**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

─────── \* ───────



**BÁO CÁO BÀI TẬP LỚN**

**Môn:** **Nhập môn Học máy và Khai phá dữ liệu**

**Đề tài:**

Phân loại văn bản Tiếng Việt với Machine Learning

Nhóm thực hiện: **Đàm Trọng Tuyên – 20173463 (Nhóm trưởng)**

**Lê Hải Nam – 20173264**

**Lê Anh Thành – 20170115**

**Trần Tất Đắc – 20172992**

Lớp: **IT3190**

Giảng viên hướng dẫn: **TS. Nguyễn Nhật Quang**

***Hà Nội, tháng 6 năm 2020***

**Mục lục**

[**I. Tổng quan về bài toán thực tế** 3](#_Toc41851843)

[**I.1. Bài toán cụ thể giải quyết.** 3](#_Toc41851844)

[**I.2. Mục tiêu mong muốn.** 3](#_Toc41851845)

[**II. Các phương pháp giải quyết bài toán** 3](#_Toc41851846)

[**II.1. Tập dữ liệu** 3](#_Toc41851847)

[**II.2. Các phương pháp sử dụng** 5](#_Toc41851848)

[**II.2.1. Sơ lược các phương pháp học máy sử dụng vào bài toán** 5](#_Toc41851849)

[**II.2.2. Xử lý dữ liệu đầu vào** 5](#_Toc41851850)

[**III. Kết quả thử nghiệm trên các phương pháp học máy** 7](#_Toc41851851)

[**III.1. Naïve Bayes** 7](#_Toc41851852)

[**III.1.1. Chi tiết phương pháp áp dụng** 7](#_Toc41851853)

[**III.1.2. Dữ liệu đầu vào và điều chỉnh tham số** 8](#_Toc41851854)

[**III.1.3. Đánh giá hiệu năng** 10](#_Toc41851855)

[**III.2. SVM** 12](#_Toc41851856)

[**III.2.1. Chi tiết phương pháp áp dụng** 12](#_Toc41851857)

[**III.2.2. Dữ liệu đầu vào và điều chỉnh tham số** 14](#_Toc41851858)

[**III.2.3. Đánh giá hiệu năng** 16](#_Toc41851859)

[**III.3. Logistic Regression** 18](#_Toc41851860)

[**III.3.1. Chi tiết phương pháp** 18](#_Toc41851861)

[**III.3.2. Dữ liệu đầu vào và điều chỉnh tham số** 19](#_Toc41851862)

[**III.3.3. Đánh giá hiệu năng** 21](#_Toc41851863)

[**IV. Tổng kết** 23](#_Toc41851864)

[**IV.1. Chức năng hệ thống** 23](#_Toc41851865)

[**IV.1.1. Phân loại văn bản trên trang web dựa trên đường dẫn đầu vào** 23](#_Toc41851866)

[**IV.1.2. Phân loại văn bản người dùng nhập** 23](#_Toc41851867)

[**IV.1.3. Quan sát chi tiết kết quả mà hệ thống phân loại** 24](#_Toc41851868)

[**IV.1.4. Gán nhãn lại dữ liệu** 24](#_Toc41851869)

[**IV.2. So sánh, đánh giá giữa các phương pháp sử dụng** 25](#_Toc41851870)

[**IV.3. Vấn đề, khó khăn gặp phải** 26](#_Toc41851871)

[**IV.4. Các hướng phát triển sau này** 26](#_Toc41851872)

[**IV.5. Tài liệu tham khảo** 26](#_Toc41851873)

# **I. Tổng quan về bài toán thực tế**

Bài toán tổng quát là text classification. Đây là một bài toán kinh điển trong lĩnh vực học máy; cụ thể là trong nhánh Natural Language Processing (NLP). Mục tiêu là phân loại dữ liệu dưới dạng văn bản (text) vào các lớp danh mục (category).

Bài toán được sử dụng trong nhiều ứng dụng thực tế như: phân loại trang web, phân loại email, phân loại ý định người dùng (intent classification), phân loại sắc thái bình luận…

## **I.1. Bài toán cụ thể giải quyết.**

Xây dựng được một hệ thống có thể tự động phân loại được một văn bản vào chủ đề, thể loại mà nó thuộc về trong các thể loại: Chính trị xã hội, Đời sống, Khoa học, Kinh doanh, Pháp luật, Sức khoẻ, Thế giới, Thể thao, Văn hoá, Vi tính.

## **I.2. Mục tiêu mong muốn.**

- Phân loại đúng, chính xác

- Xử lý trong thời gian cho phép

- Kịch bản ứng dụng

1. Thực hiện đưa vào hệ thống một văn bản

2. Hệ thống thực hiện xử lý văn bản

3. Đưa ra kết quả phân loại cho văn bản đầu vào

# **II. Các phương pháp giải quyết bài toán**

## **II.1. Tập dữ liệu**

* Tập dữ liệu được cung cấp bởi: <https://github.com/duyvuleo/VNTC>
* Tập dữ liệu bao gồm 84132 văn bản Tiếng việt chưa được xử lý, gồm 10 thể loại: Chính trị xã hội, Đời sống, Khoa học, Kinh doanh, Pháp luật, Sức khoẻ, Thế giới, Thể thao, Văn hoá, Công nghệ máy tính.
* Tập dữ liệu đã được phân lớp ground truth và chia sẵn thành:

+ Tập train: 33759 văn bản

+ Tập test: 50373 văn bản

* Phân bố dữ liệu từng lớp:

+ Tập train:

+ Tập test:

Nhận xét về tập dữ liệu: Sau khi chọn lựa ngẫu nhiên một số lượng dữ liệu để quan sát ta thấy tập dữ liệu không thực sự ‘sạch’, còn xuất hiện một số lỗi chính tả, thiếu dấu câu, gán nhãn dữ liệu còn nhập nhằng.

## **II.2. Các phương pháp sử dụng**

### **II.2.1. Sơ lược các phương pháp học máy sử dụng vào bài toán**

Sử dụng ba phương pháp học máy để thực hiện phân loại văn bản: Naïve Bayes, Support Vector Machine (SVM) và Logistic Regression.

* Naïve Bayes: Phương pháp học dựa trên xác suất, nền tảng là định lý Bayes:

Ta sẽ dựa vào xác suất các từ (tần suất xuất hiện) xuất hiện trong văn bản để tính toán phân loại.

* SVM:

SVM là một phương pháp phân lớp tuyến tính (linear classifier), để xác định một siêu phẳng (hyperplane) để phân tách hai lớp của dữ liệu – ví dụ: lớp các ví dụ có nhãn dương (positive) và lớp các ví dụ có nhãn âm (negative). Mục tiêu của SVM là xác định siêu phẳng sao cho lề của siêu phẳng phân tách đó là lớn nhất có thể.

Vì SVM chỉ là phương pháp phân loại kinh điển đối với các bài toán 2 nhãn lớp nên ta sẽ sử dụng Multi-class SVM áp dụng vào bài toán phân loại 10 lớp cần giải quyết.

