver=k)
  model.fit(X_train_trans, Y_train)
  precision = precision_score(Y_test, model.predict(X_test_trans), average="micro")
  print(k, precision)
title = "CM"
fig, ax = plt.subplots(figsize=(14,14))
disp = plot_confusion_matrix(model, X_test, Y_test,cmap=plt.cm.Blues,ax=ax)
disp.ax_.set_title(title)
print(title)
print(disp.confusion_matrix)
```

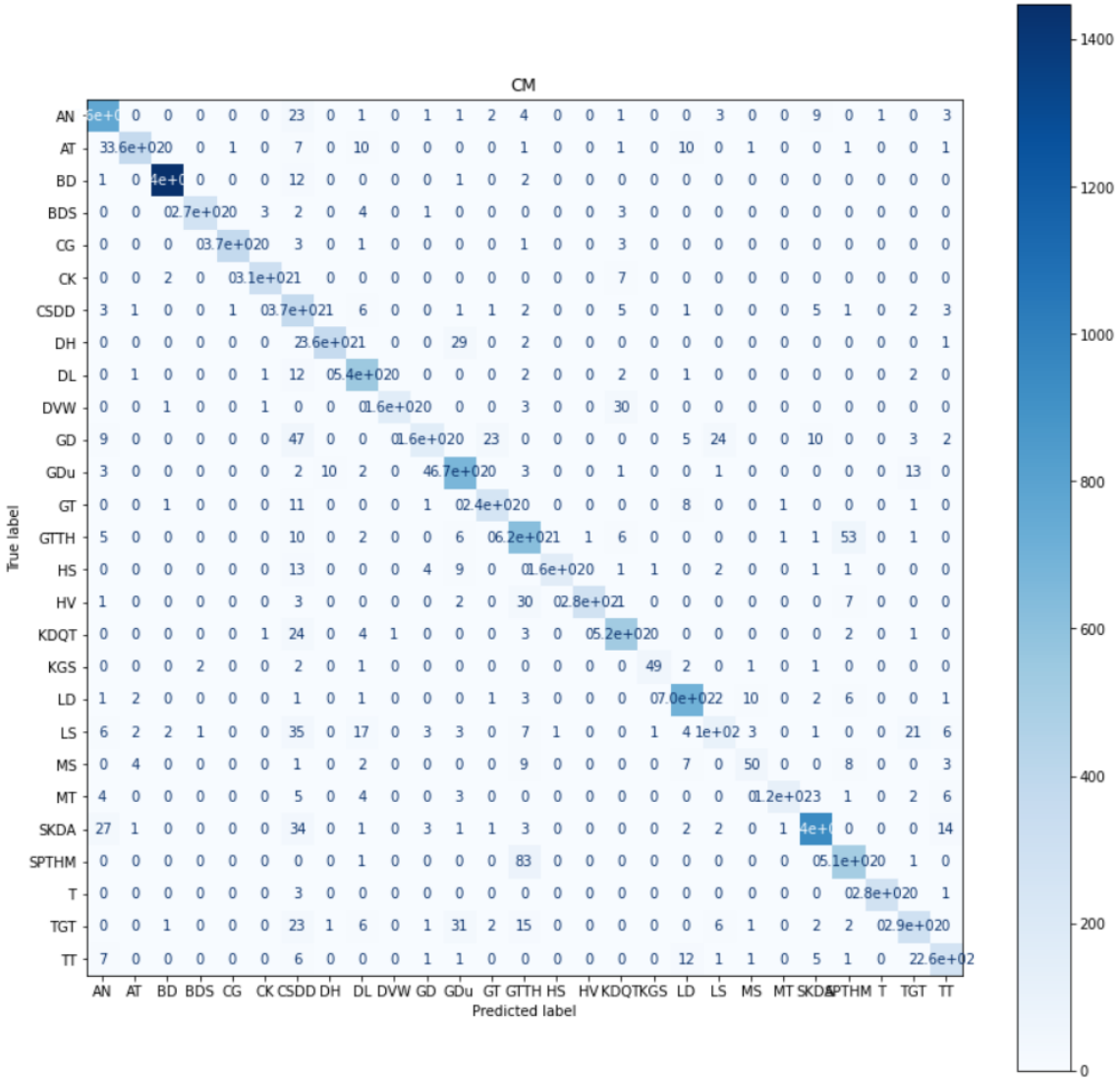
HÌNH 3.16: Train model Logistic Regression và thực hiện đánh giá bằng thang đo precision

Thời gian thực thi đánh giá bằng thang đo accuracy của Logistic Regression là lâu nhất với tận 444,19s.

```
b = ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
for k in b:
  model = LogisticRegression(solver=k)
  model.fit(X_train_trans, Y_train)
  accuracy = accuracy_score(Y_test, model.predict(X_test_trans), normalize="True")
  print(k, accuracy)
title = "CM"
fig, ax = plt.subplots(figsize=(14,14))
disp = plot_confusion_matrix(model, X_test, Y_test,cmap=plt.cm.Blues,ax=ax)
disp.ax_.set_title(title)
print(title)
print(disp.confusion_matrix)
```

HÌNH 3.17: Train model Logistic Regression và thực hiện đánh giá bằng thang đo accuracy

Với thang đo accuracy, ta có kết quả thu được như sau:



HÌNH 3.18: Confusion Matrix trên thang đo accuracy của Logistic Regression

Newton-cg: Phương pháp Newton sắp sử dụng theo nghĩa tốt hơn tối thiểu hóa hàm bậc hai. Tốt hơn bởi vì nó sử dụng xấp xỉ bậc hai (tức là đầu tiên AND giây đạo hàm riêng).

