

ĐẠI HỌC HUẾ
KHOA KỸ THUẬT VÀ CÔNG NGHỆ



BÁO CÁO ĐỒ ÁN
NĂM HỌC 2023-2024

Học phần: Xử lý ngôn ngữ tự nhiên

Giảng viên hướng dẫn: Đoàn Thị Hồng Phước

Lớp: Khoa Học Dữ Liệu và Trí Tuệ Nhân Tạo

Đề tài: Phân tích cảm xúc văn bản

Số phách

(Do hội đồng chấm thi ghi)

Thừa Thiên Huế, ngày ...tháng...năm 2024

ĐẠI HỌC HUẾ
KHOA KỸ THUẬT VÀ CÔNG NGHỆ



BÁO CÁO ĐỒ ÁN
NĂM HỌC 2023-2024

Giảng viên hướng dẫn: Đoàn Thị Hồng Phước

Lớp: Khoa Học Dữ Liệu và Trí Tuệ Nhân Tạo

Sinh viên thực hiện: Trần Văn Tuấn Phong

Mai Nhật Minh

Nguyễn Trịnh Tấn Đạt

(ký tên và ghi rõ họ tên)

Số phách

(Do hội đồng chấm thi ghi)

Thừa Thiên Huế, ngày ... tháng ... năm 2024

MỞ ĐẦU

Trong những năm gần đây, khi công nghệ phát triển như cơn lốc, cuộc sống hàng ngày của chúng ta đã trở nên đổi khác. Đặc biệt, sự xuất hiện của trí tuệ nhân tạo (AI) đã mang lại nhiều tiện ích thú vị, đặc biệt trong việc xử lý ngôn ngữ tự nhiên (NLP). Công nghệ này giúp máy tính hiểu và tương tác với ngôn ngữ của con người, từ đó làm cho cuộc sống trở nên dễ dàng hơn rất nhiều.

Trong thời đại mạng xã hội và kinh doanh online, việc hiểu rõ cảm xúc ẩn giấu đằng sau những dòng văn bản ngắn (như tin nhắn, bình luận trên các nền tảng mạng xã hội, bán hàng online) trở nên vô cùng quan trọng, giúp cho ta nắm được hành vi của người viết để ứng dụng vào những thuật toán. Vì vậy, công cụ có khả năng tự động phân loại cảm xúc trong văn bản đã trở thành một công cụ hữu ích và cần thiết.

Nhờ sự phát triển của AI và NLP, bây giờ việc tạo ra các hệ thống nhận diện cảm xúc không còn là điều quá khó khăn. Những hệ thống này có thể đọc và hiểu được cảm xúc của con người qua văn bản, từ đó giúp chúng ta phản hồi một cách phù hợp hơn. Điều này rất hữu ích trong nhiều lĩnh vực, từ trả lời phản hồi của khách hàng đến chăm sóc sức khỏe tâm lý.

Ngày nay, có rất nhiều công ty công nghệ hàng đầu đã phát triển các công cụ và hệ thống phân loại cảm xúc trong văn bản. Họ cũng chia sẻ rất nhiều tài liệu và công cụ để chúng ta có thể tự phát triển các hệ thống như vậy. Nhờ vậy, việc tạo ra một công cụ phân loại cảm xúc không còn là điều quá khó khăn. Chỉ cần một chút nỗ lực và học hỏi, ai cũng có thể tiếp cận và phát triển một hệ thống như vậy một cách dễ dàng.

LỜI CẢM ƠN

Trong khoảng thời gian làm đồ án cuối kỳ, nhóm em đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và sự dẫn dắt chỉ bảo nhiệt tình của giáo viên hướng dẫn.

Nhóm em xin gửi lời cảm ơn chân thành đến giáo viên hướng dẫn Đoàn Thị Hồng Phước - giảng viên Bộ môn Xử Lý Ngôn Ngữ Tự Nhiên - Trường Đại học Khoa học, người đã tận tình hướng dẫn nhóm.

Với điều kiện về thời gian cũng như lượng kiến thức về đề tài rất rộng mà kinh nghiệm còn hạn chế, đồ án này không thể tránh được những thiếu sót. Nhóm em rất mong nhận được sự chỉ bảo, đóng góp ý kiến để nhóm có điều kiện bổ sung, nâng cao ý thức, phục vụ tốt hơn cho công việc sau này.

Nhóm em xin chân thành cảm ơn!

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN VỀ XỬ LÝ NGÔN NGỮ TỰ NHIÊN	1
1. Giới thiệu	1
2. Khái niệm “Xử lý ngôn ngữ tự nhiên” (Natural Language Processing - NLP)	1
3. Tại sao Xử lý ngôn ngữ tự nhiên lại quan trọng ?.....	1
4. Ứng dụng NLP	2
5. Quy trình (Pipeline) của NLP:	3
5.1. Data Preprocessing:.....	4
5.1.1. Stemming và lemmatization:	4
5.1.2. Phân đoạn câu	4
5.1.3. Loại bỏ stop word	4
5.1.4. Tokenization	4
5.2. Trích xuất đặc trưng:	5
5.2.1. Bag of Words (BOW):	5
5.2.2. TF-IDF:.....	6
5.2.3. Word2Vec	8
5.2.4. GLoVe:.....	8
5.3. Modeling	8
6. Các kỹ thuật NLP phổ biến:	8
6.1. Các phương pháp học máy truyền thống:	9
6.2. Các phương pháp học sâu:	9
6.2.1. Convolutional Neural Network (CNN):	9
6.2.2. Recurrent Neural Network (RNN):	9
6.2.3. Transformers:.....	10
7. Các Framework Python cho NLP:	12
CHƯƠNG 2: XÂY DỰNG HỆ THỐNG PHÂN LOẠI CẢM XÚC VĂN BẢN BẰNG KỸ THUẬT NLP	14
1. Giới thiệu bài toán:	14
2. Mô tả tập dữ liệu:	14
3. Thực nghiệm:.....	16
3.1. Mô hình đề xuất:	16
3.1.1. Mô hình học máy truyền thống:	16
3.1.2. Mô hình học sâu (pretrained model):	16
3.2. Thực nghiệm:	17

3.2.1.	Mô hình học máy truyền thống:	17
3.2.2.	Mô hình pretrained RNN (LSTM/GRU):	22
3.2.3.	Mô hình pretrained BERT:	32
4.	Kết quả:	44
CHƯƠNG 3: KẾT LUẬN		45
1.	Những kết quả đạt được	45
2.	Những hạn chế.....	45
TÀI LIỆU THAM KHẢO		46

DANH MỤC HÌNH ẢNH

Hình 1: Phân tích cảm xúc bằng NLP (Nguồn:)	2
Hình 2: Mô tả quá trình Tokenization (Nguồn)	5
Hình 3: Bag-of-Words dựa trên tần suất xuất hiện từ (Nguồn)	5
Hình 4: Giải thích TF-IDF (Nguồn)	7
Hình 5: CNN cho bài toán text classification (Nguồn)	9
Hình 6: Kiến trúc KNN (Nguồn)	10
Hình 7: Kiến trúc Encoder – Decoder Transformer(Nguồn)	11
Hình 8: Bộ dữ liệu emotion-stimulus.csv	14
Hình 9: Bộ dữ liệu isear.csv	15
Hình 10: Mô tả train dataset	15
Hình 11: Mô tả test dataset	16
Hình 12: Import các thư viện cần dùng	17
Hình 13: Đọc dữ liệu	18
Hình 14: Visualize dataset	18
Hình 15: Preprocessing và tokenization sử dụng thư viện nltk	18
Hình 16: Tải bộ tokenizers ‘punkt’ của nltk	19
Hình 17: Thiết lập TF-IDF với các thông số cụ thể	19
Hình 18: Áp dụng vectorizer đã học để chuyển đổi tập train, test thành các vector TF-IDF	19
Hình 19: Confusion Matrix của mô hình Naïve Bayes	19
Hình 20: Confusion Matrix của mô hình Random Forest	20
Hình 21: Confusion Matrix của mô hình LR	20
Hình 22: Confusion Matrix của mô hình Linear SVM	21
Hình 23: Tạo pipeline để sử dụng đánh giá và lưu model lại	21
Hình 24: Kiến trúc RNN (Nguồn)	22
Hình 25: Gradient của RNN bị giảm dần theo thời gian learning (Nguồn)	22
Hình 26: Kiến trúc của LSTM và GRU (Nguồn)	23
Hình 27: Một review nào đó về hộp ngũ cốc	24
Hình 28: Mô tả Forget gate của LSTM (Nguồn)	24
Hình 29: Mô tả Input Gate của LSTM (Nguồn)	25
Hình 30: Mô tả quá trình tổng quát mà Forget Gate và Input Gate xử lý (Nguồn)	25
Hình 31: Mô tả Output Gate (Nguồn)	26
Hình 32: Mô tả gate của GRU (Nguồn)	27

Hình 33: Pretrained word vectors của wikipedia	27
Hình 34: Đoạn code tạo embedding matrix bằng cách ánh xạ lên bộ pretrained w2v ..	28
Hình 35: Kiểm tra kích thước của embedding matrix	28
Hình 36: Kiểm tra từ mới trong từ điển.....	28
Hình 37: Sử dụng pretrained w2v để tạo lớp embedding cho mô hình GRU	29
Hình 38: Xây dựng lớp GRU và lớp Dense	29
Hình 39: Compile model và model summary	29
Hình 40: Quá trình train model	30
Hình 41: Biểu đồ trực quan accuracy của train và validation qua 15 epochs	30
Hình 42: Biểu đồ trực quan loss của train và validation qua 15 epochs	31
Hình 43: Confusion Matrix của mô hình GRU	31
Hình 44: Thử nghiệm model GRU với message bất kì	32
Hình 45: BERT sử dụng Encoder (Nguồn)	33
Hình 46: Quá trình train mô hình cho task masked LM (Nguồn)	34
Hình 48: Mô tả cách lấy output (Nguồn).....	36
Hình 49: Đọc tập train và test.....	36
Hình 50: Mô tả dataframe.....	37
Hình 51: Biểu đồ phân bố số lượng từ trong các văn bản trong dataset	37
Hình 51: Map encoding emotion labels.....	37
Hình 52: Thêm cột label tương ứng	38
Hình 53: Tải tokenizer và pretrained model bert.....	38
Hình 54: Text tokenization	38
Hình 55: Đầu vào cho mô hình.....	40
Hình 56: Quá trình xây dựng neural cho bài toán	40
Hình 57: Tạo mô hình.....	41
Hình 58: Biên dịch mô hình	41
Hình 59: Model summary	41
Hình 60: Quá trình huấn luyện / fine tuning bert	42
Hình 61: Biến thiên của Loss trong quá trình training	42
Hình 62: Biến thiên của accuracy trong quá trình training	43
Hình 63: Confusion matrix	43
Hình 64: Kết quả dự đoán message	44

DANH MỤC TỪ VIẾT TẮT

STT	Từ viết tắt	Thuật ngữ đầy đủ
1	AI	Artificial Intelligence
2	AWS	Amazon Web Service
3	BERT	Bidirectional Encoder Representation from Transformer
4	CBOW	Continuous Bag of Words
5	CNN	Convolution Neural Network
6	DL	Deep Learning
7	DNN	Deep Neural Network
8	FAIR	Facebook AI Research
9	IDF	Inverse Document Frequency
10	GLUE	General Language Understanding Evaluation
11	GPT	Generative Pre-training Transformer
12	GRU	Gated Recurrent Unit
13	LSTM	Long short-term memory
14	LM	Language Model
15	LR	Logistic Regression
16	ML	Machine Learning
17	NLG	Natural Language Generation
18	NLP	Natural Language Processing
19	NLTK	Natural Language Toolkit
20	NLU	Natural Language Understanding
21	POS	Part of Speech
22	RNN	Recurrent Neural Network
23	SVM	Support Vector Machine
24	TF	Term Frequency

CHƯƠNG 1: TỔNG QUAN VỀ XỬ LÝ NGÔN NGỮ TỰ NHIÊN

1. Giới thiệu

Xử lý ngôn ngữ tự nhiên (NLP) là một trong những lĩnh vực nóng nhất của trí tuệ nhân tạo (AI) nhờ vào các ứng dụng như trình tạo văn bản có thể viết các bài luận mạch lạc, chatbot có thể khiến con người nghĩ rằng chúng có tri giác, và các chương trình chuyển văn bản thành hình ảnh tạo ra những hình ảnh chân thực từ bất cứ điều gì bạn mô tả. Những năm gần đây đã mang lại một cuộc cách mạng trong khả năng của máy tính để hiểu các ngôn ngữ của con người, ngôn ngữ lập trình, và thậm chí là các chuỗi sinh học và hóa học, chẳng hạn như cấu trúc DNA và protein, có tính chất tương tự ngôn ngữ. Các mô hình AI mới nhất đang mở khóa những lĩnh vực này để phân tích ý nghĩa của văn bản đầu vào và tạo ra đầu ra có ý nghĩa và biểu cảm.[1]

2. Khái niệm “Xử lý ngôn ngữ tự nhiên” (Natural Language Processing - NLP)

Xử lý ngôn ngữ tự nhiên (NLP) là lĩnh vực xây dựng các máy móc có thể thao tác ngôn ngữ của con người - hoặc dữ liệu giống ngôn ngữ của con người - theo cách nó được viết, nói và tổ chức. Nó phát triển từ ngôn ngữ học tính toán, lĩnh vực sử dụng khoa học máy tính để hiểu các nguyên tắc của ngôn ngữ, nhưng thay vì phát triển các khung lý thuyết, NLP là một ngành kỹ thuật nhằm xây dựng công nghệ để thực hiện các nhiệm vụ hữu ích.

NLP có thể được chia thành hai phân ngành:

- Hiểu ngôn ngữ tự nhiên (NLU): tập trung vào phân tích ngữ nghĩa hoặc xác định ý nghĩa dự định của văn bản.
- Tạo ngôn ngữ tự nhiên (NLG): tập trung vào việc máy tạo ra văn bản. NLP tách biệt với - nhưng thường được sử dụng cùng với - nhận dạng giọng nói, lĩnh vực tìm cách phân tích ngôn ngữ nói thành từ ngữ, chuyển đổi âm thanh thành văn bản và ngược lại.

3. Tại sao Xử lý ngôn ngữ tự nhiên lại quan trọng ?

NLP rất quan trọng khi các công nghệ ngôn ngữ này được áp dụng vào các lĩnh vực đa dạng như bán lẻ (ví dụ, trong các chatbot dịch vụ khách hàng) và y tế (diễn giải hoặc tóm tắt hồ sơ sức khỏe điện tử). Các trợ lý hội thoại như Alexa của Amazon và Siri của Apple sử dụng NLP để lắng nghe các câu hỏi của người dùng và tìm câu trả lời. Các trợ lý phức tạp nhất — chẳng hạn như GPT-3.5, Gemini, Claude, hay mới đây nhất là

GPT-4o, ra mắt vào 13 tháng 5 [3], đã được mở cho các ứng dụng thương mại — có thể tạo ra văn bản tinh vi về nhiều chủ đề khác nhau cũng như cung cấp sức mạnh cho các chatbot có khả năng giữ cuộc trò chuyện mạch lạc. Google sử dụng NLP để cải thiện kết quả tìm kiếm của mình, và các mạng xã hội như Facebook sử dụng nó để phát hiện và lọc các nội dung thù địch.

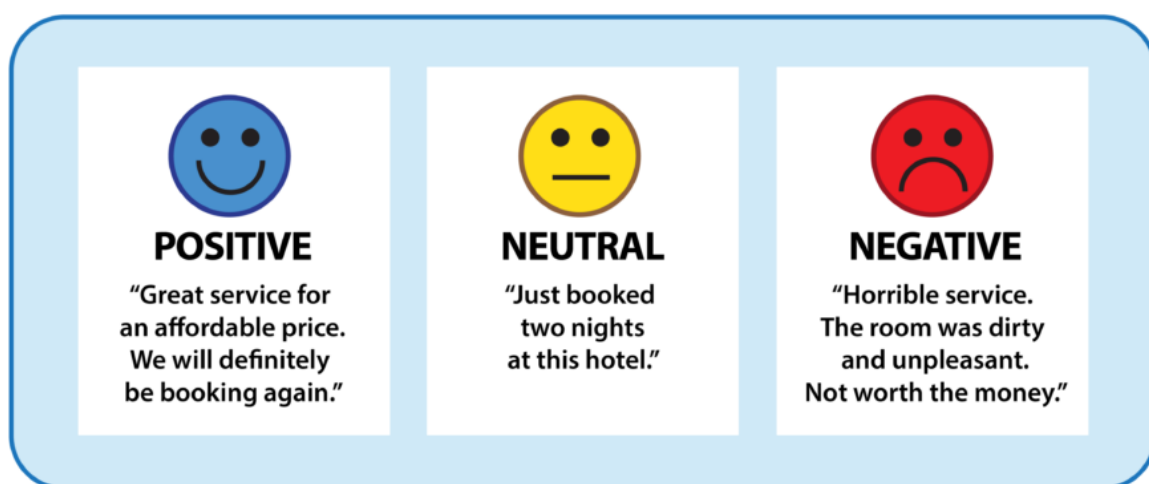
NLP ngày càng trở nên tinh vi, tuy nhiên vẫn còn nhiều việc phải làm. Các hệ thống hiện tại vẫn dễ mắc phải thiên vị và sự không nhất quán, và đôi khi hành xử một cách thất thường. Dù có những thách thức, các kỹ sư học máy có nhiều cơ hội để áp dụng NLP theo những cách ngày càng quan trọng đối với một xã hội đang vận hành.

4. Ứng dụng NLP

NLP được sử dụng cho nhiều nhiệm vụ liên quan đến ngôn ngữ, bao gồm trả lời câu hỏi, phân loại văn bản theo nhiều cách khác nhau và trò chuyện với người dùng.

Một số ứng dụng phổ biến của NLP:

Phân tích cảm xúc: là quá trình phân loại ý định cảm xúc của văn bản. Thông thường, đầu vào cho mô hình phân loại cảm xúc là một đoạn văn bản, và đầu ra là xác suất rằng cảm xúc được biểu đạt là tích cực, tiêu cực, hoặc trung tính. Thường thì xác suất này dựa trên các đặc trưng được tạo thủ công, n-grams từ ngữ, các đặc trưng TF-IDF, hoặc sử dụng các mô hình học sâu để nắm bắt các phụ thuộc dài hạn và ngắn hạn. Phân tích cảm xúc được sử dụng để phân loại đánh giá của khách hàng trên các nền tảng trực tuyến cũng như cho các ứng dụng chuyên biệt như nhận diện dấu hiệu của bệnh tâm lý¹ trong các bình luận trực tuyến.



Hình 1: Phân tích cảm xúc bằng NLP (Nguồn:²)

¹ [NLP Model Spots Signs of Mental Illness in Facebook Posts \(deeplearning.ai\)](https://www.deeplearning.ai/news/nlp-model-spots-signs-of-mental-illness-in-facebook-posts/)

² https://miro.medium.com/v2/resize:fit:720/format:webp/1*j_ewcyZlv1D3S1XPQY2zqQ.jpeg

Phát hiện toxic (phân loại độc hại): là một nhánh của phân tích cảm xúc, trong đó mục tiêu không chỉ là phân loại ý định thù địch mà còn phân loại các loại cụ thể như đe dọa, lăng mạ, tục tĩu và thù ghét đối với các danh tính nhất định. Đầu vào cho mô hình này là văn bản, và đầu ra thường là xác suất của mỗi loại độc hại. Các mô hình phân loại độc hại có thể được sử dụng để điều tiết và cải thiện các cuộc trò chuyện trực tuyến bằng cách chặn các bình luận xúc phạm³, phát hiện ngôn từ kích động thù địch⁴, hoặc quét các văn bản để tìm sự phỉ báng.

Dịch máy tự động hóa việc dịch giữa các ngôn ngữ khác nhau. Đầu vào cho mô hình này là văn bản bằng ngôn ngữ nguồn được chỉ định, và đầu ra là văn bản bằng ngôn ngữ đích được chỉ định. Google Dịch⁵ có lẽ là ứng dụng phổ biến nhất. Các mô hình như vậy được sử dụng để cải thiện giao tiếp giữa mọi người trên các nền tảng mạng xã hội như Facebook hoặc Skype. Các phương pháp dịch máy hiệu quả có thể phân biệt giữa các từ có nghĩa tương tự. Một số hệ thống cũng thực hiện nhận dạng ngôn ngữ; tức là phân loại văn bản là thuộc ngôn ngữ này hay ngôn ngữ khác.

Phát hiện spam là một vấn đề phân loại nhị phân phổ biến trong NLP, với mục đích là phân loại email là spam hay không. Các bộ phát hiện spam nhận đầu vào là văn bản email cùng với các văn bản phụ như tiêu đề và tên người gửi. Chúng nhằm mục đích đưa ra xác suất rằng email là spam. Các nhà cung cấp email như Gmail sử dụng các mô hình này để cung cấp trải nghiệm người dùng tốt hơn bằng cách phát hiện email không mong muốn và di chuyển chúng vào thư mục spam.

