ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Samsung Innovation Campus

Nhóm 3

Giảng viên : Cao Văn Chung

Phân loại bình luận tiêu cực

×

Samsung Innovation Campus

Nhóm 3

Thành viên	**Mã Sinh Viên**	**Công việc**
Nguyễn Đức Sĩ	SIC2257	Tìm hiểu, triển khai và đánh giá mô hình Multinomial Naive Bayes, XG-Boosting
Đinh Thái Tuấn	SIC2255	Tìm hiểu, triển khai và đánh giá mô hình SVM
Đỗ Minh Quang	SIC2256	Tìm hiểu, triển khai và đánh giá mô hình LSTM
Nguyễn Thuỳ Trang	SIC2266	Tìm hiểu, triển khai và đánh giá mô hình hồi quy Logistic

Contents

1 Đặt vấn đề

Trong bối cảnh các nền tảng mạng xã hội phát triển mạnh mẽ, sự xuất hiện của các bình luận tiêu cực có thể gây ảnh hưởng xấu đến trải nghiệm của người dùng và làm giảm chất lượng cuộc thảo luận. Điều này không chỉ làm tổn hại đến uy tín của thương hiệu mà còn dẫn đến việc giảm doanh thu khi người dùng rời bỏ nền tảng.

Phân loại và nhận diện sớm các bình luận tiêu cực là một giải pháp hiệu quả giúp tiết kiệm chi phí quản lý và hỗ trợ nâng cao sự hài lòng của khách hàng. Bằng cách phát hiện kịp thời các yếu tố có khả năng gây xung đột, doanh nghiệp có thể chủ động quản lý nội dung, từ đó duy trì một môi trường thảo luận tích cực và văn minh.

Chính vì lý do này, chúng tôi đã quyết định chọn đề tài nghiên cứu là **phân loại các bình luận tiêu cực trên mạng xã hội**. Giải pháp này không chỉ cải thiện trải nghiệm người dùng mà còn góp phần vào việc tối ưu hóa chi phí vận hành, từ đó nâng cao hiệu quả kinh doanh và củng cố vị thế trên thị trường.

2 Tình hình hiện tại

Trước khi xử lý ngôn ngữ tự nhiên (NLP) ra đời, các công ty đã phải sử dụng những phương pháp không hiệu quả để xác định ngôn từ thù hận, chẳng hạn như tìm kiếm từ khóa đơn giản (bag of words). Phương pháp này có "độ bao phủ cao nhưng dẫn đến tỷ lệ dương tính giả cao", khiến cho các cuộc trò chuyện bình thường bị phân loại không chính xác.

Gần đây, các công ty công nghệ lớn đã tăng cường nỗ lực trong việc phát hiện bình luận tiêu cực trên các nền tảng trực tuyến. Meta (trước đây là Facebook) đã đầu tư mạnh vào các công nghệ học máy và các ý tưởng hợp tác với các tổ chức phi lợi nhuận để đối phó với nội dung tiêu cực. Một ví dụ điển hình là Meta đang phát triển các công cụ AI tiên tiến để phát hiện bình luận tiêu cực trong nhiều ngôn ngữ và hoàn cảnh khác nhau. Tuy nhiên, thách thức vẫn còn lớn, đặc biệt là trong việc quản lý nội dung tại các quốc gia đang phát triển và những vùng có xung đột. Các công ty như Telegram cũng đang phải đối mặt với việc kiểm soát nội dung và ngôn từ cực đoan trên nền tảng của họ, nơi mà sự lan truyền thông tin độc hại vẫn còn diễn

ra mạnh mẽ.

3 Giới thiệu và phân tích dữ liệu

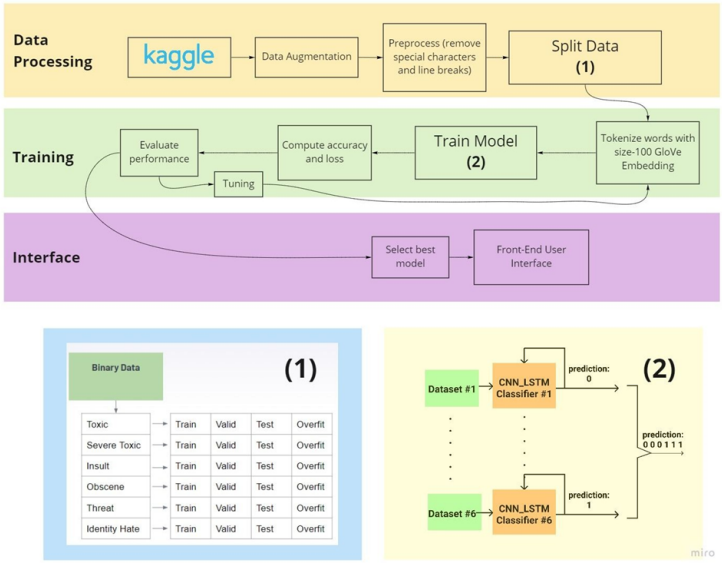

Figure 1: Model Production Pipeline

3.1 Giới thiệu về dữ liệu

Nguồn dữ liệu: Dữ liệu được thu thập từ các bình luận trên Wikipedia và có thể được tải về từ Kaggle.

Đây là tập dữ liệu về các bình luận được thu thập trên trang Wikipedia được dùng trong cuộc thi *Toxic Comment Classification Challenge* trên trang Kaggle. Cuộc thi nhằm mục đích phân loại các bình luận tiêu cực trong bộ dữ liệu. Các loại bình luận tiêu cực bao gồm:

- **toxic**: bình luận xúc phạm người khác
- **severe_toxic**: bình luận xúc phạm gay gắt người khác
- **obscene**: bình luận thô tục

- **threat**: bình luận hăm dọa
- **insult**: bình luận sỉ nhục
- **identity_hate**: bình luận mang tính thù hằn sắc tộc

Chia dữ liệu: Phân chia thành các tập **Train**, **Valid**, **sample_submission**, **test_labels**.

- **train.csv**: tập dữ liệu dùng để train mô hình chứa 159571 bình luận với các loại bình luận tiêu cực tương ứng của chúng (*toxic, severe_toxic, obscene, threat, insult, identity_hate*)

Các trường thông tin như sau:

STT	**Tên trường**	**Mô tả**	**Có null không?**
1	ID	Mã bản ghi	Không
2	Comment_text	Nội dung bình luận	Không
3	toxic	Bình luận xúc phạm	Không
4	severe_toxic	Bình luận xúc phạm gay gắt	Không
5	obscene	Bình luận thô tục	Không
6	threat	Bình luận hăm dọa	Không
7	insult	Bình luận sỉ nhục	Không
8	identity_hate	Bình luận thù hằn sắc tộc	Không

• test.csv: tập dữ liệu test chứa các bình luận.
Các trường thông tin như sau:

STT	**Tên trường**	**Mô tả**	**Có null không?**
1	ID	Mã bản ghi	Không
2	Comment_text	Nội dung bình luận	Không

- sample_submission.csv: nhãn của các bình luận trong tập test, giá trị ban đầu được đặt là 0.5. Khi xây dựng mô hình, nhóm có thể cập nhật các giá trị này dựa trên các dự đoán. Các trường thông tin như sau:

  Các trường thông tin như sau:

STT	Tên trường	Mô tả	Có null không?
1	ID	Mã bản ghi	Không
2	toxic	Bình luận xúc phạm	Không
3	severe_toxic	Bình luận xúc phạm gay gắt	Không
4	obscene	Bình luận thô tục	Không
5	threat	Bình luận hăm dọa	Không
6	insult	Bình luận sỉ nhục	Không
7	identity_hate	Bình luận thù hằn sắc tộc	Không

- test_labels.csv: nhãn của các bình luận trong tập test, giá trị -1 là đánh dấu những nhãn không được sử dụng để test. Vì vậy mà khi xây dựng mô hình, nhóm loại bỏ các dữ liệu test mang giá trị -1.
  Các trường thông tin như sau:

STT	Tên trường	Mô tả	Có null không?
1	ID	Mã bản ghi	Không
2	toxic	Bình luận xúc phạm người	Không
3	severe_toxic	Bình luận xúc phạm gay gắt	Không
4	obscene	Bình luận thô tục	Không
5	threat	Bình luận hăm dọa	Không
6	insult	Bình luận sỉ nhục	Không
7	identity_hate	Bình luận thù hằn sắc tộc	Không

3.2 Phân tích dữ liệu

Trong tập **Train** bao gồm khoảng 160000 bình luận đã được gán theo các nhãn :

- Toxic : Bình luận độc hại.
- Severe toxic : Bình luận cực kì độc hại.
- Obscene : Bình luận thô tục, nhạy cảm.
- Threat : Bình luận mang tính đe dọa.
- Insult : Bình luận mang tính xúc phạm.
- Indetity hate : Bình luận mang tính phân biệt, thù hằn sắc tộc.

Độ dài của dữ liệu vào từ 0 cho đến hơn 2000 từ cho mỗi văn bản. Trong đó, đa số các văn bản có độ dài nhỏ hơn 1000 từ.

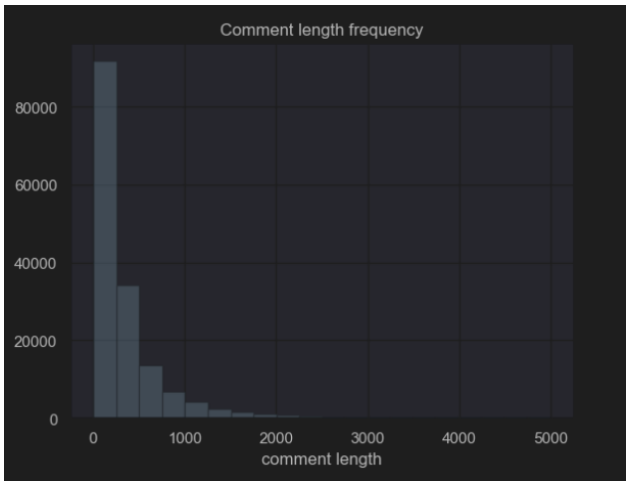

Figure 2: Độ dài của văn bản

Bình luận được coi là độc hại nếu nhãn đó là **1** và không nếu nhãn bằng **0**. Cụ thể trong tập dữ liệu được gán nhãn như sau :

Id	Comment_text	toxic	severe_toxic	...	identity_hate
00...bf	Explanation...	0	0	...	0
00...0f	D'aww! He matches ...	0	0	...	0
00...fd	Hey man, I'm ...	0	0	...	0
00...7e	"" ...	0	0	...	0
00...37	C*CKS*CKER ...	1	1	...	0
...	...	...	...	...	...

Số lượng các văn bản được gán nhãn độc hại :

Ta có thể thấy rằng dữ liệu bị mất cân bằng, và có phân phối không giống nhau trong số những dữ liệu được gán nhãn tiêu cực thì bình luận được gán nhãn *Toxic* chiếm đa số, còn các bình luận *severe toxic*, *threat* và *identity hate* chiếm rất ít.

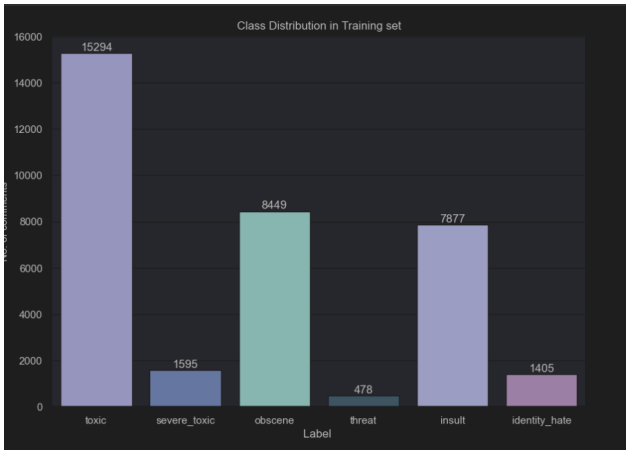

Figure 3: Số lương cụ thể của văn bản được gán nhãn theo từng loại.

Xét toàn bộ dữ liệu thì số lượng dữ liệu được gán nhãn **0** lớn hớn rất nhiều so với nhãn **1**.

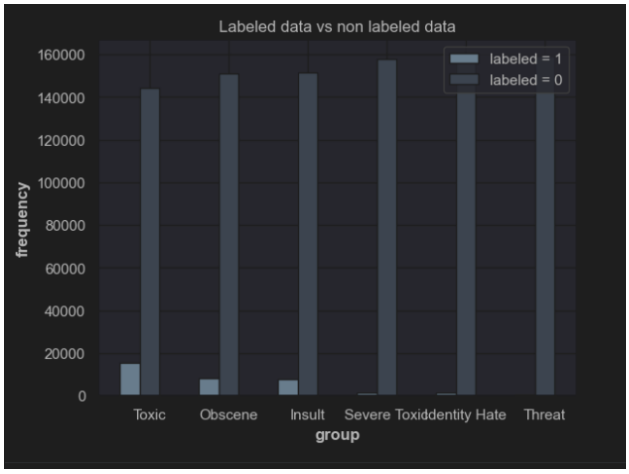

Figure 4: So sánh giữa dữ liệu có nhãn và không nhãn.

Ta có thể nhìn thấy sự tương quan giữa các bình luận tiêu cực thông qua biểu đồ sau :

Dựa vào ma trận tương quan thì ta thấy hệ số tương quan Pearson giữa cột *obscene* và *insult* là khá cao. Nghĩa là những bình luận thô tục thì cũng thường dễ có khả năng mang tính xúc phạm.

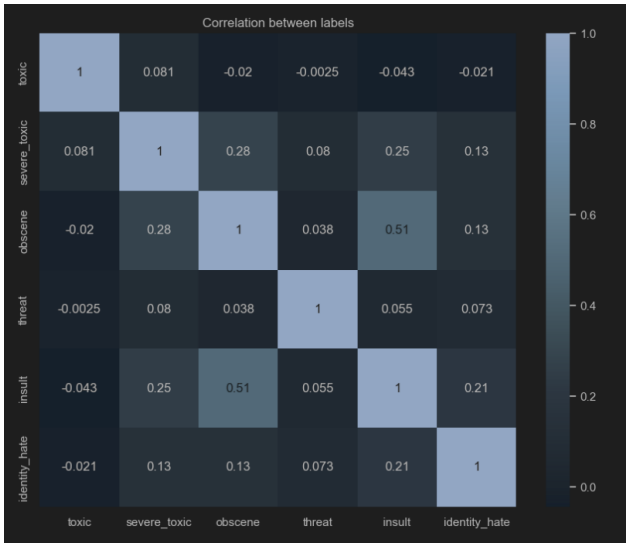

Figure 5: Tương quan về các bình luận tiêu cực

Nhưng hệ số tương quan của các cột còn lại là tương đối thấp, ví dụ như giữa *severe toxic* với *obscene* và *insult*, con số chỉ vào 0.28 và 0.25. Điều đó cho thấy các bình luận cực kỳ thô tục có liên quan đến các bình luận xúc phạm và đe dọa, nhưng liên kết này không quá mạnh.