Saga: Bộ giải SAGA là biến thể của SAG cũng hỗ trợ tùy chọn không trơn tru penalty = l1 (ví dụ: L1 Chính quy). Do đó, đây là công cụ giải quyết cho thưa thớt hồi quy logistic đa thức và nó cũng phù hợp rất lớn tập dữ liệu.

Liblinear: Phân loại tuyến tính hỗ trợ hồi quy logistic và máy vectơ hỗ trợ tuyến tính (Trình phân loại tuyến tính đạt được điều này bằng cách đưa ra quyết định phân loại dựa trên giá trị của sự kết hợp tuyến tính của các đặc tính i.e giá trị tính năng).

Sag: Phương pháp SAG tối ưu hóa tổng của một số hữu hạn các hàm lồi mịn. Giống như các phương pháp độ dốc ngẫu nhiên (SG), chi phí lặp của phương pháp SAG không phụ thuộc vào số lượng thuật ngữ trong tổng. Tuy nhiên, bởi kết hợp bộ nhớ của các giá trị gradient trước đó, phương thức SAG đạt được tốc độ hội tụ nhanh hơn so với SG hộp đen phương pháp.

Lbfgs: Tóm lại, nó tương tự Phương pháp của Newton nhưng ở đây ma trận Hessian là xấp xỉ sử dụng các cập nhật được chỉ định bởi độ dốc đánh giá (hoặc đánh giá độ dốc gần đúng). Nói cách khác, sử dụng ước lượng cho ma trận Hessian nghịch đảo.

Case	Solver
L1 penalty	“liblinear” or “saga”
Multinomial loss	“lbfgs”, “sag”, “saga” or “newton-cg”
Very Large dataset (n_samples)	“sag” or “saga”

HÌNH 3.19: Tóm tắt trường hợp dùng các giá trị trong tham số solver

3.3.3 Phương pháp Support Vector Machine

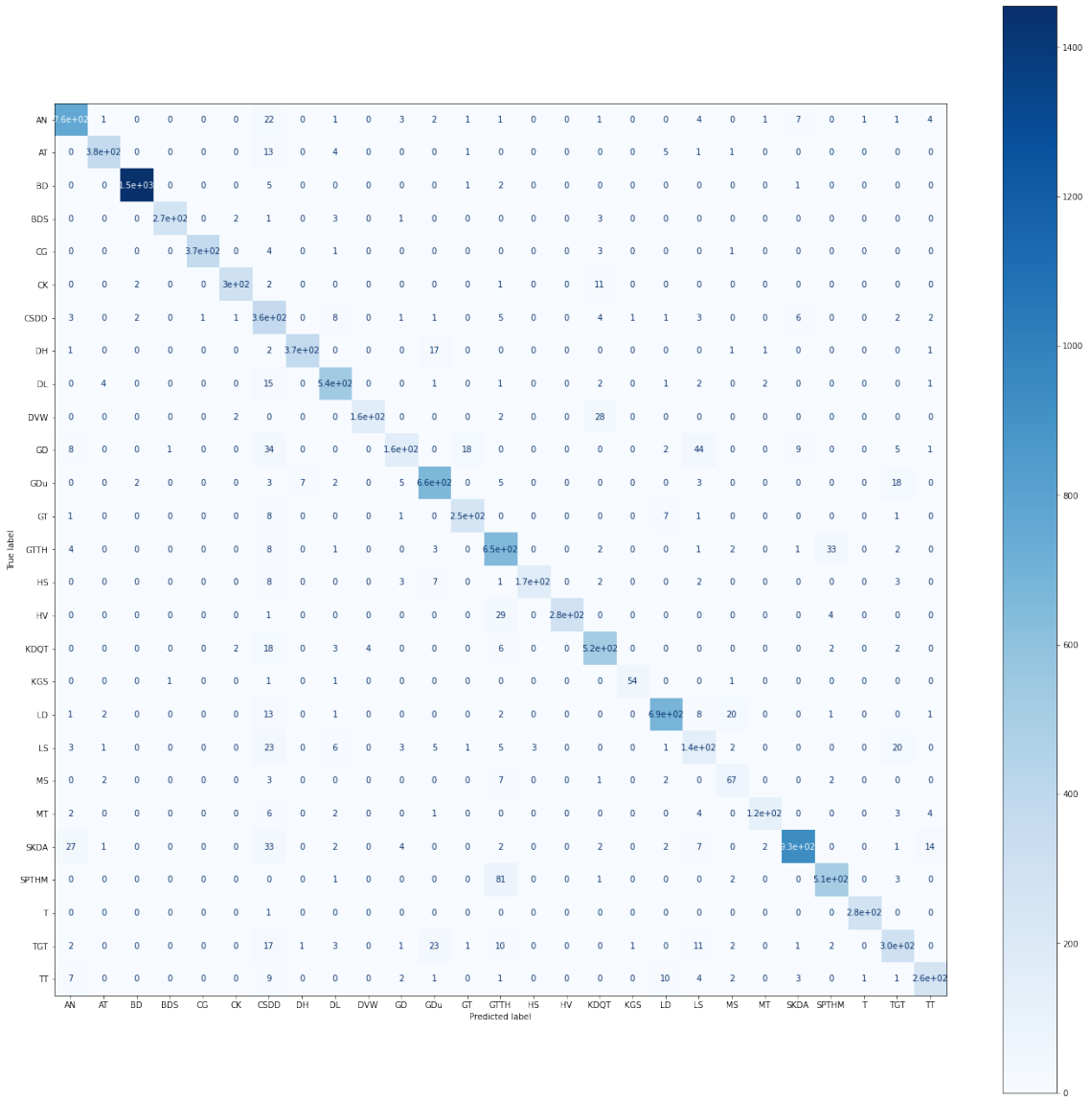
Support Vector Machine là một trong những phương pháp được sử dụng phổ biến trong bài toán phân loại văn bản. Theo những nhóm nghiên cứu từng sử dụng đã đánh giá phương pháp này sử dụng rất tốt cho bài toán phân loại văn bản. Kết quả thực nghiệm của nhóm cũng chỉ ra rằng Support Vector Machine là một trong những phương pháp sử dụng tốt nhất cho bài toán phân loại văn bản như thế này.