* Logistic Regression:

Logistic Regression là phương pháp phân lớp tuyến tính, sử dụng cho phân lớp nhị phân. Nhưng ta có thể sử dụng để phân nhiều lớp.

=> Báo cáo sẽ sử dụng 3 phương pháp khác nhau, thực hiện tối ưu trên cùng một tập dữ liệu từ đó sẽ đưa ra kết quả so sánh và đánh giá giữa 3 phương pháp.

### **II.2.2. Xử lý dữ liệu đầu vào**

a. Phân chia dữ liệu:

* Dữ liệu sẽ được chia lại thành 3 tập train, valid và test. Trong đó tập train và valid sẽ được lấy ngẫu nhiên từ tập train ban đầu với tỷ lệ train- 80% và valid- 20% (mục đíc để giữ được tính khái quát của cho mô hình); tập test giữ nguyên là tập test ban đầu.
* Ta sẽ sử dụng tập train để huấn luyện các model và valid để tối ưu tham số. Và sử dụng model tốt nhất để thử nghiệm đánh giá trên tập test.

b. Tiền xử lý dữ liệu (phần chung cho cả 3 mô hình sử dụng):

* **B1**: Xử lý các từ viết tắt trong văn bản gốc: thay thế bằng các từ đầy đủ. Sử dụng tập từ điển từ viết tắt (có tự bổ sung) được cung cấp bởi: <https://sites.google.com/site/ngo2uochung/research/dsviettat-tieng-viet?fbclid=IwAR2DxsB-DSZECfDD1r8gWJqynctvA2vYefTi52d-TVPt8Dc4LejqPR2AETQ>

Bước này có thể giúp giảm số lượng tập từ điển (xuất hiện thêm cả từ gốc và từ viết tắt), giúp cho tập từ điển đúng đắn hơn.

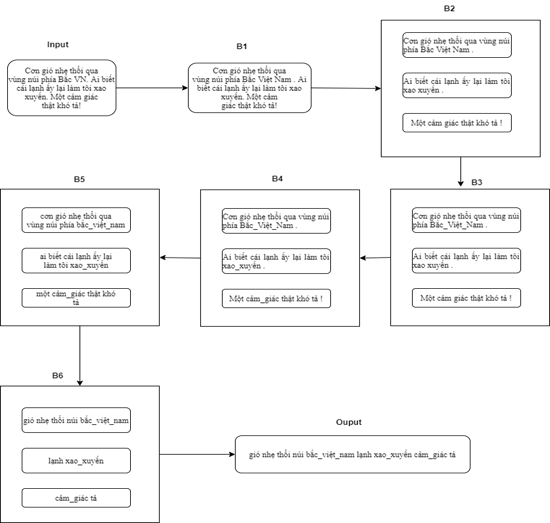
* **B2**: Thực hiện tách văn bản thành tập các câu (sử dụng thư viện nltk).

Bước này giúp việc xử lý dễ dàng hơn khi chia nhỏ văn bản và thực hiện các bước sau chính xác hơn. Các bước sau sẽ xử lý trên từng câu.

* **B3**: Thực hiện nối các từ tên riêng (chuỗi từ viết hoa liên tiếp) thành một từ trước khi ‘tokenize’.

Bước này giúp cho giảm thiểu xuất hiện các từ hiểu sai nghĩa và tăng khả năng tokenize cho từ tên riêng, địa danh. Ví dụ: rừng Đa Mi -> rừng\_đa mi (nếu không xử lý).

* **B4**: ‘Tokenize’: thực hiện xác định các từ trong văn bản (sử dụng thư viện pyvi).
* **B5**: Loại bỏ dấu câu, chữ số (sử dụng thư viện gensim).
* **B6**: Loại bỏ các từ ‘stop words’- các từ xuất hiện nhiều mà không mang nhiều ý nghĩa. Sử dụng tập từ stop words (có tự bổ sung) cung cấp bởi: <https://github.com/stopwords/vietnamese-stopwords?fbclid=IwAR1yVLDgu2bzy2NvXp3ARMO45CcZYC_Jvz14ZyRf-GmDaFXzTfiUAdRer14>
* Đầu ra là tập các văn bản chỉ gồm các từ Tiếng việt đã được xác định.

Ví dụ:

Sau đó các mô hình sẽ sử dụng kĩ thuật *Bag of words* để chuyển văn bản sang dạng vector kết hợp sử dụng với 2 phương pháp thí nghiệm: Count vectorizer và Tfidf vectorizer.

+ Bag of words: biểu diễn một văn bản dưới dạng tập (bag) các từ xuất hiện trong nó kèm theo là số lần từ đó xuất hiện, phương pháp này không quan tâm tới thứ tự hay ngữ pháp của các từ.

+ Count vectorizer (count = **count words**): Chuyển văn bản thành một vector v chiều (v: số lượng từ trong từ điển) với giá trị của các chiều là tần suất xuất hiện trong văn bản đó của một từ trong tập từ điển thu được từ toàn bộ văn bản của tập dữ liệu đầu vào.

+ Tfidf vectorizer (tfidf = **term frequency–inverse document frequency**):

Ý tưởng: Giảm tầm quan trọng của các từ xuất hiện nhiều trong một văn bản và cũng xuất hiện nhiều trong các văn bản khác (các từ stopword), đồng thời tăng sự ảnh hưởng của các từ xuất hiện nhiều trong một văn bản và ít trong các văn bản khác. Thực hiện bằng cách điều chỉnh giá trị khi vector hóa một văn bản.

1. Tf: tần suất xuất hiện của từ trong từ điển trong một văn bản xem xét.

= tần suất xuất hiện từ *t* trong văn bản *d*

2. Idf: giá trị điều chỉnh cho từ *t* theo số văn bản mà từ đó xuất hiện, tính theo:

Trong đó: *n* là tổng số lượng văn bản.

df(*t*) là số lượng văn bản mà từ t xuất hiện.

3. : Giá trị đầu ra của chiều ứng với thứ tự của từ *t* trong từ điển của văn bản *d* sau khi vector hóa.

# **III. Kết quả thử nghiệm trên các phương pháp học máy**

## **III.1. Naïve Bayes**

### **III.1.1. Chi tiết phương pháp áp dụng**

* Naïve Bayes là phương pháp học dựa trên xác suất, nền tảng là định lý Bayes:

Tập dữ liệu D = {} với = ()

Tập nhãn lớp y = {}

* Giả sử các là độc lập có điều kiện:

Trong đó:

: Xác suất tiên nghiệm của việc quan sát được dữ liệu .

: Xác suất tiên nghiệm của giả thiết .

: Xác suất của việc quan sát được dữ liệu , nếu biết giả thiết là đúng.