Thoạt nhìn ban đầu ta đã nghĩ rằng hệ số tương quan của cột *severe toxic* với *toxic* phải cao do một bình luận được gán nhấn là *severe toxic* thì nó cũng được xếp vào mục *toxic*. Nhưng điều ngược lại chưa chắc đúng, do một bình luận toxic có thể toxic ở mức độ nhẹ hơn, nên chưa chắc nó đã được gán nhãn là *severe toxic*. Hơn nữa số lượng bình luận *toxic* lại nhiều hơn số lượng bình luận *severe toxic* nhiều lần theo như trên biểu đồ cột ở trên. Đó là lí do vì sao hệ số tương quan của hai cột khá thấp.

Và các nhãn còn lại có tương quan rất thấp, có cả hệ số âm, tuy nhiên con số là rất nhỏ, điều này cho thấy các nhãn có vẻ độc lập với nhau.

Qua đó, ta có thể đoán được, dữ liệu huận luyện cho mô hình giúp nó dễ dàng phân biết các nhãn, vì việc dự đoán một nhãn sẽ ít bị ảnh hưởng bởi sự xuất hiện của các nhãn khác. Mô hình sẽ không cần lo lắng về sự phụ thuộc giữa chúng, chỉ cần tập chung vào từng nhãn riêng biệt. Mô hình có

thể hoạt động ổn định hơn.

Tuy nhiên, với độ tương quan thấp thì mô hình sẽ không tận dụng được sự tương quan đó để cải thiện hiệu suất, vì có thể từ một nhãn này mà mô hình sẽ dự đoạn được nhãn kia. Qua đó, làm giảm khả tăng tổng quát hóa mô hình và tăng nguy cơ overfitting.

Các từ xuất hiện nhiều nhất trong các bình luận được gắn nhãn tiêu cực có thể được minh họa thông qua Wordcloud :

Figure 6: Toxic

Figure 7: Severe Toxic

Figure 8: Threat

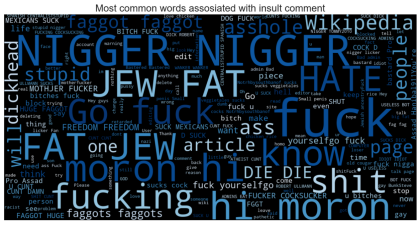

Figure 9: Insult

Figure 10: Obscene

Figure 11: Identity hate

3.3 Xử lý dữ liệu

3.3.1 Làm sạch văn bản

Bước 1: loại bỏ hư từ (Stopword).

Ta loại bỏ tất cả các hư, kí tự đặc biệt từ và dấu câu khỏi mỗi bình luận và chuyển đổi tất cả các từ thành chữ thường. Các hư từ là những từ thường không mang nhiều ý nghĩa ngữ cảnh, xuất hiện rất thường xuyên trong tiếng Anh như 'the', 'and', 'of', 'it', 'to', v.v gây nhiễu thông tin khi trích xuất đặc trưng. Việc loại bỏ các từ dừng giúp làm giảm độ nhiễu của dữ liệu, tập trung vào các từ mang ý nghĩa hơn, từ đó cải thiện hiệu quả của các mô hình xử lý ngôn ngữ tự nhiên.

Bước 2: Loại bỏ ngữ tố trong từ (Stemming) và chuẩn hoá từ (Lemmatization)

Ở bước Stemming, ta thực hiện loại bỏ các tiền tố và hậu tố của từng từ, ví dụ "running" thành "run", "natural" thành "nature",v.v. Tuy nhiên, stemming chỉ loại bỏ các phần phụ của từ mà không quan tâm đến ngữ cảnh, do đó có thể tạo ra các từ gốc không có nghĩa. Vì thế, ta sử dụng cả Lemmatization nhằm đưa 1 từ (token) về dạng gốc của nó (lemma) dựa vào ngữ cảnh và các quy tắc ngữ pháp để đảm bảo từ gốc có ý nghĩa và đúng ngữ pháp. Ta làm việc với thuật toán **PorterStemmer** và **WordNetLemmatizer** từ thư viện NLTK.

3.3.2 Trích xuất đặc trưng

Bước này sẽ chọn ra các đặc trưng tiêu biểu (chính là các từ khóa - Keywords) có tính đại diện cho tập dữ liệu để làm đầu vào (Input) cho thuật toán phân loại.

Vector hoá dữ liệu: Các mô hình học máy như SVM, Naive Bayes, hồi quy Logistics chỉ làm việc với dữ liệu dạng số, vì vậy cần một cơ chế để chuyển hoá dữ liệu dạng văn bản sang dữ liệu số. Nghiên cứu này lựa chọn từ khóa theo phương pháp TF-IDF (Term Frequency/Inverse Document Frequency), giá trị TF-IDF của một từ khóa là một con số thu được qua thống kê thể hiện mức độ quan trọng của từ khóa này trong một bình luận. TF-IDF của từ khóa w_i trong bình luận d được tính bằng công thức sau:

$$TF - IDF_{id} = f_{id} * log(\frac{n}{N})$$

TF-IDF thể hiện tầm quan trọng của một từ với toàn bộ tài liệu đầu vào, dựa trên số lần từ đó xuất hiện trong 1 tài liệu và nhiều tài liệu.Do cơ chế của TF-IDF chỉ quan tâm tới tần suất xuất hiện của một từ, hoàn toàn bỏ

qua ngữ nghĩa của từ trong văn bản đó. Ví dụ, với hai câu "Cái này tốt, không tệ lắm" và "Cái này tệ, không tốt lắm", có số điểm TF-IDF giống nhau nhưng lại mang ý nghĩa trái ngược nhau.
Ở đây chúng ta sử dụng vector hoá TF-IDF của thư viện Sklearn với bộ phân tích từ và n-gram được đặt thành 6 từ. Kết quả thu được từ TF-IDF không tốt và do đó cần một cách tiếp cận khác.

Nhúng từ: Một hướng tiếp cận khác của trích xuất đặc trưng là sử dụng nhúng từ (word embedding). Với mô hình học sâu LSTM, ta sử dụng mô hình GloVe để biểu diễn các từ dưới dạng Vector. GloVe là một thuật toán học máy không giám sát tận dụng cả thông tin toàn cục (tần suất đồng hiện của các từ trong toàn bộ tập dữ liệu) và thông tin cục bộ (mối quan hệ giữa các từ trong một cửa sổ ngữ cảnh nhất định) để tạo ra các vector chất lượng cao. Ý tưởng chính của mô hình là dựa trên quan sát rằng cách các từ xuất hiện cùng nhau trong văn bản có thể tiết lộ ý nghĩa của chúng. Ví dụ, nếu hai từ thường xuất hiện cùng nhau, chúng có thể có liên quan về mặt ngữ nghĩa. Có thể thấy, với những từ càng gần nghĩa, các vector nhúng từ của chúng càng có khoảng cách gần nhau trong không gian.

4 Các mô hình

Ta sẽ sử dụng một số mô hình để phân loại và dùng một số phương pháp để đánh giá hiệu suất mô hình cũng như so sánh các mô mình với nhau để tìm ra mô hình tốt nhất.