Support Vector Machine gồm có 5 kernel khác nhau là: sigmoid, poly, rbf, linear. Với mỗi kernel là một cách thực hiện khác nhau từ đó sẽ cho ra những kết quả khác nhau. Trong lần thực nghiệm này nhóm sẽ tiến hành thử nghiệm hết cả 5 kernel của Support Vector Machine để đưa ra kernel phù hợp nhất.

```
for kernel in tqdm(['sigmoid', 'poly', 'rbf','linear']):
  SVM = Pipeline([('vect', CountVectorizer(ngram_range=(1,1),
        max_df=0.8, max_features=None)),('tfidf', TfidfTransformer()),
        ('clf', SVC(kernel=kernel, verbose=True,gamma='scale'))])
  SVM.fit(X_train,Y_train)
  # Lưu lại model
  pickle.dump(SVM, open('/content/drive/{}_Pipeline.pkl'.format(kernel), 'wb'))
```

HÌNH 3.20: Train model với tất cả các kernel của SVM

Sau khi dùng kết quả train được để kiểm tra nhóm thu đuọc kết quả có độ chính xác khá ấn tượng:



HÌNH 3.21: Confusion Matrix với Support Vector Machine sử dụng kernel là linear

Theo kết quả Confusion Matrix trên cho thấy thuật toán có khả năng phân loại các báo khá chính xác. Tuy nhiên một số nhãn thuộc các chủ đề như: "thế giới đó đây",... thường bị các nhãn khác nhận nhầm.

Qua quá trình thực nghiệm với phương pháp Support Vector Machine thu được bảng kết quả:

SVM kernel	Tham số các độ đo	Precision	F1 score	Recall	Accuracy
Sigmoid	**macro**	0.9057872	0.8940373	0.8893005	**0.9130507**
	micro	0.9130507	0.9130507	**0.9130507**	
	weighted	0.9210688	0.9144453	**0.9130507**	
Poly	**macro**	0.8500032	0.5770243	0.5142074	0.6451640
	micro	0.6451640	0.6451640	0.6451640	
	weighted	0.8689576	0.6905360	0.6451640	
Rbf	**macro**	0.9046694	0.8636869	0.8423350	0.8855581
	micro	0.8855581	0.8855581	0.8855581	
	weighted	0.9133170	0.8924316	0.8855581	
Linear	**macro**	0.9065141	0.8944287	0.8894568	0.9129679
	micro	0.9129679	0.9129679	0.9129679	
	weighted	**0.9212582**	**0.9144884**	0.9129679	

HÌNH 3.22: Bảng kết quả phương pháp Support Vector Machine

Sau khi khảo sát qua bảng kết quả nhóm đưa ra được kernel tốt nhất trong bài toán này là linear với độ chính xác cao nhất là 0.9212582 với độ đo sử dụng là Precision và tham số độ đo là weighted.

3.3.4 Phương pháp Naive Bayes

Nhóm sử dụng phương pháp Naive Bayes với các phân phối xác xuất: Multinomial, Complement, Bernoulli.

