TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH

BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI: ỨNG DỤNG MẠNG NƠ-RON TÍCH CHẬP KẾT HỢP BiLSTM ĐỂ ĐÁNH GIÁ BÌNH LUẬN CỦA NGƯỜI DÙNG TRÊN MẠNG XÃ HỘI

Giảng viên hướng dẫn: ThS. TRẦN PHONG NHÃ

Sinh viên thực hiện: BÙI NHẬT HUY

Lớp: CQ.61.CNTT

Khoá: 61

TP. Hồ Chí Minh, năm 2024

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP

BỘ MÔN: CÔNG NGHỆ THÔNG TIN

-------***-------

Mã sinh viên: 6151071052 **Họ tên SV:** BÙI NHẬT HUY
Khóa: 61 **Lớp:** CQ. 61.CNTT

1. Tên đề tài

ỨNG DỤNG MẠNG NƠ-RON TÍCH CHẬP KẾT HỢP BiLSTM ĐỂ ĐÁNH GIÁ BÌNH LUẬN CỦA NGƯỜI DÙNG TRÊN MẠNG XÃ HỘI

2. Mục tiêu

Xây dựng mô hình Phân Loại Cảm Xúc để nhận diện bình luận tiêu cực của người dùng. Từ đó ứng dụng tích hợp vào xây dựng trang web Mạng Xã Hội.

3. Nội dung thực hiện

- Tìm hiểu sơ bộ về AI, Machine Learning, Deep Learning, NLP
- Nghiên cứu về mô hình mạng neural Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory (BiLSTM)
- Nghiên cứu các bài toán về Sentiment Analysis
- Áp dụng kiến thức: ứng dụng các mô hình mạng neural vào phân tích và đưa ra dự đoán về Cảm Xúc Văn Bản

4. Công nghệ, công cụ và ngôn ngữ lập trình

- Công nghệ sử dụng: Tensorflow, FastAPI
- Công cụ lập trình: Google Colab, Visual Studio Code
- Ngôn ngữ lập trình: Python, JavaScript

5. Các kết quả chính dự kiến

- Phân loại được các bình luận tiêu cực
- Tích hợp được vào trang web Mạng Xã Hội

6. Giảng viên và cán bộ hướng dẫn

Họ tên: ThS. TRẦN PHONG NHÃ

Đơn vị công tác: Trường Đại học Giao thông Vận tải Phân hiệu tại TP. Hồ Chí Minh

Điện thoại: 0906 761 014 Email: tpnha@utc2.edu.vn

Ngày…..tháng…..năm 2024	**Đã giao nhiệm vụ TKTN**
Trưởng BM Công nghệ thông tin	**Giảng viên hướng dẫn**
Trần Phong Nhã	**Trần Phong Nhã**

Đã nhận nhiệm vụ TKTN

Sinh viên: Bùi Nhật Huy Ký tên:

Điện thoại: 0385 827 933 Email: 6151071052@st.utc2.edu.vn

LỜI CẢM ƠN

Trước hết, em xin bày tỏ lòng biết ơn sâu sắc đến Ban Giám Hiệu, các thầy cô giáo trong **Bộ môn Công Nghệ Thông Tin - Trường Đại học Giao thông Vận tải phân hiệu tại TP. Hồ Chí Mình** đã tạo điều kiện thuận lợi và cung cấp cho em những kiến thức, kỹ năng cần thiết trong suốt quá trình học tập và thực hiện đồ án tốt nghiệp.

Đặc biệt em xin chân thành cảm ơn thầy **Trần Phong Nhã**, người đã tận tình hướng dẫn, đóng góp ý kiến quý báu và giúp đỡ em trong suốt quá trình thực hiện và hoàn thiện đồ án này. Sự nhiệt tình, tâm huyết và những kinh nghiệm quý báu của thầy đã giúp em,vượt qua những khó khăn và hoàn thành đồ án một cách tốt nhất.

Em cũng xin gửi lời cảm ơn đến tất cả các bạn bè đã luôn động viên, hỗ trợ và cùng em chia sẻ những kinh nghiệm, ý tưởng trong suốt quá trình thực hiện đồ án.

Vì thời gian làm đề tài có hạn cũng như hiểu biết còn hạn chế, em cũng đã nỗ lực hết sức để hoàn thành đề tài một cách tốt nhất, nhưng chắc chắn vẫn sẽ có những thiếu sót không thể tránh khỏi. Em kính mong nhận được sự thông cảm và những ý kiến đóng góp chân thành từ quý thầy cô.

Sau cùng, em xin kính chúc Quý Thầy Cô trong Bộ môn Công nghệ thông tin hạnh phúc và thành công hơn nữa trong công việc cũng như trong cuộc sống.

Em xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày …… tháng ……. năm 2024

Sinh viên thực hiện

Bùi Nhật Huy

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

Tp. Hồ Chí Minh, ngày … tháng …. năm 2024

Giảng viên hướng dẫn

Trần Phong Nhã

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG VIẾT TẮT

STT	Viết tắt	Diễn giải	Ý nghĩa
1	AL	Artificial Intelligence - Trí tuệ nhân tạo	Khoa học máy tính tạo ra hệ thống có khả năng thực hiện các nhiệm vụ cần trí tuệ con người.
2	NLP	Natural Language Processing - Xử lý ngôn ngữ tự nhiên	Tương tác giữa máy tính và ngôn ngữ con người, giúp máy hiểu và phân tích ngôn ngữ tự nhiên.
3	ML	Machine Learning - Học máy	Máy tính học từ dữ liệu để cải thiện hiệu suất mà không cần lập trình cụ thể.
4	BOW	Bag of Words - Túi từ	Biểu diễn văn bản bằng cách đếm tần suất từ xuất hiện, không quan tâm đến trật tự từ.
5	DL	Deep Learning - Học sâu	Học máy với nhiều tầng mạng nơ-ron, giúp nhận dạng mẫu và xử lý dữ liệu phức tạp.
6	RNN	Recurrent Neural Network - Mạng nơ-ron hồi quy	Mạng nơ-ron xử lý dữ liệu tuần tự như văn bản hoặc chuỗi thời gian.
7	ANN	Artificial Neural Network - Mạng nơ-ron nhân tạo	Hệ thống lấy cảm hứng từ não bộ, học từ dữ liệu để phân loại và phát hiện mẫu.
8	CNN	Convolutional Neural Network - Mạng nơ-ron tích chập	Mạng nơ-ron dùng trong xử lý hình ảnh và nhận dạng mẫu qua các lớp tích chập.
9	LSTM	Long Short-Term Memory - Bộ nhớ dài ngắn hạn	Loại RNN ghi nhớ thông tin dài hạn, dùng trong dịch máy và phân tích chuỗi thời gian.
10	BiLSTM	Bidirectional Long Short-Term Memory - Bộ nhớ dài	LSTM xử lý dữ liệu theo cả hai chiều, tăng cường ngữ cảnh trong

		ngắn hạn hai chiều	nhận dạng thực thể và dịch máy.
11	API	Application Programming Interface - Giao diện lập trình ứng dụng	Giao thức và công cụ giúp các ứng dụng giao tiếp và sử dụng chức năng của nhau.
12	ReLU	Rectified Linear Unit – Hàm kích hoạt phổ biến trong mạng neural	Giúp giảm bớt vấn đề Vanishing Gradient, tính toán hiệu quả và đơn giản hoá mô hình.

TỔNG QUAN

1. Lý do chọn đề tài

Cùng với sự phát triển mạnh mẽ của công nghệ thông tin và Internet, các diễn đàn mạng xã hội phổ biến ngày nay như: Facebook, Instagram, Linkedin, Twitter, Zalo, Tiktok, Snopchat, … đều là các nền tảng rất thông dụng với người dùng và chúng hoạt động tiếp diễn hằng ngày. Thông qua các công cụ ứng dụng chúng như một phương tiện hữu ích giúp người dùng bày tỏ cảm xúc thông qua các trang mạng một cách dễ dàng hơn, cũng như chia sẻ những trải nghiệm, bình luận và đánh giá trong quá trình trải nghiệm.

Dễ thấy rằng việc thể hiện cảm xúc là nhu cầu cơ bản của con người và chúng ta sử dụng ngôn ngữ không chỉ để truyền đạt suy nghĩ mà còn có cả cảm xúc của chúng ta. Cảm xúc quyết định chất lượng cuộc sống của chúng ta, và chúng ta tổ chức cuộc sống của mình để tối đa hóa trải nghiệm của cảm xúc tích cực và giảm thiểu trải nghiệm của cảm xúc tiêu cực.

Ngoài biểu hiện trên khuôn mặt, nhiều nguồn thông tin khác nhau có thể được sử dụng để phân tích cảm xúc vì nhận biết cảm xúc đã nổi lên như một lĩnh vực nghiên cứu quan trọng. Và trong những năm gần đây, nhận dạng cảm xúc trong văn bản đã trở nên phổ biến hơn do những ứng dụng tiềm năng to lớn của nó trong tiếp thị, bảo mật, tâm lý học, tương tác giữa con người với máy tính, trí tuệ nhân tạo, ...v.v.

Trong nghiên cứu này, em tập trung vào vấn đề nhận biết cảm xúc đối với bình luận của người Việt trên mạng xã hội. Cụ thể hơn, đầu vào của bài toán là một bình luận bằng tiếng Việt từ mạng xã hội và đầu ra là cảm xúc dự đoán của bình luận đó được gán nhãn: tích cực, tiêu cực hay trung lập

Chính vì vậy để có thể thấu hiểu cảm xúc người dùng thông qua ý kiến tích cực hay tiêu cực qua quá trình trải nghiệm là một trong những vấn đề quan trọng. Giải pháp cho vấn đề này, nghiên cứu đề xuất phương pháp khai thác ý kiến và phân tích cảm xúc người dùng thông qua việc thu thập tập dữ liệu là ý kiến bình luận của khách hàng trên các website trực tuyến. Sau đó, tiến hành thực nghiệm bằng phương pháp học sâu để khai phá ý kiến từ bình luận dạng văn bản của người dùng và trực quan hóa kết quả hỗ trợ ra quyết định. Kết quả thực nghiệm cho thấy độ chính xác hơn 67% của phương pháp đề xuất và kết quả khai thác được tập thông

tin, tri thức tiềm ẩn có giá trị từ tập ngữ liệu nhằm giúp việc nghiên cứu trở nên trực quan và dễ dàng cải thiện chiến lược.

Hiện nay, bài toán phân tích ý kiến cảm xúc theo người dùng được quan tâm ở rất nhiều lĩnh vực khác nhau, từ giáo dục đến khảo sát ý kiến xã hội và hơn hết là lĩnh vực dịch vụ/kinh doanh. Hầu hết các bộ ngữ liệu cũng như các thuật toán được xây dựng và thử nghiệm trên nhiều ngôn ngữ khác nhau như tiếng Anh, tiếng Trung Quốc...v.v. Tuy nhiên đối với tiếng Việt, nhằm trong phạm vi của bài báo cáo, giới hạn của em sẽ chỉ nằm ở mỗi khía cạnh cấp độ câu văn.

2. Mục tiêu và nhiệm vụ của đồ án

Xây dựng mô hình Phân Loại Cảm Xúc Văn Bản để nhận diện bình luận các bình luận tích cực, tiêu cực và trung lập. Từ đó ứng dụng vào xây dựng trang web mạng xã hội.

3. Bố cục đồ án

TỔNG QUAN

CHƯƠNG 1: CƠ SỔ LÝ THUYẾT

CHƯƠNG 2: XÂY DỰNG MÔ HÌNH PHÂN LOẠI CẢM XÚC NGƯỜI DÙNG TRÊN MẠNG XÃ HỘI

CHƯƠNG 3: PHÁT TRIỂN MÔ HÌNH VÀ ÁP DỤNG VÀO TRANG WEB MẠNG XÃ HỘI

KẾT LUẬN VÀ KIẾN NGHỊ

TÀI LIỆU THAM KHẢO

CHƯƠNG I. CƠ SỞ LÝ THUYẾT

1.1. Tổng quan về AI

1.1.1. Sơ lược về AI

Trí tuệ nhân tạo (AI) trong khoa học máy tính là trí thông minh được thể hiện bằng máy móc, trái ngược với trí thông minh tự nhiên của con người. AI thường được dùng để mô tả máy móc có khả năng bắt chước các chức năng nhận thức của con người như học tập và giải quyết vấn đề. Khi máy móc phát triển, các nhiệm vụ đòi hỏi trí thông minh thường bị loại khỏi định nghĩa AI, hiện tượng này gọi là hiệu ứng AI.

Các lĩnh vực điển hình áp dụng Trí tuệ nhân tạo:

- Học Máy (Machine Learning)
- Xử lý Ngôn ngữ Tự nhiên (Natural Language Processing - NLP)
- Thị giác Máy Tính (Computer Vision)
- Hệ Thống Chuyên Gia (Expert Systems)
- Robot Học (Robotics)
- Mạng Neural Nhân Tạo (Artificial Neural Networks)

1.1.2. Machine Learning

Tính đến thời điểm hiện tại, có rất nhiều định nghĩa về machine learning. Machine learning (ML) hay máy học là một nhánh của trí tuệ nhân tạo (AI), nó là một lĩnh vực nghiên cứu cho phép máy tính có khả năng cải thiện chính bản thân chúng dựa trên dữ liệu mẫu (training data) hoặc dựa vào kinh nghiệm (những gì đã được học). Machine learning có thể tự dự đoán hoặc đưa ra quyết định mà không cần được lập trình cụ thể.

Bài toán machine learning thường được chia làm hai loại là dự đoán (prediction) và phân loại (classification). Các bài toán dự đoán như dự đoán giá nhà, giá xe… Các bài toán phân loại như nhận diện chữ viết tay, nhận diện đồ vật…

Phân loại Machine Learning:

Supervised Learning (Học có giám sát): Supervised Learning (Học có giám sát) là một nhóm thuật toán sử dụng dữ liệu được gán nhãn nhằm mô hình hóa mối quan hệ giữa biến đầu vào (x) và biến đầu ra (y). Với Supervised Learning, bên cạnh xây dựng các mô hình mạnh, việc thu thập và gán nhãn dữ liệu tốt và hợp lý cũng đóng vai trò then chốt để giải quyết các bài toán trong thực tế.

Một số thuật toán Supervised Learning phổ biến :

- Linear Regression (Hồi quy tuyến tính)
- Logistic Regression (Hồi quy logistic)
- Classification (Phân loại)
- Naive Bayes Classifier (Phân loại Naïve Bayes)
- K-Nearest Neighbors
- Decision Tree (Cây quyết định)
- Support Vector Machine

Unsupervised Learning (Học không có giám sát): Ngược lại với Supervised Learning, Unsupervised Learning (Học không giám sát) là một nhóm thuật toán sử dụng dữ liệu không có nhãn. Các thuật toán theo cách tiếp cận này hướng đến việc mô hình hóa được cấu trúc hay thông tin ẩn trong dữ liệu.

Một số thuật toán Unsupervised Learning phổ biến:

- K-Means (Phân cụm K-Means)
- DBSCAN (Phân cụm dựa trên mật độ)
- Spectral Clustering (Phân vùng quang phổ)
- Hierarchical Clustering (Phân cụm phân cấp)
- Apriori
- SVD (Phân tích suy biến)

Reinforcement Learning (Học tăng cường): Học tăng cường (RL) là kỹ thuật máy học (ML) giúp đào tạo phần mềm đưa ra quyết định nhằm thu về kết quả tối ưu nhất. Kỹ thuật này bắt chước quy trình học thử và sai mà con người sử dụng để đạt được mục tiêu đã đặt ra. RL giúp phần mềm tăng cường các hành động hướng tới mục tiêu, đồng thời bỏ qua các hành động làm xao lãng mục tiêu.

Một số thuật toán Reinforcement Learning phổ biến:

- Q-Learning
- Hybrid (Mô hình tương tác kết hợp)
- PPO (Proximal Policy Optimization)
- Multi-agent System (Hệ thống Đa tác nhân)

1.2. Xử lý ngôn ngữ tự nhiên

1.2.1. Định nghĩa

Xử lý ngôn ngữ tự nhiên (NLP) là một nhánh của Trí tuệ nhân tạo tập trung vào việc nghiên cứu sự tương tác giữa máy tính và ngôn ngữ tự nhiên của con người, dưới dạng tiếng nói hoặc văn bản. Mục tiêu của NLP là giúp máy tính hiểu và thực hiện hiệu quả các nhiệm vụ liên quan đến ngôn ngữ con người, như tương tác người-máy, cải thiện giao tiếp giữa con người, và nâng cao hiệu quả xử lý văn bản và lời nói[10].

Xử lý ngôn ngữ tự nhiên (NLP) được chia thành hai nhánh lớn: xử lý tiếng nói và xử lý văn bản:

- Xử lý tiếng nói tập trung vào các thuật toán xử lý ngôn ngữ ở dạng âm thanh, bao gồm nhận dạng tiếng nói (chuyển từ tiếng nói sang văn bản) và tổng hợp tiếng nói (chuyển từ văn bản sang tiếng nói).