Sửa lỗi ngữ pháp mã hóa các quy tắc ngữ pháp để sửa lỗi ngữ pháp trong văn bản. Đây chủ yếu được xem như một nhiệm vụ chuỗi-tới-chuỗi, trong đó mô hình được đào tạo với câu không ngữ pháp làm đầu vào và câu đúng làm đầu ra. Các trình kiểm tra ngữ pháp trực tuyến như Grammarly và các hệ thống xử lý văn bản như Microsoft Word⁶ sử dụng các hệ thống này để cung cấp trải nghiệm viết tốt hơn cho khách hàng của họ. Các trường học cũng sử dụng chúng để chấm điểm bài luận của học sinh.

5. Quy trình (Pipeline) của NLP:

Các mô hình NLP hoạt động bằng cách tìm kiếm mối quan hệ giữa các thành phần của ngôn ngữ — ví dụ, các chữ cái (letters), từ (words) và câu (sequences) trong một tập dữ liệu. Kiến trúc NLP sử dụng các phương pháp khác nhau để tiền xử lý dữ liệu, trích xuất đặc trưng và mô hình hóa.

³ Gamers Can Mute Offensive Language With AI (deeplearning.ai)

⁴ How Facebook Built a Hate Speech Detector (deeplearning.ai)

⁵ <https://ai.googleblog.com/2020/06/recent-advances-in-google-translate.html>

⁶ Microsoft Editor checks grammar and more in documents, mail, and the web - Microsoft Support

5.1. Data Preprocessing:

Trước khi một mô hình xử lý văn bản cho một nhiệm vụ cụ thể, văn bản thường cần được tiền xử lý để cải thiện hiệu suất mô hình hoặc để chuyển đổi các từ và ký tự thành định dạng mà mô hình có thể hiểu được.

5.1.1. Stemming và lemmatization:

Stemming là quá trình không chính thức chuyển đổi các từ thành dạng gốc của chúng bằng cách sử dụng các quy tắc heuristic.

Ví dụ: "university," "universities," và "university's" có thể đều được chuyển về gốc là "univers." (Một hạn chế của phương pháp này là "universe" cũng có thể được chuyển về "univers," mặc dù "universe" và "university" không có mối quan hệ ngữ nghĩa gần gũi.)

Lemmatization là cách tìm gốc từ chính thức hơn bằng cách phân tích hình thái của từ sử dụng từ vựng từ một từ điển. Stemming và lemmatization được cung cấp bởi các thư viện như Spacy và NLTK.

5.1.2. Phân đoạn câu

Quá trình này chia một đoạn văn bản lớn thành các đơn vị câu có ý nghĩa ngôn ngữ. Điều này rõ ràng trong các ngôn ngữ như tiếng Anh, nơi dấu chấm đánh dấu kết thúc câu.

Nhưng quá trình này cũng không hề dễ, một dấu chấm có thể được sử dụng để đánh dấu một chữ viết tắt cũng như để kết thúc một câu, và trong trường hợp này, dấu chấm nên là một phần của từ viết tắt. Quá trình này trở nên phức tạp hơn trong các ngôn ngữ như tiếng Trung cổ, không có dấu phân cách để đánh dấu kết thúc câu.

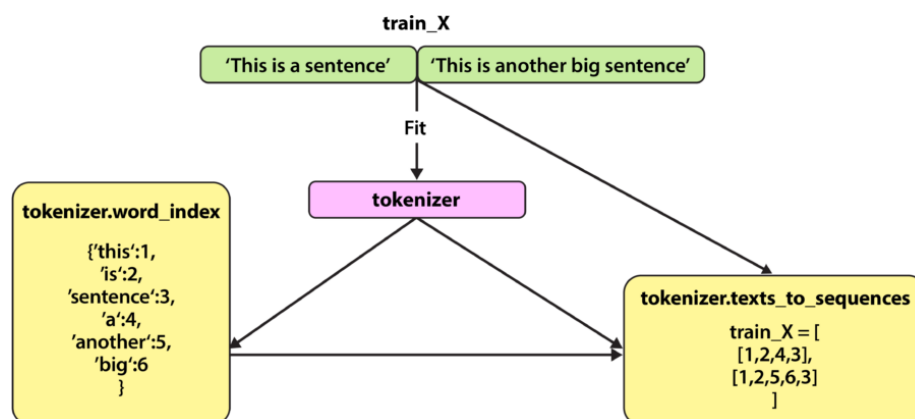
5.1.3. Loại bỏ stop word

Loại bỏ những từ xuất hiện thường xuyên nhất nhưng không thêm nhiều thông tin vào văn bản. Ví dụ, "the," "a," "an," và tương tự.

5.1.4. Tokenization

Quá trình này chia văn bản thành các từ và mảnh từ riêng lẻ. Kết quả thường bao gồm một chỉ số từ và văn bản đã được phân tách, trong đó các từ có thể được biểu diễn dưới dạng các token số để sử dụng trong các phương pháp học sâu khác nhau. Một phương

pháp hướng dẫn các mô hình ngôn ngữ bỏ qua các token không quan trọng⁷ có thể cải thiện hiệu quả.



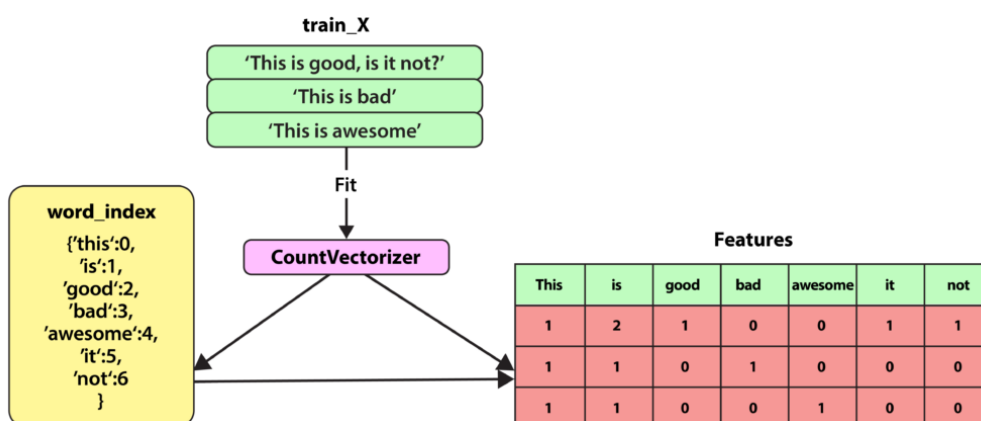
Hình 2: Mô tả quá trình Tokenization (Nguồn ⁸)

5.2. Trích xuất đặc trưng:

Hầu hết các kỹ thuật học máy truyền thống hoạt động dựa trên các đặc trưng – thường là các con số mô tả một văn bản liên quan đến tập dữ liệu chứa nó – được tạo ra bởi Bag-of-Words, TF-IDF, hoặc kỹ thuật tạo đặc trưng tổng quát như độ dài văn bản, word polarity, và metadata (ví dụ: nếu văn bản có các tags hoặc scores liên quan). Các kỹ thuật gần đây hơn bao gồm Word2Vec, GLoVE, và học các đặc trưng trong quá trình huấn luyện của mạng nơ-ron.

5.2.1. Bag of Words (BOW):

Bag-of-Words đếm số lần mỗi từ hoặc n-gram (kết hợp của n từ) xuất hiện trong một tài liệu. Ví dụ ảnh dưới đây, mô hình Bag-of-Words tạo ra một biểu diễn số của tập dữ liệu dựa trên tần suất xuất hiện mỗi từ trong word_index xuất hiện trong tài liệu.



Hình 3: Bag-of-Words dựa trên tần suất xuất hiện từ (Nguồn ⁹)

⁷ NLP Transformer Technique for More Efficient Token Usage (deeplearning.ai)

⁸ https://medium.com/@annie_/nlp-how-machine-understands-human-language-009ec4000a66

⁹ https://miro.medium.com/v2/resize:fit:720/format:webp/0*51VfuQWg--FqJRQP.png

Để minh họa thêm, đây là một ví dụ đơn giản về cách Bag-of-Words hoạt động:

- Tập dữ liệu: ["I love NLP", "NLP is fascinating", "I love learning NLP"]
- Chỉ mục từ (word_index): {"I": 0, "love": 1, "NLP": 2, "is": 3, "fascinating": 4, "learning": 5}
- Biểu diễn Bag-of-Words:
 - o "I love NLP": [1, 1, 1, 0, 0, 0]
 - o "NLP is fascinating": [0, 0, 1, 1, 1, 0]
 - o "I love learning NLP": [1, 1, 1, 0, 0, 1]

Trong ví dụ trên, mỗi tài liệu được biểu diễn dưới dạng một vector số, trong đó mỗi số đại diện cho số lần từ tương ứng xuất hiện trong tài liệu đó. Đây là cách đơn giản nhưng hiệu quả để mã hóa văn bản thành dạng mà các mô hình học máy có thể xử lý.

5.2.2. TF-IDF:

Trong Bag-of-Words, chúng ta đếm số lần xuất hiện của mỗi từ hoặc n-gram trong một tài liệu. Ngược lại, với TF-IDF, chúng ta gán trọng số cho mỗi từ dựa trên tầm quan trọng của nó. Để đánh giá tầm quan trọng của một từ, chúng ta xem xét hai yếu tố:

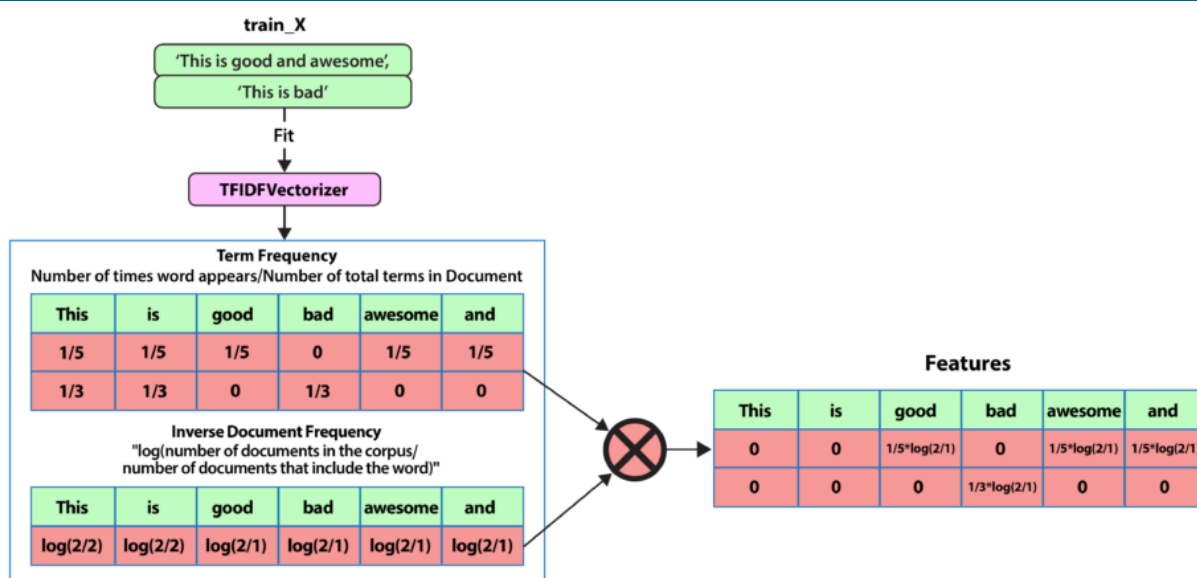
- Tần suất Từ (Term Frequency): Từ đó quan trọng như thế nào trong tài liệu?

$$TF(\text{word in a document}) = \frac{\text{Number of occurrences of that word in document}}{\text{Number of words in document}}$$

- Tần suất Nghịch Đảo Văn Bản (Inverse Document Frequency): Từ đó quan trọng như thế nào trong toàn bộ tập dữ liệu?

$$IDF(\text{word in a corpus}) = \log \left(\frac{\text{Number of documents in the corpus}}{\text{Number of documents that include the word}} \right)$$

Một từ được coi là quan trọng nếu nó xuất hiện nhiều lần trong một tài liệu. Nhưng điều này tạo ra một vấn đề: các từ như "a" và "the" xuất hiện rất thường xuyên, và vì vậy, điểm TF của chúng sẽ luôn cao. Chúng ta giải quyết vấn đề này bằng cách sử dụng Tần suất Nghịch Đảo Văn Bản (IDF), điểm này cao nếu từ đó hiếm và thấp nếu từ đó phổ biến trong toàn bộ tập dữ liệu. Điểm TF-IDF của một thuật ngữ là tích của TF và IDF.



Hình 4: Giải thích TF-IDF (Nguồn ¹⁰)

Ví dụ, ta có các câu văn sau:

Câu 1: "Tôi yêu học NLP" - Câu 2: "NLP là thú vị" - Câu 3: "Tôi yêu học về NLP"

Giả sử "NLP" xuất hiện trong 3 câu, "yêu" xuất hiện trong 2 câu, và "học" xuất hiện trong 2 câu. Chúng ta tính toán TF và IDF cho từ "NLP":

- TF(NLP trong Câu 1): $1/4 = 0.25$
- TF(NLP trong Câu 2): $1/3 = 0.33$
- TF(NLP trong Câu 3): $1/5 = 0.20$
- IDF(NLP): $\log(3/3) = \log(1) = 0$
- TF-IDF(NLP trong Câu 1) = $0.25 * 0 = 0$
- TF-IDF(NLP trong Câu 2) = $0.33 * 0 = 0$
- TF-IDF(NLP trong Câu 3) = $0.20 * 0 = 0$

Chúng ta nhận thấy rằng khi từ "NLP" xuất hiện trong tất cả các câu, giá trị IDF của nó bằng 0, cho thấy rằng từ này không có nhiều giá trị phân biệt trong tập dữ liệu này.

TF-IDF là một phương pháp mạnh mẽ để gán trọng số cho các từ dựa trên tầm quan trọng của chúng, giúp mô hình NLP xử lý văn bản hiệu quả hơn bằng cách tập trung vào những từ có ý nghĩa quan trọng hơn trong ngữ cảnh cụ thể.

¹⁰ <https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcTsFv-6Jh9XBeu2l14M5WloOQ2SjmJEpq-JcgHYX02oC02eJYvg>

5.2.3. Word2Vec

Word2Vec¹¹ sử dụng một mạng nơ-ron để thực hiện quá trình word embedding từ văn bản thô. Nó có hai biến thể:

- Skip-Gram: chúng ta cố gắng dự đoán các từ xung quanh dựa trên một từ mục tiêu.
- Continuous Bag-of-Words (CBOW): cố gắng dự đoán từ mục tiêu từ các từ xung quanh.

Sau khi loại bỏ lớp cuối sau quá trình huấn luyện, các mô hình này lấy một từ làm đầu vào và đưa ra một nhúng từ có thể được sử dụng làm đầu vào cho nhiều nhiệm vụ NLP khác nhau. Embedding từ Word2Vec bất kịp bối cảnh, nếu các từ cụ thể xuất hiện trong các ngữ cảnh tương tự, nhúng của chúng sẽ tương tự nhau.

5.2.4. GloVe:

Tương tự như Word2Vec vì cũng học các nhúng từ, nhưng làm điều đó bằng cách sử dụng các kỹ thuật phân tích ma trận thay vì học nơ-ron. Mô hình GloVe xây dựng một ma trận dựa trên số lần xuất hiện của các từ.

5.3. Modeling

Sau khi dữ liệu được tiền xử lý, nó được đưa vào một kiến trúc NLP mà mô hình hóa dữ liệu để hoàn thành một loạt các nhiệm vụ.

Các đặc trưng số được trích xuất bằng các kỹ thuật được mô tả ở trên có thể được đưa vào các mô hình khác nhau tùy thuộc vào nhiệm vụ cụ thể. Ví dụ, đối với phân loại, đầu ra từ vector hóa TF-IDF có thể được cung cấp cho Logistic Regression, Naive Bayes, Decision Tree, hoặc SVM.

Deep learning cũng được sử dụng để tạo ra các mô hình ngôn ngữ như vậy. Các mô hình học sâu nhận vào một nhúng từ làm đầu vào và, tại mỗi trạng thái thời gian, trả về phân phối xác suất của từ tiếp theo như xác suất cho mỗi từ trong từ điển. Các pretrained model học cấu trúc của một ngôn ngữ cụ thể bằng cách xử lý một tập văn bản lớn, chẳng hạn như Wikipedia. Sau đó, chúng có thể được fine tuning cho một nhiệm vụ cụ thể. Ví dụ, BERT đã được điều chỉnh tinh chỉnh cho các nhiệm vụ từ kiểm tra sự chính xác đến việc viết tiêu đề.

6. Các kỹ thuật NLP phổ biến:

¹¹ [How to Keep Up with a Fast-Changing Industry \(deeplearning.ai\)](https://deeplearning.ai)

Hầu hết các nhiệm vụ NLP được đề cập ở trên có thể được mô hình bằng một số kỹ thuật chung. Để hiểu rõ hơn, có thể chia các kỹ thuật này thành hai loại chính: Các phương pháp học máy truyền thống (traditional machine learning) và các phương pháp học sâu (deep learning).

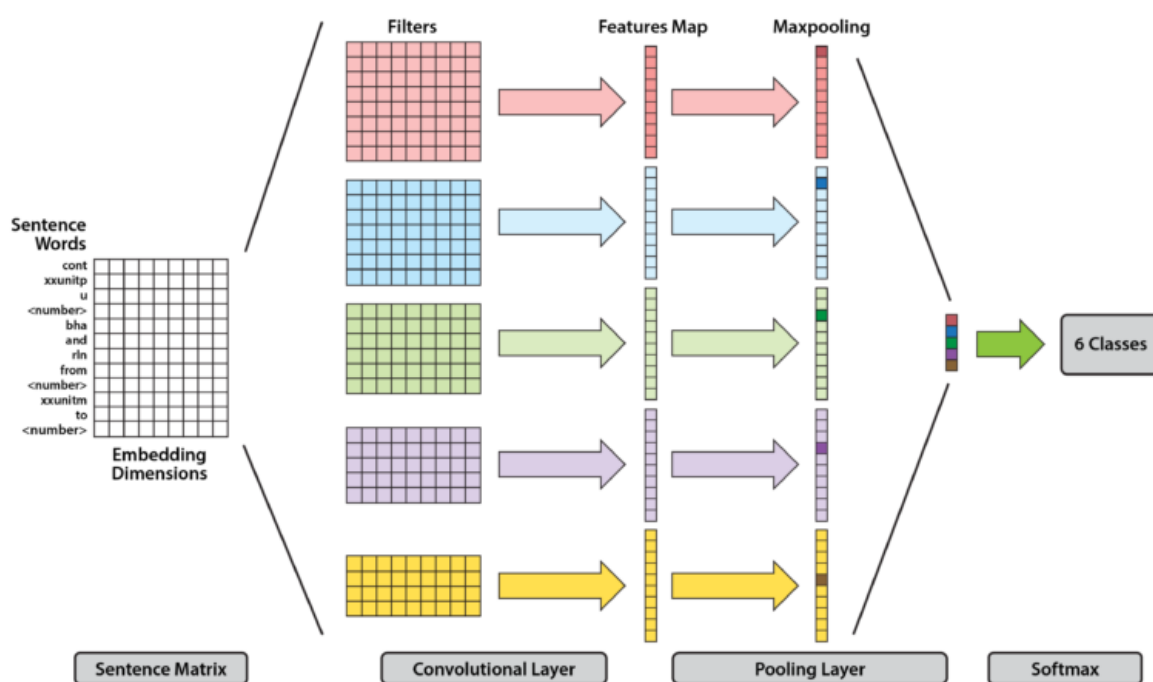
6.1. Các phương pháp học máy truyền thống:

Các phương pháp học máy truyền thống mà nhóm em đang quan tâm để phục vụ cho bài toán là Logistic Regression, Random Forest, Naïve Bayes và SVM.

6.2. Các phương pháp học sâu:

6.2.1. Convolutional Neural Network (CNN)¹²:

Ý tưởng sử dụng một CNN [4] để phân loại văn bản được trình bày lần đầu trong bài báo "Convolutional Neural Networks for Sentence Classification"[5] của Yoon Kim. Trực giác trung tâm là coi một tài liệu như một hình ảnh. Tuy nhiên, thay vì pixel, đầu vào là các câu hoặc tài liệu được biểu diễn dưới dạng ma trận các từ.