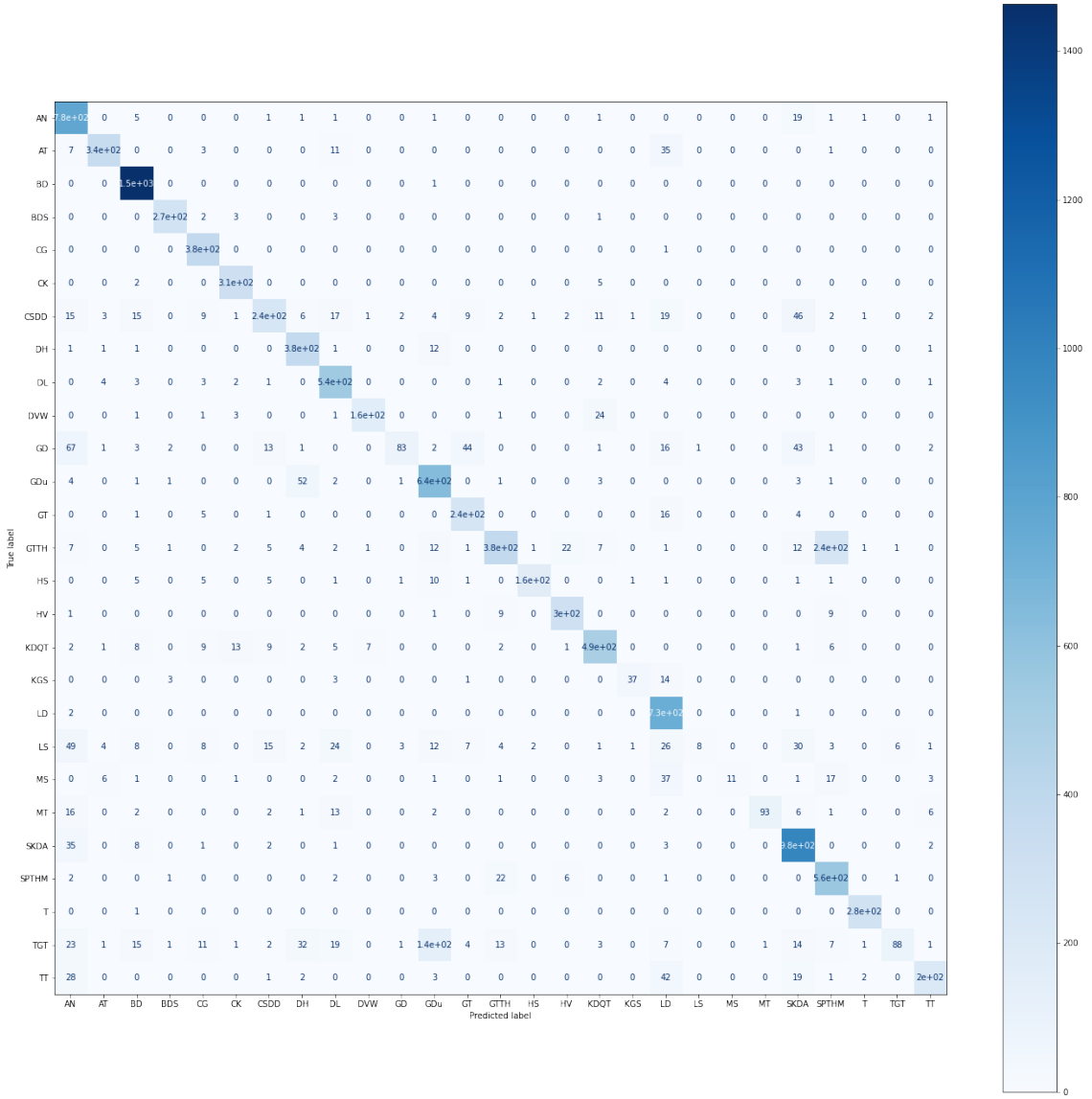
```
# dùng pipeline để training mô hình NB MultinomialNB
NB = Pipeline([('vect', CountVectorizer(ngram_range=(1,1),max_df=0.8,
    max_features=None)),('tfidf', TfidfTransformer()),('clf',MultinomialNB())])
NB = NB.fit(X_train, Y_train)
pickle.dump(, open('/content/drive/MyDrive/MultinomialNB_Pipeline.pkl', 'wb'))
```

```
# dùng pipeline để training mô hình NB ComplementNB
NB = Pipeline([('vect', CountVectorizer(ngram_range=(1,1),max_df=0.8,
    max_features=None)),('tfidf', TfidfTransformer()),('clf',ComplementNB())])
NB = NB.fit(X_train, Y_train)
pickle.dump(NB, open('/content/drive/MyDrive/ComplementNB_Pipeline.pkl', 'wb'))
```

```
# dùng pipeline để training mô hình NB BernoulliNB
NB = Pipeline([('vect', CountVectorizer(ngram_range=(1,1),max_df=0.8,
    max_features=None)),('tfidf', TfidfTransformer()),('clf',BernoulliNB())])
NB = NB.fit(X_train, Y_train)
pickle.dump(NB, open('/content/drive/MyDrive/BernoulliNB_Pipeline.pkl', 'wb'))
```

HÌNH 3.23: Sử dụng các phân phối xác suất khác nhau để train

Sau quá trình thực nghiệm nhóm nhận thấy rằng phương pháp này có thể thực hiện nhanh chóng, không mất nhiều thời gian như những phương pháp khác tuy nhiên kết quả thu có độ chính xác không quá cao.



HÌNH 3.24: Confusion Matrix với Naive Bayes sử dụng phân phối Complement

Dựa trên bảng kết quả Confusion Matrix trên ta có thể nhận thấy thuật toán còn nhiều nhãn bị nhận diện sai. Trong đó những bài báo có nhãn là "giải trí tin học" thường bị nhận nhầm sang "sản phẩm tin học mới", các bài báo thuộc nhãn "thế giới trẻ" thường bị nhận nhầm thành các bài báo có nhãn "giáo dục".

Một số bài báo thường bị nhận dạng nhầm sang các nhãn khác như: "thế giới đó đây", "lối sống",...