: Xác suất của giả thiết là đúng, nếu quan sát được dữ liệu

* Với một dữ liệu mới z, phương pháp sẽ xác định phân lớp phù hợp nhất với z bằng cách tối ưu hàm mục tiêu:

* Áp dụng vào bài toán phân loại văn bản cần giải quyết:

Tập dữ liệu D = {} trích ra tập từ điển T = {}

Tập nhãn lớp C = {} tương ứng với 10 nhãn trong bài toán đề xuất

+ Giai đoạn học:

Tính xác suất tiên nghiệm của lớp : trong đó: số lượng văn bản thuộc lớp

: số lượng văn bản trong tập D

Đối với mỗi từ khóa thuộc T: trong đó

: số lần xuất hiện từ trong lớp

: tổng số lượng từ trong lớp , các từ thuộc T

: số lượng từ trong tập từ điển T

: tham số làm mịn, giúp tránh trường hợp xác suất bằng 0

+ Giai đoạn dự đoán:

Văn bản mới d, trích các từ xuất hiện trong d mà từ đó thuộc tập T thu được tập T\_d

Void mỗi thuộc C tính:

Đưa ra dự đoán là lớp sao cho () là max

### **III.1.2. Dữ liệu đầu vào và điều chỉnh tham số**

- Dữ liệu đầu vào:

Tập dữ liệu sử dụng là tập nêu trong II.2.1.a.

Dữ liệu đầu vào cho mô hình là dữ liệu đầu ra của II.2.1.b.

- Các tham số điều chỉnh (Các hàm được cung cấp bởi thư viện sklearn)

+ Các tham số trong bước tiền xử lý: count vectorizer và tfidf vectorizer.

*min\_df*: Trong quá trình xây dựng tập từ điển loại bỏ các từ có tần suất xuất hiện (tf) nhỏ hơn ngưỡng min\_df (số lượng cụ thể hoặc phần trăm theo văn bản)

*max\_df*: Trong quá trình xây dựng tập từ điển loại bỏ các từ có tần suất xuất hiện (tf) lớn hơn ngưỡng max\_df (số lượng cụ thể hoặc phần trăm theo văn bản)

*sublinear\_tf* cho tf-idf: nhận 2 giá trị True và False, thay vì giá trị tần suất xuất hiện (tf) thì sẽ thay bằng *tf = 1 + log(tf)*

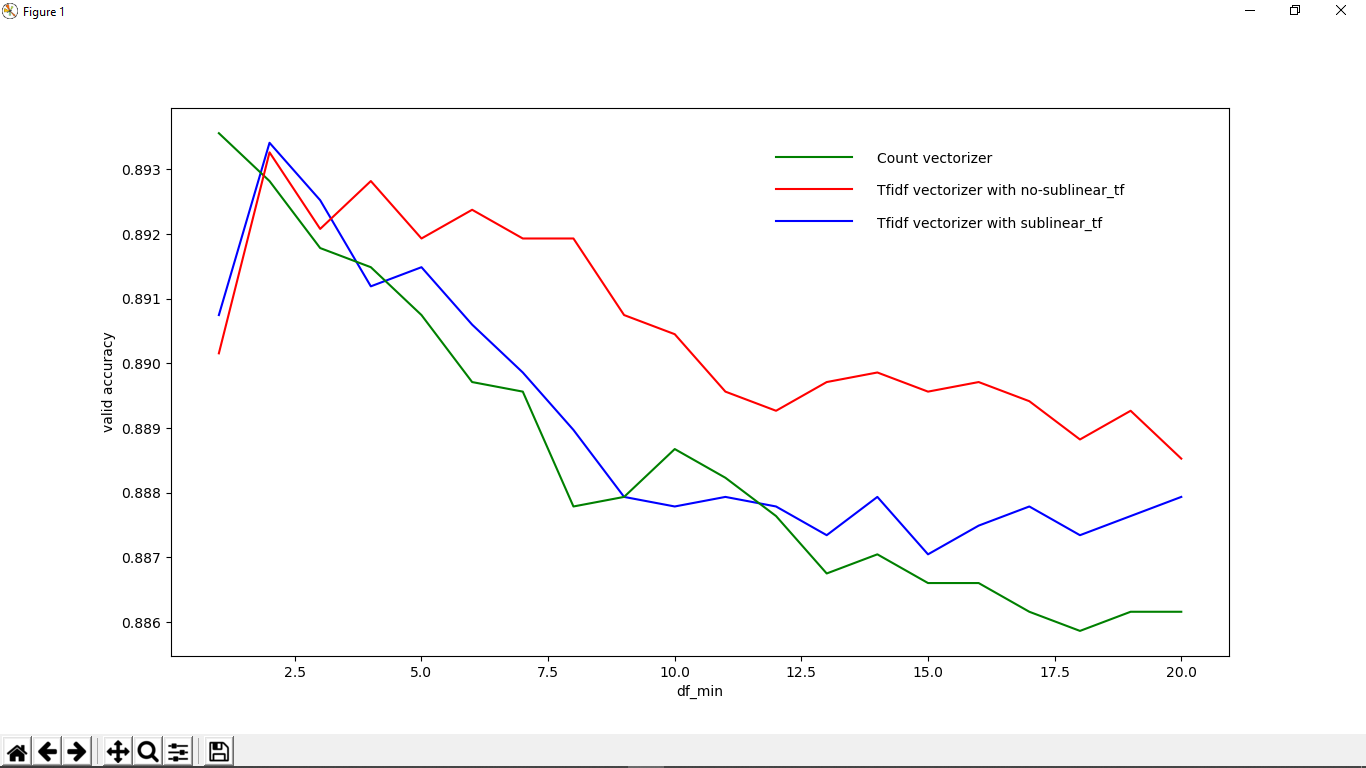
+ Các tham số của mô hình Naïve Bayes:

*alpha*: tham số giúp làm mịn tính toán xác suất , tránh trường hợp xác suất có giá trị bằng 0.

- Quá trình điều chỉnh tham số:

Các giá trị *max\_df* và alpha ta lựa chọn điều chỉnh trong khoảng (0,1) với bước nhảy 0.05, do ở đây ở đây alpha được khuyến cáo nên để trong khoảng (0,1) và *max\_df* ở đây có thể để giá trị là số nguyên nhưng ta khó xác định được số lượng tập từ điển trích xuất được nên sẽ để dưới dạng phần trăm (số float); hơn nữa ta đã thực hiện loại bỏ phần lớn các từ stopwords trong bước tiền xử lý trước đó nên trong thực nghiệm điều với một số trường hợp cách nhau 0.01 kết quả gần như không thay đổi.

Các từ cần loại bỏ nhờ trọng số *min\_df* là phần ta chưa nắm bắt được nên sẽ điều chỉnh theo ngưỡng số nguyên (các từ có tf < *min\_df* thì loại bỏ). Ta quan sát kết quả giá trị *valid accuracy* khi điều chỉnh *min\_df* trong khoảng giá trị (1, 20) thể hiện ở biểu đồ bên dưới:



Ta nhận thấy giá trị *valid accuracy* có xu hướng giảm khi min\_df càng lớn trong cả hai phương pháp vector hóa dữ liệu văn bản, như vậy ta có thể đánh giá việc càng tăng min\_df thêm nữa có thể sẽ khiến độ chính xác trên tập valid giảm.

* Ta tạm dừng việc điều chỉnh tham số và lấy ra các kết quả tốt nhất.