- Xử lý văn bản tập trung vào phân tích dữ liệu văn bản với các ứng dụng như tìm kiếm thông tin, dịch máy, tóm tắt văn bản tự động và kiểm lỗi chính tả. Xử lý văn bản có thể chia thành hiểu văn bản (phân tích) và sinh văn bản (tạo văn bản mới).

1.2.2. Các bài toán và ứng dụng

Nhận dạng tiếng nói (ASR/STT): Chuyển đổi tiếng nói thành văn bản, dùng trong điều khiển qua giọng nói.

Tổng hợp tiếng nói (TTS): Chuyển đổi văn bản thành tiếng nói, dùng trong đọc văn bản tự động.

Truy xuất thông tin (IR): Tìm kiếm và sắp xếp tài liệu từ các nguồn lớn, như công cụ tìm kiếm Google, Yahoo, Bing.

Phân tích cảm xúc: Xác định cảm xúc từ văn bản, ứng dụng trong quản lý khách hàng và phân tích thị trường.

Trả lời câu hỏi (QA): Tự động trả lời câu hỏi bằng cách truy xuất thông tin từ tài liệu.

Tóm tắt văn bản tự động: Rút gọn văn bản để tạo ra bản tóm tắt ngắn gọn, có hai phương pháp chính là trích xuất và tóm lược ý.

Chatbot: Chương trình trò chuyện tự động với người dùng, hỗ trợ khách hàng và tìm kiếm thông tin.

Dịch máy (MT): Tự động dịch văn bản giữa các ngôn ngữ, sử dụng các phương pháp như EBMT, RBMT, SMT, và NMT.

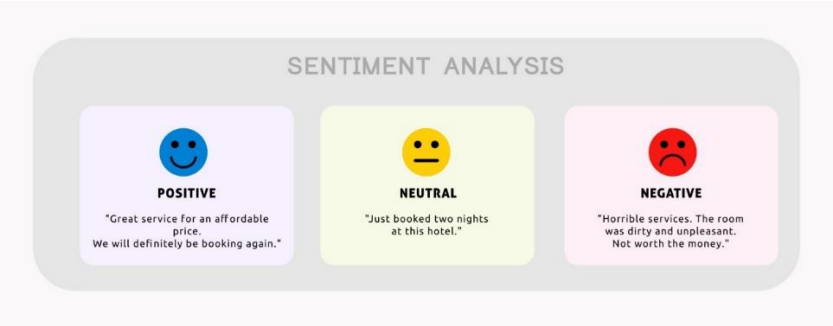
Kiểm lỗi chính tả tự động: Phát hiện và sửa lỗi chính tả, ngữ pháp, và ngữ nghĩa trong văn bản.

1.3. Sentiment Analysis

1.3.1. Định nghĩa

Phân tích cảm nghĩ hay Sentiment Analysis là một tập hợp con của xử lý ngôn ngữ tự nhiên (NLP) sử dụng học máy để phân tích và phân loại giọng điệu cảm xúc của dữ liệu văn bản. Các mô hình cơ bản chủ yếu tập trung vào việc phân loại các trạng thái tích cực, tiêu cực và trung lập nhưng cũng có thể xét đến những cảm xúc cơ bản của người nói (vui mừng, tức giận, phẫn nộ), cũng như các ý định mua hàng.

Bối cảnh sẽ thêm sự phức tạp vào quá trình Phân tích cảm nghĩ. Ví dụ: câu cảm thán "không có gì!" có ý nghĩa khác nhau đáng kể tùy thuộc vào việc người nói đang bình luận về những gì người đó thích hay không thích về một sản phẩm. Để hiểu cụm từ "Tôi thích nó", hệ thống phải có khả năng gỡ rối ngữ cảnh để hiểu "nó" ám chỉ điều gì. Ý nghĩ "mỉa mai" cũng rất phổ biến vì người nói có thể đang nói điều gì đó tích cực nhưng lại có ý ngược lại.



Hình 1.1. Sentiment Analysis

1.3.2. Các ứng dụng của Sentiment Analysis

Một số ví dụ về ứng dụng của Sentiment Analysis bao gồm:

- Các nhà thiết kế sản phẩm sử dụng Sentiment Analysis để xác định tính năng nào đang gây được tiếng vang với khách hàng và do đó xứng đáng

được đầu tư và tập trung hơn. Ngược lại, họ có thể tìm hiểu khi nào một sản phẩm hoặc tính năng không còn giá trị và điều chỉnh để ngăn chặn tồn kho tăng cao.

- Các công ty tiếp thị chủ yếu dựa vào Phân tích cảm nghĩ để giúp họ điều chỉnh thông điệp, khám phá những người có ảnh hưởng trên mạng (KoLs) và xây dựng chiến dịch marketing truyền miệng tích cực.

- Các tổ chức bán lẻ khai thác Phân tích cảm nghĩ để xác định sản phẩm nào có khả năng bán chạy và điều chỉnh hàng tồn kho cũng như khuyến mãi cho phù hợp.

- Các nhà đầu tư có thể xác định các xu hướng mới nổi lên trong các cuộc trò chuyện trực tuyến, dự báo trước các cơ hội thị trường.

- Các chính trị gia sử dụng nó để lấy mẫu thái độ của cử tri về các vấn đề quan trọng.

1.3.3. Cách hoạt động của mô hình Sentiment Analysis

- **Feature Engineering trong học máy**

Feature Engineering là quá trình chuyển đổi dữ liệu thô thành đầu vào cho thuật toán học máy. Để được sử dụng trong các thuật toán học máy, các đặc điểm (feature) phải được đưa vào các vectors đặc trưng, là các vectors số đại diện cho giá trị của từng đặc điểm. Để phân tích cảm nghĩ, dữ liệu văn bản phải được đưa vào các vectors từ ngữ, là vectors của các số biểu thị giá trị cho mỗi ký tự. Văn bản đầu vào có thể được mã hóa thành các vectors ký tự bằng cách sử dụng các kỹ thuật đếm như Bag of Words (BoW), bag-of-ngrams hoặc Term Frequency/Inverse Document Frequency (TF-IDF).

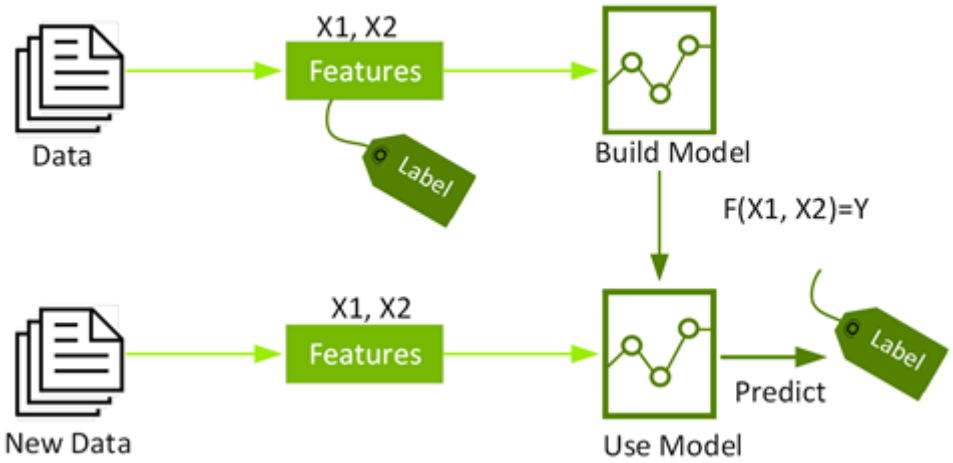
label	word	TF	DF	TF-IDF
1	love	1	1	1
1	bike	1	2	.5
0	returned	1	1	1
0	bike	1	2	.5

Hình 1.2. Feature Engineering

- **Sử dụng học máy có giám sát (Supervised Learning)**

Sau khi văn bản đầu vào đã được chuyển đổi thành vectors từ ngữ, thuật toán học máy phân loại có thể được sử dụng để phân loại cảm xúc. Phân loại là một

nhóm các thuật toán học máy được giám sát để xác định chủng loại nào mà một item thuộc về (chẳng hạn như văn bản là tiêu cực hay tích cực) dựa trên dữ liệu được gắn nhãn (chẳng hạn như văn bản được gắn nhãn là tích cực hay tiêu cực).



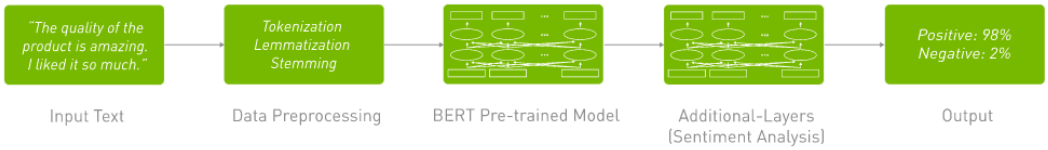
Hình 1.3. Sử dụng Supervised Learning

Các thuật toán học máy phân loại có thể được sử dụng để phân tích cảm nghĩ bao gồm:

- Naïve Bayes là một bộ các thuật toán xác suất xác định xác suất có điều kiện của lớp dữ liệu đầu vào.
- Support Vector Machines tìm thấy một "hyperplane" trong không gian N chiều (N là số lượng đối tượng địa lý) phân loại rõ ràng các điểm dữ liệu.
- Logistic regression sử dụng hàm logistic để mô hình hóa xác suất của một cấp nhất định.

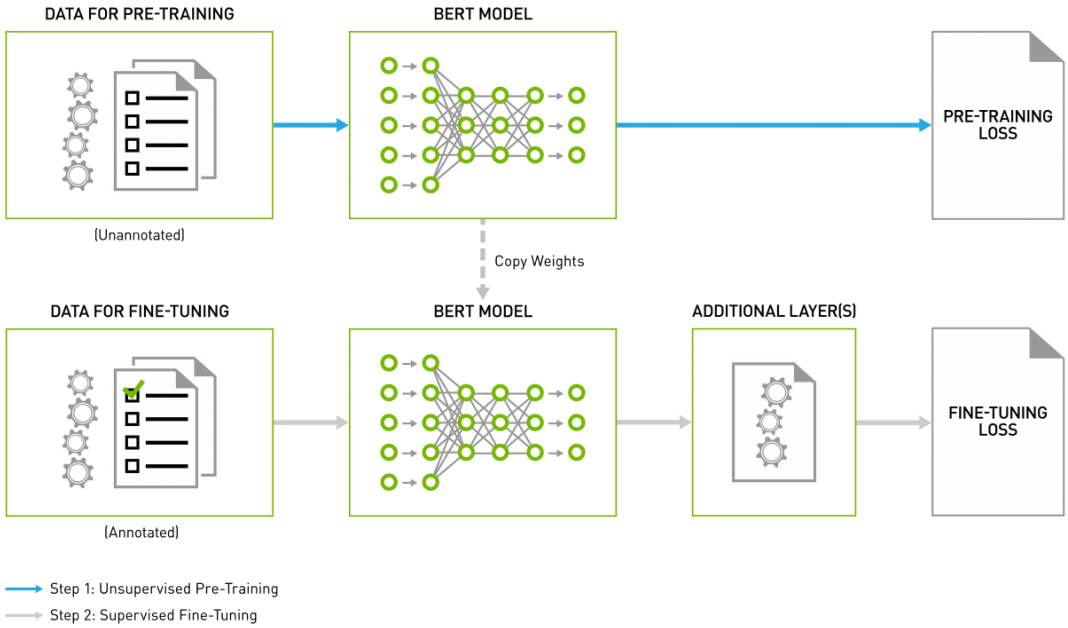
- **Phân tích cảm nghĩ bằng cách sử dụng Deep Learning**

Học sâu (DL) là một tập hợp con của học máy (ML) sử dụng các neural networks để mang lại độ chính xác cao nhất trong các tác vụ như NLP và các tác vụ khác. Các kỹ thuật nhúng từ DL chẳng hạn như Word2Vec mã hóa các ký tự theo những cách có ý nghĩa bằng cách học các liên kết ký tự, ý nghĩa, ngữ nghĩa và cú pháp . Các thuật toán DL cũng cho phép đào tạo từ đầu đến cuối các mô hình NLP mà không cần phải thiết kế thủ công các tính năng từ dữ liệu thô đầu vào.



Hình 1.4. Sử dụng Deep Learning

Có nhiều biến thể của thuật toán học sâu. Recurrent neural networks (RNNs) là công cụ toán học để phân tích các mẫu ngôn ngữ và dữ liệu theo trình tự, được dùng trong xử lý ngôn ngữ tự nhiên như thính giác và lời nói cho Alexa, dịch thuật ngôn ngữ, dự đoán chứng khoán và giao dịch thuật toán. BERT (Đại diện bộ mã hóa hai chiều từ Transformer) là một giải pháp thay thế cho RNNs, áp dụng kỹ thuật phân tích cú pháp câu bằng cách tập trung vào các ký tự liên quan trước và sau nó. BERT đã cách mạng hóa NLP bằng cách cung cấp độ chính xác cao, có thể so sánh với con người trong việc nhận biết ý định, phân tích tình cảm, và giữ ngữ cảnh tốt hơn. Một thách thức chính với các mô hình ngôn ngữ là thiếu dữ liệu được dán nhãn, nhưng BERT được đào tạo trên các nhiệm vụ không giám sát và sử dụng bộ dữ liệu phi cấu trúc từ kho sách, Wikipedia tiếng Anh, v.v.



Hình 1.5. Sử dụng BERT Model

1.4. Deep Learning

1.4.1. Deep Learning

Deep Learning hay Học sâu là một phương thức trong lĩnh vực trí tuệ nhân tạo (AI) cũng có thể được coi là một lĩnh vực thuộc Machine Learning ,được sử

dụng để dạy máy tính xử lý dữ liệu theo cách được lấy cảm hứng từ bộ não con người. Mô hình học sâu có thể nhận diện nhiều hình mẫu phức tạp trong hình ảnh, văn bản, âm thanh và các dữ liệu khác để tạo ra thông tin chuyên sâu và dự đoán chính xác. Ta có thể sử dụng các phương pháp học sâu để tự động hóa các tác vụ thường đòi hỏi trí tuệ con người, chẳng hạn như mô tả hình ảnh hoặc chép lời một tập tin âm thanh. Chủ yếu hoạt động với mạng thần kinh nhân tạo để bắt chước khả năng tư duy và suy nghĩ của bộ não con người. Thực ra các khái niệm liên quan đến mạng nơ-ron nhân tạo và Deep Learning đã được phát triển những năm 1960. Nhưng nó bị giới hạn bởi lượng dữ liệu và khả năng tính toán tại thời điểm đó[3].

Mô hình học sâu có một số trường hợp sử dụng trong lĩnh vực ô tô, hàng không vũ trụ, sản xuất, điện tử, nghiên cứu y học và nhiều lĩnh vực khác. Sau đây là một vài ví dụ về học sâu:

- Xe tự lái sử dụng các mô hình học sâu để tự động phát hiện biển báo giao thông và người đi bộ.
- Hệ thống quốc phòng sử dụng mô hình học sâu để tự động gắn cờ các khu vực được quan tâm trong ảnh vệ tinh.
- Phân tích hình ảnh y khoa sử dụng học sâu để tự động phát hiện các tế bào ung thư trong chẩn đoán y tế.
- Các nhà máy sử dụng ứng dụng học sâu để tự động phát hiện con người hoặc vật thể khi những đối tượng này đang nằm trong khoảng cách không an toàn của máy móc.

Có thể nhóm các trường hợp sử dụng mô hình học sâu khác nhau thành 4 loại chính: thị giác máy tính, nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên (NLP) và công cụ đề xuất

- Thị giác máy tính: là khả năng của máy tính thực hiện trích xuất dữ liệu cũng như thông tin chuyên sâu từ hình ảnh và video. Máy tính có thể sử dụng các kỹ thuật học sâu để hiểu hình ảnh theo cách giống như con người.

- Nhận dạng giọng nói: Các mô hình học sâu có thể phân tích giọng nói con người, bất kể mẫu giọng, cao độ, tông, ngôn ngữ và giọng vùng miền khác nhau.

- Kỹ thuật xử lý ngôn ngữ tự nhiên: Máy tính sử dụng các thuật toán học sâu để thu thập thông tin chuyên sâu và ý nghĩa từ dữ liệu văn bản và tài liệu. Khả năng xử lý văn bản tự nhiên, do con người tạo ra này có một số trường hợp sử dụng như: tổng đài viên ảo và chatbot tự động, tự động tóm tắt tài liệu hoặc bài viết, lập

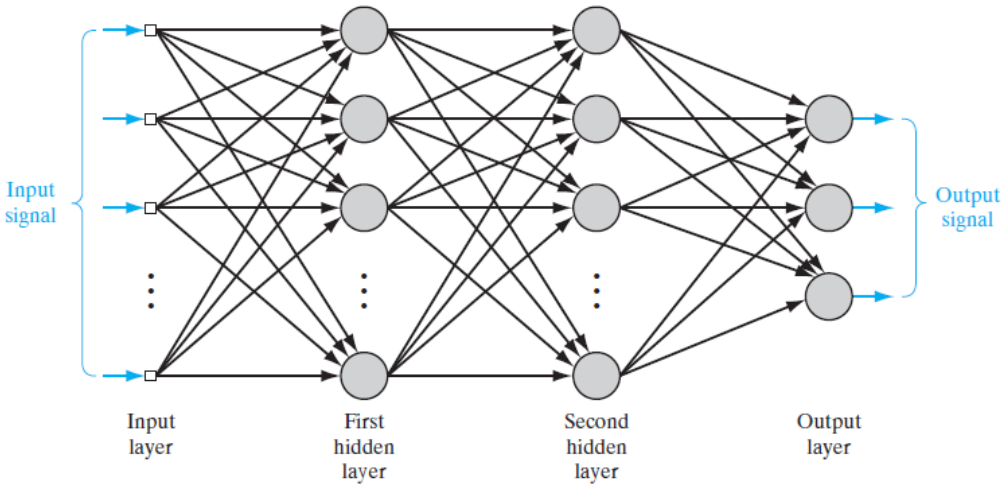
chỉ mục các cụm từ quan trọng thể hiện cảm xúc, chẳng hạn như những bình luận tích cực và tiêu cực trên mạng xã hội.

