

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI



ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC
NGÀNH KHOA HỌC MÁY TÍNH

ĐỀ TÀI:
NGHIÊN CỨU VÀ XÂY DỰNG MÔ HÌNH ĐÁNH GIÁ CẢM
XÚC BÌNH LUẬN PHIM VỚI PYTORCH

GVHD:	TS. ĐỖ MẠNH HÙNG
Sinh viên thực hiện:	NGUYỄN QUANG MINH
Mã sinh viên:	2020603771
Lớp:	2020DHKHMT02 – K15

Hà Nội – Năm 2024

NGUYỄN QUANG MINH

NGÀNH KHOA HỌC MÁY TÍNH

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI



ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC
NGÀNH KHOA HỌC MÁY TÍNH

ĐỀ TÀI:
NGHIÊN CỨU VÀ XÂY DỰNG MÔ HÌNH ĐÁNH GIÁ CẢM
XÚC BÌNH LUẬN PHIM VỚI PYTORCH

GVHD:	TS. ĐỖ MẠNH HÙNG
Sinh viên thực hiện:	NGUYỄN QUANG MINH
Mã sinh viên:	2020603771
Lớp:	2020DHKHM02 – K15

Hà Nội – Năm 2024

NGUYỄN QUANG MINH

NGÀNH KHOA HỌC MÁY TÍNH

MỤC LỤC

DANH MỤC HÌNH ẢNH	5
DANH MỤC TỪ VIẾT TẮT	6
LỜI CẢM ƠN	7
MỞ ĐẦU	8
CHƯƠNG 1. BÀI TOÁN SỬ DỤNG PYTHON/PYTORCH ĐỂ ĐÁNH GIÁ CẢM XÚC TÍCH CỰC/TIÊU CỰC CỦA BÌNH LUẬN PHIM	10
1.1. Các công trình nghiên cứu đã có của tác giả trong và ngoài nước.	10
1.1.1. Công trình “Sentiment Analysis of Movie Reviews” của tác giả Charlie Chengrui Zheng.....	10
1.1.2. Công trình “Deep Learning for Sentiment Analysis” của tác giả Bert Carremans.....	12
1.2. Những vấn đề còn tồn tại.	14
1.2.1. Dữ liệu rời rạc và thiếu cân bằng.....	15
1.2.2. Hiểu lầm ngữ cảnh.	15
1.2.3. Tính toàn diện và đa ngôn ngữ.....	15
1.2.4. Hướng giải quyết.....	16
1.3. Những vấn đề mà đề tài đồ án cần tập trung nghiên cứu giải quyết.....	16
1.3.1. Cải thiện cân bằng dữ liệu.....	16
1.3.2. Hiểu ngữ cảnh và biểu cảm.	16
1.3.3. Cải thiện tính toàn diện và đa ngôn ngữ.	17
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VÀ MỘT SỐ PHƯƠNG PHÁP, KỸ THUẬT CHÍNH ĐƯỢC SỬ DỤNG TRONG ĐỒ ÁN ĐỂ GIẢI QUYẾT BÀI TOÁN	18
2.1. Tổng quan về trí tuệ nhân tạo (Artificial Intelligence).	18
2.1.1. Các lĩnh vực của trí tuệ nhân tạo.....	19
2.1.2. Một số ứng dụng của trí tuệ nhân tạo.	19
2.2. Xử lý ngôn ngữ tự nhiên (Natural language processing).....	21
2.2.1. Tổng quan về xử lý ngôn ngữ tự nhiên.	21
2.2.2. Một số kỹ thuật xử lý ngôn ngữ tự nhiên phổ biến.....	25

2.3. Kỹ thuật phân tích quan điểm (Sentiment Analysis) trong xử lý ngôn ngữ tự nhiên.....	29
2.3.1. Tổng quan lợi ích của phân tích quan điểm.	30
2.3.2. Các dạng phân tích quan điểm chính.	30
2.3.3. Giới thiệu thuật toán xác định ngữ nghĩa theo ngữ cảnh (Contextual Semantic Search – CSS).....	31
2.4. Các gói, hàm và kỹ thuật sử dụng trong phân tích tình cảm.....	32
2.4.1. PyTorch và TorchText.	32
2.4.2. Tiền xử lý văn bản.....	32
2.4.3. Mô hình Recurrent Neural Network (RNN) với Long Short-Term Memory (LSTM).....	36
2.4.4. Kỹ thuật đóng gói tuần tự (Packed Sequence).....	39
2.4.5. Huấn luyện và đánh giá mô hình.	40
2.4.6. Dự đoán cảm xúc.	42
2.5. Giới thiệu Pytorch trong xử lý ngôn ngữ tự nhiên.....	43
2.5.1. Mô hình Transformer.	44
2.5.2. Tiền xử lý ngôn ngữ.	48
2.5.3. Huấn luyện mô hình và tối ưu hóa.....	49
2.5.4. Các ứng dụng của Pytorch.	52
CHƯƠNG 3. THỰC NGHIỆM XÂY DỰNG MÔ HÌNH	53
3.1. Bộ dữ liệu thực nghiệm.....	53
3.2. Chuẩn bị dữ liệu.	54
3.3. Định nghĩa mô hình phân tích tình cảm Recurrent Neural Network.....	55
3.4. Huấn luyện mô hình.	58
3.5. Xử lý từng câu hỏi để lấy ra các hình tròn chứa đáp án.	60
3.6. Một số so sánh cùng kết luận giữa đề án và công trình “Sentiment Analysis of Movie Reviews” của tác giả Charlie Chengrui Zheng.	61
KẾT LUẬN	64
TÀI LIỆU THAM KHẢO.....	65

DANH MỤC HÌNH ẢNH

Hình 2.1: Ví dụ về Tokenization với spaCy.....	34
Hình 2.2: Ví dụ về Embedding từ với Glove và Pytorch	36
Hình 2.3: Ví dụ RNN với pytorch.	38
Hình 2.4: Ứng dụng của BERT trong pytorch.	47
Hình 2.5: Ứng dụng GPT trong pytorch.	48
Hình 2.6: Ví dụ Cross-Entropy Loss trong pytorch.	50
Hình 2.7: Ví dụ SDG trong Pytorch.	51
Hình 3.1: Lệnh điều hướng đến thư mục chứa dữ liệu.	53
Hình 3.2: Chuẩn bị và xử lý trước dữ liệu.	54
Hình 3.3: Phân chia tập dữ liệu.	55
Hình 3.4: Tạo một lô đào tạo.	55
Hình 3.5: Định nghĩa mô hình phân tích tình cảm RNN.	56
Hình 3.6: Truyền tham số cho mô hình.	57
Hình 3.7: Lấy các trọng số nhúng được đào tạo trước.	57
Hình 3.8: Hàm tối ưu hóa và hàm mất mát.	58
Hình 3.9: Hàm độ chính xác nhị phân.	58
Hình 3.10: Hàm đào tạo.	58
Hình 3.11: Hàm đánh giá.	59
Hình 3.12: Hàm trợ giúp epoch_time.	59
Hình 3.13: Tiền xử lý đầu vào.	60
Hình 3.14: Hàm in kết quả.	60
Hình 3.15: Kết quả thực nghiệm.	61

DANH MỤC TỪ VIẾT TẮT

Viết tắt	Tiếng Anh	Tiếng Việt
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên
SGD	Stochastic Gradient Descent	Giảm độ dốc ngẫu nhiên
RNN	Recurrent Neural Network	Mạng nơ-ron hồi quy
GPT	Generative Pre-trained Transformer	Bộ chuyển đổi đào tạo trước tự sinh
LSTM	Long Shot-Term Memory	Mạng bộ nhớ ngắn hạn dài hạn
BERT	Bidirectional Encoder Representations from Transformers	Biểu diễn bộ mã hóa hai chiều từ bộ chuyển đổi
NLTK	Natural Language Toolkit	Bộ công cụ ngôn ngữ tự nhiên

LỜI CẢM ƠN

Em xin chân thành cảm ơn quý thầy, cô trường Đại Học Công Nghiệp Hà Nội, đặc biệt là các giảng viên trong Khoa Công nghệ thông tin, đã tạo điều kiện và hỗ trợ em trong quá trình thực hiện đề tài này.

Em xin bày tỏ lòng biết ơn sâu sắc đối với giảng viên hướng dẫn - TS. Đỗ Mạnh Hùng. Thầy đã tận tình hướng dẫn, cung cấp những kiến thức quý báu và lời khuyên hữu ích, giúp em hoàn thành tốt nhiệm vụ. Bên cạnh đó, em cũng xin cảm ơn các bạn sinh viên trong Khoa Công nghệ thông tin đã đóng góp ý kiến, giúp em thực hiện đề tài đạt hiệu quả hơn.

Thông qua việc thực hiện đề tài "Nghiên cứu và xây dựng mô hình đánh giá cảm xúc bình luận phim với Pytorch", em đã rèn luyện được kỹ năng tư duy phân tích, xử lý dữ liệu và trình bày thông tin một cách logic và rõ ràng. Những kiến thức và kinh nghiệm thu thập từ đề tài này sẽ tiếp tục hỗ trợ em trong tương lai, không chỉ trong học tập mà còn trong sự nghiệp và cuộc sống.

Một lần nữa, em xin chân thành cảm ơn sự hướng dẫn và định hướng của quý thầy cô và các bạn sinh viên khoa Công nghệ thông tin. Em rất mong nhận được những ý kiến đóng góp để đề tài được hoàn thiện hơn.

Em xin chân thành cảm ơn!

Sinh viên thực hiện

Nguyễn Quang Minh

MỞ ĐẦU

Lý do chọn đề tài

Xu hướng công nghệ: Sự phát triển nhanh chóng của trí tuệ nhân tạo và học sâu đã mở ra nhiều cơ hội ứng dụng mới mẻ, trong đó có phân tích cảm xúc từ văn bản.

Nhu cầu thực tiễn: Các nền tảng xem phim trực tuyến ngày càng phổ biến, việc đánh giá cảm xúc từ bình luận phim có thể giúp các nền tảng này cải thiện chất lượng dịch vụ và nội dung.

Khả năng ứng dụng cao: Kết quả nghiên cứu có thể áp dụng ngay vào thực tế, giúp các nhà sản xuất phim hiểu rõ hơn về phản ứng của khán giả.

Mục đích nghiên cứu

Phát triển mô hình AI: Tạo ra một mô hình sử dụng Python và PyTorch để phân loại cảm xúc (tích cực/tiêu cực) từ bình luận phim.

Đánh giá hiệu quả: Đánh giá độ chính xác và hiệu quả của mô hình trong việc nhận diện cảm xúc từ văn bản.

Kết quả mong đợi

Thứ nhất: Xây dựng một mô hình học sâu có khả năng phân tích cảm xúc chính xác từ các bài đánh giá phim.

Thứ hai: So sánh, đánh giá ưu điểm và nhược điểm giữa các mô hình học sâu.

Thứ ba: Đưa ra giải pháp tối ưu trong việc phân loại cảm xúc từ văn bản.

Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: Các bình luận phim trên các nền tảng trực tuyến (ví dụ: Netflix, YouTube, IMDb).

Phạm vi nghiên cứu: Tập trung vào các bình luận bằng tiếng Anh, nhằm xây dựng mô hình có tính khả dụng cao cho các ngôn ngữ phổ biến.

Ý nghĩa khoa học và thực tiễn

Ý nghĩa khoa học: Góp phần phát triển lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) và trí tuệ nhân tạo (AI) trong phân tích cảm xúc từ văn bản. Đề tài cũng mở

ra hướng nghiên cứu mới cho việc ứng dụng học sâu (deep learning) trong xử lý ngôn ngữ.

Ý nghĩa thực tiễn: Kết quả nghiên cứu có thể được ứng dụng trong nhiều lĩnh vực khác nhau như truyền thông, marketing, và dịch vụ khách hàng, giúp các tổ chức hiểu rõ hơn về phản hồi của người dùng và cải thiện chiến lược kinh doanh.

CHƯƠNG 1. BÀI TOÁN SỬ DỤNG PYTHON/PYTORCH ĐỂ ĐÁNH GIÁ CẢM XÚC TÍCH CỰC/TIÊU CỰC CỦA BÌNH LUẬN PHIM

1.1. Các công trình nghiên cứu đã có của tác giả trong và ngoài nước.

1.1.1. Công trình “Sentiment Analysis of Movie Reviews” của tác giả Charlie Chengrui Zheng.

- Mục đích: Vì số lượng tập dữ liệu lớn nên bạn cần sử dụng các công cụ xử lý ngôn ngữ tự nhiên để phân loại cảm xúc của văn bản bằng các gói mạnh mẽ như nltk và scikit-learn.
- Bộ dữ liệu: Trong dự án này, tác giả đã thực hiện phân tích cảm xúc của các bài đánh giá phim từ bộ dữ liệu các bài đánh giá trên IMDB từ Bộ dữ liệu câu được gắn nhãn cảm xúc của UCI Machine Learning Repository. Trong bộ dữ liệu này, có 1000 bài đánh giá phim, bao gồm 500 bài tích cực (khen ngợi) và 500 bài tiêu cực (chỉ trích). Ví dụ, 'Một bộ phim rất, rất, rất chậm chạp, không mục đích về một chàng trai trẻ đau khổ, trôi dạt.' được đánh dấu là một đánh giá tiêu cực.
- Các mô-đun được sử dụng:
 - + Xử lý dữ liệu: Pandas
 - + Đại số tuyến tính: Numpy
 - + Xử lý ngôn ngữ tự nhiên: nltk
 - + Học máy: scikit-learn
 - + Trực quan: matplotlib
 - + Học sâu: torch
 - + Dải tiến trình: tqdm
- Lý luận bài toán và các bước được tác giả thực hiện:
 - + Trong văn bản thô, chúng ta có thể thấy nhiều từ không chứa ngữ nghĩa quan trọng. Các con số và dấu câu xuất hiện rất nhiều trong văn bản thô của tập dữ liệu nhưng không thể hiện tình cảm tích cực hay tiêu cực. Do

đó, tác giả đã loại bỏ các số và dấu câu. Để làm sạch dữ liệu hơn nữa, chúng ta cần phải bắt đầu các từ, để các từ có các biến đổi khác nhau có thể được tính là cùng một mã thông báo, bởi vì chúng truyền tải cùng một ngữ nghĩa. ví dụ: 'distress' và 'distressed' đều sẽ được bắt nguồn là 'distress'. Sau khi tiền xử lý văn bản, chúng ta có 2 danh sách dữ liệu: 'labels' là danh sách các mục tiêu được phân loại của chúng ta; 'preprocessed' là các đoạn văn cần được phân loại. Đối với các câu trong 'preprocessed', tác giả dịch văn bản thô thành văn bản hoàn toàn không thể đọc được. Ví dụ, 'A very, very, very slow-moving, aimless movie about a distressed, drifting young man.' được tiền xử lý thành 'a veri veri veri slow move aimless movi about a distress drift young man'. Bạn có thể thắc mắc rằng: tác giả dịch văn bản thô thành văn bản không thể đọc được này vì họ muốn mỗi từ truyền tải ngữ nghĩa quan trọng. Vậy tại sao không loại bỏ các từ lặp lại vì chúng không truyền tải ngữ nghĩa quan trọng nhưng rất thường xuyên xuất hiện, chẳng hạn như 'a' và 'about'? Trong phần trích xuất tính năng tiếp theo, Charlie Chengrui Zheng sẽ sử dụng TF-IDF để xử lý các từ dừng và lặp lại đó.

+ Sau khi tiền xử lý văn bản, chúng ta sẽ trích xuất các tính năng từ dữ liệu đã được làm sạch của mình. Chúng ta sẽ sử dụng bộ véc-tơ TF-IDF vectorizer làm nhúng từ của chúng ta để vector hóa và chuẩn hóa văn bản. TF-IDF là viết tắt của thuật ngữ tần số tài liệu nghịch đảo tần số (term frequency-inverse document frequency). Nó đánh giá tầm quan trọng của từ đối với tài liệu trong kho dữ liệu. TF-IDF làm cho dữ liệu trở thành mô hình của chúng ta vì nó chuẩn hóa thuật ngữ tần số, hoặc đơn giản là số từ. Nó cũng làm giảm độ nhiễu của các từ dừng.

+ Sau khi trích xuất tính năng, chúng ta có ma trận thuật ngữ tài liệu có trọng số Tf-IDF được lưu trữ ở định dạng Hàng thừa thớt nén. Mỗi nhãn là tình cảm của câu này. Trong đó '1' có nghĩa là tích cực và '0' có nghĩa là tiêu cực. Nhưng để làm cho dữ liệu phù hợp với mô hình của chúng ta, tác

giả đã chia dữ liệu của chúng ta thành các tính năng và nhãn (mục tiêu). Scikit-learn `train_test_split` xáo trộn ngẫu nhiên dữ liệu và chia chúng thành tập huấn luyện và tập thử nghiệm. Trong trường hợp cụ thể này, tác giả sử dụng 1/5 của toàn bộ tập dữ liệu của tập thử nghiệm và 4/5 còn lại làm tập huấn luyện. Dưới đây là mã cho toàn bộ quá trình tiền xử lý.

