ỨNG DỤNG BIG DATA VÀO PHÂN LOẠI BÌNH LUẬN TRÊN LIVESTREAM YOUTUBE

Lê Minh Phước[1,2], Dương Minh Lượng[1,2], and Ngô Minh Phú[1,2]

[1] Trường Đại học Công nghệ Thông tin
[2] Đại học Quốc gia Thành phố Hồ Chí Minh
{18521262,18521071,18521243}@gm.uit.edu.vn

Tóm tắt nội dung Trong những năm gần đây, Việt Nam chứng kiến sự phát triển nhanh chóng của internet, số lượng người dùng mạng xã hội tăng lên đáng kể như Facebook, Youtube, Instagram, Tiktok. Dữ liệu được tạo ra từ các mạng xã hội vì vậy cũng phát triển theo cấp số nhân. Việc bình luận của người dùng rất khó kiểm soát và sự tồn tại của lời nói ác ý khiến không gian mạng xã hội trở nên độc hại. Do đó, một công cụ phân loại các bình luận là điều cần thiết. Nhiệm vụ phát hiện lời nói ác ý trực tuyến sau đó phân tích những lời nói này.

Keywords: Apache Spark · Multiclass classification · Pyspark.

1 Giới thiệu

Với sự phát triển nhanh chóng của internet và các ngành công nghệ trong những năm gần đây, nhiều nền tảng xã hội được ra đợi thu hút hàng tỉ người dùng, với lượng dữ liệu khổng lồ từ đó nếu ta biết cách khai thác thì sẽ giải quyết được rất nhiều bài toán khó, rất khó mà bấy lâu nay ta chưa giải quyết được. Như trong những đợt dịch do covid hoành hành vừa qua, thì với việc cách li tại nhà nên việc sử dụng các nền tảng mạng xã hội để giao tiếp, làm việc cũng như giải trí ngày càng tăng cao tuy nhiên trên các nền tảng xã hội sẽ có một số người thường hay bình luận chửi tục, lăng mạ, công kích người khác,... nhằm thỏa mãn niềm vui của bạn thân mà không nghĩ đến các hậu quả khôn lượng của nó (có thể gây cho người khác cảm thấy khó chịu, stress, trầm cảm, tự tử,...) nên việc cần có một công cụ phân loại các bình luận là cần thiết nhằm tránh ảnh hưởng đến tinh thần, thể chất của dùng trong các nền tảng này.

Trong bài báo này chúng tôi sẽ xây dựng các mô hình phân loại bình luận trực tuyến từ đó phân tích các lời nói này nhằm mong muốn có thể ẩn các bình luận tiêu cực để có thể có một môi trường tốt hơn trên các nền tảng này.

Phần còn lại của bài báo được tóm tắt như sau. Đầu tiên chúng tôi xem xét các nghiên cứu liên quan cho bài toán phân loại và các phương pháp học máy truyền thống. Phần 3 sẽ trình bày tổng quát về bộ dữ liệu dùng cho thực nghiệm. Phần 4 trình bày các phương pháp chúng tôi áp dụng để giải quyết bài

toán. Phần 6 trình bày toàn bộ quá trình thực nghiệm, đánh giá và cuối cùng là sẽ đưa ra lời nhận xét và tổng kết cho bài báo.

2 Nghiên cứu liên quan

Trong phần này, chúng tôi trình bày về bài toán phân loại và các phương pháp học máy sử dụng cho bài toán cần giải quyết.

2.1 Bài toán phân loại nhiều lớp

Trong khi bộ phân loại nhị phân chỉ phân biệt hai lớp, bộ phân loại đa lớp (multiclass classifier, còn được gọi là bộ phân loại đa thức - multinomial classifier) có thể phân biệt nhiều hơn hai lớp.

Một vài bộ phân loại (như SGD, Rừng Ngẫu nhiên và Navie Bayes) có khả năng làm việc với nhiều lớp một cách tự nhiên. Những thuật toán khác (như Hồi quy Logistic hay Máy Vector Hỗ trợ) hoàn toàn là các bộ phân loại nhị phân. Tuy nhiên, có rất nhiều chiến lược cho phép ta sử dụng nhiều bộ phân loại nhị phân cho bài toán phân loại đa lớp.