Khảo sát kết quả của phương pháp Naive Bayes ta thu được bảng kết quả:

Naive bayes	Tham số các độ đo	Precision	F1 score	Recall	Accuracy
MultinomialNB	**macro**	0.7804496	0.5736489	0.5709310	0.7456111
	micro	0.7456111	0.7456111	0.7456111	
	weighted	0.7644268	0.6955811	0.7456111	
ComplementNB	**macro**	**0.8787197**	0.7723145	0.7637553	**0.8407585**
	micro	0.8407585	**0.8407585**	**0.8407585**	
	weighted	0.8554158	0.8200834	**0.8407585**	
BernoulliNB	**macro**	0.6316218	0.3967288	0.4189293	0.6252070
	micro	0.6252070	0.6252070	0.6252070	
	weighted	0.6913821	0.5589670	0.6252070	

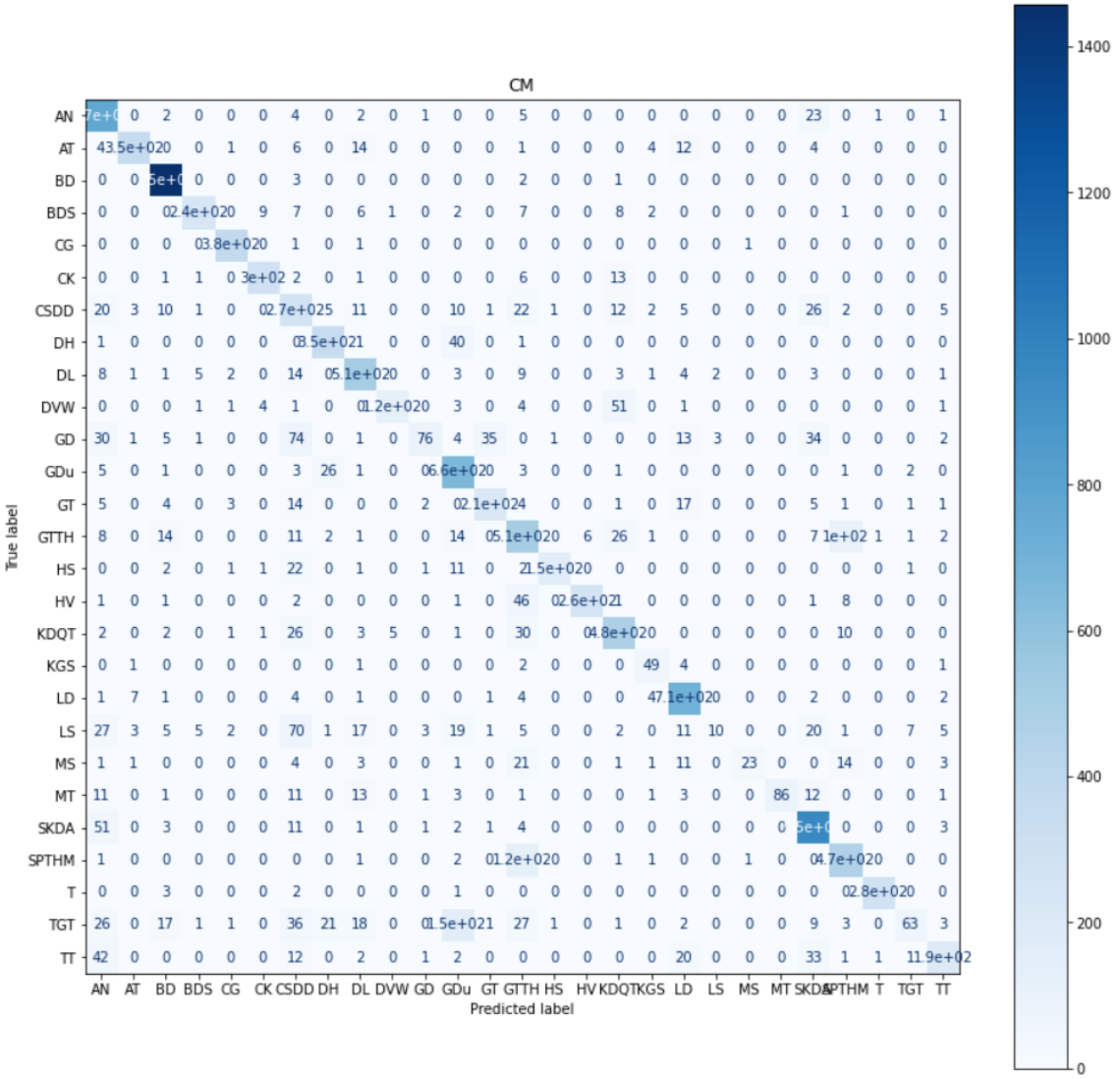
HÌNH 3.25: Kết quả sau khi sử dụng các độ đo khác nhau với phương pháp Naive Bayes

Bảng kết quả trên trình bày các kết quả khác nhau với từng phân phối khác nhau cho mỗi độ đo khác nhau và những tham số truyền vào khác nhau. Từ đó cho ta thấy được phân phối xác suất thích hợp nhất cho thuật toán Naive Bayes là phân phốiComplement Naive Bayes.

3.3.5 Phương pháp Random Forest Classification

Đây là phương pháp xây dựng một tập hợp rất nhiều cây quyết định và sử dụng phương pháp voting để đưa ra quyết định về biến target cần được dự báo. Một ví dụ về Random Forest như sau: giả sử bạn muốn đi tham quan du lịch Anh và có sự cân nhắc cho việc tham quan thành phố nào như: Manchester, Liverpool hay Birmingham. Để trả lời câu hỏi này bạn sẽ cần tham khảo rất nhiều ý kiến từ bạn bè, blog du lịch, tour lữ hành ... Mỗi một ý kiến tương ứng với một Decision Tree trả lời các câu hỏi như: thành phố này đẹp không, có được tham quan các sân vận động khi đến thăm không, số tiền bỏ ra là bao nhiêu, thời gian để tham quan thành phố là bao lâu... Sau đó bạn sẽ có một rừng các câu trả lời để quyết định xem mình sẽ đi tham quan thành phố nào. Random Forest hoạt động bằng cách đánh giá các Decision Tree sử dụng cách thức voting để đưa ra kết quả cuối cùng."

