

# TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN SAMSUNG INNOVATION CAMPUS



Artificial Intelligence

---

## Nhận diện Tin giả

---

*GV hướng dẫn: Cao Văn Chung*

Nguyễn Thị Phương Hoa - SIC2253

Nguyễn Huyền Trang - SIC2281

Bùi Đức Phú Anh - SIC2271

Nguyễn Mai Hương - SIC2376

Dương Hoàng Long - SIC2269

*Hà Nội, ngày 25 tháng 8 năm 2024*

# LỜI NÓI ĐẦU

Lời đầu tiên, chúng em xin gửi lời cảm ơn sâu sắc tới các thầy cô và anh chị trong chương trình Samsung Innovation Campus, những người đã hết mình truyền đạt và hướng dẫn cho chúng em những bài học quý báu và bổ ích.

Thời gian qua, trên mạng Internet, đặc biệt là các trang mạng xã hội, xuất hiện một số tài khoản giả mạo, đăng các thông tin không kiểm chứng liên quan đến nhiều chủ đề về chính trị, dịch bệnh, thiên tai, khí tượng thủy văn, mê tín dị đoan, quảng cáo sai sự thật... Việc này gây hoang mang, xáo trộn, ảnh hưởng lớn đến đời sống sinh hoạt của người dân. Chính vì những lý do đó, trong khuôn khổ nội dung môn Trí tuệ Nhân tạo, nhóm chúng em lựa chọn chủ đề này.

Trong quá trình nghiên cứu làm đề tài, chúng em còn nhiều hạn hẹp về kiến thức và trình độ nên không thể tránh khỏi những thiếu sót. Chúng em mong nhận được những góp ý từ quý thầy cô cũng như những bạn đang đọc bài nghiên cứu này. Chúng em xin chân thành cảm ơn!

---

# Mục lục

<b>LỜI NÓI ĐẦU</b>	<b>1</b>
<b>1 GIỚI THIỆU CHUNG</b>	<b>1</b>
1.1 Đặt vấn đề	1
1.1.1 Ví dụ	1
1.1.2 Phân loại	2
1.2 Thực trạng	3
1.3 Mục tiêu	3
<b>2 PHƯƠNG PHÁP THỰC HIỆN</b>	<b>5</b>
2.1 Thu thập và phân tích dữ liệu	5
2.2 Trích xuất đặc trưng	7
2.2.1 TF-IDF	7
2.2.2 Count Vectorization	8
2.3 Xây dựng các mô hình theo hướng Machine Learning	9
2.4 Xây dựng mô hình theo hướng Deep Learning	10
2.5 Huấn luyện và kiểm thử mô hình	11
<b>3 XÂY DỰNG MÔ HÌNH</b>	<b>12</b>
3.1 Support Vector Machine (SVM)	12
3.1.1 Tìm hệ số của đường thẳng phân loại	13
3.1.2 Giải bài toán tối ưu SVM	14
3.1.3 Biểu thức đối ngẫu	15
3.2 Logistic Regression	16
3.2.1 Xây dựng thuật toán	16
3.2.2 Hàm mất mát và tối ưu mô hình	17
3.3 Long Short-Term Memory	19
3.3.1 Cấu trúc của LSTM	19
3.3.2 Embedding	20
3.3.3 Dropout	21
3.3.4 Dense	21
3.4 Transformers	21
3.4.1 Embedding	22
3.4.2 Cấu trúc của Encoder	23

<b>4</b>	<b>THỰC HIỆN ĐỀ TÀI</b>	<b>26</b>
4.1	Đọc dữ liệu	26
4.2	Tiền xử lý dữ liệu	27
4.3	Huấn luyện mô hình	29
4.3.1	SVM và Logistic Regression	29
4.3.2	LSTM và Transformers	29
4.4	Kết quả	31
4.4.1	SVM	31
4.4.2	Logistic Regression	33
4.4.3	LSTM	35
4.4.4	Transformers	36
<b>5</b>	<b>KẾT LUẬN</b>	<b>40</b>
5.1	Nhận xét về kết quả	40
5.1.1	Về hướng xây dựng trên các mô hình Machine Learning truyền thống	40
5.1.2	Về hướng xây dựng trên các mô hình Deep Learning	41
5.2	Tổng kết	41

---

# Danh sách hình vẽ

1.1	Tin giả 1	1
1.2	Tin giả 2	2
2.1	Bộ dữ liệu LIAR-MASTER	6
2.2	Phân phối nhãn trong bộ dữ liệu LIAR	6
2.3	Phân phối phần trăm nhãn trong tập train bộ dữ liệu LIAR	6
2.4	Tokenization	7
2.5	Phân phối số kí tự, từ và câu trong bộ dữ liệu LIAR	7
3.1	Xây dựng đường phân loại tối ưu	12
3.2	Mô hình phân loại Logistic Regression.	16
3.3	Mô hình LSTM.	19
3.4	Kiến trúc transformers.	22
3.5	Encoder.	23
3.6	Đầu vào của lớp Multi-head Attention	24
3.7	Công thức tính $Z_i$	24
3.8	Công thức tính $Z$ .	24
4.1	Dữ liệu sau đọc vào chương trình	26
4.2	Dữ liệu sau khi thay đổi các nhãn	27
4.3	Các bước làm sạch văn bản	28
4.4	Bộ từ điển và số lần xuất hiện của từ đó trong dataset	28
4.5	Chỉ số TF-IDF của các từ/cụm từ trong từ điển	29
4.6	Minh họa cách hoạt động tokenizer BERT và RoBERTa trên một câu mẫu	29
4.7	Từ điển sau khi được tạo.	30
4.8	Văn bản sau khi mã hóa số nguyên.	30
4.9	Văn bản sau khi padding.	31
4.10	Kết quả mô hình SVM với tập test LIAR (1)	31
4.11	Kết quả mô hình SVM với tập test LIAR (2)	32
4.12	Confusion matrix SVM	32
4.13	Đường cong ROC cho mô hình SVM	33
4.14	Kết quả mô hình Logistic Regression với tập test LIAR (1)	33
4.15	Kết quả mô hình Logistic Regression với tập test LIAR (2)	34
4.16	Đường cong ROC mô hình Logistic Regression	34
4.17	Confusion matrix for Logistic Regression	35
4.18	Kết quả sau khi huấn luyện mô hình LSTM với số chiều của embedding là 8	35
4.19	Confusion matrix for LSTM	36
4.20	Biểu đồ so sánh độ chính xác với những số chiều khác nhau của lớp embedding	36

4.21	Kết quả khi huấn luyện mô hình với 8 head ở lớp Multi-head Attention. . . . .	37
4.22	Confusion matrix for Transformers. . . . .	37
4.23	Biểu đồ độ chính xác. . . . .	38
4.24	Biểu đồ mất mát. . . . .	38
4.25	Biểu đồ độ chính xác với số head khác nhau ở lớp Multi-head Attention. . . . .	39

---

## Chương 1

# GIỚI THIỆU CHUNG

### 1.1 Đặt vấn đề

Thuật ngữ "tin giả" là một khái niệm tương đối mới và cho đến nay vẫn chưa có một định nghĩa chung được thống nhất về tin tức giả mạo hay tin giả (Fake News). Theo từ điển Oxford "Tin giả là thông tin sai sự thật được phát sóng hoặc xuất bản dưới dạng tin tức nhằm mục đích lừa đảo hoặc có động cơ chính trị. Tin giả tạo ra sự nhầm lẫn đáng kể của công chúng về các sự kiện hiện tại. Tin giả bùng nổ trên phương tiện truyền thông xã hội, đang xâm nhập vào các kênh truyền thông chính". Tin tức giả mạo cũng đề cập đến những câu chuyện bịa đặt có rất ít hoặc không có sự thật và khó có thể xác minh được. [3].

#### 1.1.1 Ví dụ

Một số hình ảnh về tin giả



Hình 1.1: Tin giả 1  
Việc tiêm vaccine phòng COVID-19 tại TP Hồ Chí Minh.



Hình 1.2: Tin giả 2

Công tác phòng, chống dịch và hỗ trợ người dân TP Hồ Chí Minh.

### 1.1.2 Phân loại

Các trường hợp điển hình của tin giả bao gồm quảng cáo lừa đảo (trong kinh doanh và chính trị), tuyên truyền của chính phủ, các hình ảnh chỉnh sửa hoặc dùng sai mục đích ban đầu, tài liệu giả mạo, bản đồ giả, gian lận trên Internet, các trang web giả mạo và mục từ trên Wikipedia không đúng sự thật,... Tin giả có thể gây ra tác hại đáng kể nếu mọi người để nó lừa dối. Để giải quyết mối đe dọa này đối với chất lượng thông tin, trước tiên chúng ta cần hiểu chính xác các loại tin giả.

- **Thông tin sai lệch (Mis-information):** Thông tin sai lệch được phổ biến mà không có ý định gây hại. Thông tin sai lệch có 2 loại:
  - **Kết nối sai (False connection):** Khi dòng tiêu đề, hình ảnh hoặc chú thích không phù hợp với nội dung.
  - **Nội dung gây hiểu lầm (Misleading content):** Sử dụng sai thông tin và gây hiểu lầm cho người đọc. Ví dụ, nội dung quảng cáo hoặc trang web cố gắng đánh lừa khách hàng để truy cập vào các trang web không an toàn.
- **Thông tin giả mạo (Dis-information):** Được tạo và chia sẻ bởi những người có ý định gây hại.
  - **Bối cảnh sai (False context):** Loại thông tin giả mạo này được sử dụng để mô tả nội dung xác thực nhưng đã được điều chỉnh lại theo những cách nguy hiểm. Ví dụ, lợi dụng sự cố Formosa xả thải gây ra hiện tượng cá chết hàng loạt tại vùng biển khu vực các tỉnh bắc miền Trung, nhiều bản tin đã lồng ghép các ý đồ chính trị để kích động, chống phá chế độ.
  - **Nội dung mạo danh (Imposter content):** Là những nội dung sai sự thật hoặc gây hiểu lầm bằng cách sử dụng các biểu tượng nổi tiếng hoặc tin tức từ các nhân vật hoặc nhà báo có uy tín. Ví dụ, ở Việt Nam trong thời gian gần đây, các nhãn hàng đã mời các nghệ sĩ nổi tiếng quảng cáo sai sự thật đã trở thành một vấn nạn và gây khó khăn cho sự lựa chọn của khách hàng.



- **Nội dung bị thao túng (Manipulated content):** Nội dung bị thao túng là khi một khía cạnh nào đó của nội dung chính hãng bị thay đổi. Điều này thường liên quan đến ảnh hoặc video.
- **Nội dung bịa đặt (Fabricated content):** Nội dung bịa đặt là sai 100%.
- **Thông tin độc hại (Mal-information):** Chia sẻ thông tin "chính hãng" nhưng với mục đích gây hại
  - **Rò rỉ (Leaks):** Rò rỉ thông tin là một sự kiện diễn ra khi thông tin bí mật được tiết lộ cho những người hoặc bên không có thẩm quyền.
  - **Quấy rối (Harassment):** Là bất kỳ hành vi nào, dù bằng lời nói, hình ảnh, văn bản hay cách khác nhằm mục đích xúc phạm hoặc làm nhục một cá nhân, tổ chức nào đó.
  - **Gây chia rẽ, thù hận (Hate speech):** Những nội dung biểu hiện qua lời nói, văn bản hoặc các biểu hiện khác thể hiện sự căm thù, phỉ báng một người hoặc những người khác. Các nội dung gây chia rẽ, thù hận thường dựa trên một nhóm xã hội được xác định bởi các thuộc tính như chủng tộc, dân tộc, giới tính, khuynh hướng tình dục, tôn giáo, tuổi tác, khuyết tật về thể chất hoặc tinh thần.

## 1.2 Thực trạng

Hành vi đăng tải thông tin xuyên tạc, sai sự thật trên mạng xã hội, đã thu hút sự quan tâm của dư luận thời gian qua. Đây cũng là nội dung được quan tâm nhất trong phiên chất vấn với Bộ trưởng Thông tin và Truyền thông tuần qua ở diễn đàn Quốc hội.

Tin giả nguy hại đến mức, người phát ngôn Bộ Công an phải lên tiếng phủ nhận việc cấm xuất cảnh đối với một doanh nhân tại phiên họp báo Chính phủ. Có những tin giả làm nhà đầu tư và doanh nghiệp thiệt hại hàng chục, hàng trăm tỷ đồng.

Hai năm trở lại đây, các cơ quan chức năng quản lý đã ra gần 600 quyết định xử phạt vi phạm hành chính các cá nhân, tổ chức có hành vi tung tin giả, tin sai sự thật với tổng số tiền hơn 6 tỷ đồng. Tuy nhiên, số tiền phạt này quá nhỏ so với tác hại mà tin giả gây ra đối với xã hội. Thậm chí tình trạng này vẫn không giảm, nhất là việc đăng tải thông tin gây ảnh hưởng xấu tới hoạt động doanh nghiệp, tác động tiêu cực tới thị trường tài chính.

Liên quan đến các vụ việc tung tin sai sự thật trên các trang mạng, cơ quan công an đã khởi tố 63 vụ với 68 bị can, xử phạt hành chính 455 đối tượng và làm việc với khoảng 1.500 đối tượng. Rà quét liên tục 24/7 trên không gian mạng, qua đó phát hiện và cảnh báo kịp thời cho 63 tỉnh, thành phố các tin giả, thông tin xấu độc liên quan đến từng địa phương để nhanh chóng xử lý. Đó là 1 trong những giải pháp hiện nay mà Bộ Thông tin và Truyền thông đã triển khai nhằm xử lý kịp thời các thông tin sai sự thật, tin đồn thất thiệt.

## 1.3 Mục tiêu

Sự xuất hiện tràn lan của tin giả gây ra những nhầm lẫn và thậm chí ảnh hưởng tiêu cực đến nhận thức cộng đồng, thậm chí gây ra những hậu quả nghiêm trọng về mặt xã hội, kinh tế, và chính trị. Vì vậy, việc nghiên cứu và phát triển các phương pháp hiệu quả để nhận diện tin giả là vô cùng cần thiết trong bối cảnh hiện tại. Chúng ta có thể nhận diện tin giả bằng cách truyền

thông: tự kiểm chứng thông tin, tuy nhiên phương pháp này khá tốn thời gian. Bên cạnh đó, các phương pháp truyền thống chủ yếu dựa vào sự phán đoán của con người. Điều này dẫn đến sự thiếu khách quan và có thể xảy ra sai sót do con người bị ảnh hưởng bởi thành kiến cá nhân hoặc thiếu thông tin. Các công cụ công nghệ hiện nay (VD: AI) được phát triển và giúp ích cho con người rất nhiều giúp chúng ta tiết kiệm về mặt thời gian và thực hiện việc thu thập một nguồn thông tin lớn để phân tích. Với chủ đề báo cáo lần này, chúng tôi đã tiến hành sử dụng các phương pháp AI để nhận diện tin giả.