Hình 5: CNN cho bài toán text classification (Nguồn ¹³)

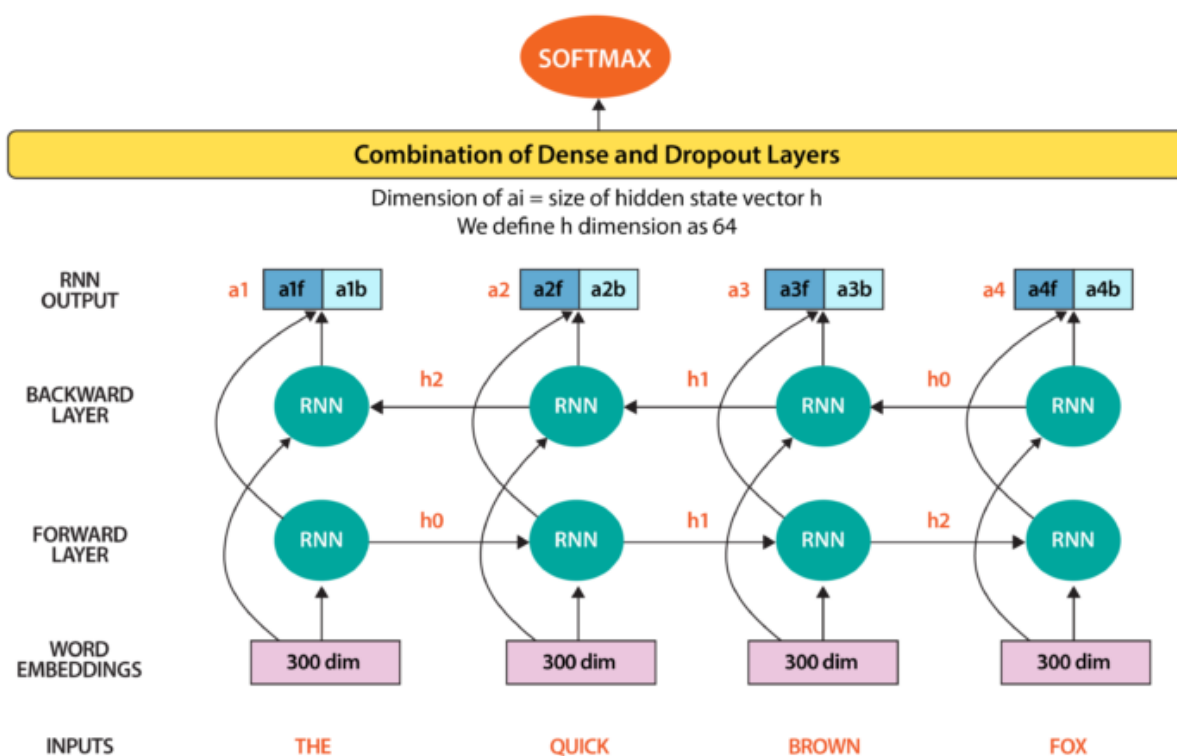
6.2.2. Recurrent Neural Network (RNN):

Nhiều kỹ thuật cho phân loại văn bản sử dụng học sâu xử lý các từ gần nhau sử dụng n-grams hoặc window (CNN). Chúng có thể nhìn thấy "New York" như một trường

¹² <https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/>

¹³ <https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRKsboIlcya0eEfM6oIhCnLFEIv8Xi33jIIMSghGwixhdg1LauF>

hợp duy nhất. Tuy nhiên, chúng không thể nắm bắt ngữ cảnh được cung cấp bởi một chuỗi văn bản cụ thể. Chúng không học cấu trúc tuần tự của dữ liệu, trong đó mỗi từ phụ thuộc vào từ trước đó hoặc từ trong câu trước đó. RNN [6] nhớ thông tin trước đó bằng cách sử dụng các trạng thái ẩn và kết nối nó với nhiệm vụ hiện tại. Các kiến trúc được biết đến như Gated Recurrent Unit (GRU) và long short-term memory (LSTM) là loại RNN được thiết kế để nhớ thông tin trong một khoảng thời gian mở rộng. Hơn nữa, LSTM/GRU hai chiều giữ thông tin ngữ cảnh ở cả hai hướng, điều này hữu ích trong phân loại văn bản. RNN cũng đã được sử dụng để tạo ra chứng minh toán học¹⁴ và dịch ý tưởng của con người¹⁵ thành từ ngữ.



Hình 6: Kiến trúc KNN (Nguồn ¹⁶)

6.2.3. Transformers¹⁷:

Một kiến trúc mô hình được mô tả lần đầu trong bài báo năm 2017 "Attention Is All You Need" (Vaswani, Shazeer, Parmar, et al.) [7], bỏ qua sự lặp lại và thay vào đó hoàn toàn phụ thuộc vào cơ chế chú ý tự thân để vẽ các phụ thuộc toàn cầu giữa đầu vào và đầu ra. Vì cơ chế này xử lý tất cả các từ cùng một lúc (thay vì một từ một lần) nên giảm tốc độ huấn luyện và chi phí suy luận so với RNNs, đặc biệt khi có thể song

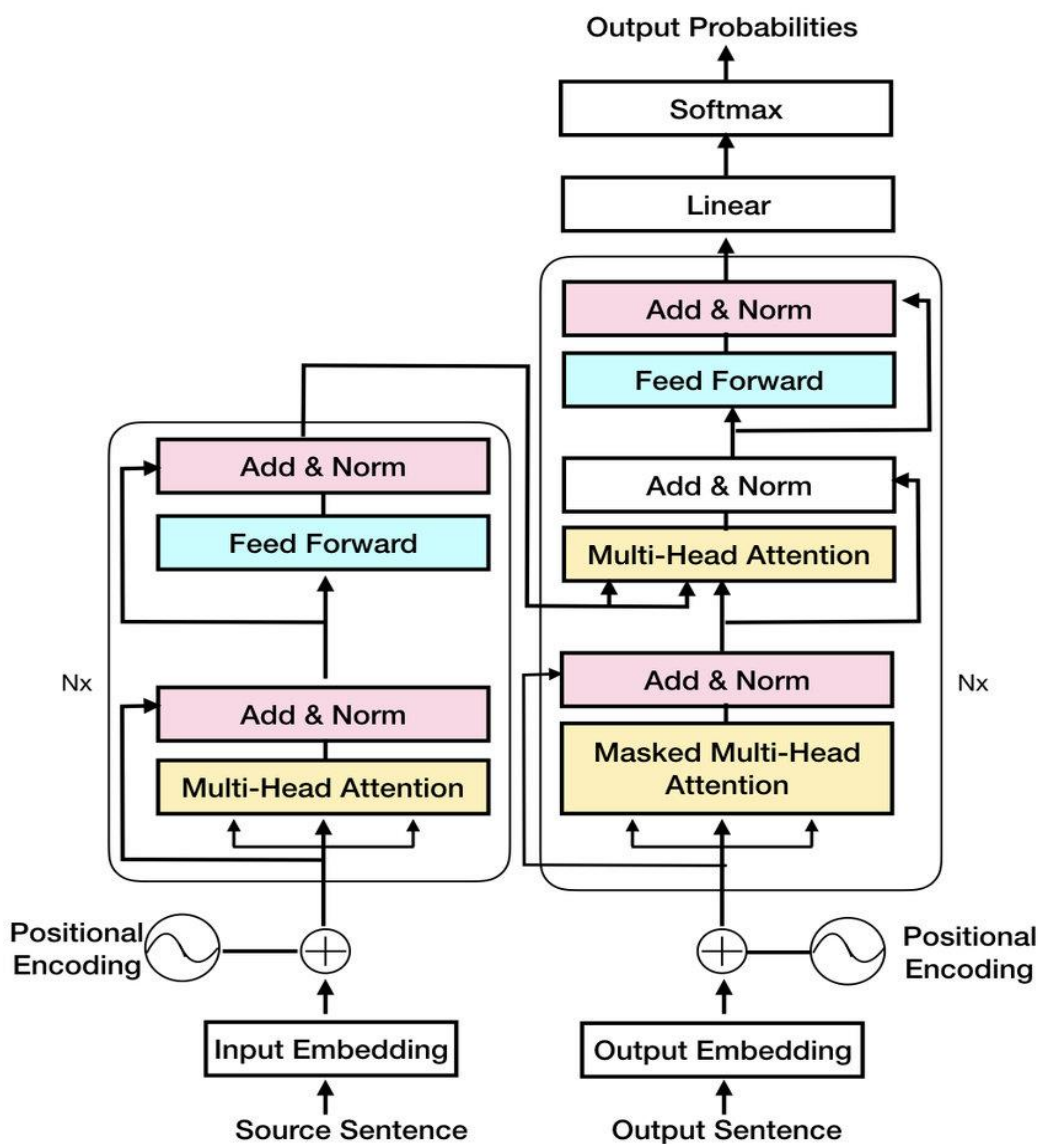
¹⁴ <https://www.deeplearning.ai/the-batch/the-proof-is-in-the-network/>

¹⁵ <https://www.deeplearning.ai/the-batch/listening-to-the-brain-2/>

¹⁶ <https://encrypted-tbn1.gstatic.com/images?q=tbn:AND9GcRvHQaMAObXk1nalhRhTKDNezjJFLxyiueK-mgKNWXH-z7rOIku>

¹⁷ <https://www.deeplearning.ai/the-batch/transformer-takeover/>

song hóa. Kiến trúc transformer đã cách mạng hóa NLP trong những năm gần đây, dẫn đến các mô hình bao gồm BLOOM, Jurassic-X, và Turing-NLG. Nó cũng đã được áp dụng thành công cho nhiều nhiệm vụ khác nhau trong thị giác, bao gồm tạo hình ảnh 3D.



Hình 7: Kiến trúc Encoder – Decoder Transformer (Nguồn ¹⁸)

Đã nhắc đến Transformers, nhóm em cũng xin giới thiệu tổng quát về BERT [8] – một mô hình ngôn ngữ được phát triển bởi Google vào năm 2018, sử dụng kiến trúc của Transformer. BERT được huấn luyện để hiểu ngữ cảnh của từ trong câu bằng cách sử dụng một phương pháp gọi là "masked language modeling" (mô hình hóa ngôn ngữ được ẩn đi). BERT có khả năng sinh ra biểu diễn từ vựng được biểu diễn dưới dạng vector có chiều sâu, giúp cải thiện hiểu biết văn bản và hiệu suất của nhiều nhiệm vụ

¹⁸ https://miro.medium.com/v2/resize:fit:2880/1*BHzGVskWGS_3jEcYYi6miQ.png

NLP. Đặc biệt, BERT có khả năng "biểu diễn chứng tỏ" (fine-tuning) để điều chỉnh mô hình cho các nhiệm vụ cụ thể, như phân loại văn bản và dịch máy. Ở chương 2 nhóm em sẽ nói kĩ hơn về cách sử dụng BERT trong bài toán cụ thể.

7. Các Framework Python cho NLP:

Natural Language Toolkit ¹⁹(NLTK) là một trong những thư viện NLP đầu tiên được viết bằng Python. Nó cung cấp các giao diện dễ sử dụng cho các tập văn bản và tài nguyên từ vựng như WordNet²⁰. Nó cũng cung cấp một bộ thư viện xử lý văn bản cho phân loại, gắn thẻ, gốc từ, phân tích cú pháp và luận lý ngữ nghĩa.

spaCy²¹ là một trong những thư viện NLP mã nguồn mở linh hoạt nhất. Nó hỗ trợ hơn 66 ngôn ngữ. spaCy cũng cung cấp các vector từ được huấn luyện trước và thực thi nhiều mô hình phổ biến như BERT. spaCy có thể được sử dụng để xây dựng các hệ thống sẵn sàng cho sản xuất cho việc nhận dạng thực thể đã đặt tên, gắn thẻ loại từ, phân tích phụ thuộc, phân đoạn câu, phân loại văn bản, gốc từ, phân tích hình thái học, liên kết thực thể, và nhiều hơn nữa.

ktrain: là một thư viện trong Python được phát triển để giúp việc huấn luyện và triển khai các mô hình học sâu trở nên dễ dàng và hiệu quả hơn. Thư viện này được xây dựng trên các framework mạnh mẽ như TensorFlow và Keras, và cung cấp các công cụ đơn giản hóa quy trình từ tiền xử lý dữ liệu, huấn luyện mô hình, đánh giá cho đến triển khai.

Thư viện Deep Learning: Các thư viện deep learning phổ biến bao gồm TensorFlow²² và PyTorch²³, giúp việc tạo ra các mô hình dễ dàng hơn với các tính năng như tự động phân biệt. Các thư viện này là các công cụ phổ biến nhất cho việc phát triển các mô hình NLP.

Hugging Face²⁴ cung cấp các triển khai và trọng số mã nguồn mở của hơn 135 mô hình hiện đại nhất. Kho lưu trữ cho phép tùy chỉnh và huấn luyện mô hình dễ dàng.

Streamlit²⁵ là một framework mã nguồn mở mạnh mẽ được thiết kế để giúp các nhà khoa học dữ liệu và lập trình viên xây dựng các ứng dụng web tương tác cho khoa học dữ liệu và machine learning một cách nhanh chóng và dễ dàng chỉ bằng Python. Với

¹⁹ <https://www.nltk.org/>

²⁰ <https://www.deeplearning.ai/the-batch/tiny-images-outsized-biases/>

²¹ <https://spacy.io/>

²² <https://www.tensorflow.org/>

²³ <https://pytorch.org/>

²⁴ <https://huggingface.co/>

²⁵ [Streamlit • A faster way to build and share data apps](#)

khả năng cập nhật thời gian thực, hỗ trợ trực quan hóa dữ liệu mạnh mẽ qua các thư viện như Matplotlib và Plotly, và tích hợp trực tiếp với các mô hình machine learning từ TensorFlow, PyTorch, và Scikit-Learn, Streamlit đơn giản hóa quá trình phát triển ứng dụng mà không cần kiến thức về HTML, CSS hay JavaScript. Thêm vào đó, các ứng dụng Streamlit có thể dễ dàng triển khai trên nhiều nền tảng như Streamlit Sharing, Heroku, AWS, và GCP, mang lại sự linh hoạt và tiện lợi tối đa cho người dùng.

CHƯƠNG 2: XÂY DỰNG HỆ THỐNG PHÂN LOẠI CẢM XÚC VĂN BẢN BẰNG KỸ THUẬT NLP

1. Giới thiệu bài toán:

Hệ thống phân loại cảm xúc văn bản (Emotion Text Classification) là bài toán mà nhóm em quan tâm đến, với mục tiêu là xác định cảm xúc hoặc tâm trạng của một đoạn văn bản.

Trong emotion text classification, mỗi đoạn văn bản được gán một nhãn cảm xúc tương ứng với trạng thái tâm trạng của tác giả, bao gồm các loại: joy - vui vẻ, sadness - buồn bã, anger - tức giận, fear - sợ hãi, neutral – trung lập. Mục tiêu của bài toán là phát hiện và phân loại cảm xúc chính xác.

Việc hiểu được cảm xúc trong văn bản là rất quan trọng vì nó có thể áp dụng trong nhiều tình huống thực tế, từ dự đoán xu hướng cảm xúc của cộng đồng trên mạng xã hội đến hỗ trợ cho việc phát triển sản phẩm và dịch vụ dựa trên phản hồi của người dùng.

Đường dẫn trình bày code và dataset của nhóm:

- Link Dataset: [Google Drive NLP Dataset](#)
- Link Google Colab (bao gồm tất cả các model): [Google Drive NLP Model Colab](#)

2. Mô tả tập dữ liệu:

Tập dữ liệu mà nhóm chúng em sử dụng là sự kết hợp của 3 bộ dữ liệu:

- Emotion-stimulus.csv [9]: File dữ liệu này bao gồm 2 cột là Text và Emotion. Cột Emotion của file này chứa các nhãn cảm xúc bao gồm “happy”, “angry”, “fearfull”, “surprised”, “neutral”. Tuy những emotion của bộ này khác so với các emotion trong bài toán của chúng em, nhưng nhóm vẫn tận dụng được những câu/đoạn văn thuộc cột Text, và dựa vào cột Emotion để label lại.

Emotion	Text
happy	I suppose I am happy being so 'tiny'; it means I am able to surprise people with what is generally seen as my confident and outgoing personality .
happy	Lennox has always truly wanted to fight for the world title and was happy taking the tough route .
happy	He was a professional musician now , still sensitive and happy doing something he loved .
happy	Holmes is happy having the freedom of the house when we are out .
happy	I had problems with tutors trying to encourage me to diversity my work and experiment with other styles , but I was quite happy with the direction my work was heading so I stubbornly stuck to it .
happy	These days he is quite happy travelling by trolley .
happy	I'm really happy in the group now . "
happy	I was given a dummy in which to do my preliminary sketches , but the dummy was designed so beautifully by Amelia Edwards that I was really happy with my work .
happy	I must say I was not totally happy about her going on at Yeo Davis , with me in the government .
happy	Indeed , the two M boys themselves were not totally happy at first at having these initially undisciplined children visiting their home and generally being around on a regular basis .
happy	I am very happy with the way I am playing but I accept that he believes he should be England 's fly-half .

Hình 8: Bộ dữ liệu emotion-stimulus.csv

- Isear.csv [10]: Tương tự bộ dữ liệu trên, isear cũng có 2 cột là Text và Emotion. Cột Emotion bao gồm các cảm xúc: “joy”, “sadness”, “anger”, “fear”, “disgust”, “shame”, “guilt”. Đối với bộ dữ liệu này, nhóm em không lấy quá nhiều câu/văn bản từ cột Text, mà chỉ tham khảo các nhãn cảm xúc là chính.

Emotion	Text
joy	During the period of falling in love, each time that we met and especially when we had not met for a long time.
fear	When I was involved in a traffic accident.
anger	When I was driving home after several days of hard work, there was a motorist ahead of me who was driving at 50 km/hour and refused, despite his low speed to let me overtake.
sadness	When I lost the person who meant the most to me.
disgust	The time I knocked a deer down - the sight of the animal's injuries and helplessness. The realization that the animal was so badly hurt that it had to be put down, and when the animal was put down.
shame	When I did not speak the truth.
guilt	When I caused problems for somebody because he could not keep the appointed time and this led to various consequences.
joy	When I got a letter offering me the Summer job that I had applied for.
fear	When I was going home alone one night in Paris and a man came up behind me and asked me if I was not afraid to be out alone so late at night.
anger	When I was talking to HIM at a party for the first time in a long while and a friend came and interrupted us and HE left.
sadness	When my friends did not ask me to go to a New Year's party with them.
disgust	When I saw all the very drunk kids (13-14 years old) in town on Walpurgis night.
shame	When I could not remember what to say about a presentation task at an accounts meeting.
guilt	When my uncle and my neighbour came home under police escort.
fear	Every time I imagine that someone I love or I could contact a serious illness, even death.
anger	When I had been obviously unjustly treated and had no possibility of elucidating this.
sadness	When I think about the short time that we live and relate it to the periods of my life when I think that I did not use this short time.
disgust	At a gathering I found myself involuntarily sitting next to two people who expressed opinions that I considered very low and discriminating.
shame	When I realized that I was directing the feelings of discontent with myself at my partner and this way was trying to put the blame on him instead of sorting out my own feelings.

Hình 9: Bộ dữ liệu isear.csv

- Dailydialog.csv [11]: Bộ dữ liệu này khá là khác so với 2 bộ dữ liệu trên. Đây là một tập hợp các cặp câu hỏi và câu trả lời từ những cuộc trò chuyện hằng ngày, rất gần gũi với chúng ta, và nhóm em cũng đã tận dụng lại những câu/đoạn văn để làm phong phú thêm bộ dữ liệu của bài toán.

Từ ba bộ dữ liệu trên, nhóm chúng em tổng hợp lại và hình thành bộ dữ liệu cân bằng cho bài toán. Bộ dữ liệu bao gồm tập train và tập test với tỉ lệ 70-30, trong đó:

- Tập train: Bao gồm 7936 hàng, tương ứng 7936 câu/đoạn văn. Cột đầu tiên là Emotion, bao gồm 5 nhãn cảm xúc (“joy”, “fear”, “anger”, “sadness” và “neutral”) và cột thứ hai là Text, gồm các câu/đoạn văn mà đã được label các emotion tương ứng.

Emotion	Text
neutral	There are tons of other paintings that I think are better .
sadness	Yet the dog had grown old and less capable , and one day the gillie had come and explained with great sorrow that the dog had suffered a stroke , and must be put down .
fear	When I get into the tube or the train without paying for the ticket.
fear	This last may be a source of considerable disquiet and one might not at first see how such obviously ` immoral " content could be defended as part of a system of morality .
anger	She disliked the intimacy he showed towards some of them , was resentful of the memories they shared of which she was not a part , and felt excluded .
sadness	When my family heard that my Mother's cousin who lives in England wrote us to tell that he had cancer of the lymph glands.
joy	Finding out I am chosen to collect norms for Chinese aphasia (I will contribute to China's catching up with the West in neuropsychology).
anger	A spokesperson said : ` Glen is furious that the new ` Anarchy " promo features footage of Sid Vicious as well as himself .
neutral	Yes .
sadness	When I see people with burns I feel sad, actually I can not even express my feelings as I think that they must suffer a lot.
fear	I had gone to the hospital for my research and got late in reaching home. I feared that when I reached home there would be a quarrel because of my being late.
sadness	One day I heard from a friend that the boy I loved had gone out with her and not with me.
joy	Connor's voice was gleeful .
anger	Very funny . What's wrong with you today ? You are my secretary and you are not supposed to talk to me in that tone of voice . Do you know that ?
neutral	Perhaps we need to have more babies ! Tina gave birth to a baby boy yesterday .
neutral	Why ?

Hình 10: Mô tả train dataset

- Tập test: Tương tự tập train. Số lượng bao gồm 3394 hàng, tương ứng 3394 câu/đoạn văn.

Emotion	Text
sadness	I experienced this emotion when my grandfather passed away.
neutral	when I first moved in , I walked everywhere . But within a week , I had my purse stolen — just a block away from the police station ! Now , I always take public transportation .
anger	‘ Oh ! ’ she bleated , her voice high and rather indignant .
fear	However , does the right hon. Gentleman recognise that profound disquiet has been expressed about some of the proposals in the Bill , particularly the fast-track ones ?
sadness	My boyfriend didn't turn up after promising that he was coming.
neutral	It's freezing .
sadness	That ' s not all ! I also had to finish writing a sales report for my boss . In the end , I finished everything . I wonder what will be waiting for me tomorrow morning .
anger	I don't have a warrant .
neutral	I guess so .
sadness	I was just robbed !
neutral	This is it ?
neutral	Well , do you think I have to ?
fear	They seemed anxious and hesitant about leaving , as if uncertain of which direction to take .
anger	I experienced anger most recently when I had committed a sin which I had gone a week and a half without doing. I had made a vow to God and had blown it. Now I had to start all over.