+ Bởi vì mục tiêu là phân loại và nhị phân nên các tính năng không có phân phối giả định, các mô hình chúng ta có thể sử dụng để phân loại văn bản là Logistics Regression, Stochastic Gradient Descent Classifier (SGDClassifier), Support Vector Classifier (SVC) và Neural Network (MLPClassifier). Bởi vì dữ liệu bình luận của chúng ta thừa thớt và rời rạc nên SVC và SGD rất hữu ích. Trong số 3 loại Bộ phân loại Naive Bayes Classifiers (Bernoulli, Multinomial and Gaussian), chúng ta cần chọn Multinomial, vì bộ dữ liệu được chuẩn hóa bởi TF-IDF. Bộ dữ liệu không phù hợp với phân phối Gaussian cũng như Bernoulli.

1.1.2. Công trình “Deep Learning for Sentiment Analysis” của tác giả Bert Carremans.

Trích lời tác giả: “Cách đây một thời gian tôi đã cố gắng dự đoán cảm xúc của các dòng tweet trong một hạt nhân Kaggle khác bằng cách sử dụng văn bản và các bộ phân loại cơ bản. Trong cuốn sổ tay này, tôi muốn thử xem liệu chúng ta có thể vượt trội hơn những mô hình này bằng mô hình học sâu hay không.”.

- Ông đã làm như sau:

- + Thay đổi phù hợp với mô hình học sâu với Keras
- + Xác định và giải quyết tình trạng trang bị quá mức
- + Sử dụng từ nhúng
- + Xây dựng trên mô hình đã được huấn luyện trước

- Về bộ dữ liệu này, ban đầu được lấy từ thư viện dữ liệu cho mọi người của Crowdfunder. Một bộ dữ liệu phân tích tình cảm về các vấn đề của từng hãng hàng không lớn của Hoa Kỳ bao gồm 14641 đánh giá. Dữ liệu Twitter được thu thập từ tháng 2 năm 2015 và những người đóng góp được yêu cầu

phân loại các tweet tích cực, tiêu cực và trung lập trước, sau đó phân loại các lý do tiêu cực (chẳng hạn như "chuyến bay muộn" hoặc "dịch vụ thô lỗ").

- Các gói được dùng trong dự án này:
 - + Chuẩn bị dữ liệu: sklearn, nltk, keras
 - + Huấn luyện mô hình: keras.models, keras.layers, keras. Regularizers
- Lý luận bài toán:
 - + Chúng ta thiết lập một số tham số sẽ được sử dụng trong toàn bộ bài toán. Số lượng từ trong từ điển, kích thước của bộ xác nhận (validation set), Số kỷ nguyên (epochs) chúng ta thường bắt đầu luyện tập, kích thước batch size. Chúng ta đọc dữ liệu tweet trong csv và thực hiện trộn ngẫu nhiên. Cách tốt nhất là trộn dữ liệu trước khi phân tách giữa huấn luyện và kiểm tra tập. Bằng cách đó, các lớp tình cảm đều được phân tích bổ sung trên tập huấn luyện và tập kiểm tra. Chúng ta sẽ chỉ giữ các bản văn cột đầu vào và các hằng hàng không làm nhãn.
 - + Chuẩn bị dữ liệu:

Điều đầu tiên chúng ta sẽ làm là xóa các từ dừng. Những từ này không có giá trị gì trong việc dự đoán cảm xúc. Hơn nữa, vì chúng ta muốn xây dựng một mô hình có thể sử dụng cho các hằng hàng không khác, nên chúng ta sẽ xóa các tên riêng. Việc đánh giá hiệu suất của mô hình cần được thực hiện thông qua một thử nghiệm riêng biệt. Như vậy, chúng ta có thể ước tính mức độ cụ thể của mô hình. Điều này được thực hiện bằng phương pháp `train_test_split` của `scikit-learn`.

Để sử dụng văn bản đầu vào cho mô hình, trước tiên chúng ta cần chuyển đổi các từ của tweet thành mã thông báo, điều đơn giản này có nghĩa là chuyển đổi các từ thành phần nguyên tham chiếu đến một chỉ mục trong từ dict. Ở đây, tôi sẽ chỉ lưu lại những từ thường gặp nhất trong tập tin. Sau khi tạo từ điển, chúng ta có thể chuyển đổi văn bản thành danh sách các số nguyên chỉ mục. Việc này được thực hiện bằng phương thức

`text_to_sequences` của `Tokenizer`. Sử dụng `to_categorical` trong `Keras` để chuyển đổi các lớp mục tiêu thành số, để các lớp này có thể được mã hóa một lần. Ngoài ra, chúng ta làm sạch văn bản bằng cách áp dụng các bộ lọc và chuyển các từ sang chữ thường. Các từ được phân tách bằng dấu cách. Bây giờ bộ xác thực này sẽ được sử dụng để đánh giá hiệu suất của mô hình khi chúng ta điều chỉnh các tham số của mô hình.

+ Mô hình học sâu:

Mô hình cơ sở: Ta bắt đầu với một mô hình có 2 lớp kết nối dày đặc bao gồm 64 phần tử ẩn. `input_shape` cho lớp đầu tiên sử dụng số từ chúng ta cho phép trong từ điển và chúng ta đã tạo các tính năng được mã hóa một lần cho lớp này. Vì chúng ta cần dự đoán 3 loại tình cảm khác nhau nên lớp cuối cùng có 3 phần tử ẩn. Hàm kích hoạt `softmax` đảm bảo tổng hiệu suất bằng 1.

Vì dự án này là dự đoán đa lớp, đơn nhãn, chúng ta sử dụng `categorical_crossentropy` làm hàm mất mát và `softmax` làm hàm kích hoạt cuối cùng. Chúng ta điều chỉnh mô hình trên dữ liệu đào tạo còn lại và xác thực trên tập xác thực. Chúng ta chạy trong một số kỷ nguyên (`epochs`) được xác định trước và sẽ xem khi nào mô hình bắt đầu quá khớp (`over-fitting`).

Xử lý quá khớp (`over-fitting`):

Tùy chọn 1: giảm kích thước mạng bằng cách loại bỏ các lớp hoặc giảm số lượng phần tử ẩn trong các lớp.

Tùy chọn 2: thêm chính quy hóa, tức là thêm chi phí vào hàm mất mát cho các trọng số lớn.

Tùy chọn 3: thêm các lớp bỏ qua, sẽ loại bỏ ngẫu nhiên một số tính năng bằng cách đặt chúng thành 0.

1.2. Những vấn đề còn tồn tại.

Mục đích của việc thực hiện đề tài là sử dụng mô hình CNN và OpenCV để xây dựng một hệ thống nhằm hiểu rõ hơn về quy trình, cách thức hoạt động của hệ thống chấm điểm tự động của bộ giáo dục. Với mong muốn học tập, áp

dụng và củng cố lại kiến thức đã được học trong quá trình học tập tại trường Đại học công nghiệp Hà Nội. Qua đó, nghiêm túc nhìn nhận, đánh giá kiến thức, năng lực của các thành viên trong nhóm để kịp thời cải thiện các thiếu sót, lỗ hổng kiến thức, kinh nghiệm và chuẩn bị cho đồ án tốt nghiệp.

1.2.1. Dữ liệu rời rạc và thiếu cân bằng.

- Mô tả vấn đề: Trong nhiều bộ dữ liệu, số lượng bình luận tích cực và tiêu cực không đều nhau. Thông thường, một bộ dữ liệu sẽ có số lượng bình luận tích cực nhiều hơn hoặc ít hơn so với bình luận tiêu cực. Điều này dẫn đến tình trạng mất cân bằng dữ liệu.
- Hệ quả: Mô hình học sâu (deep learning) có thể thiên vị hướng về cảm xúc chiếm ưu thế trong dữ liệu huấn luyện. Điều này làm giảm hiệu quả phân loại cảm xúc của mô hình khi xử lý các bình luận thực tế, đặc biệt là những bình luận thuộc lớp cảm xúc thiểu số.

1.2.2. Hiểu lầm ngữ cảnh.

- Mô tả vấn đề: Ngữ cảnh và sắc thái cảm xúc trong ngôn ngữ tự nhiên rất phong phú và phức tạp. Một số bình luận có thể mang tính châm biếm, mỉa mai, hoặc có nhiều lớp nghĩa mà mô hình khó có thể hiểu đúng.
- Hệ quả: Mô hình có thể phân loại sai những bình luận mang tính châm biếm, hoặc không nhận diện đúng cảm xúc thực sự của người dùng. Điều này làm giảm độ chính xác và tính hiệu quả của mô hình trong việc phân tích cảm xúc.

1.2.3. Tính toàn diện và đa ngôn ngữ.

- Mô tả vấn đề: Nhiều mô hình phân tích cảm xúc hiện tại chủ yếu được huấn luyện và đánh giá trên một ngôn ngữ cụ thể, thường là tiếng Anh. Việc áp dụng những mô hình này cho các ngôn ngữ khác có thể gặp khó khăn do sự khác biệt về cấu trúc ngôn ngữ, từ vựng, và biểu cảm cảm xúc.
- Hệ quả: Mô hình có thể không hoạt động tốt đối với các ngôn ngữ khác nhau hoặc trong các nền văn hóa khác nhau, làm giảm khả năng ứng dụng rộng rãi của mô hình.

1.2.4. Hướng giải quyết.

- Cân bằng dữ liệu: Sử dụng các kỹ thuật như oversampling, undersampling, hoặc tạo dữ liệu giả (synthetic data) để cân bằng số lượng bình luận tích cực và tiêu cực.
- Cải thiện hiểu ngữ cảnh: Áp dụng các mô hình ngôn ngữ tiên tiến như Transformers (ví dụ: BERT, GPT) để cải thiện khả năng hiểu ngữ cảnh và biểu cảm của bình luận.
- Phát triển mô hình đa ngôn ngữ: Huấn luyện và đánh giá mô hình trên nhiều bộ dữ liệu đa ngôn ngữ và đa văn hóa, đồng thời áp dụng các kỹ thuật học chuyển giao (transfer learning) để nâng cao hiệu quả phân tích cảm xúc cho các ngôn ngữ khác nhau.

1.3. Những vấn đề mà đề tài đồ án cần tập trung nghiên cứu giải quyết.

1.3.1. Cải thiện cân bằng dữ liệu.

- Phát triển phương pháp thu thập dữ liệu:
 - + Thu thập dữ liệu đa dạng: Tìm kiếm và thu thập bình luận từ nhiều nguồn khác nhau để đảm bảo tính đa dạng của dữ liệu.
 - + Chọn lọc dữ liệu cân bằng: Chọn lọc và sử dụng các kỹ thuật phân tích dữ liệu để đảm bảo tỷ lệ cân bằng giữa các bình luận tích cực và tiêu cực trong bộ dữ liệu.
- Kỹ thuật cân bằng dữ liệu:
 - + Oversampling: Tăng số lượng bình luận ở lớp thiểu số bằng cách sao chép dữ liệu hiện có hoặc tạo ra dữ liệu mới dựa trên dữ liệu có sẵn.
 - + Undersampling: Giảm số lượng bình luận ở lớp chiếm ưu thế để đạt được sự cân bằng giữa các lớp.
 - + Synthetic Data Generation: Sử dụng các kỹ thuật học sâu để tạo ra các bình luận nhân tạo nhưng có tính thực tế cao, giúp cân bằng bộ dữ liệu.

1.3.2. Hiểu ngữ cảnh và biểu cảm.

- Phát triển mô hình hiểu ngữ cảnh:

- + Sử dụng mô hình Transformers: Áp dụng các mô hình ngôn ngữ tiên tiến như BERT, GPT để cải thiện khả năng hiểu ngữ cảnh của bình luận.
- + Fine-tuning mô hình: Điều chỉnh mô hình đã được huấn luyện trên các dữ liệu lớn bằng cách sử dụng các bộ dữ liệu cụ thể về bình luận phim để nâng cao hiệu quả.
- Phát triển mô hình hiểu biểu cảm:
 - + Kết hợp các mô hình cảm xúc: Sử dụng các mô hình học sâu để phân tích các yếu tố biểu cảm khác nhau (ví dụ: biểu cảm khuôn mặt, giọng điệu) kết hợp với văn bản bình luận.
 - + Phân tích ngữ nghĩa sâu hơn: Phát triển các thuật toán để phân tích sâu hơn về ngữ nghĩa và mối quan hệ giữa các từ trong bình luận, giúp nhận diện chính xác cảm xúc.

1.3.3. Cải thiện tính toàn diện và đa ngôn ngữ.

- Phát triển mô hình đa ngôn ngữ:
 - + Huấn luyện mô hình trên dữ liệu đa ngôn ngữ: Sử dụng bộ dữ liệu đa ngôn ngữ để huấn luyện mô hình, đảm bảo khả năng áp dụng trên nhiều ngôn ngữ khác nhau.
 - + Học chuyển giao (transfer learning): Áp dụng kỹ thuật học chuyển giao từ các mô hình ngôn ngữ lớn để cải thiện hiệu quả phân tích cảm xúc trên các ngôn ngữ khác nhau.
- Phát triển mô hình đa văn hóa:
 - + Thu thập dữ liệu từ các nền văn hóa khác nhau: Đảm bảo rằng bộ dữ liệu được sử dụng để huấn luyện mô hình bao gồm các bình luận từ nhiều nền văn hóa, đảm bảo tính đa dạng và toàn diện.
 - + Điều chỉnh mô hình theo ngữ cảnh văn hóa: Phát triển các thuật toán có khả năng điều chỉnh mô hình dựa trên ngữ cảnh văn hóa cụ thể, giúp nâng cao hiệu quả phân tích cảm xúc.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VÀ MỘT SỐ PHƯƠNG PHÁP, KỸ THUẬT CHÍNH ĐƯỢC SỬ DỤNG TRONG ĐỒ ÁN ĐỂ GIẢI QUYẾT BÀI TOÁN

2.1. Tổng quan về trí tuệ nhân tạo (Artificial Intelligence).

Vào năm 1943, Warren McCulloch và Walter Pitts bắt đầu thực hiện nghiên cứu ba cơ sở lý thuyết cơ bản: Triết học cơ bản và chức năng của các nơ-ron thần kinh; phân tích các mệnh đề logic; lý thuyết dự đoán Turing. Các tác giả đã nghiên cứu đề xuất mô hình nơ-ron nhân tạo, mỗi nơ-ron đặc trưng bởi hai trạng thái “bật”, “tắt” và phát hiện mạng nơ-ron có khả năng học.

Trí tuệ nhân tạo (AI) được thiết lập bởi John McCarthy tại hội thảo đầu tiên về chủ đề này vào mùa hè năm 1956. Đồng thời, ông cũng đề xuất ngôn ngữ lập trình Lisp, một trong những ngôn ngữ lập trình hàm tiêu biểu, được sử dụng trong lĩnh vực AI. Sau đó, Alan Turing đưa ra “Turing test” như một phương pháp kiểm chứng hành vi thông minh.

Marvin Minsky và Seymour Papert đưa ra các chứng minh đầu tiên về giới hạn của các mạng nơ-ron đơn giản. Ngôn ngữ lập trình logic Prolog ra đời và được phát triển bởi Alan Colmerauer. Ted Shortliffe xây dựng thành công một số hệ chuyên gia đầu tiên trợ giúp chẩn đoán y học, các hệ thống này sử dụng ngôn ngữ luật để biểu diễn tri thức và suy diễn.

Vào đầu năm 1980, những nghiên cứu thành công liên quan đến AI như các hệ chuyên gia (expert systems), một dạng của chương trình AI mô phỏng tri thức và các kỹ năng phân tích của một hoặc nhiều chuyên gia con người. AI được áp dụng trong logic, khai phá dữ liệu, chẩn đoán y học và nhiều lĩnh vực ứng dụng khác trong công nghiệp. Sự thành công dựa vào nhiều yếu tố: Tăng khả năng tính toán của máy tính, tập trung giải quyết các bài toán con cụ thể, xây dựng các mối quan hệ giữa AI và các lĩnh vực khác giải quyết các bài toán tương tự và một sự chuyển giao mới của các nhà nghiên cứu cho các phương pháp trong toán học vững chắc và chẩn đoán khoa học chính xác.