Trong quá trình thực nghiệm ta nhận thấy các kết quả cho độ chính xác trên tập valid cao ứng với:

+ Tfidf vectorizer sublinear-tf = False: min\_df = 2-8, max\_df = 0.15- 0.3, alpha = 0.05 cho ra valid accuracy = 0.8919 – 0.8932.

+ Tfidf vectorizer sublinear-tf = True: min\_df = 2-5, max\_df = 0.1- 0.15, alpha = 0.05 cho ra valid accuracy = 0.8911 – 0.8934.

+ Count vectorizer: min\_df = 1-4, max\_df = 0.1- 0.15, alpha = 0.05 cho ra valid accuracy = 0.8914 – 0.8935.

Ta tiếp tục điều chỉnh tham số trong các khoảng lân cận với các giá trị tốt ở trên để lấy ra kết quả tốt nhất cuối cùng:

+ Điều chỉnh cho Tfidf vectorizer sublinear-tf = False: min\_df = 2-8, max\_df = 0.11- 0.39 bước nhảy 0.01, alpha = 0.01 – 0.09 bước nhảy 0.01.

+ Điều chỉnh cho Tfidf vectorizer sublinear-tf = True: min\_df = 2-5, max\_df = 0.06- 0.19 bước nhảy 0.01, alpha = 0.01 – 0.09 bước nhảy 0.01.

+ Điều chỉnh cho Count vectorizer: min\_df = 1-4, max\_df = 0.01- 0.19 bước nhảy 0.01, alpha = 0.01 – 0.09 bước nhảy 0.01.

* Lấy ra kết quả tốt nhất cuối cùng, dừng quá trình điều chỉnh tham số.

### **III.1.3. Đánh giá hiệu năng**

Tất cả các thử nghiệm, quá trình huấn luyện, dự đoán được thực hiện trên Google Colaboratory, ngôn ngữ sử dụng Python3.

Mô hình cuối cùng: Tfidf vectorizer(sublinear\_tf = False, min\_df = 2, max\_df = 0.14), alpha = 0.01 -> train\_acc = 0.947230039994075, valid\_acc = 0.8956328645447816

a. Tính chính xác.

Độ chính xác trên tập test: 89.83582474738451%

- Các thông số đánh giá trên tập dữ liệu test:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| Chinh tri Xa hoi | 0.82 | 0.88 | 0.85 | 7567 |
| Doi song | 0.71 | 0.70 | 0.70 | 2036 |
| Khoa hoc | 0.81 | 0.72 | 0.76 | 2096 |
| Kinh doanh | 0.92 | 0.86 | 0.89 | 5276 |
| Phap luat | 0.89 | 0.90 | 0.89 | 3788 |
| Suc khoe | 0.92 | 0.93 | 0.92 | 5417 |
| The gioi | 0.94 | 0.91 | 0.92 | 6716 |
| The thao | 0.99 | 0.96 | 0.97 | 6667 |
| Van hoa | 0.90 | 0.95 | 0.92 | 6250 |
| Vi tinh | 0.93 | 0.93 | 0.93 | 4560 |
| accuracy |  |  | 0.90 | 50373 |
| macro avg | 0.88 | 0.87 | 0.88 | 50373 |
| weighted avg | 0.90 | 0.90 | 0.90 | 50373 |

- Bảng bên dưới thể hiện số lượng văn bản với nhãn đúng thuộc lớp I bị dự đoán vào lớp J tương ứng với hàng I và cột J (confusion matrix)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Chinh tri Xa hoi | Doi song | Khoa hoc | Kinh doanh | Phap luat | Suc khoe | The gioi | The thao | Van hoa | Vi tinh |
| Chinh tri Xa hoi | 6631 | 145 | 57 | 144 | 215 | 112 | 109 | 16 | 99 | 39 |
| Doi song | 94 | 1430 | 56 | 8 | 20 | 72 | 24 | 7 | 308 | 17 |
| Khoa hoc | 125 | 167 | 1513 | 20 | 1 | 156 | 39 | 6 | 33 | 36 |
| Kinh doanh | 388 | 32 | 13 | 4545 | 54 | 13 | 52 | 3 | 19 | 157 |
| Phap luat | 257 | 32 | 0 | 41 | 3397 | 16 | 5 | 17 | 13 | 10 |
| Suc khoe | 119 | 55 | 79 | 31 | 7 | 5056 | 47 | 8 | 10 | 5 |
| The gioi | 153 | 57 | 68 | 80 | 50 | 68 | 6097 | 18 | 98 | 27 |
| The thao | 55 | 24 | 6 | 1 | 59 | 11 | 22 | 6420 | 59 | 10 |
| Van hoa | 144 | 67 | 33 | 4 | 16 | 8 | 48 | 5 | 5919 | 6 |
| Vi tinh | 107 | 19 | 39 | 50 | 11 | 6 | 45 | 7 | 31 | 4245 |

b. Tính hiệu quả.

Chi phí về thời gian: Thấp. Quá trình huấn luyện trung bình trong 0.061s. Dự đoán trung bình trong 0.078s.

Bộ nhớ: Sử dụng 441536 bytes để lưu giữ liệu đầu vào trước khi xử lý

c. Khả năng xử lý nhiễu.

Tương đối.

Do tập dữ liệu đầu vào còn nhập nhằng, lỗi chính tả, thiếu dấu câu, gán nhãn dữ liệu còn nhập nhằng (do dữ liệu lớn, nguồn lực có hạn nên khó có thể chỉnh sửa thủ công hết).

Tiền xử lý vẫn chưa xử lý được hết các trường hợp đặc biệt: tập từ viết tắt chưa đầy đủ, xử lý ghép từ tên riêng, địa danh có thể gây ra các từ không chính xác.

Phương pháp Naïve Bayes chủ yếu dựa vào xác suất của từ xuất hiện trong văn bản để đánh giá phân lớp -> có những văn bản tuy thuộc lớp A nhưng bị nhận nhầm vào B là do các từ trong văn bản A phần lớn là xuất hiện ở lớp B.

d. Khả năng mở rộng.

Tốc độ huấn luyện và dự đoán không thay đổi nhiều khi thử nghiệm đối với tập dữ liệu 3000- 30000, do việc học và dự đoán chỉ dựa trên tính toán xác suất đơn giản và lưu lại -> Có khả năng mở rộng thêm mới dữ liệu học mà không mất thêm nhiều chi phí.

e. Khả năng diễn giải.

Dựa trên các công thức xác suất cơ bản, dễ hiểu và nắm bắt. Các kết quả, hoạt động của hệ thống có thể dễ dàng giải thích.

f. Mức độ phức tạp.

Mô hình đơn giản, dễ hiểu. Quá trình học đơn giản chỉ là lưu lại giá trị xác suất cần tính toán của tập dữ liệu đầu vào, và sử dụng các xác suất đó để dự đoán dữ liệu quan sát mới.