Hình 11: Mô tả test dataset

Với bộ dataset trên, mục tiêu của nhóm chúng em là

- **Input:** Một message bất kì, ví dụ: message = “This made my heart melt, thank you for posting this!”²⁶
- **Output:** Cảm xúc tương ứng với message trên → “joy”

3. Thực nghiệm:

3.1. Mô hình đề xuất:

3.1.1. Mô hình học máy truyền thống:

Model Đánh giá	Naïve Bayes	LR	Random Forest	Linear SVM
Kiến trúc	Là mô hình xác suất dựa trên định lý Bayes với giả định các features độc lập nhau	Là mô hình tuyến tính sử dụng hàm logistic (sigmoid) để dự đoán xác suất của các lớp	Là một tập hợp của nhiều decision trees, được xây dựng trên các mẫu con của dữ liệu và đặc trưng	SVM tìm kiếm các hyperlane để phân tách các lớp trong không gian đặc trưng
Ưu điểm	<ul style="list-style-type: none"> - Tính toán nhanh chóng và tốn ít tài nguyên - Hoạt động tốt với tập dữ liệu nhỏ - Dễ dàng triển khai và giải thích 	Đơn giản, mạnh, hiệu quả, dễ hiểu	<ul style="list-style-type: none"> - Độ chính xác cao - Xử lý được dữ liệu phức tạp - Giảm thiểu được overfitting 	<ul style="list-style-type: none"> - Hiệu suất tốt - Ổn định, hiệu quả - Tối ưu khả năng phân loại
Nhược điểm	<ul style="list-style-type: none"> - Khó áp dụng thực tế, độ chính xác thấp - Hiệu suất kém nếu các features không chặt chẽ 	<ul style="list-style-type: none"> - Không xử lý tốt các quan hệ phi tuyến - Cần chuẩn hóa đặc trưng 	<ul style="list-style-type: none"> - Tốc độ chậm - Khó giải thích hơn các mô hình tuyến tính 	<ul style="list-style-type: none"> - Nhạy cảm với các parameters - Khó khăn khi gặp dữ liệu lớn - Thời gian lâu

3.1.2. Mô hình học sâu (pretrained model):

²⁶https://www.reddit.com/r/thisismylifenow/comments/diez85/comment/f3vow1k/?utm_source=share&utm_medium=web3x&utm_name=web3xcss&utm_term=1&utm_content=share_button

Mô hình LSTM/GRU: Hiệu quả cho việc mô hình hóa các thông tin ngữ cảnh dài hạn trong văn bản, nhưng tốc độ huấn luyện chậm hơn và khó song song hóa.

Mô hình BERT: Mô hình tiên tiến và mạnh mẽ cho các tác vụ NLP, nắm bắt ngữ cảnh phức tạp tốt, nhưng yêu cầu tài nguyên tính toán lớn và thời gian huấn luyện dài.

Đối với 3 mô hình này, nhóm em sẽ giải thích kỹ hơn ở phần dưới.

3.2. Thực nghiệm:

3.2.1. Mô hình học máy truyền thống:

Như đã giới thiệu ở trên, ở phần này chúng em sẽ thử nghiệm 4 mô hình học máy: Naïve Bayes, Random Forest, LR và Linear SVM, áp dụng cho bài toán phân loại cảm xúc văn bản.

Quá trình thực hiện:

Import Dataset → Preprocessing → Representation → Áp dụng model classifiers → Đánh giá mô hình (dựa trên F1 và Confusion Matrix) → Lưu lại model → Test một input khác và mong muốn output tương ứng.

- Import các thư viện cần dùng:

```
import pandas as pd
import numpy as np

# text preprocessing
from nltk import word_tokenize
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
import re

# plots and metrics
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix

# feature extraction / vectorization
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

# classifiers
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline

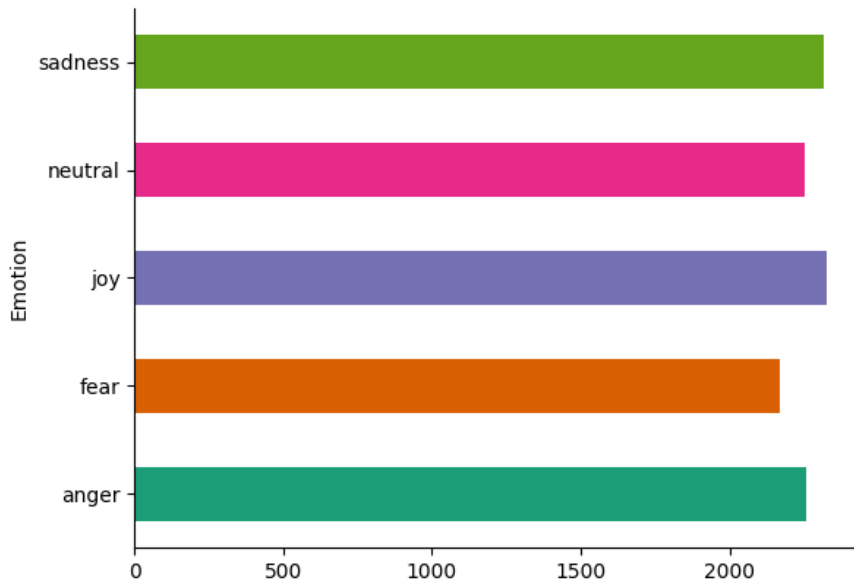
# save and load a file
import pickle
```

Hình 12: Import các thư viện cần dùng

- Import Dataset / Mô tả Dataset: sử dụng thư viện pandas để đọc dữ liệu

```
df_train = pd.read_csv('data/data_train.csv')  
df_test = pd.read_csv('data/data_test.csv')
```

Hình 13: Đọc dữ liệu



Hình 14: Visualize dataset

- Preprocessing & tokenization:

```
def preprocess_and_tokenize(data):  
  
    # xóa html markup  
    data = re.sub("<.*?>", "", data)  
  
    # xóa urls  
    data = re.sub(r'http\S+', '', data)  
  
    # xóa hashtags và @names  
    data = re.sub(r"#\d\w\.", '', data)  
    data = re.sub(r"@d\w\.", '', data)  
  
    # xóa dấu câu và ký tự đặc biệt  
    data = re.sub(r"\\W|\\d", " ", data)  
  
    # xóa các khoảng trắng thừa  
    data = data.strip()  
  
    # sử dụng nltk để tokenize  
    data = word_tokenize(data)  
  
    # sử dụng nltk để đổi lại đúng thì ban đầu  
    porter = PorterStemmer()  
    stem_data = [porter.stem(word) for word in data]  
  
    return stem_data
```

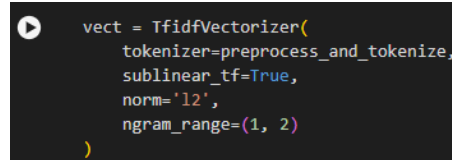
Hình 15: Preprocessing và tokenization sử dụng thư viện nltk

- Representation: sử dụng kỹ thuật TF-IDF (đã được giải thích ở Chương 1)



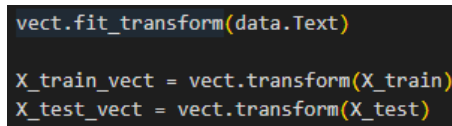
```
import nltk
nltk.download('punkt')
```

Hình 16: Tải bộ tokenizers 'punkt' của nltk



```
vect = TfidfVectorizer(
    tokenizer=preprocess_and_tokenize,
    sublinear_tf=True,
    norm='l2',
    ngram_range=(1, 2)
)
```

Hình 17: Thiết lập TF-IDF với các thông số cụ thể



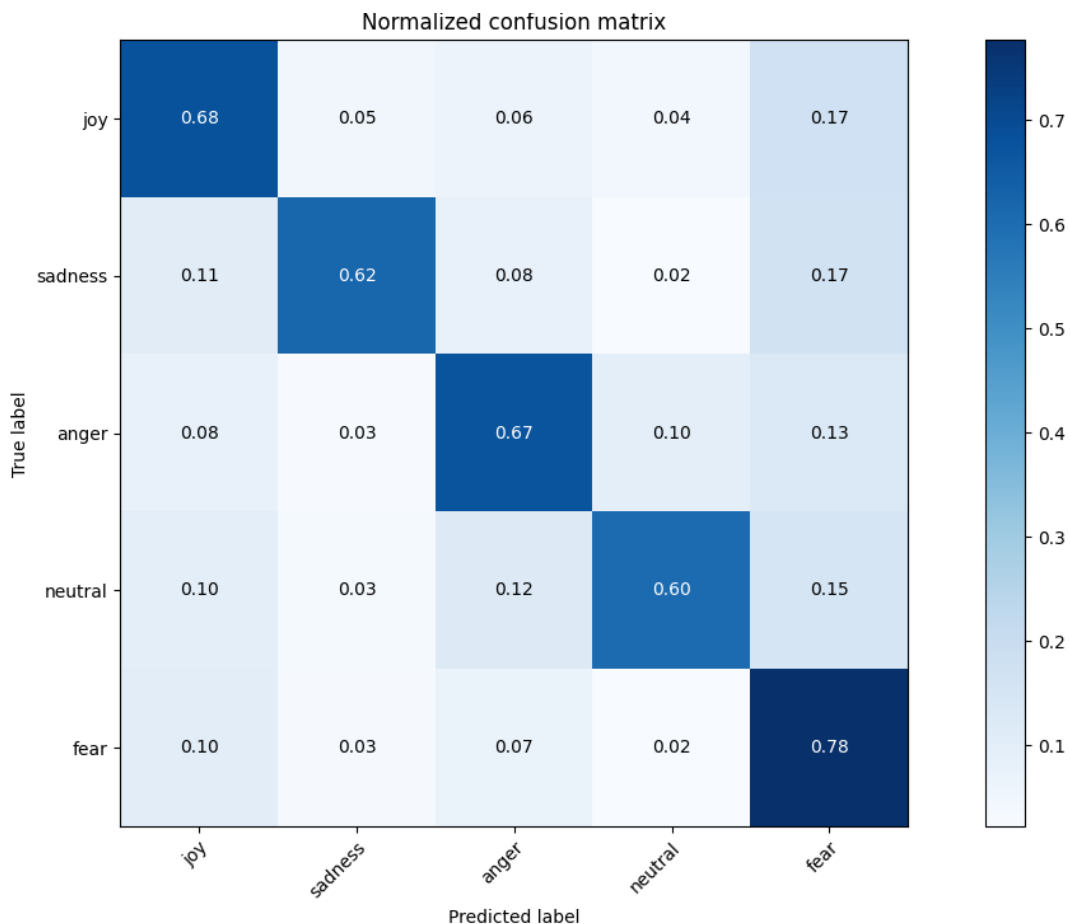
```
vect.fit_transform(data.Text)

X_train_vect = vect.transform(X_train)
X_test_vect = vect.transform(X_test)
```

Hình 18: Áp dụng vectorizer đã học được để chuyển đổi tập train, test thành các vector TF-IDF

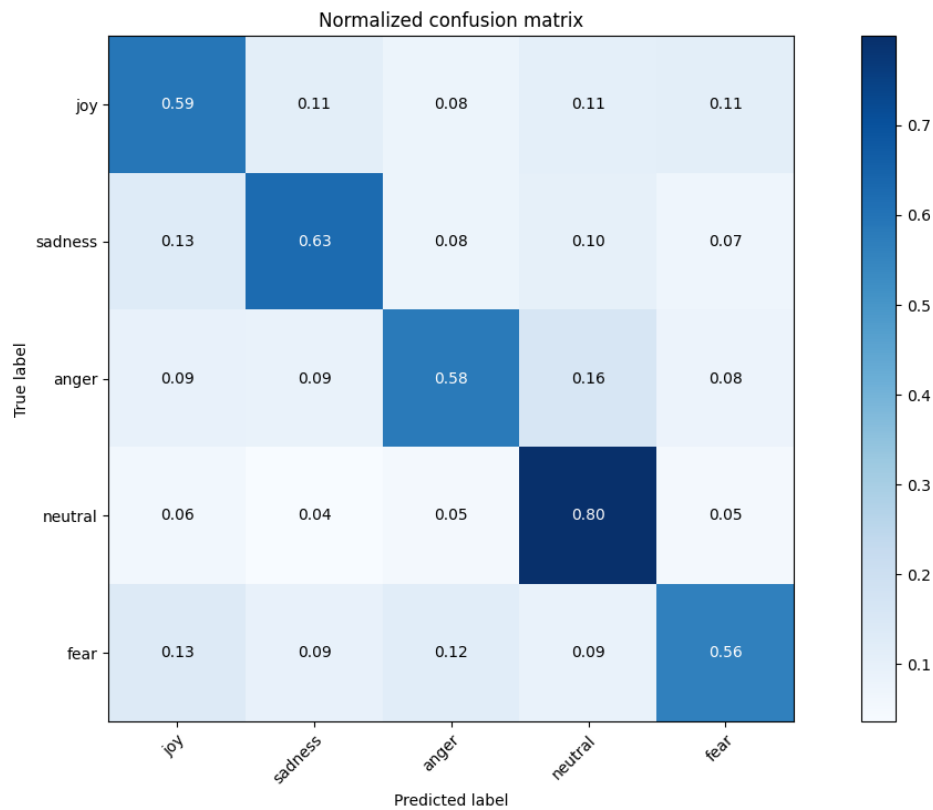
- Áp dụng các model classifier và đánh giá dựa trên F1 Score + Confusion Matrix:

- o Naïve Bayes: Tiêu chí đánh giá F1 Score: 67.02



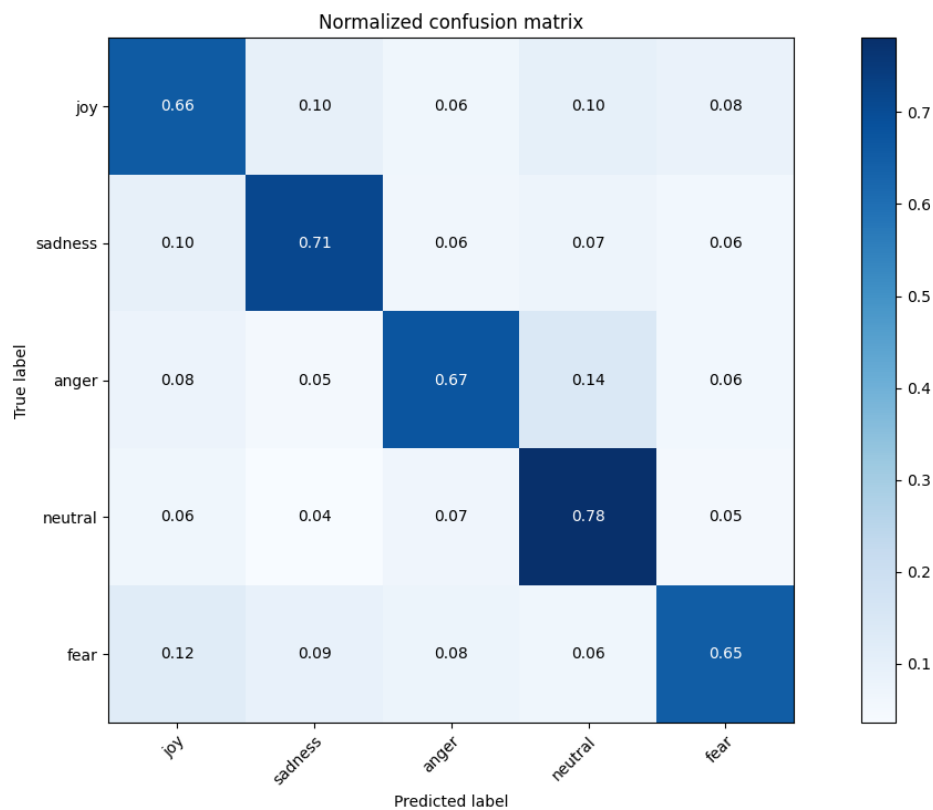
Hình 19: Confusion Matrix của mô hình Naïve Bayes

- Random Forrest: Tiêu chí đánh giá F1 Score: 62.98



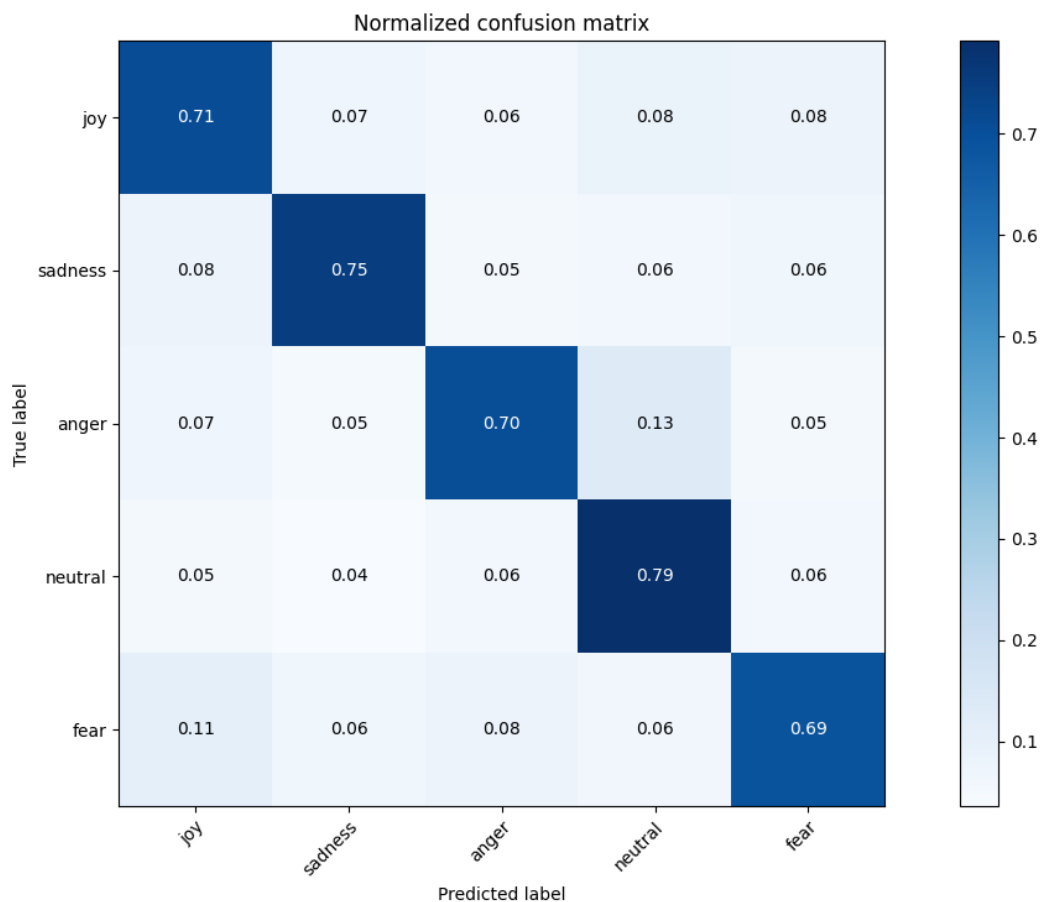
Hình 20: Confusion Matrix của mô hình Random Forest

- Logistic Regression: Tiêu chí đánh giá F1 Score: 69.35



Hình 21: Confusion Matrix của mô hình LR

- Linear SVM: Tiêu chí đánh giá F1 Score: 72.71



Hình 22: Confusion Matrix của mô hình Linear SVM

- Bảng so sánh model:

Model	F1 Score
Naïve Bayes	67.02
Random Forest	62.98
LR	69.35
Linear SVM (highest)	72.71

- Lưu lại model Linear SVM để sử dụng đánh giá cho một số message tự thu thập:

```
# tạo pipeline với tf-idf vectorizer và Linear SVM
svm_model = Pipeline([
    ('tfidf', vect),
    ('clf', svc),
])

# lưu model
filename = 'models/tfidf_svm.sav'
pickle.dump(svm_model, open(filename, 'wb'))
```

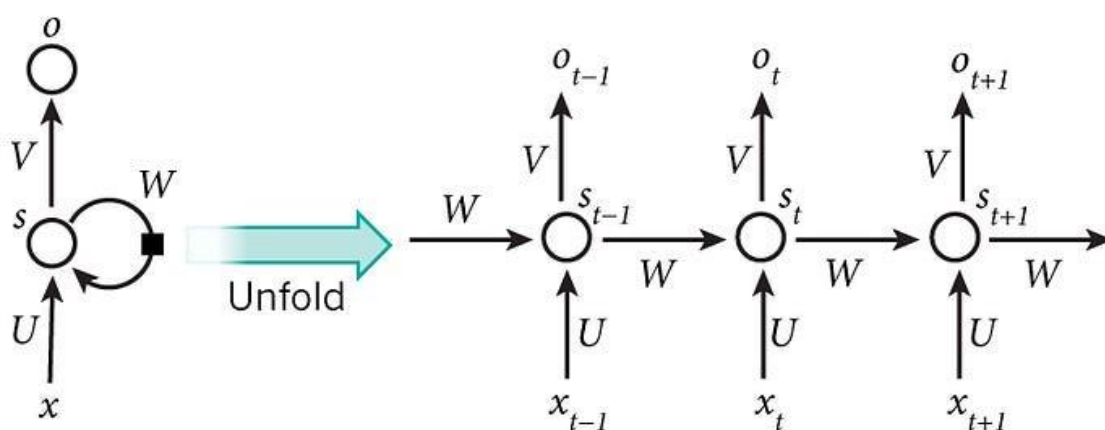
Hình 23: Tạo pipeline để sử dụng đánh giá và lưu model lại

- Dự đoán một message bất kì để kiểm tra độ chính xác: message truyền vào là “His eyes met hers, and in that moment, she knew she was deeply in love.” → Output: “joy” → Đúng với output mong muốn.