2.1.1. Các lĩnh vực của trí tuệ nhân tạo.

- *Lập luận, suy diễn tự động*: Khái niệm lập luận và suy diễn được sử dụng rất phổ biến trong lĩnh vực AI. Lập luận là suy diễn logic, dùng để chỉ một tiến trình rút ra kết luận (tri thức mới) từ những giả thiết đã cho (được biểu diễn dưới dạng cơ sở tri thức). Như vậy, để thực hiện lập luận người ta cần có các phương pháp lưu trữ cơ sở tri thức và các thủ tục lập luận trên cơ sở tri thức đó.
- *Biểu diễn tri thức*: Muốn máy tính có thể lưu trữ và xử lý tri thức thì cần có các phương pháp biểu diễn tri thức. Các phương pháp biểu diễn tri thức ở đây bao gồm các ngôn ngữ biểu diễn và các kỹ thuật xử lý tri thức. Một ngôn ngữ biểu diễn tri thức được đánh giá là “tốt” nếu nó có tính biểu đạt cao và tính hiệu quả của thuật toán lập luận trong ngôn ngữ đó. Tính biểu đạt của ngôn ngữ thể hiện khả năng biểu diễn một phạm vi rộng lớn các thông tin trong một miền ứng dụng. Tính hiệu quả của các thuật toán lập luận thể hiện chi phí về thời gian và không gian dành cho việc lập luận.
- *Lập kế hoạch*: Khả năng suy ra các mục đích cần đạt được đối với các nhiệm vụ đưa ra và xác định dãy các hành động cần thực hiện để đạt được mục đích đó.
- *Xử lý ngôn ngữ tự nhiên*: Là một nhánh của AI, tập trung vào các ứng dụng trên ngôn ngữ của con người. Các ứng dụng trong nhận dạng tiếng nói, nhận dạng chữ viết, dịch tự động, tìm kiếm thông tin...
- *Hệ chuyên gia*: Cung cấp các hệ thống có khả năng suy luận để đưa ra những kết luận. Các hệ chuyên gia có khả năng xử lý lượng thông tin lớn và cung cấp các kết luận dựa trên những thông tin đó. Có rất nhiều hệ chuyên gia nổi tiếng như các hệ chuyên gia y học MYCIN, đoán nhận cấu trúc phân tử từ công thức hóa học DENDRAL...

2.1.2. Một số ứng dụng của trí tuệ nhân tạo.

Ngày nay, AI ngày càng được ứng dụng nhiều trong các lĩnh vực khác nhau, từ việc phục vụ đời sống hàng ngày của con người cho đến giáo dục, tài

chính ngân hàng, y học, rô-bốt, ô tô tự hành... và thực tế chứng minh rằng việc ứng dụng AI giúp nâng cao hiệu suất lao động, cải thiện chất lượng cuộc sống của con người, phát triển kinh doanh cho các doanh nghiệp và nó cũng sẽ là nền tảng của rất nhiều các ứng dụng và dịch vụ mới khác trong tương lai.

Nhờ sự phát triển khoa học vật lý lượng tử giúp cho việc tính toán và xử lý song song của các hệ thống nhanh hơn đáng kể, vì thế việc áp dụng các phương pháp học máy vào xử lý các bài toán thực tế ngày một thuận lợi hơn. Trong đó, kỹ thuật học sâu (deep learning) đã được quan tâm và phát triển mạnh giúp cho máy tính giải quyết các bài toán trong lĩnh vực học máy ngày càng tốt hơn, mà cụ thể đó là các bài toán tương tác người - máy trong lĩnh vực thị giác máy tính, nhận thức sự vật, gợi ý trong các hệ thống lớn, chẩn đoán các bệnh hiếm gặp...

Một số ứng dụng phổ biến hiện nay như:

- *Lĩnh vực giáo dục*: Mô hình trường học thông minh hay học trực tuyến ngày một phát triển nhờ ứng dụng AI vào quá trình tương tác học tập giữa nhà trường và học sinh, sinh viên tạo ra một hệ sinh thái về giáo dục mà ở đó nhà trường và học sinh tương tác với nhau một cách thuận lợi và nhanh chóng thông qua hệ sinh thái này.
- *Lĩnh vực y tế*: Chẩn đoán, điều trị và theo dõi bệnh giúp nâng cao chăm sóc sức khỏe con người, giảm chi phí chữa bệnh cho người dân, trong đó kể cả các dự án như điều trị ung thư bằng AI, Rô-bốt chăm sóc y tế...
- *Công nghiệp*: Nhận dạng hình ảnh, giọng nói, rô-bốt thông minh, xe tự hành, hệ thống tương tác thực ảo... đã phát triển mạnh mẽ nhờ sự phát triển của mạng nơ-ron học sâu, hệ thống xử lý phân tán song song (Parallel Distributed processing).

2.2. Xử lý ngôn ngữ tự nhiên (Natural language processing).

2.2.1. Tổng quan về xử lý ngôn ngữ tự nhiên.

Xử lý ngôn ngữ tự nhiên (NLP) là gì?

Xử lý ngôn ngữ tự nhiên (NLP) là một lĩnh vực khoa học máy tính, đặc biệt là trong trí tuệ nhân tạo (AI), chuyên về trang bị cho máy tính khả năng hiểu văn bản và ngôn ngữ nói giống như con người.

NLP tích hợp ngôn ngữ học tính toán, sử dụng các mô hình dựa trên quy tắc của ngôn ngữ con người, với các mô hình thống kê, machine learning và deep learning. Những công nghệ kết hợp này cho phép máy tính xử lý ngôn ngữ của con người, dù ở dạng văn bản hay giọng nói và nắm bắt được ý nghĩa đầy đủ của nó, bao gồm cả ý định và cảm xúc của người nói hoặc người viết.

Các ứng dụng của NLP rất đa dạng, từ dịch ngôn ngữ và phản hồi các lệnh nói cho đến tóm tắt nhanh chóng lượng văn bản đa dạng, thường là theo thời gian thực. Rất có thể bạn đã bắt gặp công nghệ NLP thông qua hệ thống GPS điều khiển bằng giọng nói, trợ lý số, phần mềm chuyển giọng nói thành văn bản và chatbot dịch vụ khách hàng, cùng với các ứng dụng thân thiện với người tiêu dùng khác. Ngoài mục đích sử dụng trên, NLP ngày càng đóng vai trò quan trọng trong các giải pháp doanh nghiệp, tối ưu hóa hoạt động kinh doanh, nâng cao năng suất của nhân viên và đơn giản hóa các quy trình kinh doanh quan trọng.

Những thách thức của xử lý ngôn ngữ tự nhiên

Sự phức tạp của ngôn ngữ con người đặt ra những thách thức lớn trong việc phát triển phần mềm diễn giải chính xác dự định từ dữ liệu văn bản hoặc giọng nói. Việc xử lý các từ đồng âm, châm biếm, thành ngữ, ẩn dụ, ngoại lệ ngữ pháp và các biến thể trong cấu trúc câu đòi hỏi các lập trình viên phải dạy các ứng dụng ngôn ngữ tự nhiên để nhận biết và hiểu những sự phức tạp này ngay từ đầu.

Để giải quyết những thách thức này, các tác vụ NLP khác nhau chia nhỏ dữ liệu văn bản và giọng nói của con người, hỗ trợ máy tính hiểu thông tin. Những tác vụ này bao gồm:

- Nhận dạng giọng nói: Còn được gọi là chuyển giọng nói thành văn bản, công việc này liên quan đến việc chuyển đổi dữ liệu giọng nói thành văn bản. Nhận dạng giọng nói rất cần thiết cho các ứng dụng phản hồi lệnh thoại hoặc trả lời các câu hỏi bằng giọng nói, thách thức của việc này nằm ở chỗ cách nói đa dạng của mọi người - có người nói nhanh, có người lắp bắp, với sự nhấn mạnh, ngữ điệu, trọng âm khác nhau và đôi khi sai ngữ pháp.
- Gắn nhãn từ loại: Quá trình này xác định phần lời nói của một từ hoặc văn bản dựa trên ngữ cảnh của nó. Ví dụ: nó xác định từ “năm” như một động từ trong cụm “Mỗi kỹ sư cần nắm rõ chi phí liên quan đến hạ tầng”, và như một danh từ trong cụm “Trưa nay tôi ăn 2 năm cơm”.
- Định nghĩa của từ: Tác vụ này liên quan đến việc chọn nghĩa của một từ có nhiều cách hiểu bằng cách phân tích ngữ cảnh của nó. Ví dụ, nó giúp phân biệt ý nghĩa của động từ “đá” trong “đá bóng” và danh từ “cục đá”.
- Nhận dạng thực thể có tên (NER): Xác định các từ hoặc cụm từ là thực thể, chẳng hạn như nhận dạng “Hà Giang” là một địa điểm hoặc “Hà” là tên một người.
- Giải quyết đồng tham chiếu: Tác vụ này liên quan đến việc xác định xem hai từ có đề cập đến cùng một thực thể hay không, chẳng hạn như xác định rằng “cô ấy” là đề cập đến “Mai” hoặc xác định các ẩn dụ và thành ngữ trong văn bản.
- Phân tích cảm xúc: Cố gắng rút ra những sắc thái chủ quan - thái độ, cảm xúc, sự mỉa mai, bối rối, nghi ngờ - từ văn bản.
- Sinh ngôn ngữ tự nhiên (NLG): Được mô tả là đối lập với nhận dạng giọng nói, NLG liên quan đến việc chuyển đổi thông tin có cấu trúc sang ngôn ngữ của con người.

Các công cụ và chiến lược xử lý ngôn ngữ tự nhiên

- Python và Bộ công cụ ngôn ngữ tự nhiên (NLTK):

Python, một ngôn ngữ lập trình, cung cấp bộ công cụ mở rộng để giải quyết các tác vụ NLP cụ thể. Trong Python, Bộ công cụ ngôn ngữ tự nhiên (NLTK) là một kho lưu trữ nguồn mở bao gồm các thư viện, chương trình và tài nguyên giáo dục, tạo điều kiện thuận lợi cho việc phát triển các chương trình NLP.

NLTK bao gồm thư viện cho các tác vụ NLP khác nhau, cùng với các nhiệm vụ phụ như phân tích câu, phân đoạn từ, bắt nguồn, từ vựng hóa (cắt bớt các từ về gốc của chúng) và mã token (chia văn bản thành các mã token để hiểu rõ hơn). Hơn nữa, nó còn cung cấp các thư viện để triển khai các khả năng nâng cao như lý luận ngữ nghĩa, cho phép đưa ra kết luận logic dựa trên các dữ kiện được trích xuất từ văn bản.

- NLP thống kê, Machine Learning và Deep Learning:

Trong giai đoạn đầu, các ứng dụng NLP đều được mã hóa thủ công, các hệ thống rule-based có khả năng xử lý tốt các tác vụ cụ thể nhưng gặp khó khăn trong việc mở rộng quy mô do liên tục xuất hiện các ngoại lệ và khối lượng dữ liệu văn bản và giọng nói ngày càng tăng.

NLP thống kê là một giải pháp kết hợp các thuật toán máy tính với các mô hình machine learning và deep learning. Sự kết hợp này tự động trích xuất, phân loại và gắn nhãn các thành phần trong dữ liệu văn bản và giọng nói, cho phép thống kê cho các ý nghĩa tiềm ẩn. Hiện tại, các mô hình deep learning, kết hợp mạng CNN (mạng thần kinh tích chập) và mạng RNN (mạng thần kinh tái phát), cho phép hệ thống NLP học một cách linh hoạt, trích xuất ý nghĩa chính xác hơn từ các bộ dữ liệu văn bản và giọng nói mở rộng, không có cấu trúc và không được gắn nhãn.

Ứng dụng Xử lý ngôn ngữ tự nhiên (NLP)

Xử lý ngôn ngữ tự nhiên đóng vai trò là cốt lõi của trí thông minh máy tính trong nhiều tình huống thực tế khác nhau. Dưới đây là một số ứng dụng đáng chú ý của NLP:

- Công cụ dịch thuật

Google Dịch là một minh chứng cho ứng dụng rộng rãi của NLP. Máy dịch ngôn ngữ hiệu quả không chỉ dừng lại ở việc thay thế từ đơn giản, mà nhắm đến nắm bắt chính xác ý nghĩa và giọng điệu của ngôn ngữ đầu vào khi truyền tải văn bản có cùng mục đích và tác động ở ngôn ngữ đầu ra. Các công cụ dịch thuật hiện đại cho thấy sự tiến bộ đáng chú ý về độ chính xác, giải quyết những thách thức trong việc dịch thuật giữa các ngôn ngữ.

- Trợ lý ảo và chatbot

Các trợ lý ảo như Siri của Apple và Alexa của Amazon sử dụng tính năng nhận dạng giọng nói để hiểu lệnh thoại, trong khi ngôn ngữ tự nhiên cho phép chúng phản hồi một cách thích hợp. Chatbot thực hiện các tác vụ tương tự để phản hồi văn bản đã nhập. Những hệ thống tốt nhất này học hỏi theo các dấu hiệu của ngữ cảnh trong yêu cầu của con người, đưa ra phản hồi được cải thiện dần theo thời gian. Mục tiêu tiếp theo của các ứng dụng này liên quan đến việc trả lời câu hỏi bằng những câu trả lời phù hợp và hữu ích bằng ngôn từ của chúng.

- Phân tích cảm xúc trên mạng xã hội

NLP đã phát triển thành một công cụ quan trọng để trích xuất thông tin ẩn giấu từ các kênh mạng xã hội. Phân tích tình cảm kiểm tra ngôn ngữ trong các bài đăng, phản hồi, đánh giá, v.v. trên mạng xã hội để phân biệt thái độ và cảm xúc đối với sản phẩm, chương trình khuyến mãi và sự kiện. Các doanh nghiệp tận dụng thông tin này để thiết kế sản phẩm, chiến dịch quảng cáo và đưa ra quyết định chiến lược.

- Tóm tắt văn bản

Quá trình tóm tắt văn bản, được hỗ trợ bởi công nghệ NLP, xử lý khối lượng lớn văn bản số để tạo ra các bản tóm tắt cho các chỉ mục, cơ sở dữ liệu nghiên cứu hoặc trình đọc văn bản nhanh chóng. Các ứng dụng tóm tắt văn bản hàng đầu kết hợp lý luận ngữ nghĩa và NLG để cung cấp ngữ cảnh và kết luận, nâng cao tính hữu ích của các bản tóm tắt.

- Phát hiện thư rác

Mặc dù tính năng phát hiện thư rác (spam) có thể không được coi là một ứng dụng nổi bật NLP, nhưng các công nghệ hàng đầu tận dụng khả năng phân loại văn bản của NLP để xem xét kỹ lưỡng các email để tìm các mẫu ngôn ngữ cho thấy mail spam hoặc phishing. Điều này bao gồm việc xác định việc sử dụng quá nhiều thuật ngữ tài chính, ngữ pháp ít đặc trưng, ngôn ngữ mang tính đe dọa, mức độ khẩn cấp không phù hợp, tên công ty viết sai chính tả, v.v. Việc phát hiện thư rác đã được các chuyên gia xem là vấn đề “gần như đã được giải quyết”, mặc dù trải nghiệm của từng cá nhân có thể khác nhau.

Xử lý ngôn ngữ tự nhiên trên nền tảng đám mây

Xử lý ngôn ngữ tự nhiên là một công cụ mang tính biến đổi, giúp thu hẹp khoảng cách giữa ngôn ngữ của con người và máy tính. Là một nhánh của AI, NLP giúp cho máy tính không chỉ hiểu văn bản và lời nói mà còn nắm bắt được các sắc thái, bối cảnh và cảm xúc ẩn chứa bên trong. Sự phát triển của NLP đã trải qua những tiến bộ vượt bậc, từ các mô hình rule-based đến việc tích hợp các mô hình thống kê, machine learning và deep learning.

Triển khai ứng dụng NLP trên đám mây mang lại nhiều lợi ích, bao gồm khả năng mở rộng, tiết kiệm chi phí, khả năng truy cập toàn cầu, bảo mật và khả năng tận dụng nhiều loại dịch vụ đám mây để nâng cao chức năng. Những ưu điểm này làm cho điện toán đám mây trở thành một nền tảng hấp dẫn cho các tổ chức đang tìm cách khai thác toàn bộ tiềm năng của ứng dụng NLP.

Trên hành trình phát triển phức tạp của ngôn ngữ, NLP và điện toán đám mây đã trở thành công cụ và nền tảng cơ bản, không chỉ giúp thấu hiểu từ ngữ mà còn làm rõ các sắc thái trong cách diễn đạt của con người.

2.2.2. Một số kỹ thuật xử lý ngôn ngữ tự nhiên phổ biến.

Nguồn gốc và bổ đề ngôn ngữ (Lemmatization và Stemming)

Stemming (Nguồn gốc) là kỹ thuật dùng để biến đổi một từ về dạng gốc (được gọi là stem hoặc root form) bằng cách cực kỳ đơn giản là loại bỏ một số ký tự nằm ở cuối từ mà nó nghĩ rằng là biến thể của từ. Có thể lấy ví dụ đơn

giản như các từ *worked*, *working*, *works* chỉ khác nhau ở những ký tự cuối cùng, bằng cách bỏ đi các hậu tố *-ed*, *-ing*, *-s*, chúng ta đều được từ nguyên gốc là *work*. Bởi vì nguyên tắc hoạt động của Stemming rất đơn giản nên tốc độ xử lý của nó rất nhanh và kết quả stem đôi khi cho ra những kết quả không như mong muốn. Một ví dụ khác như từ *goes* sẽ được stem thành từ *goe* (bỏ chữ *s* cuối từ) trong khi đó stem của từ *go* vẫn là *go*, kết quả là 2 từ *goes* và *go* sau khi được stem thì vẫn không giống nhau. Một nhược điểm khác là nếu các từ dạng bất quy tắc như *went* hay *spoke* thì kỹ thuật Stemming sẽ không thể đưa các từ này về dạng gốc là *go* hay *speak*. Tuy có các nhược điểm như trên nhưng trong thực tế Stemming vẫn được sử dụng khá phổ biến trong NLP vì nó có tốc độ xử lý nhanh và kết quả cuối cùng nhìn chung không hề tệ khi so với Lemmatization.