4.1 Mô hình Multinomial Naive Bayes

4.1.1 Giới thiệu

Mô hình **Multinomial Naive Bayes** là một trong những thuật toán phân loại thuộc họ Naive Bayes, sử dụng **Định lý Bayes** để phân loại, thường được sử dụng đối với các bài toán phân loại văn bản. Đây là một trong những mô hình đơn giản nhưng hiệu quả, đặc biệt khi áp dụng cho các bài toán với dữ liệu có tính chất phân phối rời rạc.

4.1.2 Cách hoạt động

Giả sử ta có bài toán sau : Ta có **C** classes, và một tập các điểm dữ liệu, khi đó mô hình sẽ sử dụng công thức của **Định lý Bayes** để tính xác suất của một điểm dữ liệu sẽ thuộc vào một class nào đó. Qua đó, class của điểm dữ liệu sẽ được xác định thông qua việc xác định xác xuất lớn nhất.

Ta có công thức của **Định lý Bayes** như sau :

$$P(\mathbf{c} \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid \mathbf{c}) \cdot P(\mathbf{c})}{P(\mathbf{x})} \tag{1}$$

Trong đó

- **c** : là vị trí class đang xét.
- **x** : là điểm dữ liệu đang xét.

Nếu ta xét với $\mathbf{x} \in \mathbb{R}^d$ và giả định rằng tất cả các tính năng độc lập với nhau, ta có:

$$P(\mathbf{x} \mid \mathbf{c}) = P(x_1, x_2, \ldots, x_d \mid \mathbf{c}) = \prod_{i=1}^{d} P(x_i \mid c) \tag{2}$$

Tuy giả thiết này có vẻ "ngây ngô", do thực tế thì ít có dữ liệu nào hoàn toàn độc lập cả. Nhưng giả thiết này lại mang lại một sự hiệu quả bất ngờ. Không chỉ vậy, điều này còn giúp mô hình có tốc độ trainning và test rất nhanh.

Trong mô hình này, các giá trị $P(x_i|\mathbf{C})$ có thể được tính như sau:

$$\lambda_{c,i} = P(x_i \mid \mathbf{c}) = \frac{N_{c,i}}{N_c} \tag{3}$$

Trong đó :

- $N_{c,i}$ 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.

Để tránh trường hợp biểu thức bằng 0 khi một từ mới chưa xuất hiện bao giờ trong class, dù cho các giá trị $\lambda_{c,i}$ khác có lớn thế nào, ta dùng biểu thức sau :

$$\lambda_{c,i} = \frac{N_{c,i} + \alpha}{N_c + d \cdot \alpha} \tag{4}$$

Với α là một số dương, thường bằng 1 để tránh tử số bằng 0. Và mẫu số như thế sẽ đảm bảo tổng xác suất $\sum_{i=1}^{d} \lambda_{c,i} = 1$

(*) Xét mô hình với dữ liệu được nói phần giới thiệu về dữ liệu, mô hình hoạt động như sau:

- Sau khi thu được ma trận TF-IDF, với mỗi dòng sẽ tương ứng với một vector từ. Ta có các tập **train** và tập **test**. Tập **train** sẽ được dùng để tính các xác suất cho từng từ theo từng nhãn. Ví dụ với nhãn *toxic*, mô hình sẽ tính các giá trị $\lambda_{c,i}$ cho từng từ.

- Sau đó, tập **test** sẽ được dùng để kiểm thử, mô hình sẽ tính các giá trị $P(x \mid \mathbf{c})$ với **c** lần lượt là các nhãn trong tất cả 8 nhãn và xét qua tất cả các x là các vector hàng trong tập **test** qua đó xác định xem bình luận đó có phải là bình luận tiêu cực theo nhãn đó không. Ngưỡng xác định ở đây sẽ là ngưỡng mặc định và bằng 0.5.

- Sau khi kiểm tra xong, ta sẽ thu được một tập mới gồm các nhãn đã được gán cho từng văn bản trong tập **test**. Và tập đó sẽ được đi so sánh với kết quả chính xác, và qua đó ta có thể rút ra được các kết luận cho mô hình.

4.2 Hồi quy Logistics

4.2.1 Giới thiệu

Hồi quy logistic là một mô hình thống kê học máy dùng để dự đoán xác suất xảy ra của một sự kiện nhị phân (có hai kết quả: thành công hoặc thất bại, 0 hoặc 1). Mô hình này được sử dụng rộng rãi trong các bài toán phân loại, nơi mà mục tiêu là dự đoán một nhãn thuộc về một trong hai lớp. Thay vì dự đoán giá trị trực tiếp như trong hồi quy tuyến tính, hồi quy logistic dự đoán xác suất một sự kiện thuộc về một lớp cụ thể.

Hồi quy logistic ước lượng xác suất P(Y=1|X) thông qua một hàm Sigmoid được định nghĩa như sau:

$$P(Y = 1 \mid X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n)}}$$

Trong đó:

- $P(Y = 1 \mid X)$ là xác suất của sự kiện xảy ra (Y = 1) dựa trên các biến đầu vào $X_1, X_2, \ldots, X_n$.
- β_0 là hệ số chặn (intercept).
- $\beta_1, \beta_2, \ldots, \beta_n$ là các hệ số hồi quy tương ứng với các biến đầu vào $X_1, X_2, \ldots, X_n$.

Sau khi tìm được mô hình, việc xác định lớp y cho một điểm dữ liệu x được thực hiện bằng cách so sánh hai biểu thức xác suất: $P(y = 1 \mid x; w)$ và $P(y = 0 \mid x; w)$. Nếu biểu thức thứ nhất lớn hơn, ta kết luận điểm dữ liệu thuộc lớp 1, ngược lại, nó thuộc lớp 0.

Vì tổng của hai biểu thức này luôn bằng 1, nên một cách gọn hơn, ta chỉ cần xác định xem $P(y = 1 \mid x; w)$ lớn hơn 0.5 hay không. Nếu có, điểm dữ liệu thuộc lớp 1; nếu không, nó thuộc lớp 0.

Mô hình hồi quy logistic phù hợp để giải quyết các bài toán phân loại nhị phân, chính là dạng bài toán thường gặp trong việc phân loại comment toxic (toxic hoặc non-toxic) với khả năng xử lý đa biến đầu vào và có thể được điều chỉnh để xử lý sự mất cân bằng giữa tỷ lệ toxic - non toxic, ví dụ bằng cách sử dụng weighted loss hoặc điều chỉnh ngưỡng (threshold tuning) để cải thiện độ chính xác phân loại.