3.2.2. Mô hình pretrained RNN (LSTM/GRU):

Trước khi đi vào phần code, nhóm em sẽ trình bày cách mà RNN hoạt động, và tại sao lại sử dụng 2 biến thể LSTM và GRU. [12] [13]

RNN được sinh ra để giải quyết các bài toán có dữ liệu tuần tự, và kiến trúc nó khá đơn giản, khả năng liên kết các thành phần có khoảng cách xa trong câu không tốt → Vấn đề của RNN là Bộ nhớ của nó ngắn hạn (short-term memory [14]).



Hình 24: Kiến trúc RNN (Nguồn ²⁷)

Nguyên nhân dẫn đến vấn đề này là do RNN chịu ảnh hưởng bởi việc gradient bị thấp dần trong quá trình learning (vanishing gradient), mà gradient lại là thành phần quan trọng bậc nhất trong việc huấn luyện các model. Vì thế khi giá trị của gradient được tạo bởi các thành phần phía đầu đoạn văn trở nên quá nhỏ, nó sẽ không đóng góp gì cho việc học của model. Vì thế, chúng ta gọi trường hợp này là model "quên" các thành phần đứng đầu đoạn văn: chúng không đóng góp gì cho việc dự đoán các từ ở phía sau.

new weight = weight - learning rate*gradient

$$\boxed{2.0999} = \boxed{2.1} - \boxed{0.001}$$

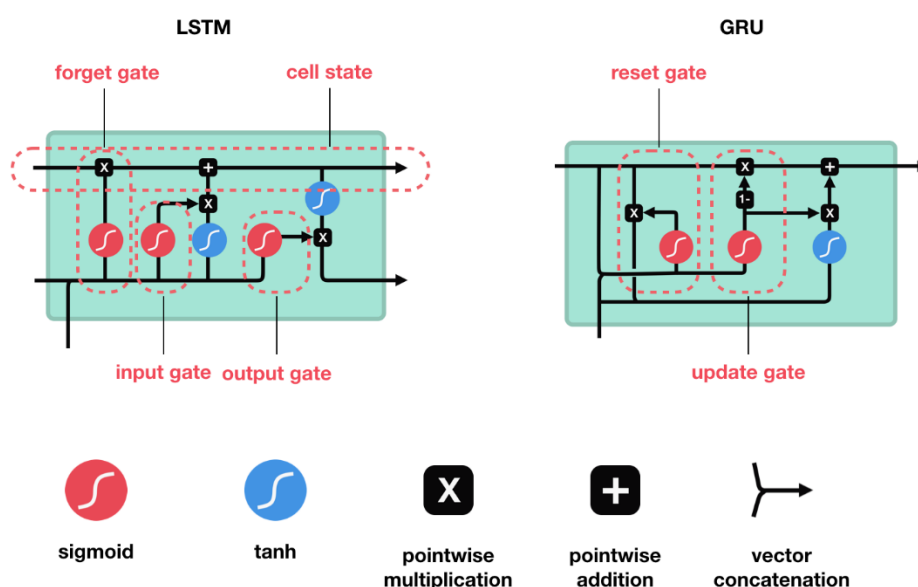
Not much of a difference update value

Hình 25: Gradient của RNN bị giảm dần theo thời gian learning (Nguồn ²⁸)

²⁷ <https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSEfQ4RUsp9KF-vgKuon8J2bvq4hKRtUFTt2XSGCP3m3WDwyyfP>

Quan sát vấn đề của RNN, nhóm em nhận thấy kiến trúc này không hề có cơ chế lọc những thông tin không cần thiết. Bộ nhớ của RNN đã có hạn rồi, mà còn lưu tất cả những thông tin không cần thiết ấy thì sẽ dẫn đến quá tải, từ đó quên những dữ liệu xa trong quá khứ (tương tự con người).

Để khắc phục vấn đề mà RNN gặp phải, người ta đã phát triển ra các biến thể khác để khắc phục nhược điểm ấy. Đó là LSTM và GRU



Hình 26: Kiến trúc của LSTM và GRU (Nguồn ²⁹)

Như hình 26, có thể thấy trong mỗi module của 2 kiến trúc đều có các cổng (gate) – giúp kiến trúc đánh giá được mức độ quan trọng của thông tin, từ đó đưa ra quyết định giữ lại hay bỏ đi. Nhờ cơ chế này, các thông tin quan trọng có thể được truyền đi rất xa và vẫn có ảnh hưởng đáng kể trong tương lai xa.

Nhóm em đưa ra 1 ví dụ: Nhóm đang đi mua một hộp ngũ cốc trên mạng, và muốn đọc review về sản phẩm này trước khi mua để xem nó có thực sự đáng đồng tiền bát gạo hay không.

²⁸ https://miro.medium.com/max/1400/1*PYiQa_bNzM8ugYz_D1yvgw.png

²⁹ https://miro.medium.com/max/1400/1*yBXV9o5q7L_CvY7quJt3WQ.png

Customers Review 2,491



Thanos

September 2018

Verified Purchase

Amazing! This box of cereal gave me a perfectly balanced breakfast, as all things should be. I only ate half of it but will definitely be buying again!



A Box of Cereal
\$3.99

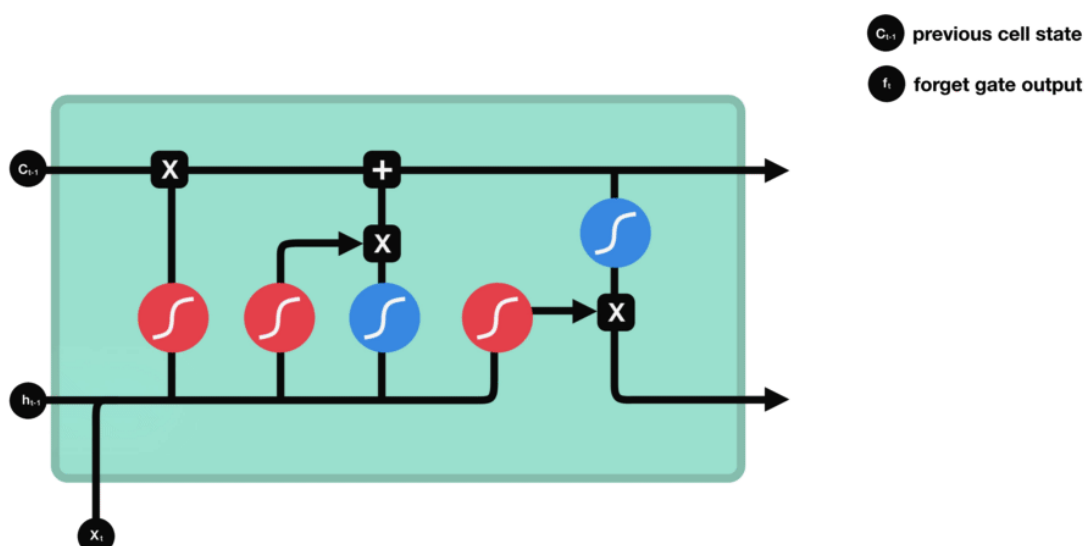
Hình 27: Một review nào đó về hộp ngũ cốc

Khi đọc review, não trong tiềm thức đã bỏ qua các từ không mang nhiều ý nghĩa, như trong trường hợp này là *this, as,....* Và nếu có ai đó muốn mua ngũ cốc, muốn hỏi ý kiến / review của nhóm em về sản phẩm này, thì chúng em cũng sẽ dựa trên những keyword của review trên ảnh mà chế ra 1 review khác, mang ý nghĩa tương tự, và những từ khác gần như sẽ “tái này qua tái kia” và quên ngay lập tức.

Đó là cơ chế hoạt động của LSTM và GRU: nó chỉ nhớ các thông tin liên quan cho việc dự đoán, các thông tin thừa / không quan trọng sẽ bị bỏ đi.

Như hình 26, ta có thể thấy kiến trúc của LSTM bao gồm 3 cổng:

- Cổng quên (Forget gate): Cổng này quyết định xem thông tin nào trong bộ nhớ hiện tại được giữ và thông tin nào bị bỏ lại. Thông tin đầu vào được cho vào hàm sigmoid³⁰. Đầu ra của hàm này đóng vai trò là mask để lọc thông tin từ trạng thái cell.

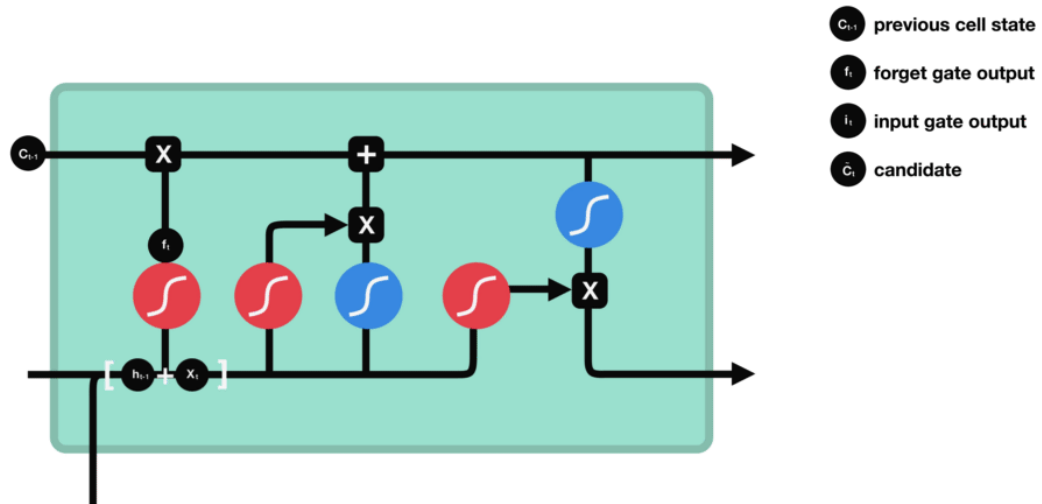


Hình 28: Mô tả Forget gate của LSTM (Nguồn ³¹)

³⁰ https://en.wikipedia.org/wiki/Sigmoid_function

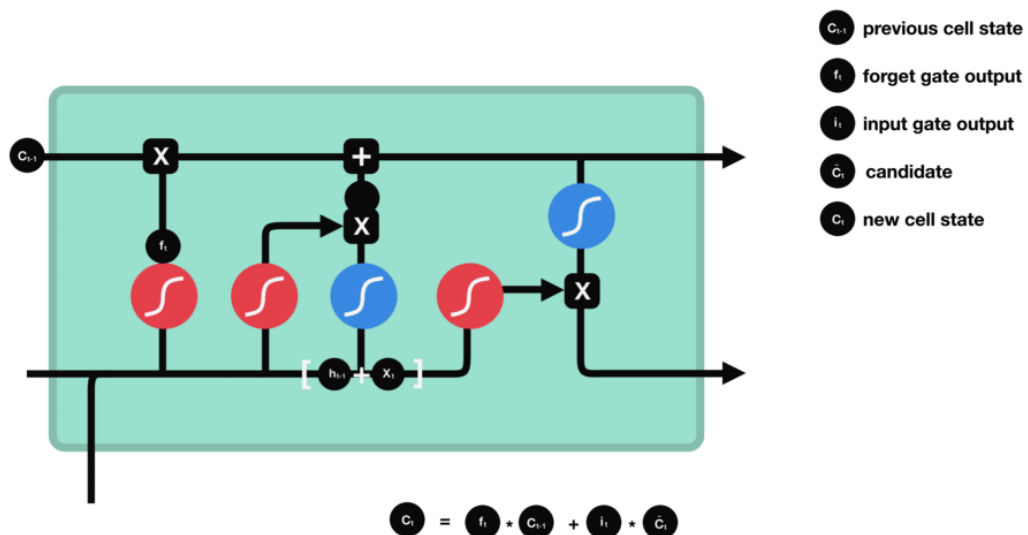
³¹ https://miro.medium.com/v2/format:webp/1*GjehOa513_BgpDDP6Vkw2Q.gif

- Cổng vào (Input gate): Cổng này dùng để cập nhật bộ nhớ với thông tin mới. Ở đây có xuất hiện 2 hàm sigmoid và hàm tanh³². Tác dụng của chúng cũng như trên. Output từ hàm sigmoid sẽ có tác dụng lọc thông tin đã qua xử lý từ output hàm tanh.



Hình 29: Mô tả Input Gate của LSTM (Nguồn ³³)

- Nhờ 2 cổng này, ta có thể tính toán được giá trị của trạng thái cell hiện tại, từ đó truyền đi cho các từ phía sau.



Hình 30: Mô tả quá trình tổng quát mà Forget Gate và Input Gate xử lý (Nguồn ³⁴)

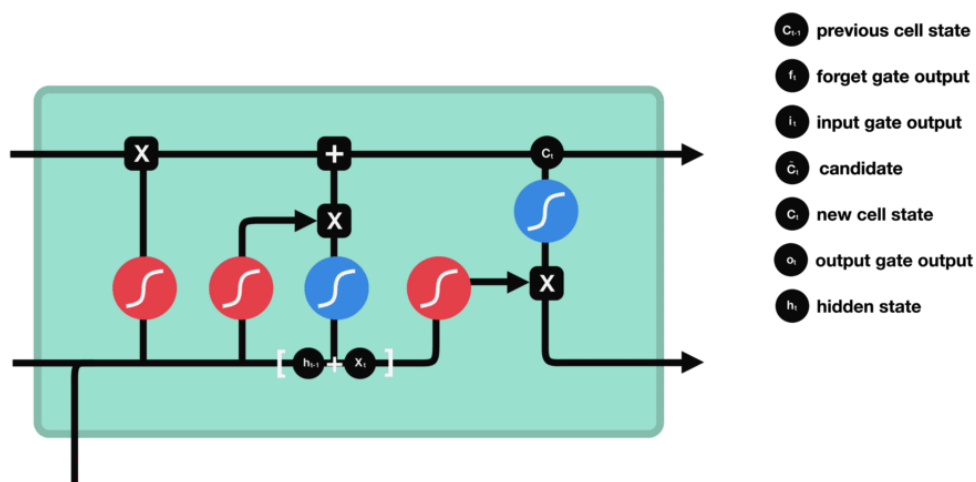
- Cổng ra: Cổng này quyết định output của từ hiện tại là gì. Nó lấy được thông tin từ 2 nguồn: trạng thái cell và input hiện tại. Trạng thái cell sau khi chỉnh sửa sẽ đi qua hàm tanh và input hiện tại thì được đi qua hàm sigmoid. Từ đây ta kết

³² <https://www.cuemath.com/trigonometry/tanh/>

³³ https://miro.medium.com/v2/format:webp/1*TTmYy7Sy8uUXxUXfzmoKbA.gif

³⁴ https://miro.medium.com/v2/format:webp/1*S0rXleO_VoUVOyrYHckUWg.gif

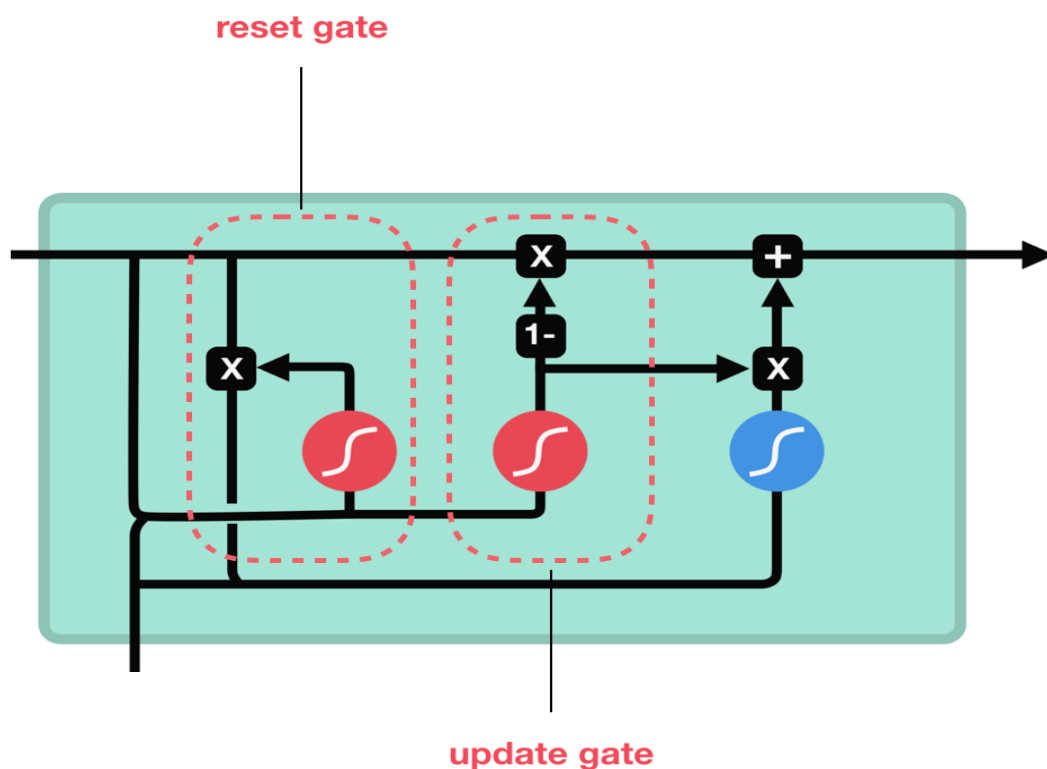
hợp 2 kết quả trên để có được kết quả đầu ra. Chú ý rằng cả kết quả đầu ra và cả trạng thái cell đều được đưa vào bước tiếp theo.



Hình 31: Mô tả Output Gate (Nguồn ³⁵)

Đó là tổng quan những gì LSTM thực hiện: Lọc bộ nhớ hiện tại, chất lọc thông tin hiện tại để đưa vào bộ nhớ và kết hợp bộ nhớ, dữ liệu hiện tại để đưa ra output.

Đối với GRU, cơ chế của nó chỉ có 2 cổng để chất lọc thông tin: cổng reset và cổng update. GRU không có trạng thái cell mà chỉ có output dùng để đưa ra quyết định, mà cũng dùng làm thông tin cho bước tiếp theo.



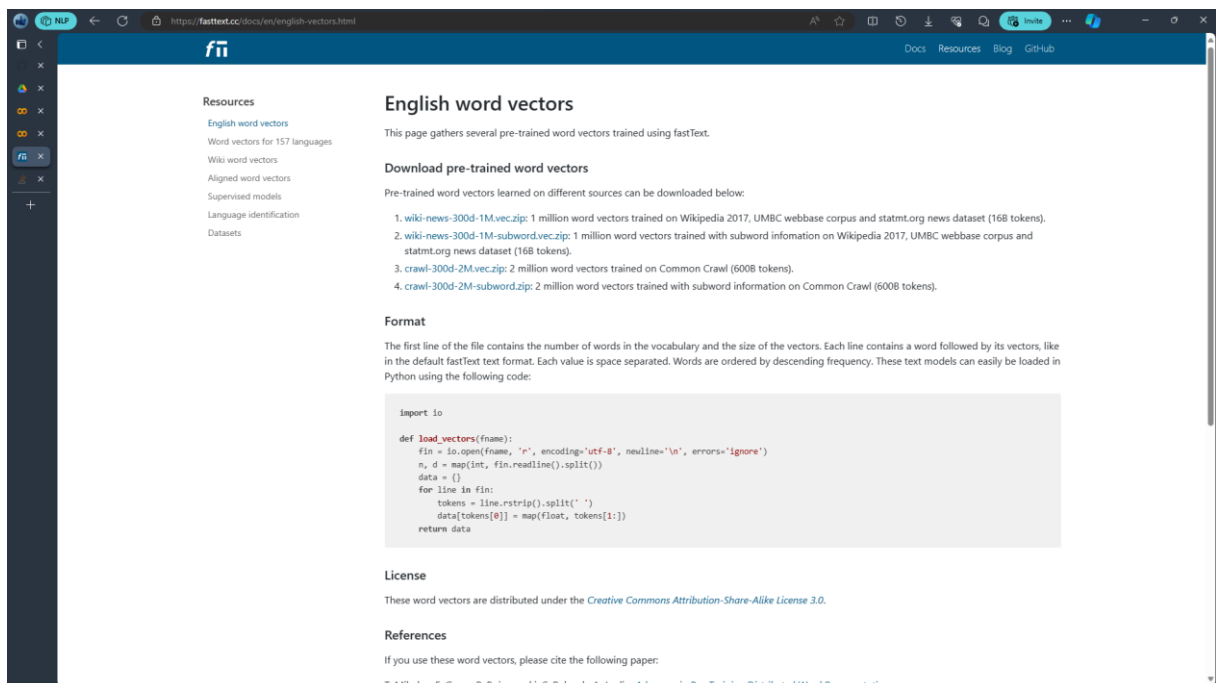
³⁵ https://miro.medium.com/v2/format:webp/1*VOXRghOShoWWks6ouoDN3Q.gif

Hình 32: Mô tả gate của GRU (Nguồn ³⁶)

Do mất đi trạng thái cell, nên tác dụng của 2 cổng trên khá khó để phân biệt rõ ràng, có thể nói là cả 2 cổng đều có tác dụng chốt lọc thông tin từ các đầu vào của cell để đưa ra một output thỏa mãn cả 2 tiêu chí là lưu giữ thông tin quá khứ và có khả năng đưa ra quyết định hiện tại một cách chính xác nhất. Do giảm bớt tính phức tạp nên GRU hoạt động nhanh hơn đôi chút so với LSTM. Về tính hiệu quả thì rất khó để đưa ra kết luận chính xác, nhưng có thể nói mức độ phổ biến LSTM là vượt trội khi so với GRU.