---

## Chương 2

# PHƯƠNG PHÁP THỰC HIỆN

Tương tự các phương pháp xây dựng mô hình AI truyền thống, chúng tôi thực hiện đề tài với các bước như sau:

1. **Thu thập và xử lý dữ liệu:** Thu thập các bài báo có nhãn fake news và real news để tạo ra tập dữ liệu. Tiền xử lý dữ liệu bao gồm các bước như loại bỏ các từ dừng (stop words), chuẩn hóa văn bản, loại bỏ các ký tự đặc biệt và chuyển đổi chúng thành dữ liệu số hóa,...
2. **Huấn luyện mô hình:**
3. Chia dữ liệu: Chia dữ liệu thành 3 phần train-test-valid, ở bài này, ta dùng tỉ lệ 80-10-10
4. Xây dựng mô hình: Sử dụng các phương pháp Học máy và Học sâu
5. Tối ưu hóa hàm mất mát
6. **Đánh giá mô hình:** Sử dụng tập dữ liệu kiểm tra để đánh giá mô hình SVM. Đánh giá bao gồm các chỉ số như độ chính xác, ví dụ như độ phủ, độ precision và recall. Chúng ta cũng có thể sử dụng các phương pháp đánh giá như cross-validation hoặc sử dụng tập dữ liệu kiểm tra.
7. **Sử dụng mô hình để dự đoán:** Sử dụng mô hình để dự đoán xem một bài báo có phải là fake news hay không.

### 2.1 Thu thập và phân tích dữ liệu

LIAR là một tập dữ liệu được sử dụng để phát hiện tin tức giả, gồm 12800 câu tuyên bố ngắn được gán nhãn bởi con người từ API của trang web politifact.com, nơi cung cấp báo cáo phân tích chi tiết và liên kết đến các tài liệu nguồn cho từng trường hợp. Bộ dữ liệu này cũng có thể được sử dụng cho nghiên cứu kiểm tra thực tế.

Dataset của LIAR được sử dụng, đã được kiểm định và đã được tiền xử lý. File train gồm có 15 cột bao gồm: ID của bài báo, nội dung bài báo, chủ đề, người viết, chức danh người viết, bối cảnh (nơi diễn ra/địa điểm phát biểu), phần lý giải trích dẫn,...

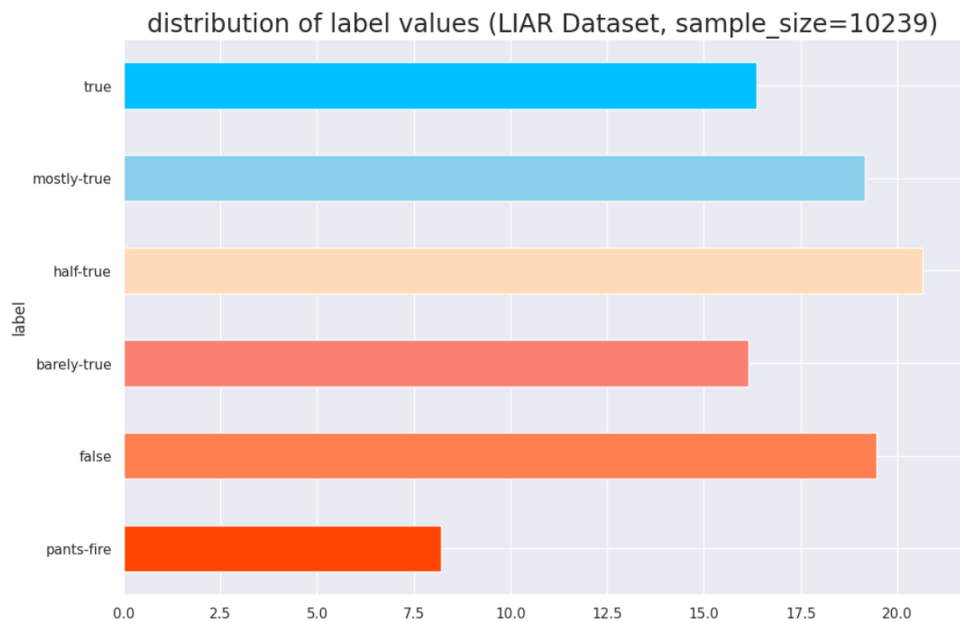
12134.json	barely-true	We have less Ar economy jobs	vicky-hartzler	U.S. Representa	Missouri	republican	1	0	1	0	0	an interview with ABC17 News
238.json	pants-fire	When Obama w obama birth cert	chain-email			none	11	43	8	5	105	
7891.json	FALSE	Says Having org campaign-financ	earl-blumenauer	U.S. representat	Oregon	democrat	0	1	1	1	0	a U.S. Ways and Means hearing
8169.json	half-true	Says nearly half poverty	jim-francesconi	Member of the S	Oregon	none	0	1	1	1	0	an opinion article
929.json	half-true	On attacks by Ri economy.stimulu	barack-obama	President	Illinois	democrat	70	71	160	163	9	Interview with CBS News
9416.json	FALSE	Says when arme guns	jim-rubens	Small business (	New Hampshire	republican	1	1	0	1	0	In an interview at gun shop in Hudson, N.H.
6861.json	TRUE	Says Tennessee education.state-i	andy-berke	Lawyer and stati	Tennessee	democrat	0	0	0	0	0	a letter to state Senate education committee chairv
1122.json	FALSE	The health care health-care	club-growth			none	4	5	4	2	0	a TV ad
13138.json	TRUE	Says Donald Tru candidates-biog	hillary-clinton	Presidential can	New York	democrat	40	29	69	76	7	the first presidential debate
1080.json	half-true	Bill White has a military	republican-party	txas	Texas	republican	3	1	1	3	1	an e-mail
12803.json	half-true	John McCain's cl economy	tim-kaine	U.S. Senator	Virginia	democrat	8	3	15	15	0	a speech at the Democratic National Convention in
5409.json	FALSE	Says 21,000 Wle job-accomplishm	kathleen-vinehout			democrat	1	1	1	1	0	remarks
7313.json	half-true	State revenue pr state-budget	steve-henson	State Senator	Georgia	democrat	0	0	1	0	0	a press release
4809.json	TRUE	The median incc income.new-han	joe-biden	U.S. senator	Delaware	democrat	11	10	21	16	4	speaking at New Hampshire's Plymouth State Univ
1671.json	barely-true	Every citizen is e gays-and-lesbia	david-dewhurst	Lieutenant gover	Texas	republican	8	8	10	5	5	a press release
4348.json	half-true	Rick Perry has a medicaid.social-i	margaret-carlsor	Columnist	District of Colum	none	0	0	1	0	0	a politics column.
6225.json	half-true	Two thirds to thn health-care.pov	elizabeth-robert	Lieutenant Gove	Rhode Island	democrat	1	0	2	0	0	a panel discussion on "A Lively Experiment"
7675.json	mostly-true	Congress has sg congress	john-barrow	Congressman	Georgia	democrat	0	0	1	1	0	a letter
2255.json	barely-true	Mark Sharpe has candidates-biog	mark-sharpe	Hillsborough Cor	Florida	republican	1	0	0	0	0	a campaign mailer
9827.json	pants-fire	Says Iowa Gov. immigration	chain-email			none	11	43	8	5	105	a chain email
12366.json	half-true	If you dont buy c taxes	philadelphia-daily-news			none	0	0	1	0	0	In an editorial
10337.json	pants-fire	Says President l civil-rights.crime	rudolph-giuliani	Attorney	New York	republican	9	11	10	7	3	an interview on Fox News
4778.json	TRUE	Georgia has hac bankruptcy	lynn-westmoreland			republican	1	1	3	0	0	a meeting

Hình 2.1: Bộ dữ liệu LIAR-MASTER

Dữ liệu của LIAR được chia làm 6 nhãn khác nhau dựa theo tính chính xác của tin tức, bao gồm: TRUE (tin thật), MOSTLY-TRUE (tin gần thật), HALFLY-TRUE (tin nửa thật), BARELY TRUE (tin gần không thật), FALSE (tin giả) và PANTS-FIRE (tin lừa đảo)

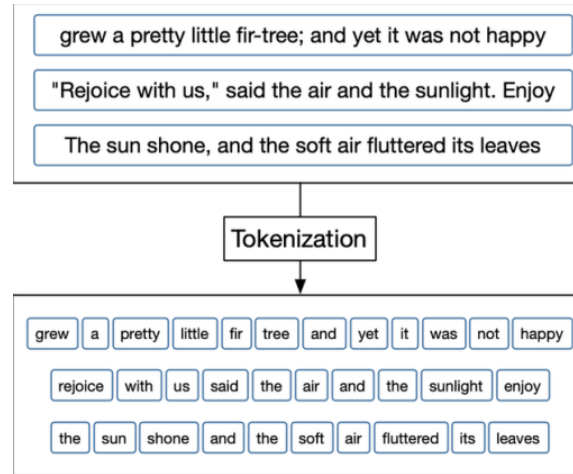
data	true	mostly-true	half-true	barely-true	false	pants-fire	total
train	1676	1962	2114	1654	1995	839	10240
test	208	241	265	212	249	92	1267
valid	169	251	248	237	263	116	1284

Hình 2.2: Phân phối nhãn trong bộ dữ liệu LIAR



Hình 2.3: Phân phối phần trăm nhãn trong tập train bộ dữ liệu LIAR

Trước khi được đưa vào mô hình học máy, nội dung của tin mẫu được chia thành các từ nhỏ (Tokenization) và sau đó đưa về dạng cơ bản nhất (Lemmatization). Trong bước này ta cũng có thể phân tích được số từ, câu và kí tự trong tập dữ liệu



Hình 2.4: Tokenization

	count	mean	std	min	25%	50%	75%	max
<b>characters</b>	12791.0	107.161520	63.452113	11.0	73.0	99.0	133.0	3192.0
<b>words</b>	12791.0	20.210695	11.457018	2.0	14.0	19.0	25.0	546.0
<b>sentences</b>	12791.0	1.167383	0.563874	1.0	1.0	1.0	1.0	19.0

Hình 2.5: Phân phối số kí tự, từ và câu trong bộ dữ liệu LIAR

## 2.2 Trích xuất đặc trưng

Trích xuất đặc trưng trong văn bản là quá trình phân tích và lấy ra những thông tin quan trọng hoặc có giá trị từ một đoạn văn bản. Đây là một bước quan trọng trong xử lý ngôn ngữ tự nhiên (NLP) và có thể được sử dụng trong nhiều ứng dụng như phân loại văn bản, tóm tắt văn bản, nhận diện thực thể, và phân tích cảm xúc.

Với mô hình Machine Learning truyền thống, chúng em sử dụng các kĩ thuật truyền thống

### 2.2.1 TF-IDF

Chỉ số TF-IDF (Term Frequency-Inverse Document Frequency) là một phương pháp được sử dụng để đánh giá tầm quan trọng của một từ trong một văn bản. Chỉ số này được tính bằng cách nhân tần suất xuất hiện của một từ trong văn bản (TF) với độ quan trọng của từ đó trong tập các văn bản (IDF).

Công thức tính toán chỉ số TF-IDF cho một từ trong văn bản như sau:

$$\text{TF-IDF} = \text{tf} * \text{idf}$$

Trong đó TF (Term Frequency) là tần số xuất hiện của một từ trong văn bản. Công thức tính toán TF là:

$$\text{tf}(t, d) = \frac{\text{Tần xuất của từ } t \text{ trong văn bản } d}{\text{Tần xuất của từ xuất hiện nhiều nhất trong văn bản } d} = \frac{f(t, d)}{\max(f(w, d) : w \in d)}$$

Còn chỉ số IDF (Inverse Document Frequency) là độ quan trọng của từ đó trong tập các văn bản. Công thức tính toán IDF là:

$$\text{idf}(t, D) = \log \frac{\text{Tổng số văn bản}}{\text{Số văn bản chứa từ mà ta đang xét}} = \frac{f(t, d)}{\max(f(w, d) : w \in d)}$$

Ví dụ: giả sử có một văn bản có nội dung sau:

"Machine learning is a subfield of artificial intelligence that focuses on designing algorithms and statistical models that allow computer systems to improve their performance on a specific task over time."

Ta cần tính chỉ số TF-IDF của từ "Machine" thì khi đó.

- Tần số xuất hiện của từ "machine" trong văn bản là 1.
- Tổng số từ trong văn bản là 20,

Do đó chỉ số TF của văn bản sẽ là 0,05.

Giả sử trong tập dữ liệu của chúng ta có 100 văn bản và từ "machine" xuất hiện trong 20 văn bản. Khi đó: idf của từ Machine là 1,609.

Vậy TF - IDF của từ "Machine" sẽ là  $0,05 * 1,609 = 0,0805$ .

### 2.2.2 Count Vectorization

Xét 3 tài liệu sau:

- **Doc 1:** "I love rock music"
- **Doc 2:** "Rock music is great"
- **Doc 3:** "I love great music"

Áp dụng Count Vectorization, từ điển có dạng như sau:

"I" : 0  
 "love" : 1  
 "rock" : 2  
 "music" : 3  
 "is" : 4  
 "great" : 5

Ma trận kết quả có dạng như sau:

	"I"	"love"	"rock"	"music"	"is"	"great"
Doc 1	1	1	1	1	0	0
Doc 2	0	0	1	1	1	1
Doc 3	1	1	0	1	0	1

## 2.3 Xây dựng các mô hình theo hướng Machine Learning

Dưới đây, chúng tôi sẽ trình bày hai cách tiếp cận theo hướng ML để xử lý bài toán Nhận diện tin giả: Support Vector Machine, Logistic Regression

### *Tại sao lựa chọn Logistic Regression?*

Hồi quy logistic là một thuật toán học máy có giám sát thực hiện các nhiệm vụ phân loại nhị phân bằng cách dự đoán xác suất của một kết quả, sự kiện hoặc quan sát. Mô hình cung cấp kết quả nhị phân hoặc nhị phân giới hạn ở hai trạng thái: có hoặc không, 0 hoặc 1, đúng hoặc sai. Hồi quy logic phân tích mối quan hệ giữa một hoặc nhiều biến độc lập và phân loại dữ liệu thành các lớp rời rạc. Do việc cung cấp kết quả nhị phân nên mô hình Logistic phù hợp với việc phân loại giữa đúng và sai bao gồm việc phân biệt tin thật và tin giả (fake news detection).

### *Tại sao lại lựa chọn SVM?*

Support Vector Machine (SVM) là một công cụ phân loại và dự đoán hồi quy sử dụng lý thuyết máy học để tối đa hóa độ chính xác của dự đoán đồng thời tự động tránh dữ liệu quá khớp. Support Vector Machine có thể được định nghĩa là các hệ thống sử dụng không gian giả thuyết của các hàm tuyến tính trong không gian đặc trưng nhiều chiều, được đào tạo bằng thuật toán học từ lý thuyết tối ưu hóa thực hiện xu hướng học bắt nguồn từ lý thuyết học thống kê.

SVM trở nên nổi tiếng khi sử dụng bản đồ pixel làm đầu vào; nó mang lại độ chính xác tương đương với các mạng thần kinh tinh vi với các tính năng phức tạp trong tác vụ nhận dạng chữ viết tay. Nó cũng đang được sử dụng cho nhiều ứng dụng, chẳng hạn như phân tích chữ viết tay, phân tích khuôn mặt, v.v., đặc biệt đối với các ứng dụng dựa trên phân loại mẫu và hồi quy.

SVM được phát triển để giải quyết vấn đề phân loại, nhưng gần đây chúng đã được mở rộng để giải các bài toán hồi quy.