2.2 Các phương pháp phân loại

Logistic Regression Hồi quy Logistic là một mô hình hồi quy nhằm dự đoán giá trị đầu ra rời rạc. Nguyên lý hoạt động của hồi quy logistic là đưa đầu ra của mô hình hồi quy tuyến tính đi qua một hàm kích hoạt có tên là sigmoid để tất cả giá trị output của hàm giả định sẽ nằm trong khoảng [0,1]. Hàm sigmoid được định nghĩa như sau:

$$f(s) = \frac{1}{1+e^{-s}}\sigma(s) \tag{1}$$

Support Vector Machine Đầu tiên, trong thuật toán SVM có một khái niệm là margin – thuật ngữ chỉ khoảng cách gần nhất từ một điểm dữ liệu tới mặt phân cách giữa các lớp. Bài toán SVM là bài toán đi tìm một siêu phẳng tối ưu mà tại đó margin của các lớp dữ liệu là lớn nhất và bằng nhau.

$$margin = \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2} \tag{2}$$

Margin được tính theo công thức 2; Sau đó tối ưu hóa bằng cách cập nhật lại w,b sao cho margin là lớn nhất:

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \left\{ \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n+b)}{||\mathbf{w}||_2} \right\} = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||_2} \min_n y_n(\mathbf{w}^T\mathbf{x}_n + b) \right\} \tag{3}$$

Naive Bayes là một thuật toán phân lớp được mô hình hoá dựa trên định lý Bayes trong xác suất thống kê. Với $P(y|X)$ là xác suất của mục tiêu y với điều kiện có đặc trưng X (posterior probability), $P(X|y)$ là xác suất của đặc trưng X khi đã biết mục tiêu y (likelihood), $P(y)$ là prior probability của mục tiêu y, $P(X)$ là prior probability của vector đặc trưng $X(x_1, x_2, ..., x_n)$. Khi đó,

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \tag{4}$$

Trong mô hình Naive Bayes, có hai giả thiết được đặt ra:

- Các đặc trưng đưa vào mô hình là độc lập với nhau. Tức là sự thay đổi giá trị của một đặc trưng không ảnh hưởng đến các đặc trưng còn lại.
- Các đặc trưng đưa vào mô hình có ảnh hưởng ngang nhau đối với đầu ra mục tiêu. Khi đó, kết quả mục tiêu y để $P(y|X)$ đạt cực đại trở thành:

$$y = argmax_y P(y) \prod_{i=1}^{N} P(x_i|y) \tag{5}$$

Chính vì hai giả thiết gần như không tồn tại trong thực tế trên, mô hình này mới được gọi là naive (ngây thơ). Tuy nhiên, chính sự đơn giản của nó với việc dự đoán rất nhanh kết quả đầu ra khiến nó được sử dụng rất nhiều trong thực tế trên những bộ dữ liệu lớn, đem lại kết quả khả quan.

Decision Trees Trong khoa học máy tính, cây quyết định được sử dụng như một mô hình dự đoán để đi từ các quan sát (thuộc tính) - đại diện là các nhánh để kết luận về giá trị mục tiêu (được thể hiện trong các lá). Các mục tiêu có thể có một tập hợp các giá trị được gọi là cây phân loại. Đây là một trong những cách tiếp cận mô hình dự đoán được sử dụng trong thống kê, khai thác dữ liệu và học máy. Trong các cấu trúc cây này, lá đại diện cho các nhãn và các nhánh đại diện cho các liên kết của thuộc tính dẫn đến những nhãn lớp. Cây quyết định là một bộ phân loại ở dạng cấu trúc cây, trong đó bao gồm các thành phần chính:

- Node quyết định (Decision node): chỉ định kiểm tra trên một thuộc tính duy nhất.
- Nút lá (Leaf node): cho biết giá trị của thuộc tính mục tiêu.
- Arc / edge: tách một thuộc tính.
- Path: một sự kết hợp để tạo ra quyết định cuối cùng.

Cây quyết định phân loại các trường hợp hoặc ví dụ bằng cách bắt đầu từ rễ của cây và di chuyển đến một nút lá. Có nhiều thuật toán được sử dụng để xây dựng cây quyết định có thể kể đến như: ID3, C4.5, CART, CHAID, MARS, Conditional Inference Trees.