Lemmatization (Bổ đề ngôn ngữ) khác với Stemming – xử lý bằng cách loại bỏ các ký tự cuối từ một cách rất “máy móc”, Lemmatization sẽ xử lý thông minh hơn bằng một bộ từ điển hoặc một bộ ontology nào đó. Điều này sẽ đảm bảo rằng các từ như *goes*, *went* và *go* sẽ chắc chắn có kết quả trả về là như nhau. Kể cả các từ danh từ như *mouse*, *mice* cũng đều được đưa về cùng một dạng như nhau. Người ta gọi bộ xử lý Lemmatization là Lemmatizer. Nhược điểm của lemmatization là tốc độ xử lý khá chậm vì phải thực hiện tra cứu từ trong cơ sở dữ liệu. Trong các ứng dụng Xử lý ngôn ngữ tự nhiên mà cần độ chính xác cao hơn và thời gian không quan trọng, người ta có thể sử dụng Lemmatization.

Mô hình chủ đề (Topic Modelling)

Mô hình chủ đề là một kiểu mô hình thống kê giúp khai phá các chủ đề ẩn trong tập dữ liệu. Một trong những phương pháp Xử lý ngôn ngữ tự nhiên nổi bật nhất để Lập mô hình Chủ đề là Phân bố Dirichlet tiềm ẩn (Latent Dirichlet Allocation). Để phương pháp này hoạt động, bạn sẽ cần phải xây dựng một danh sách các chủ đề mà bộ sưu tập tài liệu của bạn có thể được áp dụng. Lúc đầu, bạn chỉ định một văn bản cho một chủ đề ngẫu nhiên trong tập

dữ liệu của mình, sau đó xem lại mẫu nhiều lần, nâng cao khái niệm và gán lại tài liệu cho các chủ đề khác nhau.

Trích xuất từ khóa (Keyword Extraction)

Trích xuất từ khóa là một trong những nhiệm vụ quan trọng của Xử lý ngôn ngữ tự nhiên và nó chịu trách nhiệm xác định các cách khác nhau để trích xuất một số lượng đáng kể các từ và cụm từ trong một bộ sưu tập văn bản. Mục đích của việc này nhằm tổng hợp, hỗ trợ việc tổ chức, lưu trữ, tìm kiếm và truy xuất nội dung có liên quan.

Có nhiều loại thuật toán trích xuất từ khóa khác nhau, một số thuật toán chỉ trích xuất từ và một số thuật toán khác lại có thể trích xuất cả từ và cụm từ, ngoài ra còn có các thuật toán trích xuất từ khóa dựa trên nội dung của văn bản.

Dưới đây liệt kê một số thuật toán trích xuất từ khóa nổi bật:

- TextRank: thuật toán này hoạt động tương tự như việc Xếp hạng trang (PageRank). Google sử dụng thuật toán này để xếp hạng các trang web trên internet.
- Term Frequency: Thuật toán này sẽ xác định rõ tầm quan trọng của một thuật ngữ trong tài liệu.
- RAKE: Là viết tắt của Rapid Automatic Keyword Extraction – Trích xuất từ khóa tự động. Thuật toán này có thể trích xuất các từ khóa và cụm từ khóa từ nội dung của một tài liệu mà không cần tính đến các tài liệu khác trong cùng một bộ sưu tập.

Sơ đồ tri thức (Knowledge Graphs)

Sơ đồ tri thức là một tập hợp của ba mục: chủ đề, vị ngữ và thực thể – một phương pháp lưu trữ thông tin. Sơ đồ tri thức đang trở nên ngày càng phổ biến, đặc biệt là khi nó được sử dụng bởi nhiều công ty lớn (chẳng hạn như Biểu đồ thông tin của Google) cho các hàng hóa và dịch vụ khác nhau. Việc xây dựng một sơ đồ tri thức đòi hỏi nhiều kỹ thuật Xử lý ngôn ngữ khác nhau và việc sử dụng nhiều hơn các cách tiếp cận này sẽ tạo ra một sơ đồ tri thức hiệu quả và kỹ lưỡng hơn.

Đám mây từ (Word Cloud)

Đám mây từ, đôi khi được gọi là đám mây thể, là một cách tiếp cận trực quan hóa dữ liệu. Các từ của một văn bản được hiển thị trong một bảng, với các thuật ngữ quan trọng nhất được in bằng các chữ cái lớn hơn và các từ ít quan trọng hơn được mô tả ở kích thước nhỏ hơn hoặc hoàn toàn không hiển thị.

Nhận dạng thực thể có tên (Named Entity Recognition)

Thuật toán này phụ trách phân loại và xếp loại những người trong văn bản không có cấu trúc thành một tập hợp các nhóm được xác định trước. Điều này bao gồm các cá nhân, nhóm, ngày tháng, số tiền,...

Phân tích quan điểm (Sentiment Analysis)

Phân tích quan điểm là thuật toán thường được sử dụng nhất trong Xử lý ngôn ngữ tự nhiên. Phân tích quan điểm đặc biệt hữu ích trong những trường hợp người tiêu dùng đưa ra ý tưởng và đề xuất của họ, chẳng hạn như thăm dò ý kiến người tiêu dùng, xếp hạng và tranh luận trên phương tiện truyền thông xã hội. Trong phân tích quan điểm, thang điểm ba mức (tích cực/tiêu cực/trung tính) là phương pháp đơn giản nhất. Trong những trường hợp phức tạp hơn, kết quả đầu ra có thể là một điểm số thống kê có thể được chia thành nhiều loại nếu cần. Cả thuật toán được giám sát và không được giám sát đều có thể được sử dụng để phân tích quan điểm. Một trong số những mô hình phân tích quan điểm được sử dụng nhiều nhất là Naive Bayes. Cần có một kho dữ liệu đào tạo được gắn nhãn quan điểm, từ đó một mô hình có thể được đào tạo và sau đó được sử dụng để xác định quan điểm.

Tóm tắt văn bản (Text Summarization)

Tóm tắt văn bản có thể được thực hiện theo hai cách: trích dẫn và trừu tượng hóa. Bằng cách xóa các bit khỏi văn bản, các phương pháp trích xuất sẽ tạo ra một bản tóm tắt. Trong khi đó phương pháp trừu tượng tạo ra bản tóm tắt bằng cách thiết lập một văn bản mới truyền tải nội dung của văn bản gốc. Các thuật toán phổ biến dùng để tóm tắt văn bản có thể kể tới như LexRank, TextRank và Phân tích quan điểm ngầm (Latent Semantic Analysis).

Mô hình túi từ (Bag of Words)

Mô hình này thể hiện một văn bản giống như một túi (nhiều tập hợp) từ, không bao gồm ngữ pháp và trật tự từ trong khi vẫn giữ được tính đa dạng. Về bản chất, mô hình túi từ tạo ra một ma trận tỷ lệ. Các tần số hoặc trường hợp từ này sau đó được sử dụng như các tính năng trong việc đào tạo bộ phân loại (Classifier).

Tách từ (Tokenization)

Tách từ (Tokenization) là một trong những bước quan trọng nhất trong quá trình tiền xử lý văn bản. Cho dù bạn đang làm việc với các kỹ thuật NLP truyền thống hay sử dụng các kỹ thuật học sâu nâng cao thì vẫn không thể bỏ qua bước này. Nói một cách đơn giản, tokenization là quá trình tách một cụm từ, câu, đoạn văn, một hoặc nhiều tài liệu văn bản thành các đơn vị nhỏ hơn. Mỗi đơn vị nhỏ hơn này được gọi là Tokens.

Các thuật toán này được VinBigData ứng dụng để phát triển các sản phẩm thuộc hệ sinh thái VinBase về xử lý ngôn ngữ tự nhiên nhằm tạo ra các cuộc hội thoại được phản hồi một cách nhanh chóng, chính xác và tự nhiên nhất. Các sản phẩm này có thể thay thế các nhân viên trực tổng đài (AI CallBot) hoặc giúp con người thực hiện các tác vụ rảnh tay (Virtual Assistant) với khả năng hoạt động 24/7, ứng dụng được trong đa ngành, đa lĩnh vực. Việc tự động hóa này không chỉ giúp doanh nghiệp tối ưu hiệu suất và chi phí hoạt động mà còn gia tăng trải nghiệm khách hàng, thúc đẩy doanh số, mở rộng thị trường.

2.3. Kỹ thuật phân tích quan điểm (Sentiment Analysis) trong xử lý ngôn ngữ tự nhiên.

Phân tích quan điểm là một ứng dụng của trí tuệ nhân tạo, nó sử dụng các thuật toán phức tạp để xử lý ngôn ngữ tự nhiên của con người (NLP) và xác định các đặc điểm cảm xúc tiêu cực/tích cực tại một thời điểm thông qua văn bản hoặc lời nói. Các nguồn dữ liệu được phân tích phổ biến như Social media, Blog, Website đánh giá sản phẩm, tổng đài Contact center,...

Hiện nay, nhờ sự tiến bộ của các công nghệ thế hệ mới, các thuật toán phân tích quan điểm ngày càng được nâng cấp với độ chính xác cao, từ đó hỗ trợ trong các sản phẩm thông minh như trợ lý ảo tích hợp trên ô tô, căn hộ... cải thiện chất lượng cuộc sống của con người.

2.3.1. Tổng quan lợi ích của phân tích quan điểm.

- Xác định và trích xuất thông tin hữu ích từ khách hàng. Việc nhận biết được cảm xúc của người dùng trong cuộc trò chuyện giúp doanh nghiệp phân tích được mức độ quan tâm của khách hàng đối với thương hiệu, sản phẩm/dịch vụ của mình. Đây là nguồn thông tin có giá trị cao giúp doanh nghiệp điều chỉnh chiến lược về sản phẩm, kinh doanh, marketing và đặc biệt là các dịch vụ tư vấn phù hợp.
- Tự động đánh giá chất lượng của các tư vấn viên hoặc tổng đài viên. Thông qua cảm xúc của khách hàng trong các cuộc trò chuyện bằng tin nhắn hoặc cuộc gọi, doanh nghiệp dễ dàng xác định được thái độ và sự hài lòng của khách hàng đối với hoạt động tư vấn. Từ đó dễ dàng đánh giá được chất lượng của nhân viên và quản lý hiệu quả hơn.

2.3.2. Các dạng phân tích quan điểm chính.

- Loại 1: Phân tích chi tiết (Fine-Grained)

Mô hình phân tích cảm xúc này giúp xác định được độ chính xác của các tính chất. Các tính chất chính được phân chia thành: rất tích cực, tích cực, trung tính, tiêu cực hoặc rất tiêu cực. Việc phân chia chi tiết như vậy rất phù hợp để đánh giá các cuộc trò chuyện

Đối với thang điểm đánh giá từ 1 đến 5, bạn có thể coi 1 là rất tiêu cực và 5 là rất tích cực. Đối với thang điểm từ 1 đến 10, bạn có thể coi 1-2 là rất tiêu cực và 9-10 là rất tích cực.

- Loại 2: Dựa trên khía cạnh (Aspect-Based)

Trong khi phân tích chi tiết xác định tính chất tổng thể cảm xúc của khách hàng trong cuộc trò chuyện, thì phân tích dựa trên khía cạnh sẽ đi sâu hơn, nhận dạng cụ thể từng khía cạnh trong lời nói của họ.

Chẳng hạn khi khách hàng nói rằng “máy ảnh gặp khó khăn trong điều kiện ánh sáng nhân tạo”. Với phân tích dựa trên khía cạnh, chúng ta không chỉ đánh giá được đây là cảm xúc tiêu cực mà còn có thể xác định rằng người dùng đó đã nhận xét tiêu cực về đối tượng “máy ảnh”.

- Loại 3: Phát hiện cảm xúc (Emotion Detection)

Cảm xúc ở đây có thể bao gồm các sắc thái tức giận, buồn bã, hạnh phúc, thất vọng, sợ hãi, lo lắng, hoảng sợ... Hệ thống phát hiện cảm xúc thường sử dụng từ vựng – một tập hợp các từ truyền tải những cảm xúc nhất định. Một số bộ phân loại nâng cao cũng sử dụng các thuật toán học máy (Machine Learning – ML) mạnh mẽ.

- Loại 4: Phân tích ý định (Intent Analysis)

Việc xác định chính xác mục đích của người tiêu dùng có thể giúp công ty tiết kiệm thời gian, tiền bạc và công sức, bởi các doanh nghiệp có thể không tiếp tục chăm sóc những khách hàng không tiềm năng và chưa có kế hoạch mua hàng. Phân tích mục đích chính xác có thể giải quyết nhiều vấn đề cho doanh nghiệp như vậy.

Phân tích mục đích giúp doanh nghiệp xác định mục đích của người tiêu dùng – cho dù khách hàng có ý định mua hàng hay chỉ đang lướt qua. Nếu khách hàng sẵn sàng mua hàng, bạn có thể theo dõi họ và nhắm mục tiêu họ bằng các quảng cáo. Nếu người tiêu dùng chưa sẵn sàng mua, chúng ta có thể tiết kiệm thời gian và nguồn lực bằng cách không quảng cáo cho họ.

2.3.3. Giới thiệu thuật toán xác định ngữ nghĩa theo ngữ cảnh (Contextual Semantic Search – CSS).

Để nắm bắt được những thông tin hữu ích từ khách hàng, điều quan trọng nhất là phải hiểu họ đang thảo luận về khía cạnh nào của thương hiệu. Ví dụ: Amazon muốn tách biệt các tin nhắn liên quan đến: giao hàng trễ, vấn đề thanh toán, truy vấn liên quan đến khuyến mại, đánh giá sản phẩm, v.v. Mặt khác, Starbucks muốn phân loại các tin nhắn dựa trên việc chúng có liên quan đến hành vi của nhân viên, hương vị cà phê mới hay không, phản hồi về vệ sinh,

đơn đặt hàng trực tuyến, tên và vị trí cửa hàng, v.v. Nhưng làm thế nào máy tính có thể phân loại những thứ đó?

Giải pháp cho vấn đề này chính là thuật toán tìm kiếm thông minh thông minh – Xác định ngữ nghĩa theo ngữ cảnh (CSS). Cách hoạt động của CSS là nó lấy hàng nghìn thông điệp và một khái niệm làm đầu vào và lọc tất cả các thông điệp có mối liên quan chặt chẽ với khái niệm đã cho. Chẳng hạn như với khái niệm giá, thuật toán thực hiện tìm kiếm từ khóa về Giá và các từ liên quan chặt chẽ khác như (giá, phí, đô la, trả tiền).

Tuy nhiên, điểm hạn chế của phương pháp này chính là không thể tìm ra tất cả các từ khóa có liên quan và các biến thể của chúng.

2.4. Các gói, hàm và kỹ thuật sử dụng trong phân tích tình cảm.

2.4.1. PyTorch và TorchText.

- PyTorch: Là thư viện học sâu phổ biến, được sử dụng để xây dựng, huấn luyện và đánh giá mô hình học sâu. PyTorch nổi tiếng với tính linh hoạt và dễ sử dụng trong nghiên cứu và triển khai thực tế.
- TorchText: Là một phần của PyTorch, được sử dụng để quản lý và tiền xử lý dữ liệu văn bản. Các công cụ của TorchText giúp dễ dàng trong việc token hóa, xây dựng từ điển (vocab) và tạo các iterator cho huấn luyện mô hình.

2.4.2. Tiền xử lý văn bản.

- Tokenization là một bước quan trọng trong xử lý ngôn ngữ tự nhiên (NLP), giúp chuyển đổi văn bản thành các đơn vị nhỏ hơn gọi là "token". Các token này có thể là từ, cụm từ hoặc thậm chí là các ký tự riêng lẻ, tùy thuộc vào ứng dụng cụ thể. Dưới đây là một số phương pháp và lợi ích của tokenization:

+ Các phương pháp Tokenization:

- Word Tokenization (Token hóa từ):
 - Chia văn bản thành các từ đơn lẻ. Ví dụ: câu "Tôi yêu thích học máy" sẽ được token hóa thành ["Tôi", "yêu", "thích", "học", "máy"].

- Các thư viện phổ biến hỗ trợ word tokenization bao gồm: spaCy, NLTK, và BERT Tokenizer.
- Subword Tokenization (Token hóa tiểu từ):
 - Chia từ thành các phần nhỏ hơn, thường sử dụng trong các mô hình như BERT và GPT-3. Ví dụ: từ "học máy" có thể được chia thành ["học", "máy"] hoặc ["họ", "c#", "máy"].
 - Các kỹ thuật như Byte Pair Encoding (BPE) và WordPiece thường được sử dụng.
- Character Tokenization (Token hóa ký tự):
 - Chia văn bản thành các ký tự đơn lẻ. Ví dụ: câu "học" sẽ được token hóa thành ["h", "ọ", "c"].
 - Phương pháp này thường được sử dụng khi làm việc với các ngôn ngữ có cấu trúc phức tạp hoặc khi cần phân tích sâu hơn về cấu trúc từ.