## 2.4 Xây dựng mô hình theo hướng Deep Learning

Dưới đây, chúng tôi sẽ trình bày hai cách tiếp cận theo hướng Deep Learning để xử lý bài toán Nhận diện tin giả: LSTM(Long Short-Term Memory) và kiến trúc Transformers.

### *Tại sao lựa chọn LSTM?*

LSTM (Long Short-Term Memory) là một loại mạng nơ-ron hồi tiếp (recurrent neural network - RNN) được phát triển để giải quyết một số vấn đề chính mà các mô hình RNN truyền thống gặp phải, đặc biệt là vấn đề của "gradient vanishing" và "gradient exploding" trong quá trình huấn luyện.

LSTM được thiết kế đặc biệt để lưu trữ và duy trì thông tin qua các khoảng thời gian dài. Điều này cực kỳ quan trọng trong các bài toán mà thông tin từ các bước trước có thể ảnh hưởng lớn đến các bước sau, chẳng hạn như trong các chuỗi thời gian dài hoặc các văn bản dài.

Trong RNN truyền thống, các gradient có thể trở nên quá nhỏ (gradient vanishing) hoặc quá lớn (gradient exploding), làm cho việc huấn luyện trở nên khó khăn. LSTM sử dụng cơ chế cổng (gates) để điều chỉnh thông tin được truyền qua mạng, giúp ngăn ngừa những vấn đề này.

LSTM bao gồm ba cổng chính - cổng đầu vào, cổng quên, và cổng đầu ra. Những cổng này cho phép LSTM quyết định thông tin nào cần giữ lại, thông tin nào cần loại bỏ, và thông tin nào cần đưa vào đầu ra. Điều này giúp mạng học cách quản lý và duy trì thông tin một cách hiệu quả hơn.

LSTM đã chứng minh hiệu quả trong nhiều ứng dụng khác nhau, từ phân tích chuỗi thời gian, nhận diện giọng nói, dịch máy, cho đến sinh văn bản. Khả năng của LSTM trong việc xử lý và dự đoán các chuỗi dữ liệu làm cho nó rất linh hoạt và hữu ích trong các bài toán liên quan đến dữ liệu tuần tự. Các mạng LSTM có khả năng học và dự đoán các mẫu dài hạn trong dữ liệu chuỗi, điều mà các RNN truyền thống thường gặp khó khăn.

### *Kiến trúc Transformers*

Với sự ra đời của cơ chế attention thì vào năm 2017 paper 'Attention is all you need' đã giới thiệu một kiến trúc mới dành cho các bài toán NLP mà không có sự xuất hiện của các mạng nơ-ron hồi tiếp (RNN, LSTM,...) hay là mạng nơ-ron tích chập (CNN) - đó là Transformers.

Transformers là một kiến trúc mô hình học sâu dựa trên cơ chế self-attention, cho phép mô hình này hiểu được mối quan hệ giữa các từ trong một câu mà không cần đến kiến trúc tuần tự truyền thống như RNN (Recurrent Neural Networks) hay LSTM (Long Short-Term Memory). Nó có khả năng xử lý toàn bộ câu cùng một lúc, điều này giúp tăng tốc độ huấn luyện và cải thiện hiệu quả xử lý.



Transformers đã cách mạng hóa cách chúng ta xử lý các bài toán về ngôn ngữ tự nhiên và nhanh chóng trở thành nền tảng cho nhiều mô hình tiên tiến hiện nay, như BERT, GPT, T5, và nhiều mô hình khác.

## 2.5 Huấn luyện và kiểm thử mô hình

Sau bước trích chọn đặc trưng thì ta sẽ đưa dữ liệu vào mô hình để đào tạo. Những chỉ số cần tính trong các mô hình được đề cập ở phần sau.

Sau khi đã huấn luyện xong ta sẽ dùng bộ dữ liệu test để thử nghiệm độ hiệu quả của mô hình. Tập test này phải là tập test chưa có trong bộ dữ liệu huấn luyện để đảm bảo tính đúng đắn

Những chỉ số sẽ được dùng để đánh giá độ hiệu quả của mô hình bao gồm Accuracy, Recall, Precision, F1-score.

---

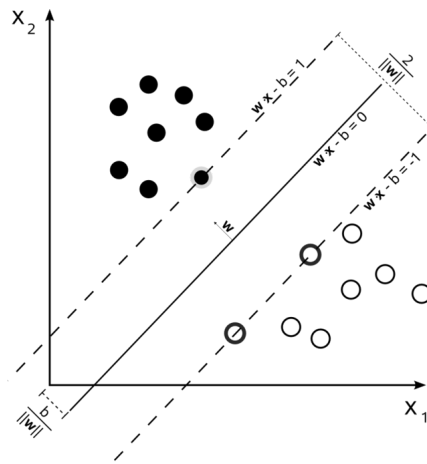
## Chương 3

# XÂY DỰNG MÔ HÌNH

Trong phần này, chúng tôi sẽ triển khai 4 mô hình phân loại như đã giới thiệu sơ lược ở chương 2.

### 3.1 Support Vector Machine (SVM)

Trong phần này, chúng ta sẽ nói về SVM và lý do tại sao chúng tôi sử dụng SVM? Với một cái nhìn tổng quan ngắn gọn về lý thuyết học thống kê. Công thức toán học của SVM được trình bày và lý thuyết về việc triển khai SVM được thảo luận ngắn gọn.



Hình 3.1: Xây dựng đường phân loại tối ưu

Với tập dữ liệu trên không gian 2-D. Ta xây dựng 2 siêu phẳng (cụ thể trong trường hợp này là đường thẳng) sao cho giữa 2 siêu phẳng này không có bất kỳ điểm dữ liệu nào. Gọi  $H$  là vùng không gian giới hạn bởi 2 siêu phẳng ta vừa xây dựng. Vì là tuyến tính nên phương trình của  $H$  có dạng:

$$H = \begin{cases} w_i x_i + b \leq 1 & \text{nếu } y_i = 1 \\ w_i x_i + b \geq -1 & \text{nếu } y_i = -1 \end{cases} \quad (3.1)$$

Đặt  $H_1 : W^t + b = 1$  và  $H_2 : W^t + b = -1$ . Kẻ một đường thẳng  $H_0$  cách đều và song song với  $H_1$  và  $H_2$  có phương trình là:

$$H_0 : W^t + b = 0$$

Trong đó:

- $W$ : là ma trận trọng số của đường thẳng cần tìm.  $b$  là hệ số của đường thẳng.  $x_i$  và  $y_i$  là các đặc trưng cũng như là nhãn của dữ liệu.
- $H_0$ : là đường phân loại tốt nhất cần phải tìm.

Sau đó khi đã có phương trình đường thẳng  $H_0$ , ta hoàn toàn có thể đưa ra dự đoán bằng cách xác định dấu của tập dữ liệu mới dựa vào phương trình của  $H_0$ :

$$\hat{y} = \text{sign}(W^t + b)$$

### 3.1.1 Tìm hệ số của đường thẳng phân loại

Gọi đường thẳng  $H_1$  và  $H_2$  là lề (margin) của vùng không gian  $H$ . Để tìm được đường phân loại tốt nhất chúng ta phải tối đa hóa được khoảng cách này.

Sở dĩ phải cho khoảng cách này là tối đa vì đường  $H_0$  phụ thuộc vào đường  $H_1$  và đường  $H_2$  nên nếu khoảng cách giữa 2 đường này là cực đại chứng tỏ ranh giới giữa 2 loại là lớn. Giải thích rõ hơn là sự khác nhau giữa 2 tập dữ liệu càng rõ ràng. Thử dụ nếu như cùng 1 đường thẳng đó nhưng lại nằm ở các vị trí gần loại 1 quá hoặc gần loại 2 quá thì dù cho việc phân loại của chúng ta vẫn đúng nhưng khả năng tổng quát sẽ sai lệch. Nói một cách dễ hiểu thì đường phân loại đúng với tập training nhưng sẽ sai lệch với tập test.

Vậy tìm khoảng cách đó như nào?

Vì  $H_0$  nằm chính giữa  $H_1$  và  $H_2$  nên khoảng cách từ  $H_0$  đến  $H_1$  bằng khoảng cách  $H_0$  đến  $H_2$ . Nên chỉ cần tìm một trong hai khoảng cách chúng ta có thể suy ra được khoảng cách tổng của các lề.

Công thức khoảng cách giữa 2 đường thẳng:

$$D(H, I) = \frac{|Ax + By + C|}{\sqrt{A^2 + B^2}}$$

Với  $H$  là đường thẳng có phương trình  $H = Ax + By + C$  và điểm  $I$  có tọa độ  $(x, y)$

Áp dụng trong trường hợp này, ta được:  $\frac{wx_0 + b}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$

Do đó khoảng cách lề tối đa là :  $\frac{2}{\|\mathbf{w}\|}$ . Khoảng cách này đạt cực tiểu khi và chỉ khi  $\|\mathbf{w}\|$  đạt cực tiểu.

Ngoài ra từ phương trình 3.1. Ta có thể viết gọn thành một biểu thức đó là:

$$y_i(w * x_i + b) \geq 1.$$

Từ đây ta xây dựng được bài toán tối ưu của thuật toán SVM tuyến tính:

$$f(x) = \frac{1}{2} * |w^t| * |w| \longrightarrow \min \quad \text{st:} \quad y_i(w * x_i + b) - 1 = 0 \quad (3.2)$$

### 3.1.2 Giải bài toán tối ưu SVM

#### Phương pháp nhân tử Lagrange

Để giải bài toán trên ta sử dụng phương pháp nhân tử Lagrange. Phương pháp được phát biểu là:

1. Ta muốn tìm cực tiểu của hàm  $f(x,y)$  với ràng buộc là  $\phi_i = 0$ .
2. Thiết lập hàm Lagrange có dạng  $L(x, y) = f(x, y) + \alpha \phi(x, y)$  với  $\alpha$  là vector hệ số Lagrange.
3. Tìm điểm dừng của  $L(x, y)$  tức là giải hệ phương trình

$$\begin{aligned} L'_x(x, y, \alpha) &= 0 \\ L'_y(x, y, \alpha) &= 0 \\ L'_\alpha(x, y, \alpha) &= 0 \end{aligned} \quad (3.3)$$

4. Xét dấu đạo hàm bậc 2 của hàm  $L$  tại điểm  $(x_0; y_0)$  mà  $(x_0; y_0; 0)$  là nghiệm của hệ phương trình

$$L''(x_0; y_0; \alpha_0) > 0 = f(x_0; y_0) \text{ Từ đó hàm số đạt cực tiểu.}$$

#### Áp dụng để giải bài toán

Hàm Lagrange:  $L(x, b, \alpha) = \frac{1}{2} |w^t| |w| - \sum_{i=1}^N \alpha_i [y_i(w x_i + b)] + \sum_{i=1}^N \alpha_i$  với  $N$  là số điểm dữ liệu.

Giải các hệ phương trình sau để tìm cực tiểu của hàm mục tiêu:

$$\begin{aligned} L'_x(x, b, \alpha) &= w - \sum_{i=1}^N \alpha_i y_i x_i = 0 \\ L'_y(x, b, \alpha) &= \sum_{i=1}^N \alpha_i y_i = 0 \\ L'_\alpha(x, b, \alpha) &= y_i(w * x_i + b) - 1 = 0 \quad (\text{ràng buộc của bài toán}) \end{aligned} \quad (3.4)$$

Từ đây ta suy ra được các nghiệm của bài toán nhân tử Lagrange là:

$$w = \sum_{i=1}^N \alpha_i y_i x_i \quad \text{và} \quad \sum_{i=1}^N \alpha_i y_i = 0 \quad (3.5)$$

### 3.1.3 Biểu thức đối ngẫu

Sau khi giải bài toán nguyên thủy bằng phương pháp nhân tử Lagrange. Ta có thể thấy thuật toán cho nghiệm tối ưu. Nhưng trong biểu thức 3.5 thì ta vẫn chưa biết  $\alpha$  là gì. Đó là lý do mà chúng ta cần phải giải bài toán đối ngẫu để tìm ra vector  $\alpha$  sau đó dùng nó ở trong lời giải của bài toán gốc. [?]

Từ biểu thức 3.5 ta thay vào bài toán ban đầu. Ta có hàm Lagrange như sau:

$$\begin{aligned} L &= \frac{1}{2} \sum_{i,j=1}^N \alpha_{ij} y_{ij} x_i x_j - \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j x_i x_j - b \sum_{i=1}^N \alpha_i y_i + \sum_{i=1}^N \alpha_i \\ &= \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_{ij} y_{ij} x_i x_j \rightarrow \max \end{aligned}$$

Đặt  $H_{ij} = y_i y_j x_i x_j$  và viết lại biểu thức của  $L$  ta được:

$$L = \sum_i \alpha_i - \frac{1}{2} \alpha_i H_{ij} \alpha_j \quad (3.6)$$

Từ đây ta có được bài toán đối ngẫu để tìm được  $\alpha$  là :

$$L = \sum_i \alpha_i - \frac{1}{2} \alpha_i H_{ij} \alpha_j \rightarrow \max \quad \text{st:} \quad \sum_{i=1}^N \alpha_i y_i = 0 \quad (3.7)$$

Phương trình tổng quát của bài toán SVM

Vừa rồi là một ví dụ của SVM được biểu diễn và xây dựng trong không gian 2-D. Thực tế các bài toán có số lượng chiều nhiều hơn hai. Và để xây dựng biểu thức đó ta cần phải tối ưu một phương trình tổng quát. [?]

$$\min ||f_k||^2 + C \sum_{i=1}^N \xi_i \quad y_i f(x_i) \geq 1 - \xi_i \forall i, \xi_i > 0 \quad (3.8)$$

Phương trình 3.8 được gọi là phương trình mềm của bài toán SVM. Đại lượng  $\xi$  được thêm vào với mục đích làm mềm cho dạng đường phân loại, nó cũng đại diện cho sai số của bài toán. Bởi lẽ đâu phải lúc nào ta cũng tìm được siêu phẳng tuyệt đối phân loại cho bài toán mà chúng ta đang xét. Ngoài ra bài toán đối còn được mở rộng với thuật toán nhân đó là:

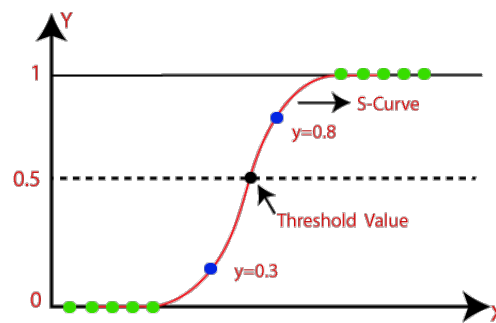
$$\min \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad 0 \leq \alpha_i \leq C \forall i \quad \sum_{i=1}^N \alpha_i y_i = 0 \quad (3.9)$$

Trong đó  $K$  là ma trận mở rộng của thuật toán SVM. Vấn đề này được mở rộng thành một phần mới được gọi là thuật toán nhân.

## 3.2 Logistic Regression

Logistic Regression là một thuật toán học máy thuộc loại phân loại (classification). Mặc dù có tên gọi là "regression" (hồi quy), mô hình này chủ yếu được sử dụng cho các bài toán phân loại nhị phân (binary classification), nghĩa là phân chia dữ liệu thành hai nhóm hoặc lớp khác nhau.