Lý thuyết tổng quát là như vậy, nhóm em sẽ đến phần code (một số phần tương tự với các mô hình học máy truyền thống nhóm em sẽ không đề cập kỹ, mà chỉ ghi theo giai đoạn):

- Import các thư viện cần thiết (Pandas, Numpy, các thư viện DL Keras và công cụ NLTK).
- Đọc dữ liệu tập train và test, làm sạch, tokenizer và chuyển thành các chuỗi số thông qua Tokenizer của Keras.
- Labeling các cảm xúc, chuyển thành dạng one-hot.
- Tạo Embedding Matrix (sử dụng pretrained w2v của wiki):



Hình 33: Pretrained word vectors của wikipedia

³⁶ https://miro.medium.com/v2/format:webp/1*jhi5uOm9PvZfmxvfaCektw.png

- Bộ wiki-news-300d-1M mà nhóm em sử dụng là một tập các vector word embeddings được huấn luyện trước bởi FastText³⁷ (một thư viện của FAIR). Bộ này được huấn luyện trên các bài viết từ Wikipedia, với mỗi từ được ánh xạ thành một vector 300 chiều. Những vector này được huấn luyện trên tập dữ liệu lớn của Wikipedia để nắm bắt được ngữ cảnh mà các từ xuất hiện cùng nhau, nhờ đó có ngữ nghĩa tương tự sẽ có các vector gần nhau trong không gian vector.

```
def create_embedding_matrix(filepath, word_index, embedding_dim):
    vocab_size = len(word_index) + 1
    embedding_matrix = np.zeros((vocab_size, embedding_dim))
    with open(filepath) as f:
        for line in f:
            word, *vector = line.split()
            if word in word_index:
                idx = word_index[word]
                embedding_matrix[idx] = np.array(
                    vector, dtype=np.float32)[:embedding_dim]
    return embedding_matrix
```

Hình 34: Đoạn code tạo embedding matrix bằng cách ánh xạ lên bộ pretrained w2v

```
fname = 'embeddings/wiki-news-300d-1M.vec'

embedd_matrix = create_embedding_matrix(fname, index_of_words, embed_num_dims)
embedd_matrix.shape

(12088, 300)
```

Hình 35: Kiểm tra kích thước của embedding matrix

- Tiếp theo, kiểm tra xem có từ nào mới trong vocabulary không:

```
new_words = 0

for word in index_of_words:
    entry = embedd_matrix[index_of_words[word]]
    if all(v == 0 for v in entry):
        new_words = new_words + 1

print('Words found in wiki vocab: ' + str(len(index_of_words) - new_words))
print('New words found: ' + str(new_words))

Words found in wiki vocab: 11442
New words found: 645
```

Hình 36: Kiểm tra từ mới trong từ điển

- Kết quả cho thấy có 11442 từ đã có trong wiki vocabulary, và 645 từ mới, tức không có sẵn trong từ điển.
- Tiếp theo, khởi tạo mô hình GRU, với kiến trúc như sau:

³⁷ <https://fasttext.cc/docs/en/english-vectors.html>

- Tạo mô hình tuần tự với dòng code: `model = Sequential()`
- Thêm lớp Embedding: `model.add(embedd_layer)`, trong đó `embedd_layer` được khởi tạo như ảnh 37

```
embedd_layer = Embedding(vocab_size,
                          embed_num_dims,
                          input_length = max_seq_len,
                          weights = [embedd_matrix],
                          trainable=True)
```

Hình 37: Sử dụng pretrained w2v để tạo lớp embedding cho mô hình GRU

- Thêm lớp GRU và lớp Dense (bao gồm lớp fully connected với `num_classes = 5` đầu ra, tương ứng với 5 nhãn cảm xúc của bài toán, activation function là soft max), được mô tả như ảnh 38

```
if bidirectional:
    model.add(Bidirectional(GRU(units=gru_output_size,
                                dropout=0.2,
                                recurrent_dropout=0.2)))
else:
    model.add(GRU(units=gru_output_size,
                  dropout=0.2,
                  recurrent_dropout=0.2))

model.add(Dense(num_classes, activation='softmax'))
```

Hình 38: Xây dựng lớp GRU và lớp Dense

- Cuối cùng là biên dịch mô hình (với hàm loss là hàm categorical crossentropy³⁸, thuật toán tối ưu Adam³⁹ thường dùng trong neural network, sử dụng accuracy làm chỉ số đánh giá mô hình) và tóm tắt lại model

```
model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 500, 300)	3626400
bidirectional (Bidirectional)	(None, 256)	330240
dense (Dense)	(None, 5)	1285

=====
Total params: 3957925 (15.10 MB)
Trainable params: 3957925 (15.10 MB)
Non-trainable params: 0 (0.00 Byte)

Hình 39: Compile model và model summary

³⁸ https://gombru.github.io/2018/05/23/cross_entropy_loss/

³⁹ <https://viblo.asia/p/thuat-toan-toi-uu-adam-aWj53k8Q56m>

- Bắt tay vào huấn luyện (train) mô hình thôi. Chúng em sẽ đặt `batch_size = 128` và `epochs = 15` (hình 40):

```

batch_size = 128
epochs = 15

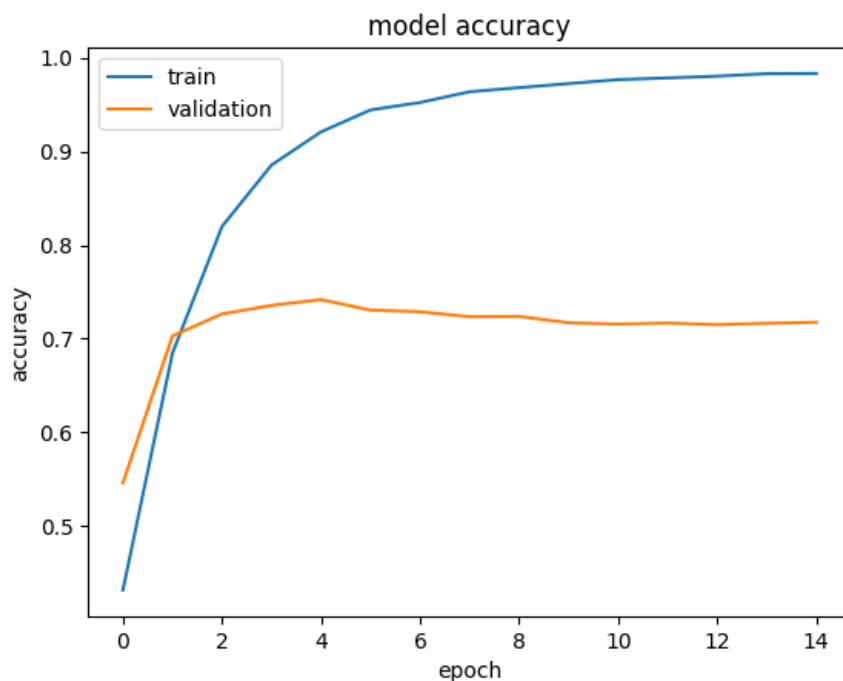
hist = model.fit(X_train_pad, y_train,
                 batch_size=batch_size,
                 epochs=epochs,
                 validation_data=(X_test_pad, y_test))

```

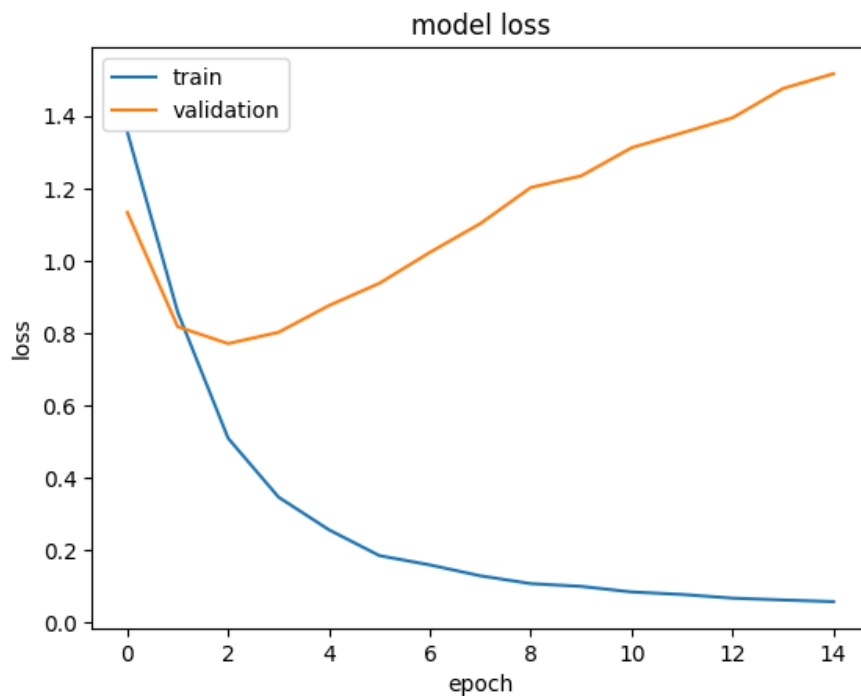
Epoch 1/15
 62/62 [=====] - 220s 3s/step - loss: 1.3564 - accuracy: 0.4317 - val_loss: 1.1334 - val_accuracy: 0.5461
 Epoch 2/15
 62/62 [=====] - 200s 3s/step - loss: 0.8574 - accuracy: 0.6845 - val_loss: 0.8170 - val_accuracy: 0.7029
 Epoch 3/15
 62/62 [=====] - 200s 3s/step - loss: 0.5087 - accuracy: 0.8198 - val_loss: 0.7704 - val_accuracy: 0.7265
 Epoch 4/15
 62/62 [=====] - 200s 3s/step - loss: 0.3457 - accuracy: 0.8854 - val_loss: 0.8017 - val_accuracy: 0.7356
 Epoch 5/15
 62/62 [=====] - 197s 3s/step - loss: 0.2554 - accuracy: 0.9208 - val_loss: 0.8758 - val_accuracy: 0.7418
 Epoch 6/15
 62/62 [=====] - 197s 3s/step - loss: 0.1841 - accuracy: 0.9443 - val_loss: 0.9374 - val_accuracy: 0.7306
 Epoch 7/15
 62/62 [=====] - 196s 3s/step - loss: 0.1584 - accuracy: 0.9522 - val_loss: 1.0228 - val_accuracy: 0.7289
 Epoch 8/15
 62/62 [=====] - 199s 3s/step - loss: 0.1283 - accuracy: 0.9637 - val_loss: 1.1024 - val_accuracy: 0.7235
 Epoch 9/15
 62/62 [=====] - 197s 3s/step - loss: 0.1067 - accuracy: 0.9681 - val_loss: 1.2021 - val_accuracy: 0.7238
 Epoch 10/15
 62/62 [=====] - 196s 3s/step - loss: 0.0990 - accuracy: 0.9724 - val_loss: 1.2344 - val_accuracy: 0.7171
 Epoch 11/15
 62/62 [=====] - 199s 3s/step - loss: 0.0836 - accuracy: 0.9767 - val_loss: 1.3121 - val_accuracy: 0.7156
 Epoch 12/15
 62/62 [=====] - 196s 3s/step - loss: 0.0767 - accuracy: 0.9784 - val_loss: 1.3531 - val_accuracy: 0.7168
 Epoch 13/15
 62/62 [=====] - 194s 3s/step - loss: 0.0665 - accuracy: 0.9803 - val_loss: 1.3948 - val_accuracy: 0.7150
 Epoch 14/15
 62/62 [=====] - 204s 3s/step - loss: 0.0613 - accuracy: 0.9830 - val_loss: 1.4758 - val_accuracy: 0.7165
 Epoch 15/15
 62/62 [=====] - 202s 3s/step - loss: 0.0570 - accuracy: 0.9832 - val_loss: 1.5167 - val_accuracy: 0.7177

Hình 40: Quá trình train model

- Thời gian huấn luyện cho mỗi epoch trung bình khoảng 210 giây, sử dụng GPU T4 của google colab, tổng thời gian khoảng 50-60 phút.
- Sau epoch 15, giá trị loss = 0.0570, accuracy = 0.9832 đối với tập train; đối với tập validation, loss = 1,5167, accuracy = 0.7177

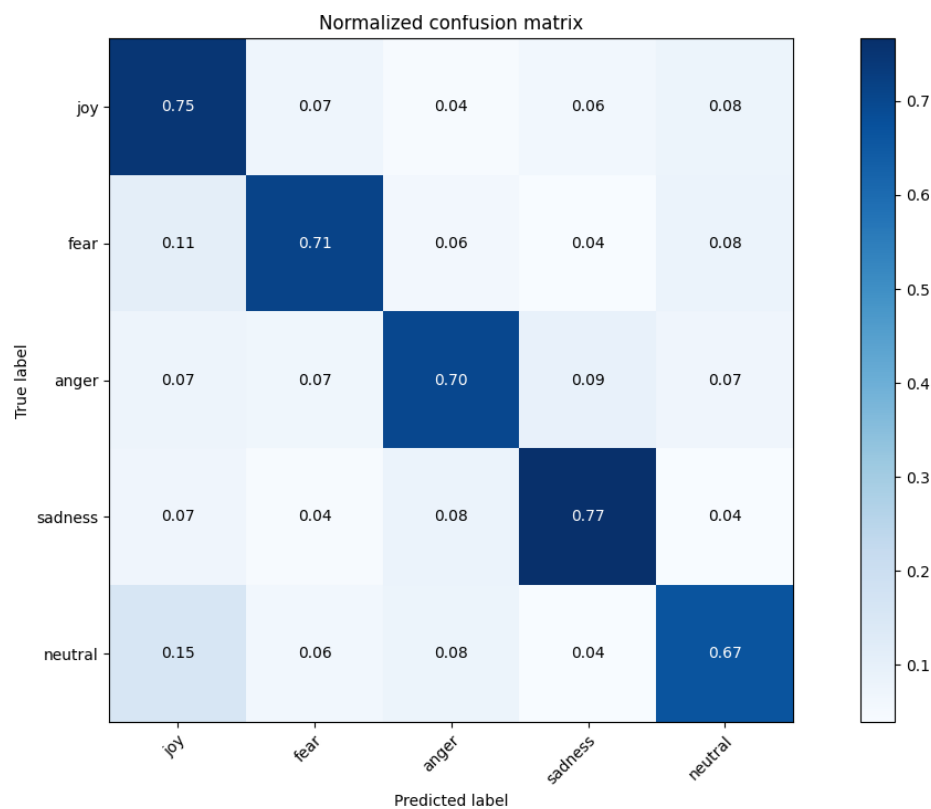


Hình 41: Biểu đồ trực quan accuracy của train và validation qua 15 epochs



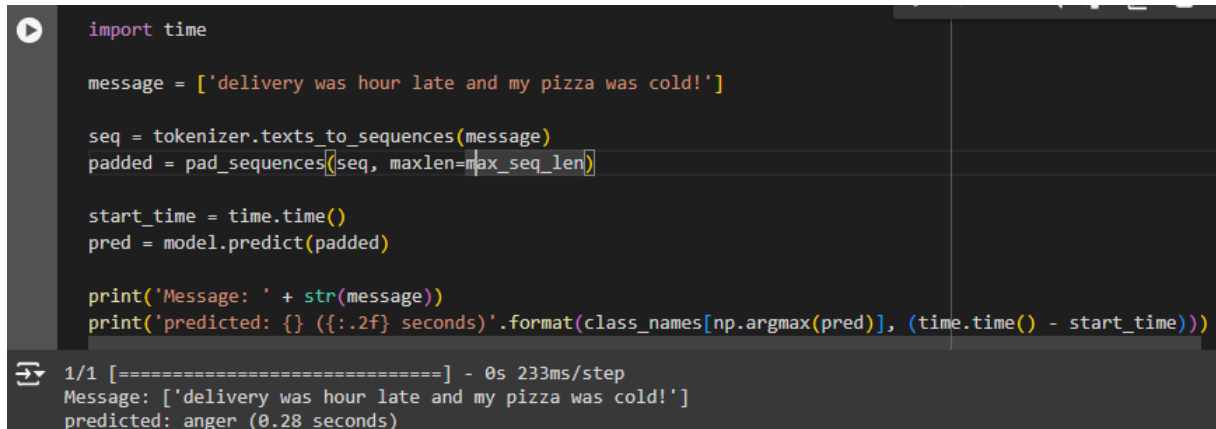
Hình 42: Biểu đồ trực quan loss của train và validation qua 15 epochs

- Cuối cùng, ta đánh giá mô hình GRU dựa trên 2 tiêu chí: độ chính xác accuracy và confusion matrix:
 - Đối với tiêu chí độ chính xác accuracy, mô hình đạt được 71.77%
 - Confusion matrix được mô tả như hình 43:



Hình 43: Confusion Matrix của mô hình GRU

- Lưu model: `model.save('model/LSTM_w2v.h5')`
- Thử nghiệm mô hình cho một câu message bất kì.



```
import time

message = ['delivery was hour late and my pizza was cold!']

seq = tokenizer.texts_to_sequences(message)
padded = pad_sequences(seq, maxlen=max_seq_len)

start_time = time.time()
pred = model.predict(padded)