- Công cụ đề xuất: Các phương pháp này có thể phân tích hành vi của nhiều người dùng khác nhau và giúp họ khám phá các sản phẩm hoặc dịch vụ mới. Ví dụ: nhiều công ty truyền thông và giải trí, chẳng hạn như Netflix, Fox và Peacock, sử dụng mô hình học sâu để đưa ra các video đề xuất cá nhân hóa.

1.4.2. Mạng Neural nhân tạo

Mạng neural nhân tạo (hay còn gọi là mạng thần kinh nhân tạo) Artificial Neural Network (ANN) là một hệ thống các chương trình và cấu trúc dữ liệu mô phỏng cách xử lý thông tin của mạng neural sinh học. Nó được tạo nên từ một số lượng lớn các phần tử neural kết nối với nhau thông qua các liên kết (trọng số liên kết) làm việc như một thể thống nhất để giải quyết một vấn đề cụ thể nào đó. Một mạng neural nhân tạo được cấu hình cho một ứng dụng cụ thể (nhận dạng mẫu, phân loại dữ liệu, ...) thông qua một quá trình học từ tập các mẫu huấn luyện. Về bản chất học chính là quá trình hiệu chỉnh trọng số liên kết giữa các neural.

Cấu tạo của neural nhân tạo:



Hình 1.6. Cấu tạo của mạng Neural

- Tầng Input Layer (tầng vào): Tầng này nằm bên trái cùng của mạng, thể hiện cho các đầu vào của mạng

- Tầng Output Layer (tầng ra): là tầng nằm bên phải cùng và nó thể hiện cho những đầu ra của mạng

- Tầng First và Second hidden layer (tầng ẩn): tầng này nằm giữa tầng vào và tầng ra, nó thể hiện cho quá trình suy luận và logic của mạng

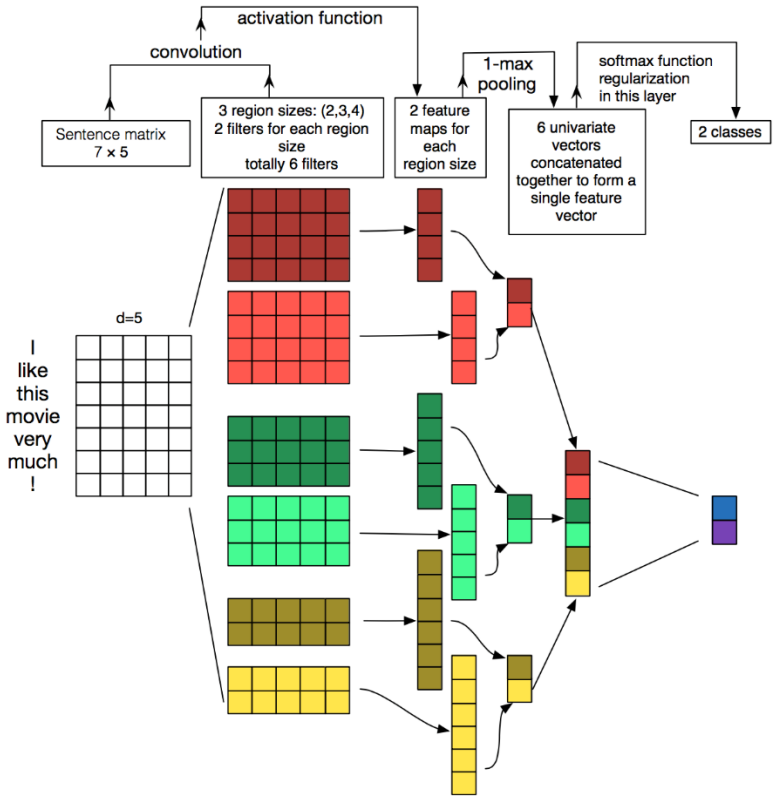
Một số loại mạng Neural nhân tạo:

- **Mạng Perceptron:** do Frank Rosenblatt tạo ra năm 1958, là mạng thần kinh đơn giản nhất với một lớp neural
- **Mạng neural truyền thẳng (Feedforward):** hay còn gọi là mạng perceptron nhiều lớp (MLP - Multi-layer perceptrons) gồm một lớp đầu vào, nhiều lớp ẩn và một lớp đầu ra, xử lý dữ liệu một chiều
- **Mạng neural tích chập (CNN):** thường dùng cho nhận dạng hình ảnh, khai thác phép nhân ma trận
- **Mạng thần kinh hồi quy (RNN):** có bộ nhớ lưu thông tin từ bước tính toán trước, dùng trong phân tích dữ liệu chuỗi thời gian như dự đoán thị trường chứng khoán

1.5. CNN trong xử lý ngôn ngữ tự nhiên

Mạng Nơ-ron tích chập là một kiến trúc mạng nơ-ron nhân tạo được phát triển ban đầu cho bài toán xử lý ảnh. Tuy nhiên, trong những năm gần đây, CNN đã được chứng minh là có hiệu quả cao trong nhiều nhiệm vụ xử lý ngôn ngữ tự nhiên (NLP), bao gồm cả phân loại văn bản[2]. Theo như François Chollet: Mạng Nơ-ron tích chập thường được sử dụng cho các bài toán xử lý hình ảnh, nhưng chúng cũng có thể được áp dụng cho các chuỗi, bao gồm cả chuỗi văn bản. Bằng cách xử lý văn bản dưới dạng một chuỗi các từ hoặc ký tự và sử dụng phép cuộn để phát hiện các mẫu cục bộ, CNN có thể xử lý hiệu quả các tác vụ như phân loại câu hoặc phân tích cảm xúc[4].

Về lý thuyết cơ bản thay vì input đầu vào là các điểm ảnh trong Computer Vision, đầu vào của phân tích ngôn ngữ tự nhiên là các mệnh đề, các văn bản được biểu diễn như một ma trận. Mỗi dòng của một ma trận tương ứng với một mã, đa phần đó là một từ, nhưng nó cũng có thể là một kí tự. Mối hàng chính là một vector đại diện cho một từ. Thông thường các vector word này được trình bày ở mức thấp như dạng word2vec hay Glove, nhưng nó cũng có thể là một vector với việc các từ sẽ được đánh chỉ số thuộc một bộ từ vựng. Và khi chung ta sử dụng nhiều chiều ánh xạ một câu thì nó sẽ tạo cho chúng ta một ma trận nhiều chiều, như thế nó đã được biến thành input image đầu vào. Chúng ta có thể xem, các bộ lọc sẽ trượt qua các mảnh vá của image đầu vào, nhưng trong NLP chúng ta thường sử dụng các bộ lọc trượt qua tất cả các dòng của ma trận (words). Do đó bộ lọc của ta sẽ - độ rộng bằng độ rộng của matrix, chiều dài có thể thay đổi nhưng thông thường ta sẽ trượt qua 2-5 từ[16]. Hình mô tả bên dưới sẽ dễ hình dung hơn:



Hình 1.7. Mô hình mạng CNN trong xử lý ngôn ngữ tự nhiên [16]

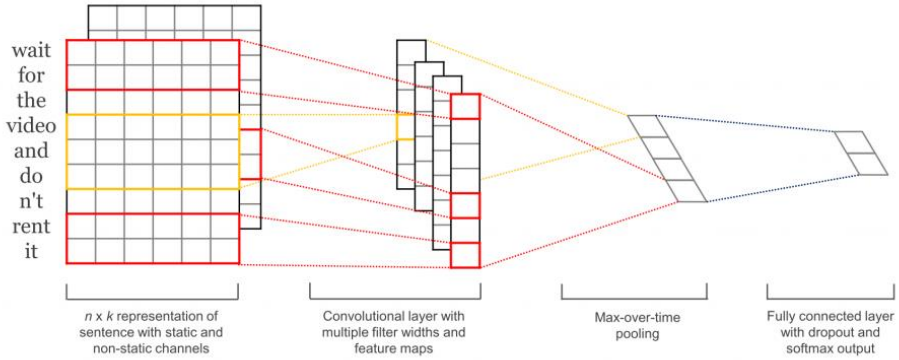
Sơ đồ trên chúng ta có matrix đầu vào cho 1 mệnh đề "I like this movie very much!" Nó sẽ là một ma trận với độ rộng là 5 và chiều dài là 7 dòng, tiếp theo như mô tả ta sẽ có các vùng để phân tích và kích thước sẽ là width-7, heigth tùy biến (2,3,4) và chúng ta sẽ apply 2 bộ lọc cho từng vùng, sau convolution đầu tiên ta sẽ có 6 output, tiếp theo sử dụng max-pooling để tìm đặc trưng cao nhất cho mỗi vùng, output layer 2 sẽ còn 3, chúng ta sẽ biến đổi để đưa 3 vector đơn này thành chung 1 vector cho 1 chức năng phân tích (cũng là input layer tiếp theo) cuối cùng softmax để đưa ra 2 lớp output cuối.

Ưu điểm của CNN trong phân loại văn bản:

Khả năng trích xuất đặc trưng: CNN có khả năng trích xuất các đặc trưng ngữ nghĩa từ văn bản một cách hiệu quả, ngay cả khi các đặc trưng này không được biểu thị rõ ràng.

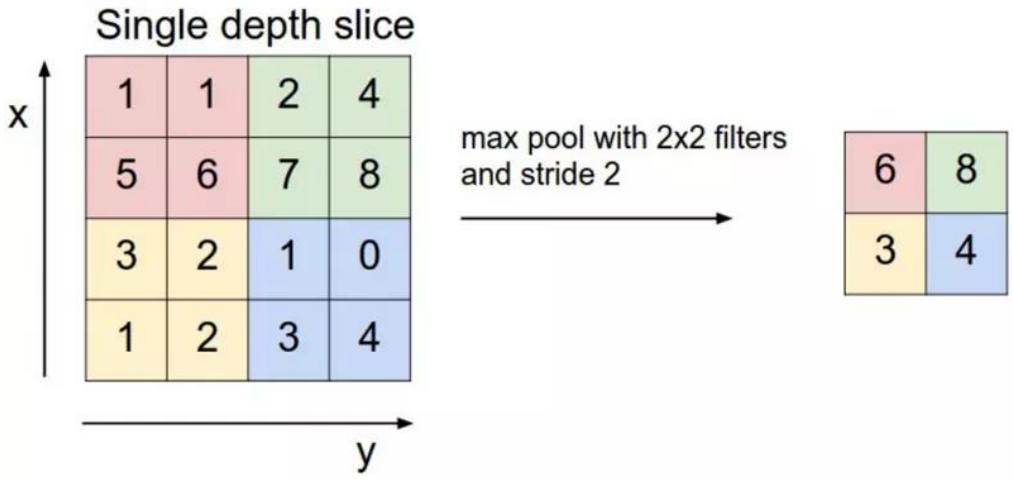
Khả năng xử lý dữ liệu dài: CNN có thể xử lý các văn bản dài một cách hiệu quả mà không cần phải chia nhỏ văn bản thành các đoạn nhỏ hơn.

Tính toán hiệu quả: CNN có thể được tính toán hiệu quả trên phần cứng máy tính hiện đại[5].



Hình 1.8. Quá trình xử lý của CNN trong bài toán NLP

Một khía cạnh nữa của Convolutional Neural Networks là các lớp pooling, thông thường nó sẽ được áp dụng sau các lớp convolutional. Phần lớn cách để tập hợp chính là áp dụng lấy max của các kết quả mỗi bộ lọc. Và chúng ta không cần phải pool cho ma trận output sau mỗi convolution layer mà có thể áp dụng ngay trên từng bộ lọc.



Hình 1.9. Lớp Pooling của mạng CNN

Vậy tại sao chúng ta cần phải gộp, phải hợp các output lại. Có những lý do sau:

- Pooling sẽ hỗ trợ việc cố định kích thước ma trận đầu ra. Ví dụ khi ta có 1000 bộ lọc, ta áp dụng tối đa pooling cho mỗi bộ lọc thì nó sẽ tạo ra 1000 chiều output, không cần quan tâm kích thước các bộ lọc hoặc kích thước đầu vào nhưng nó sẽ lấy được cùng kích thước output đầu ra.

- Pooling giúp giảm chiều của ma trận đầu ra nhưng vẫn giữ được thông tin quan trọng nhất từ các bộ lọc. Tuy nhiên, việc giảm chiều này có thể làm mất thông tin về vị trí xuất hiện của các đặc trưng trong đầu vào. Ví dụ, nếu một cụm từ tiêu cực như "not amazing" được phát hiện, thao tác max pooling sẽ giữ lại giá trị lớn ở vị trí có cụm từ này, nhưng thông tin về vị trí cụ thể của nó sẽ bị mất. Mặc dù vậy, thông tin vị trí có thể không quan trọng bằng việc nhận biết sự xuất hiện của đặc trưng đó.

1.6. Mạng BiLSTM

1.6.1. LSTM

- Trước khi hiểu về BiLSTM, ta cần hiểu về LSTM:

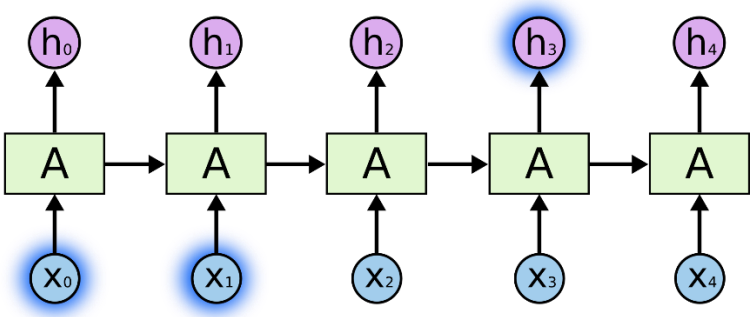
LSTM là một loại mạng neural hồi quy (RNN) được thiết kế để giải quyết vấn đề về gradient biến mất và gradient bùng nổ trong RNN truyền thống. LSTM có cấu trúc gồm các cổng (gates) như input gate, forget gate, và output gate giúp nó kiểm soát luồng thông tin một cách hiệu quả hơn[11].

Cấu trúc LSTM cho phép nó học và ghi nhớ các phụ thuộc dài hạn trong dữ liệu chuỗi, giúp nó phù hợp cho các tác vụ như dự đoán chuỗi, phân tích ngôn ngữ, và nhận dạng mẫu.

- Vấn đề phụ thuộc quá dài (Long-Term Dependencies):

Một điểm nổi bật của RNN chính là ý tưởng kết nối các thông tin phía trước để dự đoán cho hiện tại. Việc này tương tự như ta sử dụng các cảnh trước của bộ phim để hiểu được cảnh hiện thời. Nếu mà RNN có thể làm được việc đó thì chúng sẽ cực kì hữu dụng, tuy nhiên liệu chúng có thể làm được không? Câu trả lời là còn tùy.

Đôi lúc ta chỉ cần xem lại thông tin vừa có thôi là đủ để biết được tình huống hiện tại. **Ví dụ**, ta có câu: “cá thì sống ở dưới nước” thì ta chỉ cần đọc tới “cá thì sống ở dưới” là đủ biết được chữ tiếp theo là “nước” rồi. Trong tình huống này, khoảng cách tới thông tin có được cần để dự đoán là nhỏ, nên RNN hoàn toàn có thể học được[11].

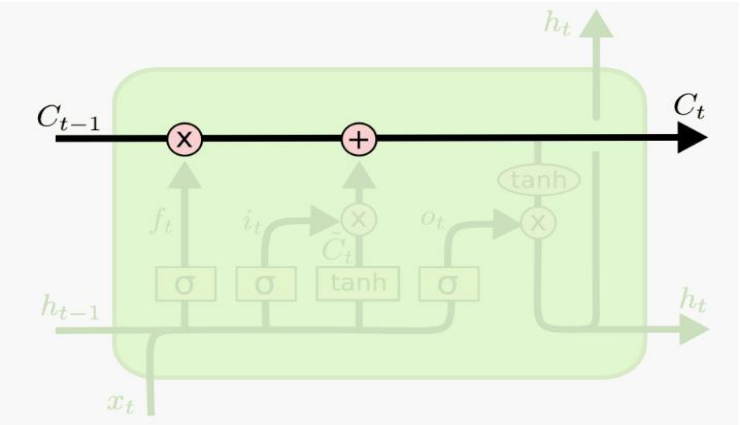


Hình 1.10. Vấn đề phụ thuộc quá dài của mạng LSTM [3]

- *Ý tưởng của LSTM:*

Chìa khóa của LSTM là trạng thái tế bào (cell state) - chính đường chạy thông ngang phía trên của sơ đồ hình vẽ.