4.2.2 Cách hoạt động

Ta có một tập dữ liệu $X = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^{d \times N}$ và $y = [y_1, y_2, \ldots, y_N]$. Có thể giả sử rằng xác suất để một điểm dữ liệu x rơi vào class 1 là $f(w^T x)$ và rơi vào class 0 là $1 - f(w^T x)$.
Với mô hình được giả sử như vậy, với các điểm dữ liệu training (đã biết đầu ra y), ta có thể viết như sau:

$$P(y_i = 1 \mid x_i; w) = f(w^T x_i)$$

$$P(y_i = 0 \mid x_i; w) = 1 - f(w^T x_i)$$

trong đó $P(y_i = 1 \mid x_i; w)$ được hiểu là xác suất xảy ra sự kiện đầu ra $y_i = 1$ khi biết tham số mô hình w và dữ liệu đầu vào x_i. Ký hiệu $z_i = f(w^T x_i)$, và viết gộp lại hai biểu thức bên trên, ta có:

$$P(y_i \mid x_i; w) = z_i^{y_i}(1 - z_i)^{1-y_i}$$

Ở trên là xác suất tại một điểm dữ liệu. Giả sử các quan sát trong bộ dữ liệu của chúng ta là độc lập. Khi đó xác suất đồng thời của toàn bộ các quan sát trong bộ dữ liệu sẽ bằng tích các xác suất tại từng điểm dữ liệu và bằng:

$$P(y|X; w) = \prod_{i=1}^{n} P(y_i|x_i; w) \tag{5}$$

Vế phải của biểu thức (5) chính là một hàm hợp lý (Likelihood Function) đo lường mức độ hợp lý (goodness of fit) của mô hình thống kê đối với dữ liệu.

Chúng ta kỳ vọng giá trị của hàm hợp lý phải lớn. Điều đó đồng nghĩa với các trường hợp tích cực phải có xác suất càng gần 1 và tiêu cực có xác suất gần bằng 0. Do đó mục tiêu của chúng ta là tìm w sao cho biểu thức (5) là lớn nhất.

Để tìm ra nghiệm của hồi quy Logistic, chúng ta sẽ thực hiện việc cập nhật nghiệm trên từng điểm dữ liệu. Các điểm dữ liệu được lựa chọn một cách ngẫu nhiên ở mỗi lượt cập nhật. Phương pháp cập nhật này được gọi là phương pháp **Stochastic Gradient Descent** (SGD) với công thức:

$$\mathbf{w} := \mathbf{w} - \alpha \mathbf{x}_i(y_i - \hat{y}_i)$$

(*) Cách mô hình hồi quy Logistics hoạt động trong bài toán phân loại toxic comments:

- **Xử lý dữ liệu:** Sau khi thu được ma trận TF-IDF , với mỗi dòng sẽ tương ứng với một vector từ. Ta có các tập **train** và tập **test**. Tập **train** sẽ được dùng để tính các xác suất cho từng từ theo từng nhãn.

- **Huấn luyện mô hình:** Trong quá trình huấn luyện, mô hình hồi quy logistic sẽ học cách tối ưu hóa các hệ số w sao cho hàm logistic $P(y = 1 \mid x; w)$ có thể dự đoán chính xác xác suất của một văn bản thuộc vào một nhãn cụ thể, ví dụ như nhãn "toxic". Mô hình sẽ tính toán các giá trị w_i cho từng từ trong tập **train**.

- **Kiểm thử mô hình:** Sau khi mô hình đã được huấn luyện, tập test sẽ được sử dụng để kiểm thử mô hình. Đối với mỗi văn bản trong tập test, mô hình sẽ tính toán giá trị xác suất $P(y = 1 \mid x; w)$ cho tất cả các nhãn (ở đây là 6 nhãn) dựa trên vector từ của văn bản đó. Sau đó, xác định xem bình luận đó có thuộc nhãn "toxic" hoặc các nhãn khác không, bằng cách so sánh giá trị xác suất với ngưỡng mặc định (threshold) là 0.5. Nếu xác suất $P(y = 1 \mid x; w)$ lớn hơn 0.5, văn bản sẽ được gán nhãn tương ứng, nếu không sẽ được gán nhãn khác.
- **Đánh giá mô hình:** Sau khi kiểm tra xong, ta sẽ thu được một tập kết quả mới gồm các nhãn đã được gán cho từng văn bản trong tập **test**. Tập kết quả này sẽ được so sánh với các nhãn chính xác ban đầu để đánh giá hiệu suất của mô hình. Qua đó, ta có thể rút ra những kết luận về hiệu quả của mô hình hồi quy logistic trong việc phân loại các văn bản theo các nhãn nhất định.

4.3 Mô hình SVM (Máy hỗ trợ véc tơ)

4.3.1 Giới thiệu

SVM là một thuật toán Supervised Learning được dùng trong các bài toán phân loại, hồi quy, nhận dạng mẫu. SVM hoạt động bằng cách tìm kiếm một siêu phẳng để phân chia tập dữ liệu thành các lớp khác nhau, và siêu phẳng này phải có khoảng cách (hay còn gọi là lề) lớn nhất giữa các điểm dữ liệu thuộc hai lớp. Nếu tập dữ liệu có thể phân chia hoàn toàn (linearly separable), thì SVM sẽ tìm kiếm siêu phẳng tối ưu sao cho khoảng cách giữa các điểm dữ liệu gần siêu phẳng nhất và siêu phẳng là lớn nhất

4.3.2 Cách hoạt động

Giả sử chúng ta có một tập dữ liệu gồm các điểm dữ liệu thuộc về hai lớp khác nhau. SVM sẽ tìm một hyperplane có dạng:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{6}$$

Trong đó:

- $\mathbf{w}$ là vector trọng số.
- $\mathbf{x}$ là vector đặc trưng của điểm dữ liệu.

- b là hệ số điều chỉnh.

Vấn đề là có rất nhiều siêu phẳng, chúng ta phải chọn cái nào để tối ưu nhất?

Giả sử chúng ta phải phân loại tập dữ liệu các lớp dương (màu xanh) nhãn là 1 và các dữ liệu lớp âm (màu đỏ) nhãn là -1 (tập dữ liệu có thể phân tách tuyến tính).

Siêu phẳng phân tách hai lớp dữ liệu H_0 thỏa mãn

$$\langle W, X \rangle + b = 0.$$

Siêu phẳng này tạo ra hai nửa không gian (half space) dữ liệu:

Không gian các dữ liệu lớp âm X_i thỏa mãn

$$\langle W, X_i \rangle + b \leq -1$$

và không gian dữ liệu lớp dương X_j thỏa mãn

$$\langle W, X_j \rangle + b \geq 1.$$

Tiếp theo, ta chọn hai siêu phẳng lề H_1 đi qua điểm thuộc lớp âm và H_2 đi qua điểm thuộc lớp dương đều song song với H_0:

$$H_1 : \langle W, X \rangle + b = -1$$
$$H_2 : \langle W, X \rangle + b = 1.$$

Khoảng cách từ H_1 đến H_0 là d_-, và khoảng cách từ H_2 đến H_0 là d_+.

$$m = d_- + d_+$$

được gọi là mức lề.

Siêu phẳng tối ưu mà chúng ta cần chọn là siêu phẳng phân tách có lề lớn nhất. Lý thuyết học máy đã chỉ ra rằng một siêu phẳng như vậy sẽ cực tiểu hóa giới hạn lỗi mắc phải. Khoảng cách từ một điểm X_k đến siêu phẳng H_0 được tính bằng công thức:

$$d = \frac{|\langle W, X_k \rangle + b|}{||W||}$$

Trong đó $||W||$ là độ dài của vector W:

$$||W|| = \sqrt{\langle W, W \rangle} = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}.$$

Khoảng cách từ một điểm X_i nằm trên H_1 đến H_0 được tính như sau:

$$d_- = \frac{|\langle W, X_i \rangle + b|}{||W||} = \frac{1}{||W||}.$$

Tương tự, khoảng cách từ một điểm X_j nằm trên H_2 đến H_0 là:

$$d_+ = \frac{|\langle W, X_j \rangle + b|}{||W||} = \frac{1}{||W||}.$$

Từ đó, ta có thể tính được mức lề m:

$$m = d_- + d_+ = \frac{2}{||W||}.$$

Giờ đây, bạn đã hiểu vì sao các điểm nằm trên hai siêu phẳng H_1 và H_2 được gọi là các Support Vector.