Random Forest Với thuật toán Random Forest, trong mỗi tập dữ liệu, ta có thể xây dựng được nhiều cây quyết định khác nhau. Random Forest sẽ kết hợp các cây quyết định khác nhau đó để tạo ra một mô hình mới. Kết quả đầu ra của mô hình được tổng hợp từ kết quả của các cây quyết định mà nó tạo ra. Tương ứng với mỗi điểm dữ liệu, toàn bộ quá trình dự đoán sẽ được thực hiện trên tất cả cây quyết định. Kết quả đầu ra của điểm dữ liệu này có thể được lấy là trung bình cộng dự đoán của tất cả các cây quyết định.

Trên đây chính là những thuật toán phổ biến thường được dùng để giải quyết bài toán phân loại nhị phân. Ngoài ra, chúng tôi còn xây dựng mạng học sâu cùng các phương pháp xử lý dữ liệu để tìm ra mô hình hoạt động tốt nhất. Trong dự án lần này, chúng tôi sẽ sử dụng Apache Spark vì tốc độ xử lý nhanh chóng của nó phù hợp với các thuật toán học máy khi giải quyết các vấn đề liên quan đến dữ liệu có kích thước lớn.

3 Bộ dữ liệu

Trong bài báo này chúng tôi sử dụng bộ dataset UIT-ViHSD[10] được thu nhập từ các bình luận người dùng về giải trí, người nổi tiếng, vấn đề xã hội và các vấn đề về chính trị từ các trang web Facebook và các bình luận từ Youtube. Các web Facebook và Youtube được lựa chọn phải có tính tương tác cao và không có lọc các bình luận ác ý. UIT-ViHSD gồm có ba loại nhãn: HATE, OFFENSIVE và CLEAN. Hai nhãn cho các bình luận ác ý và một nhãn cho bình luận không có tính ác ý.

Bảng 1. Chú thích bộ dataset.

Nhãn	Mô tả	Ví dụ
CLEAN(0)	Không chứa từ ngữ ác ý	Bình luận: M.n ơi cho mik hỏi mik theo dõi
OFFENSIVE(1)	Chứa nội dung ác ý nhưng không tấn công cụ thể một đối tượng	Bình luận: Dm chứ biết làm sao
HATE(2)	Các bình luận chứa nội dung ác ý tấn công một đối tượng cụ thể	Bình luận: Thằng Tú óc chó

- CLEAN (Không xúc phạm hay căm thù): bình thường bình luận hoặc bài đăng trên mạng xã hội, nó không chứa lời nói xúc phạm hoặc căm thù.
- OFFENSIVE (Lời nói xúc phạm nhưng không căm thù): một bài đăng hoặc bình luận có thể chứa các từ xúc phạm nhưng nó có không nhắm mục tiêu các cá nhân hoặc nhóm trên cơ sở đặc trưng.
- HATE (Lời nói căm thù): một nhận xét hoặc bài đăng được xác định là lời nói căm thù nếu nó (1) nhắm mục tiêu vào các cá nhân hoặc nhóm trên cơ

sở đặc điểm của họ; (2) thể hiện ý định rõ ràng là kích động gây hại hoặc kích động thù hận; (3) có thể có hoặc không sử dụng các từ xúc phạm hoặc tục tĩu.

Bảng 2. Một số ví dụ được trích xuất từ tập dữ liệu ViHSD.

#	Bình luận	Nhãn
0	Tôi thấy rất vui khi xem anh chơi game mỗi ngày và muốn được chơi cùng	CLEAN(0)
1	Việc nhẹ lương cao mình xin nhường cho những người thất nghiệp:))	CLEAN(0)
2	Dm biết cái gì mà nói vậy?	OFFENSIVE(1)
3	Ý thức còn ít hơn cả số tiền trong túi t	OFFENSIVE(1)
4	Anh ba láo nháo là Tú xiên giờ	HATE(2)
5	Mấy thằng già làm bộ trưởng này nọ mà đầu óc thua cả 1 bác nông dần	HATE(2)