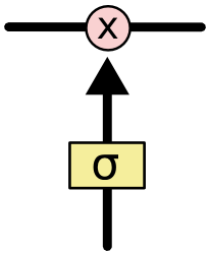
Trạng thái tế bào là một dạng giống như băng truyền. Nó chạy xuyên suốt tất cả các mắt xích (các nút mạng) và chỉ tương tác tuyến tính đôi chút. Vì vậy mà các thông tin có thể dễ dàng truyền đi thông suốt mà không sợ bị thay đổi.



Hình 1.11. Trạng thái băng truyền LSTM [3]

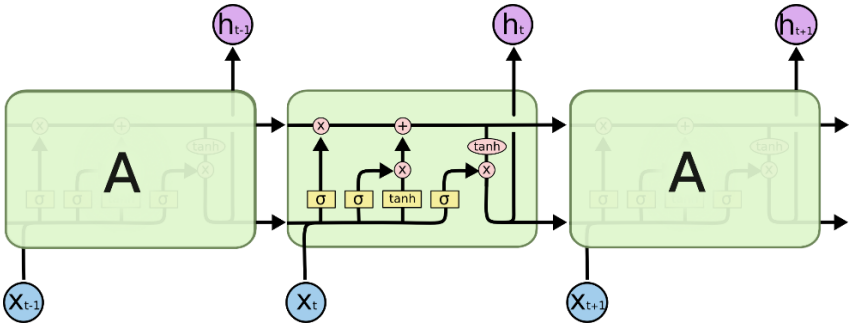
LSTM có khả năng bỏ đi hoặc thêm vào các thông tin cần thiết cho trạng thái tế báo, chúng được điều chỉnh cẩn thận bởi các nhóm được gọi là cổng (gate).

Các cổng là nơi sàng lọc thông tin đi qua nó, chúng được kết hợp bởi một tầng mạng sigmoid và một phép nhân. Một LSTM gồm có 3 cổng như vậy để duy trì và điều hành trạng thái của tế bào.



Hình 1.12. Cổng nơi để LSTM băng qua [3]

LSTM được thiết kế để tránh được vấn đề phụ thuộc xa (long-term dependency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của chúng, chứ ta không cần phải huấn luyện nó để có thể nhớ được. Tức là ngay nội tại của nó đã có thể ghi nhớ được mà không cần bất kì can thiệp nào.



Hình 1.13. Mô-đun lặp lại trong một LSTM chứa bốn lớp tương tác [3]

1.6.2. BiLSTM

BiLSTM là một mở rộng của LSTM, cho phép xử lý chuỗi dữ liệu theo cả hai hướng thời gian. Điều này giúp nắm bắt ngữ cảnh tốt hơn và cải thiện hiệu suất của các tác vụ xử lý ngôn ngữ tự nhiên và các tác vụ liên quan đến chuỗi dữ liệu khác.

- BiLSTM mở rộng LSTM bằng cách thêm một chiều xử lý ngược lại. Cụ thể:
  - Forward LSTM: Xử lý chuỗi dữ liệu từ đầu đến cuối. Ví dụ, với một câu, forward LSTM sẽ đọc từ từ đầu tiên đến từ cuối cùng.
  - Backward LSTM: Xử lý chuỗi dữ liệu từ cuối đến đầu. Ví dụ, backward LSTM sẽ đọc từ từ cuối cùng đến từ đầu tiên.
- Kiến trúc của BiLSTM:
  - Forward Pass: Một LSTM xử lý chuỗi dữ liệu theo chiều từ đầu đến cuối.
  - Backward Pass: Một LSTM khác xử lý chuỗi dữ liệu theo chiều từ cuối đến đầu.

- Kết Hợp Kết Quả: Tại mỗi thời điểm, kết quả từ forward LSTM và backward LSTM được kết hợp lại (thường bằng cách nối hoặc cộng) để tạo ra một biểu diễn duy nhất.

- Ưu điểm của BiLSTM:

- Nắm Bắt Ngữ Cảnh Toàn Diện: Bằng cách xử lý dữ liệu theo cả hai hướng, BiLSTM có thể nắm bắt thông tin ngữ cảnh từ cả trước và sau mỗi đơn vị trong chuỗi. Điều này đặc biệt quan trọng trong NLP, nơi ngữ cảnh xung quanh một từ hoặc cụm từ có thể thay đổi ý nghĩa của nó.
- Hiệu Quả Cao Trong Nhiều Tác Vụ: BiLSTM đã chứng minh hiệu quả cao trong nhiều tác vụ NLP như gán nhãn từ loại (POS tagging), nhận dạng thực thể có tên (NER), phân tích cảm xúc (sentiment analysis), và dịch máy (machine translation).

- Ví dụ minh hoạ:

Giả sử chúng ta có một câu: "The movie was not bad at all."

- Forward LSTM: Đọc câu từ trái sang phải và có thể học rằng "The movie was not" dẫn đến một dự đoán nào đó.
- Backward LSTM: Đọc câu từ phải sang trái và có thể học rằng "bad at all" mang ý nghĩa tích cực khi nhìn từ ngữ cảnh tổng thể.
- Kết Hợp: Khi kết hợp cả hai chiều, BiLSTM có thể nhận ra rằng toàn bộ câu "The movie was not bad at all" mang ý nghĩa tích cực.

CHƯƠNG II. XÂY DỰNG MÔ HÌNH PHÂN LOẠI CẢM XÚC NGƯỜI DÙNG TRÊN MẠNG XÃ HỘI

2.1. Chuẩn bị tập dữ liệu, môi trường và các thư viện cần thiết

2.1.1. Môi trường triển khai

Google Colab (Colaboratory) là một công cụ cung cấp môi trường thực thi mã lệnh tương tác miễn phí trên nền tảng đám mây, cho phép ta viết và chạy Python trực tiếp trong trình duyệt. Colab đặc biệt hữu ích cho các tác vụ học máy, khoa học dữ liệu và phân tích dữ liệu. Nó có nhiều tính năng mạnh mẽ, như hỗ trợ GPU và TPU, mà ta có thể sử dụng mà không cần phải cấu hình thêm gì nhiều.

Các Đặc Điểm Chính của Google Colab:

- Miễn phí và Dễ Sử Dụng: Ta chỉ cần có tài khoản Google là có thể sử dụng Colab. Môi trường này không yêu cầu cài đặt phần mềm hoặc cấu hình phức tạp.

- Hỗ trợ GPU và TPU: Colab cho phép ta sử dụng GPU và TPU miễn phí, giúp tăng tốc các tác vụ đòi hỏi hiệu suất cao như huấn luyện mô hình học sâu.

- Tích hợp với Google Drive: Ta có thể dễ dàng lưu trữ và chia sẻ các notebook của mình qua Google Drive.

- Môi Trường Cài Đặt Sẵn: Colab đi kèm với nhiều thư viện Python phổ biến đã được cài đặt sẵn, như TensorFlow, Keras, PyTorch, OpenCV, và nhiều thư viện khác.

- Tích hợp với GitHub: Ta có thể mở các notebook từ GitHub trực tiếp trong Colab, và ngược lại, dễ dàng chia sẻ và cộng tác với người khác.

- Giao diện Thân thiện với Người Dùng: Giao diện tương tự như Jupyter Notebook, dễ dàng sử dụng cho cả người mới bắt đầu và các nhà nghiên cứu chuyên sâu.

2.1.2. Tập dữ liệu

- **Vietnamese Social Media Emotion** [16]

  Nhận biết cảm xúc là một cách tiếp cận cao hơn hoặc trường hợp đặc biệt của phân tích tình cảm. Trong nhiệm vụ này, kết quả được tạo ra theo 3 nhãn: tích cực tiêu cực hoặc trung lập. Nhận biết cảm xúc đóng một vai trò

quan trọng trong việc đo lường giá trị thương hiệu của một sản phẩm bằng cách ghi nhận những cảm xúc cụ thể trong nhận xét của khách hàng.

Trong bộ dữ liệu [16] với khoảng hơn 6.900 câu có chú thích của con người với 3 nhãn, góp phần vào nghiên cứu nhận dạng cảm xúc trong tiếng Việt, vốn là ngôn ngữ sử dụng ít tài nguyên trong Xử lý ngôn ngữ tự nhiên. Với dữ liệu này giúp chúng ta huấn luyện cho mạng neural của máy.

Mô tả file ***.CSV** dữ liệu:

	A	B	C	D	E	F	G	H	I	J	K
1	label	text									
2	trung lập	cho mình xin bài nhạc tên là gì với ạ									
3	tiêu cực	cho đáng đời con quỷ . về nhà lôi con nhà mày ra mà đánh 😡									
4	tiêu cực	lo học đi . yêu đương lol gì hay lại thích học sinh học									
5	tích cực	ước gì sau này về già vẫn có thể như cụ này :))									
6	tích cực	mỗi lần có video của con là cứ coi đi coi lại hoài . cưng con quá .									
7	tiêu cực	thằng kia sao mày bắt vợ với bồ tao dọn thế kia . nhà mày ở đâu tao đến thịt mày chết									
8	trung lập	một lí do trog muôn vàn lí do									
9	tích cực	thật hay đùa ác vậy . không thể tin được									
10	tiêu cực	ko phải con mình , mà xem còn thấy đau như vậy huống gì người trong cuộc . thật là phẫn nộ mà . cơ q									
11	trung lập	nghe đi rồi khóc 1 trận cho thoải mái . đừng cố gồng mình lên nữa									
12	tích cực	công nhận sáng tạo thật đấy									
13	tiêu cực	đòn tấn công cực gắt và cục súc của anh 😂😂😂									
14	trung lập	trời nắng nóng thế này mình muốn bán nước không biết thu nhập có cao không ạ :3									
15	tích cực	minh biết nữa ne									
16	tiêu cực	mấy thằng củ lol việt nam nhảm nhí :))									
17	trung lập	tui thí ch vả i lã mày mà ăn nhỉ u nơ người lã mày mọi người anh , bị lở miệ người									
18	trung lập	bếp dầu , nhiều nhà vẫn dùng									
19	trung lập	nếu thấy phụ nữ quá phức tạp để hiểu và chinh phục thì đây là cách giải quyết vấn đề									
20	trung lập	thời buổi bây chừ chuyện gì cũng có thể xảy ra , vậy nên thôi trất , đừng nên tin ai									
21	tích cực	sau em này mà làm cô giáo ngồi kể chuyện hs lại há mồm ra nghe 😄									
22	tiêu cực	con chó đăng video , mày bị ngu à mà mỗi mẩu lại hiện chữ ăn cặc à									
23	tiêu cực	coi video chưa hiểu gì đã làm ầm lên									

Hình 2.1. Tập dữ liệu [16]

2.1.3. Các thư viện.

```
import numpy as np
import pandas as pd
import tensorflow as tf
import pickle
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow import keras
from tensorflow.keras.layers import Embedding, Dense, Dropout, Bidirectional, LSTM, GRU, Input, GlobalMaxPooling1D, LayerNormalization, Conv1D, MaxPooling1D
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras import Sequential
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from pyvi import ViTokenizer
from pyvi import ViUtils
```

Hình 2.2. Các thư viện cần để xây dựng mô hình

- Numpy(Numerical Python): là một thư viện cơ bản cho tính toán khoa học với Python. Nó cung cấp hỗ trợ cho mảng (array), ma trận (matrix) và nhiều hàm toán học khác.

- Pandas: là một thư viện mạnh mẽ cho thao tác và phân tích dữ liệu trong Python. Nó cho phép ta làm việc với các tệp file dữ liệu như CSV hay excel.

- Maplotlib: là một thư viện toàn diện để hiển thị đồ thị, hình ảnh mình hoạ cho dữ liệu của bài toán.

- Pickle: là một module trong Python và được sử dụng để lưu các cấu trúc dữ liệu phức tạp vào file hoặc cơ sở dữ liệu.

- TensorFlow: là một framework mã nguồn mở được phát triển bởi Google dành cho học máy (Machine Learning) và học sâu (Deep Learning). TensorFlow cung cấp các công cụ hỗ trợ cho việc xây dựng, huấn luyện và triển khai các mô hình học máy một cách linh hoạt và hiệu quả trên nhiều nền tảng khác nhau[1]. TensorFlow sở hữu nhiều ưu điểm vượt trội như sau:

+ Hỗ trợ nhiều nền tảng

+ Hiệu suất cao

+ Được sử dụng rộng rãi

+ Hỗ trợ mạnh mẽ cho các mô hình học sâu

+ API linh hoạt

- Keras: là một API cấp cao cho mạng neural và được tích hợp trong TensorFlow, nó có thể giúp xây dựng các mô hình học máy và học sâu một cách hiệu quả bằng việc cung cấp các lớp (Layers) và các API như Sequential API và Preprocessing API

+ Layers:

- Core Layers: các khối xây dựng cơ bản như lớp Dense, lớp Activation, lớp Dropout,v.v
- Convolutional Layers: được sử dụng cho những dữ liệu liên quan tới mô hình Convolutional Neural Network
- Recurrent Layers: được sử dụng cho dữ liệu tuần tự, ví dụ: LSTM, GRU
- Normalization Layers: được sử dụng để chuẩn hoá đầu vào, ví dụ như LayerNormalization

+ Preprocessing API: là một API có chức năng tiền xử lý dữ liệu trước khi đưa vào mạng neural như là Tokenization, padding chuỗi và các tiện ích khác cho dữ liệu văn bản

2.2. Tiền xử lý dữ liệu

Trong các bài toán Xử lý ngôn ngữ tự nhiên (NLP) khi xử lý dữ liệu ta thường có một bước quan trọng gọi là Tokenization. Quá trinh này giúp phân chia văn bản thành các đơn vị nhỏ hơn, gọi là token. Token có thể là từ đơn, cụm từ, câu hoặc thậm chí là các ký tự riêng lẻ, tuỳ thuộc vào ứng dụng cụ thể của NLP.

Tokenization thường được sử dụng nhất là Tokenization từ (Word Tokenization): phân chia văn bản thành các từ riêng lẻ. Ví dụ, câu "Bầu trời hôm nay đẹp nhỉ" sẽ được token hoá thành ["Bầu", "trời", "hôm", "nay", "đẹp", "nhỉ"]. Tiếng Việt khác tiếng Anh ở chỗ là đối với tiếng Việt một token có thể là 2 từ kết nối với nhau chứ không phải 1 từ như tiếng Anh. Do đó đối với bài toán này trước khi chúng ta thực hiện Tokenization bằng các thư viện của Tensorflow thì chúng ta cần phải Tokenization bằng các thư viện hỗ trợ xử lý ngôn ngữ tiếng Việt như pyvi. Ta sẽ tiến hành nhân đôi 1 câu thành 2 câu, 1 câu sẽ giữ nguyên dấu và 1 câu sẽ loại bỏ dấu và sử dụng cả 2 câu này để tiến hành huấn luyện nhằm đưa ra kết quả tốt nhất[15].

```
input_pre

['cho mình xin bài nhạc tên là gì với ạ',
 'cho minh xin bai nhac ten la gi voi a',
 'cho đáng_đời con quỷ về nhà lôi con nhà mày ra mà đánh 😡',
 'cho dang doi con quy ve_nha loi con nha may_ra ma_danh',
 'lo học đi yêu_đương lol gì hay lại thích học_sinh_học',
 'lo hoc di_yeu duong lol gi hay lai thich hoc sinh hoc',
 'ước gì sau_này về già vẫn có_thể như cụ này',
 'uoc gi sau nay ve_gia van co the nhu cu nay',
 'mỗi lần có video của con là cứ coi đi coi lại hoài cưng con quá',
 'moi lan co video cua con la cu coi di_coi lai hoai cung con qua',
 'thằng kia sao mày bắt vợ với bồ tao dọn thế kia nhà mày ở đâu tao đến thịt mày chết',
 'thang kia sao may bat vo voi bo tao don the kia nha may o dau tao den thit may chet',
 'một lí_do trog muôn_vàn lí_do',
 'mot li do trog muon van li do',
 'thật hay đùa ác_vậy không_thể tin được',
 'that hay dua ac vay khong the tin duoc',
```

Hình 2.3. Tokenization bằng thư viện pyvi

Sau khi thực hiện Tokenization, một bước quan trọng tiếp theo là xây dựng từ điển (Vocabulary Construction). Đây là bước tạo ra một danh sách các từ duy nhất (token) từ toàn bộ tập dữ liệu văn bản. Từ điển này sẽ chứa ít nhất tất cả các token xuất hiện trong tập dữ liệu và thường được sử dụng trong các bước tiếp theo của xử lý ngôn ngữ tự nhiên.

```
word_index = tokenizer_data.word_index
word_index
```

```
{'<OOV>': 1,
 'tao': 2,
 'con': 3,
 'cho': 4,
 'co': 5,
 'khong': 6,
 'không': 7,
 'la': 8,
 'là': 9,
 'có': 10,
 'nay': 11,
 'mà': 12,
 'ma': 13,
 'may': 14,
 'no': 15,
 'ra': 16,
 'thi': 17,
 'nó': 18,
 'này': 19,
 'đi': 20,
 'qua': 21,
 'nguoi': 22,
 'thì': 23,
```