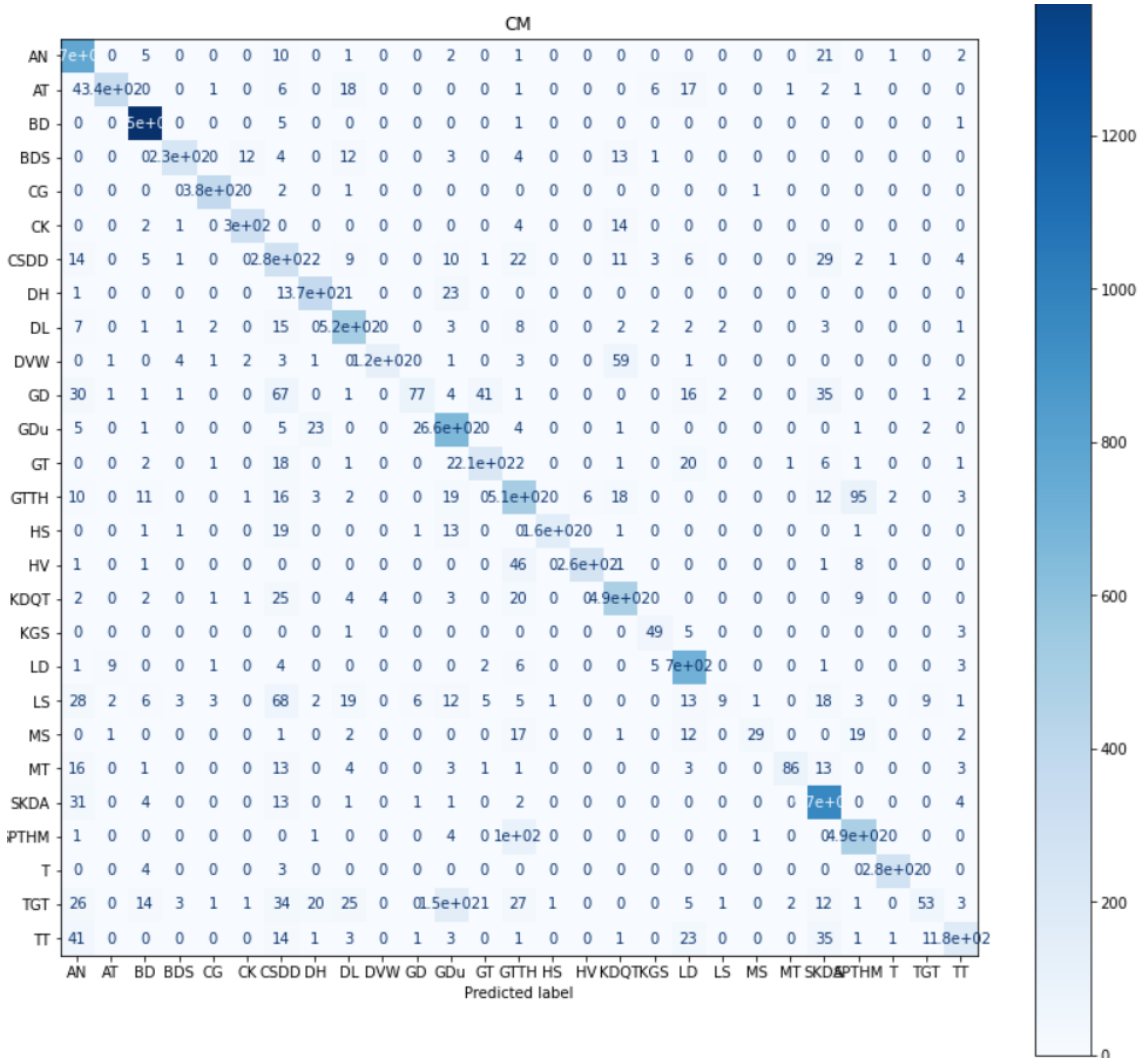
Với phương pháp Random Forest Classification thì nhóm sử dụng các tham số: gini, entropy. Với thang đo là F1 và tham số truyền vào là micro, ta thu được kết quả:



HÌNH 3.26: Confusion Matrix trên thang đo F1 của Random Forest Classification

Thời gian thực thi đánh giá bằng thang đo F1 của Random Forest Classification là 19666,53s.

Với thang đo precision và tham số truyền vào là micro, ta có kết quả thu được như sau:



HÌNH 3.27: Confusion Matrix trên thang đo precision của Random Forest Classification

Thời gian thực thi đánh giá bằng thang đo precision của Random Forest Classification là 6745,33s.