+ Lợi ích của Tokenization:

- Chuẩn hóa dữ liệu: Tokenization giúp chuẩn hóa văn bản, làm cho dữ liệu trở nên đồng nhất và dễ xử lý hơn.
- Tăng độ chính xác của mô hình: Với các token chuẩn hóa, mô hình có thể học và dự đoán tốt hơn, cải thiện độ chính xác của các bài toán NLP như phân loại văn bản, phân tích cảm xúc, và dịch máy.
- Hiểu ngữ nghĩa và ngữ cảnh: Tokenization giúp mô hình nắm bắt được ngữ nghĩa và ngữ cảnh của từ hoặc cụm từ trong văn bản, từ đó cải thiện hiệu quả của các thuật toán NLP.

The image shows a Python code editor window with a dark background. The title bar says 'Python' and there is a 'Sao chép' (Copy) button. The code is as follows:

```
import spacy
nlp = spacy.load("en_core_web_sm")

text = "This is a simple example."
doc = nlp(text)

tokens = [token.text for token in doc]
print(tokens)
```

Below the code, it says 'Kết quả sẽ là:' (The result will be:). Below that is another code editor window showing the output:

```
['This', 'is', 'a', 'simple', 'example', '.']
```

There is also a 'Sao chép' button next to the output.

Hình 2.1: Ví dụ về Tokenization với spaCy

Tokenization là một bước không thể thiếu trong xử lý ngôn ngữ tự nhiên, giúp chuyển đổi văn bản thô thành các đơn vị có ý nghĩa, phục vụ cho các bước phân tích và xử lý tiếp theo.

- Embedding từ (Word Embedding) là một kỹ thuật trong xử lý ngôn ngữ tự nhiên (NLP) dùng để ánh xạ các từ trong văn bản thành các vector số có ý nghĩa, phản ánh mối quan hệ ngữ nghĩa giữa các từ. Dưới đây là các khía cạnh quan trọng của embedding:

+ Khái niệm Embedding từ:

- Embedding từ là quá trình chuyển đổi các từ thành các vector số trong không gian có nhiều chiều. Mỗi từ được biểu diễn dưới dạng một vector, trong đó các từ có ngữ nghĩa gần nhau sẽ có vector gần nhau trong không gian vector.
- Các vector embedding có thể được học từ dữ liệu văn bản lớn và chứa đựng thông tin ngữ nghĩa về từ đó.

+ Lợi ích của Embedding từ:

- **Bắt được ngữ nghĩa:** Embedding từ giúp mô hình hiểu được ngữ nghĩa của các từ, giúp cải thiện hiệu quả của các bài toán NLP như phân loại văn bản, dịch máy, và tạo văn bản.
- **Giảm chiều dữ liệu:** So với việc sử dụng các phương pháp biểu diễn từ truyền thống như one-hot encoding, embedding từ giúp giảm đáng kể kích thước của dữ liệu, từ đó giảm thời gian huấn luyện và yêu cầu lưu trữ.

+ Các kỹ thuật Embedding phổ biến:

- **Word2Vec:** Là một kỹ thuật embedding do Google phát triển, sử dụng các mô hình neural để học vector từ. Có hai kiến trúc chính là CBOW (Continuous Bag of Words) và Skip-gram.
- **GloVe (Global Vectors for Word Representation):** Là kỹ thuật embedding do Stanford phát triển, dựa trên ma trận đồng thời từ để học vector từ. GloVe tận dụng thông tin thống kê toàn cục của corpus để tạo ra vector từ.
- **FastText:** Là phiên bản cải tiến của Word2Vec do Facebook phát triển, học vector cho cả từ và các n-gram con của từ, giúp mô hình xử lý tốt hơn các từ mới hoặc ít xuất hiện.

+ Ứng dụng của Embedding từ:

- **Phân loại văn bản:** Sử dụng embedding từ để biến đổi văn bản thành các vector đầu vào cho các mô hình học sâu như LSTM, Transformer.
- **Dịch máy:** Sử dụng embedding từ để biểu diễn các từ trong cả ngôn ngữ nguồn và ngôn ngữ đích, giúp mô hình dịch hiểu và tạo ra các câu dịch chính xác hơn.
- **Tìm kiếm ngữ nghĩa:** Sử dụng embedding từ để tìm kiếm các tài liệu hoặc từ ngữ có ngữ nghĩa tương tự nhau.

```

import torch
import torch.nn as nn

# Giả sử ta đã có pre-trained GloVe embeddings
pretrained_embeddings = torch.FloatTensor([[0.1, 0.2], [0.3, 0.4], [0.5, 0.6]])

# Số từ, kích thước embedding
num_embeddings, embedding_dim = pretrained_embeddings.shape

# Tạo lớp embedding và gán pre-trained embeddings
embedding = nn.Embedding(num_embeddings, embedding_dim)
embedding.weight.data.copy_(pretrained_embeddings)

# Lấy embedding cho từ có chỉ số 1
input_idx = torch.LongTensor([1])
embedded = embedding(input_idx)

print(embedded)

```

Hình 2.2: Ví dụ về Embedding từ với GloVe và Pytorch

Kết quả sẽ là vector embedding tương ứng với từ có chỉ số 1 trong pre-trained GloVe embeddings. Embedding từ là một phần không thể thiếu trong xử lý ngôn ngữ tự nhiên, giúp cải thiện hiệu quả và tính chính xác của các mô hình NLP hiện đại.

2.4.3. Mô hình Recurrent Neural Network (RNN) với Long Short-Term Memory (LSTM).

- Recurrent Neural Network (RNN) là một loại mạng nơ-ron nhân tạo đặc biệt hiệu quả trong việc xử lý dữ liệu tuần tự, chẳng hạn như văn bản, âm thanh và chuỗi thời gian. Khác với các mạng nơ-ron truyền thống, RNN có khả năng ghi nhớ thông tin từ các bước trước đó trong chuỗi dữ liệu, cho phép nó nắm bắt được mối quan hệ dài hạn trong dữ liệu tuần tự. Dưới đây là một số khái niệm và đặc điểm chính của RNN:

+ Kiến trúc của RNN:

- Hidden State: RNN có một thành phần gọi là trạng thái ẩn (hidden state), giúp lưu trữ thông tin từ các bước trước đó trong chuỗi dữ liệu.
- Recurrent Connection: Tại mỗi bước trong chuỗi, trạng thái ẩn hiện tại được cập nhật dựa trên trạng thái ẩn của bước trước đó và đầu vào hiện tại.

+ Hoạt động của RNN:

- Input Sequence: RNN nhận vào một chuỗi dữ liệu (ví dụ: các từ trong một câu).
- Hidden State Update: Trạng thái ẩn được cập nhật tuần tự qua từng bước trong chuỗi.
- Output: Tại mỗi bước, RNN có thể tạo ra một đầu ra dựa trên trạng thái ẩn hiện tại. Đầu ra này có thể là đầu ra cuối cùng của toàn bộ chuỗi hoặc đầu ra cho từng bước trong chuỗi.

+ Ưu điểm và Hạn chế của RNN:

- Ưu điểm:
 - Ghi nhớ thông tin tuần tự: RNN có khả năng ghi nhớ và sử dụng thông tin từ các bước trước đó trong chuỗi, giúp nó nắm bắt được mối quan hệ dài hạn.
 - Ứng dụng rộng rãi: RNN được sử dụng rộng rãi trong nhiều bài toán như dịch máy, phân tích cảm xúc, nhận dạng giọng nói, và dự đoán chuỗi thời gian.
- Hạn chế:
 - Vanishing Gradient Problem: Khi huấn luyện RNN với các chuỗi dài, các gradient có thể trở nên rất nhỏ, làm cho mô hình khó học được mối quan hệ dài hạn. Điều này được biết đến như là vấn đề "vanishing gradient".
 - Ghi nhớ ngắn hạn: RNN truyền thống thường gặp khó khăn trong việc ghi nhớ thông tin qua các khoảng cách dài.

+ Các Biến thể của RNN:

- LSTM (Long Short-Term Memory): Là một loại RNN đặc biệt được thiết kế để khắc phục vấn đề vanishing gradient bằng cách sử dụng các cổng (gates) để kiểm soát dòng chảy của thông tin.
- GRU (Gated Recurrent Unit): Là một biến thể khác của RNN, tương tự LSTM nhưng có cấu trúc đơn giản hơn.

```

import torch
import torch.nn as nn

# Định nghĩa lớp RNN
class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(RNN, self).__init__()
        self.hidden_size = hidden_size
        self.rnn = nn.RNN(input_size, hidden_size, batch_first=True)
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        h0 = torch.zeros(1, x.size(0), self.hidden_size).to(x.device)
        out, _ = self.rnn(x, h0)
        out = self.fc(out[:, -1, :])
        return out

# Khởi tạo và sử dụng mô hình RNN
input_size = 10
hidden_size = 20
output_size = 1
model = RNN(input_size, hidden_size, output_size)

# Tạo dữ liệu mẫu
x = torch.randn(5, 3, input_size) # batch_size=5, seq_len=3, input_size=10
output = model(x)
print(output)

```

Hình 2.3: Ví dụ RNN với pytorch

- Long Short-Term Memory (LSTM) là một loại mạng nơ-ron hồi quy (Recurrent Neural Network - RNN) được thiết kế để khắc phục các hạn chế của RNN truyền thống, đặc biệt là vấn đề "vanishing gradient" (gradient biến mất). LSTM có khả năng ghi nhớ và duy trì thông tin trong khoảng thời gian dài hơn, điều này giúp nó trở nên cực kỳ hiệu quả trong việc xử lý dữ liệu tuần tự như văn bản, âm thanh và chuỗi thời gian.

+ Kiến trúc của LSTM

- LSTM được cấu tạo từ các đơn vị (unit) đặc biệt gọi là LSTM cell. Mỗi LSTM cell bao gồm ba thành phần chính:
- Cell State (trạng thái ô): Đây là thành phần quan trọng nhất của LSTM, giúp truyền thông tin từ bước này sang bước khác.
- Forget Gate (cổng quên): Quyết định thông tin nào sẽ được giữ lại và thông tin nào sẽ bị loại bỏ khỏi cell state.

- Input Gate (cổng đầu vào): Quyết định thông tin nào từ đầu vào hiện tại sẽ được thêm vào cell state.
- Output Gate (cổng đầu ra): Quyết định thông tin nào từ cell state sẽ được dùng để tính toán đầu ra tại bước hiện tại.

+ Ưu điểm của LSTM

- Ghi nhớ dài hạn: LSTM có khả năng ghi nhớ thông tin qua các khoảng thời gian dài, khắc phục vấn đề vanishing gradient mà RNN truyền thống gặp phải.
- Linh hoạt và mạnh mẽ: LSTM đã chứng minh hiệu quả trong nhiều bài toán khác nhau như dịch máy, phân tích cảm xúc, nhận dạng giọng nói, và dự đoán chuỗi thời gian.

LSTM là một công cụ mạnh mẽ trong xử lý ngôn ngữ tự nhiên và các bài toán dữ liệu tuần tự nhờ khả năng ghi nhớ và cập nhật thông tin hiệu quả. Các kiến trúc đặc biệt của LSTM như các cổng forget, input và output giúp nó vượt qua những hạn chế của RNN truyền thống và mang lại kết quả tốt hơn trong nhiều ứng dụng thực tế.

2.4.4. Kỹ thuật đóng gói tuần tự (Packed Sequence).

Kỹ thuật Packed Sequence là một phương pháp trong xử lý ngôn ngữ tự nhiên (NLP) và học sâu (deep learning) để làm việc hiệu quả hơn với các chuỗi dữ liệu có độ dài không đồng đều, thường sử dụng trong các mô hình Recurrent Neural Networks (RNNs) và các biến thể của nó như LSTM và GRU.

- Vấn đề của chuỗi dữ liệu có độ dài không đồng đều: Trong NLP, các câu thường có độ dài khác nhau, nhưng để huấn luyện mô hình, tất cả các chuỗi thường cần được điều chỉnh về cùng một độ dài thông qua padding (thêm các ký tự đặc biệt để chuỗi có độ dài như nhau). Tuy nhiên, việc này có thể làm giảm hiệu quả và tốc độ của mô hình vì phải xử lý thêm các ký tự padding không mang ý nghĩa.
- Giải pháp Packed Sequence: Kỹ thuật Packed Sequence cho phép mô hình xử lý các chuỗi dữ liệu có độ dài khác nhau mà không cần thêm padding dư

thừa. Bằng cách đóng gói các chuỗi dữ liệu ngắn hơn lại với nhau, mô hình có thể tập trung vào các phần dữ liệu thực tế, tăng hiệu quả huấn luyện và giảm thời gian tính toán.

- Các bước thực hiện Packed Sequence:
 - + Chuẩn bị dữ liệu: Đầu tiên, các chuỗi dữ liệu cần được sắp xếp theo thứ tự giảm dần của độ dài.
 - + Đóng gói chuỗi: Sử dụng PyTorch, các chuỗi được đóng gói lại thành một đối tượng PackedSequence.
 - + Giải nén chuỗi: Sau khi mô hình xử lý, kết quả sẽ được giải nén để lấy lại các chuỗi dữ liệu ban đầu với các chiều phù hợp.

Kỹ thuật Packed Sequence giúp xử lý hiệu quả các chuỗi dữ liệu có độ dài không đồng đều, giảm thiểu việc sử dụng padding dư thừa và tăng tốc độ cũng như hiệu quả của mô hình học sâu. Đây là một công cụ quan trọng trong xử lý ngôn ngữ tự nhiên và học sâu với các mô hình tuần tự như LSTM và GRU.

2.4.5. Huấn luyện và đánh giá mô hình.

- Optimizer (Adam): Adam - Adaptive Moment Estimation là một thuật toán tối ưu hóa phổ biến và hiệu quả được sử dụng rộng rãi trong huấn luyện mạng nơ-ron sâu. Adam kết hợp những ưu điểm của hai kỹ thuật tối ưu hóa khác là AdaGrad và RMSProp, giúp nó thích ứng với việc huấn luyện các mô hình học sâu phức tạp. Adam tính toán các giá trị trung bình di động của gradient và bình phương của gradient theo thời gian, sau đó điều chỉnh các bước cập nhật trọng số dựa trên những giá trị này.

Ưu điểm của Adam:

- + Tốc độ hội tụ nhanh: Adam thường hội tụ nhanh hơn so với các thuật toán tối ưu hóa khác, đặc biệt là khi làm việc với các mô hình sâu và dữ liệu lớn.
- + Điều chỉnh học tự động: Adam tự động điều chỉnh tốc độ học cho từng trọng số, giúp mô hình học tốt hơn và tránh các vấn đề như gradient biến mất hoặc gradient bùng nổ.

+ Ổn định và hiệu quả: Adam ổn định và hiệu quả trên nhiều loại bài toán và cấu trúc mạng khác nhau.

Adam là một thuật toán tối ưu hóa mạnh mẽ và hiệu quả, kết hợp các ưu điểm của nhiều kỹ thuật khác để mang lại hiệu suất cao trong huấn luyện các mô hình học sâu. Với khả năng tự động điều chỉnh tốc độ học và xử lý tốt các vấn đề liên quan đến gradient, Adam đã trở thành lựa chọn hàng đầu cho nhiều nhà nghiên cứu và phát triển trong lĩnh vực học sâu.

- Loss Function (BCEWithLogitsLoss - Binary Cross Entropy with Logits Loss) là một hàm mất mát được sử dụng phổ biến trong các bài toán phân loại nhị phân trong học sâu. Hàm này kết hợp giữa Binary Cross Entropy Loss và Sigmoid Activation thành một bước duy nhất, giúp cải thiện độ ổn định số và hiệu quả tính toán. Trong các bài toán phân loại nhị phân, đầu ra của mô hình thường là một giá trị logits (tức là giá trị trước khi áp dụng hàm sigmoid). Hàm sigmoid được sử dụng để chuyển đổi giá trị logits thành xác suất nằm trong khoảng từ 0 đến 1. BCEWithLogitsLoss là một hàm mất mát quan trọng và hiệu quả trong các bài toán phân loại nhị phân. Bằng cách kết hợp giữa BCE Loss và hàm sigmoid, nó giúp tối ưu hóa quy trình tính toán và đảm bảo độ ổn định số, giúp mô hình học sâu hội tụ nhanh hơn và chính xác hơn.
- Accuracy Calculation là một trong những thước đo hiệu quả của mô hình trong các bài toán phân loại. Độ chính xác (accuracy) biểu thị tỷ lệ phần trăm của các dự đoán đúng (true positives và true negatives) so với tổng số dự đoán. Tuy nhiên, nó có thể không đủ để đánh giá toàn diện hiệu quả của mô hình, đặc biệt trong các bài toán với dữ liệu mất cân bằng (imbalanced data). Trong các trường hợp này, các thước đo khác như Precision, Recall và F1 Score cũng rất quan trọng để đánh giá hiệu quả của mô hình. Khi sử dụng accuracy, cần xem xét cả ngữ cảnh của bài toán và cấu trúc dữ liệu để đảm bảo đánh giá đúng hiệu quả của mô hình. Sử dụng PyTorch, chúng ta

có thể dễ dàng tính toán accuracy và áp dụng các thước đo khác để có cái nhìn toàn diện hơn về hiệu quả của mô hình.