## **III.2. SVM**

### **III.2.1. Chi tiết phương pháp áp dụng**

* SVM là phương pháp học có một nền tảng lý thuyết chặt chẽ, được dựa trên nhiều định lý toán học. SVM dùng để xác định mặt phẳng phân tách có lề lớn nhất có thể:

Phương pháp SVM kinh điển ban đầu chỉ dùng cho bài toán phân loại hai lớp

Tập dữ liệu D = {} với = ()

Tập nhãn lớp y = {} trong đó

Mục tiêu của SVM là xác định một hàm phân tách tuyến tính, hay cụ thể hơn là một vec-tơ trọng số các thuộc tính tương ứng với hàm phân tách tuyến tính để phân loại

Trong đó: là hàm phân tách tuyến tính, là vec-tơ trọng số các thuộc tính, là một giá trị số thực.

Đối với mỗi ví dụ , nếu có thì ví dụ mang nhãn , ngược lại ví dụ có . Tương tự như vậy, khi phân loại, ta cũng dùng hàm mục tiêu như trên để phân loại cho mỗi ví dụ đó.

Để làm được việc đó, SVM xác định hai siêu phẳng lề song song thỏa mãn:

sao cho thỏa mãn điều kiện:

nếu còn nếu với bất kì ví dụ nào.

Mức lề đước tính chính bằng khoảng cách giữa hai siêu phẳng lề đó, theo tính toán khoảng cách đó là:

Khi đó bài toán sẽ trở thành bài toán tối ưu bậc hai sau đây: Tìm sao cho:

đạt giá trị cực đại với điều kiện:

với bất kì ví dụ huấn luyện nào.

Bài toán đó tương đương với bài toán cực tiểu hóa có ràng buộc sau:

Cực tiểu hóa:

Với điều kiện: đối với mọi ví dụ.

Từ đó sử dụng lý thuyết tối ưu có ràng buộc, chuyển sang dạng biểu thức Lagrange để tối ưu, và theo điểu kiện Karush-Kuhn-Tucker để chỉ ra các điều kiện cho một lời giải tối ưu.

Tuy nhiên việc giải quyết bài toán này vẫn rất khó khăn, phương pháp Lagrange đã chuyển việc giải quyết bài toán tối ưu hàm lồi sang đến một biểu thức đối ngầu dễ giải quyết hơn rất nhiều so với bài toán ban đầu. Bài toán đối ngẫu đó trở thành:

Cực đại hóa:

Với điều kiện:

Tối ưu bài toán trên cùng với tập điều kiện Karush-Kuhn-Tucker, ta sẽ xác định được các giá trị thỏa mãn.

Khi đó

chính là siêu phẳng phân tách cần tìm.

* SVM là phương pháp phân tách tuyến tính. Tuy nhiên trong thực tế, nhiều khi tập dữ liệu không phải là phân tách tuyến tính, ví dụ tập đó có thể chứa nhiễu lỗi, thì SVM không phân tách được.

Do đó SVM được cải tiến để cho phép chấp nhận một mức độ lỗi nào đó. Phương pháp đó được gọi là Soft-margin SVM.

So với phương pháp SVM truyền thống, phương pháp này vẫn giữ nguyên hàm phân tách tuyến tính, tuy nhiên các điều kiện lề đã được nới lỏng hơn.

Trong hàm mục tiêu cần tối ưu của Soft-margin SVM, có xuất hiện hằng số , đây là hằng số dùng để xác định mức độ chi phí đối với các lỗi. Cụ thể hơn, chúng là hằng số cho phép học mô hình mà chịu một mức độ sai số nhất định.

* Đối với những tập không phân tách tuyến tính, thường có dạng phân tách là phi tuyến, có một phương pháp cải tiến SVM để có thể áp dụng vào bài toán phân tách đó. Đó là sử dụng hàm nhân (kernel), qua hàm nhân, tập các ví dụ được biến đổi vào một không gian nhiều chiều hơn, và trong không gian đó chúng phân tách tuyến tính được, và lúc đó có thể áp dụng được SVM. Phương pháp này là Kernel SVM.
* Cả SVM và Soft-margin SVM đều chỉ là phương pháp phân loại hai lớp. Để có thể phân loại cho nhiều lớp hơn. Ta cần dùng một phương pháp cải tiến của SVM, đó là Multi-class SVM.

Phương pháp này dùng chiến lược one-vs-rest, nghĩa là phân tách giữa bên có nhãn và bên không mang nhãn . Vì thế để phân loại ra lớp, ta phải học ra hàm phân tách tuyến tính để phân loại từng lớp một.

* Áp dụng vào bài toán phân loại văn bản cần giải quyết:

Tập dữ liệu D = {} trích ra tập từ điển T = {}

Tập nhãn lớp C = {} tương ứng với 10 nhãn trong bài toán đề xuất

+ Giai đoạn học:

Sử dụng phương pháp Multi-class SVM, mô hình cần phải học được 10 bộ phân loại để phân nhãn lớp.

Xác định giá trị của các tham số trong mô hình dựa vào tính toán tối ưu bài toán đối ngẫu với hệ điều kiện đã định. Từ đó xác định ra các bộ tham số để tính toán phân lớp.

+ Giai đoạn dự đoán:

Văn bản mới d, trích các từ xuất hiện trong d mà từ đó thuộc tập T thu được tập T\_d.

Văn bản được đưa qua mô hình đã được học, và được tính toán xem phù hợp nhất với nhãn nào để được phân loại.

### **III.2.2. Dữ liệu đầu vào và điều chỉnh tham số**

- Dữ liệu đầu vào:

Tập dữ liệu sử dụng là tập nêu trong II.2.1.a.

Dữ liệu đầu vào cho mô hình là dữ liệu đầu ra của II.2.1.b.

- Các tham số điều chỉnh (Các hàm được cung cấp bởi thư viện sklearn)

+ Các tham số trong bước tiền xử lý: count vectorizer và tfidf vectorizer.

*min\_df*: Trong quá trình xây dựng tập từ điển loại bỏ các từ có tần suất xuất hiện (tf) nhỏ hơn ngưỡng min\_df (số lượng cụ thể hoặc phần trăm theo văn bản)

*max\_df*: Trong quá trình xây dựng tập từ điển loại bỏ các từ có tần suất xuất hiện (tf) lớn hơn ngưỡng max\_df (số lượng cụ thể hoặc phần trăm theo văn bản)

*sublinear\_tf* cho tf-idf: nhận 2 giá trị True và False, thay vì giá trị tần suất xuất hiện (tf) thì sẽ thay bằng tf = 1 + log(tf)

*use\_idf:* Xác định xem có sử dụng đánh giá lại idf không.

+ Các tham số của mô hình SVM:

*C*: tham số xác định mức độ chi phí đối với các lỗi, luôn dương.

*kernel:* tham số xác định hàm nhân được sử dụng trong thuật toán. Tham số này phải là một trong các tham số ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’.

*gamma:* hệ số kernel cho các mô hình ‘rbf’, ‘poly’ và ‘sigmoid’.