Với thang đo recall và tham số truyền vào là micro, ta có kết quả thu được như sau:

```
90 gini 0.849205034779728 4
90 entropy 0.826018549188473
91 gini 0.855498509440212
91 entropy 0.823782709506459
92 gini 0.852103454786353
92 entropy 0.823203047366677 7
93 gini 0.850198741305067 9
93 entropy 0.824362371646240 5
94 gini 0.853014243126863 2
94 entropy 0.824196753892017 2
95 gini 0.851358065584630 7
95 entropy 0.821215634315998 6
96 gini 0.854339185160649 2
96 entropy 0.823865518383570 7
97 gini 0.84928784365684
97 entropy 0.822209340841338 2
98 gini 0.852186154355747
98 entropy 0.820553163299105 7
99 gini 0.848625372639947
99 entropy 0.823368665120901
```

HÌNH 3.28: Kết quả cao nhất thang đo recall của Random Forest Classification

```
a = ["gini", "entropy"]
for i in range(1,100):
  for j in a:
    model = RandomForestClassifier(n_estimators=i, criterion = j)
    model.fit(X_train_trans, Y_train)
    score = recall_score(Y_test, model.predict(X_test_trans), average = "micro")
    print(i, j , score)
title = "CM"
fig, ax = plt.subplots(figsize=(14,14))
disp = plot_confusion_matrix(model, X_test, Y_test,cmap=plt.cm.Blues,ax=ax)
disp.ax_.set_title(title)
print(title)
print(disp.confusion_matrix)
```

HÌNH 3.29: Train model Random Forest Classification và thực hiện đánh giá bằng thang đo recall

Với thang đo accuracy, ta có kết quả thu được như sau:

```
90 gini 0.850695594567737 7
90 entropy 0.819973501159324 3
91 gini 0.851606492215965 5
91 entropy 0.822374958595561 4
92 gini 0.851689301093077 2
92 entropy 0.822623385226896 4
93 gini 0.852268963232858 6
93 entropy 0.827095064590924 2
94 gini 0.851275256707519
94 entropy 0.823037429612454 4
95 gini 0.851854918847300 5
95 entropy 0.823285856243789 4
96 gini 0.850198741305067 9
96 entropy 0.823948327260682 3
97 gini 0.852931434249751 6
97 entropy 0.827757535607817 2
98 gini 0.853262669758198 1
98 entropy 0.823368665120901
99 gini 0.852600198741305 1
99 entropy 0.823782709506459
```

HÌNH 3.30: Kết quả cao nhất thang đo accuracy của Random Forest Classification

```
a = ["gini", "entropy"]
for i in range(1,100):
  for j in a:
    model = RandomForestClassifier(n_estimators=i, criterion = j)
    model.fit(X_train_trans, Y_train)
    score = accuracy_score(Y_test, model.predict(X_test_trans), normalize="True")
    print(i, j , score)
title = "CM"
fig, ax = plt.subplots(figsize=(14,14))
disp = plot_confusion_matrix(model, X_test, Y_test,cmap=plt.cm.Blues,ax=ax)
disp.ax_.set_title(title)
print(title)
print(disp.confusion_matrix)
```

HÌNH 3.31: Train model Random Forest Classification và thực hiện đánh giá bằng thang đo accuracy

Theo những kết quả trên, ta có thể thấy với tham số n càng lớn thì kết quả càng cao.

Sau khi thực nghiệm, có thể thấy thời gian của các phương pháp thực hiện mất rất nhiều thời gian, kết quả cao dần theo tham số n của mỗi phương pháp với tham số n chính là số lượng phần tử trong một tập hợp.

Entropy: Dùng trong các thuật toán sinh cây ID3, C4.5 và C5.0. Số đo này dựa trên khái niệm entropy trong lý thuyết thông tin (information theory).

$$I_E(i) = -\sum_{j=1}^{m} f(i,j) \log_2 f(i, approxeq. dashv. sdot)$$

HÌNH 3.32: Công thức entropy

Gini: Dùng trong thuật toán CART (Classification and Regression Trees). Nó dựa vào việc bình phương các xác suất thành viên cho mỗi thể loại đích trong nút. Giá trị của nó tiến đến cực tiểu (bằng 0) khi mọi trường hợp trong nút rơi vào một thể loại đích duy nhất.

Giả sử y nhận các giá trị trong 1, 2,..., m và gọi f(i,j) là tần suất của giá trị j trong nút i. Nghĩa là f(i,j) là tỷ lệ các bản ghi với y=j được xếp vào nhóm i.

$$I_G(i) = 1 - \sum_{j=1}^{m} f(i,j)^2$$

HÌNH 3.33: Công thức gini

3.4 Phương pháp Long short term memory (LSTM)

Theo Hình 3.34, LSTM được áp dụng trong bài toán được cài đặt gồm: Input có kích thước là 1000; 4 Hidden Layer với activation function là relu; Output Layer với 27 chủ đề và activation function là softmax và categorical crossentropy loss function cho bài toán phân loại.