### 3.2.1 Xây dựng thuật toán



Hình 3.2: Mô hình phân loại Logistic Regression.

Gần giống như mô hình SVM, Logistic Regression sẽ phân chia tập dữ liệu thành hai phần với đầu ra biểu diễn dưới dạng nhị phân 0 và 1. Logistic Regression dựa trên ý tưởng sử dụng hàm logic (hàm sigmoid) [2] để biến đổi đầu ra của một tổ hợp tuyến tính thành xác suất.

$$\sigma(s) = \frac{1}{1 + e^{-s}} \quad (3.10)$$

Trong đó:

- $z$  là tổ hợp tuyến tính của các biến độc lập:  $s = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$ .
- $e$  là số Euler (khoảng 2.718).
- $x_1, x_2, \dots, x_n$  là các đặc trưng đầu vào của dữ liệu.
- $\beta_0, \beta_1, \dots, \beta_n$  là các tham số cần học.

Hàm sigmoid có thể được biểu diễn cụ thể hơn trong Logistic Regression như sau:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n)}} \quad (3.11)$$

Trong đó,  $P(y = 1|x)$  là xác suất dữ liệu thuộc lớp 1.

$$\lim_{s \rightarrow -\infty} \sigma(s) = 0; \quad \lim_{s \rightarrow +\infty} \sigma(s) = 1 \quad (3.12)$$

Hàm sigmoid có đầu ra là một giá trị trong khoảng từ 0 đến 1, biểu thị xác suất của dữ liệu thuộc về một lớp nhất định. Quyết định phân lớp được đưa ra dựa trên ngưỡng xác suất, thường là 0.5. Nếu xác suất lớn hơn 0.5, dữ liệu được phân vào lớp 1, ngược lại sẽ phân vào lớp 0.

### 3.2.2 Hàm mất mát và tối ưu mô hình

#### Xây dựng hàm Loss function

Loss function (Hàm mất mát) đo lường sự khác biệt giữa giá trị dự đoán của mô hình và giá trị thực tế (nhân đúng) trong một bài toán cụ thể. Dựa trên giá trị của hàm mất mát, mô hình sẽ được điều chỉnh để cải thiện độ chính xác của dự đoán trong quá trình huấn luyện.

Ta có thể giả sử rằng xác suất để một điểm dữ liệu  $\mathbf{x}$  rơi vào *class* 1 là  $f(\mathbf{w}^T \mathbf{x})$  và rơi vào *class* 0 là  $1 - f(\mathbf{w}^T \mathbf{x})$ . 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:

$$P(y_i = 1 | \mathbf{x}_i; \mathbf{w}) = f(\mathbf{w}^T \mathbf{x}_i) \quad (1) \quad (3.13)$$

$$P(y_i = 0 | \mathbf{x}_i; \mathbf{w}) = 1 - f(\mathbf{w}^T \mathbf{x}_i) \quad (2) \quad (3.14)$$

Trong đó  $P(y_i = 1 | \mathbf{x}_i; \mathbf{w})$  được hiểu là xác suất xảy ra sự kiện đầu ra  $y_i = 1$  khi biết tham số mô hình  $\mathbf{w}$  và dữ liệu đầu vào  $\mathbf{x}_i$ . Mục đích của chúng ta là tìm các hệ số  $\mathbf{w}$  sao cho  $f(\mathbf{w}^T \mathbf{x}_i)$  càng gần với 1 càng tốt với các điểm dữ liệu thuộc *class* 1 và càng gần với 0 càng tốt với những điểm thuộc *class* 0.

Ký hiệu  $z_i = f(\mathbf{w}^T \mathbf{x}_i)$  và viết gộp lại hai biểu thức bên trên ta có:

$$P(y_i | \mathbf{x}_i; \mathbf{w}) = z_i^{y_i} (1 - z_i)^{1-y_i} \quad (3.15)$$

Với tập dữ liệu đầu vào là các biến độc lập với nhau  $\mathbf{X} = [X_1, X_2, \dots, X_N]$ , khi đó ta có xác suất của toàn bộ tập dữ liệu như sau:

$$P(y | \mathbf{X}; \mathbf{w}) = \prod_{i=1}^N P(y_i | x_i; \mathbf{w}) = \prod_{i=1}^N z_i^{y_i} (1 - z_i)^{1-y_i} \quad [1] \quad (3.16)$$

Vectơ trọng số  $\mathbf{w}$  hợp lý nhất tuân theo thuật toán maximum likelihood estimation với hàm số argmax gọi là likelihood function:

$$\mathbf{w} = \arg \max_{\mathbf{w}} P(y | \mathbf{X}; \mathbf{w}) \quad (3.17)$$

Thuật toán maximum likelihood estimation là một phương pháp thống kê dùng để ước lượng tham số của một mô hình xác suất dựa trên dữ liệu quan sát được. Mục tiêu của maximum likelihood estimation là tìm bộ tham số sao cho khả năng mà mô hình với bộ tham số sinh ra là lớn nhất [2]. Khi  $N$  lớn, tích của  $N$  số nhỏ hơn 1 có thể dẫn tới sai số trong tính toán (numerical error) vì tích quá nhỏ sắp xỉ 0. Vì vậy, chúng ta thường lấy logarit tự nhiên cơ số  $e$  biến phép nhân thành phép cộng để tránh việc số quá bé.

$$J(\mathbf{w}) = -\log(P(y | \mathbf{X}; \mathbf{w})) = -\sum_{i=1}^N (y_i \log(z_i) + (1 - y_i) \log(1 - z_i)) \quad [1] \quad (3.18)$$

Dấu trừ được thêm vào để hàm được coi là hàm mất mát được sử dụng trong Logistic Regression. Lúc này bài toán tìm giá trị lớn nhất của thuật toán maximum likelihood estimation trở thành bài toán tìm giá trị nhỏ nhất của hàm mất mát. Hàm này còn gọi là hàm cross-entropy loss. Hàm này đo lường độ sai lệch giữa giá trị dự đoán và giá trị thực tế, và có dạng:

### Tối ưu hàm mất mát

Mục tiêu của quá trình tối ưu hóa là tìm ra các giá trị tham số  $w$  (weights) sao cho hàm mất mát được tối thiểu hóa. Có một số thuật toán phổ biến để thực hiện tối ưu hóa. Trong bài nghiên cứu này, chúng tôi xin trình bày phương pháp Gradient Descent để tối ưu hàm. Gradient Descent là một thuật toán tối ưu dựa trên gradient (đạo hàm) của hàm mất mát. Ý tưởng cơ bản là di chuyển các tham số  $w_j$  theo hướng ngược lại các gradient của hàm mất mát, nhằm làm giảm giá trị hàm này.

$$w := w - \alpha \left( \frac{\partial L(w, x_i, y_i)}{\partial w_i} \right) \quad (3.19)$$

Hàm mất mát với chỉ một điểm dữ liệu  $(x_i, y_i)$ :

$$J(w; x; y_i; z_i) = -(y_i \log(z_i) + (1 - y_i) \log(1 - z_i)) \quad (3.20)$$

Với đạo hàm

$$\frac{\partial J(w; x_i, y_i)}{\partial w} = - \left( \frac{y_i}{z_i} - \frac{1 - y_i}{1 - z_i} \right) \frac{\partial z_i}{\partial w} = \frac{z_i - y_i}{z_i(1 - z_i)} \frac{\partial z_i}{\partial w} \quad (3.21)$$

Với  $z = f(w^T x)$  và đặt  $s = w^T x$ , chúng ta sẽ có:

$$\frac{\partial z_i}{\partial w} = \frac{\partial z_i}{\partial s} \cdot \frac{\partial s}{\partial w} = \frac{\partial z_i}{\partial s} \cdot x \quad (3.22)$$

$$\frac{\partial z}{\partial s} = \frac{\partial f(s)}{\partial s} = z(1 - z) \quad (3.23)$$

Thế phương trình (3.21) và (3.22) vào phương trình đạo hàm, ta có:

$$\frac{\partial J(w; x_i, y_i)}{\partial w} = (z_i - y_i) \cdot x \quad (3.24)$$

Do đó, nếu chỉ có một mẫu huấn luyện  $(x_i, y_i)$  thì ta có quy tắc giảm gradient sau:

$$w_j := w_j - \alpha(z_i - y_i) \cdot x \quad (3.25)$$

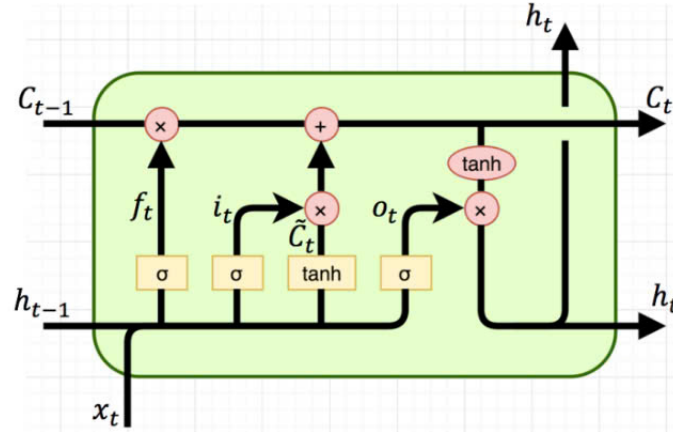
Ta thấy  $w_j$  được cập nhật tỉ lệ với độ lớn của sai số  $(z_i - y_i)$

- Nếu sai số dự báo càng lớn thì trọng số  $w$  càng cần thay đổi nhiều
- Nếu không có sai số thì không cần cập nhật trọng số



### 3.3 Long Short-Term Memory

#### 3.3.1 Cấu trúc của LSTM



Hình 3.3: Mô hình LSTM.

Đầu vào  $x_t$  và đầu ra của bước trước  $h_{t-1}$  được đưa vào ba cổng (cổng đầu vào, cổng quên, và cổng đầu ra).

#### Cổng đầu vào(Input Gate)

Chức năng: Xác định thông tin nào từ đầu vào hiện tại nên được lưu trữ trong trạng thái bộ nhớ của LSTM.

Cách hoạt động:

-Bước 1:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t]) + b_i \quad (3.26)$$

Trong đó:  $\sigma$  là hàm sigmoid cho giá trị từ 0 đến 1

$W_i$  là ma trận trọng số cho cổng đầu vào

$h_{t-1}$  là đầu ra từ bước thời gian trước

$x_t$  là đầu vào hiện tại

$b_i$  là bias

-Bước 2: Tạo các giá trị candidate cho trạng thái bộ nhớ mới:

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (3.27)$$

Trong đó:

tanh: hàm tanh, tạo ra giá trị mới cho trạng thái bộ nhớ

$W_c$ : là ma trận trọng số cho giá trị candidate.

$b_c$ : là bias.

-Bước 3: Cập nhật trạng thái  $C_t$  bằng cách kết hợp các giá trị từ cổng đầu vào và các giá trị candidate:

$$C_t = f_{t-1} \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Trong đó:  $\odot$ : là phép nhân Hadamard (nhân từng phần tử).

### Cổng Quên (Forget Gate)

Chức năng: Quyết định phần nào của thông tin lưu trữ trong trạng thái bộ nhớ hiện tại nên bị quên đi.

Cách hoạt động:

Bước 1: Tính giá trị của cổng quên:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3.28)$$

Trong đó:

$W_f$ : ma trận trọng số cho cổng quên

$b_f$ : bias

Bước 2: Cập nhật trạng thái bộ nhớ hiện tại bằng cách nhân trạng thái bộ nhớ trước đó với cổng quên:

$$C_t = f_t \odot C_{t-1} \quad (3.29)$$

Phần  $i_t \odot$  từ cổng đầu vào sẽ được thêm vào.

### Cổng Đầu Ra (Output Gate)

Chức năng: Quyết định phần nào của trạng thái bộ nhớ sẽ được xuất ra như là đầu ra của mạng.

Cách hoạt động:

-Hàm sigmoid: Tạo ra một vector cổng đầu ra, các giá trị trong vector này quyết định phần nào của trạng thái bộ nhớ sẽ được sử dụng để tạo đầu ra. Hàm sigmoid có dạng:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (3.30)$$

Trong đó  $W_o$  : là ma trận trọng số;  $b_o$ : bias

Hàm tanh: Được áp dụng lên trạng thái bộ nhớ hiện tại để tạo ra giá trị đầu ra, sau đó giá trị này được nhân với cổng đầu ra:

$$h_t = o_t \odot \tanh(C_t) \quad (3.31)$$

trong đó  $\tanh(C_t)$  là trạng thái đã được điều chỉnh và  $h_t$  là đầu ra của LSTM tại thời điểm t.

## 3.3.2 Embedding

-Embedding là một kỹ thuật dùng để ánh xạ các phần tử rời rạc của một không gian (như các từ trong từ vựng) vào một không gian liên tục có kích thước nhỏ hơn. Mục đích là để gán mỗi phần tử một vectơ số, giúp mô hình dễ dàng xử lý và học các quan hệ giữa chúng.

- Quá Trình Embedding:

+) Tạo Ma Trận Embedding:

Khi bạn tạo một lớp embedding trong một mô hình, bạn sẽ khởi tạo một ma trận embedding có kích thước  $[V, D]$ ,

trong đó:

V là số lượng phần tử trong từ vựng (vocabulary size).

D là kích thước của không gian nhúng (embedding dimension).

Mỗi hàng của ma trận này là một vectơ đại diện cho một phần tử trong từ vựng.

+) Nhúng Dữ Liệu: Khi dữ liệu đầu vào là các chỉ số nguyên (như chỉ số của từ trong từ vựng), lớp embedding sử dụng ma trận đã học để chuyển đổi các chỉ số này thành các vectơ nhúng.

Ví dụ: Nếu từ thứ i trong từ vựng được ánh xạ đến chỉ số i, lớp embedding sẽ lấy hàng thứ i trong ma trận embedding để tạo ra vectơ nhúng của từ đó.

+) Kết Nối với LSTM: Các vectơ nhúng sau đó được cung cấp cho LSTM như là đầu vào. LSTM sẽ xử lý các vectơ này để học các mẫu và quan hệ trong chuỗi dữ liệu.

### 3.3.3 Droupout

-Dropout là một phương pháp đơn giản nhưng hiệu quả để ngăn chặn mô hình học quá nhiều chi tiết không cần thiết từ dữ liệu huấn luyện. Trong quá trình huấn luyện, dropout "bỏ qua" một tỷ lệ ngẫu nhiên các neuron hoặc kết nối giữa các lớp trong mạng nơ-ron. Điều này có nghĩa là, trong mỗi lần lặp huấn luyện, một số neuron được "tắt" ngẫu nhiên, và chỉ một phần của mô hình được sử dụng để cập nhật trọng số.

Cách Dropout Hoạt Động a. Trong Quá Trình Huấn Luyện: +)Ngẫu Nhiên Bỏ Qua Neuron: Trong mỗi lần lặp huấn luyện, dropout sẽ chọn ngẫu nhiên một tỷ lệ  $p$  các neuron để bỏ qua (set to zero).

Ví dụ, nếu tỷ lệ dropout là 0.5, thì 50 % các neuron trong lớp đó sẽ được đặt bằng 0 và không tham gia vào quá trình tính toán

+ )Cập Nhật Trọng Số: Những neuron không bị dropout sẽ được sử dụng để tính toán và cập nhật trọng số của mô hình như bình thường.