- Quá trình điều chỉnh tham số:

Trong mô hình sử dụng SVM này, ta chủ yếu sẽ điều chỉnh các tham số *kernel* (hàm nhân), và *C* (tham số xác định mức độ chi phí đối với các lỗi) cho cả mô hình sử dụng tiền xử lý Count vectorizer và Tfidf vectorizer.

Tham số *kernel* khi chọn bằng thì xảy ra hiện tượng overfitting, còn khi chọn bằng thì xảy ra hiện tượng underfitting. Đối với hai trường hợp còn lại là và , kết quả cho khá tương đồng, nhưng thời gian huấn luyện của ngắn hơn đáng kể so với nên ta lựa chọn tham số *kernel*

Dưới đây là các kết quả khi thay đổi tham số trong các mô hình khác nhau, ta so sánh giá trị để chọn ra một mô hình tốt.

Trên đây là biểu đồ sự phụ thuộc của khi thay đổi trong mô hình SVM với thông qua xử lý dùng phương pháp . Tại đây đạt giá trị lớn nhất bằng 0.899629903774981với giá trị .

Còn đây là biểu đồ thể hiện sự thay đổi của khi thay đổi trong mô hình SVM với , các văn bản được xử lý thông qua phương pháp . Tại đây đạt giá trị lớn nhất bằng 0.915321983715766 với giá trị .

### **III.2.3. Đánh giá hiệu năng**

Tất cả các thử nghiệm, quá trình huấn luyện, dự đoán được thực hiện trên Google Colaboratory, ngôn ngữ sử dụng Python3.

Mô hình cuối cùng: Tfidf vectorizer, sublinear\_tf = False, use-idf = true, kernel = linear, gamma = auto, C = 6.5, train\_acc = 0.9634498592801066, valid\_acc = 0.9144337527757217

a. Tính chính xác.

Độ chính xác trên tập test: 91.32670279713339%

- Các thông số đánh giá trên tập dữ liệu test:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| Chinh tri Xa hoi | 0.84 | 0.89 | 0.87 | 7567 |
| Doi song | 0.75 | 0.71 | 0.73 | 2036 |
| Khoa hoc | 0.83 | 0.77 | 0.80 | 2096 |
| Kinh doanh | 0.93 | 0.88 | 0.90 | 5276 |
| Phap luat | 0.90 | 0.91 | 0.91 | 3788 |
| Suc khoe | 0.93 | 0.94 | 0.93 | 5417 |
| The gioi | 0.95 | 0.92 | 0.94 | 6716 |
| The thao | 0.98 | 0.98 | 0.98 | 6667 |
| Van hoa | 0.93 | 0.95 | 0.94 | 6250 |
| Vi tinh | 0.94 | 0.96 | 0.95 | 4560 |
| accuracy |  |  | 0.91 | 50373 |
| macro avg | 0.90 | 0.89 | 0.89 | 50373 |
| weighted avg | 0.91 | 0.91 | 0.91 | 50373 |

- Bảng bên dưới thể hiện số lượng văn bản với nhãn đúng thuộc lớp I bị dự đoán vào lớp J tương ứng với hàng I và cột J (confusion matrix)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Chinh tri Xa hoi | Doi song | Khoa hoc | Kinh doanh | Phap luat | Suc khoe | The gioi | The thao | Van hoa | Vi tinh |
| Chinh tri Xa hoi | 6745 | 80 | 53 | 136 | 203 | 112 | 61 | 30 | 92 | 55 |
| Doi song | 184 | 1451 | 49 | 14 | 19 | 66 | 46 | 15 | 173 | 19 |
| Khoa hoc | 102 | 128 | 1620 | 11 | 2 | 127 | 29 | 5 | 32 | 40 |
| Kinh doanh | 325 | 28 | 22 | 4650 | 56 | 8 | 53 | 7 | 11 | 116 |
| Phap luat | 224 | 16 | 2 | 38 | 3449 | 13 | 3 | 20 | 13 | 10 |
| Suc khoe | 90 | 66 | 88 | 17 | 4 | 5089 | 42 | 10 | 4 | 7 |
| The gioi | 128 | 63 | 57 | 74 | 32 | 46 | 6218 | 18 | 50 | 30 |
| The thao | 17 | 22 | 3 | 3 | 21 | 8 | 12 | 6547 | 31 | 3 |
| Van hoa | 129 | 68 | 15 | 9 | 16 | 17 | 43 | 26 | 5912 | 15 |
| Vi tinh | 55 | 16 | 25 | 40 | 11 | 4 | 29 | 7 | 10 | 4363 |

b. Tính hiệu quả.

Chi phí về thời gian: Khá cao. Quá trình huấn luyện trung bình trên 200s. Dự đoán trung bình trong 0.013s.

Không phù hợp với những mô hình yêu cầu thời gian huấn luyện ngắn.

c. Khả năng xử lý nhiễu.

Tương đối tốt, tuy nhiên phụ thuộc vào việc lựa chọn tham số C, tham số dùng để xác định mức độ chi phí đối với các lỗi.

Do tập dữ liệu đầu vào còn nhập nhằng, lỗi chính tả, thiếu dấu câu, gán nhãn dữ liệu còn nhập nhằng (do dữ liệu lớn, nguồn lực có hạn nên khó có thể chỉnh sửa thủ công hết). Cần phải có một phương pháp để xác định ra những dữ liệu ngoại lai.

Tiền xử lý vẫn chưa xử lý được hết các trường hợp đặc biệt: tập từ viết tắt chưa đầy đủ, xử lý ghép từ tên riêng, địa danh có thể gây ra các từ không chính xác.

Đặc biệt các văn bản có nhãn Chính trị xã hội hay bị nhầm với các nhóm văn bản khác và ngược lại do có sự giao thoa trong lượng từ sử dụng bởi các loại văn bản này. Cần có một phương pháp tiền xử lý hiệu quả hơn.

d. Khả năng mở rộng.

Cả tốc độ huấn luyện và dự đoán đều tăng khá nhanh khi tập dữ liệu tăng. Nên để huấn luyện một tập dữ liệu lớn hơn thì lượng chi phí tăng lên cũng phải tương xứng.

e. Khả năng diễn giải.

Dù có nền tảng lý thuyết chặt chẽ, nhưng cách xác định bởi SVM thường khó hiểu với người dùng, đặc biệt khi sử dụng các hàm nhân.

f. Mức độ phức tạp.

Mô hình tương đối phức tạp, nhưng có nền tảng lý thuyết rõ ràng. Quá trình học cần phải tối ưu hàm mục tiêu dựa trên dữ liệu huấn luyện. Khi thực hiện phân lớp cho ví dụ mới, chỉ cần tính giá trị hàm phân lớp theo mô hình đã học được ở trên.

## **III.3. Logistic Regression**

### **III.3.1. Chi tiết phương pháp**

* Logistics Regression là một mô hình thuộc loại mô hình tuyến tính (bao gồm linear regression, PLA, ...), nó giống với linear regression là đầu ra là số thực và giống với PLA là đầu ra bị chặn trong đoạn [0,1]. Tuy tên là hồi quy (regression) nhưng mô hình này lại hay được dùng trong việc phân lớp (classification).