print('Message: ' + str(message))
print('predicted: {} ( {:.2f} seconds)'.format(class_names[np.argmax(pred)], (time.time() - start_time)))
```

1/1 [=====] - 0s 233ms/step
 Message: ['delivery was hour late and my pizza was cold!']
 predicted: anger (0.28 seconds)

Hình 44: Thử nghiệm model GRU với message bất kì

- Đối với message = ‘delivery was hour late and my pizza was cold!’, kết quả là anger (0.28 giây) → Đúng với output mong muốn.

3.2.3. Mô hình pretrained BERT:

Đây là phần chính mà nhóm em nghiên cứu – transfer learning pretrained model BERT. Trước khi đi vào phần thực hành, nhóm em sẽ tìm hiểu cách hoạt động của BERT. [15] [16]

Như đã giới thiệu ở Chương 1, BERT sử dụng kiến trúc Encoder – Decoder của Transformer. Tại thời điểm công bố paper “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding⁴⁰”, BERT đã nhanh chóng trở nên “bá đạo” trong mảng NLP bởi những cải tiến chưa từng có ở những model trước đó (tăng độ chính xác, GLUE score,...). Vậy BERT có gì mà lại vượt trội hơn so với các mô hình khác? → Lý do chính là bởi nó có thêm context embedding vào các vector embedding. Context (ngữ cảnh) là một thứ rất quan trọng trong language. Với các ngữ cảnh khác nhau thì các từ trong câu được hiểu theo ý nghĩa hoàn toàn khác nhau, và các LM trước đây nếu bỏ qua ngữ cảnh của câu thì khó có thể đạt được chất lượng tốt (tiêu biểu như RNN). Nhóm em sẽ đưa ra 2 trường hợp cho thấy ngữ cảnh ảnh hưởng thế nào đến embedding:

Trường hợp 1: Embedding không ngữ cảnh. Ví dụ ta có 2 câu:

- Some apple are on the table.
- In 2001, Apple changed course with three announcements.

⁴⁰ <https://arxiv.org/abs/1810.04805>

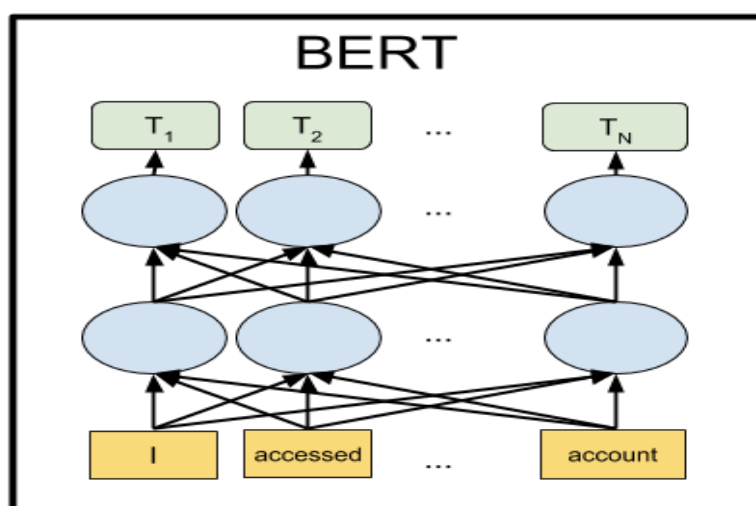
- Rõ ràng 2 từ **apple** ở đây mang nghĩa khác nhau, một mặt là trái táo, mặt khác là một công ty công nghệ. Nhưng với việc embedding không ngữ cảnh thì cả 2 từ này sẽ đều ánh xạ thành chung 1 vector word embedding → ảnh hưởng đến bài toán.

Trường hợp 2: Embedding có ngữ cảnh 1 chiều (tiêu biểu là RNN, chiều từ trái sang phải). Ví dụ về một bài toán dự đoán từ đã được masking:

- Câu gốc: “Hôm nay Nam đưa bạn gái đi chơi”
- Câu được mask: “Hôm nay Nam đưa **[mask]** đi chơi” (từ “bạn gái” được mask)
- Với yêu cầu thì mô hình cần đưa ra dự đoán là “bạn gái” vào mask. Nhưng do nó chỉ training 1 chiều từ trái sang phải, nên nó chỉ dựa vào những từ bên trái để đoán. Ở đây, các từ bên trái là “Hôm nay Nam đưa”, thì mô hình chỉ việc ráp các từ vào sao cho có nghĩa (“tiền”, “hàng”, “đồ vật”,...) với những từ ở trước, mà không quan tâm đến từ đằng sau (“đi chơi”).

→ Mô hình BERT sẽ giải quyết vấn đề này. Tên của mô hình cũng đã đề cập đến vấn đề này (bidirectional – 2 chiều). Tóm lại, BERT sẽ đưa ra dự đoán dựa trên ngữ cảnh của cả những từ ở bên trái lẫn bên phải. Vậy kiến trúc của BERT ra sao mà hay vậy?

Nhóm em đã cung cấp kiến trúc của Transformer ở hình 7, bao gồm Encoder và Decoder, và BERT chính là sử dụng kiến trúc ấy. Tuy nhiên BERT chỉ sử dụng phần Encoder thôi



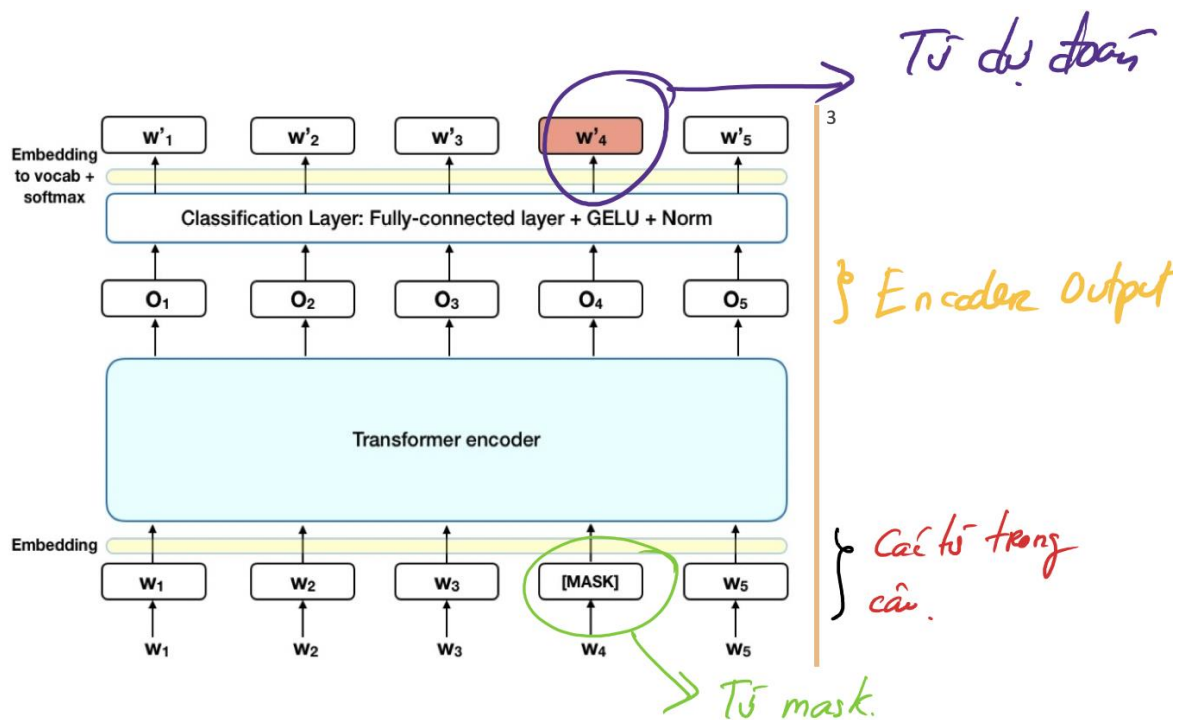
Hình 45: BERT sử dụng Encoder (Nguồn ⁴¹)

⁴¹ [2bd0ba1c4fb80fe4d771f555168c9ff0.png \(438×367\) \(habrastorage.org\)](#)

BERT được train đồng thời 2 task: đầu tiên là Masked LM (Dự đoán từ thiếu trong câu) và Next Sentence Prediction (dự đoán câu tiếp theo của câu hiện tại). Hai task này được train đồng thời và loss tổng sẽ kết hợp loss của cả 2, và model sẽ cố gắng minimize loss tổng. Chi tiết như sau:

Masked LM: với task này, ta sẽ mask đi tầm 15% số từ trong câu và đưa vào model. Nhiệm vụ là sẽ train để model predict ra các từ bị che đó dựa vào các từ còn lại, cụ thể:

- Thêm 1 lớp classification lên trên encoder output
- Đưa các vector trong encoder output về vector bằng kích thước của vocab, sau đó softmax để chọn ra từ tương ứng tại mỗi vị trí trong câu
- Loss sẽ được tính tại vị trí masked và bỏ qua các vị trí khác.



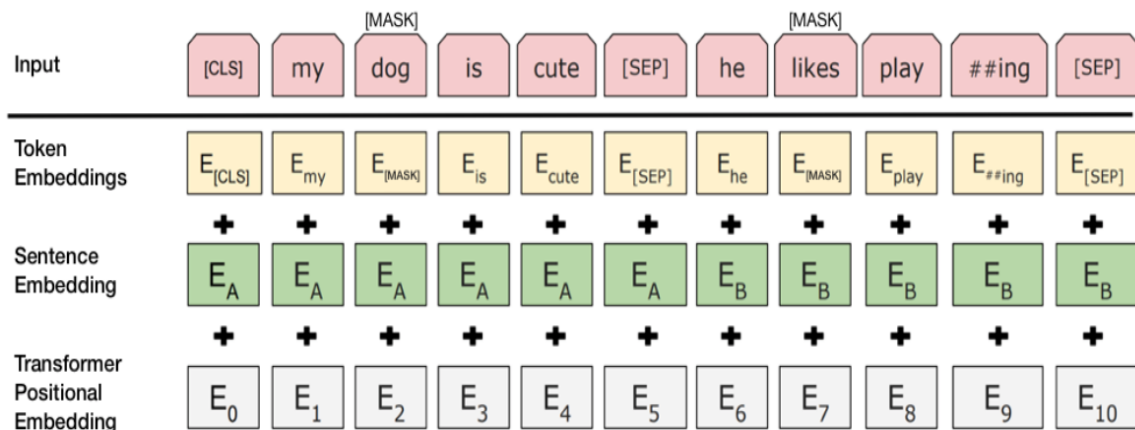
Hình 46: Quá trình train mô hình cho task masked LM (Nguồn ⁴²)

Task thứ 2 là NSP, model sẽ được feed cho một cặp câu và nhiệm vụ là dự đoán xem câu thứ 2 có đúng là câu đi sau câu đầu tiên không (output là 1 nếu đúng, và 0 nếu ngược lại). Trong quá trình train, chọn 50% mẫu là Positive (output là 1), 50% là Negative (các câu được ghép linh tinh, output là 0). Cụ thể như sau:

- Đầu tiên, ghép 2 câu vào nhau và thêm 1 số token đặc biệt để phân tách câu. Token [CLS] thêm vào đầu câu thứ nhất, token [SEP] thêm vào cuối mỗi câu.

⁴² <https://github.com/thangnch/photo2/blob/main/bert01.png?raw=true>

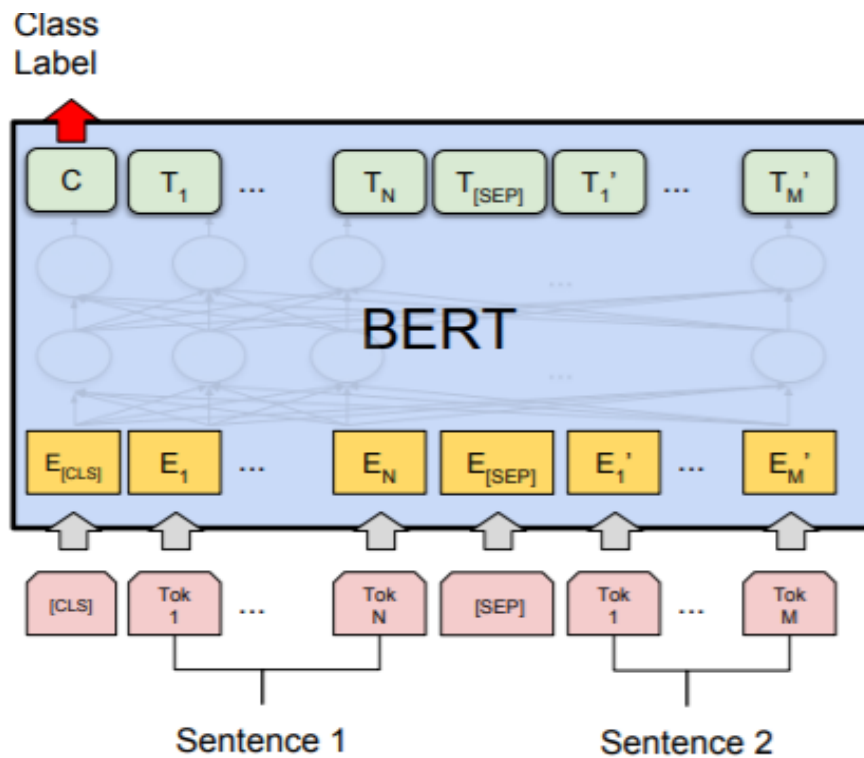
Ví dụ ghép 2 câu “Hôm nay em đi học” và “Học ở trường rất hay” thì sẽ thành “[CLS] Hôm nay em đi học [SEP] Học ở trường rất vui [SEP]”.



Hình 47: Các bước tạo input (Nguồn ⁴³)

- Tiếp theo, mỗi token trong câu sẽ được cộng thêm 1 vector Sentence Embedding (đơn giản là đánh dấu xem từ đó thuộc câu thứ nhất hay thứ hai). Ví dụ nếu thuộc câu thứ 1 thì cộng thêm 1 vector toàn 0 (size bằng word embedding), còn nếu thuộc câu thứ 2 thì cộng thêm một vector toàn số 1.
- Sau đó các từ trong câu đã ghép sẽ được thêm vector Positional Encoding vào để đánh dấu vị trí từng từ trong câu đã ghép.
- Tiếp tục đưa chuỗi sau bước 3 vào network.
- Lấy encoder output tại vị trí token [CLS] được transform sang một vector có 2 phần từ [c1 c2].
- Cuối cùng là tính softmax trên vector đó và output ra probability của 2 class: Đi sau và Không đi sau (lấy argmax để thể hiện câu thứ 2 là đi sau hay không đi sau câu đầu tiên).

⁴³ https://miro.medium.com/v2/resize:fit:1400/0*m_kXt3uqZH9e7H4w.png



Hình 48: Mô tả cách lấy output (Nguồn ⁴⁴)

Đó là các bước để tạo ra pretrained model BERT.

Mô hình pretrained BERT có hai dạng tổng quát:

- Mô hình BERT Base: Kiến trúc mạng NN chứa 12 lớp, 110 triệu tham số.
- Mô hình BERT Large: kiến trúc mạng NN chứa 24 lớp, 340 triệu tham số.

Cả hai đều được huấn luyện từ BooksCorpus⁴⁵ với 800 triệu từ, và một phiên bản của Wikipedia tiếng Anh⁴⁶ với 2,500 triệu từ.

Đó chính là giới thiệu tổng quát về BERT. Tiếp theo là phần thực hành [17] của nhóm em (một số bước đơn giản nhóm em sẽ không giải thích chi tiết mà chỉ đề cập đến trong quá trình thực hành)

- Import các thư viện cần thiết.
- Đọc tập train và test.

```
train = pd.read_csv("data/train.csv")
test = pd.read_csv("data/test.csv")
```

Hình 49: Đọc tập train và test

⁴⁴ https://cdn-contents.anymindgroup.com/corporate/wp-uploads/2021/09/22102605/blog_BERT.png

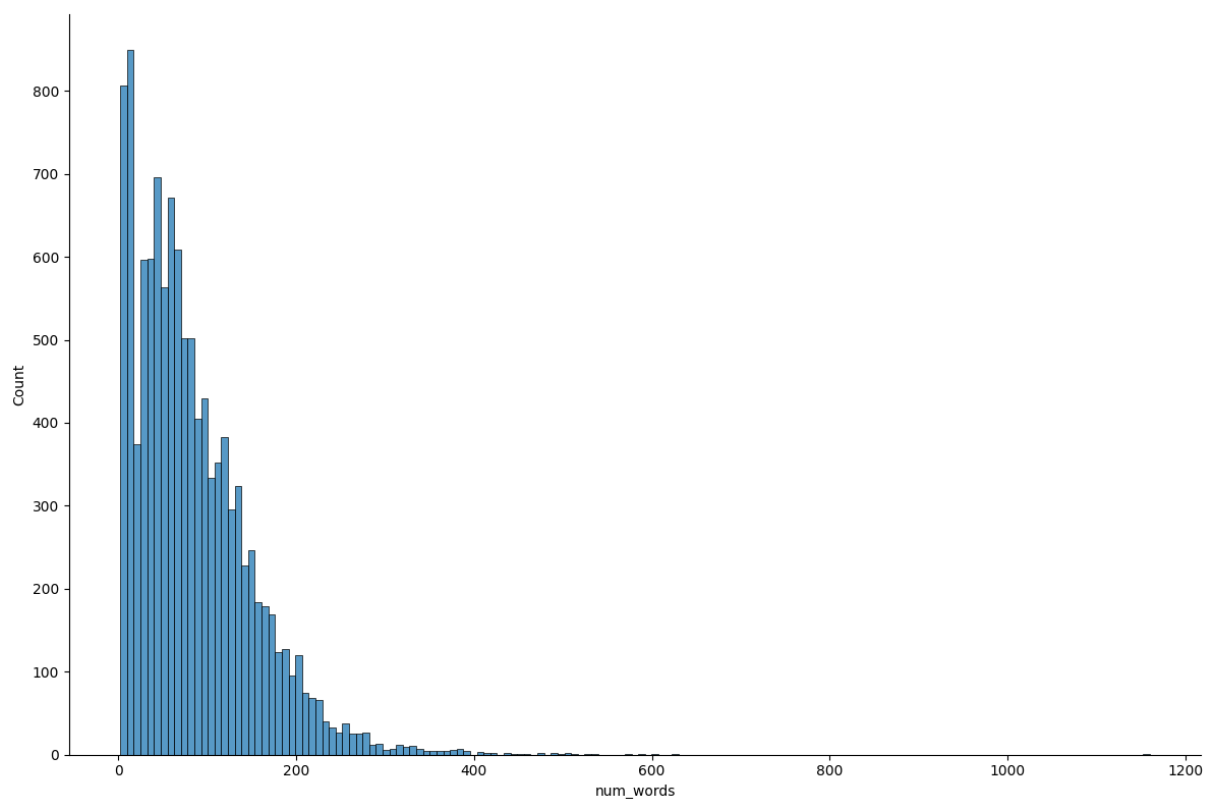
⁴⁵ <https://vi.wikipedia.org/w/index.php?title=BooksCorpus&action=edit&redlink=1>

⁴⁶ https://vi.wikipedia.org/wiki/Wikipedia_t%C3%BAi_m%C3%A0n_h%E1%BB%80ng_Anh

- Gộp tập train và test thành dataframe (df) để tiện xử lý và kiểm tra số lượng từ trong câu

	Text	Emotion	num_words
0	There are tons of other paintings that I think...	neutral	59
1	Yet the dog had grown old and less capable , a...	sadness	169
2	When I get into the tube or the train without ...	fear	68
3	This last may be a source of considerable disq...	fear	173
4	She disliked the intimacy he showed towards so...	anger	151

Hình 50: Mô tả dataframe



Hình 51: Biểu đồ phân bố số lượng từ trong các văn bản trong dataset

- Tạo map encoding các emotion label.

```

▶ encoded_labels = {
    'joy': 0,
    'sadness': 1,
    'fear': 2,
    'anger': 3,
    'neutral': 4
}

```

Hình 51: Map encoding emotion labels

- Dựa trên map encoding, thêm 1 cột label gồm các giá trị tương ứng với emotion.

	Text	Emotion	num_words	Label
3369	When we rearranged furniture in our flat and g...	anger	66	3
5409	I will .	neutral	8	4
151	I get angry when people disbelieve me or misun...	anger	58	3
1442	It happened in a tram: some older people start...	anger	139	3
1006	If a close relative's life is in danger.	fear	40	2

Hình 52: Thêm cột label tương ứng

- Sử dụng thư viện Transformers của Hugging Face để tạo tokenizer cho model bert-base-uncased (Tokenizer có nhiệm vụ chuyển đổi các câu văn bản thành chuỗi token được Bert hiểu, bao gồm việc chia các từ thành các token, thêm token đặc biệt cho đầu vào, và thêm các token padding nếu cần thiết) và để tải pretrained model bert-base-uncased.

```

tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
bert_model = TFBERTModel.from_pretrained("bert-base-uncased")

tokenizer_config.json: 100% 48.0/48.0 [00:00<00:00, 1.91kB/s]
config.json: 100% 570/570 [00:00<00:00, 28.9kB/s]
vocab.txt: 100% 232k/232k [00:00<00:00, 1.39MB/s]
tokenizer.json: 100% 466k/466k [00:00<00:00, 1.89MB/s]
model.safetensors: 100% 440M/440M [00:01<00:00, 229MB/s]

```

Hình 53: Tải tokenizer và pretrained model bert

- Tiến hành quá trình text tokenization:

```

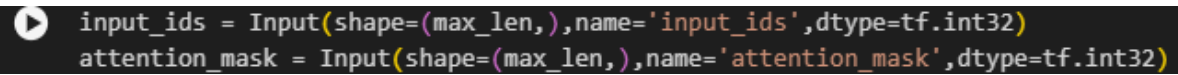
X_train = tokenizer(text=train_data.Text.tolist(),
                    add_special_tokens=True,
                    return_tensors='tf',
                    max_length=max_len,
                    padding='max_length',
                    truncation=True,
                    return_token_type_ids=False,
                    return_attention_mask=True,
                    verbose=True
                    )

X_test = tokenizer(text=test_data.Text.tolist(),
                   add_special_tokens=True,
                   return_tensors='tf',
                   max_length=max_len,
                   padding='max_length',
                   truncation=True,
                   return_token_type_ids=False,
                   return_attention_mask=True,
                   verbose=True
                   )

```

Hình 54: Text tokenization

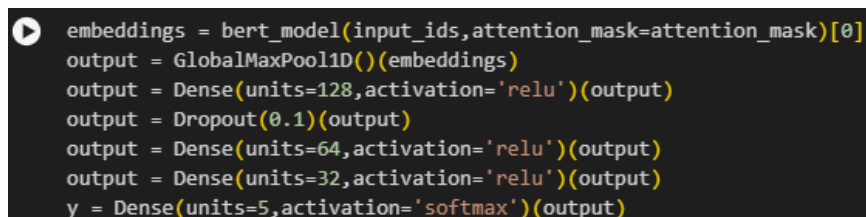
- `text=train_data.Text.tolist()` và `text=test_data.Text.tolist()`: Chuyển đổi cột Text của dataframe thành danh sách các chuỗi văn bản để tokenizer có thể xử lý.
- `add_special_tokens=True`: Thêm các token đặc biệt ([CLS] và [SEP]) cần thiết cho mô hình BERT.
- `return_tensors='tf'`: Trả về kết quả dưới dạng tensor của TensorFlow, thích hợp cho việc sử dụng với các mô hình TensorFlow.
- `max_length=max_len`: Thiết lập độ dài tối đa của chuỗi token (ở đây là 350).
- `padding='max_length'`: Đệm thêm các token đặc biệt để các chuỗi có độ dài bằng max_length. Điều này đảm bảo rằng tất cả các chuỗi đầu vào có cùng độ dài, cần thiết cho việc xử lý bằng BERT.
- `truncation=True`: Cắt bớt các chuỗi dài hơn max_length.
- `return_token_type_ids=False`: Không trả về các token type ids. Token type ids thường được sử dụng trong các tác vụ như phân loại câu đôi (sentence pair classification), nhưng không cần thiết trong bài toán này.
- `return_attention_mask=True`: Trả về attention masks, là một tensor chỉ ra các vị trí của các token thực sự (không phải padding) trong chuỗi. Điều này giúp mô hình BERT phân biệt giữa các token thực và các token được đệm.
- `verbose=True`: Bật chế độ ghi nhật ký chi tiết để hiển thị thêm thông tin trong quá trình mã hóa.
- Xây dựng kiến trúc cho mô hình:
 - Định nghĩa đầu vào cho mô hình:
 - **input_ids**: một tensor đầu vào có hình dạng (max_len,), nơi max_len = 350 là độ dài tối đa của chuỗi token. Tensor này chứa các token ID của văn bản.
 - **attention_mask**: một tensor đầu vào khác có cùng hình dạng (max_len,), tensor này chứa các giá trị 1 cho các token thực sự và 0 cho các token padding.