. -Trong Quá Trình Dự Đoán:Sử Dụng Tất Cả Neuron: Khi mô hình đã được huấn luyện và đang ở chế độ dự đoán (inference), tất cả các neuron sẽ được sử dụng. Để bù đắp cho việc dropout trong quá trình huấn luyện, các trọng số của các neuron sẽ được điều chỉnh bằng cách nhân với tỷ lệ không bị dropout ( $1 - p$ ).

Ví dụ, nếu dropout là 0.5, các trọng số sẽ được nhân với 0.5 trong giai đoạn dự đoán.

### 3.3.4 Dense

-Hàm Dense đại diện cho một lớp nơ-ron đầy đủ kết nối (fully-connected layer), trong đó mỗi nơ-ron trong lớp này kết nối với tất cả các nơ-ron trong lớp trước đó. Lớp Dense thường bao gồm:

Trọng số (Weights): Một ma trận trọng số kết nối các nơ-ron của lớp này với lớp trước đó.

Bias: Một vectơ bias được cộng vào đầu ra của mỗi nơ-ron.

Hàm kích hoạt (Activation Function): Một hàm phi tuyến được áp dụng để đưa ra đầu ra của lớp, giúp mạng nơ-ron học các quan hệ phức tạp trong dữ liệu. -Kết quả đầu ra của lớp Dense có thể được tính theo công thức:

$$\text{Output} = \text{Activation}(\text{Input} \times \text{Weights} + \text{Bias})$$

Trong đó:

Input: Vectơ đầu vào từ lớp trước.

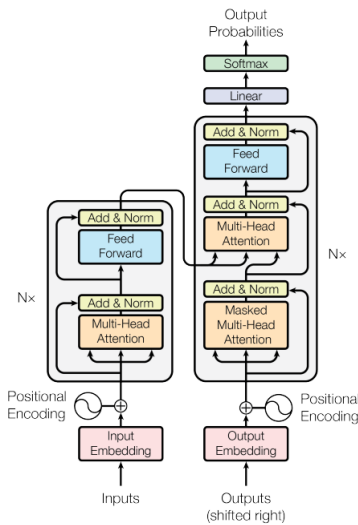
Weights: Ma trận trọng số của lớp hiện tại.

Bias: Vectơ bias.

Activation: Hàm kích hoạt được áp dụng sau khi tính toán tích ma trận và cộng bias.

## 3.4 Transformers

Transformers được cấu trúc thành hai phần chính là encoder và decoder.



Hình 3.4: Kiến trúc transformers.

Encoder: Encoder xử lý dữ liệu đầu vào (gọi là "Source") và nén dữ liệu vào vùng nhớ hoặc context mà Decoder có thể sử dụng sau đó.

Decoder: Decoder nhận đầu vào từ đầu ra của Encoder (gọi là "Encoded input") kết hợp với một chuỗi đầu vào khác (gọi là "Target") để tạo ra chuỗi đầu ra cuối cùng.

Trong mô hình em sẽ chỉ sử dụng phần encoder của transformers.

### 3.4.1 Embedding

Đầu vào của khối encoder là các vector embeddings của các từ trong câu. Đầu vào của các khối encoder còn lại là đầu ra của khối endcoder phía dưới. Các embeddings được tạo thành từ việc kết hợp vector word embedding + positional embedding.

Vector word embedding là vector biểu diễn các từ được tạo ra từ các pre-model như word2vec, glove...

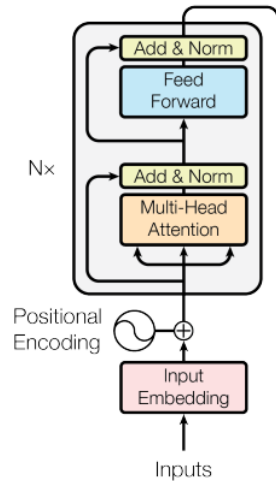
Vector positional embedding là vector biểu diễn thứ tự của các từ trong chuỗi, chúng mang thông tin về vị trí và khoảng cách của các từ. Nếu không có thông tin này việc học biểu diễn cho câu "Trời hôm nay trở lạnh vì gió mùa tràn về" và câu "về tràn gió mù trở lạnh vì hôm nay Trời" sẽ không có gì khác biệt. Rõ ràng vị trí của các từ mang ý nghĩa quan trọng, việc thay đổi vị trí của một từ cũng có thể làm thay đổi ý nghĩa của cả câu. Vector positional embedding(PE) được tính theo công thức:

$$PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (3.32)$$

$$PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (3.33)$$

Trong đó  $d_{model}$  là số chiều của vector, pos là số vị trí (0, 1, 2, 3...),  $i$  là chiều thứ  $i$  của vector ( $i \in 0, 1, 2 \dots d_{model}$ )

### 3.4.2 Cấu trúc của Encoder



Hình 3.5: Encoder.

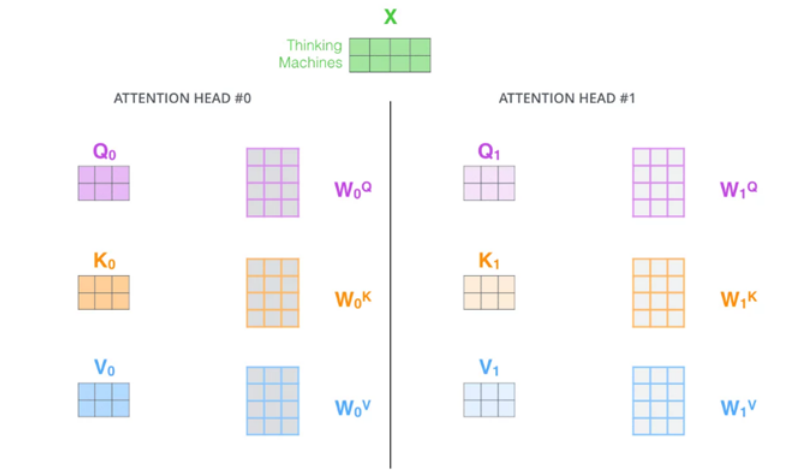
Encoder trong Transformers chịu trách nhiệm nhận diện và mã hóa các đặc trưng của đầu vào. Nó chuyển đổi các từ trong câu thành các vector biểu diễn ngữ nghĩa.

Encoder trong Transformers bao gồm hai thành phần chính: Multi-Head Attention và Feed-Forward Neural Network. Các thành phần này kết hợp với nhau để tạo ra các đại diện ngữ nghĩa của đầu vào.

#### Multi-Head Attention

Kiến trúc transformers được thiết kế với nhiều lớp self-attention kiến trúc giống hệt nhau nhưng trọng số của 3 ma trận Q, K, V (được tạo thành từ bộ 3 ma trận trọng số  $W_Q$ ,  $W_K$ ,  $W_V$ ) khác nhau. Việc tính toán của các lớp này được thực hiện song song. Các vector biểu diễn qua mỗi lớp self-attention sẽ được nối lại với nhau sau đó được nhân với một ma trận trọng số  $W_o$  để nén thông tin từ các vector (các vector này cùng biểu diễn cho 1 từ) thành một vector duy nhất. Vector này sau đó đi qua một bước gọi là Add & Normalize nữa trước khi đưa vào layer Feed Forward.

Ý nghĩa của cơ chế multi-head này là để tăng thêm phần chắc chắn trong việc quyết định thông tin nào cần khuếch đại, thông tin nào cần bỏ qua. Vì rằng các cái đầu sẽ cùng vote và đưa ra lựa chọn khách quan, đáng tin cậy hơn 1 cái đầu.



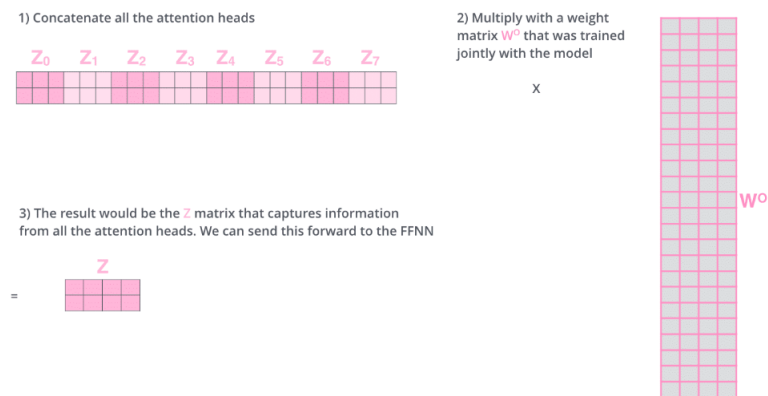
Hình 3.6: Đầu vào của lớp Multi-head Attention

Với multi-headed attention các ma trận trọng số  $W_Q$ ,  $W_K$ ,  $W_V$  là riêng biệt cho mỗi đầu. Như đã giới thiệu nhân  $X$  với ma trận  $W_Q$ ,  $W_K$ ,  $W_V$  để tạo ra các ma trận  $Q$ ,  $K$ ,  $V$ .

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V = Z$$

Hình 3.7: Công thức tính  $Z_i$

$Z_i$  sẽ được tính theo công thức như trên với  $d_k$  là số chiều của ma trận  $K$ .



Hình 3.8: Công thức tính  $Z$ .

$Z$  sẽ được tính bằng cách nối các  $Z_i$  lại với nhau và nhân với ma trận trọng số  $W_o$ .

### Feed-Forward Networks(FFN)

Các vector sau khi đi qua bước Add & Normalize(sẽ được nói ở mục sau) sẽ được gửi tới FFN. Lớp này bao gồm 2 tầng biến đổi thông tin và 1 hàm ReLU (các giá trị  $< 0$  được gán lại  $= 0$ ) ở giữa. Dropout cũng được áp dụng ở lần biến đổi thứ nhất sau khi các vector qua hàm ReLU. Sau khi qua layer FFN các vector cũng phải qua bước Add & Normalize trước khi đi vào khối encoder kế tiếp.

Ý nghĩa của layer FFN này là để học mối quan hệ tiềm ẩn giữa các vector độc lập mà chưa được mô tả rõ ràng. Khác với mối quan hệ giữa các từ được khuếch đại qua lớp self-attention, vẫn còn những mối quan hệ tiềm ẩn khác không thể diễn giải bằng công thức toán học sẽ được học thông qua lớp này.

### Add & Norm

Thêm (Add):

Residual Connection: Lớp này thực hiện một phép cộng giữa đầu ra của một lớp trước đó và đầu ra của lớp hiện tại. Đây là khái niệm residual connection (kết nối dư) hay còn gọi là skip connection (kết nối bỏ qua). Kết nối dư giúp mô hình học tốt hơn bằng cách cho phép thông tin gốc từ các lớp trước đó đi qua mà không bị thay đổi nhiều, từ đó cải thiện khả năng học và giảm độ lỗi khi huấn luyện sâu.

Chuẩn Hóa (Normalization):

Layer Normalization: Sau khi thực hiện phép cộng (thêm), lớp được chuẩn hóa bằng cách sử dụng Layer Normalization. Đây là một kỹ thuật chuẩn hóa giúp làm giảm biến động giữa các đầu ra của các lớp, đảm bảo rằng các đầu ra có trung bình là 0 và phương sai là 1. Layer Normalization giúp tăng cường sự ổn định và tốc độ hội tụ của quá trình huấn luyện.

Giả sử đầu vào của lớp là một vector  $x$  với các phần tử  $x_1, x_2, \dots, x_d$  công thức Layer Normalization là:

$$y_i = \gamma \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

Giải thích ký hiệu:

$x_i$ : Giá trị đầu vào của phần tử thứ  $i$  trong vector đầu vào.

$\mu$ : Trung bình của các giá trị đầu vào trong vector.

$\sigma$ : Độ lệch chuẩn của các giá trị đầu vào trong vector.

$\epsilon$ : Một hằng số nhỏ để tránh chia cho 0 (thường là  $10^{-5}$  hoặc  $10^{-6}$ ).

$\beta$ : Hệ số dịch chuyển (shifting factor), cũng được học trong quá trình huấn luyện.

---

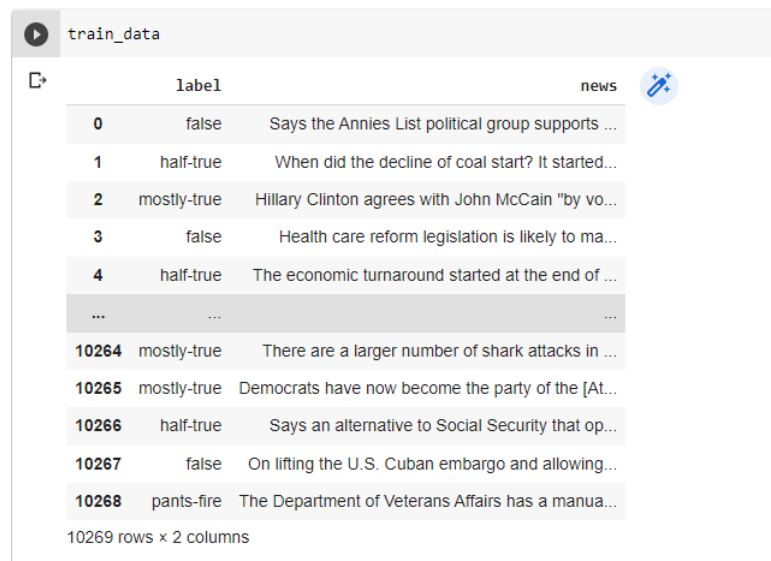
## Chương 4

# THỰC HIỆN ĐỀ TÀI

Ở chương này sẽ trình bày quá trình nhóm thực hiện đề tài. Các bước thực hiện như tại phần 2. Nhóm sử dụng ngôn ngữ Python để thực hiện bài toán và thực hiện trên nền tảng Google Collab.

### 4.1 Đọc dữ liệu

Ta đọc dữ liệu LIAR-MASTER vào trong bài làm. Lúc này dữ liệu sẽ có hơn 11000 bài báo với mức độ tin cậy khác nhau.



	label	news
0	false	Says the Annies List political group supports ...
1	half-true	When did the decline of coal start? It started...
2	mostly-true	Hillary Clinton agrees with John McCain "by vo...
3	false	Health care reform legislation is likely to ma...
4	half-true	The economic turnaround started at the end of ...
...	...	...
10264	mostly-true	There are a larger number of shark attacks in ...
10265	mostly-true	Democrats have now become the party of the [At...
10266	half-true	Says an alternative to Social Security that op...
10267	false	On lifting the U.S. Cuban embargo and allowing...
10268	pants-fire	The Department of Veterans Affairs has a manua...