Với tập dữ liệu và

Đầu ra của mô hình sẽ là xác xuất điểm dữ liệu x rơi vào lớp y:

Trong đó: f là hàm sigmoid

Đặt có

Ta cần tìm trọng số mô hình w để tổng xác xuất đầu ra lớn nhất:

Đây là bài toán MLE, nên cách giải quyết chung là lấy logarit và đổi dấu (negative log likelihood) ta sẽ có hàm mất mát của mô hình:

Dùng Gradient Descent ta sẽ có công thức cập nhật cho mô hình này:

Tuy nhiên, mô hình này là một mô hình phân lớp nhị phân nên chưa phù hợp với bài toán có nhiều hơn 2 lớp. Nên ta cần áp dụng thêm kĩ thuật 1 vs rest để dùng cho bài toán phân lớp với số lớp lớn hơn 2.

### **III.3.2. Dữ liệu đầu vào và điều chỉnh tham số**

- Dữ liệu đầu vào:

Tập dữ liệu sử dụng là tập nêu trong II.2.1.a.

Dữ liệu đầu vào cho mô hình là dữ liệu đầu ra của II.2.1.b.

- Các tham số điều chỉnh (Các hàm được cung cấp bởi thư viện sklearn)

+ Các tham số trong bước tiền xử lý: count vectorizer và tfidf vectorizer:

*min\_df*: Trong quá trình xây dựng tập từ điển loại bỏ các từ có tần suất xuất hiện (tf) nhỏ hơn ngưỡng min\_df (số lượng cụ thể hoặc phần trăm theo văn bản)

*max\_df*: Trong quá trình xây dựng tập từ điển loại bỏ các từ có tần suất xuất hiện (tf) lớn hơn ngưỡng max\_df (số lượng cụ thể hoặc phần trăm theo văn bản)

*sublinear\_tf* cho tf-idf: nhận 2 giá trị True và False, thay vì giá trị tần suất xuất hiện (tf) thì sẽ thay bằng *tf = 1 + log(tf)*

+ Tham số mô hình logistics regression:

*max\_iter:* số vòng lặp tối đa cho quá trình huấn luyện.

- Quá trình điều chỉnh:

* Vì thư viện của sklearn không hỗ trợ việc dừng train model bất cứ lúc nào nên cần điều chỉnh số vòng lặp (epochs)
* Ta sẽ điều chỉnh tham số này trong 2 trường hợp tiền xử lí: Count vectorizer và Tfidf vectorizer.
* Với Count vectorizer, thử với số vòng lặp từ 15 – 25 ta đã thấy được việc accuracy trên tập valid giảm, độ chính xác cao nhất đạt tại iter\_max = 22.

Với Tfidf vectorizer, sau nhiều lần thử ta có độ chính xác trên tập valid:

Tuy với cách tiền xử lí này độ chính xác trên tập valid cao hơn một chút so với cách trước , nhưng chi phí bỏ ra gần như cao gấp đôi vì thế sẽ quyết định lấy model train bằng dữ liệu tiền xử lí bằng Count vectorizervới max\_iter = 22.

### **III.3.3. Đánh giá hiệu năng**

Tất cả các thử nghiệm, quá trình huấn luyện, dự đoán được thực hiện trên Google Colaboratory, ngôn ngữ sử dụng Python3.

Mô hình cuối cùng: Count vectorizer, max\_iter = 22, train\_acc = 97.8%, valid\_acc = 89.94%

a. Tính chính xác

- Độ chính xác trên tập test: 89.79%

- Các thông số đánh giá trên tập test:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | support |
| Chính trị xã hội | 0.82 | 0.83 | 0.83 | 7567 |
| Đời sống | 0.61 | 0.80 | 0.70 | 2036 |
| Khoa Học | 0.73 | 0.81 | 0.77 | 2096 |
| Kinh Doanh | 0.91 | 0.87 | 0.89 | 5276 |
| Pháp Luật | 0.91 | 0.82 | 0.87 | 3788 |
| Sức Khỏe | 0.94 | 0.90 | 0.92 | 5417 |
| Thế Giới | 0.88 | 0.95 | 0.92 | 6716 |
| Thể Thao | 0.96 | 0.98 | 0.97 | 6667 |
| Văn Hóa | 0.96 | 0.89 | 0.92 | 5250 |
| Vi Tính | 0.93 | 0.92 | 0.93 | 4560 |
| accuracy |  |  | 0.89 | 50373 |
| Macro avg | 0.87 | 0.88 | 0.87 | 50373 |
| Weighted avg | 0.90 | 0.89 | 0.89 | 50373 |

- Bảng bên dưới thể hiện số lượng văn bản với nhãn đúng thuộc lớp I bị dự đoán vào lớp J tương ứng với hàng I và cột J (confusion matrix)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Chinh tri Xa hoi | Doi song | Khoa hoc | Kinh doanh | Phap luat | Suc khoe | The gioi | The thao | Van hoa | Vi tinh |
| Chinh tri Xa hoi | 6576 | 238 | 86 | 14 | 263 | 96 | 59 | 17 | 152 | 77 |
| Doi song | 210 | 1264 | 121 | 236 | 128 | 103 | 42 | 20 | 96 | 18 |
| Khoa hoc | 165 | 77 | 1606 | 24 | 1 | 109 | 51 | 6 | 27 | 59 |
| Kinh doanh | 184 | 18 | 18 | 4620 | 42 | 16 | 59 | 0 | 9 | 61 |
| Phap luat | 261 | 31 | 7 | 67 | 3384 | 13 | 51 | 22 | 24 | 14 |
| Suc khoe | 104 | 77 | 135 | 9 | 13 | 4979 | 47 | 7 | 16 | 4 |
| The gioi | 81 | 79 | 46 | 75 | 10 | 59 | 6168 | 22 | 52 | 28 |
| The thao | 27 | 14 | 8 | 6 | 20 | 11 | 27 | 6530 | 22 | 11 |
| Van hoa | 98 | 214 | 38 | 8 | 19 | 22 | 76 | 39 | 5837 | 19 |
| Vi tinh | 61 | 24 | 31 | 117 | 8 | 9 | 36 | 4 | 15 | 4269 |

b. Tính hiệu quả k

* Thời gian huấn luyện khá ngắn :13.1s
* Thời gian dự đoán: 0.04s
* Với thời gian dự đoán ngắn mô hình có thể phục vụ được các tác vụ realtime

c. Khả năng xử lí nhiễu

Tương đối.

Do tập dữ liệu đầu vào còn nhập nhằng, lỗi chính tả, thiếu dấu câu, gán nhãn dữ liệu còn nhập nhằng (do dữ liệu lớn, nguồn lực có hạn nên khó có thể chỉnh sửa thủ công hết).