2.4.6. Dự đoán cảm xúc.

Sentence Prediction: Mô hình được sử dụng để dự đoán cảm xúc của các câu bình luận mới, sử dụng thư viện spaCy để token hóa câu và tính toán xác suất cảm xúc. Sentence Prediction là một quy trình phức tạp bao gồm nhiều bước từ tiền xử lý dữ liệu, trích xuất đặc trưng, đến dự đoán và đánh giá kết quả. Sử dụng các mô hình học sâu như LSTM hoặc Transformer, quá trình này giúp mô hình có thể hiểu và dự đoán thông tin từ các câu văn bản một cách hiệu quả. PyTorch cung cấp các công cụ mạnh mẽ để triển khai và huấn luyện các mô hình này, giúp chúng dễ dàng được áp dụng trong nhiều ứng dụng xử lý ngôn ngữ tự nhiên. Quá trình này bao gồm các bước chính sau:

- Tiền xử lý dữ liệu:
 - + Tokenization: Chia câu văn bản thành các token (từ hoặc ký tự) để mô hình có thể xử lý.
 - + Embedding: Chuyển đổi các token thành vector số có ý nghĩa bằng cách sử dụng các kỹ thuật embedding như Word2Vec, GloVe, hoặc các embedding học được từ mô hình ngôn ngữ.
- Trích xuất đặc trưng (Feature Extraction):
 - + Sử dụng các mô hình học sâu như LSTM, GRU, hoặc Transformer để trích xuất các đặc trưng quan trọng từ các vector embedding.
 - + Các mô hình này có khả năng ghi nhớ ngữ cảnh và mối quan hệ giữa các từ trong câu, giúp mô hình hiểu ngữ nghĩa của câu tốt hơn.
- Dự đoán:
 - + Output Layer: Kết quả từ các lớp ẩn được đưa qua lớp đầu ra để dự đoán. Trong bài toán phân loại nhị phân, lớp đầu ra thường là một lớp Sigmoid, trong khi bài toán phân loại đa lớp sử dụng lớp Softmax.

- + Decision Threshold: Trong bài toán phân loại nhị phân, xác suất đầu ra sẽ được so sánh với ngưỡng quyết định (thường là 0.5) để xác định nhãn dự đoán cuối cùng.
- Đánh giá:
 - + Đo lường hiệu quả của mô hình bằng các thước đo như độ chính xác (accuracy), độ chính xác từng lớp (precision), độ nhạy (recall), và F1 score.
 - + Sử dụng tập dữ liệu kiểm tra để đánh giá khả năng tổng quát hóa của mô hình.

Sentence Prediction là một quy trình phức tạp bao gồm nhiều bước từ tiền xử lý dữ liệu, trích xuất đặc trưng, đến dự đoán và đánh giá kết quả. Sử dụng các mô hình học sâu như LSTM hoặc Transformer, quá trình này giúp mô hình có thể hiểu và dự đoán thông tin từ các câu văn bản một cách hiệu quả. PyTorch cung cấp các công cụ mạnh mẽ để triển khai và huấn luyện các mô hình này, giúp chúng dễ dàng được áp dụng trong nhiều ứng dụng xử lý ngôn ngữ tự nhiên.

2.5. Giới thiệu Pytorch trong xử lý ngôn ngữ tự nhiên.

PyTorch là một framework học sâu (deep learning) phổ biến, được sử dụng rộng rãi trong nghiên cứu và phát triển các mô hình xử lý ngôn ngữ tự nhiên (NLP). Một số nội dung ứng dụng và kỹ thuật quan trọng của PyTorch trong xử lý ngôn ngữ tự nhiên bao gồm: Embedding từ ngữ, các mô hình tuần hoàn, mô hình transformer, tiền xử lý ngôn ngữ, huấn luyện mô hình và tối ưu hóa, cuối cùng là các ứng dụng của Pytorch. Tuy nhiên có nhiều phần đã được đề cập đến ở phần trước nên ở phần này chúng ta sẽ tìm hiểu về:

Mô hình transformer

Tiền xử lý ngôn ngữ (Padding và Truncating)

Huấn luyện mô hình và tối ưu hóa: Cross-Entropy Loss, SGD

Các ứng dụng của Pytorch

2.5.1. Mô hình Transformer.

- Transformer – một mô hình tiên tiến cho xử lý ngôn ngữ tự nhiên, đặc biệt hiệu quả trong việc xử lý ngữ cảnh dài và đồng thời huấn luyện nhanh hơn so với các mô hình tuần hoàn. Transformer là một kiến trúc mạng nơ-ron sâu được giới thiệu lần đầu trong bài báo "Attention is All You Need" của Vaswani et al. năm 2017. Nó đã cách mạng hóa nhiều lĩnh vực trong xử lý ngôn ngữ tự nhiên (NLP) và học sâu nhờ khả năng xử lý các chuỗi dữ liệu một cách hiệu quả mà không cần sử dụng các mạng tuần tự như RNN hay LSTM.

+ Kiến trúc tổng quan của Transformer. Transformer bao gồm hai thành phần chính:

- Encoder: Mã hóa đầu vào thành các biểu diễn ngữ cảnh.
- Decoder: Giải mã các biểu diễn ngữ cảnh để tạo ra đầu ra.

Cả Encoder và Decoder đều bao gồm nhiều lớp (layers), mỗi lớp có cấu trúc tương tự và được xây dựng từ các thành phần cơ bản sau:

+ Các thành phần cơ bản của Transformer

- Self-Attention Mechanism (Cơ chế tự chú ý): Cho phép mỗi từ trong câu chú ý đến các từ khác trong cùng câu. Cơ chế này giúp mô hình nắm bắt được mối quan hệ ngữ nghĩa giữa các từ bất kể khoảng cách giữa chúng. Chúng ta có hai khái niệm:

Scaled Dot-Product Attention: Là lõi của cơ chế tự chú ý, tính toán điểm chú ý (attention scores) dựa trên các vector truy vấn (query), khóa (key), và giá trị (value).

Multi-Head Attention: Sử dụng nhiều "đầu" chú ý (attention heads) để học các mối quan hệ ngữ nghĩa từ nhiều góc độ khác nhau.

- Feed-Forward Neural Network (FFN): Sau khi các từ được mã hóa bởi cơ chế tự chú ý, chúng được đưa qua một mạng nơ-ron đơn giản để tăng cường khả năng biểu diễn.

- Positional Encoding: Transformer không có cơ chế tự nhiên để xử lý vị trí của các từ trong chuỗi, do đó cần thêm các vector mã hóa vị trí để thông tin về thứ tự của các từ được đưa vào mô hình.

+ Hoạt động của Transformer

Encoder: Nhận đầu vào là một chuỗi các từ, được chuyển đổi thành các vector embedding. Vector embedding được kết hợp với vector positional encoding để giữ thông tin vị trí của các từ. Các vector này được đưa qua nhiều lớp self-attention và feed-forward neural network để tạo ra các biểu diễn ngữ cảnh cho từng từ.

Decoder: Nhận đầu vào là các biểu diễn ngữ cảnh từ encoder và các từ đã dịch trước đó (trong quá trình huấn luyện hoặc dịch thực tế). Các biểu diễn này được xử lý qua các lớp self-attention, encoder-decoder attention, và feed-forward neural network để tạo ra các từ tiếp theo trong chuỗi đầu ra.

+ Ứng dụng của Transformer

Transformer đã chứng minh hiệu quả vượt trội trong nhiều ứng dụng:

- Dịch máy (Machine Translation): Ví dụ điển hình là Google Translate sử dụng kiến trúc Transformer để cải thiện độ chính xác và tốc độ dịch.
 - Tóm tắt văn bản (Text Summarization): Giúp tóm tắt nội dung các bài báo hoặc tài liệu dài một cách hiệu quả.
 - Sinh văn bản (Text Generation): Sử dụng các biến thể của Transformer như GPT-3 để tạo ra văn bản tự động.
 - Phân loại văn bản (Text Classification): Sử dụng Transformer để phân loại các tài liệu hoặc bài viết dựa trên nội dung.
- BERT (Bidirectional Encoder Representations from Transformers) và GPT (Generative Pre-trained Transformer): Là các ứng dụng của Transformer, được sử dụng rộng rãi trong nhiều bài toán NLP như phân loại văn bản, dịch máy, và tạo văn bản.

+ Bidirectional Encoder Representations from Transformers (BERT)

BERT là một mô hình ngôn ngữ được phát triển bởi Google, nổi bật với khả năng hiểu ngữ cảnh hai chiều của từ trong câu. Thay vì đọc văn bản từ trái sang phải hoặc từ phải sang trái như các mô hình trước đây, BERT đọc văn bản theo cả hai hướng cùng lúc. Điều này giúp BERT nắm bắt ngữ nghĩa và ngữ cảnh tốt hơn.

1. Kiến trúc của BERT

BERT sử dụng kiến trúc Transformer với nhiều lớp encoder. Các lớp này giúp mã hóa văn bản đầu vào thành các biểu diễn ngữ cảnh.

- Bidirectional: Đặc điểm nổi bật nhất của BERT là khả năng đọc từ cả hai chiều.
- Masked Language Model (MLM): Trong quá trình huấn luyện, BERT che một số từ trong câu và dự đoán các từ đó dựa trên ngữ cảnh xung quanh.
- Next Sentence Prediction (NSP): BERT cũng được huấn luyện để dự đoán xem liệu một câu có phải là câu tiếp theo của một câu khác hay không.

1. Ứng dụng của BERT

- Phân loại văn bản: Như phân loại cảm xúc, phân loại email, hoặc phân loại tin tức.
- Trả lời câu hỏi: Giúp trả lời các câu hỏi dựa trên một đoạn văn bản đã cho.
- Nhận dạng thực thể có tên (NER): Tìm kiếm và phân loại các thực thể như tên người, địa điểm trong văn bản.

```

from transformers import BertTokenizer, BertModel

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')

# Token hóa và mã hóa văn bản
input_text = "BERT là một mô hình mạnh mẽ."
inputs = tokenizer(input_text, return_tensors="pt")
outputs = model(**inputs)

print(outputs.last_hidden_state.shape)

```

Hình 2.4: Ứng dụng của BERT trong pytorch

+ Generative Pre-trained Transformer (GPT)

GPT là một loạt các mô hình ngôn ngữ được phát triển bởi OpenAI. GPT nổi bật với khả năng sinh văn bản (text generation), sử dụng kiến trúc Transformer với cơ chế tự chú ý. Khác với BERT, GPT chủ yếu là một mô hình sinh văn bản, chỉ đọc văn bản từ trái sang phải.

1. Kiến trúc của GPT

GPT cũng sử dụng kiến trúc Transformer, nhưng chỉ bao gồm các lớp decoder. Các lớp này giải mã đầu vào thành các đoạn văn bản tiếp theo.

- Unidirectional: GPT đọc văn bản theo một chiều (từ trái sang phải).
- Pre-training: GPT được huấn luyện trước trên một lượng lớn dữ liệu văn bản để học các cấu trúc ngữ pháp, kiến thức thế giới và các mẫu văn bản.
- Fine-tuning: Sau quá trình huấn luyện trước, GPT có thể được tinh chỉnh cho các nhiệm vụ cụ thể như dịch máy, tạo văn bản, hoặc trả lời câu hỏi.

2. Ứng dụng của GPT

- Sinh văn bản: Tạo ra các đoạn văn bản tự động với chất lượng cao.
- Hội thoại AI: GPT có thể được sử dụng để tạo ra các hội thoại tự động, như chatbot.
- Tóm tắt văn bản: Tóm tắt các đoạn văn bản dài thành các đoạn ngắn gọn hơn.

```

from transformers import GPT2Tokenizer, GPT2LMHeadModel

tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
model = GPT2LMHeadModel.from_pretrained('gpt2')

# Token hóa và mã hóa văn bản
input_text = "GPT có thể tạo ra văn bản tuyệt vời."
inputs = tokenizer(input_text, return_tensors="pt")
outputs = model(**inputs, labels=inputs["input_ids"])

# Tính toán loss
loss = outputs.loss
print(loss)

```

Hình 2.5 : Ứng dụng GPT trong pytorch

BERT và GPT đều là các mô hình ngôn ngữ mạnh mẽ dựa trên kiến trúc Transformer, nhưng chúng có những mục tiêu và cách tiếp cận khác nhau. BERT chủ yếu tập trung vào việc hiểu ngữ cảnh và ngữ nghĩa hai chiều, trong khi GPT tập trung vào việc sinh văn bản và hiểu ngữ cảnh đơn chiều. Cả hai mô hình đã cách mạng hóa nhiều ứng dụng trong xử lý ngôn ngữ tự nhiên và học sâu.

2.5.2. Tiền xử lý ngôn ngữ.

Padding và Truncating: Điều chỉnh độ dài của các chuỗi văn bản để phù hợp với kích thước batch trong quá trình huấn luyện mô hình. Trong xử lý ngôn ngữ tự nhiên, các câu và đoạn văn bản thường có độ dài khác nhau. Để các mô hình học sâu như RNN, LSTM hoặc Transformer có thể xử lý dữ liệu một cách hiệu quả, chúng ta cần chuẩn hóa độ dài của các chuỗi này. Hai kỹ thuật phổ biến để làm điều này là Padding và Truncating. Dưới đây là giải thích chi tiết về từng kỹ thuật:

+ **Padding** là kỹ thuật thêm các ký tự đặc biệt (thường là số 0) vào cuối hoặc đầu các chuỗi để tất cả các chuỗi có độ dài bằng nhau. Điều này giúp cho việc xử lý các batch dữ liệu trở nên đồng nhất.

Ví dụ, giả sử chúng ta có các chuỗi với độ dài khác nhau:

- "cat"
- "cats"

- "categorically"

Để chuẩn hóa các chuỗi này thành độ dài 10, chúng ta sẽ thêm padding:

- "cat " (7 padding)
- "cats " (6 padding)
- "categorically"

+ Truncating là kỹ thuật cắt bớt các chuỗi văn bản dài hơn một độ dài cố định. Điều này giúp đảm bảo tất cả các chuỗi có độ dài tối đa bằng nhau.

Ví dụ, nếu chúng ta đặt độ dài cố định là 5:

- "cat" sẽ giữ nguyên vì độ dài là 3.
- "cats" sẽ giữ nguyên vì độ dài là 4.
- "categorically" sẽ bị cắt thành "categ".

Padding và Truncating là hai kỹ thuật quan trọng trong xử lý ngôn ngữ tự nhiên để chuẩn hóa độ dài các chuỗi dữ liệu. Padding giúp thêm các ký tự đặc biệt vào chuỗi để chúng có cùng độ dài, trong khi Truncating giúp cắt bớt các chuỗi quá dài. Sử dụng PyTorch, chúng ta có thể dễ dàng thực hiện các kỹ thuật này để chuẩn bị dữ liệu cho các mô hình học sâu.

2.5.3. Huấn luyện mô hình và tối ưu hóa.

- Loss Functions: PyTorch cung cấp nhiều loại hàm mất mát (loss function) phù hợp với các bài toán NLP như Cross-Entropy Loss cho bài toán phân loại. Cross-Entropy Loss là một hàm mất mát phổ biến trong học sâu, đặc biệt là trong các bài toán phân loại. Trong xử lý ngôn ngữ tự nhiên (NLP), Cross-Entropy Loss được sử dụng rộng rãi để đo lường sự khác biệt giữa phân phối xác suất dự đoán của mô hình và phân phối xác suất thực tế của các nhãn lớp.

Cross-Entropy Loss thường được sử dụng trong các bài toán phân loại văn bản, phân tích cảm xúc, nhận dạng thực thể có tên (NER), và nhiều bài toán khác trong NLP. Các mô hình phổ biến như RNN, LSTM, Transformer, BERT, và GPT đều sử dụng Cross-Entropy Loss để huấn luyện.

```

import torch
import torch.nn as nn

# Khởi tạo dữ liệu mẫu
inputs = torch.tensor([[0.1, 0.2, 0.7], [0.3, 0.5, 0.2]], requires_grad=True)
labels = torch.tensor([2, 1]) # Các nhãn thực tế (0, 1, 2)

# Khởi tạo hàm mất mát Cross-Entropy Loss
criterion = nn.CrossEntropyLoss()

# Tính toán giá trị mất mát
loss = criterion(inputs, labels)
print(f'Loss: {loss.item():.4f}')

# Tính gradient và cập nhật trọng số mô hình
loss.backward()

```

Hình 2.6: Ví dụ Cross-Entropy Loss trong pytorch

Cross-Entropy Loss là một hàm mất mát mạnh mẽ và phổ biến trong học sâu, đặc biệt trong các bài toán xử lý ngôn ngữ tự nhiên như phân loại văn bản, phân tích cảm xúc, và nhận dạng thực thể có tên. Nó đo lường sự khác biệt giữa phân phối xác suất dự đoán và phân phối xác suất thực tế, giúp tối ưu hóa mô hình để đạt được dự đoán chính xác hơn. PyTorch cung cấp các công cụ mạnh mẽ để sử dụng Cross-Entropy Loss, giúp đơn giản hóa quá trình huấn luyện và tối ưu hóa mô hình trong NLP.