10269 rows x 2 columns

Hình 4.1: Dữ liệu sau đọc vào chương trình



## 4.2 Tiền xử lý dữ liệu

Mô hình của chúng ta là mô hình phân loại nhị phân nên trước khi ta đưa vào mô hình ta phải xử lý nhãn sao cho đưa về dạng true - false. Cụ thể như sau:

- "true" → "true"
- "mostly-true" → "true"
- "half-true" → "true"
- "false" → "false"
- "barely-true" → "false"
- "pants-fire" → "false"

Sau khi thay đổi các nhãn ta sẽ được kết quả như sau:

```
test_data = Pre_process(test_data)
test_data
```

	label	news
0	true	Building a wall on the U.S.-Mexico border will...
1	false	Wisconsin is on pace to double the number of l...
2	false	Says John McCain has done nothing to help the ...
3	true	Suzanne Bonamici supports a plan that will cut...
4	false	When asked by a reporter whether hes at the ce...
...	...	...
1278	true	Says his budget provides the highest state fun...
1279	false	Ive been here almost every day.
1280	false	In the early 1980s, Sen. Edward Kennedy secret...
1281	false	Says an EPA permit languished under Strickland...
1282	false	Says the governor is going around the state ta...

Hình 4.2: Dữ liệu sau khi thay đổi các nhãn

Tiền xử lý dữ liệu là một bước quan trọng trong việc chuẩn bị dữ liệu cho các mô hình máy học. Đây là quy trình giúp làm sạch, chuẩn hóa, và biến đổi dữ liệu để cải thiện hiệu quả và độ chính xác của mô hình. Sau khi thay đổi nhãn, chúng em sử dụng một số phương pháp làm sạch văn bản như sau:

```
def preprocess_text(text):
    # Tokenize the text
    tokens = nltk.word_tokenize(text) # Tokenize the text here

    # Remove stop words
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word.lower() not in stop_words]

    # Remove rare words
    word_counts = Counter()
    for word in tokens: # Use tokens here
        word_counts.update([word])

    common_words = [word for word, count in word_counts.items() if count > 5]
    for word in common_words:
        text = text.replace(word, '')

    # Remove special characters
    tokens = [re.sub(r'^a-zA-Z0-9\s', '', token) for token in tokens]
    tokens = [token for token in tokens if token]

    # Stem and lemmatize
    stemmer = PorterStemmer()
    lemmatizer = WordNetLemmatizer()
    preprocessed_text = [lemmatizer.lemmatize(stemmer.stem(word)) for word in tokens]

    return ' '.join(preprocessed_text)
```

Hình 4.3: Các bước làm sạch văn bản

```
# Convert text to word count vectors with CountVectorizer
# create the transform
cvec = CountVectorizer()

# tokenize, build vocab and encode training data
traindata_cvec = cvec.fit_transform(traindata['new'].values)

# summarize
print(cvec.vocabulary_)
# print(cvec.get_feature_names())
```

{'we': 9447, 'have': 4160, 'less': 5141, 'americans': 710, 'working': 9605, 'now': 5977, 'than': 8725, 'in': 4483, 'the': 8739, '70s': 323, 'however': 4354, 'hartzler': 4146, 'was': 9419, 'talking': 8619, 'about': 413, 'entire': 3203, 'decade': 2501, 'of': 6051, 'first': 3619, 'eight': 3073, 'years': 9645, '1970': 104, 'through': 8802, '1977': 110, 'lower': 5297, 'employment': 3143, 'population': 6653, 'ratio': 7087, 'and': 733, 'labor': 4985, 'force': 3704, 'participation': 6338, 'rate': 7079, '2015': 153, 'when': 9510, 'obama': 6006, 'sworn': 8584, 'into': 4684, 'office': 6063, 'he': 4176, 'did': 2715, 'not': 5961, 'use': 9224, 'holy': 4301, 'bible': 1214, 'but': 1499, 'instead': 4631, 'kuran': 4975, 'their': 8745, 'equivalency': 3235, 'to': 8836, 'our': 6175, 'very': 9291, 'different': 2724, 'beliefs': 1167, 'ellison': 3108, 'used': 9225, 'koran': 4960, 'that': 8730, 'once': 6098, 'belonged': 1175, 'thomas': 8780, 'jefferson': 4801, 'borrowing': 1344, 'rare': 7076, 'book': 1325, 'from': 3799, 'library': 5166, 'congress': 2104, 'it': 4761, 'goes': 3957, 'without': 9576, 'saying': 7703, 'is': 4736, 'update': 9196, 'barack': 1085, 'resigned': 7395, 'trinity': 8972, 'united': 9146, 'church': 1800, 'christ': 1788, 'on': 6096, 'may': 5462, '31': 205, '2008': 144, 'after': 579, 'pastor': 6366, 'jeremiah': 4807, 'wright': 9623, 'jr': 4846, 'made': 5323, 'controversial': 2222, 'remarks': 7296, 'for': 3714, 'policy': 6622, 'other': 6170, 'matters': 5454, 'says': 7704, 'having': 4165, 'organizations': 6}

Hình 4.4: Bộ từ điển và số lần xuất hiện của từ đó trong dataset

	tfidf
70s	0.328225
decade	0.242670
lower	0.191335
70s years	0.164112
70s hartzler	0.164112
lower labor	0.164112
talking entire	0.164112
hartzler	0.164112
ratio lower	0.164112
1977 lower	0.164112
hartzler talking	0.164112
population ratio	0.164112

Hình 4.5: Chỉ số TF-IDF của các từ/cụm từ trong từ điển

Original: abortion dwayne-bohac State representative Texas republican a mailer Says the Annies List political gr  
 Tokenized BERT: ['abortion', 'd', '##way', '##ne', '-', 'bo', '##ha', '##c', 'state', 'representative', 'texas',  
 Token IDs BERT: [11324, 1040, 4576, 2638, 1011, 8945, 3270, 2278, 2110, 4387, 3146, 3951, 1037, 5653, 2121, 2758,  
 Tokenized RoBERT: ['abortion', 'Ġd', 'Ġwayne', '-', 'b', 'oh', 'ac', 'ĠState', 'Ġrepresentative', 'ĠTexas', 'Ġrepu  
 Token IDs RoBERTa: [27275, 385, 20143, 12, 428, 2678, 1043, 331, 4915, 1184, 37958, 10, 7107, 254, 15674, 5, 3921

Hình 4.6: Minh họa cách hoạt động tokenizer BERT và RoBERTa trên một câu mẫu

## 4.3 Huấn luyện mô hình

### 4.3.1 SVM và Logistic Regression

Trong quá trình thí nghiệm Logistic Regression và SVM để nhận diện tin giả, chúng tôi tiến hành kiểm tra mô hình với hai phương pháp phổ biến là CountVectorizer và TF-IDF (Term Frequency-Inverse Document Frequency) được sử dụng để chuyển đổi văn bản thành các vector đặc trưng, phục vụ cho việc huấn luyện và kiểm thử mô hình. Đối với mô hình SVM, Gọi Pipeline từ thư viện sklearn: *from sklearn.pipeline import Pipeline*. Và cũng từ thư viện này, em chạy kiểm thử trên một số mô hình khác như Decision Tree, Naive Bayes,...

### 4.3.2 LSTM và Transformers

Đối với LSTM và Transformers, em sử dụng các phương pháp sau để xử lý dữ liệu trước khi đưa vào huấn luyện mô hình.

Tạo từ điển ánh xạ mỗi từ với 1 số nguyên bằng Tokenizer từ thư viện tensorflow.keras.preprocessing.text.

```
[8]: # Tạo TextVectorization để chuyển đổi văn bản thành các chỉ số
tokenizer = Tokenizer(num_words=20000)# Từ điển có 20000 từ
maxlen = 256# Độ dài tối đa của chuỗi

tokenizer.fit_on_texts(train_texts)# Train Tokenizer

[19]: print(tokenizer.word_index)

{'the': 1, 'in': 2, 'of': 3, 'to': 4, 'a': 5, 'and': 6, 'says': 7, 'for': 8, 'that': 9, 'is': 10, 'on': 11, 'has': 12, 'have': 13, 'percent': 14, 'are': 15, 'than': 16, 'was': 17, 'more': 18, 'we': 19, 'by': 20, 'state': 21, 'from': 22, 'it': 23, 'with': 24, '000': 25, 'as': 26, 'obama': 27, 'tax': 28, 'not': 29, 'our': 30, 'year': 31, 'health': 32, 'years': 33, 'president': 34, 'people': 35, 'he': 36, 'states': 37, 'at': 38, 'would': 39, 'million': 40, 'an': 41, 'be': 42, 'care': 43, 'his': 44, 'one': 45, 'i': 46, 'you': 47, 'their': 48, 'were': 49, 'this': 50, 'jobs': 51, 'ove': 52, 'will': 53, 'new': 54, 'u': 55, 's': 56, 'been': 57, 'they': 58, 'or': 59, '1': 60, 'who': 61, 'about': 62, 'when': 63, 'billion': 64, 'al': 65, 'bill': 66, 'texas': 67, 'only': 68, 'out': 69, 'every': 70, 'said': 71, 'taxes': 72, 'under': 73, 'there': 74, 'federal': 75, 'no': 76, 'voted': 77, 'barack': 78, 'since': 79, 'up': 80, 'government': 81, 'budget': 82, 'had': 83, 'if': 84, 'now': 85, 'country': 86, 'any': 87, 'law': 88, 'last': 89, 'united': 90, 'pay': 91, 'its': 92, 'plan': 93, 'because': 94, 'time': 95, 'wisconsin': 96, '10': 97, 'rate': 98, 'other': 99, 'mo': 100, 'clinton': 101, 'first': 102, 'cut': 103, 'republican': 104, '2': 105, 'but': 106, 'into': 107, 'get': 108, 'americans': 109, 'america': 110, 'against': 111, 'even': 112, 'just': 113, 'public': 114, 'senate': 115, 'most': 116, 'spending': 117, 'florida': 118, 'school': 119, 'tw': 120, 'obamacare': 121, 'office': 122, 'insurance': 123, 'average': 124, 'scott': 125, 'debt': 126, 'hillary': 127, 'house': 128, 'can': 129, 'do': 130, 'security': 131, 'three': 132, 'income': 133, 'medicare': 134, 'city': 135, 'governor': 136, 'america': 137, 'gov': 138, 'national': 139, 'congress': 140, '5': 141, 'down': 142, 'times': 143, 'women': 144, 'four': 145, 'never': 146, '3': 147, 'world': 148, 'right': 149, 'nearly': 150, 'unemployment': 151, 'job': 152, 'per': 153, 'cost': 154, 'dollars': 155, 'could': 156, 'illegal': 157, 'dont': 158, 'what': 159, 'so': 160, 'while': 161, 'day': 162, 'today': 163, 'did': 164, 'trump': 165, 'bush': 166, 'going': 167, 'vote': 168, '100': 169, 'less': 170, 'half': 171, 'c': 172, 'campaign': 173, 'my': 174, 'after': 175, 'increase': 176, 'administration': 177, 'them': 178, 'your': 179, 'took': 180, 'education': 181, 'oil': 182, 'highest': 183, 'social': 184, 'romney': 185, '50': 186, 'history': 187, '20': 188, 'which': 189, 'democrats': 190, 'john': 191, 'some': 192, 'like': 193, 'cuts': 194, 'five': 195, '4': 196, 'before': 197, 'georgia': 198, 'rick': 199, 'those': 200, 'donald': 201, 'during': 2
```

Hình 4.7: Từ điển sau khi được tạo.

Mã hóa văn bản thành số nguyên theo từ điển đã tạo ở trên bằng `texts_to_sequences` từ `Tokenizer`.

```
[9]: # Chuyển text thành chuỗi số nguyên theo từ điển
train_sequences = tokenizer.texts_to_sequences(train_texts)
test_sequences = tokenizer.texts_to_sequences(test_texts)
val_sequences = tokenizer.texts_to_sequences(val_texts)

[17]: print(train_sequences)

[[7, 1, 6968, 1141, 520, 621, 385, 444, 5119, 585, 11, 1601], [63, 164, 1, 2091, 3, 964, 866, 23, 602, 63, 1142, 315, 180, 241, 9, 602, 4, 1959, 2, 34, 310, 560, 1365, 177], [127, 101, 3546, 24, 191, 254, 20, 329, 4, 343, 310, 166, 1, 1093, 3, 1, 3547, 11, 416], [32, 43, 266, 298, 10, 666, 4, 667, 404, 467, 417, 4148], [1, 325, 4149, 602, 38, 1, 408, 3, 174, 505], [1, 1827, 3548, 13, 83, 18, 1602, 5120, 2, 1, 89, 97, 33, 16, 1, 532, 216, 3, 5121, 2740, 2265, 1203, 202, 1, 89, 120, 935], [725, 6969, 12, 29, 2266, 2, 1, 372, 36, 2092, 8, 33, 85], [1310, 1, 68, 481, 11, 50, 1603, 61, 12, 689, 3088, 113, 89, 31, 1828, 1311, 24, 1011, 965, 192, 3, 1, 3089, 1143, 266, 79, 6970], [5122, 23, 180, 936, 141, 40, 2, 395, 2093, 474, 8, 1, 2481, 3, 3549, 4, 3550, 743, 1, 54, 5123, 3551, 1436, 1094, 1437], [7, 668, 632, 2741, 2267, 5124, 6, 622, 6971, 1366, 5, 2742, 168, 9, 154, 6972, 40, 2, 246, 1960, 257], [8, 1, 102, 95, 2, 187, 1, 1095, 3, 1, 139, 901, 168, 1829, 10, 1367, 16, 1, 2094, 168, 1829], [79, 779, 150, 271, 40, 109, 13, 6973, 69, 3, 1, 304, 335, 6, 107, 451], [63, 224, 105, 17, 136, 3, 445, 19, 348, 113, 4150, 1, 90, 3, 353, 3, 30, 81, 19, 220, 103, 2, 3], [1, 217, 6974, 824, 64, 595, 4, 1, 81, 1604], [116, 3, 1, 506, 43, 223, 12, 356, 2, 192, 2268, 57, 3552, 59, 3553, 3554], [2, 50, 89, 209, 2, 1252, 2269, 14, 3, 1, 110, 35, 2095, 29, 4, 168, 409, 14, 3, 507, 35, 6, 902, 14, 3, 521, 133, 275, 2095, 29, 4, 168], [254, 522, 5, 1961, 9, 1, 8, 1, 410, 110, 231, 5125, 6, 36, 71, 65, 410, 110, 1830, 49, 2096, 6975, 726], [55, 56, 262, 937, 966, 468, 3090, 6, 44, 1831, 190, 287, 11, 5, 117, 3555, 6, 85, 48, 561, 1368, 10, 2743, 69], [455, 289, 2, 6976, 5126, 49, 332, 80, 4, 6977, 14, 63, 5, 6978, 3, 1, 148, 825, 586, 5, 2744, 2745], [218, 169, 25, 35, 446, 3091, 4151, 89, 31], [144, 6, 491, 654, 15, 379, 170, 63, 47, 6979, 8, 1144, 16, 63, 191, 3092, 17, 102, 475, 136], [1, 90, 37, 12, 1, 183, 623, 28, 98, 2, 1, 404, 148], [19, 113, 83, 1, 562, 31, 8, 1, 843, 500, 2, 137, 2, 187], [7, 125, 258, 3093, 867, 80, 4, 1832, 25, 294, 6, 232, 241, 32, 43], [7, 224, 185, 237, 4, 108, 1605, 3, 411, 476], [46, 158, 290, 61, 6980, 6981, 10], [1720, 868, 111, 110, 938, 6, 5127, 13, 2097, 175, 2746, 6, 1145, 4152], [199, 447, 12, 146, 228, 41, 279, 6, 1833, 1, 68, 481, 4, 13, 643, 1, 67, 3094, 132, 143, 2, 3095, 669], [574, 2482, 3556, 38, 97, 1204, 5, 2483, 1834, 1205, 38, 97, 903, 5, 2483, 2, 5128, 6982], [1721, 151, 2, 1509, 1096, 10, 62, 214, 4, 903, 14], [7, 425, 575, 10, 276, 3557, 165], [84, 47, 603, 38, 37, 9, 15, 149, 4, 210, 58, 5129, 130, 29, 13, 82, 1206, 6, 58, 13, 316, 670, 280, 6983], [15
```

Hình 4.8: Văn bản sau khi mã hóa số nguyên.