Hình 2.4. Bộ từ điển được mapping

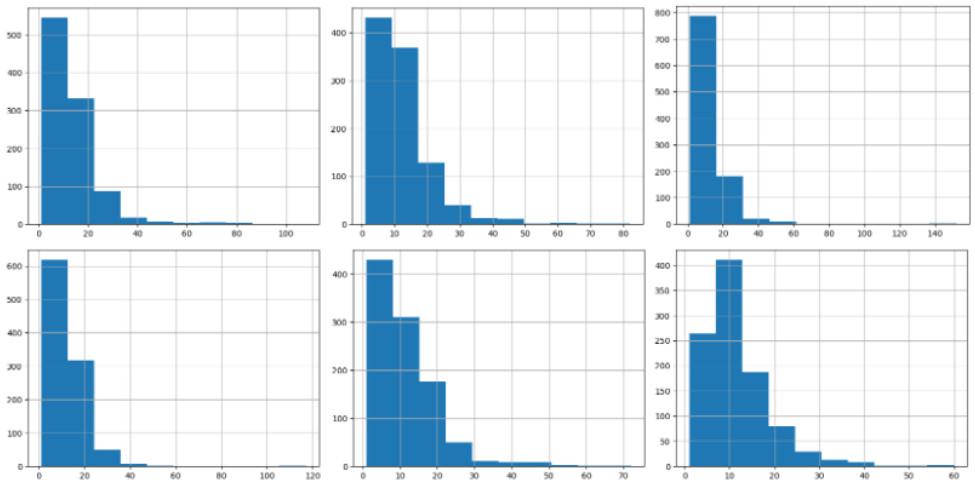
Tuy nhiên tập dữ liệu không có độ dài giống nhau nên sau khi xây dựng bộ từ điển chúng ta cần thực hiện padding cho mỗi câu trong tập dữ liệu.

```
tokenizer_data_text
```

```
[[4, 56, 346, 411, 823, 735, 9, 40, 120, 428],
 [4, 47, 346, 347, 654, 720, 8, 41, 100, 133],
 [4, 5360, 3, 1300, 169, 146, 2351, 3, 146, 71, 16, 12, 508, 824],
 [4, 83, 104, 3, 531, 1206, 308, 3, 86, 2594, 3649],
 [291, 284, 20, 1766, 230, 40, 34, 62, 250, 3650],
 [291, 231, 3651, 311, 230, 41, 34, 45, 232, 231, 355, 231],
 [5361, 40, 940, 169, 749, 164, 476, 63, 655, 19],
 [721, 41, 143, 11, 2595, 124, 5, 39, 43, 141, 11],
 [369, 205, 10, 248, 75, 3, 9, 234, 198, 20, 198, 62, 1431, 546, 3, 53],
 [60, 182, 5, 248, 59, 3, 8, 141, 198, 1767, 45, 1023, 25, 3, 21],
```

Hình 2.5. Độ dài mỗi câu trước khi padding

Padding có nghĩa là thêm các giá trị đặc biệt (thường là 0) vào các chuỗi token để đảm bảo rằng tất cả chuỗi có cùng độ dài. Để padding phù hợp ta cần xác định được maxlength thông qua việc visualization tập dữ liệu. Sau khi visualize ta thấy tập dữ liệu trung bình ở khoảng 500 nên ta lấy maxlen = 512 để padding.



Hình 2.6. Visualization tập dữ liệu

```
[ ]  vec_data

array([[   4,   56,  346, ...,    0,    0,    0],
       [   4,   47,  346, ...,    0,    0,    0],
       [   4, 5360,    3, ...,    0,    0,    0],
       ...,
       [ 159,  379,   48, ...,    0,    0,    0],
       [  31, 5359,  102, ...,    0,    0,    0],
       [  32, 5359,  102, ...,    0,    0,    0]], dtype=int32)
```

Hình 2.7. Độ dài mỗi câu sau khi padding

Cuối cùng ta sẽ chia tập dữ liệu thành các tập train, test và validation để tiến hành huấn luyện mô hình.

```
input data.shape:  (13856, 512)
data_vocab_size:  9346
training sample:  9975
validation sample:  2772
test sample:  1109
```

Hình 2.8. Chia tập dữ liệu

2.3. Xây dựng mô hình CNN và BiLSTM

Sau khi chuẩn bị tập dữ liệu và tiền xử lý dữ liệu, em sẽ tiến hành xây dựng mô hình để dự đoán. Ở đây em sử dụng mô hình Convolutional Neural Network kết hợp với Bidirectional-LSTM.

Trước khi xây dựng các lớp model em sẽ định nghĩa các siêu tham số và bộ khởi tạo như sau:

```
def generate_model():
  dropout_threshold = 0.2
  input_dim = data_vocab_size
  output_dim = 32
  input_length = 512
  initializer = tf.keras.initializers.GlorotNormal()
```

Hình 2.9. Siêu tham số và bộ khởi tạo

- **Dropout_threshold**: tỷ lệ dropout để tránh overfitting.
- **Input_dim** là kích thước từ vựng của dữ liệu đầu vào, em sẽ để nó bằng với data_vocab_size để đảm bảo rằng lớp Embedding có thể ánh xạ tất cả các từ trong tập từ vựng một cách chính xác và hiệu quả.
- **Input_length** sẽ tương đương với max_length trong quá trình padding tạo ra các véc tơ.
- **Initializer** là bộ khởi tạo trọng số, ở đây em sử dụng Glorot Normal (hay còn gọi là Xavier Normal). Đây là một phương pháp khởi tạo trọng số hiệu quả, giúp duy trì sự ổn định của gradient và cải thiện quá trình học của mạng nơ rơn.

Tiếp theo em tạo ra feature là một lớp Embedding, chuyển đổi các giá trị số nguyên đầu vào (tương ứng với các từ trong từ vựng) thành các vector nhúng có kích thước cố định. Lớp Embedding này giúp biểu diễn từ trong không gian vector có ý nghĩa ngữ nghĩa.

```
input_layer = Input(shape=(input_length))
feature = Embedding(input_dim=input_dim, output_dim=output_dim, input_length=input_length, embeddings_initializer="GlorotNormal")(input_layer)
```

Hình 2.10. Khởi tạo Embedding

Sau khi có được lớp feature Embedding ở trên em sẽ tiến hành xây dựng các feature từ CNN và Bi-LSTM.

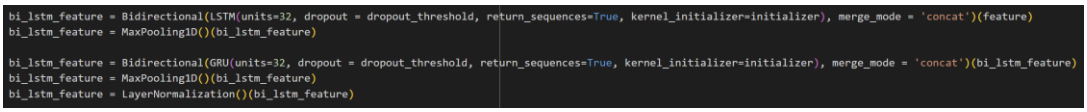
```
cnn_feature = Conv1D(filters=32, kernel_size=3, padding = 'same', activation='relu')(feature)
cnn_feature = MaxPooling1D()(cnn_feature)
cnn_feature = Dropout(dropout_threshold)(cnn_feature)
cnn_feature = Conv1D(filters=32, kernel_size=3, padding = 'same', activation='relu')(cnn_feature)
cnn_feature = MaxPooling1D()(cnn_feature)
cnn_feature = LayerNormalization()(cnn_feature)
cnn_feature = Dropout(dropout_threshold)(cnn_feature)
```

Hình 2.11. Tạo các lớp mạng CNN

Ở đây em sẽ sử dụng các lớp tích chậm 1D như Con1D và MaxPooling1D vì đây là bài toán về xử lý các chuỗi văn bản, được biểu diễn dưới dạng các chuỗi các

chỉ số nguyên (sau khi nhúng vào không gian véc tơ bởi lớp Embedding). So với các lớp tích chập 2D thì các lớp tích chập 1D có ít tham số hơn (chỉ cần một chiều không gian để áp dụng kernel) và đòi hỏi ít tài nguyên tính toán hơn, làm cho chúng phù hợp cho việc xử lý chuỗi dài.

Tiếp theo em tạo thêm các lớp LSTM và GRU có tính chất song song hai chiều (Bidirectional) cùng với các lớp MaxPooling1D và LayerNormalization để xử lý và chuẩn hoá đầu ra



Hình 2.12. Tạo lớp mạng BiLSTM

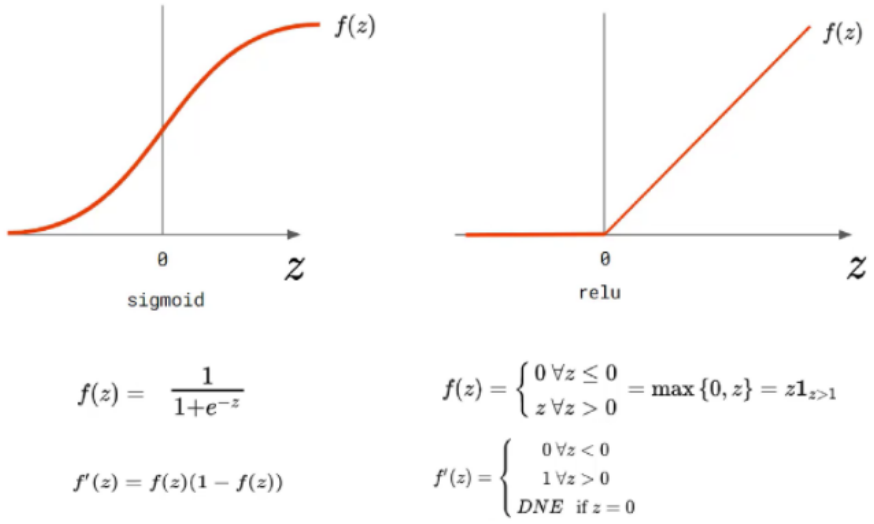
Em sẽ áp dụng các lớp Bi-LSTM để học các đặc trưng từ chuỗi đầu vào theo cả hai hướng từ trái qua phải và từ phải qua trái. Sau đó sử dụng MaxPooling1D để giảm độ dài của chuỗi và giữ lại các đặc trưng quan trọng. Áp dụng thêm lớp GRU Bidirectional để học các đặc trưng từ đầu ra của LSTM và lặp lại MaxPooling1D để tiếp tục giảm độ dài của chuỗi. Cuối cùng là chuẩn hoá đầu ra bằng LayerNormalization để ổn định quá trình huấn luyện.

Sau khi có được hai feature CNN và Bi-LSTM em sẽ tiến hành kết hợp các feature và tạo lớp phân loại.

```
combine_feature = tf.keras.layers.Concatenate()([cnn_feature, bi_lstm_feature])
combine_feature = GlobalMaxPooling1D()(combine_feature)
combine_feature = LayerNormalization()(combine_feature)

classifier = Dense(90, activation = 'relu')(combine_feature)
classifier = Dropout(0.2)(classifier)
classifier = Dense(70, activation = 'relu')(classifier)
classifier = Dropout(0.2)(classifier)
classifier = Dense(50, activation = 'relu')(classifier)
classifier = Dropout(0.2)(classifier)
classifier = Dense(30, activation = 'relu')(classifier)
classifier = Dropout(0.2)(classifier)
classifier = Dense(3, activation = 'softmax')(classifier)

model = tf.keras.Model(inputs = input_layer, outputs = classifier)

return model
```

Hình 2.13. Nối các lớp đặc trưng

Ở đây em sử dụng ReLU để làm hàm kích hoạt giữa các lớp. Hàm ReLU thường được ưa chuộng hơn trong các mạng neural vì nó giúp giảm thiểu vấn đề vanishing gradient, đơn giản và hiệu quả hơn về mặt tính toán so với hàm Sigmoid.



Hình 2.14. So sánh hàm Sigmoid và hàm ReLU

Có thể thấy ở hàm Sigmoid, khi |z| tăng thì f(z) có xu hướng giảm xuống 0 và tăng lên 1.

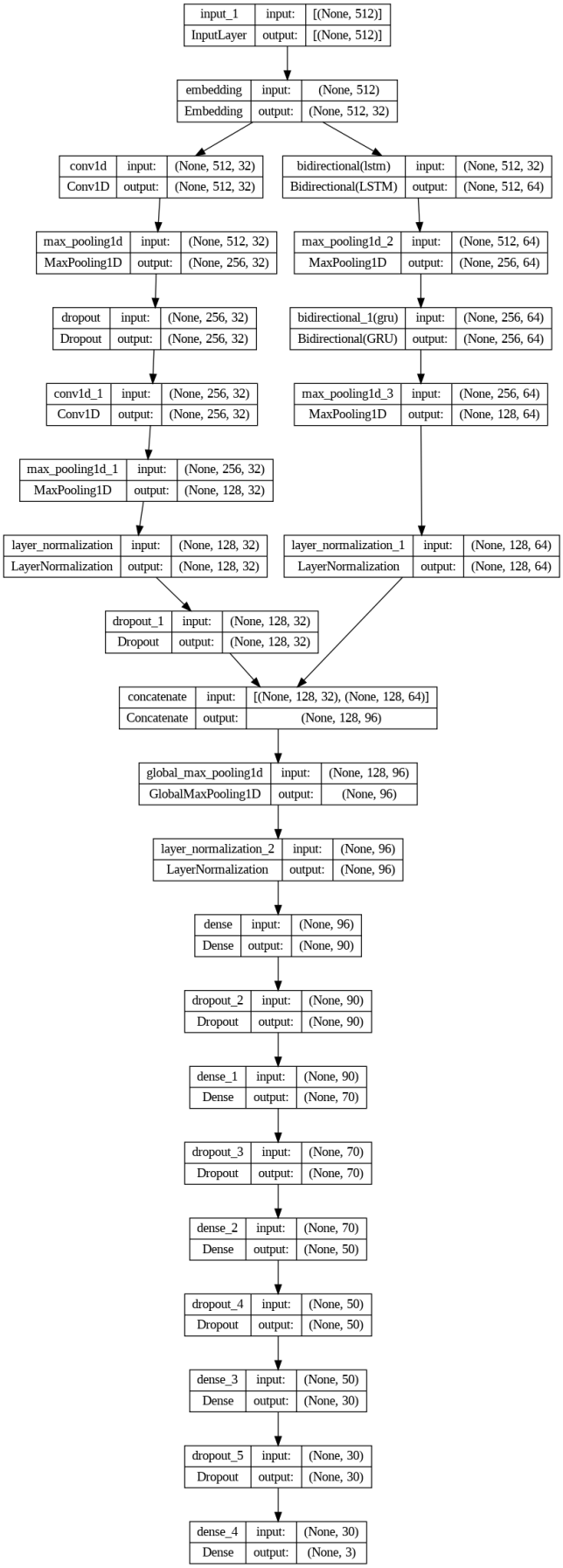
=> Đạo hàm f'(z) giảm về 0 và không cập nhật được tham số

Giá trị f(z) này tiếp tục được sử dụng cho các lớp ẩn phía sau và khi mạng càng sâu, mô hình càng khó train.

=> Dẫn đến hiện tượng Vanishing Gradient, hiện tượng này xảy ra hầu hết ở các model Deep Learning

Còn ở hàm ReLU, vì giá trị của f'(z) luôn bằng 1 hoặc 0, không phụ thuộc vào giá trị z nên không tồn tại đạo hàm tại z = 0 (trong thực tế khi z = 0 đặt đạo hàm thành 0.1 hoặc 0.5). Do đó, gradient có thể được truyền ngược hiệu quả qua nhiều lớp trong mạng neural.