4 Phương pháp

4.1 Tiền xử lý

STOPWORDS[3] là những từ xuất hiện nhiều trong ngôn ngữ tự nhiên, tuy nhiên lại không mang nhiều ý nghĩa. Ở tiếng việt StopWords là những từ như: để, này, kia... Tiếng anh là những từ như: is, that, this... Tham khảo thêm tại danh sách stopwords[2] trong tiếng việt. Có rất nhiều cách để loại bỏ StopWords nhưng có 2 cách chính là:

- Dùng từ điển: cách này đơn giản nhất, chúng ta tiến hành filter văn bản, loại bỏ những từ xuất hiện trong từ điển StopWord.
- Dựa theo tần suất xuất hiện của từ: với cách này, chúng ta tiến hành đếm số lần xuất hiện của từng từ trong data sau đó sẽ loại bỏ những từ xuất hiện nhiều lần (cũng có thể là ít lần). Khoa học đã chứng minh những từ xuất hiện nhiều nhất thường là những từ không mang nhiều ý nghĩa.

Tokenizer[3] là quá trình lấy văn bản (chẳng hạn như một câu) và chia nó thành các thuật ngữ riêng lẻ (thường là các từ). Một lớp Tokenizer đơn giản cung cấp chức năng này. Ví dụ dưới đây cho thấy cách chia câu thành các chuỗi từ.

Ví dụ: Phiên trần / luận tội → Phiên / điều trần / luận tội

RegexTokenizer cho phép mã hóa nâng cao hơn dựa trên đối sánh biểu thức chính quy (regex). Theo mặc định, tham số "pattern" (regex, default: "
s +") được sử dụng làm dấu phân tách để tách văn bản đầu vào. Ngoài ra, người dùng có thể đặt tham số "khoảng trống" thành false cho biết "mẫu" regex biểu thị "mã thông báo" thay vì chia nhỏ khoảng trống và tìm tất cả các lần xuất hiện phù hợp dưới dạng kết quả mã hóa.

CountVectorizer[3] CountVectorizer tạo một ma trận trong đó mỗi từ duy nhất được biểu diễn bằng một cột của ma trận và mỗi mẫu văn bản từ tài liệu là một hàng trong ma trận. Giá trị của mỗi ô chỉ là số lượng từ trong mẫu văn bản cụ thể đó. Điều này có thể được hình dung như sau(hình 1):

	at	each	four	geek	geeks	geeksforgeeks	help	helps	many	one	other	two
document[0]	0	0	0	1	1	0	0	1	0	0	0	1
document[1]	0	0	1	0	2	0	1	0	0	0	0	1
document[2]	1	1	0	1	1	1	0	1	1	0	1	0

Hình 1. Ma trận vector chạy bằng countvectorize

Quan sát bảng:

- Có 12 từ duy nhất trong tài liệu, được biểu diễn dưới dạng các cột của bảng.
- Có 3 mẫu văn bản trong tài liệu, mỗi mẫu được biểu thị dưới dạng các hàng của bảng.
- Mỗi ô chứa một số, đại diện cho số lượng từ trong văn bản cụ thể đó.
- Tất cả các từ đã được chuyển đổi thành chữ thường.
- Các từ trong các cột đã được sắp xếp theo thứ tự bảng chữ cái.

Như vậy chúng ta sẽ thu được một ma trận vector nhị phân như trên với các từ riêng biệt trong document cũng được mã hóa thành số.

HashingTF[3] là một Transformer nhận các tập hợp các số hạng và chuyển đổi các tập hợp đó thành các vectơ đặc trưng có độ dài cố định. Trong xử lý văn bản, một "set of terms" có thể là một túi các từ. HashingTF sử dụng thủ thuật băm. Một đối tượng địa lý thô được ánh xạ thành một chỉ mục (thuật ngữ) bằng cách áp dụng một hàm băm. Hàm băm được sử dụng ở đây là MurmurHash 3. Sau đó, các tần số thuật ngữ được tính toán dựa trên các chỉ số được ánh xạ. Cách tiếp cận này tránh được sự cần thiết phải tính toán một bản đồ từ thành chỉ mục toàn cầu, vốn có thể tốn kém đối với một kho dữ liệu lớn, nhưng nó phải chịu các xung đột tiềm ẩn, trong đó các đối tượng địa lý thô khác nhau có thể trở thành cùng một cụm từ sau khi băm.