Em thực hiện padding để các văn bản có cùng độ dài.

```
[10]: # Padding de text co cung do dai
x_train = tf.keras.preprocessing.sequence.pad_sequences(train_sequences, maxlen=maxlen)
x_test = tf.keras.preprocessing.sequence.pad_sequences(test_sequences, maxlen=maxlen)
x_val = tf.keras.preprocessing.sequence.pad_sequences(val_sequences, maxlen=maxlen)

[20]: print(x_train)

[[ 0  0  0 ... 585 11 1601]
 [ 0  0  0 ... 560 1365 177]
 [ 0  0  0 ... 3547 11 416]
 ...
 [ 0  0  0 ... 73 184 131]
 [ 0  0  0 ... 2415 4 1676]
 [ 0  0  0 ... 4 1876 1758]]
```

Hình 4.9: Văn bản sau khi padding.

Sau đó, em tiến hành huấn luyện mô hình trên 2 model là LSTM và Transformers. Kết quả huấn luyện sẽ được trình bày ở phần sau.

## 4.4 Kết quả

### 4.4.1 SVM

Sau khi chạy mô hình với tập test của LIAR-master. Ta có được các chỉ số như sau.

	precision	recall	f1-score	support
false	0.59	0.43	0.50	556
true	0.64	0.77	0.70	727
accuracy			0.62	1283
macro avg	0.61	0.60	0.60	1283
weighted avg	0.62	0.62	0.61	1283

Hình 4.10: Kết quả mô hình SVM với tập test LIAR (1)  
Sử dụng phương pháp CountVectorizer làm vector đặc trưng

Kết quả cho thấy các chỉ số về precision, recall và f1-score. Trong đó Precision (Độ chính xác) là một chỉ số dùng để đánh giá hiệu suất của các mô hình phân loại trong học máy. Precision được định nghĩa là tỉ lệ số điểm true positive trong số những điểm được phân loại là positive (TP + FP). Recall (hay còn gọi là Sensitivity hoặc True Positive Rate) là một chỉ số đánh giá tỉ lệ số điểm true positive trong số những điểm thực sự là positive (TP + FN) [4].

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

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

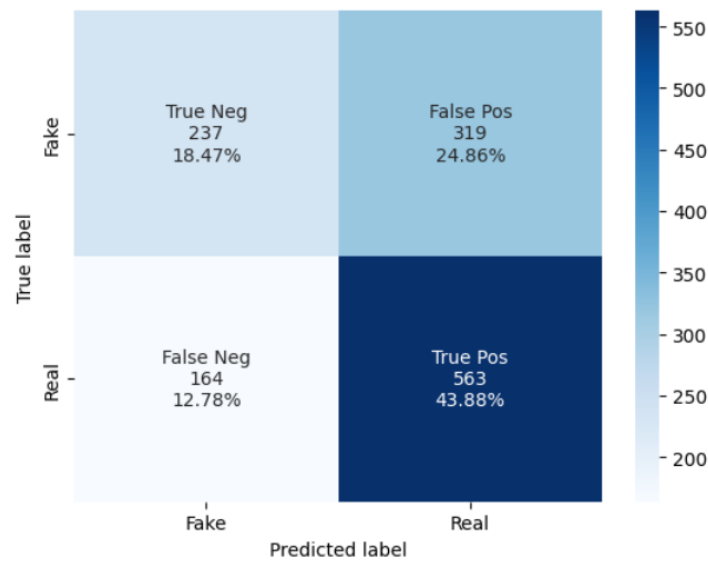
Trong đó, TP là lớp dự đoán đúng và nó thực sự đúng, FP là lớp dự đoán đúng nhưng thực chất là sai, FN là lớp dự đoán sai nhưng thực chất lại đúng.

	precision	recall	f1-score	support
false	0.63	0.38	0.48	556
true	0.64	0.83	0.72	727
accuracy			0.64	1283
macro avg	0.63	0.61	0.60	1283
weighted avg	0.63	0.64	0.61	1283

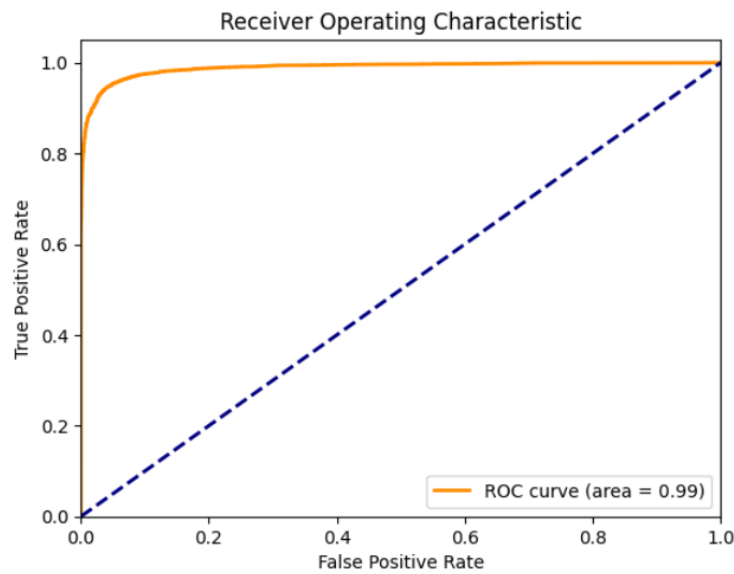
Hình 4.11: Kết quả mô hình SVM với tập test LIAR (2)  
Sử dụng phương pháp TF-IDF làm vector đặc trưng

F-score (hay còn gọi là F1-score) là một thước đo tổng hợp trong học máy và thống kê, được sử dụng để đánh giá hiệu quả của mô hình phân loại, đặc biệt là trong các tình huống dữ liệu mất cân bằng. F-score là sự kết hợp của precision (độ chính xác) và recall (độ nhạy), giúp cung cấp một cái nhìn tổng thể về hiệu suất của mô hình phân loại.

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



Hình 4.12: Confusion matrix SVM



Hình 4.13: Đường cong ROC cho mô hình SVM

#### 4.4.2 Logistic Regression

Cũng giống như phương pháp SVM, kết quả của Logistic Regression cũng được thí nghiệm trên hai tập test và sử dụng hai phương pháp: CountVectorizer và TF-IDF làm vectơ đặc trưng

	precision	recall	f1-score	support
false	0.56	0.51	0.53	556
true	0.65	0.69	0.67	727
accuracy			0.61	1283
macro avg	0.60	0.60	0.60	1283
weighted avg	0.61	0.61	0.61	1283

Hình 4.14: Kết quả mô hình Logistic Regression với tập test LIAR (1)

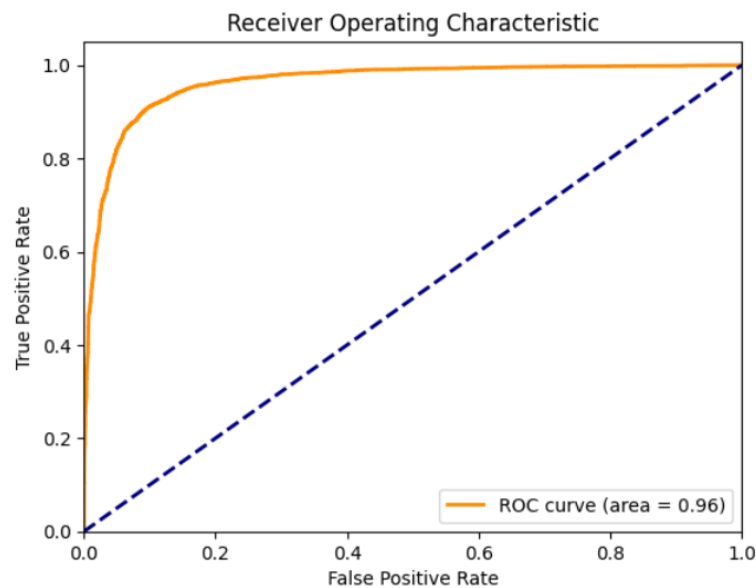
Sử dụng phương pháp CountVectorizer làm vector đặc trưng

	precision	recall	f1-score	support
false	0.59	0.41	0.48	556
true	0.63	0.78	0.70	727
accuracy			0.62	1283
macro avg	0.61	0.59	0.59	1283
weighted avg	0.61	0.62	0.61	1283

Hình 4.15: Kết quả mô hình Logistic Regression với tập test LIAR (2)  
Sử dụng phương pháp TF-IDF làm vector đặc trưng

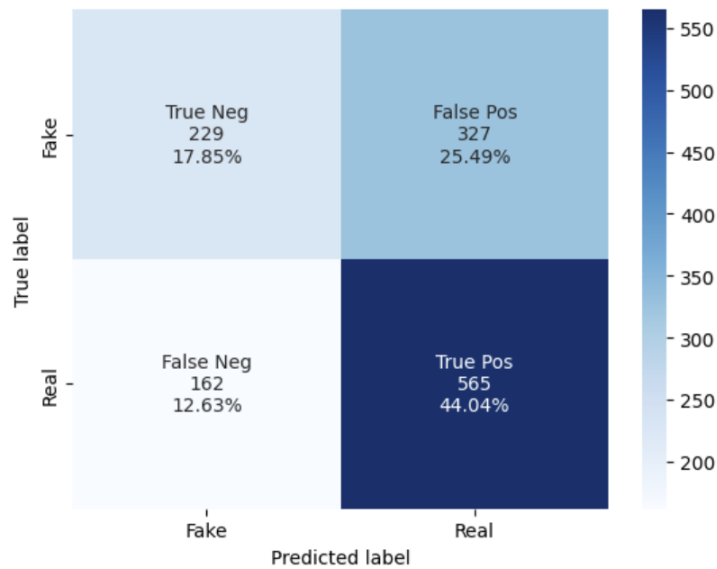
Có thể thấy kết quả thu được của mô hình Logistic Regression cho ra các chỉ số khá tương đồng với nhau. Vì vậy việc sử dụng Count Vectorizer và TF-IDF làm vecto đặc trưng đối với thuật toán Logistic Regression cho ra kết quả gần giống nhau.

ROC (Receiver Operating Characteristic) là một đường cong biểu diễn mối quan hệ giữa tỷ lệ True Positive (TPR) và tỷ lệ False Positive (FPR) ở các ngưỡng phân loại khác nhau của một mô hình phân loại nhị phân. Dưới đây là biểu đồ biểu hiện đường cong ROC của mô hình.



Hình 4.16: Đường cong ROC mô hình Logistic Regression





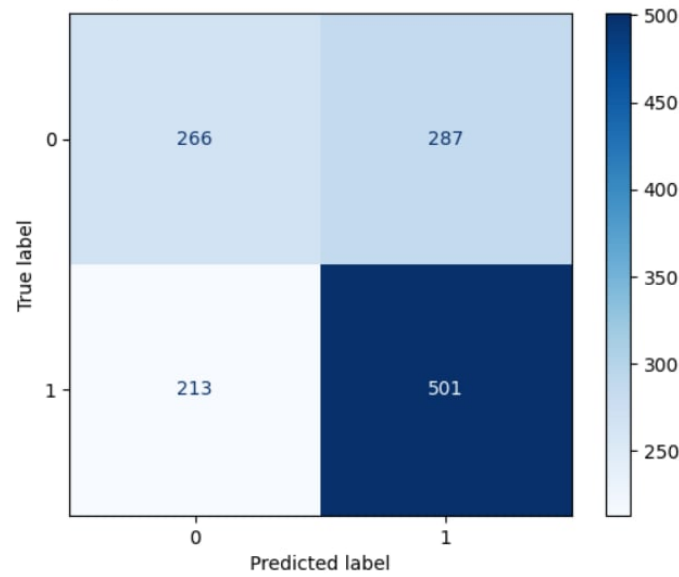
Hình 4.17: Confusion matrix for Logistic Regression

#### 4.4.3 LSTM

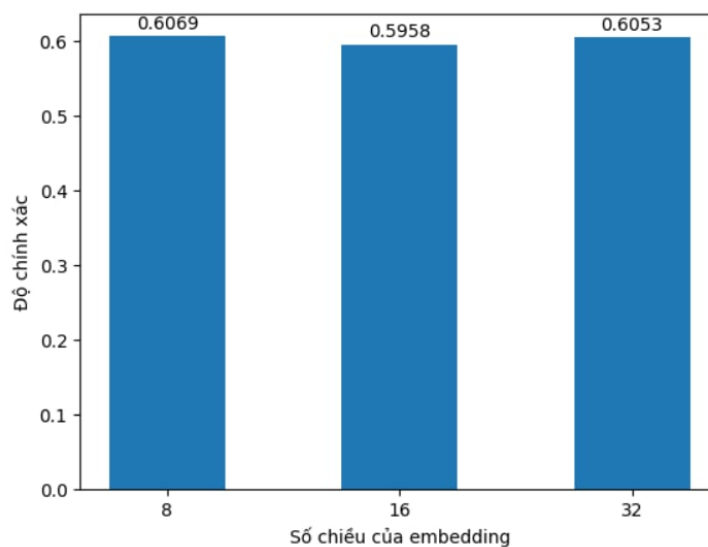
Sau khi chạy mô hình với tập test của LIAR-master. Ta có được các chỉ số như sau:

Classification Report:					
	precision	recall	f1-score	support	
False	0.56	0.48	0.52	553	
True	0.64	0.70	0.67	714	
accuracy			0.61	1267	
macro avg	0.60	0.59	0.59	1267	
weighted avg	0.60	0.61	0.60	1267	

Hình 4.18: Kết quả sau khi huấn luyện mô hình LSTM với số chiều của embedding là 8



Hình 4.19: Confusion matrix for LSTM



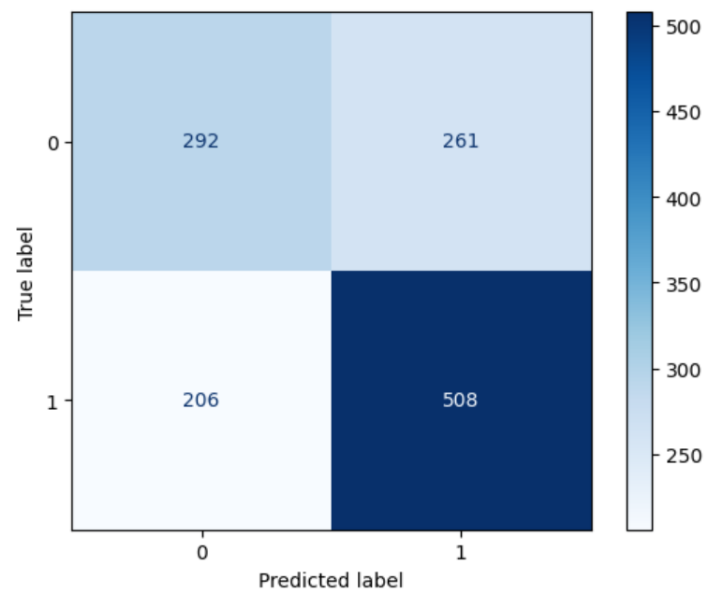
Hình 4.20: Biểu đồ so sánh độ chính xác với những số chiều khác nhau của lớp embedding