```
input_ids = Input(shape=(max_len,),name='input_ids',dtype=tf.int32)
attention_mask = Input(shape=(max_len,),name='attention_mask',dtype=tf.int32)
```

Hình 55: Đầu vào cho mô hình

- Tạo mô hình mạng neural cho tác vụ phân tích cảm xúc văn bản
 - ***embeddings = bert_model(input_ids, attention_mask = attention_mask)[0]***: đầu tiên, sử dụng mô hình bert để mã hóa các token đầu vào (*input_ids*) và áp dụng attention mask (*attention_mask*), kết quả trả về là embeddings của các token, ta chỉ quan tâm đến phần đầu tiên của kết quả.
 - ***output = GlobalMaxPool1D()(embeddings)***: sau khi có được embeddings của các token, ta sử dụng lớp GlobalMaxPool1D để trích xuất đặc trưng toàn cục từ các embedding, giúp giảm chiều dữ liệu và tóm tắt thông tin quan trọng từ toàn bộ văn bản.
 - ***output = Dense(units=128, activation='relu')(output)***: lớp Dense này có 128 nôt với activation function là ReLU, tạo ra một biểu diễn phi tuyến của đặc trưng đã được trích xuất từ văn bản.
 - ***output = Dropout(0.1)(output)***: dropout để ngẫu nhiên bỏ qua 10% các neuron trong quá trình huấn luyện, giúp tránh overfitting.
 - ***output = Dense(units=64, activation='relu')(output)***: lớp Dense tiếp theo có 64 nôt và hàm kích hoạt là ReLU, tương tự như lớp Dense trước đó.
 - ***output = Dense(units=32, activation='relu')(output)***: lớp Dense khác với 32 nôt và hàm kích hoạt ReLU.
 - ***y = Dense(units=5, activation='softmax')(output)***: cuối cùng, là một lớp Dense với 5 nôt, tương ứng 5 cảm xúc và hàm kích hoạt softmax. Lớp này sẽ đưa ra xác suất dự đoán cho mỗi lớp cảm xúc.



```
embeddings = bert_model(input_ids,attention_mask=attention_mask)[0]
output = GlobalMaxPool1D()(embeddings)
output = Dense(units=128,activation='relu')(output)
output = Dropout(0.1)(output)
output = Dense(units=64,activation='relu')(output)
output = Dense(units=32,activation='relu')(output)
y = Dense(units=5,activation='softmax')(output)
```

Hình 56: Quá trình xây dựng neural cho bài toán

- Tạo mô hình:

```
model = Model(inputs=[input_ids,attention_mask],outputs=y)
model.layers[2].trainable = True
```

Hình 57: Tạo mô hình

- Sử dụng thuật toán tối ưu Adam⁴⁷ và loss function cho bài toán phân loại nhiều lớp là categorical crossentropy.

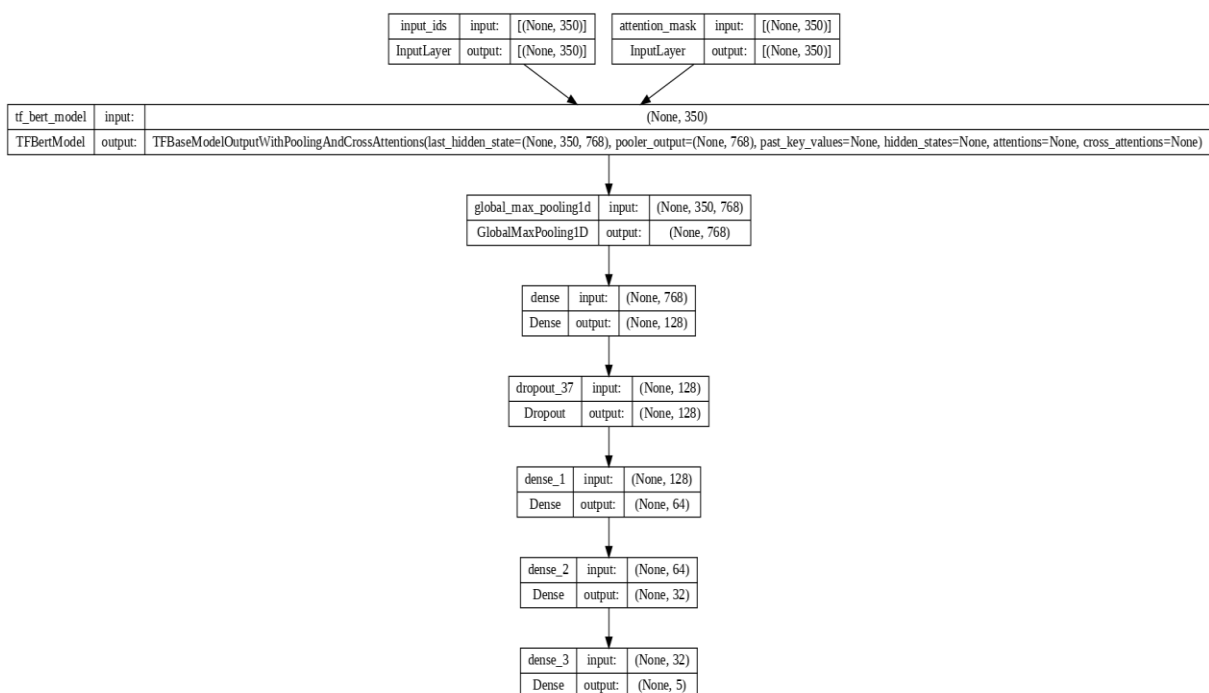
```
from tensorflow.keras.optimizers import legacy

optimizer = legacy.Adam(
    learning_rate=5e-5,
    epsilon=1e-8,
    decay=0.01,
    clipnorm=1.0)

model.compile(
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
    optimizer=optimizer,
    metrics=[tf.keras.metrics.CategoricalAccuracy(name='balanced_accuracy')])
```

Hình 58: Biên dịch mô hình

- Bảng tóm tắt lại mô hình



Hình 59: Model summary

- Huấn luyện mô hình: sử dụng ‘fit’ để huấn luyện mô hình trên dữ liệu đã chuẩn bị, sử dụng validation data để theo dõi hiệu suất.

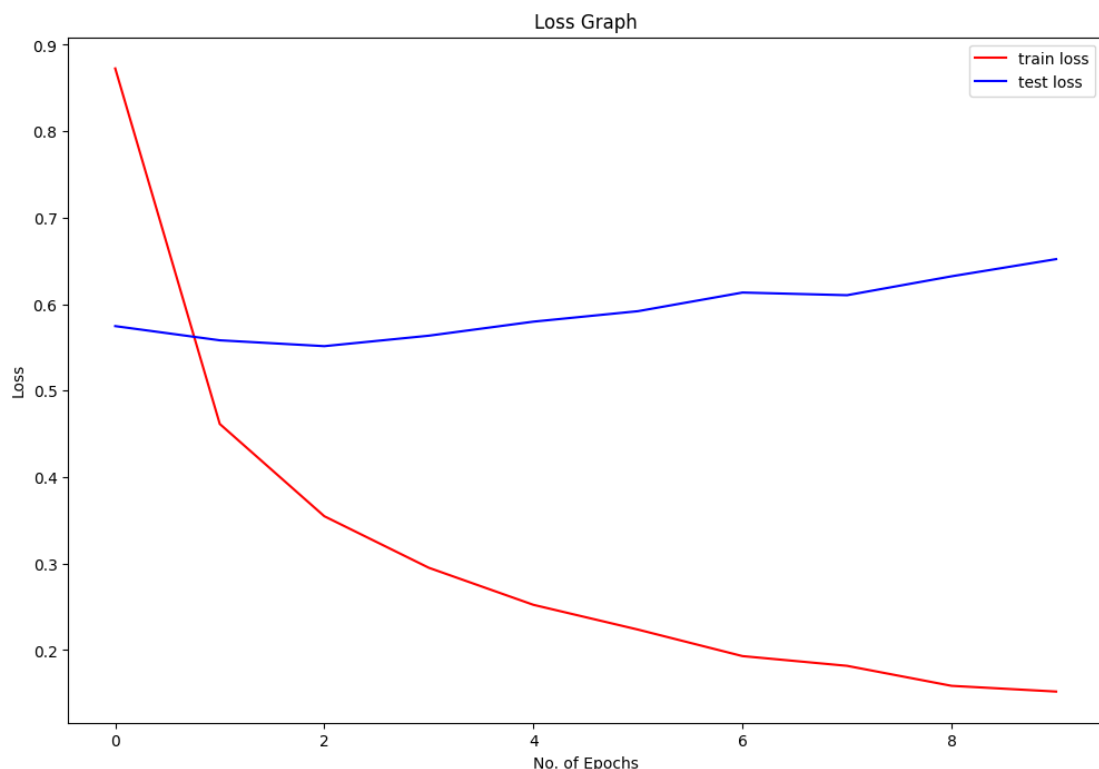
⁴⁷ [What is Adam Optimizer? - Analytics Vidhya](#)

```
[ ] r = model.fit(x={'input_ids': X_train['input_ids'], 'attention_mask': X_train['attention_mask']},
                y=to_categorical(train_data.Label),
                epochs=10,
                batch_size=16,
                validation_data=({'input_ids': X_test['input_ids'], 'attention_mask': X_test['attention_mask']}, to_categorical(test_data.Label))
                )
```

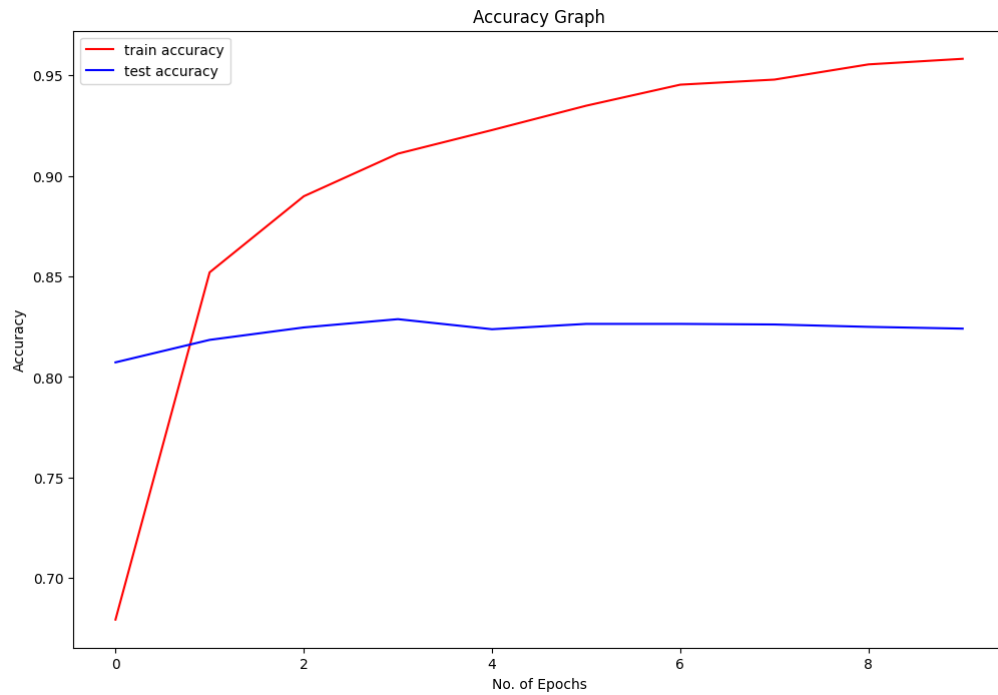
Epoch 1/10
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model/bert/pooler/dense/kernel:0', 'tf_bert_model/bert/pooler/dense/bias:0'] when mir
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model/bert/pooler/dense/kernel:0', 'tf_bert_model/bert/pooler/dense/bias:0'] when mir
496/496 [=====] - 785s 2s/step - loss: 0.8726 - balanced_accuracy: 0.6794 - val_loss: 0.5746 - val_balanced_accuracy: 0.8073
Epoch 2/10
496/496 [=====] - 703s 1s/step - loss: 0.4615 - balanced_accuracy: 0.8520 - val_loss: 0.5582 - val_balanced_accuracy: 0.8185
Epoch 3/10
496/496 [=====] - 703s 1s/step - loss: 0.3548 - balanced_accuracy: 0.8899 - val_loss: 0.5515 - val_balanced_accuracy: 0.8247
Epoch 4/10
496/496 [=====] - 756s 2s/step - loss: 0.2952 - balanced_accuracy: 0.9111 - val_loss: 0.5636 - val_balanced_accuracy: 0.8288
Epoch 5/10
496/496 [=====] - 756s 2s/step - loss: 0.2523 - balanced_accuracy: 0.9228 - val_loss: 0.5798 - val_balanced_accuracy: 0.8238
Epoch 6/10
496/496 [=====] - 756s 2s/step - loss: 0.2237 - balanced_accuracy: 0.9349 - val_loss: 0.5919 - val_balanced_accuracy: 0.8264
Epoch 7/10
496/496 [=====] - 756s 2s/step - loss: 0.1930 - balanced_accuracy: 0.9454 - val_loss: 0.6135 - val_balanced_accuracy: 0.8264
Epoch 8/10
496/496 [=====] - 757s 2s/step - loss: 0.1818 - balanced_accuracy: 0.9479 - val_loss: 0.6104 - val_balanced_accuracy: 0.8261
Epoch 9/10
496/496 [=====] - 756s 2s/step - loss: 0.1586 - balanced_accuracy: 0.9555 - val_loss: 0.6323 - val_balanced_accuracy: 0.8249
Epoch 10/10
496/496 [=====] - 704s 1s/step - loss: 0.1520 - balanced_accuracy: 0.9582 - val_loss: 0.6521 - val_balanced_accuracy: 0.8241

Hình 60: Quá trình huấn luyện / fine tuning bert

- Sử dụng GPU T4 của Google Colab để train. Với 10 epoch và batch size 16, thời gian để train là 2h03m, trung bình mỗi epoch mất 12 phút.
- Phân tích quá trình train dựa vào 2 chỉ số là loss và accuracy (hình 61,62):

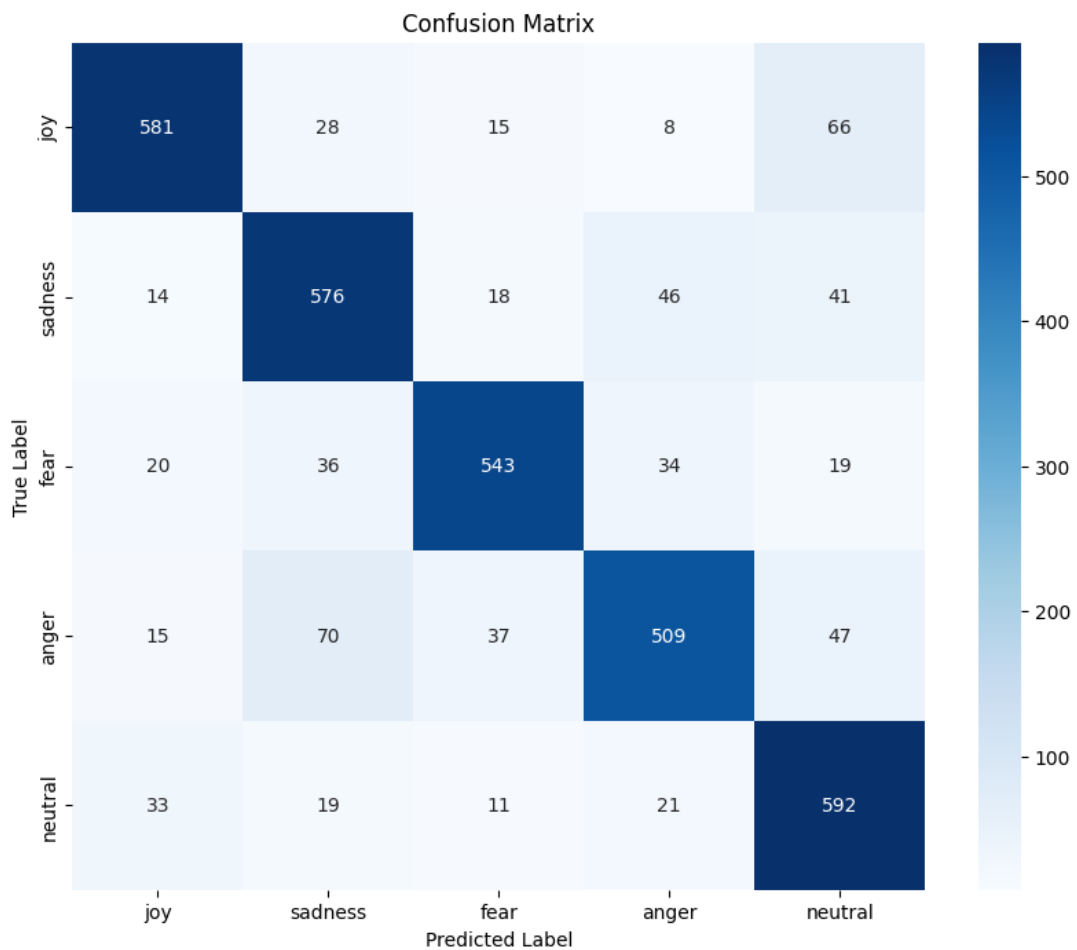


Hình 61: Biến thiên của Loss trong quá trình training



Hình 62: Biến thiên của accuracy trong quá trình training

- Biểu đồ heatmap cho confusion matrix:



Hình 63: Confusion matrix

- Lưu lại kết quả: `model.save("emotion_detector", save_format='tf')`
- Dự đoán cho 1 message bất kì:

```
message = "I feel really happy today"
predicted_emotion = predict_emotion(message)
print("Predicted emotion:", predicted_emotion)
```

1/1 [=====] - 0s 98ms/step
Predicted emotion: joy

Hình 64: Kết quả dự đoán message

- Đối với message = 'I feel really happy today', kết quả là joy (0.98 giây) → Đúng với output mong muốn.

4. Kết quả:

Bảng Benchmark so sánh các mô hình:

Model		Accuracy
Traditional Model	Naïve Bayes	67.02%
	Random Forest	62.98%
	Logistic Regression	69.35%
	SVM	72.71%
LSM / GRU + w2v_wiki		71.77%
Finetuned pretrained BERT model		82.41%

CHƯƠNG 3: KẾT LUẬN

1. Những kết quả đạt được

Thông qua quá trình nghiên cứu NLP áp dụng cho bài toán phân loại cảm xúc văn bản, nhóm em không chỉ có cơ hội thực nghiệm lại các mô hình học máy cơ bản, mà còn được tiếp cận được những mô hình học sâu, chuyên dụng trong mảng xử lý ngôn ngữ tự nhiên. Hai mô hình mà chúng em dành thời gian tìm hiểu nhiều nhất là LSTM / GRU, và BERT. Nhóm em đã hiểu hơn về cách hoạt động của LSTM / GRU, kiến trúc của mô hình, và các hạn chế mà mô hình đang gặp phải. Và đặc biệt đối với mô hình BERT, nhóm đã hiểu được kiến trúc encode của BERT, cùng với đó là biết cách fine tune pretrained model dựa theo yêu cầu của bài toán, mà cụ thể ở đây là phân loại cảm xúc.

2. Những hạn chế

Nhóm còn nhiều hạn chế về mặt kiến thức, nên chưa tìm hiểu kỹ để can thiệp quá sâu vào mô hình. Đồng thời tiếng Anh cũng là một bước cản, vì những tài liệu của mô hình này đa số được viết bằng tiếng Anh.

TÀI LIỆU THAM KHẢO

- [1] <https://www.deeplearning.ai/resources/natural-language-processing/>
- [2] https://en.wikipedia.org/wiki/Natural_language_processing?ref=200lab.io
- [3] <https://openai.com/index/hello-gpt-4o/>
- [4] <https://arxiv.org/abs/1511.08458>
- [5] <https://aclanthology.org/D14-1181/>
- [6] <https://arxiv.org/abs/1808.03314>
- [7] <https://arxiv.org/pdf/1706.03762>
- [8] <https://arxiv.org/abs/1810.04805>
- [9] https://www.site.uottawa.ca/~diana/resources/emotion_stimulus_data/
- [10] http://www.affective-sciences.org/index.php/download_file/view/395/296/
- [11] <http://yanran.li/dailydialog.html>
- [12] <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>
- [13] <https://youtu.be/8HyCNIVRbSU>
- [14] <https://arxiv.org/abs/1808.03314>
- [15] <https://www.miai.vn/2020/12/14/bert-series-chuong-1-bert-la-cai-chi-chi/>
- [16] <https://phamdinhhkhanh.github.io/2020/05/23/BERTModel.html>
- [17] <https://arxiv.org/pdf/1905.05583>

Học kỳ II, năm học 2022 - 2023

(Ký và ghi rõ họ và tên)