=> Giải quyết được vấn đề Vanishing Gradient


input_1 | input: | [(None, 512)]
InputLayer | output: | [(None, 512)]
embedding | input: | (None, 512)
Embedding | output: | (None, 512, 32)
conv1d | input: | (None, 512, 32)
Conv1D | output: | (None, 512, 32)
bidirectional(lstm) | input: | (None, 512, 32)
Bidirectional(LSTM) | output: | (None, 512, 64)
max_pooling1d | input: | (None, 512, 32)
MaxPooling1D | output: | (None, 256, 32)
max_pooling1d_2 | input: | (None, 512, 64)
MaxPooling1D | output: | (None, 256, 64)
dropout | input: | (None, 256, 32)
Dropout | output: | (None, 256, 32)
bidirectional_1(gru) | input: | (None, 256, 64)
Bidirectional(GRU) | output: | (None, 256, 64)
conv1d_1 | input: | (None, 256, 32)
Conv1D | output: | (None, 256, 32)
max_pooling1d_3 | input: | (None, 256, 64)
MaxPooling1D | output: | (None, 128, 64)
max_pooling1d_1 | input: | (None, 256, 32)
MaxPooling1D | output: | (None, 128, 32)
layer_normalization | input: | (None, 128, 32)
LayerNormalization | output: | (None, 128, 32)
layer_normalization_1 | input: | (None, 128, 64)
LayerNormalization | output: | (None, 128, 64)
dropout_1 | input: | (None, 128, 32)
Dropout | output: | (None, 128, 32)
concatenate | input: | [(None, 128, 32), (None, 128, 64)]
Concatenate | output: | (None, 128, 96)
global_max_pooling1d | input: | (None, 128, 96)
GlobalMaxPooling1D | output: | (None, 96)
layer_normalization_2 | input: | (None, 96)
LayerNormalization | output: | (None, 96)
dense | input: | (None, 96)
Dense | output: | (None, 90)
dropout_2 | input: | (None, 90)
Dropout | output: | (None, 90)
dense_1 | input: | (None, 90)
Dense | output: | (None, 70)
dropout_3 | input: | (None, 70)
Dropout | output: | (None, 70)
dense_2 | input: | (None, 70)
Dense | output: | (None, 50)
dropout_4 | input: | (None, 50)
Dropout | output: | (None, 50)
dense_3 | input: | (None, 50)
Dense | output: | (None, 30)
dropout_5 | input: | (None, 30)
Dropout | output: | (None, 30)
dense_4 | input: | (None, 30)
Dense | output: | (None, 3)

Hình 2.15. Kiến trúc mô hình

2.4. Huấn luyện và đánh giá mô hình

Ta sẽ tiến hành huấn luyện mô hình, ở đây em train 10 epochs, batch_size=128. Train xong em sẽ tiến hành lưu model thành file *.h5 để deploy lên trang web Mạng xã hội.

```
callback_model = tf.keras.callbacks.ModelCheckpoint('/content/drive/MyDrive/Colab Notebooks/data_uit-vsmec/model_cnn_bilstm.h5', monitor='val_loss')
history = model.fit(x = X_train, y = y_train, validation_data = (X_val, y_val), epochs = 10, batch_size = 128, callbacks = [callback_model])
```

Hình 2.15. Huấn luyện mô hình

Kết quả sau khi train xong 10 epochs

```
Epoch 1/10
78/78 [==============================] - ETA: 0s - loss: 1.0961 - accuracy: 0.3754/usr/local/lib/python3.10/dist-packages/keras/src/eng
  saving_api.save_model(
78/78 [==============================] - 37s 230ms/step - loss: 1.0961 - accuracy: 0.3754 - val_loss: 1.0870 - val_accuracy: 0.4066
Epoch 2/10
78/78 [==============================] - 12s 152ms/step - loss: 1.0859 - accuracy: 0.4016 - val_loss: 1.0798 - val_accuracy: 0.4066
Epoch 3/10
78/78 [==============================] - 8s 100ms/step - loss: 1.0609 - accuracy: 0.4211 - val_loss: 1.0051 - val_accuracy: 0.4553
Epoch 4/10
78/78 [==============================] - 8s 104ms/step - loss: 0.9576 - accuracy: 0.5091 - val_loss: 0.9132 - val_accuracy: 0.5292
Epoch 5/10
78/78 [==============================] - 8s 101ms/step - loss: 0.8370 - accuracy: 0.5927 - val_loss: 0.8762 - val_accuracy: 0.5786
Epoch 6/10
78/78 [==============================] - 6s 80ms/step - loss: 0.7076 - accuracy: 0.6943 - val_loss: 0.8837 - val_accuracy: 0.6111
Epoch 7/10
78/78 [==============================] - 6s 78ms/step - loss: 0.5819 - accuracy: 0.7715 - val_loss: 0.8829 - val_accuracy: 0.6255
Epoch 8/10
78/78 [==============================] - 7s 90ms/step - loss: 0.4677 - accuracy: 0.8245 - val_loss: 0.9397 - val_accuracy: 0.6356
Epoch 9/10
78/78 [==============================] - 6s 76ms/step - loss: 0.3852 - accuracy: 0.8644 - val_loss: 1.0114 - val_accuracy: 0.6356
Epoch 10/10
78/78 [==============================] - 7s 88ms/step - loss: 0.3189 - accuracy: 0.8867 - val_loss: 1.2355 - val_accuracy: 0.6263
```

Hình 2.16. Kết quả huấn luyện

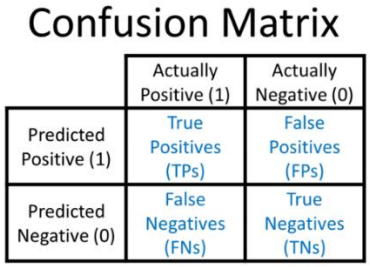
Tiếp theo sẽ đánh giá mô hình thông qua phép đo Presicion, Recall, F1_Score, đây là các phép đo thường được sử dụng cho bài toán phân loại.

Khi thực hiện bài toán phân loại, có 4 trường hợp của dự đoán có thể xảy ra:

- **True Positive (TP):** đối tượng ở lớp Positive, mô hình phân đối tượng vào lớp Positive (dự đoán đúng)
- **True Negative (TN):** đối tượng ở lớp Negative, mô hình phân đối tượng vào lớp Negative (dự đoán đúng)
- **False Positive (FP):** đối tượng ở lớp Negative, mô hình phân đối tượng vào lớp Positive (dự đoán sai) – Type I Error
- **False Negative (FN):** đối tượng ở lớp Positive, mô hình phân đối tượng vào lớp Negative (dự đoán sai) – Type II Error

Bốn trường hợp trên thường được biểu diễn dưới dạng ma trận hỗn loạn (confusion matrix). Chúng ta có thể tạo ra ma trận này sau khi dự đoán xong trên

tập dữ liệu thử nghiệm và rồi phân loại các dự đoán vào một trong bốn trường hợp.[13]



Hình 2.17. Confusion Matrix [13]

Ba độ đo chủ yếu để đánh giá một mô hình phân loại là Accuracy, Presicion và Recall.[13]

- Accuracy được định nghĩa là tỷ lệ phần trăm dự đoán đúng cho dữ liệu thử nghiệm. Nó có thể được tính toán dễ dàng bằng cách chia số lần dự đoán đúng cho tổng số lần dự đoán

$$\text{Accurary} = \frac{\text{TP} + \text{TN}}{\text{TP+TN+FP+FN}}$$

- Precision kiểm tra xem có bao nhiêu kết quả thật là kết quả tích cực trong tổng số các kết quả được dự đoán tích cực

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

- Recall: kiểm tra các kết quả dự đoán tích cực chính xác trong số các kết quả tích cực

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

- F1_score: là trung bình hài hòa của Accuracy và recall. Sự đóng góp phụ thuộc vào giá trị beta, nếu sự đóng của cả 2 là như nhau thì ta có:

$$F_1 - score = 2.\frac{\text{Precision. Recall}}{\text{Precision} + \text{Recall}}$$

Đánh giá mô hình qua các độ đo trên ta được tỉ lệ khoảng 63%

```
Accuracy: 0.633904
Precision: 0.635933
Recall: 0.624920
F1 score: 0.629378
```

Hình 2.18. Kết quả đánh giá model

2.5. Dự đoán với mô hình đã huấn luyện

```python
def preprocess_raw_input(raw_input, tokenizer):
  input_text_pre = list(tf.keras.preprocessing.text.text_to_word_sequence(raw_input))
  input_text_pre = " ".join(input_text_pre)
  input_text_pre_accent = ViTokenizer.tokenize(input_text_pre)
  print("Text preprocessed:", input_text_pre_accent)
  tokenized_data_text = tokenizer.texts_to_sequences([input_text_pre_accent])
  vec_data = pad_sequences(tokenized_data_text, padding='post', maxlen = 512)
  return vec_data

def inference_model(input_feature, model):
  output = model(input_feature).numpy()[0]
  result = output.argmax()
  conf = float(output.max())
  label_dict = {'tiêu cực':0, 'trung lập':1, 'tích cực':2}
  label = list(label_dict.keys())
  return label[int(result)], conf

def prediction(raw_input, tokenizer, model):
  input_model = preprocess_raw_input(raw_input, tokenizer_data)
  result, conf = inference_model(input_model, model)

  return result, conf
```

Hình 2.19. Các hàm để dự đoán đầu vào mới

Các hàm `preprocess_raw_input`, `inference_model`, `prediction` giúp xử lý đầu vào, thực hiện suy luận với mô hình và dự đoán kết quả. Sau đó sẽ thực hiện load mô hình đã huấn luyện và tokenizer từ tệp và sử dụng để thực hiện dự đoán trên một câu đầu vào.

```python
my_model = generate_model()
my_model = load_model('/content/drive/MyDrive/Colab Notebooks/data_uit-vsmec/model_cnn_bilstm.h5')

with open(r"/content/drive/MyDrive/Colab Notebooks/data_uit-vsmec/tokenizer_data.pkl", "rb") as input_file:
  my_tokenizer = pickle.load(input_file)

print(prediction("đánh chết mẹ mày giờ", my_tokenizer, my_model))
```

```
Text preprocessed: đánh chết mẹ mày giờ
('tiêu cực', 0.9301496148109436)
```

Hình 2.20. Load mô hình và tokenizer để dự đoán đầu vào mới

Có thể thấy kết quả chính xác với tỷ lệ khoảng 93%. Ta thử thêm một số input khác thì kết quả trả ra cũng tương đối chính xác.

```
cái đụ má thằng chó kia
Text preprocessed: cái đụ má thằng chó kia
tiêu cực

cái dm mày nói ít thôi để người khác nói nữa
Text preprocessed: cái dm mày nói ít thôi để người khác nói nữa
tiêu cực

thứ súc sinh
Text preprocessed: thứ súc_sinh
tiêu cực

con đĩ mẹ mày luôn
Text preprocessed: con đĩ mẹ mày luôn
tiêu cực
```

Hình 2.21. Kết quả đầu vào mới

CHƯƠNG III. PHÁT TRIỂN MÔ HÌNH VÀO TRANG WEB MẠNG XÃ HỘI

3.1. Sơ lược về trang web Mạng xã hội

3.1.1. Xây dựng

a. Giao diện Frontend

Phần giao diện phía người dùng công nghệ em sử dụng chủ yếu là ReactJs. ReactJs là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook. Nó được sử dụng để xây dựng các giao diện người dùng (UI) cho ứng dụng web, giúp cho việc phát triển trở nên dễ dàng, hiệu quả và bảo trì tốt hơn. Các tính năng nổi bật của ReactJs bao gồm khả năng tạo các thành phần giao diện có thể tái sử dụng, quản lý trạng thái ứng dụng và tối ưu hóa hiệu suất thông qua việc sử dụng Virtual DOM[8].

Một số thành phần cơ bản trong ReactJs

- Component:

- Functional Components: Các component được định nghĩa bằng hàm, không có state riêng và sử dụng hooks để quản lý state và side effects.
- Class Components: Các component được định nghĩa bằng class, có state riêng và sử dụng lifecycle methods.

- JSX (JavaScript XML): JSX (nói ngắn gọn là JavaScript extension) là một React extension giúp chúng ta dễ dàng thay đổi cây DOM bằng các HTML-style code đơn giản. Và kể từ lúc ReactJS browser hỗ trợ toàn bộ những trình duyệt Web hiện đại, ta có thể tự tin sử dụng JSX trên bất kỳ trình duyệt nào mà ta đang làm việc.

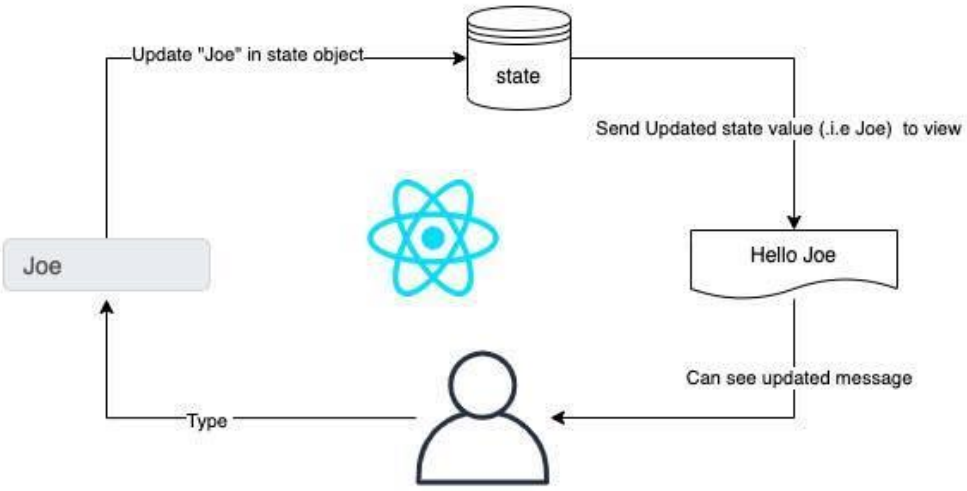
- Props (Properties):

- Props được sử dụng để gửi dữ liệu đến component.
- Mọi component được coi là một hàm javascript thuần khiết (Pure Function).
- Trong ReactJS, props tương đương với các tham số của hàm javascript thuần khiết.
- Props là bất biến (không thể thay đổi được). Bởi vì điều này được phát triển trong khái niệm về các hàm thuần khiết. Trong các hàm thuần

khiết, chúng ta không thể thay đổi dữ liệu của các tham số. Vì vậy, cũng không thể thay đổi dữ liệu của prop trong ReactJS.

- State:

- State cũng tương tự như props, nhưng nó là của riêng component và được kiểm soát hoàn toàn bởi chúng và state có thể thay đổi được và mỗi khi state thì đổi thì component đó sẽ được render lại.
- State chỉ tồn tại trong class component hoặc được quản lý bởi hooks trong functional component.



Hình 3.1. Vòng đời của State trong ReactJs

- Lifecycle Methods:

- Các phương thức được gọi tại các giai đoạn khác nhau trong vòng đời của một component.
- Các giai đoạn bao gồm mounting, updating và unmounting.

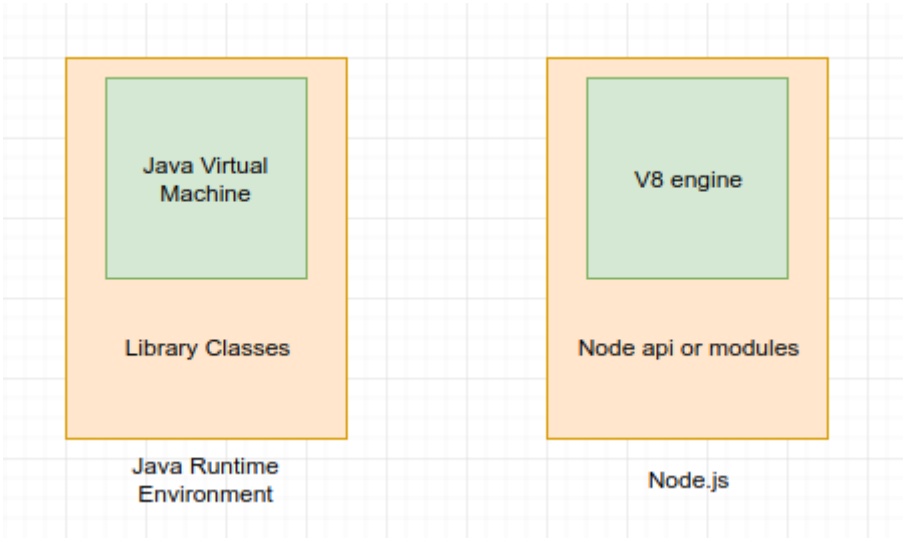
- Hook: Các hàm đặc biệt cho phép sử dụng state và các tính năng khác của React trong functional components. Một số hooks phổ biến là *useState, useEffect,…*

- *useState* là một hook trong React được sử dụng để khởi tạo và quản lý trạng thái (state) trong Functional Components. Hook này cho phép ta thêm trạng thái vào Functional Components mà trước đây chỉ có thể được quản lý trong Class Components. useState trả về một mảng với hai phần tử: State variable (biến trạng thái) và State updater function (hàm cập nhật trạng thái).

- *useEffect* là một trong những hooks quan trọng trong React, được sử dụng để thực hiện các tác vụ phụ (side effects) trong Functional Components. Các tác vụ phụ bao gồm lệnh gọi API, thay đổi trạng thái, đăng ký và hủy đăng ký sự kiện, và các tác vụ không thuộc về việc render giao diện người dùng. *useEffect* giúp ta thực hiện các tác vụ này tại các thời điểm cụ thể trong vòng đời của component.

b. Xử lý Backend

Về phần xử lý backend thì em sử dụng một trong những framework phổ biến hiện nay là Node.Js. Node.js là một môi trường runtime JavaScript mã nguồn mở, đa nền tảng, cho phép thực thi mã JavaScript trên phía server. Được xây dựng trên nền tảng V8 JavaScript engine của Google Chrome, Node.js được thiết kế để xây dựng các ứng dụng mạng có khả năng mở rộng cao[8].



Hình 3.2. So sánh giữa Node.js và Java

Node.js được biết đến với một số điểm nổi bật như sau

- Event-Driven và Non-Blocking I/O:

- Event-Driven: Node.js sử dụng một kiến trúc dựa trên sự kiện, nơi một vòng lặp sự kiện (event loop) duy nhất quản lý tất cả các yêu cầu đến. Điều này giúp Node.js xử lý nhiều yêu cầu đồng thời mà không cần phải tạo ra các luồng (threads) riêng lẻ.
- Non-Blocking I/O: Các hoạt động nhập/xuất (I/O) như đọc/ghi tệp hoặc giao tiếp mạng không chặn luồng thực thi chính. Thay vào đó, chúng được thực hiện không đồng bộ, cho phép xử lý các yêu cầu khác trong khi chờ kết quả.