Việc huấn luyện trong thuật toán SVM tương đương với bài toán cực tiểu hóa có ràng buộc sau đây:

Cực tiểu hóa:

$$\frac{\langle W, W \rangle}{2}$$

Với điều kiện:

$$\begin{cases} \langle W, X_i \rangle + b \leq -1, & \text{if } y_i = -1 \quad (1) \\ \langle W, X_i \rangle + b \geq 1, & \text{if } y_i = 1 \quad (2) \end{cases}$$

Nhân hai vế bất đẳng thức của (1) và (2) với y_i, ta có điều kiện thu gọn:

$$y_i \cdot \langle W, X_i \rangle \geq 1 \quad \forall i = 1, \ldots, n.$$

Mục tiêu của SVM là tối đa hóa khoảng cách giữa hyperplane và các điểm dữ liệu gần nhất, tức là:

$$\max \left(\frac{2}{\|\mathbf{w}\|} \right) \tag{7}$$

Đối với bài toán phân loại bình luận độc hại, chúng ta sử dụng vector hóa TF-IDF để chuyển đổi các bình luận thành các vector số. Mô hình SVM sẽ hoạt động như sau:

- Đầu tiên, bình luận được chuyển đổi thành các vector sử dụng phương pháp TF-IDF.
- Sau đó, tập dữ liệu được chia thành tập huấn luyện và tập kiểm tra. Mô hình SVM sẽ được huấn luyện trên tập huấn luyện để tìm ra hyperplane tối ưu phân tách các bình luận độc hại và không độc hại.
- Cuối cùng, mô hình sẽ được áp dụng trên tập kiểm tra để phân loại các bình luận và đánh giá hiệu suất của mô hình dựa trên các chỉ số như Hamming_loss,Accuracy, log_loss,precision,ROC

4.4 Mô hình XGBoosting

4.4.1 Giới thiệu

Mô hình **XGBoosting** (hay Extreme Gradient Boosting) là một mô hình của phương pháp Ensemble trong Machine Learning, sử dụng sự kết hợp của các mô hình yếu hơn để tạo nên mô hình tốt hơn. **XGBoosting** sử dụng biến thể **Boosting**, biến thể này sẽ xây dựng một lượng lớn các model (thường là cùng loại). Mỗi model sau sẽ học cách sửa những lỗi của model trước (dữ liệu mà model trước dự đoán sai), tạo thành một chuỗi các model mà model sau sẽ tốt hơn model trước bởi trọng số được update qua mỗi model. Và chúng ta sẽ lấy kết quả của model cuối cùng trong chuỗi model này làm kết quả trả về.

4.4.2 Cách hoạt động

XGBoosting hoạt động như sau:

- **Bước 1** : XGBoost bắt đầu bằng cách khởi tạo một mô hình dự đoán cơ bản. Thường là giá trị trung bình của mục tiêu y trong trường hợp hồi quy hoặc xác suất lớp trong trường hợp phân loại.

$$\hat{y}^{(0)} = \text{mean}(y)$$

- **Bước 2** : Ở mỗi bước tiếp theo, XGBoost tính toán residuals (phần dư) bằng cách lấy hiệu giữa giá trị thực và giá trị dự đoán hiện tại:

$$r_i^{(t)} = y_i - \hat{y}_i^{(t-1)}$$

Trong đó:

- $r_i^{(t)}$ là residual của mẫu i tại bước t.
- y_i là giá trị thực.
- $\hat{y}_i^{(t-1)}$ là giá trị dự đoán từ mô hình hiện tại.

- **Bước 3** : Xây dựng cây quyết định. Sử dụng các residuals, XGBoost xây dựng một cây quyết định mới. Tại mỗi nút trong cây, XGBoost thực hiện quá trình chia nhỏ dựa trên giá trị "gain" (lợi ích):

$$\text{Gain} = \frac{1}{2}\left[\frac{(\sum g_{\text{left}})^2}{\sum h_{\text{left}} + \lambda} + \frac{(\sum g_{\text{right}})^2}{\sum h_{\text{right}} + \lambda} - \frac{(\sum g_{\text{parent}})^2}{\sum h_{\text{parent}} + \lambda}\right] - \gamma$$

  Trong đó:

  - $\sum g_{\text{left}}$, $\sum g_{\text{right}}$, $\sum g_{\text{parent}}$ là tổng gradient tại các nút trái, phải và cha.
  - $\sum h_{\text{left}}$, $\sum h_{\text{right}}$, $\sum h_{\text{parent}}$ là tổng hessian tại các nút trái, phải và cha.
  - λ là tham số điều chuẩn để tránh overfitting.
  - γ là giá trị điều chỉnh cho việc thêm một nút.

- **Bước 4** : Cập nhật dự đoán. Mỗi cây mới được thêm vào mô hình để cập nhật dự đoán của mô hình:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta \cdot f^{(t)}(x_i)$$

  Trong đó:

  - η là tốc độ học (learning rate).
  - $f^{(t)}(x_i)$ là dự đoán từ cây mới được xây dựng tại bước t cho mẫu x_i.

- **Bước 5** : Quá trình từ Bước 2 đến bước 4 được lặp lại cho đến khi đạt được số lượng cây tối đa hoặc hàm mất mát được tối ưu hóa đến mức đủ tốt.

- **Bước 6** : Tổng hợp và dự đoán. Cuối cùng, dự đoán cuối cùng được tạo ra bằng cách tổng hợp dự đoán từ tất cả các cây:

$$\hat{y}_i = \sum_{t=1}^{T} \eta \cdot f^{(t)}(x_i)$$

Trong đó T là số lượng cây.

Đối với bài toán toán phân loại đa nhãn như **Phân loại bình luận tiêu cực** mô hình sẽ phân loại dựa trên từng nhãn như một bài toán phân loại nhị phân. Thêm vào đó, việc dự đoán mỗi nhãn không chỉ dựa vào đặc trưng của nó theo nhãn đó, mà còn dựa vào các dự đoán của mô hình từ trước đó.

4.5 Mô hình LSTM

4.5.1 Mô hình RNN

Mô hình RNN (Recurrent Neural Network) là một loại mạng nơ-ron được thiết kế đặc biệt để làm việc với dữ liệu theo chuỗi, như chuỗi thời gian, văn bản, hoặc các tín hiệu liên tục. Khác với các mạng nơ-ron thông thường, RNN có khả năng lưu trữ và sử dụng thông tin từ các bước thời gian trước đó. Kiến trúc của mô hình RNN:

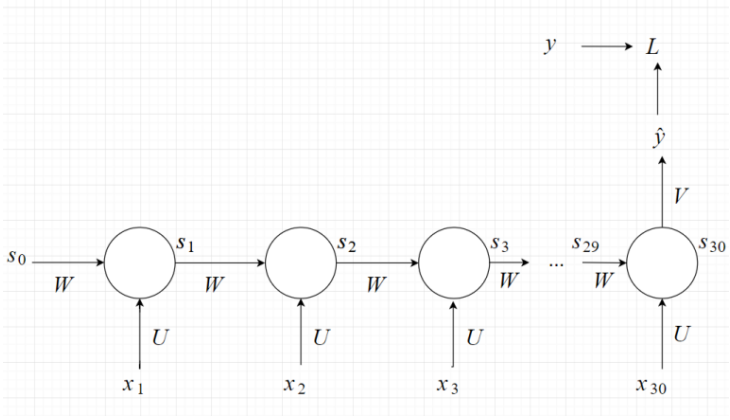

Figure 12: Mô hình RNN cho dữ liệu với 30 input

Các bước hoạt động của mô hình:

- Mỗi hình tròn được gọi là một state, state t có input là x_t và s_{t-1}.