```
def create_lstm_model():
    input_layer = Input(shape=(1000,))
    layer = Reshape((1, 1000))(input_layer)
    layer = LSTM(512, activation='relu')(layer)
    layer = Dense(512, activation='relu')(layer)
    layer = Dense(512, activation='relu')(layer)
    output_layer = Dense(27, activation='softmax')(layer)

    classifier = Model(input_layer, output_layer)
    classifier.compile(optimizer=optimizers.Adam(),
      loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return classifier
```

HÌNH 3.34: Hàm LSTM

```
classifier = create_lstm_model()
train_model(classifier=classifier, X_data=X_data_tfidf_svd, y_data=y_data_n,
    X_test=X_test_tfidf_svd, y_test=y_test_n, is_neuralnet=True, n_epochs= 10)
```

```
se cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU
 0s 15ms/step - loss: 3.0941 - accuracy: 0.1282 - val_loss: 2.8033 - val_accuracy: 0.1996
 0s 8ms/step - loss: 2.2453 - accuracy: 0.4193 - val_loss: 1.5841 - val_accuracy: 0.6259
 0s 7ms/step - loss: 1.1023 - accuracy: 0.7281 - val_loss: 0.8409 - val_accuracy: 0.7893
 0s 8ms/step - loss: 0.5269 - accuracy: 0.8588 - val_loss: 0.5264 - val_accuracy: 0.8512
 0s 8ms/step - loss: 0.2957 - accuracy: 0.9181 - val_loss: 0.4672 - val_accuracy: 0.8672
 0s 8ms/step - loss: 0.2048 - accuracy: 0.9418 - val_loss: 0.4599 - val_accuracy: 0.8734
 0s 8ms/step - loss: 0.1567 - accuracy: 0.9552 - val_loss: 0.4531 - val_accuracy: 0.8707
 0s 7ms/step - loss: 0.1284 - accuracy: 0.9635 - val_loss: 0.4500 - val_accuracy: 0.8790
 0s 8ms/step - loss: 0.1041 - accuracy: 0.9723 - val_loss: 0.4759 - val_accuracy: 0.8672
 0s 7ms/step - loss: 0.0922 - accuracy: 0.9753 - val_loss: 0.4921 - val_accuracy: 0.8727
```

HÌNH 3.35: Hàm LSTM

Bài toán phân loại chủ để văn bản được áp dụng phương pháp LSTM đã cho ra kết quả khá cao trên tập test với 0.8766 trên độ đo Accuracy (Hình 3.35).

3.5 Kết quả thực nghiệm

Kết quả thực nghiệm được đánh giá trên các độ đo Accuracy, F1-Score, Precision và Recall.

- Kết quả chạy thực nghiệm trên tập Test có cho ra kết quả cao nhất ở độ đo Precision là 0.92 của thuật toán Support Vector Machine với đối số đầu vào kernel = Linear.
- Trên độ đo Accuracy dù không đạt được 0.92 như thuật toán SVM nhưng với số điểm 0.88 thuật toán LSTM đã cho ra kết quả rất tốt.

Ngoài ra, dựa trên các bảng confusion matrix, có thể thấy rằng do sự phân bố không đồng đều trong tập dữ liệu, các chủ đề Không gian sống, Lối sống, Mua sắm và Thời trang thường bị phân loại sai so với các chủ đề khác.

Chương 4

Kết luận

Trong chương này chúng tôi sẽ tổng hợp lại những gì chúng tôi đã làm được đối với bài toán phân loại chủ đề văn bản.

4.1 Những kết quả đạt được

- Tiếp cận các bài toán Natural Language Processing (NLP).
- Tiếp cận qui trình xử lý dữ liệu lớn 26451 điểm dữ liệu.
- Tìm hiểu các phương pháp máy học trong việc ứng dụng vào bài toán phân loại chủ đề văn bản.
- Tìm hiểu phương pháp học sâu Long short term memory (LSTM).

4.2 Khó khăn

- Dữ liệu quá lớn khiến cho việc xử lý trở nên khó khăn.
- Trong thời gian hạn chế phương pháp học sâu chưa được tìm hiểu được kỹ nên vẫn còn thiếu sót.

4.3 Hướng phát triển

Một số hướng phát triển cho đồ án chuyên ngành này bao gồm:

- Tìm hiểu xử lý tốt hơn cho dữ liệu lớn.
- Nghiên cứu sâu hơn phương pháp LSTM.
- Nghiên cứu thêm các phương pháp học sâu khác như BERT, PhoBERT,...
- Kế thừa nghiên cứu hiện tại trong đồ án chuyên ngành để thực hiện các nghiên cứu cho các bài toán NLP khác.

Bibliography

[1] URL: https://viblo.asia/p/phan-loai-van-ban-tu-dong-bang-machine-learning-nhu-the-nao-4P856Pa1ZY3.

[2] URL: https://nttuan8.com/bai-14-long-short-term-memory-lstm/.

[3] URL: http://machinelearningcoban.com/2017/04/09/smv/?fbclid=IwAR3WuujHgwSl7RAX5fSOekInGxrGd1orXBiYPFcD7cVNDSCNf8yMFZG24ng.

[4] URL: https://machinelearningcoban.com/2017/08/08/nbc/?fbclid=IwAR08omPuzbEi2SCW1w2QxfUCJ9t6uHya98dkppSNf0cOf5r8uBFPmBwwNsA.

[5] URL: https://github.com/stopwords/vietnamese-stopwords.

[6] URL: https://nguyenvanhieu.vn/phan-loai-van-ban-tieng-viet/?fbclid=IwAR0XAYawQpEL6JbQPNNcBSORkoryjSVoNwNmZon9a3CA2HqUUXC4SnNO81k#phan-loai-van-ban-voi-svm.

[7] URL: https://codetudau.com/lam-quen-pipeline-sklearn-python/?fbclid=IwAR2Z8TAznRVfVIzHDY-LVICNqD8vt4J8RvpTOxW72YEWz-25ZkoZxaaux_s.

[8] URL: https://machinelearningcoban.com/2017/08/31/evaluation/?fbclid=IwAR0dG0014tkDP7WV33rJHVX85r6edgagH-OeLIG7s1HkFe0Nnrd8gA3M2Xw#-precision-va-recall.