- **Optimizers:** Sử dụng thuật toán tối ưu hóa như Stochastic Gradient Descent (SGD) để cập nhật trọng số mô hình trong quá trình huấn luyện. SGD là một trong những thuật toán tối ưu hóa cơ bản và phổ biến nhất được sử dụng trong huấn luyện các mô hình học sâu. Trong xử lý ngôn ngữ tự nhiên (NLP), SGD và các biến thể của nó như Mini-batch SGD, SGD with Momentum, và Adaptive Methods (như Adam) đóng vai trò quan trọng trong việc tối ưu hóa các mô hình.

SGD là một phương pháp tối ưu hóa dựa trên gradient descent, nhưng thay vì sử dụng toàn bộ dữ liệu để tính toán gradient, SGD chỉ sử dụng một mẫu ngẫu nhiên tại mỗi bước.

Ưu điểm:

- Tính toán nhanh: SGD sử dụng ít tài nguyên tính toán hơn so với Gradient Descent truyền thống vì chỉ cần cập nhật trên một mẫu tại mỗi bước.
- Khả năng tổng quát hóa: Việc cập nhật theo các mẫu ngẫu nhiên giúp SGD tránh được việc rơi vào các cực tiểu cục bộ và có khả năng tổng quát hóa tốt hơn.

Hạn chế:

- Dao động lớn: Do chỉ sử dụng một mẫu ngẫu nhiên, các cập nhật của SGD có thể gây ra dao động lớn trong quá trình huấn luyện.
- Cần điều chỉnh hyperparameters: Tốc độ học và các thông số khác cần được điều chỉnh cẩn thận để đảm bảo hiệu quả tối ưu.

```
# Định nghĩa mô hình đơn giản
class SimpleModel(nn.Module):
    def __init__(self, input_size, output_size):
        super(SimpleModel, self).__init__()
        self.fc = nn.Linear(input_size, output_size)

    def forward(self, x):
        return self.fc(x)

# Khởi tạo mô hình, hàm mất mát và optimizer
model = SimpleModel(input_size=10, output_size=2)
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)

# Tạo dữ liệu mẫu
inputs = torch.randn(5, 10) # batch_size=5, input_size=10
labels = torch.tensor([0, 1, 1, 0, 1]) # Các nhãn thực tế

# Huấn luyện mô hình với SGD
for epoch in range(100):
    optimizer.zero_grad() # Xóa gradient cũ
    outputs = model(inputs) # Dự đoán
    loss = criterion(outputs, labels) # Tính hàm mất mát
    loss.backward() # Tính gradient
    optimizer.step() # Cập nhật trọng số

    if (epoch + 1) % 10 == 0:
        print(f'Epoch [{epoch+1}/100], Loss: {loss.item():.4f}')
```

Hình 2.7: Ví dụ SDG trong Pytorch

SGD và các biến thể của nó là các thuật toán tối ưu hóa quan trọng trong học sâu và xử lý ngôn ngữ tự nhiên. Chúng giúp tối ưu hóa các mô hình một cách hiệu quả, đồng thời giảm thiểu tài nguyên tính toán và tăng khả năng tổng quát hóa. PyTorch cung cấp các công cụ mạnh mẽ để sử dụng SGD và các biến thể của nó, giúp đơn giản hóa quá trình huấn luyện và tối ưu hóa mô hình trong NLP.

2.5.4. Các ứng dụng của Pytorch.

- Phân loại văn bản: Sử dụng các mô hình học sâu để phân loại văn bản vào các nhóm khác nhau, ví dụ như phân tích cảm xúc, phân loại tin tức.
- Dịch máy: Ứng dụng các mô hình sequence-to-sequence (Seq2Seq) với cơ chế Attention để dịch văn bản từ ngôn ngữ này sang ngôn ngữ khác.
- Tạo văn bản (Text Generation): Sử dụng mô hình GPT để tạo ra các đoạn văn bản mới dựa trên ngữ cảnh đã cho.
- Trích xuất thông tin (Information Extraction): Sử dụng các mô hình Named Entity Recognition (NER) để trích xuất các thực thể có tên trong văn bản.

CHƯƠNG 3. THỰC NGHIỆM XÂY DỰNG MÔ HÌNH

Bây giờ chúng ta đã hiểu cơ bản về những gì cần xây dựng, chúng ta sẽ thử triển khai nó. Chúng ta sẽ xây dựng một bộ phân loại phân tích tình cảm đơn giản trên các bài đánh giá phim, bộ phân loại này sẽ phân loại xem đánh giá của người dùng về bộ phim là tích cực, tiêu cực hay trung tính. Đối với dự án python phân tích tình cảm này, chúng ta sẽ sử dụng tập dữ liệu đánh giá phim IMDB.

3.1. Bộ dữ liệu thực nghiệm.

Bộ dữ liệu mà chúng ta sẽ sử dụng trong đề tài phân tích tình cảm này là cơ sở dữ liệu phim quốc tế IMDB với 50.000 đánh giá về phim từ khắp nơi trên thế giới, đây là bộ dữ liệu phân loại nhị phân phân loại từng đánh giá theo hướng tích cực hoặc tiêu cực. Nó có 25000 mẫu để đào tạo và 25000 mẫu để thử nghiệm. Bạn không cần phải tải xuống riêng cho dự án này nhưng bạn có thể xem trên trang web chính thức của nó . Vì đây là tập dữ liệu văn bản nên nó khá nhẹ, khoảng 80MB.

Chúng ta sẽ mã hóa tất cả những thứ này trong một jupyter notebook trên google colab để sử dụng GPU miễn phí. Nếu bạn tự làm theo trên hệ thống của mình, mọi thứ sẽ khá giống nhau ngoại trừ việc gắn ổ đĩa google để sử dụng làm tùy chọn lưu trữ liên tục. Vì vậy, chúng ta bắt đầu bằng cách gắn Google Drive và điều hướng đến thư mục mà chúng ta muốn làm việc.



```
[ ] 1 import os
    2 from google.colab import drive
    3 drive.mount('/content/drive')
    4
    5
```

Mounted at /content/drive

```
1
2 os.chdir('/content/drive/MyDrive/Techvidvan/Sentiment')
3 !ls
```

aclImdb_v1 tut2-model.pt

Hình 3.1: Lệnh điều hướng đến thư mục chứa dữ liệu.

3.2. Chuẩn bị dữ liệu.

Chúng ta sẽ sử dụng pytorch cho dự án này và may mắn là nó được cài đặt sẵn một số chức năng giúp chúng ta tăng tốc công việc. Thư viện torch.text là một công cụ tuyệt vời cho các dự án xử lý ngôn ngữ tự nhiên. Nó có trình tải cho một số tập dữ liệu xử lý ngôn ngữ tự nhiên phổ biến như tập chúng ta sẽ sử dụng hôm nay, cũng như quy trình hoàn chỉnh để trừu tượng hóa véc-tơ hóa dữ liệu, tải dữ liệu và lặp lại dữ liệu.

```
[ ] 1 import random
    2 import torch
    3 # from torchtext.legacy import data
    4 # from torchtext.legacy import datasets
    5 from torchtext import data
    6 from torchtext import datasets
    7
    8 seed = 42
    9
   10 torch.manual_seed(seed)
   11 torch.backends.cudnn.deterministic = True
   12 device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
   13
   14 txt = data.Field(tokenize = 'spacy',
   15                  tokenizer_language = 'en_core_web_sm',
   16                  include_lengths = True)
   17
   18 labels = data.LabelField(dtype = torch.float)
```

Hình 3.2: Chuẩn bị và xử lý trước dữ liệu.

Chúng ta sẽ sử dụng phương thức field của lớp data để quyết định cách dữ liệu cần được xử lý trước. Các tham số chúng ta truyền vào đó sẽ xác định quá trình xử lý trước, chúng ta chỉ cần điều chỉnh một số tham số và để phần còn lại theo mặc định. Tham số đầu tiên, tokenize của tôi (xác định cách các câu sẽ được chia nhỏ hoặc mã hóa theo tiêu chuẩn NLP) là spacy, đây là một công cụ mạnh mẽ để mã hóa một dòng. Tôi khuyên bạn nên sử dụng công cụ này, nhưng mặc định chỉ là mã hóa chuỗi dựa trên khoảng trắng. Ngoài ra, chúng ta cần cho spacy tokenize biết mô hình ngôn ngữ nào sẽ được sử dụng cho tác vụ. Tôi cũng đặt giá trị khởi tạo của trình tạo số ngẫu nhiên thành một số nhất định, có thể là bất kỳ số nào nhưng chúng ta chỉ đề cập đến nó cho mục đích tái tạo. Bạn có thể thay đổi nó hoặc thậm chí bỏ qua nó mà không có bất kỳ tác động đáng kể nào.

```
[ ] 1 train_data, test_data = datasets.IMDB.splits(txt, labels)
    2 train_data, valid_data = train_data.split(random_state = random.seed(seed))
    3
    4 num_words = 25_000
    5
    6 txt.build_vocab(train_data,
    7                 max_size = num_words,
    8                 vectors = "glove.6B.100d",
    9                 unk_init = torch.Tensor.normal_)
    10
    11 labels.build_vocab(train_data)
```

Hình 3.3: Phân chia tập dữ liệu.

Ở đây, tôi đã tải xuống tập dữ liệu IMDB để phân tích tình cảm và chia thành tập kiểm tra và tập đào tạo. Sau đó, tập đào tạo được tách ra một phần để làm tập xác thực. Chúng ta tiếp tục giới hạn số lượng từ mà mô hình sẽ học được ở mức 25000, điều này sẽ chọn 25000 từ được sử dụng nhiều nhất từ tập dữ liệu và sử dụng chúng để đào tạo. Giảm đáng kể khối lượng công việc của mô hình mà không làm mất đi độ chính xác thực sự.

```
[ ] 1 batch_size = 64
    2
    3 train_itr, valid_itr, test_itr = data.BucketIterator.splits(
    4     (train_data, valid_data, test_data),
    5     batch_size = batch_size,
    6     sort_within_batch = True,
    7     device = device)
```

Hình 3.4: Tạo một lô đào tạo.

Hiện tại, chúng ta đang tạo một lô đào tạo, thử nghiệm và xác thực từ dữ liệu mà chúng ta có để chuẩn bị đưa vào mô hình dưới dạng lô 64 mẫu cùng một lúc. Giảm số lượng này nếu bạn gặp lỗi hết bộ nhớ.

3.3. Định nghĩa mô hình phân tích tình cảm Recurrent Neural Network.

Bây giờ chúng ta chuẩn bị mô hình và xác định kiến trúc của nó. Chúng ta đang sử dụng LSTM-RNN cho nhiệm vụ của mình. Chúng ta sẽ sử dụng RNN song hướng nhiều lớp. Điều đó có nghĩa là sẽ có nhiều lớp RNN xếp chồng lên nhau. RNN song hướng có lợi thế là có nhiều ngữ cảnh hơn so với mạng một hướng. Ví dụ, trong một câu chảy về phía trước qua một mô hình, nếu một mô hình phải đoán từ tiếp theo, nó sẽ thực hiện dựa trên kiến thức trước đó. Nhưng

trong một mạng song hướng, nó cũng sẽ có kiến thức về những gì tiếp theo do hai mạng chảy theo các hướng ngược nhau được xếp chồng lên nhau. Đầu vào chảy theo hướng ngược nhau cũng như trình tự. Một câu "tôi yêu HaUI" sẽ chảy trong mạng đầu tiên là "tôi", "yêu", "HaUI" nhưng trong mạng thứ hai, nó sẽ chảy như "HaUI", "yêu", "tôi". Điều này cung cấp ngữ cảnh và mối quan hệ tốt hơn trong dữ liệu với mạng. Trạng thái ẩn đầu ra từ lớp đầu tiên sẽ là đầu vào cho lớp tiếp theo, ngoài ra mỗi lớp có các nút ẩn độc lập.

Tôi sử dụng các lớp `nn.LSTM` thay vì các lớp `nn.RNN` thông thường. Chúng ta sẽ đưa mọi thứ qua một lớp nhúng, chỉ là một biểu diễn của các từ trong không gian chiều thấp hơn. Nói một cách đơn giản là sắp xếp các từ sao cho các từ tương tự được nhóm lại với nhau.

```

1 class RNN(nn.Module):
2     def __init__(self, word_limit, dimension_embedding, dimension_hidden,
3                   dimension_output, num_layers,
4                   bidirectional, dropout, pad_idx):
5         super().__init__()
6         self.embedding = nn.Embedding(word_limit,
7                                       dimension_embedding,
8                                       padding_idx = pad_idx)
9         self.rnn = nn.LSTM(dimension_embedding,
10                            dimension_hidden,
11                            num_layers=num_layers,
12                            bidirectional=bidirectional,
13                            dropout=dropout)
14         self.fc = nn.Linear(dimension_hidden * 2, dimension_output)
15         self.dropout = nn.Dropout(dropout)
16     def forward(self, text, len_txt):
17         embedded = self.dropout(self.embedding(text))
18         packed_embedded = nn.utils.rnn.pack_padded_sequence(embedded,
19                                                             len_txt.to('cpu'))
20         packed_output, (hidden, cell) = self.rnn(packed_embedded)
21         output, output_lengths = nn.utils.rnn.pad_packed_sequence(packed_output)
22         hidden = self.dropout(torch.cat((hidden[-2,:,:], hidden[-1,:,:]),
23                                         dim = 1))
24         return self.fc(hidden)

```

Hình 3.5: Định nghĩa mô hình phân tích tình cảm RNN.

Chúng ta định nghĩa các tham số cho mô hình phân tích tình cảm và truyền nó đến một thể hiện của lớp mô hình mà chúng ta vừa định nghĩa. Số lượng tham số đầu vào, lớp ẩn và chiều đầu ra cùng với tỷ lệ thông lượng và boolean song hướng được định nghĩa. Chúng ta cũng truyền chỉ mục mã thông báo từ vựng mà ta đã tạo trước đó.


```

1 dimension_input = len(txt.vocab)
2 dimension_embedding = 100
3 dimension_hddn = 256
4 dimension_out = 1
5 layers = 2
6 bidirectional = True
7 dropout = 0.5
8 idx_pad = txt.vocab.stoi[txt.pad_token]
9
10 model = RNN(dimension_input,|
11             dimension_embedding,
12             dimension_hddn,
13             dimension_out,
14             layers,
15             bidirectional,
16             dropout,
17             idx_pad)

```

Hình 3.6: Truyền tham số cho mô hình.

Bây giờ, chúng ta làm rõ một số chi tiết về mô hình của mình. Lấy số lượng tham số có thể đào tạo được có trong mô hình. Sau đó, chúng ta lấy các trọng số nhúng được đào tạo trước và sao chép chúng vào mô hình của mình để mô hình không cần phải học các nhúng đó và có thể tập trung trực tiếp vào công việc đang thực hiện là học các tình cảm liên quan đến các nhúng đó. Trọng số nhúng được đào tạo trước sẽ được đặt thay cho trọng số ban đầu.

```

1 def count_parameters(model):
2     return sum(p.numel() for p in model.parameters() if p.requires_grad)
3
4 pretrained_embeddings = txt.vocab.vectors
5
6 model.embedding.weight.data.copy_(pretrained_embeddings)
7
8 unique_id = txt.vocab.stoi[txt.unk_token]
9
10 model.embedding.weight.data[unique_id] = torch.zeros(dimension_embedding)
11 model.embedding.weight.data[idx_pad] = torch.zeros(dimension_embedding)

```

Hình 3.7: Lấy các trọng số nhúng được đào tạo trước.

Bây giờ chúng ta định nghĩa một số tham số liên quan đến mô hình, đó là trình tối ưu hóa mà chúng ta sẽ sử dụng và tiêu chí mất mát mà ta cần. Tôi đã chọn trình tối ưu hóa adam để hội tụ nhanh mô hình cùng với hàm mất mát logistic. Ta đặt mô hình và tiêu chí trên GPU.

```
[ ] 1 import torch.optim as optim
    2
    3 optimizer = optim.Adam(model.parameters())
    4
    5 criterion = nn.BCEWithLogitsLoss()
    6
    7 model = model.to(device)
    8 criterion = criterion.to(device)
```

Hình 3.8: Hàm tối ưu hóa và hàm mất mát.