#### 4.4.4 Transformers

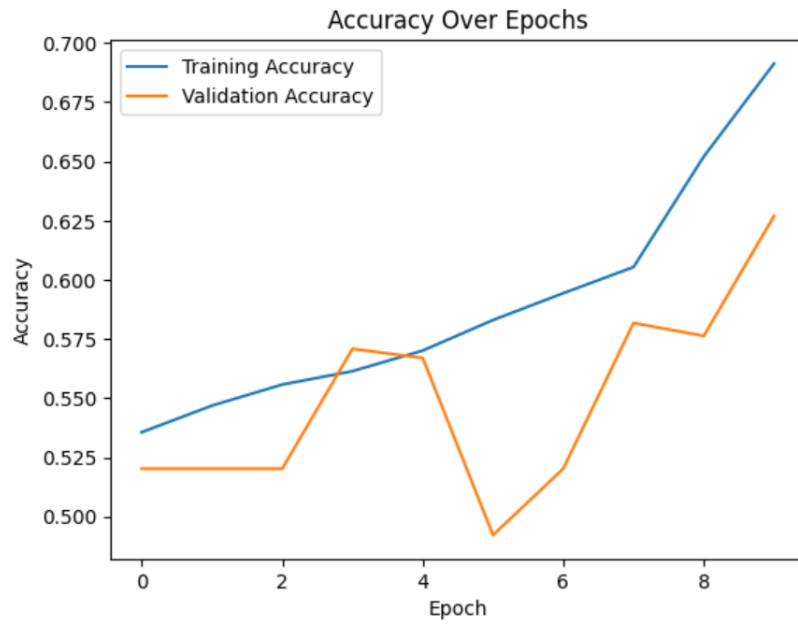
Sau khi huấn luyện mô hình qua 10 epoch em đã tính precision, recall, f1-score và vẽ biểu đồ về độ chính xác(Accuracy) và hàm mất mát(Loss), ma trận nhầm lẫn. Kết quả được thể hiện trong bảng và chi tiết qua các biểu đồ trực quan bên dưới:

Classification Report:				
	precision	recall	f1-score	support
false	0.59	0.53	0.56	553
true	0.66	0.71	0.69	714
accuracy			0.63	1267
macro avg	0.62	0.62	0.62	1267
weighted avg	0.63	0.63	0.63	1267

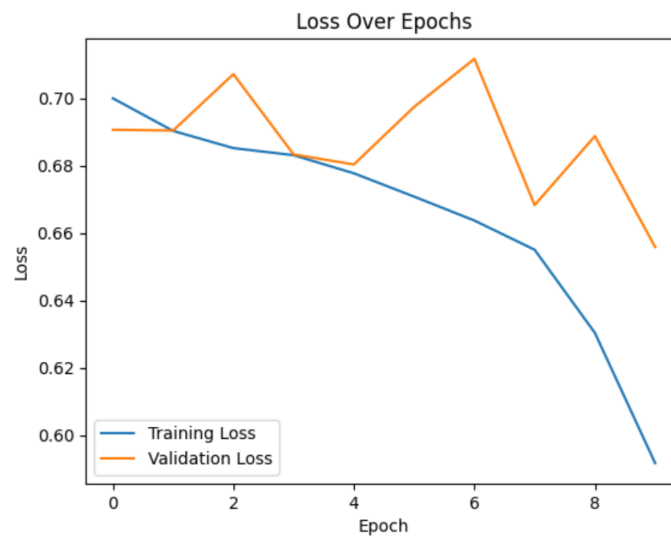
Hình 4.21: Kết quả khi huấn luyện mô hình với 8 head ở lớp Multi-head Attention.



Hình 4.22: Confusion matrix for Transformers.



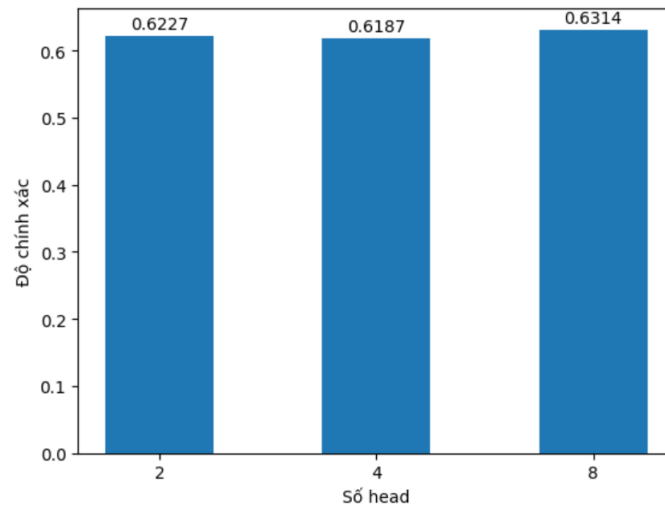
Hình 4.23: Biểu đồ độ chính xác.



Hình 4.24: Biểu đồ mất mát.

Qua các biểu đồ, em thấy hàm mất mát trên tập train và validation đều có xu hướng giảm dần sau 10 epoch, độ chính xác trên tập train và validation đều có xu hướng tăng dần sau 10 epoch. Tuy nhiên, trên tập validation, hàm mất mát và độ chính xác thiếu ổn định hơn.

Bên dưới là biểu đồ so sánh độ chính xác của mô hình với số head ở lớp multi-head attention khác nhau.



Hình 4.25: Biểu đồ độ chính xác với số head khác nhau ở lớp Multi-head Attention.

---

## Chương 5

# KẾT LUẬN

### 5.1 Nhận xét về kết quả

#### 5.1.1 Về hướng xây dựng trên các mô hình Machine Learning truyền thống

Ưu điểm:

- Hiệu quả trên các tập dữ liệu nhỏ đến trung bình: Các mô hình ML truyền thống thường hoạt động tốt trên các tập dữ liệu không quá lớn và có tính toán nhanh hơn so với các mô hình học sâu.
- Dễ dàng triển khai: Đa số các phương pháp này dễ cài đặt và không đòi hỏi nhiều tài nguyên tính toán.
- *Logistic Regression có thể cho kết quả tốt khi kết hợp với các kỹ thuật xử lý văn bản như TF-IDF hoặc các bộ biểu diễn từ (word embeddings).*
- *SVM*: Tránh được overfitting do sử dụng hàm hiệu chỉnh L2, hoạt động tốt trên cả dữ liệu tuyến tính và phi tuyến, đặc biệt là với kernel tricks, có thể hiệu quả trong việc phân loại văn bản, bao gồm cả phân loại tin giả. Nó hoạt động tốt với các tập dữ liệu có nhiều đặc trưng, trong khi lượng data points hạn chế.

Nhược điểm:

- Khả năng mở rộng kém: Khi dữ liệu trở nên rất lớn hoặc phức tạp, các mô hình này có thể gặp khó khăn trong việc mở rộng và xử lý.
- Hiệu suất hạn chế: Độ chính xác có thể bị giới hạn khi so sánh với các mô hình học sâu trên các bài toán phức tạp như phân loại văn bản.
- Khả năng mô hình hóa các mối quan hệ phức tạp trong dữ liệu có thể bị hạn chế. Cần tinh chỉnh các siêu tham số (hyperparameters) cẩn thận. Có thể chậm khi làm việc với các tập dữ liệu lớn.

Vì vậy, đối với quy mô bài toán còn hạn chế về dữ liệu huấn luyện và tài nguyên, Logistic Regression, SVM là những lựa chọn tốt.

### 5.1.2 Về hướng xây dựng trên các mô hình Deep Learning

#### Ưu điểm:

- **LSTM:**
  - Giải quyết vấn đề biến mất và bùng nổ gradient: Với cơ chế cổng (gates), LSTM khắc phục được vấn đề gradient biến mất và bùng nổ thường gặp trong các mô hình RNN truyền thống khi xử lý chuỗi dữ liệu dài.
  - Xử lý thông tin tuần tự dài hạn: LSTM có khả năng giữ thông tin trong thời gian dài, giúp mô hình nhớ được các mối quan hệ từ xa trong chuỗi dữ liệu. Điều này đặc biệt quan trọng trong các ứng dụng như xử lý ngôn ngữ tự nhiên (NLP) và phân tích chuỗi thời gian.
- **Transformers:**
  - Hiệu suất cao trong xử lý dữ liệu tuần tự: Transformers không yêu cầu xử lý tuần tự như LSTM, nhờ cơ chế Attention, giúp mô hình có thể xử lý toàn bộ chuỗi dữ liệu cùng lúc. Điều này làm tăng tốc độ xử lý và hiệu quả hơn so với các mô hình RNN truyền thống.
  - Khả năng song song hóa: Do không bị giới hạn bởi tính tuần tự, Transformers có thể dễ dàng được song song hóa, giúp tăng hiệu quả tính toán và giảm thời gian huấn luyện.
  - Mô hình hóa sự phụ thuộc dài hạn tốt hơn: Cơ chế Attention cho phép Transformers dễ dàng học được các mối quan hệ xa trong chuỗi dữ liệu mà không bị giới hạn bởi độ dài của chuỗi.

#### Nhược điểm:

- **LSTM:**
  - Tính toán phức tạp: So với các mô hình truyền thống như RNN, LSTM yêu cầu nhiều phép toán hơn và do đó tốn nhiều tài nguyên tính toán hơn, dẫn đến thời gian huấn luyện lâu hơn.
  - Khó khăn trong việc mô hình hóa sự phụ thuộc dài hạn: Mặc dù LSTM được thiết kế để xử lý các mối quan hệ dài hạn, nhưng nếu chuỗi dữ liệu quá dài, khả năng lưu giữ thông tin vẫn có thể bị giới hạn.
  - Khả năng mở rộng cao: khi tăng số head ở lớp multi-head attention hay dùng nhiều lớp encoder chồng lên nhau có thể giúp tăng hiệu quả của mô hình.
- **Transformers:**
  - Đòi hỏi tài nguyên lớn: Transformers thường yêu cầu nhiều tài nguyên tính toán và bộ nhớ hơn, đặc biệt là trong các ứng dụng với dữ liệu lớn và phức tạp như NLP.
  - Khó huấn luyện: Transformers cần có kỹ thuật tối ưu hóa cẩn thận để đạt được hiệu suất cao, và việc điều chỉnh siêu tham số có thể phức tạp.

Vì vậy nên chọn LSTM khi bạn cần xử lý chuỗi thời gian, âm thanh, hoặc dữ liệu tuần tự có tính dài hạn và yêu cầu mô hình duy trì thông tin qua nhiều bước thời gian và chọn Transformers khi bạn cần xử lý dữ liệu lớn, phức tạp, hoặc khi cần mô hình hóa các mối quan hệ xa trong dữ liệu với hiệu suất cao và khả năng song song hóa nên LSTM và Transformers là những lựa chọn hợp lý cho bài toán phân loại văn bản.

## 5.2 Tổng kết

Trong bài toán phát hiện tin giả, các mô hình machine learning truyền thống như Support Vector Machine (SVM) và Logistic Regression thường được sử dụng nhờ vào tính đơn giản, dễ hiểu và hiệu quả trên các tập dữ liệu nhỏ hoặc có cấu trúc rõ ràng. Tuy nhiên, khi dữ liệu ngày

càng phức tạp và kích thước dữ liệu lớn hơn, đặc biệt là trong các bài toán xử lý ngôn ngữ tự nhiên, các mô hình truyền thống này thường gặp khó khăn trong việc nắm bắt được mối quan hệ phi tuyến tính và phụ thuộc dài hạn trong dữ liệu.

Các mô hình học sâu như LSTM (Long Short-Term Memory) và Transformers đã chứng minh khả năng vượt trội trong việc xử lý các bài toán liên quan đến chuỗi thời gian và ngôn ngữ tự nhiên.

- **SVM và Logistic Regression:** Các mô hình này hoạt động tốt với các đặc trưng đã được trích xuất sẵn từ dữ liệu văn bản, chẳng hạn như bag-of-words hay TF-IDF. Tuy nhiên, chúng thường không hiệu quả trong việc xử lý các tương tác phức tạp giữa các từ trong một câu hoặc đoạn văn. Đối với bài toán phát hiện tin giả, việc dựa vào các đặc trưng bề mặt có thể dẫn đến độ chính xác không cao, do không thể nắm bắt đầy đủ ngữ cảnh và ý nghĩa sâu xa của văn bản.
- **LSTM:** LSTM là một loại mạng nơ-ron hồi tiếp được thiết kế để xử lý dữ liệu chuỗi và có khả năng ghi nhớ thông tin trong khoảng thời gian dài. Trong bài toán phát hiện tin giả, LSTM có thể mô hình hóa mối quan hệ ngữ nghĩa giữa các từ trong một chuỗi văn bản, giúp cải thiện hiệu quả so với các mô hình truyền thống. Tuy nhiên, LSTM có thể gặp khó khăn với dữ liệu rất dài do hạn chế về khả năng ghi nhớ lâu dài.
- **Transformers:** Transformers là một mô hình học sâu mạnh mẽ hơn, có khả năng xử lý song song và vượt trội trong việc nắm bắt các mối quan hệ phụ thuộc xa trong văn bản. Với cơ chế Attention, Transformers có thể dễ dàng tìm ra các phần quan trọng của văn bản cần chú ý, từ đó đưa ra quyết định chính xác hơn trong bài toán phát hiện tin giả. Transformers thường đạt được độ chính xác cao hơn so với LSTM và các mô hình truyền thống, đặc biệt là khi áp dụng trên các tập dữ liệu lớn.

Tóm lại, mặc dù các mô hình truyền thống như SVM và Logistic Regression có ưu điểm về tính đơn giản và dễ hiểu, nhưng trong các bài toán phức tạp như phát hiện tin giả, các mô hình học sâu như LSTM và Transformers tỏ ra vượt trội hơn nhờ khả năng nắm bắt ngữ cảnh và mối quan hệ phức tạp trong dữ liệu.

Trong khuôn khổ bài báo cáo cuối khoá, với thời gian và nhân lực còn hạn chế, không thể tránh khỏi những sai sót và sơ sài trong cách trình bày, mô tả các khía cạnh của bài toán.



---

## Tài liệu tham khảo

- [1] BERWICK, R. An idiot's guide to support vector machines (svms). *Retrieved on October 21* (2003), 2011.
- [2] BISHOP, C. M. Pattern recognition and machine learning. *Springer google schola 2* (2006), 1122–1128.
- [3] HÙNG, V. T., CHI, N. K., AND KIẾT, T. A. Phát hiện tự động tin giả: Thành tựu và thách thức. *Tạp chí Khoa học và Công nghệ-Đại học Đà Nẵng* (2022), 71–78.
- [4] YACOUBY, R., AND AXMAN, D. Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. In *Proceedings of the first workshop on evaluation and comparison of NLP systems* (2020), pp. 79–91.