IDF[4] Tần suất tài liệu nghịch đảo là thước đo liệu một thuật ngữ hiếm gặp hay thường xuyên trên các tài liệu trong toàn bộ ngữ liệu. Nó làm nổi bật những từ xuất hiện trong rất ít tài liệu trong kho ngữ liệu, hoặc trong ngôn ngữ đơn giản, những từ hiếm có điểm IDF cao. IDF là giá trị chuẩn hóa log, giá trị này nhận được bằng cách chia tổng số tài liệu D trong kho tài liệu cho số lượng tài liệu chứa thuật ngũ t và lấy logarit của thuật ngữ tổng thể.

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

Trong đó, $f_{t,D}$ tần suất của thuật ngữ t trong tài liệu D. $|D|$ là tổng số tài liệu trong kho ngữ liệu. $\{d \epsilon D : t \in D\}$ là số lượng tài liệu trong kho ngữ liệu, chứa thuật ngữ t.

4.2 Độ đo

Precision

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

- Precision được định nghĩa là tỉ lệ số điểm Positive mô hình dự đoán đúng trên tổng số điểm mô hình dự đoán là Positive.
- Precision càng cao, tức là số điểm mô hình dự đoán là positive đều là positive càng nhiều. Precision = 1, tức là tất cả số điểm mô hình dự doán là Positive đều đúng, hay không có điểm nào có nhãn là Negative mà mô hình dự đoán nhầm là Positive.
- Precision là một metric để xác định khi mà việc dự đoán sai các mẫu Positive là rất nguy hiểm. Ví dụ đối với bài toán xác định spam email. Với bài toán này mẫu positive sẽ là spam mail, vậy false positive sẽ là việc dự đoán 1 email không phải spam bị đưa vào hòm mail spam. Việc này sẽ ảnh hưởng rất nhiều tới người sử dụng.

Recall

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

- Recall được định nghĩa là tỉ lệ số điểm Positive mô hình dự đoán đúng trên tổng số điểm thật sự là Positive (hay tổng số điểm được gán nhãn là Positive ban đầu).
- Khi Recall = 1, mọi điểm positive đều được tìm thấy. Tuy nhiên, đại lượng này lại không đo liệu có bao nhiêu điểm negative bị lẫn trong đó. Nếu mô hình phân loại mọi điểm là positive thì chắc chắn Recall = 1, tuy nhiên dễ nhận ra đây là một mô hình cực tồi.
- Đối với metric này người ta sẽ thấy nó thể hiện rằng bao nhiêu mẫu positive thực tế được xác định đúng. Metric này dùng để đánh giá 1 model khi mà việc dự đoán sai 1 mẫu positive thực tế là rất nguy hiểm. Như việc dự đoán các bệnh nhân bị bệnh. Đối với bài toán ví dụ trên ta sẽ thấy:

$$Recall = \frac{1}{1+1} = 50\%$$

F1-Score Chỉ có Precision hay chỉ có Recall thì không đánh giá được chất lượng mô hình:

- Chỉ dùng Precision, mô hình chỉ đưa ra dự đoán cho một điểm mà nó chắc chắn nhất. Khi đó Precision = 1, tuy nhiên ta không thể nói là mô hình này tốt.

– Chỉ dùng Recall, nếu mô hình dự đoán tất cả các điểm đều là positive. Khi đó Recall = 1, tuy nhiên ta cũng không thể nói đây là mô hình tốt.

Khi đó F1-score được sử dụng. F1-score là trung bình điều hòa (harmonic mean) của precision và recall (giả sử hai đại lượng này khác 0). F1-score được tinh theo công thức:

$$F_1 = 2\frac{precion \cdot recall}{precision + recall}$$

Accuracy (Độ chính xác tổng quát) - là độ đo đơn giản thường dùng nhất 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.