- Tại bước t, mô hình sẽ tính toán state t là $s_t = f(U * x_t + W * s_{t-1})$, trong đó s_{t-1} nhận được từ bước tính toán trước. Có thể thấy s_t mang thông tin của state trước (s_{t-1}), còn f là activation function, thường là tanh hoặc ReLU.

- s_0 được thêm vào chỉ cho chuẩn công thức nên thường được gán bằng 0 hoặc giá trị ngẫu nhiên.

- Cuối cùng, ta tính được giá trị của output $\hat{y} = g(V * s_{30})$, trong đó g cũng là một activation function.

Dựa vào cấu trúc mô hình cho phép lưu trữ thông tin từ mỗi bước, RNN có khả năng sử dụng thông tin từ các bước thời gian trước đó để đưa ra dự đoán cho bước tiếp theo. Tuy nhiên, khi làm việc với các chuỗi dài, thì RNN lại xuất hiện một vài vấn đề, chẳng hạn như vấn đề Gradient Vanishing và Gradient Exploding.

4.5.2 Mô hình LSTM

Mạng trí nhớ ngắn hạn định hướng dài hạn còn được viết tắt là LSTM làm một kiến trúc đặc biệt của RNN có khả năng học được sự phục thuộc trong dài hạn (long-term dependencies) được giới thiệu bởi Hochreiter và Schmidhuber (1997). Kiến trúc này đã được phổ biến và sử dụng rộng rãi cho tới ngày nay. LSTM đã tỏ ra khắc phục được rất nhiều những vấn đề của RNN trước đây, chẳng hạn như vấn đề về Gradient Vanishing hoặc Gradient Exploding. Tuy nhiên cấu trúc của chúng có phần phức tạp hơn mặc dù vẫn giữ được ý tưởng chính của RNN là sự sao chép các kiến trúc theo dạng chuỗi. Kiến trúc của mô hình LSTM:

Kiến trúc của mô hình:

- Output: gồm c_t (cell state), h_t (hidden state).

- Input: c_{t-1}, h_{t-1}, x_t. Trong đó x_t là input tại state t, c_{t-1}, h_{t-1} là output từ state trước.

- f_t, i_t, o_t ứng với forget gate, input gate, output gate.

  - Forget gate: $f_t = \sigma(U_f * x_t + W_f * h_{t-1} + b_f)$.
  - Input gate: $i_t = \sigma(U_i * x_t + W_i * h_{t-1} + b_i)$.
  - Forget gate: $o_t = \sigma(U_o * x_t + W_o * h_{t-1} + b_o)$.

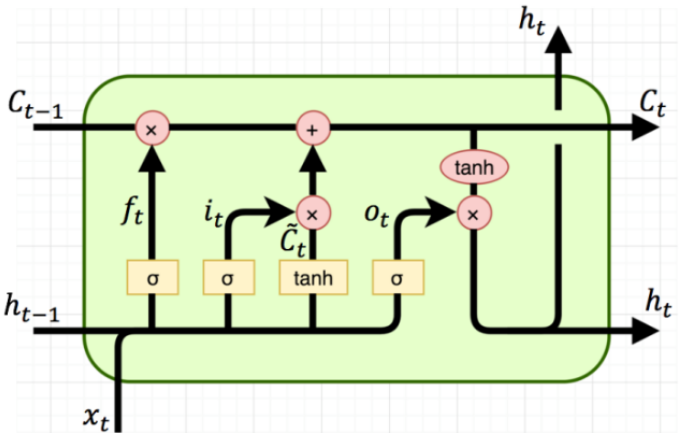

Figure 13: Kiến trúc mô hình LSTM tại một state

- $\tilde{c}_t = tanh(U_c * x_t + W_c * h_{t-1} + b_c)$.
- $c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$, nghĩa là **forget gate** sẽ quyết định xem cần lấy bao nhiêu phần từ cell state trước, và **input gate** sẽ quyết định lấy bao nhiêu từ input của state và hidden state của state trước. Điều này là dễ hiểu do $\tilde{c}_t$ chứa thông tin về input x_t và hidden state của state trước h_{t-1}. Lưu ý rằng $0 < f_t, i_t, o_t < 1$.
- $h_t = o_t * tanh(c_t)$, nghĩa là **output gate** quyết định xem cần lấy bao nhiêu phần từ cell state để đưa ra hidden state.

Để áp dụng mô hình LSTM vào bài toán phân loại bình luận, ta thực hiện các bước sau:

- **Bước 1:** Ta đi Tokenize dữ liệu. Mỗi bình luận ta bỏ dấu câu, loại bỏ stopwords, tiến hành lemmatizing, stemming, sau đó tách thành các từ riêng lẻ. Sau đó ta sử dụng mô hình GloVe, cụ thể là mô hình **glove.6B.100d** - một mô hình được pre-trained trên 6 tỷ từ - để nhúng từ.
- **Bước 2:** Ta khởi tạo mô hình Sequential bằng thư viện Keras. Tiếp theo ta lần lượt thêm vào các lớp:
  - **Embedding**: Thêm lớp Embedding với kích thước vector nhúng là 100 chiều tương ứng với mô hình **glove.6B.100d**. Ma trận

trọng số ta dùng là ma trận đã được huấn luyện bằng mô hình GloVe.

- **Bidirectional LSTM**: Thêm lớp Bidirectional LSTM với 128 đơn vị nhớ.
- **GlobalMaxPool1D**: Lớp này sẽ có tác dụng giảm đầu ra của lớp LSTM xuống còn một véc tơ.
- **Dropout**: Thêm lớp Dropout với tỉ lệ 25% để tránh hiện tượng overfitting.
- **Dense**: Thêm một lớp Dense với 6 đơn vị tương ứng với 6 lớp phân loại, cùng với hàm kích hoạt là hàm *sigmoid*.

- **Bước 3:** Huấn luyện mô hình.

5 Đánh giá và kết luận

5.0.1 Các chỉ số để đánh giá mô hình

Để đánh giá độ tốt của một mô hình, chúng ta sẽ sử dụng một vào các chỉ số sau để thực hiện.

- **Hamming loss** : tính toán tỷ lệ nhãn bị đoán sai trên tổng số nhãn. Trong bài toán đã nhãn chỉ số này được tính bằng số lượng nhãn bị phân loại sai chia cho tổng số nhãn. Nếu chỉ số Hamming Loss càng thấp thì cho thấy mô hình đang dự đoán đúng nhiều nhãn trên mẫu.

- **Accuracy** : chỉ số cho biết mô hình của bạn chính xác bao nhiêu phần trăm trong các dự đoán của nó.

- **Log Loss (Logarithmic Loss)** : Đây là chỉ số đo lường độ chính xác của mô hình bằng cách tính toán mức độ chênh lệch giữa các dự đoán xác suất của mô hình và nhãn thực tế.