- Single-Threaded nhưng Scalable: Node.js chạy trên một luồng đơn (single-threaded) nhưng có thể xử lý hàng nghìn kết nối đồng thời nhờ vào mô hình non-blocking I/O và event loop. Điều này giúp tăng khả năng mở rộng mà không cần tăng số lượng luồng xử lý.

- Cross-Platform: Node.js hoạt động trên nhiều hệ điều hành khác nhau bao gồm Windows, macOS, và các bản phân phối của Linux. Điều này làm cho nó trở thành một công cụ linh hoạt và phổ biến trong phát triển phần mềm.

- High Performance: Được xây dựng trên V8 JavaScript engine của Google, Node.js tận dụng hiệu suất cao của V8 để thực thi mã JavaScript nhanh chóng. Hơn nữa, mô hình non-blocking I/O của Node.js giúp tối ưu hóa hiệu suất xử lý các yêu cầu I/O.

- NPM (Node Package Manager): NPM là trình quản lý gói đi kèm với Node.js, cung cấp hàng trăm nghìn module và thư viện sẵn có để sử dụng. NPM giúp quản lý các phụ thuộc của dự án dễ dàng và cho phép chia sẻ mã nguồn với cộng đồng.

- JSON Support: Node.js hỗ trợ JSON (JavaScript Object Notation) một cách tự nhiên, làm cho việc trao đổi dữ liệu giữa server và client trở nên dễ dàng và hiệu quả.

- Real-Time Web Applications: Node.js đặc biệt phù hợp cho việc xây dựng các ứng dụng web thời gian thực như chat, hệ thống thông báo, và các dịch vụ theo dõi thời gian thực.

- Microservices and Serverless Architectures: Node.js hỗ trợ phát triển các kiến trúc microservices và serverless, cho phép xây dựng các ứng dụng với các thành phần nhỏ, độc lập, dễ dàng triển khai và mở rộng.

Ngoài ra một số tính năng của trang web như trò chuyện, thông báo thì em sử dụng thư viện Socket.IO. Nó là một thư viện mạnh mẽ và linh hoạt cho phép các ứng dụng web có khả năng giao tiếp hai chiều (real-time) giữa client và server. Nó được xây dựng trên nền tảng Node.js và có thể dễ dàng tích hợp vào các ứng dụng web để tạo ra các tính năng như chat, thông báo tức thì, cập nhật dữ liệu trực tiếp và nhiều hơn nữa.

Một số tính năng chính của SocketIO

- **Real-time Communication:** Hỗ trợ giao tiếp hai chiều giữa client và server trong thời gian thực.

- **Cross-browser:** Tương thích với nhiều trình duyệt khác nhau, bao gồm cả những trình duyệt cũ.
- **Auto-reconnection:** Tự động kết nối lại nếu kết nối bị mất.
- **Room and Namespace:** Hỗ trợ phân chia các kết nối vào các phòng (room) và không gian tên (namespace) để quản lý giao tiếp hiệu quả.
- **Binary Support:** Hỗ trợ gửi và nhận dữ liệu nhị phân.
- **Scalability:** Dễ dàng mở rộng để hỗ trợ hàng nghìn kết nối đồng thời với sự hỗ trợ từ các adapter như Redis.

Cách hoạt động của SocketIO

- **Client-side library:** Chạy trên trình duyệt và kết nối đến server qua WebSocket hoặc các phương thức dự phòng khác.
- **Server-side library:** Chạy trên Node.js và quản lý kết nối từ client.

Cuối cùng là về cơ sở dữ liệu, ở đây em sử dụng một hệ quản trị cơ sở dữ liệu phi quan hệ (NoSQL) mã nguồn mở đó chính là MongoDB. Nó được phát triển để lưu trữ và truy vấn dữ liệu theo mô hình tài liệu (document-oriented model). MongoDB lưu trữ dữ liệu dưới dạng các tài liệu JSON linh hoạt và cho phép lưu trữ dữ liệu có cấu trúc không đồng nhất.

MongoDB có khả năng mở rộng tốt, cho phép lưu trữ và xử lý các tải trọng công việc lớn và phức tạp. Nó hỗ trợ các tính năng như replica set (bộ sao chép dữ liệu), sharding (phân vùng dữ liệu) và indexing (chỉ mục) để tăng hiệu suất và đảm bảo sẵn sàng cao[9].

Với MongoDB, việc thay đổi cấu trúc dữ liệu trở nên dễ dàng và linh hoạt, giúp phát triển ứng dụng nhanh chóng và linh hoạt hơn. Nó phù hợp cho các ứng dụng web, mobile, IoT và nhiều ngữ cảnh lưu trữ dữ liệu khác[7].

Dưới đây là một số cơ sở dữ liệu được xây dựng bằng MongoDB

```
import mongoose from "mongoose";

const userSchema = mongoose.Schema(
  {
    firstName: {
      type: String,
      required: true,
    },
    lastName: {
      type: String,
      required: true,
    },
    fullName: String,
    avatar: {
      type: String,
      default: "https://res.cloudinary.com/drwm3i3g4/image/
    },
```

Hình 3.3. Cơ sở dữ liệu Người dùng

```
import mongoose from "mongoose";

const postSchema = mongoose.Schema(
  {
    userPost: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
      required: true,
    },
    audience: {
      type: String,
      enum: ["public", "friends", "private"],
      default: "public",
    },
    assets: [
      {
        media_type: String,
        url: String,
      },
    ],
    content: {
      type: String,
      required: true,
    },
    tagsPeople: [{ type: mongoose.Schema.Types.ObjectId, ref: "User", default: [] }],
    likes: [{ type: mongoose.Schema.Types.ObjectId, ref: "User", default: [] }],
    shares: [{ type: mongoose.Schema.Types.ObjectId, ref: "User", default: [] }],
    comments: [{ type: mongoose.Schema.Types.ObjectId, ref: "Comment", default: [] }],
  },
  { timestamps: true },
  {
    toJSON: { virtuals: true }, // So `res.json()` and other `JSON.stringify()` functio
    toObject: { virtuals: true }, // So `console.log()` and other functions that use `t
  }
);
```

Hình 3.4. Cơ sở dữ liệu Bài viết

```
import mongoose from "mongoose";

const commentSchema = mongoose.Schema(
  {
    postId: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "Post",
      required: true,
    },
    userComment: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
      required: true,
    },
    content: {
      type: String,
      required: true,
    },
  },
  { timestamps: true }
);

export default mongoose.model("Comment", commentSchema);
```

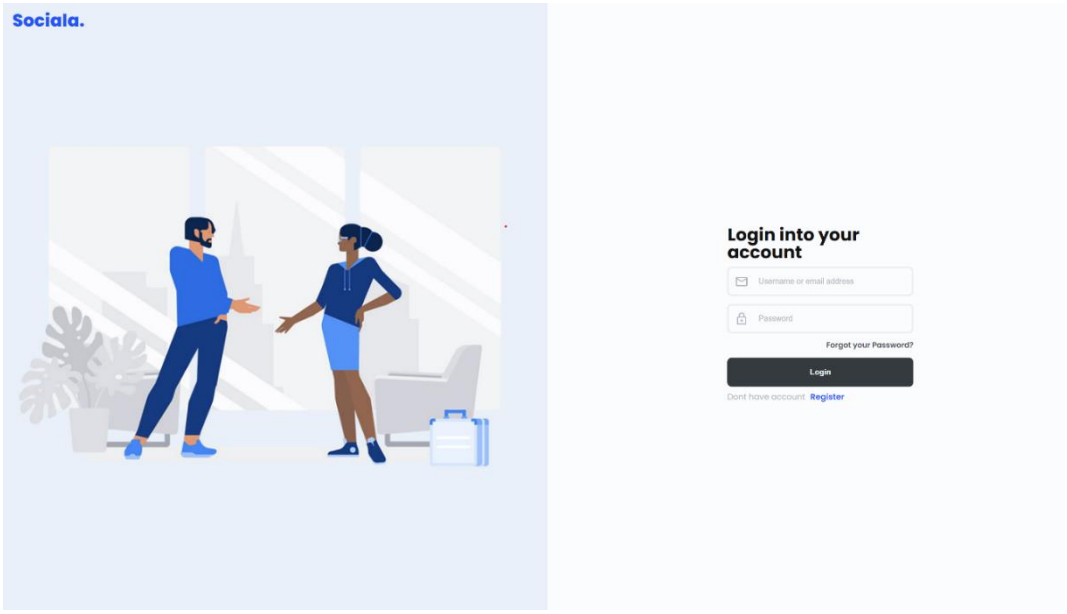
Hình 3.5. Cơ sở dữ liệu Bình luận

```
import mongoose from "mongoose";

const MessageSchema = new mongoose.Schema(
  {
    type: {
      type: String,
      enum: ["message", "asset", "file", "notification"],
      default: "message",
    },
    sender: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
      required: true,
    },
    content: {
      type: String,
      trim: true,
    },
    asset: { url: String, media_type: String },
    conversation: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "Conversation",
    },
    isDeleted: {
      type: Boolean,
      default: false,
    },
  },
  { timestamps: true }
);

export default mongoose.model("Message", MessageSchema);
```

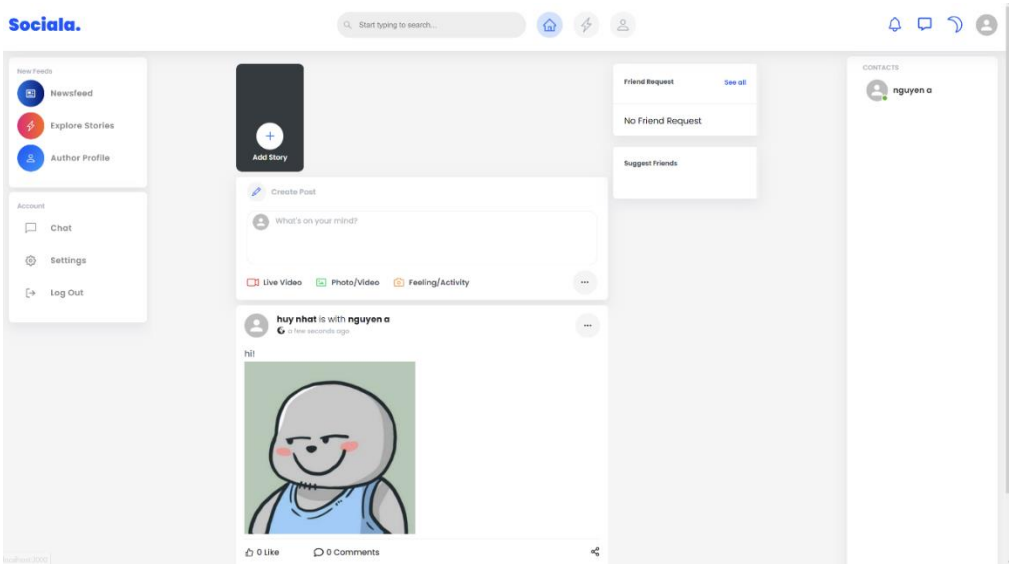
Hình 3.6. Cơ sở dữ liệu Trò chuyện

3.1.2. Giao diện

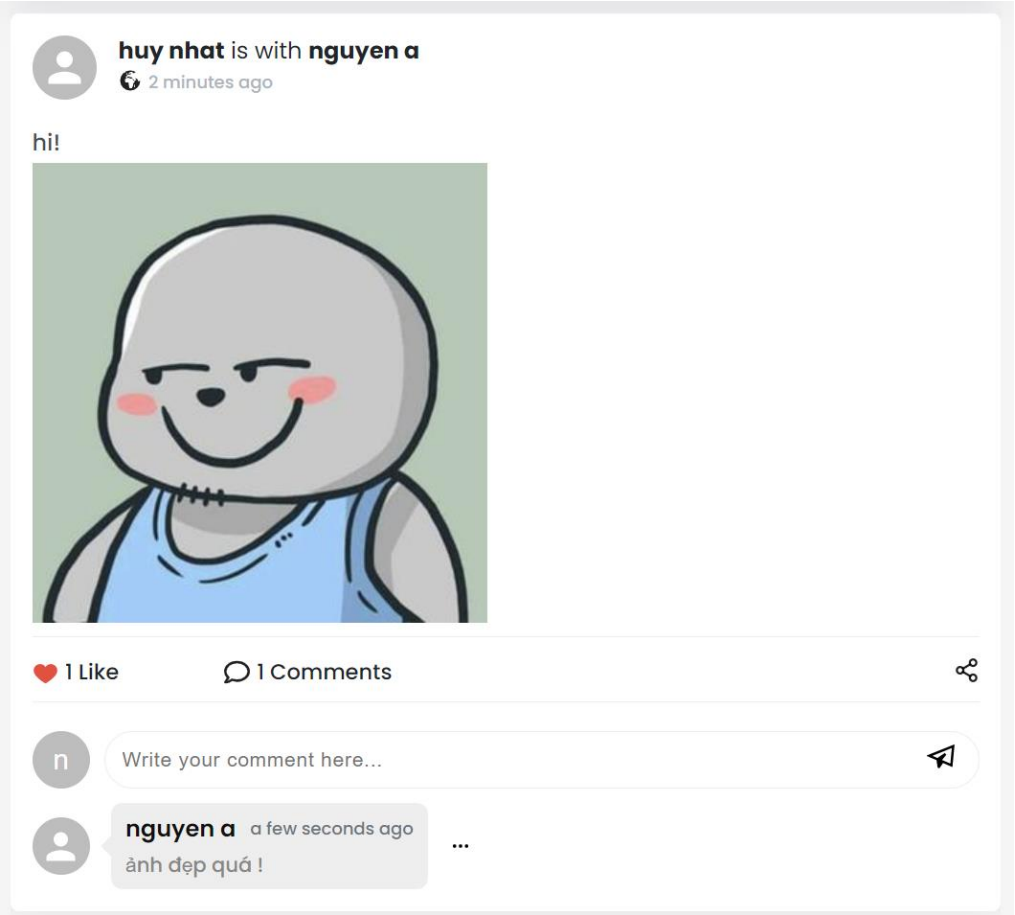
Dưới đây là một số giao diện của trang web Mạng xã hội



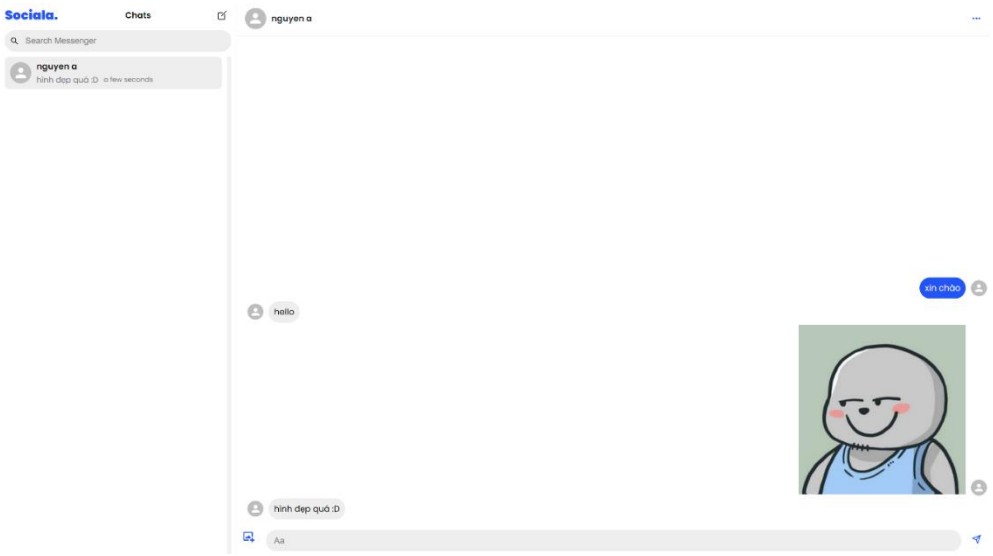
Hình 3.7. Giao diện đăng nhập



Hình 3.8. Giao diện Trang chủ



Hình 3.9. Giao diện Bài viết



Hình 3.10. Giao diện Trò chuyện

3.2. Xây dựng API cho mô hình

Để xây dựng API cho mô hình em sử dụng một framework là FastAPI. Đây là một framework hiện đại và nhanh cho việc xây dựng các API web. Một số tính năng nổi bật của FastAPI như:

- Nhanh: hiệu suất cao khi so sánh với NodeJS và Go
- Code nhanh: tăng tốc độ phát triển tính năng từ 200% tới 300%
- Ít lỗi hơn: Giảm khoảng 40% những lỗi phát sinh bởi con người (nhà phát triển).
- Trực giác tốt hơn: Được các trình soạn thảo hỗ tuyệt vời. Completion mọi nơi. Ít thời gian gỡ lỗi.
- Dễ dàng: Được thiết kế để dễ dàng học và sử dụng. Ít thời gian đọc tài liệu.
- Ngắn: Tối thiểu code bị trùng lặp. Nhiều tính năng được tích hợp khi định nghĩa tham số. Ít lỗi hơn.
- Tăng tốc: Có được sản phẩm cùng với tài liệu (được tự động tạo) có thể tương tác.
- Được dựa trên các tiêu chuẩn: Dựa trên (và hoàn toàn tương thích với) các tiêu chuẩn mở cho APIs : OpenAPI (trước đó được biết đến là Swagger) và JSON Schema[12].