Tiền xử lý vẫn chưa xử lý được hết các trường hợp đặc biệt: tập từ viết tắt chưa đầy đủ, xử lý ghép từ tên riêng, địa danh có thể gây ra các từ không chính xác.

d. Khả năng mở rộng

Thời gian training khá ngắn với chỉ khoảng 0.5s/epoch, với bộ dữ liệu gồm 30.000 dữ liệu training; việc mở rộng là có thể, nhưng nếu bộ dữ liệu mới mở rộng lớn hơn gấp nhiều lần so với ban đầu thì chi phí tăng khá nhiều.

e. Khả năng diễn giải

Lí thuyết cơ sở của mô hình khá khó hiểu với người dùng thông thường, tuy nhiên có thể giải thích một cách ngắn gọn, dễ hiểu vẫn giúp người dùng nắm bắt được cách mô hình hoạt động.

f. Mức độ phức tạp

Mô hình đơn giản. Số lượng tham số cần train không quá lớn. Quá trình train dựa vào dữ liệu để cập nhất các tham số đó.

Khi thực hiện dự đoán, ta đưa vector đầu vào vào mô hình, mô hình thực hiện nhân vector đầu vào với trọng số mô hình, tiếp tục đưa qua hàm sigmoid để có đầu ra

# **IV. Tổng kết**

## **IV.1. Chức năng hệ thống**

### **IV.1.1. Phân loại văn bản trên trang web dựa trên đường dẫn đầu vào**

A screenshot of a social media post

Description automatically generated

* Hệ thống sẽ crawl dữ liệu trang web mà người dùng đã nhập link, xử lý và trả về kết quả phân loại trang web (vào 1 trong 10 chủ đề cho trước) với confidence tương ứng.

### **IV.1.2. Phân loại văn bản người dùng nhập**

A screenshot of a social media post

Description automatically generated

* Hệ thống nhận đầu vào là đoạn văn bản người dùng nhập, xử lý và trả về kết quả phân loại văn bản (vào 1 trong 10 chủ đề cho trước) với confidence tương ứng.

### **IV.1.3. Quan sát chi tiết kết quả mà hệ thống phân loại**

A screenshot of a cell phone

Description automatically generated

* Chức năng “*more detail”* cho phép người dùng quan sát giá trị trả về của thuật toán phân loại (ở đây sử dụng 3 thuật toán: SVM, Naïve Bayes vàLogistic Regression) phù hợp trong quá trình đánh giá và cải tiến thuật toán.

### **IV.1.4. Gán nhãn lại dữ liệu**

A screenshot of a social media post

Description automatically generated

* “***relabel****”* giúp gán nhãn lại dữ liệu mà người dùng cảm thấy sai, lưu trữ lại cho quá trình tối ưu thuật toán, cải tiến model sau này.

## **IV.2. So sánh, đánh giá giữa các phương pháp sử dụng**

* Về độ chính xác, có thể thấy ba mô hình có độ chính xác tương đồng như nhau (~ 90%) tuy nhiên so với 2 thuật toán còn lại thì Logistic Regression có xu hướng overfit và khó có khả năng cải tiến hơn. Một shallow neural network với 1 tầng ẩn cũng là tầng đầu ra duy nhất với activation function là Sigmoid đã tạo ra được bộ phân loại 2 nhãn (từ đó áp dụng cho bài toán nhiều nhãn). Tuy nhiên, có thể thấy, dữ liệu ban đầu sau khi biểu diễn bằng hai phương pháp *countVectorize* và *tfidfVectorize* chứa khá nhiều nhiễu, thuật toán này có khả năng học trên tập huấn luyện tốt gây ra hiện tượng overfit. Để cải tiến thuật toán này cách tốt nhất có thể là thay đổi cách biểu diễn dữ liệu đầu vào.
* Với giả thiết “naïve” đó là các biến ngẫu nhiên độc lập với nhau Naïve Bayes, một mô hình xác suất cổ điển đã cho thấy khả năng phân loại tốt của nó, không chỉ về độ chinh xác mà còn về cả tốc độ train cũng như dự đoán.
* Một nhược điểm có thể nhận thấy ở phương pháp này đó là việc dự đoán dựa vào tần suất xuất hiện của các từ, khi đó, 1 từ xuất hiện ở nhiều nhãn khác nhau sẽ gây nhập nhằng và đặc biệt sẽ không phát hiện ra chủ đề chính xác của văn bản trong trường hơp văn bản thuộc chủ đề này nhưng chứa nhiều từ thuộc chủ đề khác.
* Tuy nhiên, nhóm nhận thấy về khả năng phát triền, cải tiến thì SVM là mô hình khả quan nhất trong tương lai về khả năng chịu nhiễu, lỗi của nó cũng như sức mạnh của “kernel” trong việc biến dữ liệu trở nên có thể phân tách tuyến tính.

## **IV.3. Vấn đề, khó khăn gặp phải**

* Dữ liệu ban đầu chưa được chuẩn hóa, có chữa cả lỗi chính tả, cú pháp.
* Dữ liệu ban đầu chứa sẵn nhiễu, lỗi và các văn bản nhập nhằng giữa các chủ đề.
* Vấn đề overfit với thuật toán Logistic Regression vẫn chưa thể giải quyết.
* Số lượng data lớn, thời gian có hạn nên việc phân tích lỗi đánh giá cải tiến mô hình gặp khó khăn.

## **IV.4. Các hướng phát triển sau này**

* Tạo bộ dữ liệu training sạch hơn, các nhãn gán chính xác và hạn chế nhiễu.
* Biểu diễn dữ liệu bằng các phương pháp cải tiến hơn (ví dụ: thay vì biểu diễn word vector bằng 1 one hot vector như phương pháp CountVecorizer thì có thể sử dụng các phương pháp word embedding khác như word2vec, fastText, Glove hay gần đây nhất là Bert; một số đã có pretrain model cho tiếng Việt).
* Phân tích chi tiết các lỗi sai gặp phải-các nhãn bị phân loại sai, để có hướng xử lý với từng loại.
* Tích hợp nhiều mô hình để tạo nên một mô hình với độ chính xác cao hơn.
* Sử dụng các phương pháp học sâu, mạng neural dùng cho sử lý chuỗi dữ liệu khi đã thu thập được lượng dữ liệu lớn (RNN, GRU, LSTM…).

## **IV.5. Tài liệu tham khảo**

1. Scikit-learn documentation: <https://scikit-learn.org/stable/modules/classes.html>.
2. Bài giảng môn Nhập môn học máy và khai phá dữ liệu, giảng viên Nguyễn Nhật Quang, Trường Đại học Bách Khoa Hà Nội, Viện công nghệ thông tin và truyền thông.
3. Và một số tài liệu online khác.

|  |  |  |
| --- | --- | --- |
| Họ tên | Mssv | Công việc |
| Đàm TrọngTuyên (Nhóm trưởng) | 20173463 | Code tiền xử lý, thuật toán, demo, reiew báo cáo, trình bày. |
| Lê Hải Nam | 20173264 | Code tiền xử lý, naïve bayes, tối ưu naïve bayes, viết báo cáo. |
| Trần Tất Đắc | 20172992 | Code Logistic Regression, tối ưu Logis tic regression, viết báo cáo. |
| Lê Anh Thành | 20170115 | Code SVM, tối ưu SVM, viết báo cáo. |