4.3 Streaming Spark

Spark Streaming[5] là một phần mở rộng của API Spark core cho phép xử lý luồng dữ liệu trực tiếp có thể mở rộng, thông lượng cao, chịu được lỗi. Dữ liệu có thể được nhập từ nhiều nguồn như Kafka, Kinesis hoặc TCP socket và có thể được xử lý bằng cách sử dụng các thuật toán phức tạp được thể hiện bằng các chức năng cấp cao như map, reduce, join và window. Cuối cùng, dữ liệu đã xử lý có thể được đẩy ra hệ thống tệp, cơ sở dữ liệu và trang tổng quan trực tiếp. Trên thực tế, bạn có thể áp dụng các thuật toán graph processing và học máy của Spark trên các luồng dữ liệu. Điều này có thể được hình dung như(hình 2): Nó hoạt động như sau. Spark Streaming nhận các luồng dữ liệu

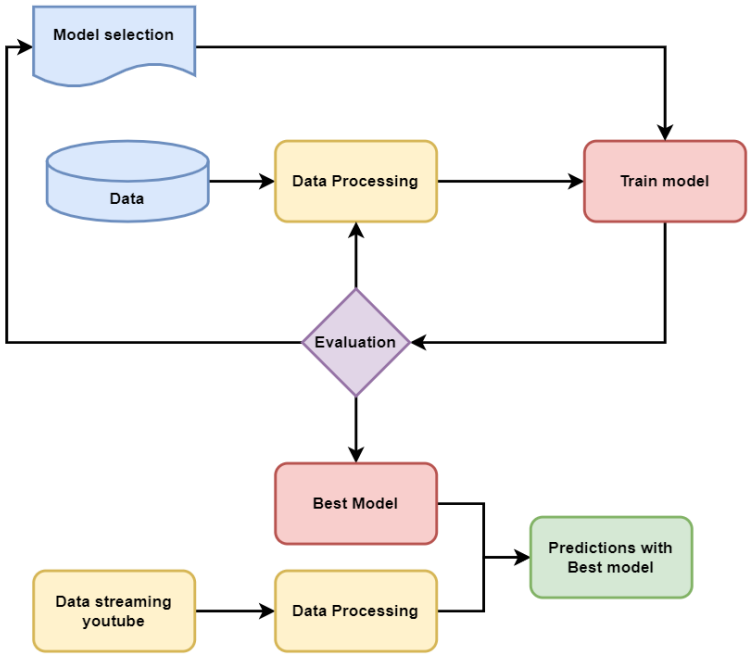
Hình 2. Cách dữ liệu có thể được thu thập

đầu vào trực tiếp và chia dữ liệu thành các batches, sau đó được xử lý bởi công cụ Spark để tạo ra luồng kết quả cuối cùng theo batches.(hình 3):

Hình 3. Hoạt động Spark Streaming

5 Thực nghiệm

5.1 Quy trình thực nghiệm



Hình 4. Quy trình thực hiện

5.2 Môi trường thử nghiệm

Đồ án được thực nghiệm trên môi trường Google Colab bản ram 25gb. Google Colab (Google Colaboratory) là một dịch vụ miễn phí của Google nhằm hỗ trợ nghiên cứu và học tập về AI. Colaboratory cung cấp môi trường Code như Jupyter Notebook và có thể sử dụng GPU và TPU miễn phí.

5.3 Kết quả thực nghiệm và đánh giá

Sau quá trinh phân tích và xử lý dữ liệu chúng tôi có được tập huấn luyện bao gồm 24,048 dòng và tập kiểm tra bao gồm 6,680 dòng. Đối với tất cả các thử nghiệm, chúng tôi dùng độ đo F1 Score để đánh giá, từ đây chọn ra mô hình đạt hiệu suất tốt nhất. Bảng 3 minh họa kết quả của mô hình Pipiline trên bộ dữ liệu ViHSD. Kết quả được đo bằng F1 Score và Accuracy. Theo bảng 3, với Pipiline(Tokenizer + CountVectorizer + IDF) thì LinearSVM đã thu được kết quả tốt nhất F1 Score với 84.43%. Còn với Pipiline(Tokenizer + HashingTF + IDF) thì Naive Bayes đã thu được kết quả tốt nhất F1 Score với 84.57% và là kết quả cao nhất trong tất cả các phương pháp chúng tôi thử nghiệm.