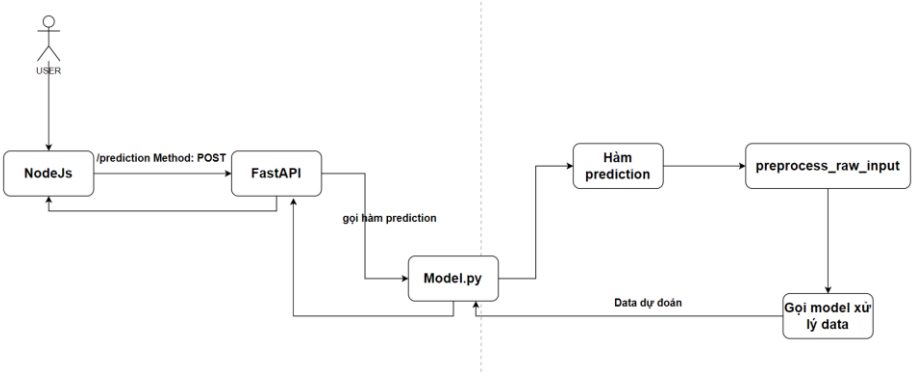
Dưới đây em sẽ sử dụng FastAPI để tạo một API cho model:

```python
from fastapi import FastAPI
from model import prediction
from pydantic import BaseModel
app = FastAPI()

class RequestDTO(BaseModel):
    content: str

@app.post("/prediction")
def get_prediction(request: RequestDTO):
    return prediction(request.content)
```

Hình 3.11. Tạo FastAPI

Cách FastAPI hoạt động sẽ như sau:

- Người dùng tạo bình luận sẽ gửi yêu cầu tới NodeJs
- NodeJs sẽ gọi qua FastAPI với đường dẫn là **/prediction** và **Method: POST**
- FastAPI sẽ gọi hàm prediction từ file model.py
- Từ hàm prediction trong file model.py sẽ gọi hàm preprocess_raw_input để tiền xử lý dữ liệu và gọi model xử lý data
- Sau đó sẽ gửi lại data dự đoán cho file model.py để trả về FastAPI và trả về NodeJs.



Hình 3.12. Mô hình FastAPI hoạt động

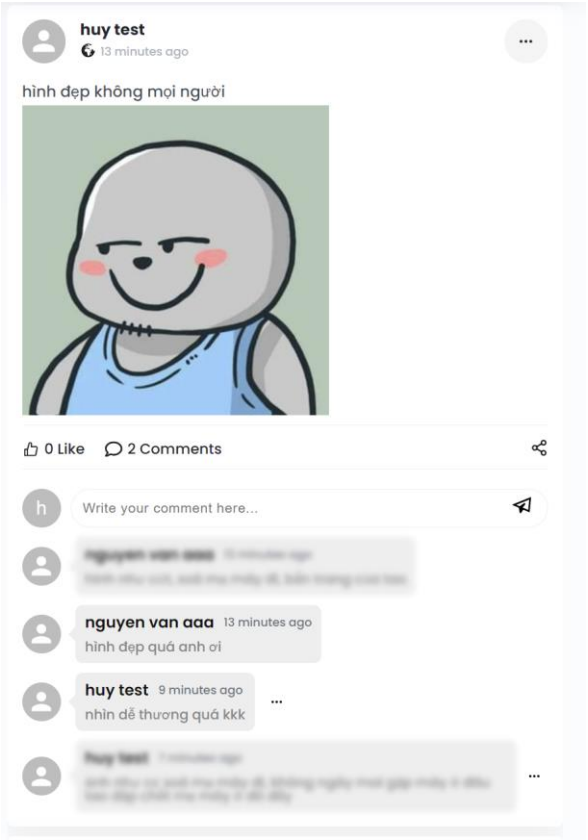
FastAPI cung cấp cho ta một documentation, trang này cho phép người dùng thực hiện các yêu cầu trực tiếp từ giao diện web. Điều này rất hữu ích để thử nghiệm và kiểm tra nhanh các endpoint mà không cần sử dụng các công cụ khác hiện nay như Postman hay cURL.

Em sẽ tiến hành test và thử nghiệm API

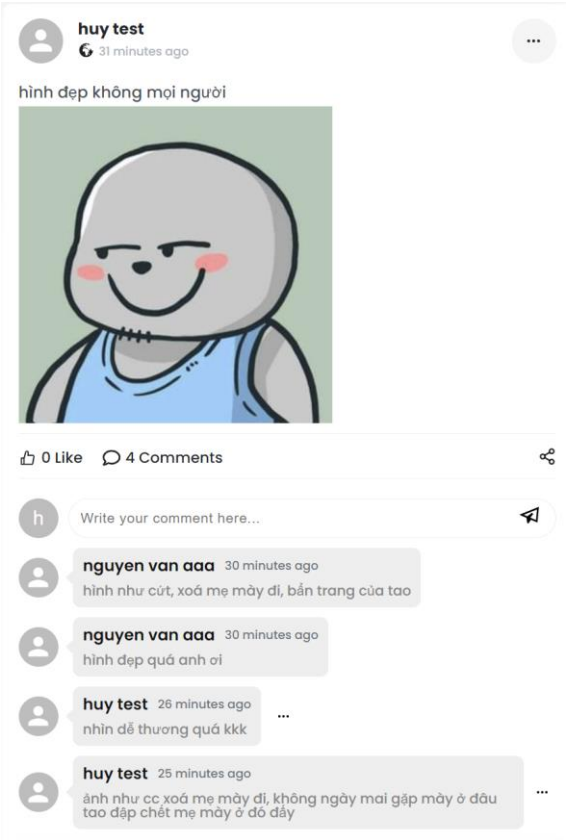
Hình 3.13. Kiểm thử API qua docs của FastAPI

3.3. Thử nghiệm vào trang web Mạng xã hội

Ở đây em có lấy ý tưởng từ trang xã hội được xem là lớn nhất hiện nay là Facebook, khi người dùng bình luận những câu văn câu từ nhạy cảm thì hệ thống sẽ làm mờ bình luận đó. Và khi ấn vào bình luận đó sẽ bỏ làm mờ. Chủ bài viết có thể dựa vào đó và xoá bình luận tiêu cực mà họ muốn.

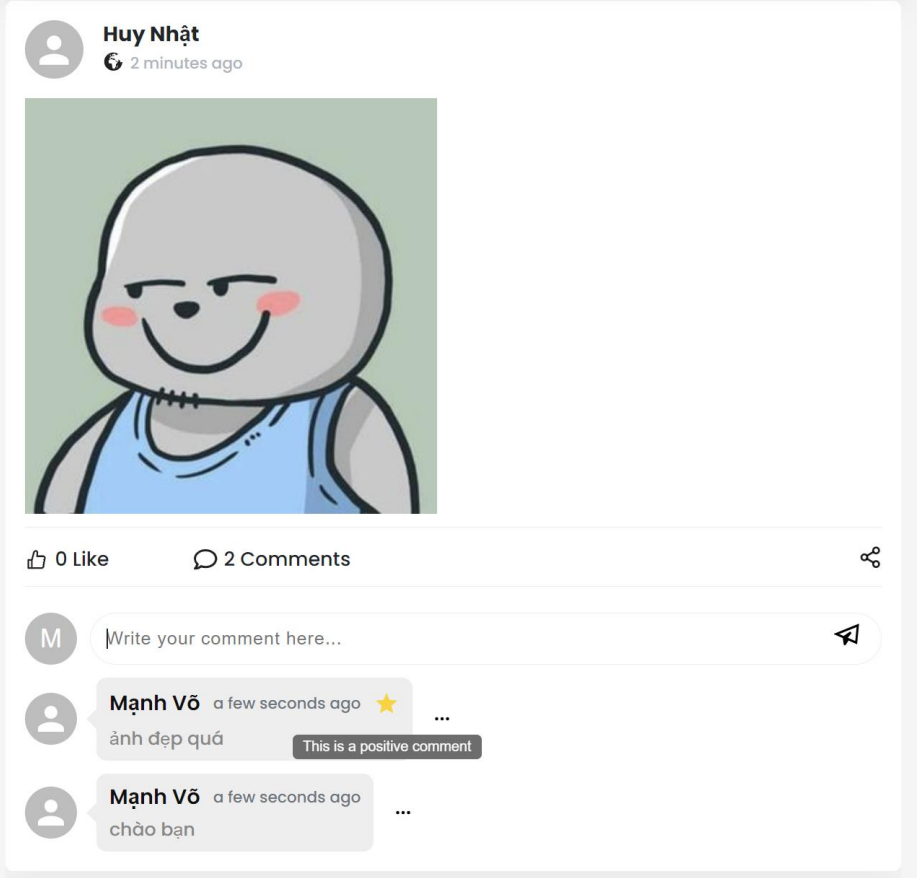


Hình 3.14. Bình luận bị làm mờ



Hình 3.15. Bình luận loại bỏ làm mờ

Đối với những bình luận tích cực thì bình luận sẽ được hiển thị thêm một ký tự đặc biệt để nhận dạng và còn lại là những bình luận trung lập thì sẽ giữ nguyên không thêm gì cả.



Hình 3.16. Nhận diện bình luận tích cực và trung lập

KẾT LUẬN VÀ KIẾN NGHỊ

1. Kết quả đạt được

Trong quá trình nghiên cứu và ứng dụng mô hình, đề tài đã đạt được những kết quả sau:

- Mô hình được xây dựng và triển khai đã cho thấy việc phân loại và đánh giá bình luận của người dùng tương đối chính xác khi đã nhận diện được đúng các bình luận tiêu cực.
- Mô hình đã được tích hợp thành công vào một hệ thống website Mạng xã hội.
- Kết quả chạy thử sau khi tích hợp vào trang web Mạng xã hội cho thấy mô hình có thể hoạt động ổn định và chính xác.

Ngoài ra sau khi thực hiện xong đề tài này em đã tiếp nhận thêm được nhiều kiến thức hay và bổ ích về mặt kiến thức và cả kỹ năng mềm.

Về kỹ năng mềm em đã học được nhiều kỹ năng sau:

- Kỹ năng hỏi đáp và tìm kiếm tài liệu
- Kỹ năng đọc sách
- Kỹ năng giải quyết vấn đề
- Kỹ năng quản lý thời gian

Còn về mặt kiến thức, sau đề tài này em cũng đã học được rất nhiều kiến thức mới mà 4 năm đại học qua em chưa được tiếp xúc:

- Học được những kiến thức về Trí tuệ nhân tạo – AL, Học máy – Machine Learning và Học sâu – Deep Learning
- Những kiến thức về mô hình mạng neural, hay cụ thể là Mạng tích chập – CNN, các mạng LSTM và BiLSTM.
- Biết được những kỹ thuật xử lý ngôn ngữ tự nhiên như Tiền xử lý văn bản – Tokenization, vector hoá văn bản và các phương pháp đánh giá hiệu quả của mô hình NLP.
- Kỹ năng làm việc với các công cụ như Google Colab và thư viện như TensorFlow hay Keras.

2. Những vấn đề còn tồn đọng

Mặc dù đạt được nhiều kết quả khả quan, đề tài vẫn còn một số vấn đề tồn đọng cần được giải quyết:

- Một số tập dữ liệu huấn luyện có sự mất cân bằng giữa các loại bình luận (tích cực, tiêu cực, trung tính), dẫn đến khả năng mô hình có thể bị thiên vị đối với các loại bình luận chiếm đa số.
- Mô hình CNN-BiLSTM có độ phức tạp cao, yêu cầu tài nguyên tính toán lớn, điều này có thể gây khó khăn trong việc triển khai trên các hệ thống có tài nguyên hạn chế.
- Hiện tại, mô hình mới chỉ được thử nghiệm trên một số trang web mạng xã hội cụ thể. Để có thể áp dụng rộng rãi, cần phải kiểm tra và điều chỉnh mô hình để phù hợp với nhiều nền tảng và ngữ cảnh khác nhau.
- Một số bình luận có chứa từ lóng, viết tắt hoặc ngữ cảnh phức tạp mà mô hình hiện tại chưa xử lý tốt. Điều này đòi hỏi phải cải thiện khả năng xử lý ngôn ngữ tự nhiên của mô hình.

3. Hướng phát triển tiếp theo của đề tài

Để khắc phục những vấn đề tồn đọng và nâng cao hiệu quả của mô hình, hướng phát triển tiếp theo của đề tài sẽ tập trung vào các điểm sau:

- Cải thiện và mở rộng tập dữ liệu, tiến hành thu thập thêm dữ liệu bình luận từ nhiều nguồn khác nhau, đảm bảo tính đa dạng và cân bằng giữa các loại bình luận. Áp dụng các kỹ thuật xử lý dữ liệu như oversampling hoặc sử dụng các thuật toán xử lý dữ liệu không cân bằng.
- Tìm kiếm các phương pháp tối ưu hóa mô hình để giảm độ phức tạp, cải thiện tốc độ xử lý mà không làm giảm hiệu quả phân loại. Có thể áp dụng các kỹ thuật như pruning, quantization hoặc sử dụng các mô hình nhẹ hơn như MobileNet.
- Tiến hành thử nghiệm và điều chỉnh mô hình trên nhiều trang web mạng xã hội và nền tảng khác nhau để đảm bảo tính khả thi và hiệu quả trong nhiều ngữ cảnh.

- Nghiên cứu và tích hợp các kỹ thuật xử lý ngôn ngữ tự nhiên tiên tiến như BERT, GPT để cải thiện khả năng hiểu và phân tích ngữ cảnh của bình luận, đặc biệt là các bình luận có ngôn ngữ phức tạp.
- Xây dựng các cơ chế học tập tự động cho mô hình, cho phép mô hình tự cập nhật và cải thiện theo thời gian dựa trên phản hồi từ người dùng và dữ liệu mới.

TÀI LIỆU THAM KHẢO

[1] Krishna Rungta. "*TensorFlow in 1 Day: Make your own Neural Network*", Nhà xuất bản Amazon Digital Services LLC - KDP Print US, 2018.

[2] Palash Goyal, Pandey Sumit & Karan Jain. "*Deep Learning for Natural Language Processing: Creating Neural Networks with Python*", Nhà xuất bản Apress, 2018.

[3] Nguyễn Thanh Tuấn. "*Deep Leaning Basic version 2*", Nhà xuất bản Nguyễn Thanh Tuấn, 2020.

[4] Francois Chollet. *"Deep Learning with Python"*, Nhà xuất bản Manning, 2017.

[5] Yoon Kim. *"Convolutional Neural Networks for Sentence Classification"*, Nhà xuất bản Association for Computational Linguistics, 2014.

[6] Yoav Goldberg. *"Neural Network Methods for Natural Language Processing (Synthesis Lectures on Human Language Technologies, 37)"*, Nhà xuất bản Morgan & Claypool, 2017.

[7] Kyle Banker, Peter Bakkum, Shaun Verch, Doug Garrett, Tim Hawkins. *"MongoDB in Action: Covers MongoDB version 3.0 (2nd Edition)"*, Nhà xuất bản Manning, 2016.

[8] *Tài liệu về ReactJs và Node.Js*. Link: https://www.w3schools.com (truy cập 05/2024).

[9] *Tài liệu về MongoDB*. Link: https://viblo.asia/p/tim-hieu-ve-mongodb-4P856ajGlY3 (truy cập 05/2024).

[10] *NLP – Natural Language processing*. Link: https://aws.amazon.com/vi/what-is/nlp/ (truy cập 03/2024).

[11] *LSTM – Long Short-Term Memory*. Link: https://nttuan8.com/bai-14-long-short-term-memory-lstm/ (truy cập 04/2024).

[12] *FastAPI*. Link: https://fastapi.tiangolo.com/vi/ (truy cập 05/2024).

[13] *Evaluation Model*. Link: https://www.miai.vn/2020/06/16/oanh-gia-model-ai-theo-cach-mi-an-lien-chuong-2-precision-recall-va-f-score/ (truy cập 05/2024).

[14] *Tensorflow*. Link: https://www.tensorflow.org/api_docs/python/tf (truy cập 04/2024).

[15] *Tokenization*. Link: https://www.geeksforgeeks.org/nlp-how-tokenizing-text-sentence-words-works/ (truy cập 04/2024).

[16] *Convolutional Neural Networks for Natural Language processing*. Link: https://viblo.asia/p/understanding-convolutional-neural-networks-for-natural-language-processing-bJzKmW0Bl9N#_how-does-any-of-this-apply-to-nlp-3 (truy cập 04/2024).

[17] Datasets – UIT Vietnamese Social Media Emotion Corpus version 1.0 (UIT-VSMEC version 1.0). Link: https://nlp.uit.edu.vn/datasets#h.p_FxJKMfavctsJ (truy cập 05/2024).