- **Micro-averaged over all classes** : Một phương pháp tính toán chỉ số đánh giá cho các mô hình phân loại trong các bài toán nhiều lớp hoặc nhiều nhãn. Khi áp dụng phương pháp micro-averaging, các chỉ số như Precision, Recall, F1-Score, và ROC-AUC được tính toán bằng cách gộp tất cả các dự đoán từ mọi lớp hoặc nhãn lại với nhau, thay vì tính riêng rẽ cho từng lớp rồi trung bình.

- **Micro-Averaged over All Toxicity Levels** : Một chỉ số dùng để đánh giá hiệu suất của mô hình phân loại trong bài toán nhiều nhãn, chẳng hạn như phân loại các mức độ toxic khác nhau trong một tập dữ liệu. Khi tính toán chỉ số này, hiệu suất của mô hình trên tất cả các nhãn (toxic, severe toxic, obscene, threat, insult, identity hate, v.v.) được gộp lại để tạo ra một chỉ số duy nhất.

- **ROC Score** : Một công cụ quan trọng trong việc đánh giá hiệu suất của các mô hình phân loại, đặc biệt là trong các bài toán phân loại nhị phân. ROC chủ yếu được sử dụng để đánh giá khả năng phân biệt giữa các lớp của mô hình.

5.1 Đánh giá

Sau quá trình huấn luyện và kiểm tra, các chỉ số đánh giá hiệu suất của các mô hình được tổng hợp như sau:

Chỉ số	**SVM**	**Bayes**	**LR**	**XGBoost**	**LSTM**
Hamming_loss	3.53	3.34	3.61	2.04	-
Accuracy	85.43	89.77	86.16	91.62	99.48
Log_loss	1.40	4.70	0.73	1.40	6.68
Micro-averaged over all classes	0.44	0.27	0.44	0.49	0.73
ROC	84.60	62.36	89.02	70.04	94.54
Micro-averaged over all toxicity levels	89.91	70.53	91.05	77.16	-

Table 1: So sánh hiệu suất các mô hình

5.1.1 Hamming Loss

- Trong các mô hình được so sánh, XGBoost có Hamming Loss thấp nhất (2.04), cho thấy đây là mô hình có hiệu suất cao nhất trong việc giảm thiểu lỗi đa nhãn.

- Mô hình SVM, Bayes và Logistic Regression có Hamming Loss khá gần nhau, lần lượt là 3.53, 3.34 và 3.61. Điều này chỉ ra rằng các mô hình

này có mức độ lỗi tương đối tương đương, với Bayes có chút lợi thế hơn.

- Mô hình LSTM không có dữ liệu về Hamming Loss, do đó không thể so sánh trực tiếp.

5.1.2 Accuracy

- Mô hình **LSTM** có độ chính xác cao nhất (99.48%), cho thấy khả năng phân loại rất mạnh mẽ và vượt trội so với các mô hình khác.
- **XGBoost** cũng có độ chính xác cao (91.62%), vượt trội hơn so với **SVM** (85.43%), **Bayes** (89.77%) và **Logistic Regression** (86.16%).
- **Bayes** và **Logistic Regression** có độ chính xác khá tương đồng, nhưng cả hai đều kém hơn so với LSTM và XGBoost.

5.1.3 Log Loss

- Mô hình **Logistic Regression** có Log Loss thấp nhất (0.73), cho thấy khả năng dự đoán xác suất tốt nhất trong số các mô hình, ngoại trừ LSTM.
- Mô hình **LSTM** có Log Loss cao (6.68), điều này có thể là kết quả của quá trình huấn luyện lâu dài và khả năng quá mức hóa của mô hình.
- **Bayes** có Log Loss cao nhất (4.70), cho thấy khả năng dự đoán xác suất của mô hình này kém hơn so với các mô hình khác.

5.1.4 Micro-Averaged Over All Classes

- Mô hình **LSTM** có giá trị micro-averaged cao nhất (0.73), cho thấy mô hình này hoạt động hiệu quả trên toàn bộ các lớp.
- **XGBoost** có giá trị micro-averaged cao thứ hai (0.49), cho thấy khả năng xử lý dữ liệu tốt nhưng không bằng LSTM.
- **SVM** và **Logistic Regression** có cùng giá trị micro-averaged (0.44), điều này chỉ ra rằng cả hai mô hình này đều có hiệu suất tương tự nhau.

- **Bayes** có giá trị micro-averaged thấp nhất (0.27), cho thấy mô hình này không tốt như các mô hình khác trong việc xử lý dữ liệu đa lớp.

5.1.5 ROC (Receiver Operating Characteristic)

- Mô hình **LSTM** có giá trị ROC cao nhất (94.54), cho thấy mô hình này có khả năng phân biệt tốt giữa các lớp.
- **Logistic Regression** cũng có giá trị ROC cao (89.02), chỉ đứng sau LSTM.
- **SVM** và **XGBoost** có giá trị ROC lần lượt là 84.60 và 70.04, cho thấy khả năng phân biệt của chúng thấp hơn so với LSTM và Logistic Regression.
- **Bayes** có ROC thấp nhất (62.36), cho thấy mô hình này có hiệu suất phân loại kém hơn so với các mô hình khác.

5.1.6 Micro-Averaged Over All Toxicity Levels

- Trong tiêu chí này, **XGBoost** dẫn đầu với giá trị 91.05, cho thấy mô hình này hoạt động tốt trên các mức độ độc tính khác nhau.
- **Logistic Regression** và **SVM** cũng có giá trị cao, lần lượt là 91.05 và 89.91.
- **Bayes** có giá trị thấp nhất (70.53), điều này một lần nữa khẳng định rằng mô hình này không tốt trong việc xử lý dữ liệu với nhiều mức độ độc tính khác nhau.
- Mô hình **LSTM** không có số liệu trong tiêu chí này, do đó không thể so sánh trực tiếp.

5.2 Kết luận so sánh

LSTM là mô hình vượt trội nhất trong các mô hình được so sánh, với độ chính xác cao nhất, Log Loss thấp nhất, và giá trị ROC và micro-averaged over all classes tốt nhất. Điều này cho thấy LSTM có khả năng phân loại mạnh mẽ và đáng tin cậy nhất trong tất cả các mô hình.

XGBoost cũng là một mô hình rất tốt, đặc biệt trong tiêu chí Hamming Loss và Micro-Averaged over All Toxicity Levels. Mô hình này có thể là lựa chọn tốt nếu mục tiêu là giảm thiểu lỗi đa nhãn hoặc xử lý dữ liệu với nhiều mức độ độc tính khác nhau.

Logistic Regression là mô hình truyền thống có hiệu suất khá tốt, đặc biệt trong tiêu chí ROC và Log Loss. Đây có thể là lựa chọn hợp lý nếu bạn cần một mô hình đơn giản nhưng hiệu quả.

SVM có hiệu suất tương đối ổn định, nhưng không nổi bật trong bất kỳ tiêu chí nào. Điều này khiến SVM trở thành một lựa chọn an toàn nhưng không phải là tối ưu nhất.

Bayes là mô hình yếu nhất trong số các mô hình được so sánh, với các chỉ số thấp hơn rõ rệt so với các mô hình khác, đặc biệt là trong các tiêu chí ROC và Micro-Averaged over All Classes. Mô hình này có thể không phải là lựa chọn tốt nhất cho bài toán này.

6 Tài liệu tham khảo

[1] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated Hate Speech Detection and the Problem of Offensive Language," dissertation, 2017. 1

[2] "Facebook, Telegram, and the Ongoing Struggle Against Online Hate Speech," Carnegie Endowment for International Peace, 2023. 2

[3] Naive Bayes - Machine Learning cơ bản. 3