Bảng 3. Kết quả thực nghiệm trên các phương pháp

Pipiline	**Phương Pháp**	**F1 Score**
Tokenizer + CountVectorizer + IDF	Logistic Regression	81.19
	Naive Bayes	78.32
	LinearSVM	**84.43**
	Decision Tree	84.17
	Random Forest	82.79
Tokenizer + HashingTF + IDF	Logistic Regression	81.16
	Naive Bayes	**84.57**
	LinearSVM	84.43
	Decision Tree	83.31
	Random Forest	82.79

Nhìn chung, hiệu suất của Pipiline (Tokenizer + HashingTF + IDF) có kết quả tốt hơn so với Pipiline(Tokenizer + CountVectorizer + IDF) thông qua Naive Bayes với F1 Score cao nhất với 84.57%, cho thấy sức mạnh của nó trong nhiệm vụ text classification, đặc biệt là về phát hiện lời nói căm thù ngay cả khi chúng được đào tạo về nhiều ngôn ngữ khác nhau. Mặc dù kết quả đạt được không quá tệ nhưng chúng tôi sẽ tiếp tục tìm cách cải tiến hiệu suất phân loại bằng phương pháp học sâu.

6 Kết luận

Trong báo cáo này, nhóm sinh viên chúng tôi đã tìm hiểu tổng quan về bài toán phân loại các bình luận trên livestream youtube. Sử dụng Apache Spark để phân tích dữ liệu. Đã triển khai các phương pháp khác nhau cho các đánh giá trên tập dữ liệu và Pipiline (Tokenizer + HashingTF + IDF) có kết quả tốt nhất Naive Bayes với F1 Score 84.57%. So sánh, phân tích và đánh giá độ chính xác của các phương pháp trên thông qua các độ đo tiêu chuẩn. Về cơ bản, trong bài báo cáo này, chúng tôi đã giải quyết được bài toán đặt ra, tuy nhiên độ chính xác vẫn chưa cao. Hướng phát triển được chúng tôi đề xuất trong tương lai là sử dụng các mô hình tiên tiến hơn và nhiều cách xử lí dữ liệu đầu vào phức tạp hơn

để từ đó làm tăng hiệu suất tổng thể của hệ thống. Phát triển các chức năng trên website như tiến hành (xóa, giấu bình luận, tự động trả lời, hoặc tiếp tục trích xuất nội dung bình luận để tìm thêm lý do người dùng gửi bình luận xấu).

Lời cảm ơn Nhóm chúng tôi xin chân thành gửi lời cảm ơn đến TS. Đỗ Trọng Hợp giảng viên phụ trách môn Phân tích Dữ liệu lớn, đã tận tình giúp đỡ, đưa ra những góp ý để chúng tôi hoàn thiện hơn bài báo cáo.

Tài liệu

1. Luu, Son T. et al. "A Large-Scale Dataset For Hate Speech Detection On Vietnamese Social Media Texts". Advances And Trends In Artificial Intelligence. Artificial Intelligence Practices, 2021, pp. 415-426. Springer International Publishing, https://doi.org/:10.1007/978-3-030-79457-6_35. Accessed 16 Jan 2022.
2. Github - Stopwords/Vietnamese-Stopwords: Vietnamese Stopwords. Github, 2022, https://github.com/stopwords/vietnamese-stopwords. Accessed 16 Jan 2022.
3. Extracting, Transforming And Selecting Features - Spark 3.2.0 Documentation. Spark.Apache.Org, 2022, https://spark.apache.org/docs/latest/ml-features. Accessed 25 Jan 2022.
4. "An Improved Approach To Terms Weighting In Text Classification". Ieeexplore.Ieee.Org, 2022, https://ieeexplore.ieee.org/document/5778755. Accessed 25 Jan 2022.
5. Spark Streaming - Spark 3.2.0 Documentation. Spark.Apache.Org, 2022, https://spark.apache.org/docs/latest/streaming-programming-guide.html. Accessed 25 Jan 2022.
6. Wright, Raymond E. "Logistic regression." (1995)
7. Cortes, Corinna, and Vladimir Vapnik. "Support vector machine." Machine learning 20.3 (1995): 273-297.
8. Rish, Irina. "An empirical study of the naive Bayes classifier." IJCAI 2001 workshop on empirical methods in artificial intelligence. Vol. 3. No. 22. 2001.
9. Rokach, Lior, and Oded Maimon. "Decision trees." Data mining and knowledge discovery handbook. Springer, Boston, MA, 2005. 165-192.
10. Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.