3.4. Huấn luyện mô hình.

Bây giờ chúng ta bắt đầu các chức năng cần thiết để đào tạo và đánh giá mô hình phân tích tình cảm. Đầu tiên là hàm độ chính xác nhị phân, chúng ta sẽ sử dụng hàm này để có được độ chính xác của mô hình mỗi lần chạy.

```
[ ] 1 def bin_acc(preds, y):
    2
    3     predictions = torch.round(torch.sigmoid(preds))
    4     correct = (predictions == y).float()
    5     acc = correct.sum() / len(correct)
    6     return acc
```

Hình 3.9: Hàm độ chính xác nhị phân.

Bây giờ, chúng ta định nghĩa hàm để đào tạo và đánh giá các mô hình. Quy trình ở đây là chuẩn. Ta bắt đầu bằng cách lặp qua số kỷ nguyên và số lần lặp trong mỗi kỷ nguyên theo kích thước lô mà chúng ta đã định nghĩa trước đó. Chúng ta truyền văn bản cho mô hình, lấy dự đoán từ mô hình, tính toán tổn thất cho mỗi lần lặp và sau đó truyền ngược tổn thất đó.

```
[ ] 1 def train(model, itr, optimizer, criterion):
    2     epoch_loss = 0
    3     epoch_acc = 0
    4     model.train()
    5     for i in itr:
    6         optimizer.zero_grad()
    7         text, len_txt = i.text
    8         predictions = model(text, len_txt).squeeze(1)
    9         loss = criterion(predictions, i.label)
   10         acc = bin_acc(predictions, i.label)
   11         loss.backward()
   12         optimizer.step()
   13         epoch_loss += loss.item()
   14         epoch_acc += acc.item()
   15     return epoch_loss / len(itr), epoch_acc / len(itr)
```

Hình 3.10: Hàm đào tạo.

Sự thay đổi lớn duy nhất trong hàm đánh giá so với hàm đào tạo là chúng ta không truyền ngược lại tổn thất qua mô hình và sử dụng `torch.no_grad` về cơ bản có nghĩa là không có sự giảm dần độ dốc trong khi đánh giá.

```

1 def evaluate(model, itr, criterion):
2     epoch_loss = 0
3     epoch_acc = 0
4     model.eval()
5     with torch.no_grad():
6         for i in itr:
7             text, len_txt = i.text
8             predictions = model(text, len_txt).squeeze(1)
9             loss = criterion(predictions, i.label)
10            acc = bin_acc(predictions, i.label)
11            epoch_loss += loss.item()
12            epoch_acc += acc.item()
13    return epoch_loss / len(itr), epoch_acc / len(itr)

```

Hình 3.11: Hàm đánh giá.

Chúng ta xây dựng một hàm trợ giúp `epoch_time` để tính toán thời gian mỗi epoch cần để hoàn thành lượt chạy của nó và in ra. Ở đây tôi đặt số epoch là 5 và sau đó bắt đầu quá trình đào tạo của mình. Thêm mất mát đào tạo và xác thực ở mỗi giai đoạn, nếu chúng ta cần hiểu hoặc vẽ đường cong đào tạo tại một thời điểm sau đó. Ta sẽ lưu mô hình phân tích tình cảm python có mất mát xác thực tốt nhất.

```

1 def epoch_time(start_time, end_time):
2     used_time = end_time - start_time
3     used_mins = int(used_time / 60)
4     used_secs = int(used_time - (used_mins * 60))
5     return used_mins, used_secs
6
7 num_epochs = 5
8
9 best_valid_loss = float('inf')
10
11 for epoch in range(num_epochs):
12     start_time = time.time()
13     train_loss, train_acc = train(model, train_itr, optimizer, criterion)
14     valid_loss, valid_acc = evaluate(model, valid_itr, criterion)
15     end_time = time.time()
16     epoch_mins, epoch_secs = epoch_time(start_time, end_time)
17     if valid_loss < best_valid_loss:
18         best_valid_loss = valid_loss
19         torch.save(model.state_dict(), 'tut2-model.pt')
20     print(f'Epoch: {epoch+1:02} | Epoch Time: {epoch_mins}m {epoch_secs}s')
21     print(f'\tTrain Loss: {train_loss:.3f} | Train Acc: {train_acc*100:.2f}%')
22     print(f'\tVal. Loss: {valid_loss:.3f} | Val. Acc: {valid_acc*100:.2f}%')

```

Hình 3.12: Hàm trợ giúp `epoch_time`.

3.5. Xử lý từng câu hỏi để lấy ra các hình tròn chứa đáp án.

Chúng ta tải điểm kiểm tra đã lưu của mô hình và kiểm tra nó trên bộ kiểm tra mà chúng ta đã tạo trước đó. Trong quá trình chạy thử mô hình phân tích tình cảm python, chúng ta đã đạt được điểm chính xác khá tốt khoảng 85,87%. Chúng ta cũng có thể kiểm tra mô hình trên dữ liệu của mình. Mô hình này được đào tạo để phân loại các đánh giá phim thành tích cực, tiêu cực và trung lập, do đó chúng ta sẽ chuyển dữ liệu có thể liên quan đến nó để kiểm tra. Vì vậy, đối với điều đó, chúng ta sẽ nhập và tải spacy để mã hóa dữ liệu mà chúng ta cần cung cấp cho mô hình. Ban đầu, trong khi xác định tiền xử lý, chúng ta đã sử dụng torch.text tích hợp spacy, nhưng ở đây chúng ta không sử dụng các lô và tiền xử lý mà chúng ta cần thực hiện có thể được xử lý bởi thư viện spacy. Chúng ta xác định một hàm dự đoán tình cảm cho việc này. Sau khi tiền xử lý, chúng ta chuyển đổi nó thành ten-xơ và sẵn sàng để chuyển đến mô hình.

```
[ ] 1 import spacy
    2 nlp = spacy.load('en_core_web_sm')
    3
    4 def pred(model, sentence):
    5     model.eval()
    6     tokenized = [tok.text for tok in nlp.tokenizer(sentence)]
    7     indexed = [txt.vocab.stoi[t] for t in tokenized]
    8     length = [len(indexed)]
    9     tensor = torch.LongTensor(indexed).to(device)
   10     tensor = tensor.unsqueeze(1)
   11     length_tensor = torch.LongTensor(length)
   12     prediction = torch.sigmoid(model(tensor, length_tensor))
   13     return prediction.item()
```

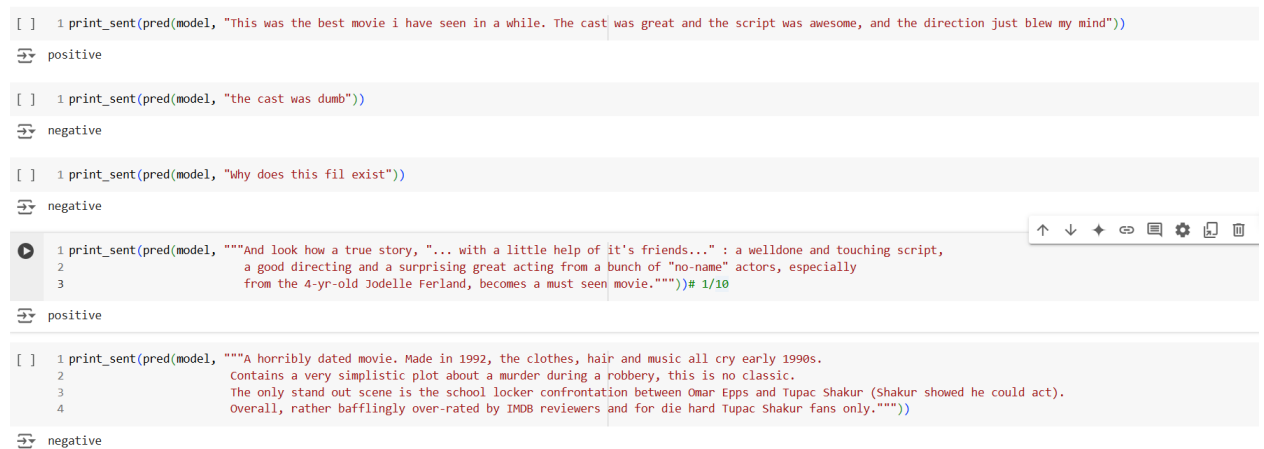
Hình 3.13: Tiền xử lý đầu vào.

Tôi định nghĩa một hàm trợ giúp khác sẽ in ra cảm nghĩ của bình luận dựa trên điểm số mà mô hình cung cấp “positive” – tích cực (0 đến 0,3 điểm), “neutral” – trung tính (trên 0.3 đến 0.7 điểm), “negative” – tiêu cực (trên 0.7 điểm).

```
[ ] 1 sent=["positive", "neutral", "negative"]
    2 def print_sent(x):
    3     if (x<0.3): print(sent[0])
    4     elif (x>0.3 and x<0.7): print(sent[1])
    5     else: print(sent[2])
```

Hình 3.14: Hàm in kết quả.

Bây giờ chúng ta chỉ cần truyền bất kỳ dữ liệu nào và kiểm tra xem mô hình nghĩ gì về nó.



```
[ ] 1 print_sent(pred(model, "This was the best movie i have seen in a while. The cast was great and the script was awesome, and the direction just blew my mind"))
positive

[ ] 1 print_sent(pred(model, "the cast was dumb"))
negative

[ ] 1 print_sent(pred(model, "why does this fil exist"))
negative

1 print_sent(pred(model, ""And look how a true story, "... with a little help of it's friends..." : a welldone and touching script,
2 a good directing and a surprising great acting from a bunch of "no-name" actors, especially
3 from the 4-yr-old Jodelle Ferland, becomes a must seen movie."""))# 1/10
positive

[ ] 1 print_sent(pred(model, ""A horribly dated movie. Made in 1992, the clothes, hair and music all cry early 1990s.
2 Contains a very simplistic plot about a murder during a robbery, this is no classic.
3 The only stand out scene is the school locker confrontation between Omar Epps and Tupac Shakur (Shakur showed he could act).
4 Overall, rather bafflingly over-rated by IMDB reviewers and for die hard Tupac Shakur fans only."""))
negative
```

Hình 3.15: Kết quả thực nghiệm

3.6. Một số so sánh cùng kết luận giữa đề án và công trình “Sentiment Analysis of Movie Reviews” của tác giả Charlie Chengrui Zheng.

- Về bộ dữ liệu: Đề án này sử dụng bộ dữ liệu IMDB với 50.000 bài đánh giá về các bộ phim lớn hơn nhiều so với bộ dữ liệu gồm 1000 bài đánh giá trong công trình “Sentiment Analysis of Movie Reviews” của tác giả Charlie Chengrui Zheng. IMDB là một bộ dữ liệu cân bằng với 25.000 đánh giá mang cảm xúc tiêu cực và 25.000 đánh giá mang cảm xúc tích cực.
- Về thư viện Natural Language Toolkit (NLTK) và PyTorch :
 - + NLTK:
 - Mục đích: NLTK là một thư viện Python mạnh mẽ được thiết kế dành riêng cho nghiên cứu và phát triển các ứng dụng xử lý ngôn ngữ tự nhiên.
 - Khả năng: Cung cấp một bộ công cụ phong phú cho các tác vụ NLP cơ bản như tokenization, stemming, lemmatization, tagging, parsing, và nhiều hơn nữa.
 - Công cụ Tiền xử lý: Hỗ trợ đầy đủ các công cụ tiền xử lý văn bản như Tokenization, Stemming, Lemmatization, POS Tagging, và Named Entity Recognition (NER).

- Tập dữ liệu phong phú: NLTK đi kèm với nhiều tập dữ liệu mẫu và các corpus đã được chuẩn bị sẵn, giúp việc thử nghiệm và nghiên cứu trở nên dễ dàng hơn.
- Phân tích ngôn ngữ: Cung cấp các công cụ phân tích ngôn ngữ tự nhiên như phân tích cú pháp, cú pháp dependency, và phân tích ngữ nghĩa.
- Tốc độ: NLTK có thể không nhanh như các thư viện học sâu vì nó chủ yếu được thiết kế cho các tác vụ tiền xử lý và phân tích ngôn ngữ cơ bản.
- Tiết kiệm tài nguyên: Ít đòi hỏi về tài nguyên tính toán so với các mô hình học sâu.
- Thân thiện với người dùng: Giao diện đơn giản, dễ sử dụng với tài liệu phong phú.
- Học tập nhanh: Phù hợp cho người mới bắt đầu với các tác vụ NLP cơ bản.

+ Pytorch:

- Mục đích: PyTorch là một thư viện học sâu, chủ yếu được sử dụng để xây dựng và huấn luyện các mô hình học sâu phức tạp.
- Khả năng: Mạnh mẽ trong việc xây dựng và huấn luyện các mô hình neural networks, đặc biệt là các mô hình học sâu như RNN, LSTM, Transformer, BERT, GPT, v.v.
- Neural Networks: Mạnh mẽ trong việc xây dựng và huấn luyện các mô hình học sâu phức tạp để xử lý ngôn ngữ tự nhiên như LSTM, GRU, Transformer.
- Thư viện hỗ trợ: Kết hợp tốt với các thư viện NLP khác như spaCy và HuggingFace's Transformers để xử lý ngôn ngữ tự nhiên.
- Khả năng tùy biến: Linh hoạt trong việc tùy biến và tinh chỉnh các mô hình để phù hợp với các bài toán NLP phức tạp.
- Hiệu suất cao: Hiệu suất tốt với các mô hình học sâu nhờ khả năng tối ưu hóa trên GPU.

- Yêu cầu tài nguyên: Đòi hỏi tài nguyên tính toán lớn hơn, đặc biệt khi làm việc với các mô hình lớn và dữ liệu lớn.
- Linh hoạt và mạnh mẽ: Cần kiến thức sâu hơn về lập trình và học sâu để sử dụng hiệu quả.
- Khả năng tùy biến cao: Cho phép tùy chỉnh các thành phần của mô hình một cách dễ dàng, phù hợp cho các nhà nghiên cứu và chuyên gia.

Cả NLTK và PyTorch đều có những ưu điểm và ứng dụng riêng biệt trong xử lý ngôn ngữ tự nhiên. NLTK phù hợp cho các tác vụ tiền xử lý và phân tích ngôn ngữ cơ bản, trong khi PyTorch mạnh mẽ và linh hoạt, phù hợp cho các mô hình học sâu phức tạp.

- Để có thể phân loại cảm xúc từ văn bản một cách tối ưu chúng ta cần chú ý đến một số yếu tố sau khi xây dựng mô hình học máy:
 - + Phân tích nội dung: Đọc và hiểu kỹ nội dung văn bản để nhận diện các từ, cụm từ hoặc cảm xúc được thể hiện.
 - + Lựa chọn mô hình học máy và các thư viện phù hợp cho từng mục đích để có thể được sử dụng để phân loại cảm xúc từ văn bản một cách tự động và chính xác.
 - + Xây dựng và sử dụng các bộ từ điển cảm xúc để phân loại từng từ trong văn bản theo các cảm xúc như vui, buồn, tức giận, sợ hãi, v.v.

KẾT LUẬN

Đồ án này đã cải thiện được một số vấn đề như sau:

- Cải thiện cân bằng dữ liệu: Phát triển các phương pháp thu thập và cân bằng dữ liệu để đảm bảo mô hình không thiên vị.
- Hiểu ngữ cảnh và biểu cảm: Phát triển các mô hình có khả năng hiểu ngữ cảnh và biểu cảm mạnh mẽ hơn.

Vấn đề chưa thể cải thiện:

- Tính toàn diện và đa ngôn ngữ: Chưa phát triển để mô hình có thể áp dụng trên nhiều ngôn ngữ và văn hóa khác nhau mà chỉ có thể áp dụng cho tiếng Anh.

TÀI LIỆU THAM KHẢO

- [1]: Giáo trình “Tổng quan trí tuệ nhân tạo”. Tác giả: Nguyễn Phương Nga, Trần Hùng Cường. Trường Đại Học Công nghiệp Hà Nội: Nhà xuất bản thống kê, 2021.
- [2]: Giáo trình “Quản lý dự án công nghệ thông tin”. Tác giả: Trần Tiến Dũng, Nguyễn Đức Lưu. Trường Đại học Công nghiệp Hà Nội: Nhà Xuất Bản Thống Kê, 2022.
- [3]: Giáo trình “Khai phá dữ liệu”. Tác giả: Trần Hùng Cường, Trần Thanh Hùng. Trường Đại Học Công nghiệp Hà Nội: Nhà xuất bản thống kê, 2017.
- [4]: Công trình nghiên cứu “Sentiment Analysis of Movie Reviews” của tác giả Charlie Chengrui Zheng: <https://github.com/udacity/deep-learning-v2-pytorch/blob/master/README.md> (Lần truy cập cuối 20/12/2024)
- [5]: Công trình nghiên cứu “Deep Learning for Sentiment Analysis” của tác giả Bert Carremans: <https://www.kaggle.com/code/bertcarremans/deep-learning-for-sentiment-analysis/notebook> (Lần truy cập cuối 20/12/2024)